POLITECNICO DI MILANO
DIPARTIMENTO DI ELETTRONICA, INFORMAZIONE E BIOINGEGNERIA
DOCTORAL PROGRAMME IN INFORMATION TECHNOLOGY

COOPERATIVE MACHINE LEARNING METHODS IN DISTRIBUTED SYSTEMS

Doctoral dissertation of:
Bernardo Camajori Tedeschini

Supervisor:
Prof. Monica Nicoli

Chair of the doctoral programme:
Prof. Luigi Piroddi

Cycle XXXVII

Acknowledgements

In the journey of problem-solving, constraints often delineate the boundaries within which solutions must be sought. However, true innovation emerges when we transcend these boundaries, seeking not just good-enough solutions, but also developing new techniques and methods to expand these boundaries. This iterative process, known to me as trial-and-error, is a relentless pursuit of improvement.
Occasionally, it is necessary to consciously embrace mistakes. This approach, while counterintuitive, challenges the usual tendency to follow feedback, comments, or suggestions without question. Instead, in my experience, the most profound insights often come from resisting immediate guidance or delaying reliance on feedback, allowing independent exploration to lead the way. Embracing naïve optimism, one should tackle problems fueled by curiosity, unconstrained by the fear of consequences or the effort required. Persistence is paramount; each failure is a stepping stone, and the accumulation of failures, not successes, fuels my dedication and passion.

... and it turned out to be a better world

I would like to first thank Prof. Monica Nicoli, for guiding me throughout the entire Ph.D. period and, most of all, for granting me the freedom to explore various research directions and supporting every decision I made. I know that this is a more than rare privilege, and I will always be grateful to her.

I would like to thank Prof. Moe Z. Win for hosting me in his lab at the Massachusetts Institute of Technology (MIT) and for teaching me the rigorous approach to research. We will always share the passion for the greatest country in the world, Italy. I also thank all the researchers and students at the lab for the inspiring conversations and the different perspectives from which to tackle a problem.

I would like to thank Dr. Serenella Sferza for giving me the opportunity to study at MIT. Thank you for the wonderful dinners and for having the noble objective of enabling once-in-a-lifetime experiences. Young scientists are and always will be the enablers of our future.

I would like to thank my greatest co-authors, Dr. Mattia Brambilla and Dr. Stefano Savazzi, who carefully revised my works and taught me invaluable suggestions for academic research and writing.

I would like to thank all the Researchers and Professors of the department and those I have encountered over the years. The University is made by people, and without their dedication and passion, it would lack soul and purpose.

I would like to thank all my dear friends in lexicographical order: Davide, Federica, Francesco, Gabriella, Giuliano, Laura, Mario, Paolo, and Stefano, who have been a

constant source of joy throughout this journey. A special thanks goes to Filippo, whose brotherly support and companionship have been invaluable along this path.

Lastly, I want to extend my deepest gratitude to my parents. This work is as much yours as it is mine. Thank you.

Cambridge (MA), June 2024

All'Italia

Bernardo Camajori Tedeschini

Financial Support

The research described in this thesis was supported, in part, by the Roberto Rocca Doctoral Fellowship awarded by Politecnico di Milano and Massachusetts Institute of Technology (MIT), by the project Centro Nazionale per la Mobilità Sostenibile (MOST), funded by the Italian Ministry of University and Research under the Piano Nazionale Ripresa Resilienza (PNRR) funding program, by the Ph.D. Grant from the Ministry of the Italian Government Ministero dell'Università e della Ricerca (MUR), by the European Space Agency (ESA) Navigation Innovation and Support Program (NAVISP) Element 2 pillar, by the Horizon EU project TRUSTroke in the call HORIZON-HLTH-2022-STAYHLTH-01-two-stage under GA No. 101080564.

Abstract

Cooperative learning and inference in multi-agent systems (MAS) are increasingly pivotal in addressing the complexities and dynamic demands of modern technological environments. Spanning domains from robotics and internet of things (IoT) to telecommunications and healthcare, these collaborative strategies enhance the robustness and adaptability of systems handling intricate or resource-intensive tasks. This thesis explores innovative approaches in both centralized and decentralized frameworks, focusing on optimizing system performance through advanced machine learning (ML) methods. The goal is to introduce novel methods that expand the capabilities of MAS in practical scenarios, ensuring efficient, scalable, and privacy-aware solutions that adapt dynamically to changing conditions and maintain high performance among varied and unpredictable environmental factors.

The thesis is divided into two main parts, each dedicated to analyzing one of the two key components of a MAS: learning and inference. In the first part of the thesis, the focus is on cooperative learning which is investigated in graph-aware centralized machine learning (C-ML), privacy-preserving decentralized machine learning (D-ML), and non-stationary learning frameworks. For graph-aware learning, we considered the tasks of data association (DA) and cooperative positioning (CP) in vehicular networks, where exploiting a logical graph structure enables the handling of non-linear distributions and scalable architectures. When data exchanged between agents is private or sensitive, D-ML algorithms can be used to exchange only model parameters or latent features, reducing the disclosure of privacy information. In this context, we proposed a real platform for performing decentralized and fully-decentralized learning in medical and IoT networks. In particular, we studied federated learning (FL) algorithms in asynchronous learning processes and FL weighted averaged consensus (WAC) techniques for serverless learning in non-independent and identically distributed (IID) conditions with heterogeneous devices. In the presence of resource-constrained devices, we proposed decentralized split learning (SL) algorithms that iteratively distribute the computational burden of training among agents. Finally, whenever agents are in the presence of highly-dynamic environments and non-stationary distributions of data, multi-agent reinforcement learning (MARL) algorithms can be adopted. In the thesis, we developed a novel MARL algorithm for performing implicit cooperative positioning (ICP) in vehicular networks, where passive objects (or targets) are exploited to refine the agents' state estimate.

After having investigated techniques for cooperative learning, in the second part of the thesis, we turned our attention to cooperative inference, where we studied efficient and reliable techniques for the tasks of non-line-of-sight (NLoS) identification, static

and mobile position in next-generation cellular networks. The agents, i.e., the base stations (BSs) in this case, estimate and compress the channel into a latent representation which is subsequently adopted for sensing. For NLoS identification, we proposed an anomaly detection scheme which efficiently evaluates the likelihood of channel samples of belonging to the line-of-sight (LoS) normal distribution. On the contrary, for static positioning, we presented a cooperative inference scheme for efficiently combining latent features. In particular, in LoS conditions, the BSs cooperatively localize the user equipment (UE) by fusing latent features, whereas in NLoS conditions, the BSs perform independently positioning. Finally, for mobile positioning, we first proposed a novel Bayesian neural network (BNN) algorithm for estimating the full uncertainty of predictions in real-time. Then, we integrated this uncertainty into tracking filters, optimally combining the fingerprint-based likelihood functions of different BSs in out of distribution (OoD) areas.

In conclusion, this thesis presents a comprehensive exploration of cooperative learning and inference strategies within MASs, offering scalable and adaptive solutions across a diverse range of technological domains. By integrating advanced ML techniques with the implicit complexities of cooperative environments, we have developed robust models that significantly enhance both the precision and reliability of various application areas, ranging from vehicular and IoT networking to healthcare and cellular systems. These innovative approaches not only demonstrate the practical benefits of cooperative strategies but also highlight the potential for future advancements in data-driven technologies.

Contents

List of Figures

List of Algorithms

List of Acronyms

3GPP 3rd Generation Partnership Project
5G fifth generation
6G sixth generation
A2A agent-to-agent
A2T agent-to-target
AC averaged consensus
ADCPM angle-delay channel power matrix
ADCRM angle-delay channel response matrix
AE autoencoder
AI artificial intelligence
AoA angle of arrival
AoD angle of departure
BBK Bayesian bright knowledge
BBP Bayes by backpropagation
BDK Bayesian dark knowledge
BNN Bayesian neural network
BP belief propagation
BS base station
C-ITS cooperative intelligent transportation systems
C-ML centralized machine learning
C-V2X cellular vehicle-to-everything
CAV connected automated vehicle
CFAdp consensus-driven FedAdp
CFA consensus-driven FedAvg
CFR channel frequency response
CIR channel impulse response
CNN convolutional neural network
CP cooperative positioning

CSI channel state information
D-ML decentralized machine learning
DAGMM deep autoencoding Gaussian mixture model
DAKDM deep autoencoding kernel density model
DA data association
DFT discrete Fourier transform
DL deep learning
DNN deep neural network
DS deep sets
Dec-POMDP decentralized-partially observable Markov decision process
EKF extended Kalman filter
FD-ML fully decentralized machine learning
FG factor graph
FHIR Fast Healthcare Interoperability Resources
FL federated learning
FedAdp federated adaptive weighting
FedAvg federated averaging
FedProx federated proximal
GDPR General Data Protection Regulation
GMM Gaussian mixture model
GNN graph neural network
GNSS global navigation satellite system
GP Gaussian process
HL7 Health Level Seven International
ICP implicit cooperative positioning
IID independent and identically distributed
IRB independent recurrent blocks
ISD inter-site distance
IoT internet of things
KDE kernel density estimation
KF Kalman filter
KL Kullback-Leibler
LIDAR light detection and ranging
LSTM long short-term memory
LoS line-of-sight
MAE mean absolute error
MAPPO multi-agent proximal policy optimization

MARL multi-agent reinforcement learning
MAS multi-agent systems
MCMC Markov chain Monte Carlo
MC Monte Carlo
MDP Markov decision process
MIMO multiple-input multiple-output
MLE maximum likelihood estimation
ML machine learning
MMSE minimum mean square error
MOT multiple object tracking
MPA message passing algorithm
MPNN message passing neural network
MQTT message queuing telemetry transport
NLoS non-line-of-sight
NLNN non-local neural networks
NN neural network
OFDM orthogonal frequency-division multiplexing
OoD out of distribution
PDF probability density function
PEB position error bound
PS parameter server
RADAR radio detection and ranging
RBF radial basis function
RF random forest
RL reinforcement learning
RN relational networks
RNN recurrent neural network
RSS received signal strength
SCFL split consensus federated learning
SFCRM space-frequency channel response matrix
SFL split federated learning
SGD stochastic gradient descent
SGLD stochastic gradient Langevin dynamics
SL split learning
SPADA sum-product algorithm for data association
SPA sum-product algorithm
SUMO Simulation of Urban MObility

SVM support vector machine
TCN temporal convolutional network
ToF time of flight
UAV unmanned aerial vehicle
UE user equipment
UMi urban micro
UPA uniform planar array
URLLC ultra-reliable low-latency communications
V2X vehicle-to-everything
VAE variational autoencoder
VI variational inference
VRU vulnerable road user
WAC weighted averaged consensus
mMIMO massive multiple-input multiple-output
mmWave millimeter waves
t-SNE t-distributed stochastic neighbor embedding

Notation

Random variables are displayed in sans serif, upright type; their actual values in serif, italic type. Matrices and vectors are represented by bold uppercase and lowercase letters, respectively. For instance, a random variable and its realization are indicated by y and y; a random vector and its realization by $\mathbf{y}$ and $\boldsymbol{y}$; a random matrix and its realization by $\mathbf{Y}$ and $\boldsymbol{Y}$, respectively. Random sets and their realizations are represented by upright sans serif and calligraphic fonts, respectively. For example, a random set and its realization are represented by Y and $\mathcal{Y}$, respectively. Random sets and their realizations are denoted by up-right sans serif and calligraphic font, respectively. For example, a random set and its realization are denoted by Y and $\mathcal{Y}$, respectively. The function $p_{\mathsf{y}}(y)$, and simply $p(y)$ when there is no ambiguity, denotes the probability density function (PDF) of y. The likelihood function, parameterized by the parameter θ, is denoted as $p_{\mathsf{y}|\boldsymbol{\theta}}(y|\theta)$ or $p_{\theta}(y)$. The notation $\mathbf{Y}^{\mathrm{H}}$, $\mathbf{Y}^{*}$, and $\mathbf{Y}^{\top}$ indicate the matrix conjugate transposition, conjugation, and transposition. $\mathrm{Tr}(\cdot)$ and $\det(\cdot)$ denote the trace and the determinant of the matrix argument, respectively. The Kronecker and the Hadamard products between two matrices are denoted with the symbols $\otimes$ and $\odot$, respectively. The inner product between two vectors $\boldsymbol{y}$ and $\boldsymbol{x}$ is represented as $\langle \boldsymbol{y}, \boldsymbol{x} \rangle$. The Cartesian product between two sets $\mathcal{Y}$ and $\mathcal{X}$ is denoted as $\mathcal{Z} = \mathcal{Y} \times \mathcal{X}$. With the notation $\mathsf{y} \sim \mathcal{N}(\mu, \sigma^2)$ we indicate a Gaussian random variable y with mean μ and standard deviation σ, whose PDF is denoted by $\mathcal{N}(x; \mu, \sigma^2)$. With the notation $\mathsf{y} \sim \mathcal{U}(a, b)$ we indicate a Uniform random variable y with support $[a, b]$. We use $\mathbb{V}\{\mathsf{y}\} = \mathbb{V}_{\mathsf{y} \sim p(y)}\left\{\mathsf{y}\right\}$ and $\mathbb{E}\{\mathsf{y}\} = \mathbb{E}_{\mathsf{y} \sim p(y)}\left\{\mathsf{y}\right\}$ to denote the variance and the expectation of random variable, respectively. $\mathbb{C}$ and $\mathbb{R}$ stand for the set of complex and real numbers, respectively. $\mathrm{Im}(y)$ and $\mathrm{Re}(y)$ are the complex and real part of the complex number y, respectively. $j = \sqrt{-1}$ denotes the imaginary unit. $\lfloor x \rfloor$ indicates the largest integer not greater than y, $|\boldsymbol{y}|$ and $|\mathcal{Y}|$ denote the length of the vector $\boldsymbol{y}$ and size of the set $\mathcal{Y}$, respectively, while $\delta[\cdot]$ and $\delta(\cdot)$ are the Kronecker and Dirac delta functions, respectively. Finally, $\|\boldsymbol{y}\|_2$ and $\|\boldsymbol{y}\|_1$ represent the L2 and L1 norms of vector $\boldsymbol{y}$, respectively.

Part I

Overview

CHAPTER 1

Introduction

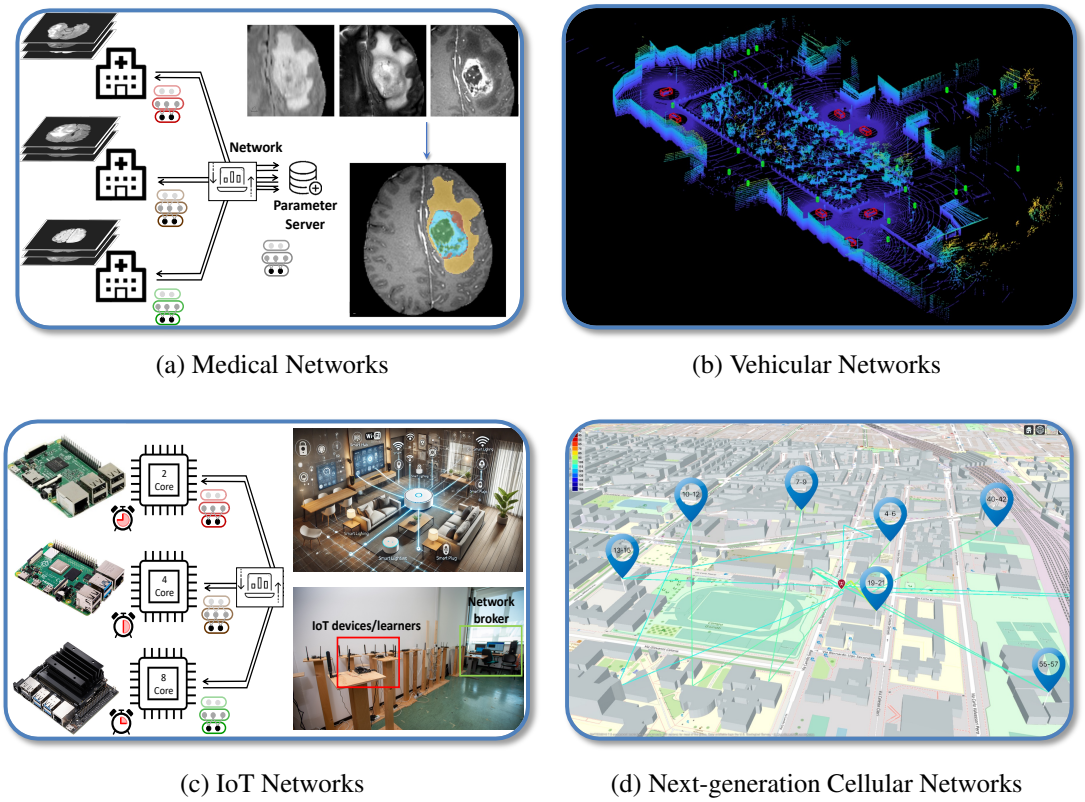
This chapter presents the thesis work and its structure. Sec. 1.1 introduces the main motivations behind the thesis research on cooperative machine learning (ML) methods for distributed systems. Sec. 1.2 highlights the current state-of-the-art on cooperative learning and inference, with main applications on vehicular, medical, internet of things (IoT) and next-generation cellular networks. Finally, Sec. 1.3 reports the main contributions and achievements of the thesis, whereas Sec. 1.4 describes the structure of the thesis and the publications written during the Ph.D. research.

1.1 Motivations

COOPERATION between agents, particularly in dynamic and complex environments, is fundamental to achieving effective and efficient task execution [1]. The necessity of such collaboration is pronounced in scenarios where tasks are too complicated or resource-intensive for individual agents to handle alone [2]. Multi-agent systems (MAS) span various domains from medical fields [3] and IoT [4] to cooperative intelligent transportation systems (C-ITS) [5] and next-generation cellular networks [6] (see Fig. 1.1). According to the specific application domain, the focus can shift from learning the most accurate and/or reliable model to efficiently predicting the model outcome. For example, in a medical network, the focus is on learning the most accurate and reliable models for tasks like diagnosis, ensuring data privacy, and compliance with regulations. Whereas in vehicular networks, the emphasis is on efficiently predicting the model outcome in real-time for applications such as cooperative positioning (CP). Thus, we can distinguish between cooperative learning and inference based on their distinct roles and methodologies, each tailored to optimize different aspects of MAS.

Regarding cooperative learning, two distinct methodologies emerge: centralized machine learning (C-ML) and decentralized machine learning (D-ML) approaches. C-ML involves a central node, usually known as parameter server (PS), that processes data collected from various agents to build a comprehensive global model. C-ML is often applied in contexts where performances are favored with respect to scalability and where the exchange of data within the network does not imply privacy issues. Moreover, it is also suitable when the nodes are not mobile devices, as continuous and intensive data exchange could be too demanding in such cases. According to the specific task, we may model the problem as a learning process over a graph. Indeed, learning directly on graphs is a particularly effective mechanism to exploit the knowledge of the graph structure to push the limits of the performances [7]. Vehicular networks represent an area where cooperative learning is pivotal [8]. Here, vehicles communicate and learn from each other to improve navigation, safety, and traffic management. By sharing information such as motion dynamics, detections, and road conditions, vehicles create a dynamic, adaptive network that enhances travel efficiency and safety for all vulnerable road user (VRU). This C-ML approach is particularly effective in vehicular networks because of the roadside infrastructure that can act as an aggregator and perform edge processing, enabling real-time data collection and model updates with low latency. The main challenges include performing accurate data association (DA) between inter-agent, i.e., vehicle, measurements [9], aided for subsequent sensing applications such as CP and multiple object tracking (MOT) [10].

On the other hand, D-ML, often applied in medical networks, employs individual devices to learn and make decisions independently, thereby improving scalability while preserving the privacy and security of locally stored data [11]. Indeed, in the medical context, the new regulations on data protection, such as the General Data Protection Regulation (GDPR) [12], impose strict requirements on how data is processed and shared, compelling the adoption of decentralized models that ensure data privacy and security. D-ML are also adopted in IoT networks [13], where additional requirements include energy efficiency [14], fast convergence [15], and attack resilience [16]. Whenever heterogeneous agents with different temporal alignments or computation capabilities are



(a) Medical Networks

(b) Vehicular Networks

(c) IoT Networks

(d) Next-generation Cellular Networks

Figure 1.1: *Application domain of MAS.*

present, asynchronous orchestration of the learning process must be taken into account [17]. Moreover, when dealing with non-independent and identically distributed (IID) data distribution among agents, convergence may be difficult to achieve, especially with skewed data in the feature or sample domain. In order to handle resilience, e.g., node failures, and even more scalability, fully decentralized machine learning (FD-ML) methods need to be employed, where agents collaboratively train models while also reaching a consensus without a central coordinating entity [18]. Finally, the last challenges include non-stationarity of both the learning graph and data distributions [19, 20]. Here, the models must be robust and adaptive, capable of evolving as the underlying data and network dynamics change.

Turning to cooperative inference, the focus becomes the application of these cooperatively learned models to make predictions or decisions. Here the objective is not only to exploit cooperation (e.g., data exchange in C-ML or parameters/latent features in D-ML settings), but also to improve efficiency and reliability, especially in mission-critical applications. One example is in advanced sensing applications within next-generation cellular networks, e.g., fifth generation (5G) and sixth generation (6G) [21, 22], where strict requirements on positioning accuracy, latency and reliability represent huge challenges [23, 24]. In cellular vehicle-to-everything (C-V2X) enhanced services, such as cooperative adaptive cruise control and cooperative emergency maneuvers of connected automated vehicles (CAVs), the requirements on the positioning accuracy and latency can be as low as 20 cm and 10 ms, respectively [25, 26]. Regarding reliability,

stringent guidelines encompass not only packet drop probability, e.g., up to 99.999% for the ultra-reliable low-latency communications (URLLC) use-cases [27], but also trustworthiness of measurements and positioning predictions [28]. It is clear that, despite the technological enablers such as high carrier frequencies, bandwidths and massive multiple-input multiple-output (mMIMO) systems [29], conventional non-ML-based models fail in fulfilling the above requirements, especially against high blockage situations and complex real world noise distributions [30–32]. Indeed, 3rd Generation Partnership Project (3GPP) specifications already foresee ML/artificial intelligence (AI) for assisted or even direct positioning in the latest 5G-Advanced Release 18 [33].

Examples of relevant tasks that cooperative inference enables in cellular systems for sensing applications are cooperative non-line-of-sight (NLoS) identification, positioning, and tracking. The first step for assisting positioning is understanding whether the radio signals come from line-of-sight (LoS) or NLoS conditions, to mainly determine the un/reliability of those observations. Here the main challenge is to create a precise and compressed LoS/NLoS distribution representation to efficiently perform sampling and establish the likelihood of being in one of the two conditions. On the other hand, CP focuses on peak performances on the estimated position by optimally combining the output of other agents, i.e., base stations (BSs) in this case. The difficulties comprise the choice of the information type to be exchanged between agents and how effectively combining those information. Finally, in cooperative tracking, the objectives are achieving low latency, in order to track a fast moving object, and producing a reliability measure of the models' output. Indeed, frequentist-based ML algorithms do not provide a reliable uncertainty quantification of the prediction and tend to overfit in low-density training regions. Here, the main challenge is to discern between the uncertainty due to the intrinsic noise in the data, i.e., aleatoric uncertainty, and the uncertainty due to the variability on the estimated model parameters, also known as epistemic uncertainty [34]. Bayesian neural networks (BNNs) offer a partial solution to these issues [35], whereas a real-time, uncertainty-aware, and efficient framework is still an open research direction.

Driven by the increased demand for real-world cooperative applications, this thesis aims at proposing novel solutions to the main open problems in cooperative learning and inference at enhancing robustness and adaptability under dynamic conditions. This involves the development of new methodologies that allow cooperative systems to maintain high performance even when facing unpredictable environmental changes or when operating with incomplete or imperfect information. A representative image of these two cooperative steps can be found in Fig. 1.2 and 1.3. In the former, agents exploit cooperation to build a more accurate model. Each cluster of agents may have different characteristics, e.g., computational capabilities, graph awareness, and data types. On the other hand, in the latter, agents (BSs or CAVs in this case) adapt their predictions according to the world interactions and neighbors' presence. In particular, agents predict efficient real-time function representation, e.g., model outputs, which are exchanged and fused into the network to enhance the coherence and accuracy of collective predictions.

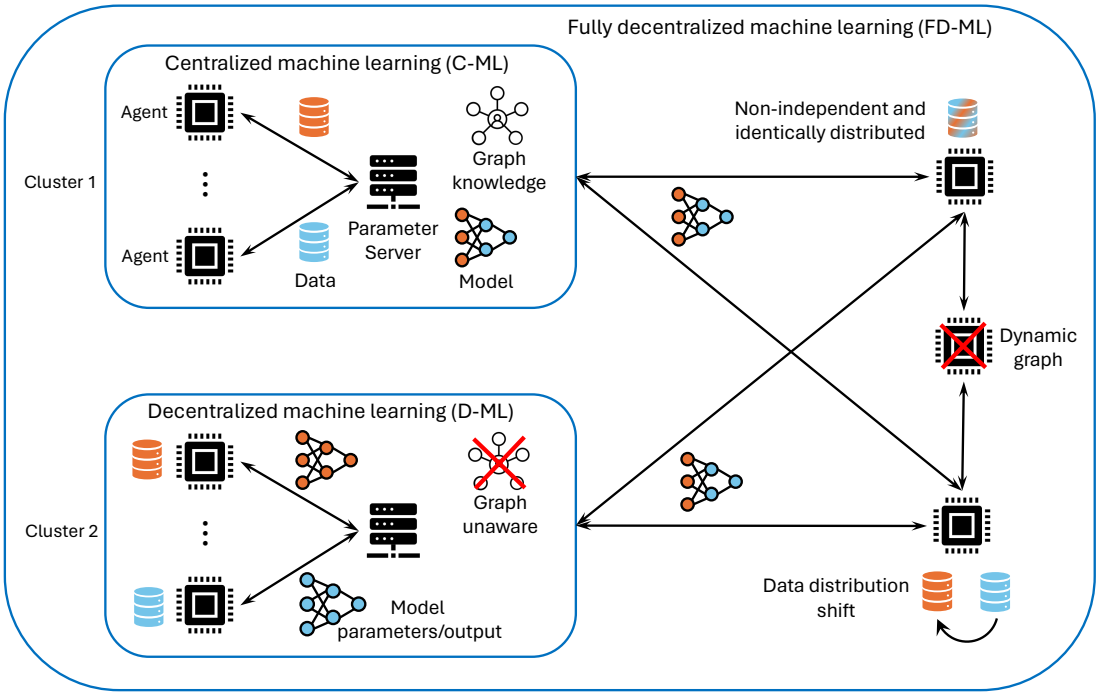


Figure 1.2: *Cooperative learning in MAS.*

1.2 State of the Art

In this chapter, we provide a comprehensive overview of current advancements in cooperative learning and inference methods. We will mainly focus on specific applications to give tangible examples of where these algorithms can be applied in real-world scenarios.

1.2.1 Graph-aware Centralized Learning

Exploiting the graph knowledge to perform a specific task is a common technique in standard signal processing methods. An example is the sum-product algorithm (SPA) (or message passing algorithm (MPA)), which is based on factor graphs (FGs), a graphical-based intuitive way to visually simplify the factorization of probability density function (PDF), where nodes represent variables and factors, and bidirectional links embody the dependencies [36]. MPAs leverage these graphs for iterative message passing, enhancing cooperative tasks across diverse applications such as CAVs. Historically, MPAs have been instrumental in enabling sensor data fusion across vehicular networks in C-ITS. By integrating data from various sensors distributed across vehicles and infrastructural elements, MPAs facilitate advanced CP and MOT use-cases [37, 38]. These applications benefit significantly from the distributed or single-unit sensor data [39, 40], aggregated via vehicle-to-everything (V2X) communication links and correctly associated by means of DA algorithms [41, 42]. However, despite their scalability, SPA are optimal only for linear and Gaussian models and become approximations in scenarios involving non-linear distributions or graph loops [42, 43], mostly presented in real-world systems.

In contrast, ML models, particularly deep neural networks (DNNs) embedded within

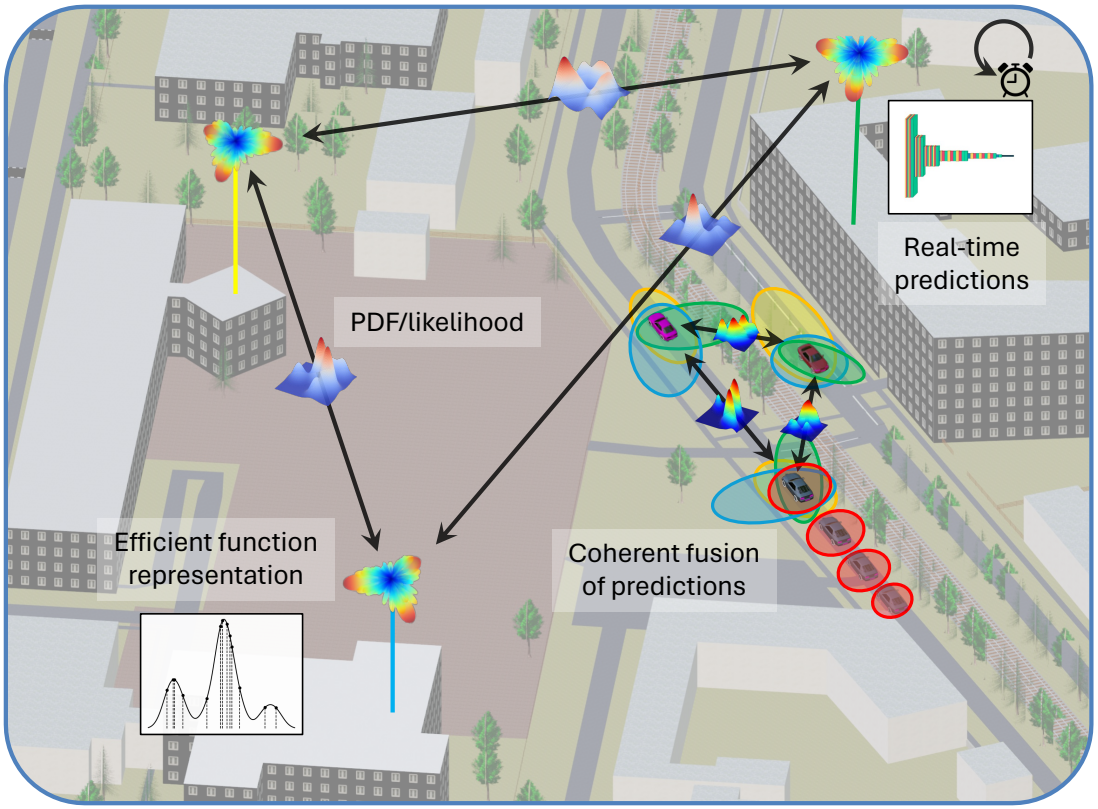


Figure 1.3: *Cooperative inference in MAS.*

graphs as graph neural networks (GNNs) [44], offer a robust alternative by directly learning from data generated at each vehicle. GNNs, and especially message passing neural networks (MPNNs), inherit SPA's message-passing framework but enhance it by addressing errors due to cycles and mismatches in model assumptions. Indeed, the integration of MPNNs and SPA have been used in the past for correcting cycle errors and mismatches in the model [45, 46], offering improved performances in the overall system. Although MPNNs maintain a scalable architecture with fewer parameters than typical DNNs [47], they effectively capture both linear and non-linear relationships, providing a powerful tool for processing loopy graphs when sufficient training data is available. This integration ensures that MPNNs can outperform traditional MPAs in managing non-linear distributions and complex network topologies frequently encountered in vehicular and infrastructural networks [48].

1.2.2 Privacy-preserving Decentralized Learning

Whenever privacy requirements have to be fulfilled on the data, D-ML algorithms may be adopted to ensure secure and confidential handling of the data. These algorithms exchange the model parameters and/or intermediate layers' outputs that should not disclose the private information within the nodes.

First, D-ML methods were developed within the federated learning (FL) framework. In vanilla federated averaging (FedAvg) algorithm [49], a central entity, i.e., PS, coordinates the learning process among the participating agents, or clients, by aggregating their

locally computed model updates. While FL offers undoubted advantages, it also faces multiple challenges that need to be tackled to guarantee robustness and efficiency. One of the key difficulties is handling heterogeneous agents with different temporal alignments and possible disconnections during local training. Indeed, vanilla FL algorithms are synchronous, meaning that the PS has to pause for the selected agents to terminate their local training to update the global model. To address this problem, asynchronous FL algorithms have been investigated in the literature, where the global model is updated independently of the agents' local updates completion. However, current state-of-the-art asynchronous FL methods exhibit several key limitations. Firstly, in standard strategies, the PS updates the global model immediately upon receiving a local model [50], disregarding agent-specific resource constraints. This can result in biased updates from faster agents. Secondly, since the number of local epochs is typically pre-specified, the agents' learning process is not optimized or adjusted based on the quality and quantity of data or the typology of traffic [17].

Other major limitations of conventional FL are the handling of non-IID data among agents and the reliance on the PS for model aggregation. For handling non-IID data, federated proximal (FedProx) algorithm [51], and its variants [15], adopt an inexact proximal point update for local optimization by penalizing the divergence of the local parameters from the PS global ones. While FedProx is implemented independently by each client, other algorithms have aimed at developing specialized PS aggregation weighting to manage the influence of each local model in the global update. For example, the federated adaptive weighting (FedAdp) algorithm proposed in [52] employs PS aggregation weights based on the inner product between the global and local gradients. This metric can act as a dis/similarity measure to gauge the contribution of the local model. Intuitively, the more orthogonal the local and global gradients, the less the local model will positively affect the global aggregation.

To tackle the challenges linked with PS-based FL and other centralized approaches, a distributed version of FL, known as consensus-based FL or consensus-driven FedAvg (CFA) [18, 53], has been recently introduced. Consensus-FL enables agents to collaboratively train models and reach consensus on model updates without a central coordinating entity, resulting in more efficient and scalable learning. These methods have evolved from traditional distributed maximum likelihood estimation based on consensus [54], where individual nodes exclusively depend on their own local data and the data exchanged with adjacent nodes to refine their local estimates. In its simplest form, consensus, such as averaged consensus (AC) [55], involves synchronized model parameter updates with constant or no weighting aggregation. Similar to AC, consensus-FL encounters challenges due to non-IID datasets and related convergence [56, 57]. While both non-IIDness and decentralization have been intensively investigated, the literature still lacks of approaches that address both the issues.

An alternative to FL that still meets privacy demands is split learning (SL) [58], a method within D-ML tailored for resource-limited settings such as IoT environments. In SL, the model training is distributed between the agents and a PS, where each party has access only to certain parts of the model [59]. This partitioning improves the privacy of both the model and the data, and enhances training speed and communication efficiency relative to FL [60]. The neural network (NN) undergoing training is segmented into two sub-networks at a particular layer, namely the split or cut layer. The upper layers

are managed by the agents, while the lower layers are controlled by the PS. During the training phase, agents perform forward propagation and transmit the intermediate outputs, known as smashed data, to the PS. The PS calculates the final output and performs back-propagation, sending the gradients of the split layer back to the agents. Despite the advantages, SL presents two main drawbacks, related to the lack of parallelization and to the presence of the PS.

Specifically, in SL, agent interaction with the PS occurs sequentially, which keeps other agents' resources idle during the training sequence. This sequential process increases both the training overhead and latency, particularly when the learning involves many devices. To address this, researchers have suggested varying the training sequence and modifying the data sizes within the nodes [61]. A significant advancement in achieving full parallelism has been realized with the introduction of the split federated learning (SFL) framework [62, 63]. SFL fuses the principal advantage of FL-parallel processing across decentralized agents with the key benefit of SL, which involves dividing the network into server-side and client-side components throughout training. Unlike SL, SFL enables all agents to process data simultaneously, interacting concurrently with both a split PS and a federated PS. This allows for enhanced computational efficiency and reduced latency compared to traditional SL. On the contrary, the centralization issue related to the presence of the PS is still an open problem in the literature, with ongoing research efforts aimed at finding viable solutions.

1.2.3 Non-stationary Cooperative Learning

In conditions where labeled data are expensive or scarce, or when the environment is subject to dynamic changes, traditional ML techniques often fail to adapt and perform effectively. Under such conditions, the reinforcement learning (RL) approach [64] and its deep learning (DL)-enhanced variants [65–67] have demonstrated remarkable capabilities in managing challenging single-agent Markov decision process (MDP). Here, the agent's actions are guided on achieving long-term goals, or rewards, and have direct consequences on the world state [68, 69]. In scenarios involving multiple agents within environments where the state is not directly observable, we employ a decentralized-partially observable Markov decision process (Dec-POMDP) modelization [70–73] that is addressed with multi-agent reinforcement learning (MARL) [74] algorithms. MARL involves autonomous agents whose actions mutually affect their perception of the environment, and it is typically managed by employing recurrent neural networks (RNNs) that exploit histories of observations and actions. MARL and its DL-related methodologies are particularly well-suited for cooperative sensing tasks, such as CP and MOT, as they are effective in complex decision-making scenarios where autonomous agents collaborate towards a shared goal (i.e., reward) and base their decisions on incomplete or uncertain information about the system's state [75]. Effectively, MARL extends the concept of Bayesian filtering, where agents not only predict the state through belief calculations but also make strategic decisions aimed at maximizing long-term rewards, guided by a policy that transitions from state to action.

In the context of MARL applied to CP, most research has concentrated on the usage of intelligent unmanned aerial vehicles (UAVs) for target tracking [76] and on agent scheduling to enhance CP [77]. The study in [76] aimed at steering the agents to track passive objects, assuming the agents' state is known. However, they discarded the

primary challenge of simultaneously estimating the agents' states and sensing targets from the measurements. The research in [77] utilized conventional MPA for state estimation, with the goal of activating links between agents to boost cooperative positioning performance (i.e., enhancing the position error bound (PEB)). The limitations here include using RL merely to assist MPA rather than for direct positioning, and modeling links as isolated agents instead of fully utilizing the capabilities of MAS. In contrast, MARL strategies focused on communications typically overlook the aspect of state estimation [78] and prioritize not the efficiency of communication but rather the optimal integration of information received from neighboring agents [79]. Given this overview, the pressing unresolved issue is the development of a decentralized MARL algorithm that can concurrently handle both the computation of agent state beliefs and the management of agent-to-agent communication resources, aiming to optimize both location accuracy and communication efficiency.

1.2.4 Sensing in Cellular Networks by Cooperative Inference

A major field of application for cooperative inference is sensing in next-generation cellular networks, where NLoS identification is required for enhancing the reliability of positioning. By accurately determining whether a signal is obstructed (i.e., in a NLoS condition), the system can adjust its algorithms to compensate for signal degradation or latency issues. Subsequently, cooperation is exploited to augment single-BS positioning by coherently combining latent representations of observations among BSs. This cooperation allows for improved accuracy, robustness, and continuity in location-based services, even in dense urban environments where NLoS conditions frequently occur. Finally, cooperative tracking is crucial for mobile positioning, where the fusion of data from multiple sources (e.g., BSs, user equipment (UE), and CAVs) provides a more comprehensive and dynamic understanding of an object's trajectory, compensating for signal loss or errors encountered by individual nodes.

Therefore, in the following, we describe the current state-of-the-art of cooperative inference techniques for sensing applications in next-generation cellular networks. The learning is mainly performed in a centralized manner, i.e., C-ML, where data is shared among nodes to build a unique ML model. Extension to D-ML can be obtained by applying decentralized learning techniques such as FL or SL. Since it is not the primary emphasis of this chapter, we focus on exploring techniques that achieve optimal performance without compromising on efficiency and latency.

1.2.4.1 Machine Learning for NLoS Identification

The usage of ML for NLoS identification has gained significant attention due to its ability to bypass the need for complex statistical models, which are often required to determine the combined probability distributions of key features in traditional methods. ML-based approaches overcome these limitations by automatically learning the underlying patterns without statistical modeling of input characteristics. Initial studies on ML for NLoS identification adopted manually-selected channel state information (CSI) features, such as maximum amplitude, kurtosis, and energy [80–82], in combination with non-deep ML models, comprising random forests (RFs), Gaussian process (GP), and support vector machines (SVMs) [83, 84]. Despite being simple and efficient, these methods

heavily rely on predefined features, which may limit their performance. With the advent of 5G, comprising higher spatial and temporal resolutions, recent studies have leveraged the combination of the full raw channel impulse response (CIR) and the convolutional neural network (CNN)'s automatic feature extraction capabilities for highly non-linear feature extraction and compression [85, 86]. In particular, the compression is achieved by means of a autoencoder (AE) structure, which tries to reconstruct the input CIR data in a lower-dimensional form while preserving essential information for classification tasks. Subsequently, the compressed channel representation, also known as latent features, can be leveraged to effectively decrease the processing demands of prediction models. However, when dealing with supervised ML techniques, where both LoS/NLoS labels are provided for training, significant challenges still remain regarding the extensive data measurement and labor-intensive labeling required. Moreover, supervised methods need frequent updates to their training datasets to reflect changing conditions and adequately represent all potential NLoS anomalies.

To overcome these challenges, semi-supervised methods, such as anomaly detection techniques, may be employed. These methods often involve learning a single distribution, requiring fewer examples of anomalies compared to fully supervised learning, and having better performances with respect to unsupervised approaches [87]. Works in this direction may be found in [88], where authors exploited variational autoencoder (VAE) to induce the distribution of latent features towards a Gaussian PDF for easier representation and computation. Higher performances can also be achieved by adopting a two-step training AE-kernel density estimation (KDE), where the KDE is employed for obtaining the anomaly scoring from the latent features [89, 90]. This method, however, requires retaining the entire dataset for inference, which may be infeasible for storing requirements. Although the usage of VAEs and AE-KDE can yield excellent results, they still require a sampling-based mechanism for performing prediction, making them less suited for real-time applications. To overcome this, one study proposed a deep autoencoding Gaussian mixture model (DAGMM) that simultaneously learns the latent features and their densities within a Gaussian mixture model (GMM) framework [91]. While this method avoids the complexities of two-step training, GMMs are often prone to singularities and may not completely represent the latent distribution of normal samples. To conclude, in the literature there is a lack of real-time methodologies that can efficiently represent distributions of latent features and perform predictions without sampling or storing training datasets.

1.2.4.2 Machine Learning for Static Positioning

The vast majority of works on ML for static positioning are based on fingerprinting methodologies, since they permit to map complex high-blockage environments to their related position. Initial studies employed received signal strength (RSS) fingerprinting to perform positioning with Wi-Fi technology [92, 93]. Following the introduction of multiple-input multiple-output (MIMO)-orthogonal frequency-division multiplexing (OFDM) technologies in the IEEE 802.11a/n protocol, which facilitated the extraction of CSI from commercial Wi-Fi equipment, there has been a rise in studies focused on wireless positioning and target tracking through CSI. The availability of the channel information across multiple antennas and frequencies permits the learning of not just the user's position [94–96], but also the environmental dynamics influencing such

propagation [97]. DL methods have been applied to directly learn optimal nonlinear feature combinations to generate outputs for NLoS classification and position estimation. Relevant studies include [98, 99], where CNN was utilized for feature extraction.

The use of complete CIR data, particularly when formatted into image-like structures, has recently been recognized as a promising technique. Researchers in [100] used both geographic data and the CIR knowledge (i.e., phase, power gain, angle of departure (AoD), angle of arrival (AoA), and time of flight (ToF)) to estimate UE locations in 5G cellular systems. However, they presumed perfect knowledge of CIR via ray-tracing, which is challenging under real-world conditions. Another study [101] employed the channel frequency response (CFR) matrix, calculated through practical channel estimation and enhanced with additive noise during training. While successful, the CFR matrix does not explicitly represent the ToF or AoA for each path, potentially complicating feature extraction. A novel approach [102] involved a 3D CNN with multi-scale convolutional layers to infer the location directly using an angle-delay channel power matrix (ADCPM). Yet, this method depends heavily on fingerprint sampling distance and fails to differentiate between NLoS and LoS states, addressing each position uniformly without distinguishing among geometric features, that are advantageous in LoS scenarios, and those dependent solely on NLoS fingerprints. Moreover, while existing approaches focus on single-BS positioning based on aggregated data, there is a lack of literature on DL-based location prediction that incorporates measurements from different antenna panels at the inference step through BSs' cooperation.

1.2.4.3 Machine Learning for Mobile Positioning

The task of UE tracking via ML techniques within 5G environments is still poorly investigated, with most prior works opting for traditional Bayesian methods like extended Kalman filter (EKF) [103] or MPA [104], augmented by millimeter waves (mmWave) and MIMO technologies. The study in [105] explored the use of cutting-edge temporal convolutional network (TCN) models for NLoS outdoor tracking, achieving a mean absolute error (MAE) of 1.8 meters. In parallel, studies within indoor settings [106] applied long short-term memory (LSTM) and CNN to raw CSI data. However, both LSTMs and TCNs encounter two primary limitations. Firstly, they necessitate training datasets comprising highly accurate ground truth trajectories. Although feasible for static scenarios, acquiring such precise ground truth for dynamic positioning, particularly outdoors, proves challenging without sophisticated optical laser positioning systems. Secondly, traditional LSTMs and TCNs do not provide predictions with uncertainty metrics, which restricts their application in environments where safety is paramount.

BNNs can provide a solution to these issues, while maintaining the advantages of conventional NN in providing NLoS position estimates. Indeed, BNNs are able to differentiate and fully evaluate the uncertainties that characterize the predictions, i.e., the aleatoric and epistemic uncertainties [107]. Identifying these uncertainties is critical as it clarifies the motivations behind the model's prediction uncertainty, i.e., whether it is due to insufficient training data or intrinsic data variability. Furthermore, understanding these uncertainties can pinpoint where more training data would be most advantageous (i.e., areas with high epistemic uncertainty but low aleatoric uncertainty). However, in the literature of BNNs for positioning, BNNs have been merely used for providing static point estimates with uncertainty [108] or for enhancing mean performances

of flight trajectory predictions [109, 110], without actively exploiting their predicted uncertainty within the tracking solution. Moreover, these approaches require sampling procedures and are not suitable for real-time applications. To this aim, many works tried to tackle the problem of BNN real-time uncertainty prediction. A potential resolution to these challenges involves the implementation of *teacher-student* methodologies, like Bayesian dark knowledge (BDK) [111], where a non-Bayesian student NN is taught to emulate a Bayesian teacher BNN, thereby learning both point estimates and associated uncertainties. During deployment, the student NN manages real-time uncertainty assessments without the need for labor-intensive sampling techniques. Yet, a significant obstacle with teacher-student models is the student's inability to discern among the various types of uncertainties in its outputs. Currently, no real-time methods that effectively learn both aleatoric and epistemic uncertainties are available, especially for safety-critical tasks like autonomous driving.

1.3 Contributions and Objectives

The objective of the thesis is to design innovative solutions that enable efficient and reliable inference in networks of interconnected agents, where each agent must collaboratively learn a shared model from distributed data. These solutions aim to address key challenges in the current open problems of the state-of-the-art in the domain of cooperative MAS.

The cooperation is first exploited for building models that leverage the vast data produced at the edge, and then adopted in improving the efficiency, latency, and reliability of inference procedures. The open problems are investigated in key application areas to prove the concrete effectiveness of the developed solutions and algorithms. We studied cooperative learning methodologies mainly in vehicular networks (for DA and CP tasks), medical networks (for brain tumor segmentation tasks), and IoT networks (for resource-constraint tasks). On the contrary, cooperative inference has been optimized in next-generation cellular networks (for NLoS identification, static positioning, and mobile positioning tasks).

Specifically, the key contributions of this thesis are outlined as follows:

- In vehicular networks, we tackled the issues of MPA in DA and CP tasks by designing data-drived methods, i.e., MPNN, that exploit the knowledge of the network graph. In Papers [C1] and [J2], we introduced MPNN models for the cooperative association of 3D bounding boxes from lidar sensing in CAVs, evaluating their performance against the standard sum-product algorithm for data association (SPADA) [43]. Conversely, in Paper [J3], we present a combined architecture that integrates a MPNN with a LSTM. This model is designed to learn the motion patterns of agents over time and iteratively refine their position estimates using a message-passing procedure.

- The D-ML and FD-ML methods have been first analyzed in Paper [J4] by developing a custom FL system built on the message queuing telemetry transport (MQTT) protocol that permitted the real-world validation of algorithms for brain tumor segmentation in physical separated medical nodes. With the developed FL platform, in Paper [J5], we tackled the main issue of training synchronization procedures by proposing a PS-policy for the orchestration of agents' updates in heterogeneous IoT settings. We

then moved, in Paper [J6], to analyze fully-decentralized FL by proposing weighted averaged consensus (WAC) techniques, namely consensus-driven FedAdp (CFAdp), for the improving performances and convergence of consensus-schemes under non-IID data distributions. Finally, in Paper [J7], we proposed the decentralized version of SFL algorithms, namely split consensus federated learning (SCFL), which permits the distributed learning and inference in resource-constraint IoT networks where agents improve the performances by exchanging smashed data without the PS coordination.

- Under non-stationary vehicular networks, in Papers [C8] and [J9], we propose a new MARL algorithm, namely implicit cooperative positioning (ICP)-multi-agent proximal policy optimization (MAPPO), for performing CP by exploiting detected passive objects among agents as common reference positioning points. We adopted a novel *centralized-training and dynamic-decentralized-execution* scheme, which permits the agents to automatically learn the state world representation by means of belief estimation. At the same time, the agents learn an optimal policy to dynamically de/activate the radio links with the neighbors to optimize communication efficiency. We proved the superior performances of the method against state-of-the-art Bayesian filtering approaches, such as ICP, concerning both positioning errors and communication overhead.

- Driven by the lack of real-time methodologies for efficiently representing compressed distributions of latent variables, in Paper [J10], we proposed a NLoS identification method for next-generation cellular networks that automatically learn the LoS latent distributions of ADCPM samples. In particular, we presented an anomaly-detection scheme, namely deep autoencoding kernel density model (DAKDM), that is able to learn the KDE likelihood of normal, i.e., LoS samples, in a single-stage training, and without requiring a sampling procedure or storing the entire dataset for inference.

- In the context of efficient latent-feature combinations for CP tasks, in Paper [J11], we introduced a DL model designed for network-based localization, which dynamically alternated between cooperative-positioning in LoS environments and ego-positioning in NLoS conditions. It enhanced the location accuracy of CAVs in urban settings by exploiting predictions from nearby BSs. We formulated both NLoS identification and position estimation as a combined task, creating a unique loss function that simultaneously maximized the log-likelihood of the joint task and learned a concise representation of the channel.

- Finally, for performing cooperative tracking in next-generation cellular networks, in Papers [C12] and [J13], we designed of a novel teacher-student BNN method, namely Bayesian bright knowledge (BBK), capable of predicting both epistemic and aleatoric uncertainties without requiring a sampling procedure during inference, enabling it to be well-suited for real-time and safety-critical environments. We then developed a unique integration of BNNs into cellular systems with cooperative BSs to track moving targets, ensuring ease of implementation and general compatibility with any BNN method. The proposed methodology and tracking procedure highly improved performances and reliability, especially in out of distribution (OoD) scenarios, compared to traditional tracking systems and frequentist DL models.

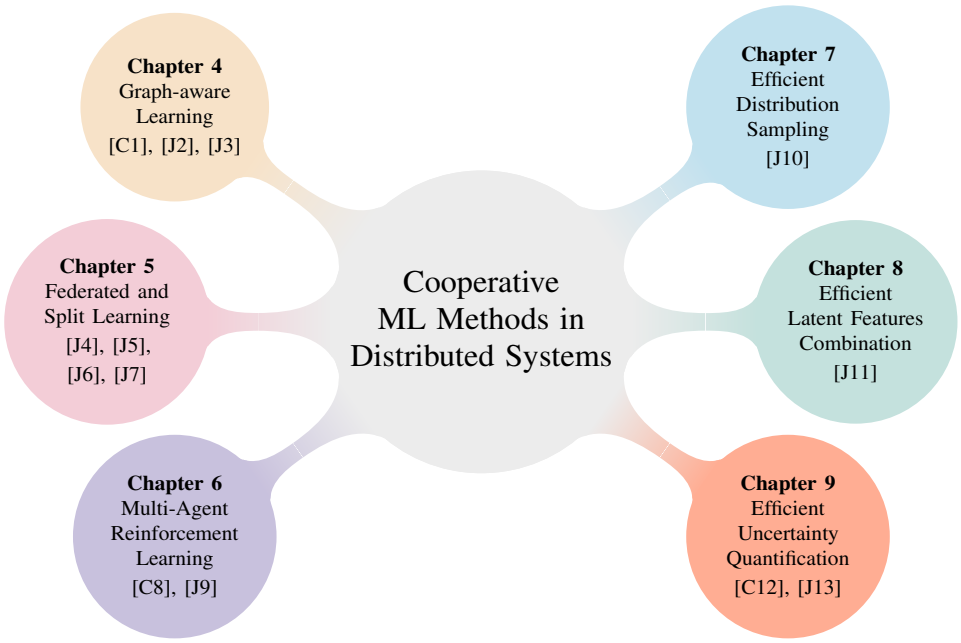


Figure 1.4: *Mind map visualizing the contributions of the PhD thesis: cooperative learning (left nodes) and cooperative inference (right nodes).*

1.4 Outline and Related Publications

The thesis is divided into three main parts. **Part I** which describe the main methodologies needed for the comprehension of the related publications. **Part II** which includes the papers centered around the cooperative learning task, mainly divided into MPNN under the C-ML paradigm, FL and SL under the stationary D-ML and FD-ML paradigms, and MARL under the non-stationary centralized-training and dynamic-decentralized-execution paradigm. **Part III** which includes the papers focused on the cooperative inference task, mainly partitioned into efficient anomaly detection techniques for NLoS identification, effective latent feature combination methods for static positioning, and novel real-time BNN methodologies for mobile positioning. For a concise summary of the publications included in the thesis, we refer to Fig. 1.4.

In particular, the reminder of the **Part I** is as follows:

- **Chapter 2**: Illustrates the main methodologies for cooperative learning in MAS, introducing the background and explaining the techniques adopted in the thesis. Moreover, we present discussions on the specific contributions and the references to the papers in the related sections.
- **Chapter 3**: Presents the main system model, DL model input, and architecture adopted for efficient inference in next-generation cellular networks. Moreover, we discuss key methodologies adopted in the papers, such as the variational inference (VI) approach.

Part II is organized as follows:

- **Chapter 4**: Includes the Papers [C1], [J2] and [J3]. This chapter mainly deals with graph-based C-ML solutions for DA and CP tasks in vehicular networks. The

proposed solutions are then compared with their corresponding Bayesian-filtering methodologies under different noise statistics and/or complex graph structures.

[C1] **B. Camajori Tedeschini**, M. Brambilla, L. Barbieri, and M. Nicoli, "Addressing data association by message passing over graph neural networks," in *2022 25th Int. Conf. Inf. Fusion (FUSION)*, Jul. 2022, pp. 01–07.

[J2] **B. Camajori Tedeschini**, M. Brambilla, L. Barbieri, G. Balducci, and M. Nicoli, "Cooperative lidar sensing for pedestrian detection: Data association based on message passing neural networks," *IEEE Trans. Signal Process.*, vol. 71, pp. 3028–3042, Aug. 2023. © 2023 IEEE. Reprinted, with permission, from IEEE.

[J3] **B. Camajori Tedeschini**, M. Brambilla, and M. Nicoli, "Message passing neural network versus message passing algorithm for cooperative positioning," *IEEE Trans. Cogn. Commun. Netw.*, vol. 9, no. 6, pp. 1666–1676, Aug. 2023. © 2023 IEEE. Reprinted, with permission, from IEEE.

- **Chapter 5**: Refers to the Papers [J4], [J5], [J6] and [J7]. This chapter explores cooperative learning in D-ML and FD-ML settings, mainly by means of real-world experiments. Here, medical and IoT networks are studied, with main attention to the privacy of data, non-IID characterizations, and convergence.

  [J4] **B. Camajori Tedeschini**, S. Savazzi, R. Stoklasa, L. Barbieri, I. Stathopoulos, M. Nicoli, and L. Serio, "Decentralized federated learning for healthcare networks: A case study on tumor segmentation," *IEEE Access*, vol. 10, pp. 8693–8708, Jan. 2022. © 2022 IEEE. Reprinted, with permission, from IEEE.

  [J5] **B. Camajori Tedeschini**, S. Savazzi, and M. Nicoli, "A traffic model based approach to parameter server design in federated learning processes," *IEEE Commun. Lett.*, vol. 27, no. 7, pp. 1774–1778, May 2023. © 2023 IEEE. Reprinted, with permission, from IEEE.

  [J6] **B. Camajori Tedeschini**, S. Savazzi, and M. Nicoli, "Weighted consensus algorithms in distributed and federated learning," *submitted to IEEE Trans. Netw. Sci. and Eng.*, pp. 1–13, 2024

  [J7] **B. Camajori Tedeschini**, M. Brambilla, and M. Nicoli, "Split consensus federated learning: an approach for distributed training and inference," *IEEE Access*, vol. 12, pp. 119 535–119 549, Aug. 2024. © 2024 IEEE. Reprinted, with permission, from IEEE.

- **Chapter 6**: Relates to the Papers [C8] and [J9]. This chapter investigates the usage of MARL algorithms in highly dynamic vehicular networks for performing CP. We compared the proposed algorithm with Bayesian-filtering methods, especially regarding communication efficiency and cooperation capabilities.

  [C8] **B. Camajori Tedeschini**, M. Brambilla, M. Nicoli, and M. Z. Win, "Cooperative positioning with multi-agent reinforcement learning," in *2024 27th Int. Conf. Inf. Fusion (FUSION)*, 2024, pp. 1–7.

  [J9] **B. Camajori Tedeschini**, M. Brambilla, M. Nicoli, and M. Z. Win, "Multi-agent reinforcement learning for distributed cooperative positioning," *IEEE Trans. Intell. Veh.*, pp. 1–16, Oct. 2024. © 2024 IEEE. Reprinted, with permission, from IEEE.

Finally, **Part III** is divided into:

- **Chapter 7**: Pertains to the Paper [J10]. This chapter explores the distribution characterization and compression of channel latent features in next-generation cellular networks. We propose an anomaly detection scheme to efficiently predict the channel NLoS identification, which outperforms current state-of-the-art methods.

  [J10] **B. Camajori Tedeschini**, M. Nicoli, and M. Z. Win, "On the latent space of mmWave MIMO channels for NLOS identification in 5G-advanced systems," *IEEE J. Sel. Areas Commun.*, vol. 41, no. 6, pp. 1655–1669, May 2023. © 2023 IEEE. Reprinted, with permission, from IEEE.

- **Chapter 8**: Describes the Paper [J11]. In this chapter, we address the problem of joint task optimization for simultaneous NLoS identification and static positioning in next-generation cellular networks. The CP is then performed by means of efficient latent features exchange and combinations among BSs.

  [J11] **B. Camajori Tedeschini** and M. Nicoli, "Cooperative deep-learning positioning in mmWave 5G-advanced networks," *IEEE J. Sel. Areas Commun.*, vol. 41, no. 12, pp. 3799–3815, Dec. 2023. © 2023 IEEE. Reprinted, with permission, from IEEE.

- **Chapter 9**: Outlines the Papers [C12] and [J13]. This chapter investigates the general integration of BNN into cellular tracking systems. For real-time mobile positioning, we propose a novel BNN methodology that outputs the data and model uncertainties, i.e., aleatoric and epistemic uncertainties, respectively, without requiring sampling procedures.

  [C12] **B. Camajori Tedeschini**, G. Kwon, M. Nicoli, and M. Z. Win, "Empowering 6G positioning and tracking with Bayesian neural networks," in *ICC 2024 - 2024 IEEE Int. Conf. Commun. (ICC)*. IEEE, Jun. 2024, pp. 1–6.

  [J13] **B. Camajori Tedeschini**, G. Kwon, M. Nicoli, and M. Z. Win, "Real-time Bayesian neural networks for 6G cooperative positioning and tracking," *IEEE J. Sel. Areas Commun.*, vol. 42, no. 9, pp. 2322–2338, Aug. 2024. © 2024 IEEE. Reprinted, with permission, from IEEE.

- **Chapter 10**: Outlines the main contributions and key takeaways of the thesis, with a discussion of possible future works.

Note that the conference articles have not been included in the related chapters for space reasons. Given the vast amount of topics and presented methods, in Fig. 1.5 we report an overview of the methods described both in **Part I** regarding background methodologies, as well as in the attached published papers in **Part II** and **Part III**. Concisely, when the objective is to cooperative learn a model, we may adopt graph-aware GNN models (e.g., the proposed MPNN for DA and CP), privacy-preserving FL and SL methods and their proposed fully-decentralized versions (e.g., CFAdp and SCFL), or non-stationary MARL algorithms such as the proposed ICP-MAPPO. On the contrary, when the model is trained, e.g., with C-ML, the next step is to perform inference. We may want, for example, to have a compressed representation of the latent features for tasks such as anomaly detection with the proposed DAKDM model, or we

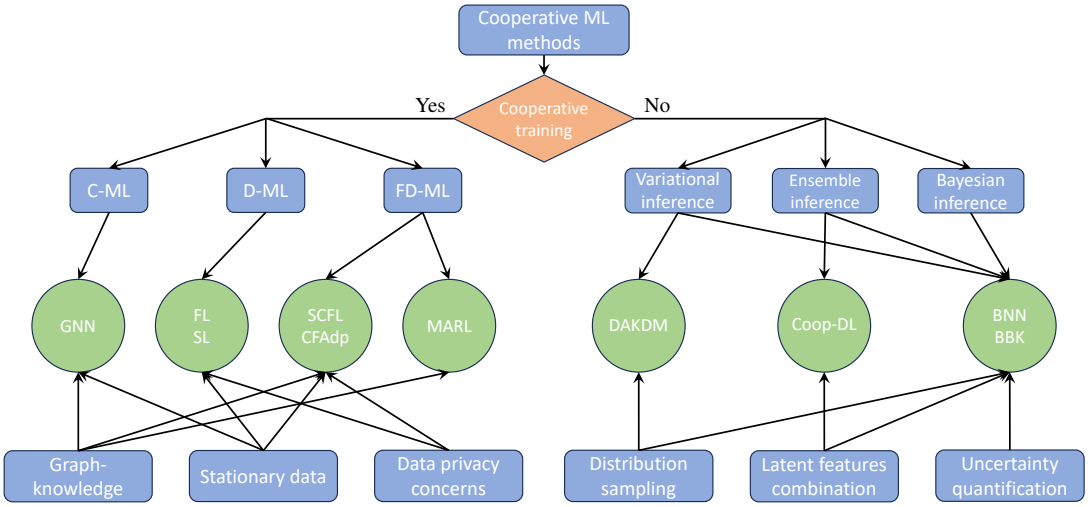


Figure 1.5: *Overview of all presented methods and their relationships.*

may want to efficiently combine different DL models' outputs to enhance inference, e.g., positioning, performances. Finally, for applications requiring accurate uncertainty quantification in real-time, we have developed advanced BNN methodologies such as the proposed BBK approach.

In addition to the mentioned contributions and articles, the research carried out during the Ph.D. has led to several other publications that have not been included as they are beyond the coverage of this thesis. A summary of all publications is provided below:

[C1] **B. Camajori Tedeschini**, M. Brambilla, L. Barbieri, and M. Nicoli, "Addressing data association by message passing over graph neural networks," in *2022 25th Int. Conf. Inf. Fusion (FUSION)*, Jul. 2022, pp. 01–07.

[J2] **B. Camajori Tedeschini**, M. Brambilla, L. Barbieri, G. Balducci, and M. Nicoli, "Cooperative lidar sensing for pedestrian detection: Data association based on message passing neural networks," *IEEE Trans. Signal Process.*, vol. 71, pp. 3028–3042, Aug. 2023. © 2023 IEEE. Reprinted, with permission, from IEEE.

[J3] **B. Camajori Tedeschini**, M. Brambilla, and M. Nicoli, "Message passing neural network versus message passing algorithm for cooperative positioning," *IEEE Trans. Cogn. Commun. Netw.*, vol. 9, no. 6, pp. 1666–1676, Aug. 2023. © 2023 IEEE. Reprinted, with permission, from IEEE.

[J4] **B. Camajori Tedeschini**, S. Savazzi, R. Stoklasa, L. Barbieri, I. Stathopoulos, M. Nicoli, and L. Serio, "Decentralized federated learning for healthcare networks: A case study on tumor segmentation," *IEEE Access*, vol. 10, pp. 8693–8708, Jan. 2022. © 2022 IEEE. Reprinted, with permission, from IEEE.

[J5] **B. Camajori Tedeschini**, S. Savazzi, and M. Nicoli, "A traffic model based approach to parameter server design in federated learning processes," *IEEE Commun. Lett.*, vol. 27, no. 7, pp. 1774–1778, May 2023. © 2023 IEEE. Reprinted, with permission, from IEEE.

[J6] **B. Camajori Tedeschini**, S. Savazzi, and M. Nicoli, "Weighted consensus algorithms in distributed and federated learning," *submitted to IEEE Trans. Netw. Sci. and Eng.*, pp. 1–13, 2024

[J7] **B. Camajori Tedeschini**, M. Brambilla, and M. Nicoli, "Split consensus federated learning: an approach for distributed training and inference," *IEEE Access*, vol. 12, pp. 119 535–119 549, Aug. 2024. © 2024 IEEE. Reprinted, with permission, from IEEE.

[C8] **B. Camajori Tedeschini**, M. Brambilla, M. Nicoli, and M. Z. Win, "Cooperative positioning with multi-agent reinforcement learning," in *2024 27th Int. Conf. Inf. Fusion (FUSION)*, 2024, pp. 1–7.

[J9] **B. Camajori Tedeschini**, M. Brambilla, M. Nicoli, and M. Z. Win, "Multi-agent reinforcement learning for distributed cooperative positioning," *IEEE Trans. Intell. Veh.*, pp. 1–16, Oct. 2024. © 2024 IEEE. Reprinted, with permission, from IEEE.

[J10] **B. Camajori Tedeschini**, M. Nicoli, and M. Z. Win, "On the latent space of mmWave MIMO channels for NLOS identification in 5G-advanced systems," *IEEE J. Sel. Areas Commun.*, vol. 41, no. 6, pp. 1655–1669, May 2023. © 2023 IEEE. Reprinted, with permission, from IEEE.

[J11] **B. Camajori Tedeschini** and M. Nicoli, "Cooperative deep-learning positioning in mmWave 5G-advanced networks," *IEEE J. Sel. Areas Commun.*, vol. 41, no. 12, pp. 3799–3815, Dec. 2023. © 2023 IEEE. Reprinted, with permission, from IEEE.

[C12] **B. Camajori Tedeschini**, G. Kwon, M. Nicoli, and M. Z. Win, "Empowering 6G positioning and tracking with Bayesian neural networks," in *ICC 2024 - 2024 IEEE Int. Conf. Commun. (ICC)*. IEEE, Jun. 2024, pp. 1–6.

[J13] **B. Camajori Tedeschini**, G. Kwon, M. Nicoli, and M. Z. Win, "Real-time Bayesian neural networks for 6G cooperative positioning and tracking," *IEEE J. Sel. Areas Commun.*, vol. 42, no. 9, pp. 2322–2338, Aug. 2024. © 2024 IEEE. Reprinted, with permission, from IEEE.

[J14] **B. Camajori Tedeschini**, M. Brambilla, L. Italiano, S. Reggiani, D. Vaccarono, M. Alghisi, L. Benvenuto, A. Goia, E. Realini, F. Grec *et al.*, "A feasibility study of 5G positioning with current cellular network deployment," *Sci. Reports*, vol. 13, no. 1, Sep. 2023.

[C15] L. Barbieri, **B. Camajori Tedeschini**, M. Brambilla, and M. Nicoli, "Implicit vehicle positioning with cooperative lidar sensing," in *ICASSP 2023 - 2023 IEEE Int. Conf. Acoust., Speech Signal Process. (ICASSP)*, Jun. 2023, pp. 1–5, iSSN: 2379-190X.

[J16] S. Roger, M. Brambilla, **B. Camajori Tedeschini**, C. Botella-Mascarell, M. Cobos, and M. Nicoli, "Deep-learning-based radio map reconstruction for V2X communications," *IEEE Trans. on Veh. Technol.*, pp. 1–9, Oct. 2023.

[J17] L. Barbieri, **B. Camajori Tedeschini**, M. Brambilla, and M. Nicoli, "Deep learning-based cooperative LiDAR sensing for improved vehicle positioning," *IEEE Trans. Signal Process.*, vol. 72, pp. 1666–1682, Mar. 2024.

[C18] U. Milasheuski, L. Barbieri, **B. Camajori Tedeschini**, M. Nicoli, and S. Savazzi, "On the impact of data heterogeneity in federated learning environments with application to healthcare networks," in *IEEE Conf. Artif. Intell.* IEEE, Jun. 2024, pp. 1017–1023.

[C19] L. Italiano, **B. Camajori Tedeschini**, M. Brambilla, and M. Nicoli, "Pedestrian positioning in urban environments with 5G technology," in *IEEE Mediterranean Commun. and Comput. Netw. Conf.* IEEE, Jun. 2024, pp. 1–6.

[J20] L. Italiano, **B. Camajori Tedeschini**, M. Brambilla, H. Huang, M. Nicoli, and H. Wymeersch, "A tutorial on 5G positioning," *IEEE Commun. Surveys & Tuts*, pp. 1–48, Aug. 2024.

[J21] M. Brambilla, M. Alghisi, **B. Camajori Tedeschini**, A. Fumagalli, F. Grec, L. Italiano, C. Pileggi, L. Biagi, S. Bianchi, A. Gatti *et al.*, "Integration of 5G and GNSS technologies for enhanced positioning: an experimental study," *IEEE Open J. of the Commun. Soc.*, pp. 1–20, Nov. 2024.

[C22] N. Schatz, S. Kim, G. Kwon, **B. Camajori Tedeschini**, M. Ricard, T. Klein, V. Weerackody, A. Conti, and M. Z. Win, "Location verification in next-generation non-terrestrial networks," in *IEEE Military Commun. Conf.* IEEE, Nov. 2024, pp. 1–6.

[C23] J. C. Morrison, N. Schatz, S. Kim, G. Kwon, **B. Camajori Tedeschini**, V. Weerackody, A. Conti, and M. Z. Win, "Sidelink-enabled cooperative localization for xG non-terrestrial networks," in *IEEE Military Commun. Conf.* IEEE, Nov. 2024, pp. 1–6.

CHAPTER 2

Learning in Agent Networks

In this chapter, we introduce the main learning techniques adopted in the thesis, by highlighting where and how they are employed. The main application areas are the medical field, IoT, and vehicular networks (see Figs. 1.1a, 1.1b, and 1.1c). We begin by characterizing the general system model of MAS and main objective functions in Sec. 2.1. Then, in Sec. 2.2, we restrict to the field of stationary C-ML, with particular focus on uncertainty quantification. Subsequently, in Sec. 2.3, we analyze the case of directly learning on graphs by leveraging the network structure. Furthermore, in Sec. 2.4, we pass from C-ML to D-ML with a main focus on data privacy and scalability requirements. Finally, in Sec. 2.5, we briefly recap the Bayesian filtering paradigm in its both centralized and distributed versions for the task of CP.

2.1 System Model and Problem Formulation

We consider a network of agents that engage in a cooperative process to reach a common objective. This is obtained by sensing the agents' state and the surroundings, and by taking actions based on the sensed data. The concept is illustrated in a C-ITS scenario in Fig. 2.1, where the agents, i.e., vehicles, cooperate to improve state estimation by dynamically selecting the radio links with the neighbors and sharing observations of passive targets, e.g., poles.

The model is generally described as a Dec-POMDP [70–73], where at time t, a set of cooperative agents $\mathcal{V} = \{1, \dots, N^{(\mathrm{A})}\}$ interacts with each other forming a connectivity graph $\mathcal{G} = \{\mathcal{V}, \mathcal{E}_t\}$. An edge $(i, j) \in \mathcal{E}_t$, where $i \neq j$, signifies the existence of a communication link from agent i to agent j. The neighbors' set of agent $i \in \mathcal{V}$ is denoted with $\mathcal{N}_{i,t}$, and each agent is defined by the state $\mathbf{s}_{i,t}^{(\mathrm{A})} \in \mathcal{S}_i^{(\mathrm{A})}$, with $\mathcal{S}_i^{(\mathrm{A})}$ being the state space of agent i which usually comprises kinematic parameters. Moreover, each agent has the possibility of performing an action $\mathbf{a}_{i,t}$ according to the specific task. We indicate with $\mathbf{a}_t = \left[\mathbf{a}_{i,t}\right]_{i\in\mathcal{V}} \in \mathcal{A}$ and $\mathbf{s}_t^{(\mathrm{A})} = \left[\mathbf{s}_{i,t}^{(\mathrm{A})}\right]_{i\in\mathcal{V}} \in \mathcal{S}^{(\mathrm{A})}$ the joint action and state, respectively, where $\mathcal{A}$ and $\mathcal{S}^{(\mathrm{A})}$ represent the joint action and state spaces, respectively. As a result of the joint action, the system produces an instantaneous reward $\mathsf{r}_t = R\Big(\mathbf{s}_t^{(\mathrm{A})}, \mathbf{a}_t\Big) \in \mathbb{R}$, where R denotes the reward function. The scenario may include a set of passive targets $\mathcal{T} = \{1, \dots, N^{(\mathrm{T})}\}$ that are detected by the agents. An example can be found in cooperative sensing applications, e.g., C-ITS, where the agents are the CAVs and the passive targets can be people, traffic lights, or trees (see e.g., DA [135, 136], CP [37] and MOT [43]). Each target $k \in \mathcal{T}$ has a state $\mathbf{s}_{i,t}^{(\mathrm{T})}$ and it is detected by agent i if $k \in \mathcal{T}_{i,t}$. In case the targets are present, the aggregate state of the system is denoted as $\mathbf{s}_t = \left[{\mathbf{s}_t^{(\mathrm{A})}}^\top {\mathbf{s}_t^{(\mathrm{T})}}^\top\right]^\top$, where $\mathbf{s}_t^{(\mathrm{T})} = \left[\mathbf{s}_{k,t}^{(\mathrm{T})}\right]_{k\in\mathcal{T}}$.

The kinematic state transition of agent i and target k at time t are modelled respectively as:

$$\mathbf{s}_{i,t}^{(\mathrm{A})} = f_{\mathbf{s}}^{(\mathrm{A})}\Big(\mathbf{s}_{i,t-1}^{(\mathrm{A})}, \mathbf{a}_{i,t}, \mathbf{z}_{\mathbf{s},i,t-1}^{(\mathrm{A})}\Big)\,, \tag{2.1}$$

and

$$\mathbf{s}_{k,t}^{(\mathrm{T})} = f_{\mathbf{s}}^{(\mathrm{T})}\Big(\mathbf{s}_{k,t-1}^{(\mathrm{T})}, \mathbf{z}_{\mathbf{s},k,t-1}^{(\mathrm{T})}\Big)\,, \tag{2.2}$$

where $\mathbf{z}_{\mathbf{s},i,t}^{(\mathrm{A})}$ and $\mathbf{z}_{\mathbf{s},k,t}^{(\mathrm{T})}$ represent the driving noise process, incorporating the uncertainty in motion. The models in (2.1) and (2.2) are associated to state-transition PDFs denoted as $p\Big(\boldsymbol{s}_{i,t}^{(\mathrm{A})}|\boldsymbol{s}_{i,t-1}^{(\mathrm{A})}, \boldsymbol{a}_{i,t}\Big)$ and $p\Big(\boldsymbol{s}_{k,t}^{(\mathrm{T})}|\boldsymbol{s}_{k,t-1}^{(\mathrm{T})}\Big)$, respectively. Since the presence of the targets is a peculiarity of sensing applications, from now on we do not consider the target state, while it will be reintroduced for the DA and CP use-cases. In the case of physically distributed agents, the state is considered agent-wise factored, i.e., $\mathcal{S}^{(\mathrm{A})} = \mathcal{S}_1^{(\mathrm{A})} \times \ldots \times \mathcal{S}_{N^{(\mathrm{A})}}^{(\mathrm{A})}$, and transition-independent, i.e., $p\Big(\boldsymbol{s}_t^{(\mathrm{A})}|\boldsymbol{s}_{t-1}^{(\mathrm{A})}, \boldsymbol{a}_t\Big) = \prod_{i=1}^{N^{(\mathrm{A})}} p\Big(\boldsymbol{s}_{i,t}^{(\mathrm{A})}|\boldsymbol{s}_{i,t-1}^{(\mathrm{A})}, \boldsymbol{a}_{i,t}\Big)$, meaning that agent actions cannot affect each other state.

Agents usually cannot directly observe the state of the system. Therefore, this uncertainty is modeled by incorporating observations (or measurements) of the state $\mathbf{o}_{i,t}$, which may vary according to the application. At each time t, the system of agents

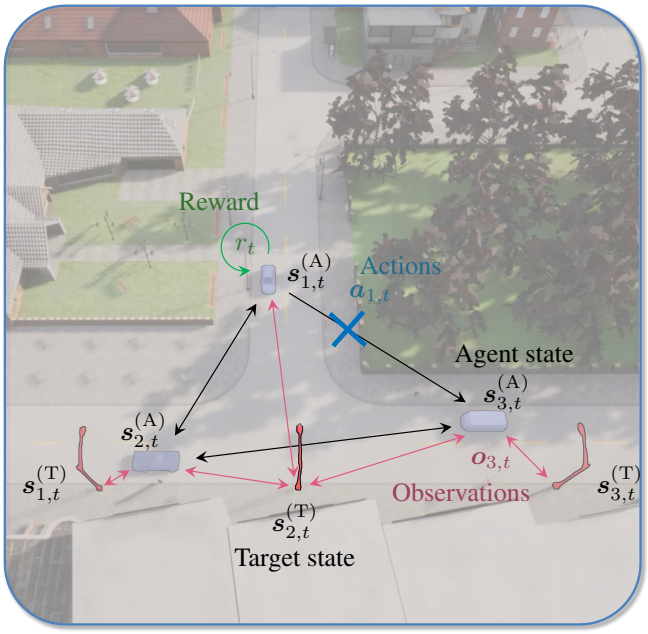


Figure 2.1: *3D representation of the scenario with three vehicles and three objects (snapshot extracted from CARLA software). Agent-to-agent communication links and agent-to-target detections are indicated with black and red arrows, respectively.*

receive a joint observation or measurement $\boldsymbol{o}_t = \left[\boldsymbol{o}_{i,t}\right]_{i\in\mathcal{V}} \in \mathcal{O}$ which is sampled from the distribution $p\left(\boldsymbol{o}_t|\boldsymbol{a}_{t-1}, \boldsymbol{s}_t^{(\mathrm{A})}\right)$, where $\mathcal{O}$ is the set of joint observations. Each observation $\mathbf{o}_{i,t}$ of agent i is modelled as:

$$\mathbf{o}_{i,t} = f_{\mathbf{o}}\left(\mathbf{a}_{i,t-1}, \mathbf{s}_{i,t}^{(\mathrm{A})}, \mathbf{z}_{\mathbf{o},i,t}^{(\mathrm{A})}\right), \tag{2.3}$$

where $\mathbf{z}_{\mathbf{o},i,t}^{(\mathrm{A})}$ is the measurement noise process. If we fix the observation $\boldsymbol{o}_t$ and action $\boldsymbol{a}_{t-1}$, the PDF $p\left(\boldsymbol{o}_t|\boldsymbol{a}_{t-1}, \boldsymbol{s}_t^{(\mathrm{A})}\right)$ becomes function of the state $\boldsymbol{s}_t^{(\mathrm{A})}$ and it is usually called likelihood function. The system is also considered as observation-independent, i.e., $p\left(\boldsymbol{o}_t|\boldsymbol{a}_{t-1}, \boldsymbol{s}_t^{(\mathrm{A})}\right) = \prod_{i=1}^{N^{(\mathrm{A})}} p\left(\boldsymbol{o}_{i,t}|\boldsymbol{a}_{i,t-1}, \boldsymbol{s}_{i,t}^{(\mathrm{A})}\right)$, meaning that each agent's observation is influenced exclusively by its own state and action, independently by the other agents' observations. Finally, since the states and rewards are not directly observable by the agents (partially observable MDP), each agent i keeps track of the so-called *histories* defined as $\boldsymbol{h}_{i,1:t} = \boldsymbol{h}_{i,t} = \left[(\boldsymbol{a}_{i,t'-1}, \boldsymbol{o}_{i,t'})\right]_{t'=1}^{t}$.

The goal of the agents is to exploit cooperation to optimize over a time horizon H two main objective functions. First they aim at estimating the state $\widehat{\boldsymbol{s}}_{i,t}^{(\mathrm{A})}$ according to the minimum mean square error (MMSE) criterion from the PDF $b_{\boldsymbol{\psi}}(\boldsymbol{s}_{i,t}^{(\mathrm{A})}|\boldsymbol{o}_{i,t}, \boldsymbol{a}_{i,t-1}, \boldsymbol{h}_{i,t-1}) = p_{\boldsymbol{\psi}}(\boldsymbol{s}_{i,t}^{(\mathrm{A})}|\boldsymbol{o}_{i,t}, \boldsymbol{a}_{i,t-1}, \boldsymbol{h}_{i,t-1})$, also kwown as beliefs, parameterized by $\boldsymbol{\psi}$. The optimal parameters $\boldsymbol{\psi}^*$ are computed by minimizing the following objective as:

$$\boldsymbol{\psi}^* = \underset{\boldsymbol{\psi}}{\operatorname{argmin}}\, J(b_{\boldsymbol{\psi}}) = \underset{\boldsymbol{\psi}}{\operatorname{argmin}}\, \mathbb{E}\left\{\sum_t \left\|\mathbf{s}_t^{(\mathrm{A})} - \widehat{\mathbf{s}}_t^{(\mathrm{A})}\right\|_2^2\right\}. \tag{2.4}$$

The second objective is to perform actions, sampled according to a policy PDF $\pi_{\boldsymbol{\theta}}(\boldsymbol{a}_{i,t}|\boldsymbol{h}_{i,t}) = p_{\boldsymbol{\theta}}(\boldsymbol{a}_{i,t}|\boldsymbol{h}_{i,t})$ defined by $\boldsymbol{\theta}$, so to maximize the expected discounted cumulative reward:

$$\boldsymbol{\theta}^* = \underset{\boldsymbol{\theta}}{\operatorname{argmax}}\, J(\pi_{\boldsymbol{\theta}}) = \underset{\boldsymbol{\theta}}{\operatorname{argmax}}\, \mathbb{E}\{\mathsf{R}_0\}\,, \tag{2.5}$$

where $\mathsf{R}_t = \sum_{t'=t}^{H-1} \gamma^{t'-t}\, \mathsf{r}_{t'}$, namely *reward-to-go*, is the cumulative discounted reward from time t to H (i.e., the end of the episode).

A compact representation of the Dec-POMDP scheme for a two-agent case is shown in Fig. 2.2, where the process is generated by the interaction over time of an agent and environment component. With reference to Fig. 2.1 and the task of efficient CP, we may see the state of the environment (also known as *world*) component as the position of the agents and the observations as the measurements of the state, e.g., agent-to-agent (A2A) ranging measurements. On the contrary, the actions may be identified as the communication link de/activation, while the reward plays the resulting effect on the positioning error. It should be noted that, in the execution phase, the agents do not observe either the state transitions of the environment or the rewards, and thus they remain on the environment component. On the opposite, only observations are directly transferred into the agent component and stored within histories, which are then used for action sampling and state estimation. Histories are simply a direct or compressed (e.g., latent) collection of all past actions and observations, and they are exploited in an analogous way with respect to Bayesian filtering approaches, e.g., Kalman filter (KF). Indeed, the state (e.g., position) estimation takes into account these histories to update the agents' beliefs about the current state. Finally, the actions produced by the agents on communication link activation have a direct effect on the environment, and thus, they are transitioned within the environment component.

2.1.1 Discussion and Contributions

One of the main difficulties in Dec-POMDP problems is how to simultaneously optimize the two objective functions. Usually, conventional MARL algorithms just solve the policy optimization problem, discarding the belief estimation for precise real-world modelling. On the other hand, Bayesian filtering or time-aware DL solutions (e.g., RNN) mainly focus on the belief estimation problem. These concepts and one possible unified solution are discussed in Papers [C8] and [J9], where we propose an algorithm for performing simultaneous CP and efficient communication optimization. The main idea is to link the two objective functions such that they become dependent on the same input, e.g., state estimate. Subsequently, to prevent optimization conflicts, the training process is divided into two phases, with each phase placing more focus on optimizing one of the objective functions while still considering the other.

2.2 From Non-Stationary to Stationary Data

In order to optimize the objective functions in (2.4) and (2.5), the agents may either continuously interact with the environment (online learning) or first collect a training dataset and then perform optimization without interacting again with the environment (offline learning) [137]. Online learning can subsequently be divided into on-policy,

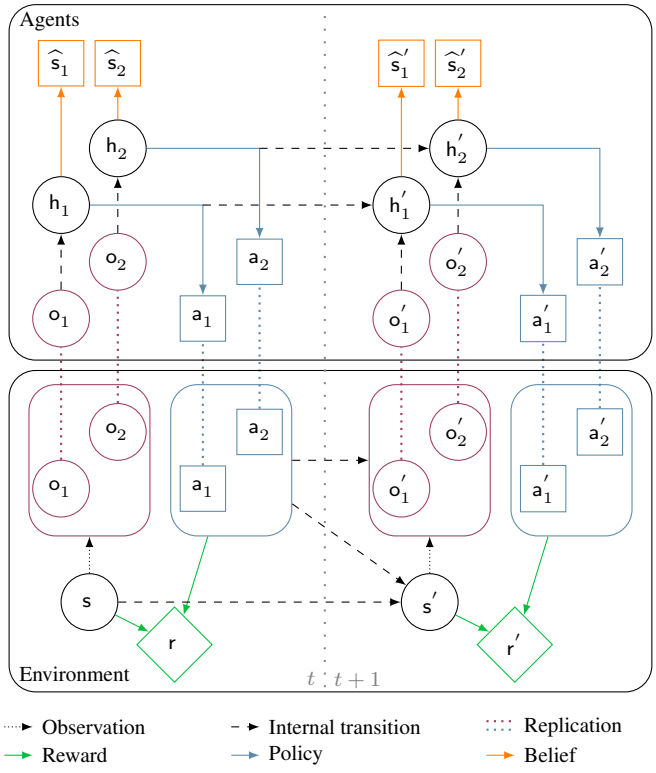


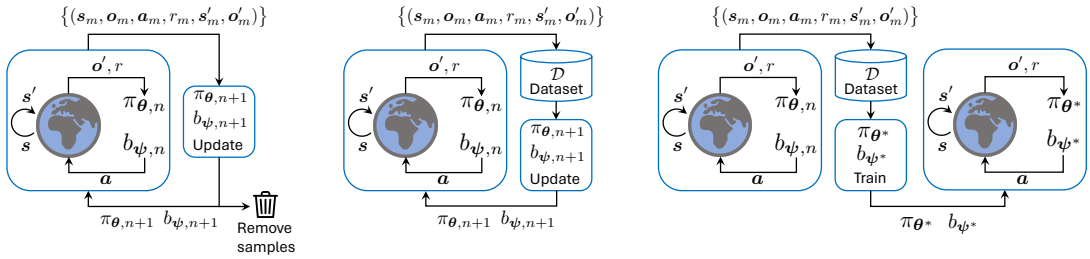
Figure 2.2: *Dec-POMDP scheme for agent and environment evolutions. Superscript* $(\cdot)'$ *stands for* $t+1$ *for graphical purposes.*

where we need to generate new samples each time the policy is changed, and off-policy, where the learning process can utilize data collected from different policies, not necessarily the one currently being improved. For a representation of online and offline learning, we refer to Fig. 2.3, where we denote the training dataset or buffer as $\mathcal{D}$. In general, both in online and offline learning, we assume non stationarity of the data, since to improve upon the behaviors captured in the dataset $\mathcal{D}$, the learned policy must enact a series of actions that diverge from those previously observed. Essentially, the goal is to derive a policy that alters the established behavior in $\mathcal{D}$, ideally enhancing the performances. However, when the assumption on the data is IID among agents and temporal dimensions, we can exploit standard ML techniques, e.g., supervised learning, to achieve high performances on data belonging to the same distribution of training data. In the following, we describe how stationarity of data can be exploited by the network of agents to optimize problems in (2.4) and (2.5).

2.2.1 Stationary Learning

Stationary learning can be viewed as a subcategory of non-stationary learning, where the optimization in (2.5) collapses to (2.4). Indeed, in conventional ML, i.e., non-RL, frameworks, there is no more concept of agents' actions, and the state becomes the target variable, usually referred to as $\mathtt{t}$[1]. Note that the *new* state $\mathtt{t}$ does not necessarily represent

[1]For easy of notation, we describe a univariate target variable, but the formulation can be straightforwardly extended to multivariate cases.



(a) Online (on-policy) (b) Online (off-policy) (c) Offline

Figure 2.3: *Different typologies of agent learning: (a) online (on-policy), (b) online (off-policy), and (c) offline learning.*

the kinematic parameters of the agents, but may be related to the ground truth or label of a specific task, e.g., regression or classification. Moreover, in stationary learning, the observations are usually referred to as input variables or features $\mathbf{x}$. The complete, i.e., aggregated, training dataset is defined as $\mathcal{D} = \{(t_m, \boldsymbol{x}_m) \mid t_m \in \mathcal{D}_t, \boldsymbol{x}_m \in \mathcal{D}_{\boldsymbol{x}}\}_{m=1}^{|\mathcal{D}|}$, where m is the sample index. A widely employed model assumes the relation between the dependent (t) and independent ($\mathbf{x}$) variables in case of regression[2] as:

$$\mathsf{t} = f_{\mathbf{x}}(\mathbf{x}) + \mathsf{z}_{\mathbf{x}}(\mathbf{x}) \,, \tag{2.6}$$

with $f_{\mathbf{x}}(\mathbf{x})$ being a non-linear function and $\mathsf{z}_{\mathbf{x}}(\mathbf{x}) \sim \mathcal{N}\left(0, \sigma_{\mathsf{z}_{\mathbf{x}}}(\mathbf{x})^2\right)$ is a random noise. We point out that (2.6) is a particular case of the relation in (2.3), where we directly express the definition of the target (i.e., state) t just in function of the input (i.e., observation) and the measurement noise.

The goal of stationary C-ML is to approximate the function $f_{\mathbf{x}}(\mathbf{x})$ with a NN $y(\mathbf{x}, \boldsymbol{\theta})$ with parameters $\boldsymbol{\theta}$, i.e., $\mathsf{t} \backsimeq y(\mathbf{x}, \boldsymbol{\theta}) + \mathsf{z}_{\mathbf{x}}(\mathbf{x})$. Therefore, the final objective is to derive the so called *predictive distribution* $p_{\mathsf{t}|\mathbf{x},\mathsf{D}}(t|\boldsymbol{x}, \mathcal{D})$, which plays an analogous role of the belief PDF. In case the parameters $\boldsymbol{\theta}$ are considered deterministic, the predictive distribution is directly parameterized by the likelihood function of $\boldsymbol{\theta}$ as $p_{\mathsf{t}|\mathbf{x},\mathsf{D}}(t|\boldsymbol{x}, \mathcal{D}) = p_{\mathsf{t}|\mathbf{x},\boldsymbol{\theta},\mathsf{D}}(t|\boldsymbol{x}, \boldsymbol{\theta}, \mathcal{D})$, and the parameters are obtained by maximum likelihood estimation (MLE) as:

$$\boldsymbol{\theta}_{\text{MLE}} = \underset{\boldsymbol{\theta}}{\text{argmin}}\, J(\mathcal{D}_t|\mathcal{D}_{\boldsymbol{x}}, \boldsymbol{\theta}) = \underset{\boldsymbol{\theta}}{\text{argmin}}\{-\log p(\mathcal{D}_t|\mathcal{D}_{\boldsymbol{x}}, \boldsymbol{\theta})\} = \sum_{m=1}^{|\mathcal{D}|} \left\| t_m - \widehat{t}_m \right\|_2^2, \tag{2.7}$$

where the negative log-likelihood of the whole training dataset $J(\mathcal{D}_t|\mathcal{D}_{\boldsymbol{x}}, \boldsymbol{\theta}) = -\log p_{\mathsf{D}_\mathsf{t}|\mathsf{D}_\mathsf{x},\boldsymbol{\theta}}(\mathcal{D}_t|\mathcal{D}_{\boldsymbol{x}}, \boldsymbol{\theta})$ is called loss or error function and it is derived from:

$$p_{\mathsf{D}_\mathsf{t}|\mathsf{D}_\mathsf{x},\boldsymbol{\theta}}(\mathcal{D}_t|\mathcal{D}_{\boldsymbol{x}}, \boldsymbol{\theta}) = \prod_{m=1}^{|\mathcal{D}|} \mathcal{N}\left(t_m; y(\boldsymbol{x}_m, \boldsymbol{\theta}), \sigma_{\mathsf{z}_{\mathbf{x}}}(\boldsymbol{x}_m)^2\right) , \tag{2.8}$$

assuming no multicollinearity in the sample and feature domains. In (2.7), $\widehat{t}$ is called *predictive mean* and it is identified as:

$$\widehat{t} = \mathbb{E}_{\mathsf{t} \sim p(t|\boldsymbol{x},\boldsymbol{\theta},\mathcal{D})}\left\{\mathsf{t}|\mathbf{x}, \boldsymbol{\theta}, \mathsf{D}\right\} \backsimeq y(\boldsymbol{x}, \boldsymbol{\theta}) \,. \tag{2.9}$$

[2] A similar formulation can be carried out for classification tasks.

Note that (2.7) is the corresponding optimization problem in (2.4), where the beliefs are substituted by the predictive distribution. In case the parameters $\mathsf{\theta}$ are considered stochastic, a Bayesian framework can be adopted, namely BNN. This implies defining a prior distribution on the parameters $p_{\mathsf{\theta}}(\boldsymbol{\theta})$, which accounts for the uncertainty of the model due to the finite size of the training dataset. The predictive distribution is then computed as:

$$p_{\mathsf{t}|\mathsf{x},\mathsf{D}}(t|\boldsymbol{x},\mathcal{D}) = \int_{\boldsymbol{\theta}} p_{\mathsf{t}|\mathsf{x},\mathsf{\theta}}(t|\boldsymbol{x},\boldsymbol{\theta'}) p_{\mathsf{\theta}|\mathsf{D}}(\boldsymbol{\theta'}|\mathcal{D}) d\boldsymbol{\theta'} \backsimeq \frac{1}{N_{\mathrm{s}}} \sum_{\ell=1}^{N_{\mathrm{s}}} p(t|\boldsymbol{x},\boldsymbol{\theta}_\ell), \tag{2.10}$$

where the posterior PDF $p_{\mathsf{\theta}|\mathsf{D}}(\boldsymbol{\theta}|\mathcal{D})$ is directly or indirectly estimated according to the specific BNN methods. The integral approximation in (2.10) is obtain by sampling N_{s} times $\boldsymbol{\theta}_\ell$ from $p_{\mathsf{\theta}|\mathsf{D}}(\boldsymbol{\theta}|\mathcal{D})$. Finally, the predictive mean is obtained as:

$$\widehat{t} = \mathbb{E}\{\mathsf{t}|\mathsf{x},\mathsf{D}\} \backsimeq \frac{1}{N_{\mathrm{s}}} \sum_{\ell=1}^{N_{\mathrm{s}}} \int_t t' p(t'|\boldsymbol{x},\boldsymbol{\theta}_\ell) dt' \backsimeq \frac{1}{N_{\mathrm{s}}} \sum_{\ell=1}^{N_{\mathrm{s}}} y(\boldsymbol{x},\boldsymbol{\theta}_\ell)\,. \tag{2.11}$$

2.2.2 Learning Uncertainty Quantification

Whenever computing the predictive mean in (2.11), we may be interested in also evaluating the uncertainty of the prediction. Indeed, this can be useful to assess the reliability of the ML model, particularly in critical applications such as CAVs. We usually distinguish between aleatoric and epistemic uncertainties. Aleatoric uncertainty originates from the data generation process described in (2.6) as:

$$\mathbb{V}\{\mathsf{t}|\mathsf{x},\mathsf{\theta}\} = \mathbb{V}\{\mathsf{z}_{\mathsf{x}}(\boldsymbol{x})|\mathsf{x},\mathsf{\theta}\} = \sigma_{\mathsf{z}_{\mathsf{x}}}(\boldsymbol{x})^2. \tag{2.12}$$

Given that every training point in $\mathcal{D}$ includes a realization of the noise $\mathsf{z}_{\mathsf{x}}(\boldsymbol{x})$, this type of uncertainty is inherent within the data itself and cannot be diminished by adding more training samples. In case the noise $\mathsf{z}_{\mathsf{x}}(\mathsf{x})$ is a function of x, we talk about heteroscedastic aleatoric uncertainty, whereas if it is independent of the input x, we talk about homoscedastic aleatoric uncertainty [138]. Still, in both these cases since it is data-dependent, it can be learned by a deterministic NN through a specific loss function with the model $\sigma_{\mathsf{z}_{\mathsf{x}}}(\mathsf{x})^2 \backsimeq y_{\mathrm{al}}(\mathsf{x},\mathsf{\theta}) + \mathsf{\xi}_{\mathrm{al}}$ [107], where $y_{\mathrm{al}}(\mathsf{x},\mathsf{\theta})$ is an additional NN output which predicts the aleatoric uncertainty of x, and $\mathsf{\xi}_{\mathrm{al}} \sim \mathcal{N}(0, {\sigma_{\mathsf{\xi}_{\mathrm{al}}}}^2)$.

Conversely, epistemic uncertainty arises from the variability in the NN parameters, represented by the random variables $\mathsf{\theta}$. This variability contributes to the measure of uncertainty in the output as:

$$\mathbb{V}\{\mathsf{t}|\mathsf{x},\mathsf{z}_{\mathsf{x}}(\mathsf{x})\} \backsimeq \mathbb{V}\{y(\boldsymbol{x},\mathsf{\theta})|\mathsf{x},\mathsf{z}_{\mathsf{x}}(\mathsf{x})\}\,. \tag{2.13}$$

In traditional NNs, where parameters are estimated as point values, this type of uncertainty is considered zero. However, in BNN, epistemic uncertainty can be mitigated and decreased by incorporating additional training data [138]. Therefore, with BNN we are able to express the total predictive variance as follows:

$$\mathbb{V}\{\mathsf{t}|\mathsf{x},\mathsf{D}\} \backsimeq \frac{1}{N_{\mathrm{s}}} \sum_{\ell=1}^{N_{\mathrm{s}}} \int_t \Big(t' - \mathbb{E}\{\mathsf{t}|\mathsf{x},\mathsf{D}\}\Big)^2 p(t'|\boldsymbol{x},\boldsymbol{\theta}_\ell) dt'$$

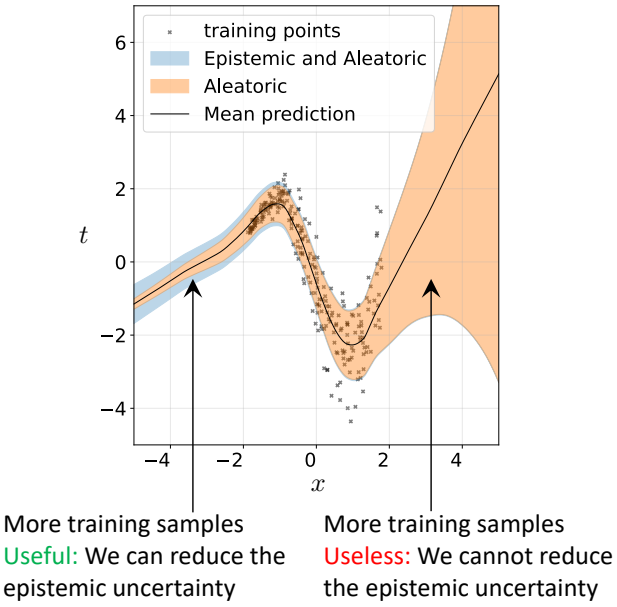


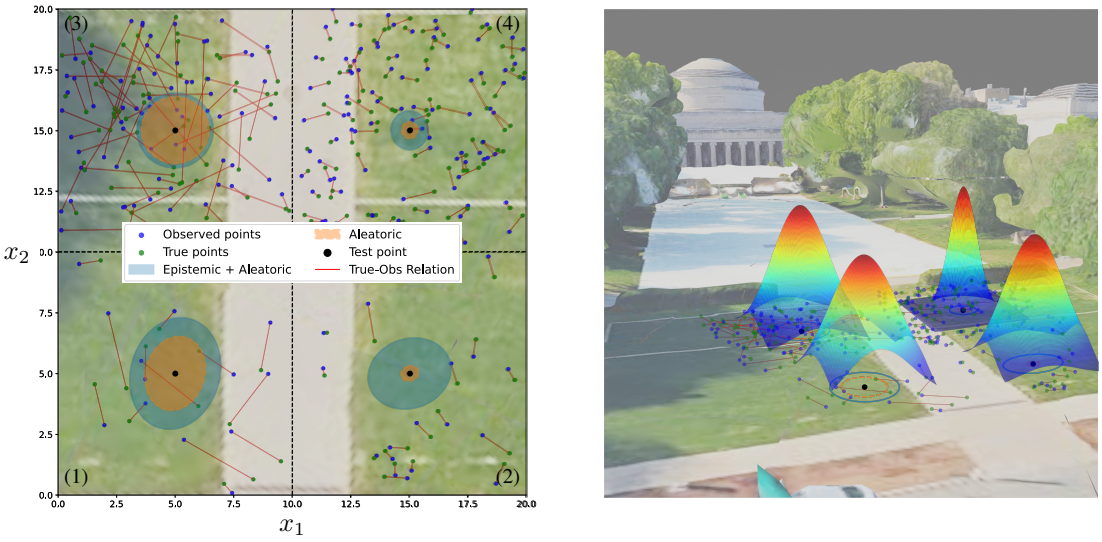
Figure 2.4: *Epistemic and aleatoric uncertainty visualization.*

$$\begin{aligned} &\backsimeq \mathbb{V}\{y(\boldsymbol{x}, \boldsymbol{\theta})|\mathbf{x}, \mathsf{D}\} + \mathbb{V}\{\mathsf{z}_{\mathbf{x}}(\boldsymbol{x})|\mathbf{x}, \mathsf{D}\} \\ &\backsimeq \frac{1}{N_{\mathrm{s}}} \sum_{\ell=1}^{N_{\mathrm{s}}} y(\boldsymbol{x}, \boldsymbol{\theta}_\ell)^2 - \left[\frac{1}{N_{\mathrm{s}}} \sum_{\ell=1}^{N_{\mathrm{s}}} y(\boldsymbol{x}, \boldsymbol{\theta}_\ell)\right]^2 + \frac{1}{N_{\mathrm{s}}} \sum_{\ell=1}^{N_{\mathrm{s}}} y_{\mathrm{al}}(\boldsymbol{x}, \boldsymbol{\theta}_\ell)\,, \end{aligned} \tag{2.14}$$

where the first two terms are the epistemic uncertainty prediction, while the last term is the aleatoric uncertainty prediction.

2.2.3 Discussion and Contributions

The primary advantage of BNNs is their ability to differentiate between the two types of uncertainties, thus separating the uncertainty due to the variance of the estimated parameters and the intrinsic variance present in the data. This is particularly useful for identifying input domains where additional data collection could reduce overall prediction uncertainty. For illustration, in Fig. 2.4, we present a toy case where the intrinsic noise in the data increases along the x-axis. By distinguishing between these uncertainties, we can explain the source of model uncertainty and make informed decisions on where to gather more useful data. To help better visualize the uncertainties in a familiar context, in Fig. 2.5 we show the uncertainty estimation obtained in a 3D scenario (similar to Fig. 2.1) with a BNN. In particular, we divided an area of $20 \times 20\,\mathrm{m}$ into four zones: bottom-left (Fig.2.5a.1) with high aleatoric and epistemic uncertainties; top-left (Fig.2.5a.2) with high aleatoric and low epistemic uncertainties; bottom-right (Fig.2.5a.3) with low aleatoric and high epistemic uncertainties; and top-right (Fig.2.5a.4) with low aleatoric and low epistemic uncertainties. In each quadrant, we then reported the 1-σ uncertainty ellipse for the aleatoric and total uncertainty, and the derived approximated Gaussian PDF. Note that, by gathering more data points, we are able to decrease only the blue area (epistemic). Therefore, by fully evaluating the



(a) 2D representation

(b) 3D representation

Figure 2.5: *Epistemic and aleatoric uncertainty visualization in a 3D scenario. (a) Bird's-eye view representation and (b) 3D representation obtained through Google Maps, RenderDoc, and Blender software.*

two uncertainties, we conclude that it is beneficial to gather more data in Fig. 2.5a.2 and it is useless in Fig. 2.5a.3.

However, the main drawback of BNNs is that they require N_s samples to fully evaluate those uncertainties. This can represent an issue in real-time applications, such as mobile tracking, where performing sampling operations may accumulate an unacceptable delay. Existing real-time BNNs methods, e.g., BDK, are based on teacher-student techniques where we practically train two models: a teacher BNN (T), trained with any Bayesian methodology, and a student NN (S), trained to imitate the output of the teacher (both in terms of output mean and variance). The new input-output model becomes:

$$\begin{aligned} &(\text{T}): \quad \mathsf{t} = y^{(\text{T})}(\mathbf{x}, \boldsymbol{\theta}) + \mathsf{z}_{\mathbf{x}}^{(\text{T})}(\mathbf{x}) \\ &(\text{S}): \quad \begin{cases} \mathsf{t} & = y^{(\text{S})}(\mathbf{x}, \boldsymbol{w}) + \mathsf{z}_{\mathbf{x}}^{(\text{S})}(\mathbf{x}) \\ \sigma_{\mathsf{z}_{\mathbf{x}}^{(\text{S})}}(\mathbf{x})^2 & = y_{\text{al}}^{(\text{S})}(\mathbf{x}, \boldsymbol{w}) + \xi_{\text{al}}^{(\text{S})}, \end{cases} \end{aligned} \tag{2.15}$$

where $\mathsf{z}_{\mathbf{x}}^{(\text{S})}(\mathbf{x})$, $\mathsf{z}_{\mathbf{x}}^{(\text{S})}(\mathbf{x})$ and $\xi_{\text{al}}^{(\text{S})}$ are Gaussian noises, and $\boldsymbol{w}$ are the parameters of the student NN. These methods, however, fail to fully capture the distinction between the two uncertainties, simply because they only output the aleatoric uncertainty. To handle this problem, in Paper [J13], we present an original BNN technique, with specific loss functions and training algorithm, to perform real-time predictions, while maintaining the full advantages of BNNs in terms of robustness and uncertainty evaluation.

To better highlight the advantages of the proposed BNN technique, namely BBK, we now report a comparison analysis of the quality of the uncertainty quantification provided by traditional sampling-based BNN methods, conventional NN, and the proposed BBK algorithm in the same case of Fig. 2.4. In particular, for the ground truth process, we employed a GP with a radial basis function (RBF) kernel to generate the data. Specifically, the GP has an input space defined by $x \in [-5, 5]$, with a length scale and

variance set to capture smooth variations in the data. The aleatoric (data) noise is defined as a heteroscedastic function, increasing as $|x + 2|$, to simulate scenarios where data uncertainty grows across the input space. Regarding the BNN methods, we compared a Markov chain Monte Carlo (MCMC) method, i.e., the stochastic gradient Langevin dynamics (SGLD) [139] algorithm, and two VI-based methods, i.e., Monte Carlo (MC)-Dropout [140] and Bayes by backpropagation (BBP) [141]. The conventional NN was trained to learn the aleatoric uncertainty with the following loss function [138]:

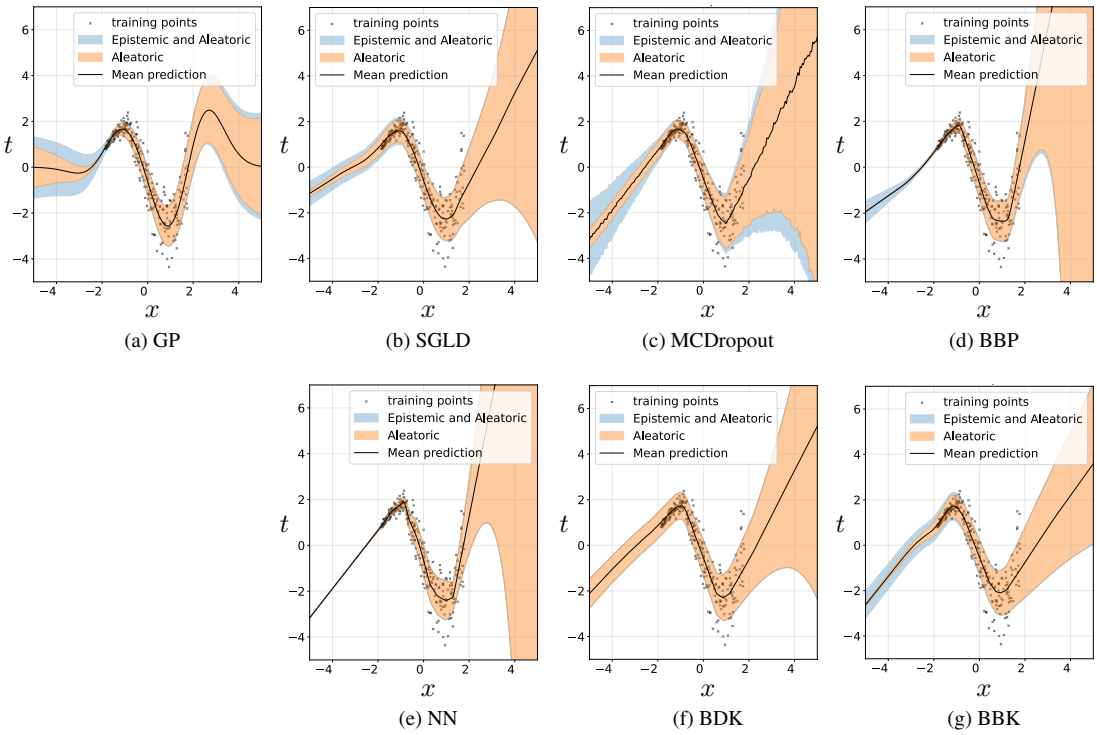
$$J(\mathcal{D}_t|\mathcal{D}_x, \boldsymbol{\theta}) = \frac{1}{|\mathcal{D}|}\sum_{m=1}^{|\mathcal{D}|}\left\{\frac{\|y(x_m, \boldsymbol{\theta}) - t_m\|_2^2}{2y_{\text{al}}(x_m, \boldsymbol{\theta})} + \frac{1}{2}\log\left(y_{\text{al}}(\boldsymbol{x}_m, \boldsymbol{\theta})\right)\right\}, \tag{2.16}$$

where the first term in the summation penalizes the squared error inversely weighted by the predicted aleatoric uncertainty, encouraging the model to increase the predicted uncertainty in regions with higher residual errors. The second term acts as a regularizer, preventing the model from assigning excessively high uncertainty values by penalizing large aleatoric predictions. Finally, the teacher-students methods, i.e., BDK [111] and the proposed BDK, adopted the SGLD as the teacher model to train a student network with high-quality, sample-based estimates of uncertainty.

From the results, shown in Fig. 2.6, we can see that the sampling-based BNN models in the first row provide both aleatoric and epistemic uncertainties estimation through multiple forward passes but require extensive computational resources for real-time applications. In contrast, conventional NN models lack mechanisms to quantify the epistemic uncertainty and just provide the estimation of the aleatoric one. Moreover, we can see that in OoD areas, e.g., $|x| > 2$, the model tends to be overconfident in its prediction by assigning either very large or small values of uncertainties. The teacher-student method BDK reduces this effect by learning from the teacher the overall uncertainty quantification in OoD areas, however, without distinguishing them. Finally, unlike standard BDK, the proposed BBK student model achieves comparable uncertainty quantification with respect to the ground truth process, and it is able to differentiate between aleatoric and epistemic uncertainties, allowing for more fine and efficient uncertainty quantification.

2.3 From Single Node to Graph Learning

Whenever dealing with learning in graphs, it is intuitive to use NN that are directly integrated with the graph's structure, leveraging its topology and connectivity. These types of NN are known as GNN [44, 142] and they have been explored across various learning paradigms, including RL, unsupervised, semi-supervised, and supervised learning contexts, for their capability to scale and adapt to larger, unseen graphs. Indeed, similar to boosting methods [143], they adopt very simple or weak models, whose bias is reduced with subsequent N_{MP} message passing iterations across the graph. We would like to point out that GNN perform *centralized* learning on graphs, that is, all the data is gathered or transferred into a single computing node where graph learning is performed. Therefore, the graph structure is a logical structure that may or not reflect the physical graph. The main advantage is that, if the real network structure that generated the data changes, the same NN can be adopted for inference (and/or retraining).



(a) GP (b) SGLD (c) MCDropout (d) BBP

(e) NN (f) BDK (g) BBK

Figure 2.6: *Comparison of different methodologies for uncertainty estimation: (a) GP, (b) SGLD, (c) MC-Dropout, (d) BBP, (e) NN, (f) BDK, and (g) BBK.*

In particular, vanilla GNN are composed of three NN, or update functions, namely $y_{\mathrm{v}}(\cdot)$, $y_{\mathrm{e}}(\cdot)$, and $y_{\mathrm{t}}(\cdot)$ for node, edge and global inference, respectively. As in conventional learning, the goal is to learn the correspondence between target $\mathtt{t}$ and input features $\mathbf{x}$ by optimizing the update functions' parameters. The forward, i.e., inference part, starts with the encoding[3] of the input features $\mathbf{x}$ into node and edge embedding representations, defined as $\mathbf{h}_{\mathrm{v}}^{(0)}$ and $\mathbf{h}_{\mathrm{e}}^{(0)}$, respectively. These are composed by the single-node and edge embeddings as $\mathbf{h}_{\mathrm{v}}^{(0)} = \left\{\mathbf{h}_{\mathrm{v},i}^{(0)}\right\}_{i\in\mathcal{V}}$ and $\mathbf{h}_{\mathrm{e}}^{(0)} = \left\{\left\{\mathbf{h}_{\mathrm{e},j\to i}^{(0)}\right\}_{j\in\mathcal{N}_i}\right\}_{i\in\mathcal{V}}$, respectively. Moreover, also the target $\mathtt{t}$ is encoded into global attributes or embeddings $\mathbf{h}_{\mathrm{t}}^{(0)}$. For each message passing iteration p, the logical building blocks and detailed algorithm are shown in Fig. 2.7 and Algorithm 1, respectively, where $\rho_{\mathrm{e}\to\mathrm{v}}$, $\rho_{\mathrm{v}\to\mathrm{t}}$ and $\rho_{\mathrm{e}\to\mathrm{t}}$ are called aggregation functions and can be whatever functions that are insensitive to input order (e.g., maximum, mean, element-wise summation).

GNN are a general framework that includes many typologies of learning on graphs, such as independent recurrent blocks (IRB) [144], MPNN [145], non-local neural networks (NLNN) [146], relational networks (RN) [147] and deep sets (DS) [148]. Among these, MPNN are ideal when dealing with heterogeneous graphs with rich edge attributes and where the focus is on accurately capturing the interaction dynamics between nodes via these attributes. Indeed, the node and edge embeddings are not any more functions of the global ones, but just depend on the local neighborhood's aggregated messages and possibly the individual node's or edge's features. As an

[3]We do not define the encoding NN to focus on the peculiar mechanism and models of GNN.

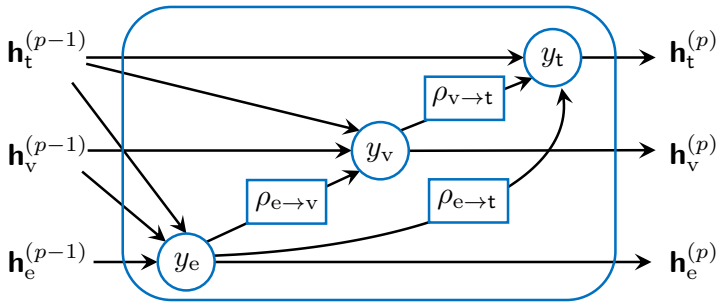


Figure 2.7: *Single building blocks iteration of the GNN message passing procedure.*

Algorithm 1 GNN message passing procedure

procedure GNN($\mathbf{h}_{\mathrm{v}}^{(p-1)}, \mathbf{h}_{\mathrm{e}}^{(p-1)}, \mathbf{h}_{\mathrm{t}}^{(p-1)}$) ▷ Iteration p
  for each node $i \in \mathcal{V}$ **do** ▷ Update edge embeddings
    $\mathbf{h}_{\mathrm{e},\,j\to i}^{(p)} = y_{\mathrm{e}}\big(\mathbf{h}_{\mathrm{v},i}^{(p-1)}, \mathbf{h}_{\mathrm{v},j}^{(p-1)}, \mathbf{h}_{\mathrm{e},\,j\to i}^{(p-1)}, \mathbf{h}_{\mathrm{t}}^{(p-1)}, \boldsymbol{\theta}\big) \quad \forall j \in \mathcal{N}_i$
  end for
  for each node $i \in \mathcal{V}$ **do** ▷ Update node embeddings
    $\mathbf{h}_{\mathrm{v},i}^{(p)} = y_{\mathrm{v}}\Big(\mathbf{h}_{\mathrm{v},i}^{(p-1)}, \rho_{\mathrm{e}\to\mathrm{v}}\Big(\big\{\mathbf{h}_{\mathrm{e},\,j\to i}^{(p)}\big\}_{j\in\mathcal{N}_i}\Big), \mathbf{h}_{\mathrm{t}}^{(p-1)}, \boldsymbol{\theta}\Big)$
  end for
  $\mathbf{h}_{\mathrm{t}}^{(p)} = y_{\mathrm{t}}\Big(\rho_{\mathrm{e}\to\mathrm{t}}\big(\mathbf{h}_{\mathrm{e}}^{(p)}\big), \rho_{\mathrm{v}\to\mathrm{t}}\big(\mathbf{h}_{\mathrm{v}}^{(p)}\big), \mathbf{h}_{\mathrm{t}}^{(p-1)}, \boldsymbol{\theta}\Big)$ ▷ Update global embeddings
end procedure

example, we now describe the problem of node regression and its particular loss function to update the NN parameters.

2.3.1 Message Passing Neural Networks

In MPNN for node regression, the global embeddings are usually represented directly by the target $\mathtt{t}_i$ (for each node i), or by an encoded version of it, without being elaborated by the message passing procedure. At each iteration $p = 1, \ldots, N_{\mathrm{MP}}$, each node $i \in \mathcal{V}$ sends the following message to its neighbors $j \in \mathcal{N}_i$:

$$\mathbf{h}_{\mathrm{e},\,j\to i}^{(p)} = y_{\mathrm{e}}\Big(\mathbf{h}_{\mathrm{v},i}^{(p-1)}, \mathbf{h}_{\mathrm{v},j}^{(p-1)}, \mathbf{h}_{\mathrm{e},\,j\to i}^{(p-1)}, \boldsymbol{\theta}\Big) \quad \forall j \in \mathcal{N}_i\,, \tag{2.17}$$

with:

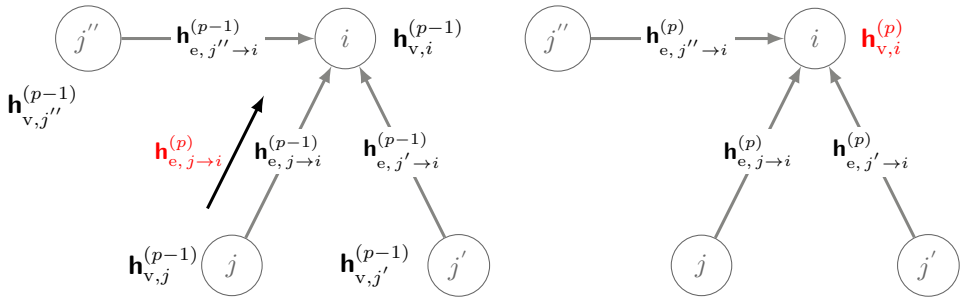
$$\mathbf{h}_{\mathrm{v},i}^{(p)} = y_{\mathrm{v}}\Big(\mathbf{h}_{\mathrm{v},i}^{(p-1)}, \textstyle\sum_{j\in\mathcal{N}_i} \mathbf{h}_{\mathrm{e},\,j\to i}^{(p)}, \boldsymbol{\theta}\Big), \tag{2.18}$$

where we adopted the summation as an aggregation function. An illustration of the updates is shown in Fig. 2.8. Subsequently, the targets are predicted as:

$$\widehat{\mathtt{t}}_i^{(p)} = y_{\mathtt{t}}\Big(\mathbf{h}_{\mathrm{v},i}^{(p)}\Big) \quad \forall i \in \mathcal{V}, \tag{2.19}$$

where we considered the global update function for each node. After N_{MP} message passing iterations, the final target prediction is $\widehat{\mathtt{t}}_i^{(N_{\mathrm{MP}})}$. For each input sample $\boldsymbol{x}_m$ (related to each node and/or edge according to the problem) of the training dataset $\mathcal{D}$, the NN parameters are updated with MLE through the following loss function:

$$J(\mathcal{D}_t|\mathcal{D}_{\boldsymbol{x}}, \boldsymbol{\theta}) = \sum_{m=1}^{|\mathcal{D}|} \sum_{p=1}^{N_{\mathrm{MP}}} \sum_{i\in\mathcal{V}} \Big\| t_{i,m} - \widehat{t}_{i,m}^{(p)} \Big\|_2^2, \tag{2.20}$$



(a) Edge embeddings' update

(b) Node embeddings' update

Figure 2.8: *MPNN iteration with (a) edge embeddings' update, and (b) node embeddings' update, represented in red.*

where we removed constant values in $\boldsymbol{\theta}$ such as normalization terms. Note that the target at each node i and input sample m are considered as additional input samples that reduce the variance of the model. On the contrary, the message passing iterations p are considered as subsequent layer of a bigger model (i.e., the target $t_{i,m}$ is not function of the iteration p), thus reducing the bias of the weak update functions models.

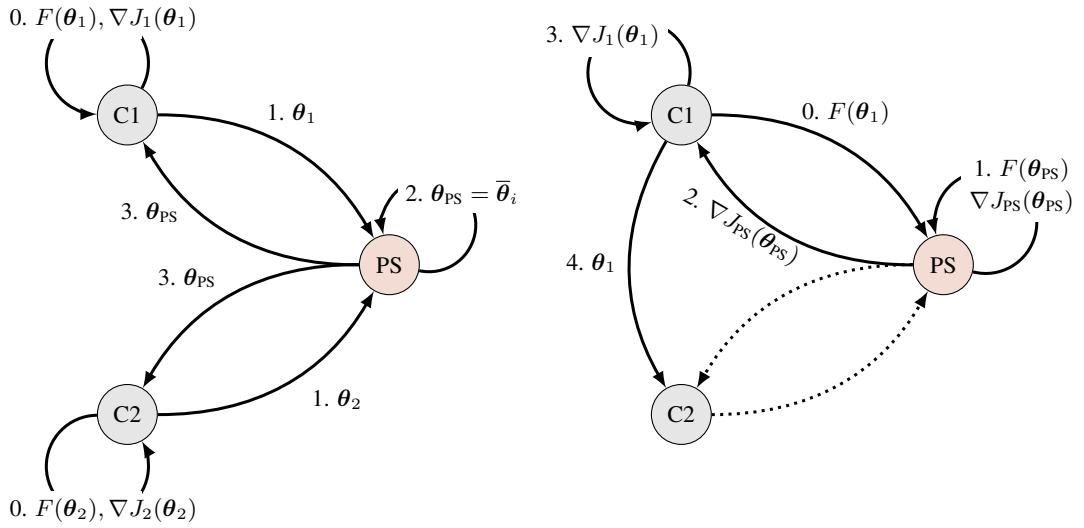
2.3.2 Discussion and Contributions

Given the ability of MPNN on learning complex non-linear relations within graphs and being scalable in unseen much bigger networks, in Papers [C1], [J2] and [J3], we propose their application or inclusion into sensing tasks such as DA and CP. In particular, in Papers [C1] and [J2], we introduce a new logic graph that maps the vehicles' detections into nodes, for subsequent edge classification (i.e., detections' association) by means of a custom MPNN model. Given the foreseen usage of these methods in real-world applications, we tested the MPNN against conventional SPADA algorithms in terms of realistic noise distributions, e.g., derived from object detection models like Pointpillars [149]. The proposed MPNN-based DA method has also been integrated into general ICP systems in Papers [J15] and [J17] (not included in the thesis).

On the contrary, in Paper [J3], due to their analogy with the message passing procedure in MPA, we integrated MPNN into a belief propagation (BP)-inspired CP scheme. We propose a new LSTM-MPNN model that is trained with the C-ML paradigms, whereas the inference is performed in a distributed way among agents. In particular, the LSTM learns the kinematic models of the agents and produces the *prediction phase* (or step) of conventional Bayesian-filtering methods. On the contrary, the MPNN elaborates the observations gathered at each agent by means of the message passing procedure. For a detailed description of the MPA methodology for CP, we refer to Sec. 2.5.

2.4 From Centralized to Decentralized Learning

With the rapid increase in volume and complexity of data available at different locations and machines, C-ML may face difficulties in terms of scalability and efficiency. Consequently, there is a growing demand for alternative methodologies that can effectively address the increased complexity and security issues while retaining the benefits of



(a) Federated Learning

(b) Split Learning

Figure 2.9: *Schematic example of (a) FL and (b) SL in a network of two clients and a PS. The client model parameters and gradients are indicated with $\boldsymbol{\theta}_i$ and $\nabla J_i(\boldsymbol{\theta}_i)$, respectively. On the contrary, the global or PS model parameters and gradients are indicated with $\boldsymbol{\theta}_{PS}$ and $\nabla J_{PS}(\boldsymbol{\theta}_{PS})$, respectively. The weighted average of the FL is indicated with a thick bar. Finally, the forward pass is indicated with $F(\cdot)$, back-propagation is indicated with the model gradients inside a self-loop, and dashed lines represent the next timestamp.*

conventional ML techniques. D-ML [53, 150, 151] has emerged as a viable solution for distributing the processing and avoiding the aggregation of data at a single central entity. Popular D-ML mechanisms are FL [11] and SL [58], which allows spatially distributed agents (i.e., clients) to cooperatively train a global ML model without the need of exchanging their local private data. In the following, we report the FL and SL methodology for the deterministic parameters $\boldsymbol{\theta}$ case, but a similar Bayesian formulation can be applied as well. The objective is always to optimize (2.7) but in a distributed manner.

A conceptual comparison of the two D-ML methods is reported in Figure 2.9, in which we present the FL (Figure 2.9a) and SL (Figure 2.9b) frameworks, indicating their main steps indexed in chronological order. Specifically, the steps for FL are: 0) local model optimization, 1) aggregation, 2) broadcast of the updated global model; while the steps for SL are: 0) client forward pass and exchange of smashed data, 1) PS forward pass and back-propagation, 2) exchange of PS gradients, 3) client back-propagation, 4) client model exchange with next neighbors.

2.4.1 Federated Learning

In FL, clients individually maintain and train local copies of DL models using their own local data, while a coordinating PS combines these models into a global one shared among all clients. Specifically, the objective function of the FL procedure is to obtain a

global DL model defined by the parameters $\boldsymbol{\theta}$ by minimizing:

$$\boldsymbol{\theta}_{\mathrm{PS}} = \underset{\boldsymbol{\theta}}{\operatorname{argmin}}\, J(\mathcal{D}_t|\mathcal{D}_{\boldsymbol{x}}, \boldsymbol{\theta})\,, \tag{2.21}$$

where the loss function $J(\mathcal{D}_t|\mathcal{D}_{\boldsymbol{x}}, \boldsymbol{\theta})$ is specified by:

$$J(\mathcal{D}_t|\mathcal{D}_{\boldsymbol{x}}, \boldsymbol{\theta}) = \frac{1}{N^{(\mathrm{A})}} \sum_{i=1}^{N^{(\mathrm{A})}} \frac{|\mathcal{D}_i|}{\sum_{i' \in \mathcal{V}} |\mathcal{D}_{i'}|} J_i(\mathcal{D}_{t,i}|\mathcal{D}_{\boldsymbol{x},i}, \boldsymbol{\theta})\,, \tag{2.22}$$

with J_i representing the loss determined by client i utilizing the local dataset $\mathcal{D}_i = \{(t_m, \boldsymbol{x}_m) \mid t_m \in \mathcal{D}_{t,i}, \boldsymbol{x}_m \in \mathcal{D}_{\boldsymbol{x},i}\}_{m=1}^{|\mathcal{D}_i|}$. To obtain the global model $\boldsymbol{\theta}_{\mathrm{PS}}$, an iterative process is carried out with each iteration involving a *local model optimization* step performed by the client and followed by an *aggregation* step executed on the PS.

At federated round $n = 1, \ldots, N_{\mathrm{FL}}$, a set $\mathcal{V}_n \subseteq \mathcal{V}$ of clients is chosen to carry out the training procedure, i.e., the local model optimization step. Clients are required to generate local models via optimization in the FL process, typically employing supervised and gradient-based schemes, e.g., Adam or stochastic gradient descent (SGD) optimizers [152], with mini-batch $\mathcal{B}_i$ and learning rate η. Each client $i \in \mathcal{V}_n$ carries out N_{epoch} local epochs before sending their model to the PS, which is tasked with updating the global model.

In the vanilla PS-based FL, i.e., FedAvg [49], the number of samples $|\mathcal{D}_i|$ from each client determines the weights used in the aggregation step to execute a weighted average:

$$\boldsymbol{\theta}_{\mathrm{PS},n} = \sum_{i \in \mathcal{V}_n} \alpha_{i,\mathcal{V}_n} \boldsymbol{\theta}_{i,n}, \tag{2.23}$$

where $\boldsymbol{\theta}_{\mathrm{PS},n}$ is the PS global model, $\boldsymbol{\theta}_{i,n}$ is the local model of client i and $\alpha_{i,\mathcal{V}_n} = \frac{|\mathcal{D}_i|}{\sum_{i' \in \mathcal{V}_n} |\mathcal{D}_{i'}|}$ are the mixing weights related to client i.

On the contrary, FD-ML architectures, such as decentralized FL, do not employ the PS but rather share their local model(s) repeatedly over A2A links so as to reach a consensus on a global model (consensus-based FL) [18]. The consensus-based algorithm, referred to as CFA, operates as follows. At round n, each client $i \in \mathcal{V}_n$ performs a local optimization step and then exchanges the model weights with its neighbors $\mathcal{N}_{i,n}$. Subsequently, an aggregation step is executed, similar to the PS:

$$\boldsymbol{\psi}_{i,n} = \boldsymbol{\theta}_{i,n} + \epsilon_{\mathrm{FL},n} \sum_{k \in \mathcal{N}_{i,n}} \alpha_{k,\mathcal{N}_{i,n}} \left(\boldsymbol{\theta}_{k,n} - \boldsymbol{\theta}_{i,n}\right), \tag{2.24}$$

where $\boldsymbol{\psi}_{i,n}$ represents the aggregated model, $\epsilon_{\mathrm{FL},n}$ is the consensus step-size which modulates the memory of previous models and $\alpha_{k,\mathcal{N}_{i,n}} = \frac{|\mathcal{D}_k|}{\sum_{k' \in \mathcal{N}_{i,n}} |\mathcal{D}_{k'}|}$ are the mixing weights related to client k, based on the number of samples retained in each client. Note that the aggregated model $\boldsymbol{\psi}_{i,n}$ represents an estimate of the global model as seen by client i and its neighborhood $\mathcal{N}_{i,n}$. However, as opposed to (2.23), here the aggregated model is obtained by taking into account the error between the local model and the neighbor ones. The complete pseudo-code for CFA is presented in Algorithm 2. For ease of notation, the local model optimization step has been represented as a single-batch model update. It should be noted that the algorithm operates in the same manner if clients exchange gradients of the local model update instead of model parameters.

Algorithm 2 Consensus-driven Federated Averaging

procedure CFA($\mathcal{N}_{i,n}, \epsilon_{\mathrm{FL},n}, \eta$) ▷ Run on client i
  initialize $\boldsymbol{\theta}_{i,0} \leftarrow$ client i
  for each round $n = 1, \ldots, N_{\mathrm{FL}}$ **do** ▷ Training loop
    broadcast $\boldsymbol{\theta}_{i,n}$
    receive $\{\boldsymbol{\theta}_{j,n}\}_{j \in \mathcal{N}_{i,n}}$
    eq. (2.24)
    $\boldsymbol{\theta}_{i,n} = \mathrm{ModelUpdate}(\boldsymbol{\psi}_{i,n})$
  end for
end procedure
procedure MODELUPDATE($\boldsymbol{\psi}_{i,n}$) ▷ Model opt. step
  compute $F(\boldsymbol{\psi}_{i,n})$ ▷ Forward-pass
  compute $\nabla J_{i,n}(\boldsymbol{\psi}_{i,n})$ ▷ Backward-pass
  $\boldsymbol{\psi}_{i,n} \leftarrow \boldsymbol{\psi}_{i,n} - \eta \nabla J_{i,n}(\boldsymbol{\psi}_{i,n})$ ▷ Local SGD
end procedure

2.4.1.1 Discussion and Contributions

A main drawback of conventional FL algorithms, is that they are synchronous, meaning that the scan of time is regulated by the federated round index n. This can be an issue whenever we are in the presence of heterogeneous agents, i.e., with different local update completion times due to heterogeneous computational capabilities or data. Moreover, in case of low network or agent reliability, the lack of local updates may delay or even stop the FL process (as the PS usually waits for the selected agents' updates). To address the synchronization issues, asynchronous FL can be adopted, where the PS updates the global model at $t = nT_{\mathrm{PS}}$ according to:

$$\boldsymbol{\theta}_{\mathrm{PS},t} = (1 - \epsilon_{\mathrm{FL},t})\boldsymbol{\theta}_{\mathrm{PS},t-T_{\mathrm{PS}}} + \epsilon_{\mathrm{FL},t} \sum_{i \in \mathcal{V}_t} \alpha_{i,\mathcal{V}_t} \boldsymbol{\theta}_{i,t-T_i}, \tag{2.25}$$

where T_{PS} controls the time interval between two updates of the global model, T_i is the time gap needed by the agent i to obtain a local update, and $\boldsymbol{\theta}_{i,t-T_i}$ is the model available at time $t - T_i$ at client i. Given the key importance of carefully adjusting the T_{PS} interval, in Paper [J5], we proposed a T_{PS} policy-optimization that accounts for both different sample distributions among agents, as well as, different computational capabilities.

Another way for handling the heterogeneity of agents, especially under highly non-IID conditions, is to perform a suitable weighting of the local parameters so as to value differently the contribution of each local update in the global model. In PS-based FL, this has been proposed with the FedAdp algorithm [52], where the new aggregation weights $\widetilde{\alpha}_{i,\mathcal{V}_n}$ are computed by taking into account the angle among the gradients of the local and global models as:

$$\beta_{i,n} = \arccos \frac{\left\langle \nabla J(\boldsymbol{\theta}_{\mathrm{PS},n})^\top, \nabla J_i(\boldsymbol{\theta}_{\mathrm{PS},n}) \right\rangle}{\left\| \nabla J(\boldsymbol{\theta}_{\mathrm{PS},n}) \right\|_2 \left\| \nabla J_i(\boldsymbol{\theta}_{\mathrm{PS},n}) \right\|_2}, \tag{2.26}$$

where $\nabla J_i(\boldsymbol{\theta}_{\mathrm{PS},n}) = -(\boldsymbol{\theta}_{i,n} - \boldsymbol{\theta}_{\mathrm{PS},n-1})/\eta$ is the approximated local gradient of agent i, and $\nabla J(\boldsymbol{\theta}_{\mathrm{PS},n}) = \sum_{i \in \mathcal{V}_n} \alpha_{i,\mathcal{V}_n} \nabla J_i(\boldsymbol{\theta}_{\mathrm{PS},n})$ is the global gradient. In principle, the bigger the angle $\beta_{i,n}$ is, the greater the contribution of agent i will be in the global model. Inspired by the adaptive weighting mechanism in PS-based FL and by the WAC schemes

Algorithm 3 Split Learning

procedure SL(η)
  initialize $\boldsymbol{\theta}_{i,0} \quad \forall\, i \in \mathcal{V}$
  for each round $n = 1, \ldots, N_{\mathrm{FL}}$ **do** ▷ Training loop
    for client $i = 1, \ldots, N^{(\mathrm{A})}$ **do** ▷ Run on client i
      compute $F(\boldsymbol{\theta}_{i,n})$ ▷ Client Forward-pass
      send $F(\boldsymbol{\theta}_{i,n})$ to PS
      receive
        $\nabla J_{\mathrm{PS},n}(\boldsymbol{\theta}_{\mathrm{PS},n}) = \mathrm{PSUpdate}(F(\boldsymbol{\theta}_{i,n}))$
      $\boldsymbol{\theta}_{i,n} = \mathrm{ClientUpdate}(\boldsymbol{\theta}_{i,n})$
      send $\boldsymbol{\theta}_{i,n}$ to client $i+1$
    end for
  end for
end procedure
procedure CLIENTUPDATE($\boldsymbol{\theta}_{i,n}$) ▷ Model opt. step
  compute $\nabla J_{i,n}(\boldsymbol{\theta}_{i,n})$ ▷ Client Backward-pass
  $\boldsymbol{\theta}_{i,n} \leftarrow \boldsymbol{\theta}_{i,n} - \eta \nabla J_{i,n}(\boldsymbol{\theta}_{i,n})$ ▷ Local SGD
end procedure
procedure PSUPDATE($F(\boldsymbol{\theta}_{i,n})$) ▷ Model opt. step
  compute $F(\boldsymbol{\theta}_{\mathrm{PS},n})$ ▷ PS Forward-pass
  compute $\nabla J_{\mathrm{PS},n}(\boldsymbol{\theta}_{\mathrm{PS},n})$ ▷ PS Backward-pass
  $\boldsymbol{\theta}_{\mathrm{PS},n} \leftarrow \boldsymbol{\theta}_{\mathrm{PS},n} - \eta \nabla J_{\mathrm{PS},n}(\boldsymbol{\theta}_{\mathrm{PS},n})$ ▷ Local SGD
end procedure

in consensus approaches, in Paper [J6], we propose a decentralized FL algorithm, i.e., CFAdp, that optimizes the neighbors model weighting for improved convergence and accuracy under both sample and label data skewness.

2.4.2 Split Learning

In the simplest SL framework, the model parameters $\boldsymbol{\theta}$ are split into two parts (one for the PS and for the clients), i.e., $\boldsymbol{\theta}_{\mathrm{PS},n}$ and $\boldsymbol{\theta}_{i,n}$. Note that, here, differently from FL, the model structures of $\boldsymbol{\theta}_{\mathrm{PS},n}$ and $\boldsymbol{\theta}_{i,n}$ are distinct. Thus, to complete inference and back-propagation procedures, an exchange of smashed data and gradients must be carried out between PS and clients. The pseudo-code for SL algorithm is provided in Algorithm 3, reporting for simplicity only the synchronization of the learning process in peer-to-peer mode [59]. At the end of the procedure, SL permits to attain identical results to a traditional (i.e., centralized) training procedure, where all layers are available at the same entity, since it involves the same steps and processes (forward propagation and back-propagating gradients), just applied in a different order.

A drawback of the SL procedure is that it must be executed sequentially by each client. To address this issue, SFL algorithms [62, 63, 153] remove the constraint on the sequentiality of inter-client model exchange, performing parallel forward propagation of the client-side models. The PS then executes both the forward propagation and back-propagation on its server-side model using each client's transformed data independently, allowing a high degree of parallelism. Once the gradients are sent back to the clients for their own back-propagation, a step of FedAvg is performed by the PS and by the clients through an additional PS for the federated part, i.e., FPS. We refer to Figure 2.10 for a synthetic representation of the SFL workflow.

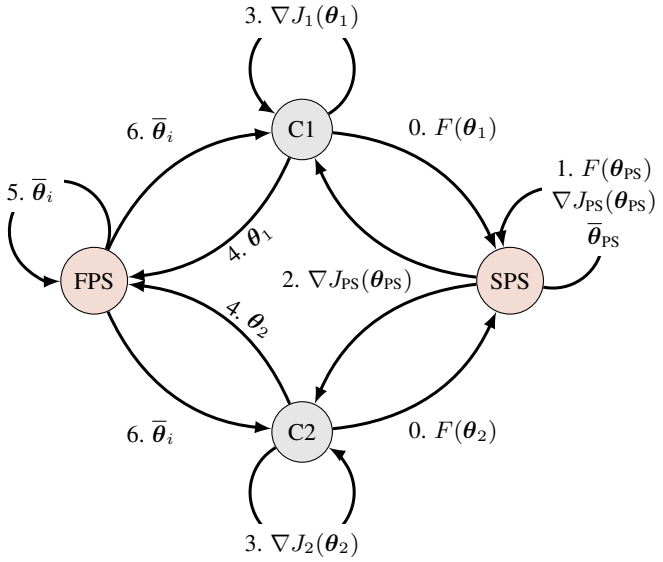


Figure 2.10: *SFL framework with vanilla architecture composed of a split PS, i.e., SPS, and a federated PS, i.e., FPS.*

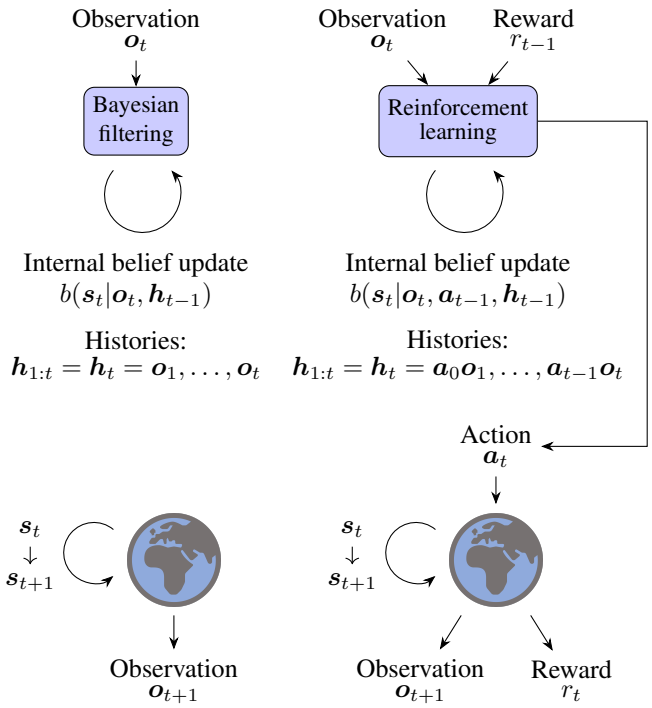
2.4.2.1 Discussion and Contributions

Despite solving the parallelization issues of SL, SFL algorithms still rely on a centralized PS, which can be a single point of failure, as well as a limitation for massive networks. In Paper [J7], we tried to solve this issue by proposing a fully decentralized, privacy-preserving, and low-complexity architecture that performs both parallel training and testing procedures. This architecture, namely SCFL, can be seen as the union between a fully decentralized MPNN and CFA algorithms. Practically, each agent holds a low-complexity model, whose bias is reduced by means of a message passing procedure where the messages are physically exchanged among agents. The system is privacy-preserving since the agents only exchange the smashed data resulting from intermediate message passing steps. Moreover, the training is performed in parallel between agents through a CFA step, that can be included or excluded from the message passing iteration.

2.5 From Learning to Bayesian Filtering

The problem of state learning can be simplified by manually defining the motion and measurement models in (2.1) and (2.3), respectively. This framework, known as Bayesian filtering [154], aims at optimizing the objective in (2.4) without the presence of agents' actions. Therefore, the agents receive just the observations from the environment and the stored histories coincide with the collected observations. We refer to Fig. 2.11 for a comparison between Bayesian filtering and RL frameworks.

In CP use cases, the agents' motion is modelled according to the category of agents, e.g., random walk and constant velocity models for pedestrians and vehicles, respectively. The parameters of the driving noise process in $\mathbf{z}_{\mathbf{s},i,t}^{(\text{A})}$ are tuned according to predetermined noise statistics on collected agents' trajectories. Whenever passive targets are present



(a) Bayesian filtering (b) Reinforcement learning

Figure 2.11: *Comparison between Bayesian filtering and RL.*

in the scene, an ICP approach can be adopted to coherently fuse the measurements at different agents [10]. Indeed, passive entities, such as traffic lights, road signs, or poles, are cooperatively detected by multiple agents and exploited as noisy anchor points to enhance the state (e.g., location) accuracy. In C-ITS scenarios, the vector of all available measurements of agent i at time t is defined as $\mathbf{o}_{i,t} = \left[\mathbf{o}_{i,t}^{(\mathrm{GNSS})^\top} \mathbf{o}_{i,t}^{(\mathrm{A2A})^\top} \mathbf{o}_{i,t}^{(\mathrm{A2T})^\top}\right]^\top$, where $\mathbf{o}_{i,t}^{(\mathrm{GNSS})}$, $\mathbf{o}_{i,t}^{(\mathrm{A2A})}$ and $\mathbf{o}_{i,t}^{(\mathrm{A2T})}$ are the global navigation satellite system (GNSS), A2A and agent-to-target (A2T) observations, respectively. The A2A observations are obtained from an active sensing technology for sidelink positioning offering relative A2A location measurements for any pair of agents $(i, j) \in \mathcal{E}_t$. On the contrary, the A2T observations derive from a passive technology (e.g., camera, light detection and ranging (LIDAR), radio detection and ranging (RADAR), or any combination), used by agent i to detect the set of passive objects $\mathcal{T}_{i,t} \subseteq \mathcal{T}$ in proximity at time t.

By considering the noise statistics as Gaussian and by modelling the agent state $\mathbf{s}_{i,t}^{(\mathrm{A})}$ with the 2D position and velocity and the target state $\mathbf{s}_{k,t}^{(\mathrm{T})}$ with the 2D position, we can define the observations as:

$$\mathbf{o}_{i,t}^{(\mathrm{GNSS})} = \boldsymbol{H}^{(\mathrm{A})}\, \mathbf{s}_{i,t}^{(\mathrm{A})} + \mathbf{z}_{\mathbf{o},i,t}^{(\mathrm{GNSS})}, \tag{2.27}$$

$$\mathbf{o}_{i,j,t}^{(\mathrm{A2A})} = \boldsymbol{H}^{(\mathrm{A})}\left(\mathbf{s}_{i,t}^{(\mathrm{A})} - \mathbf{s}_{j,t}^{(\mathrm{A})}\right) + \mathbf{z}_{\mathbf{o},i,j,t}^{(\mathrm{A2A})}, \tag{2.28}$$

$$\mathbf{o}_{i,k,t}^{(\mathrm{A2T})} = \boldsymbol{H}^{(\mathrm{A})}\mathbf{s}_{i,t}^{(\mathrm{A})} - \mathbf{s}_{k,t}^{(\mathrm{T})} + \mathbf{z}_{\mathbf{o},i,k,t}^{(\mathrm{A2T})}, \tag{2.29}$$

where $\boldsymbol{H}^{(\mathrm{A})} = [\boldsymbol{I}_2\, \boldsymbol{0}_{2\times 2}] \in \mathbb{R}^{2\times 4}$ and $\mathbf{z}_{\mathbf{o},i,t}^{(\mathrm{GNSS})}$, $\mathbf{z}_{\mathbf{o},i,j,t}^{(\mathrm{A2A})}$ and $\mathbf{z}_{\mathbf{o},i,k,t}^{(\mathrm{A2T})}$ are zero-mean Gaussian with covariances $\boldsymbol{R}_{\mathbf{o},i,t}^{(\mathrm{GNSS})} = {\sigma^{(\mathrm{GNSS})}}^2 \boldsymbol{I}_2$, $\boldsymbol{R}_{\mathbf{o},i,j,t}^{(\mathrm{A2A})} = {\sigma^{(\mathrm{A2A})}}^2 \boldsymbol{I}_2$ and $\boldsymbol{R}_{\mathbf{o},i,k,t}^{(\mathrm{A2T})} = {\sigma^{(\mathrm{A2T})}}^2 \boldsymbol{I}_2$, respectively. Note that the choice of Gaussianity is completely arbitrary and that, as for the driving noise statistics, the measurements' covariances need to be tuned from collected observations.

The overall state estimate $\widehat{\boldsymbol{s}}_t$ is obtained as in RL through the MMSE estimator as:

$$\widehat{\boldsymbol{s}}_t = \mathbb{E}\{\mathbf{s}_t|\mathbf{h}_{1:t}\} = \mathbb{E}\{\mathbf{s}_t|\mathbf{o}_{1:t}\} = \int \boldsymbol{s}_t\, b(\boldsymbol{s}_t|\boldsymbol{o}_{1:t})\, d\boldsymbol{s}_t\,, \tag{2.30}$$

where the system belief $b(\boldsymbol{s}_t|\boldsymbol{o}_{1:t})$, known as joint posterior PDF, is computed at each time t from a *prediction phase* with the Chapman-Kolmogorov equation:

$$b(\boldsymbol{s}_t|\boldsymbol{o}_{1:t-1}) = \int p(\boldsymbol{s}_t|\boldsymbol{s}_{t-1})\, b(\boldsymbol{s}_{t-1}|\boldsymbol{o}_{1:t-1})\, d\boldsymbol{s}_{t-1}\,, \tag{2.31}$$

and an *update phase* [155]:

$$b(\boldsymbol{s}_t|\boldsymbol{o}_{1:t}) \propto p(\boldsymbol{o}_t|\boldsymbol{s}_t) b(\boldsymbol{s}_t|\boldsymbol{o}_{1:t-1})\,. \tag{2.32}$$

Since, as assumed in Chap. 2.1, the system is observation-independent, the likelihood function of $\boldsymbol{s}_t$ is computed as:

$$\begin{aligned} p\Big(\boldsymbol{o}_t|\boldsymbol{s}_t\Big) = p\Big(\boldsymbol{o}_t^{(\mathrm{GNSS})}|\boldsymbol{s}_t^{(\mathrm{A})}\Big) \prod_{i=1}^{N^{(\mathrm{A})}} \prod_{j\in\mathcal{N}_{i,t}} p\Big(\boldsymbol{o}_{i,j,t}^{(\mathrm{A2A})}|\boldsymbol{s}_{i,t}^{(\mathrm{A})}, \boldsymbol{s}_{j,t}^{(\mathrm{A})}\Big) \\ \times \prod_{i=1}^{N^{(\mathrm{A})}} \prod_{k\in\mathcal{T}_{i,t}} p\Big(\boldsymbol{o}_{i,k,t}^{(\mathrm{A2T})}|\boldsymbol{s}_{i,t}^{(\mathrm{A})}, \boldsymbol{s}_{k,t}^{(\mathrm{T})}\Big)\,. \end{aligned} \tag{2.33}$$

In case the motion and measurements models in (2.1) and (2.3), respectively, are linear and with a Gaussian noise, the state estimate in (2.30) reduces to a KF as described in [10, 128], with efficient resolution in matrix form.

However, the centralized Bayesian filtering approach is impractical for extensive networks due to two major limitations: the dependence on a single central computing unit creates a potential point of failure, and the computational complexity increases cubically with the number of agents and passive objects [10]. To overcome such limitations, distributed or consensus-based Bayesian filtering algorithms have been studied in the past [43, 156]. Since the marginalization of the joint posterior PDF $b(\boldsymbol{s}_t|\boldsymbol{o}_{1:t})$ to compute the agents' beliefs $b\Big(\boldsymbol{s}_{i,t}^{(\mathrm{A})}|\boldsymbol{o}_{1:t}\Big)$ can be unfeasible or extremely complex, the distributed Bayesian filtering adopts a MPA. This approach approximates the agents' beliefs with an iterative message passing procedure over a factor graph which factorizes the joint posterior PDF. As an example, indicating the beliefs of agent i at timestep t and message passing iteration $p \in \{1, \ldots, N_{\mathrm{MP}}\}$ with $b_{i,t}^{(p)} = b^{(p)}\Big(\boldsymbol{s}_{i,t}^{(\mathrm{A})}|\boldsymbol{o}_{1:t}\Big)$, the MPA-based CP (without the presence of targets) executes the subsequent steps in parallel for each agent.

1. *Prediction message*: The message representing the predicted state of agent i:

$$\mu_{i,\overrightarrow{t}}\Big(\boldsymbol{s}_{i,t}^{(\mathrm{A})}\Big) \propto \int p(\boldsymbol{s}_{i,t}^{(\mathrm{A})}|\boldsymbol{s}_{i,t-1}^{(\mathrm{A})})\, b_{i,t-1}^{(N_{\mathrm{MP}})} d\boldsymbol{s}_{i,t-1}^{(\mathrm{A})}\,, \tag{2.34}$$

where $b_{i,t-1}^{(N_{\mathrm{MP}})}$ is the belief estimated by agent i at previous time $t-1$ after N_{MP} message passing steps. We mention that the beliefs are instantiated at time $t=0$ as $b_{i,0}^{(N_{\mathrm{MP}})} \triangleq p\Big(\boldsymbol{s}_{i,0}^{(\mathrm{A})}\Big)$.

2. *Beliefs exchange*: At iteration $p \in \{1,\dots,N_{\mathrm{MP}}\}$, each agent i transmits $b_{i,t}^{(p-1)}$ and receives $b_{j,t}^{(p-1)}$ from its neighbors $j \in \mathcal{N}_{i,n}$. At $p=1$, the shared beliefs are $b_{i,t}^{(0)} = \mu_{i,\overrightarrow{t}}\Big(\boldsymbol{s}_{i,t}^{(\mathrm{A})}\Big)$.

3. *Measurement messages computation*: During message passing iteration $p \in \{1,\dots,N_{\mathrm{MP}}\}$, each agent i computes one message for each observation typology:

$$\mu_{i,t}^{(p)(\mathrm{GNSS})}\Big(\boldsymbol{s}_{i,t}^{(\mathrm{A})}\Big) \triangleq p\Big(\boldsymbol{o}_{i,t}^{(\mathrm{GNSS})}|\boldsymbol{s}_{i,t}^{(\mathrm{A})}\Big), \tag{2.35}$$

$$\mu_{j\to i,t}^{(p)(\mathrm{A2A})}\Big(\boldsymbol{s}_{i,t}^{(\mathrm{A})}\Big) \propto \int p\Big(\boldsymbol{o}_{j,i,t}^{(\mathrm{A2A})}|\boldsymbol{s}_{j,t}^{(\mathrm{A})},\boldsymbol{s}_{i,t}^{(\mathrm{A})}\Big)\, b_{j,t}^{(p-1)} d\boldsymbol{s}_{j,t}^{(\mathrm{A})} \quad \forall j \in \mathcal{N}_{i,n}. \tag{2.36}$$

4. *Beliefs update*: At message passing iteration $p \in \{1,\dots,N_{\mathrm{MP}}\}$, the beliefs are updated as:

$$b_{i,t}^{(p)} \propto \mu_{i,\overrightarrow{t}}\Big(\boldsymbol{s}_{i,t}^{(\mathrm{A})}\Big)\, \mu_{i,t}^{(p)(\mathrm{GNSS})}\Big(\boldsymbol{s}_{i,t}^{(\mathrm{A})}\Big) \prod_{j\in\mathcal{N}_{i,n}} \mu_{j\to i,t}^{(p)(\mathrm{A2A})}\Big(\boldsymbol{s}_{i,t}^{(\mathrm{A})}\Big). \tag{2.37}$$

5. *State inference*: At last, after completing N_{MP} message passing steps, the agent state is determined using the MMSE estimator as:

$$\widehat{\boldsymbol{s}}_{i,t}^{(\mathrm{A})} = \mathbb{E}_{\mathbf{s}_{i,t}\sim b_{i,t}^{(p)}}\Big\{\mathbf{s}_{i,t}^{(\mathrm{A})}|\mathbf{o}_{1:t}\Big\}. \tag{2.38}$$

Step 1), known as the *prediction step*, is executed once per timestep t. In contrast, steps 2), 3), and 4), referred to as *update steps*, use the measurements obtained from the current timestep and they are carried out during all N_{MP} message passing iterations for each timestep t.

2.5.1 Discussion and Contributions

Although MPA is scalable, it has a limitation in that convergence to the centralized solution is only assured in acyclic (i.e., tree-structured) factor graphs. Moreover, even in case of convergence, the result would be optimal only with Gaussian and linear models (i.e., in (2.1) and (2.3)). In all the other cases, optimality is not guaranteed. In Fig. 2.12 we summarized all cases and highlighted those where improvements could be provided by new data-driven designs. We would like to point out that in real-world dynamics, the factor graph is usually not acyclic and the models are typically neither Gaussian nor linear. For improving upon MPA in conditions where the optimality is not guaranteed, in Papers [J2] and [J3], we proposed MPNN-based solutions for the specific tasks of DA and CP, respectively. On the contrary, for ICP frameworks, in Papers [C8] and [J9], we proposed a novel MARL algorithm that generalizes the Bayesian-filtering approach by including actions of agents for de/activating communications links.

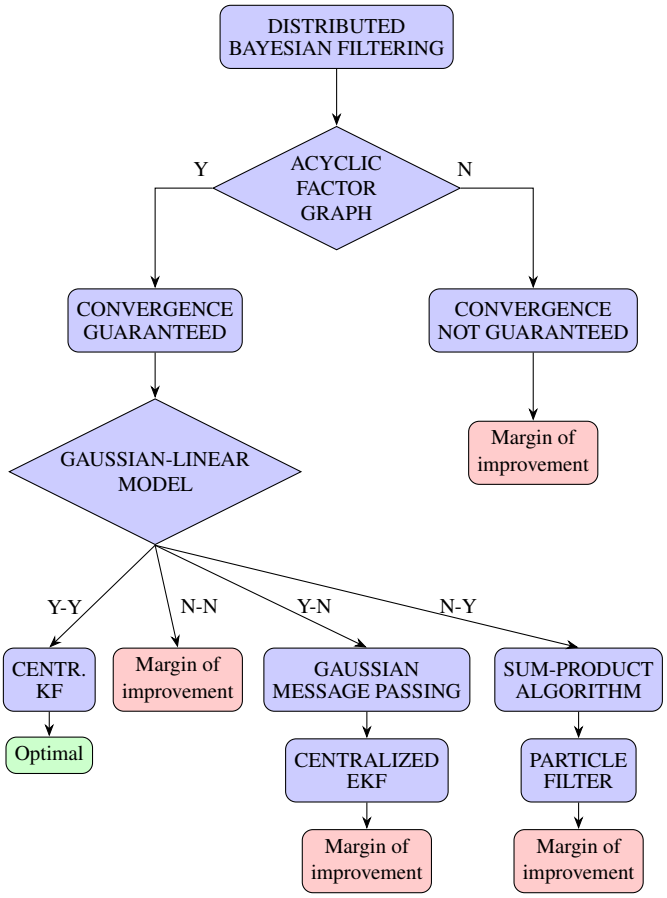
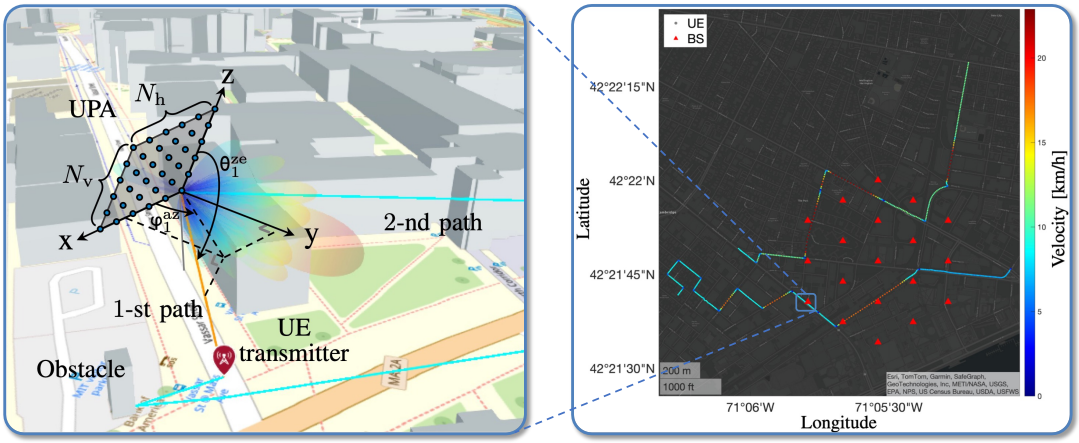


Figure 2.12: *Convergence diagram flow of distributed Bayesian filtering methods.*

CHAPTER 3

Inference in Agent Networks

In this chapter, we summarize the main concepts and methodologies adopted for performing cooperative inference in agent networks. The application focuses on next-generation cellular networks (see Fig. 1.1d), where the agents are represented by BSs. We start by describing, in Sec. 3.1, the system model adopted between the BSs and the UE. Then, in Sec. 3.2, we report the extraction of the proposed fingerprinting measurement for performing positioning, whereas in Sec. 3.3 we describe the main DL model for channel compression into a latent features representation. Finally, in Sec. 3.4, we discuss the main concept of VI and its application to the anomaly detection task for NLoS identification.



(a) Uplink scenario

(b) UE trajectories

Figure 3.1: *Representation of the cooperative inference system model: (a) BS receiving the uplink signal from the UE through an UPA, with highlighted AoA of the 1-st path composed by the zenith angle θ_1^{ze} and the azimuth angle φ_1^{az}, and (b) example of two UE trajectories in the area of Cambridge, MA, USA with red triangles indicating the BS positions.*

3.1 System Model

We consider a mmWave OFDM system where a UE transmits to BS at a carrier wavelength λ_c. The BS is fitted with a uniform planar array (UPA) consisting of $N_v \times N_h$ isotropic antenna elements, in the vertical and horizontal direction, respectively, with vertical and horizontal antenna spacings of d_v and d_h, respectively. In contrast, the UE is provided with a single omni-directional antenna. The channel involves N_p distinct propagation paths, where each path $p = 1, \dots, N_p$ is characterized by an AoA, specified by a zenith angle $\theta_p^{ze} \in [0, \pi]$ and an azimuth angle $\varphi_p^{az} \in [0, \pi]$, and by a ToF τ_p. The scenario is represented in Fig. 3.1 where we reported the UPA at the BS receiving two paths (one direct and one reflected) (a) and the corresponding BSs locations and UE trajectories (b). In the example, the BSs disposition has been chosen according to the 3GPP urban micro (UMi) scenario [157] within a 1000×1000 m area near the MIT campus in Cambridge, MA, USA. The setting includes 19 sites, each with an inter-site distance (ISD) of 200 m, forming a hexagonal layout. Each site comprises 3 BSs, elevated 25 m and angled 120 degrees apart.

The system adopts an OFDM scheme with a sampling interval of length T_s, N_c sub-carriers, and an OFDM symbol duration of $T_c = N_c T_s$. Considering baseband formulation, the frequency at the k-th sub-carrier is $f_k = \frac{k}{T_c}$, $k = 0, \dots, N_c - 1$. We indicate with N_g the number of sampling intervals constituting a guard interval and we assume that the maximum channel delay τ_{MAX} is lower than the cyclic-prefix duration $T_g = N_g T_s$.

By considering a sampling rate of $1/T_s$ and treating each path as wide-sense stationary and independent [158], we can express the CFR at the k-th sub-carrier as [159, 160]:

$$\mathbf{h}_k^{ch} = \sum_{p=1}^{N_p} \bar{\alpha}_{k,p}\, \mathbf{e}(\theta_p^{ze}, \varphi_p^{az}) \in \mathbb{C}^{N_h N_v}, \tag{3.1}$$

$\mathbf{e}(\theta_p^{\mathrm{ze}}, \varphi_p^{\mathrm{az}}) \in \mathbb{C}^{N_{\mathrm{h}} N_{\mathrm{v}}}$ is the array response vector [158], and $\bar{\alpha}_{k,p} = \alpha_p e^{-j2\pi\tau_p f_k}$ are the frequency channel gains, with $\alpha_p = a_p e^{-j2\pi(\frac{d_p}{\lambda_c} - \nu_p \tau_p)}$ being the complex gain of p-th path which includes the Doppler frequency shift ν_p and has average power $\sigma_p^2 = \mathbb{E}\{\|a_p\|^2\}$. The traveled distance of the p-th path is indicated with $d_p = c\,\tau_p$, where c is the speed of light in air. By grouping the CFRs at every sub-carrier, we recover the space-frequency channel response matrix (SFCRM):

$$\mathbf{H}^{\mathrm{ch}} = [\mathbf{h}_0^{\mathrm{ch}}\ \mathbf{h}_1^{\mathrm{ch}}\ \ldots\ \mathbf{h}_{N_{\mathrm{c}}-1}^{\mathrm{ch}}] \in \mathbb{C}^{N_{\mathrm{h}} N_{\mathrm{v}} \times N_{\mathrm{c}}}. \tag{3.2}$$

In the following, we show how to extract the ADCPM fingerprint from the SFCRM obtained at the BS.

3.2 Location-dependent Fingerprint

Mapping the channel response into the angle-delay domain is beneficial for location estimation as it simplifies the process of identifying macro-paths, which include clusters of both LoS and NLoS components. These clusters vary based on the surrounding environment and act as distinctive location-specific features or fingerprints. Furthermore, the estimation's robustness and accuracy are enhanced by increasing the number of antenna elements and the bandwidth, which improves resolution in both the spatial and frequency dimensions. We obtain the angle-delay domain features by defining the phase-shifted discrete Fourier transform (DFT) matrices $\boldsymbol{V}_{N_{\mathrm{h}}} \in \mathbb{C}^{N_{\mathrm{h}} \times N_{\mathrm{h}}}$ and $\boldsymbol{V}_{N_{\mathrm{v}}} \in \mathbb{C}^{N_{\mathrm{v}} \times N_{\mathrm{v}}}$, where $[\boldsymbol{V}_{N_{\mathrm{h}}}]_{\bar{i},\bar{j}} = \frac{1}{\sqrt{N_{\mathrm{h}}}} e^{-j2\pi \frac{\bar{i}(\bar{j} - \frac{N_{\mathrm{h}}}{2})}{N_{\mathrm{h}}}}$ and $[\boldsymbol{V}_{N_{\mathrm{v}}}]_{\bar{i},\bar{j}} = \frac{1}{\sqrt{N_{\mathrm{v}}}} e^{-j2\pi \frac{\bar{i}(\bar{j} - \frac{N_{\mathrm{v}}}{2})}{N_{\mathrm{v}}}}$, and the unitary DFT matrix $\boldsymbol{F} \in \mathbb{C}^{N_{\mathrm{c}} \times N_{\mathrm{g}}}$, with $[\boldsymbol{F}]_{\bar{i},\bar{j}} = \frac{1}{\sqrt{N_{\mathrm{c}}}} e^{-j2\pi \frac{\bar{i}\bar{j}}{N_{\mathrm{c}}}}$. We can therefore obtain the angle-delay channel response matrix (ADCRM) as [102]:

$$\mathbf{G} = \frac{1}{\sqrt{N_{\mathrm{h}} N_{\mathrm{v}} N_{\mathrm{c}}}} (\boldsymbol{V}_{N_{\mathrm{h}}}^H \otimes \boldsymbol{V}_{N_{\mathrm{v}}}^H)\, \mathbf{H}^{\mathrm{ch}}\, \boldsymbol{F}^* \in \mathbb{C}^{N_{\mathrm{h}} N_{\mathrm{v}} \times N_{\mathrm{g}}}, \tag{3.3}$$

where $(\boldsymbol{V}_{N_{\mathrm{h}}}^H \otimes \boldsymbol{V}_{N_{\mathrm{v}}}^H)$ and $\boldsymbol{F}^*$ project the SFCRM into the angle and delay domain, respectively.

Finally, the ADCPM is computed from the ADCRM as:

$$\boldsymbol{P} = \mathbb{E}\{\mathbf{G} \odot \mathbf{G}^*\} \in \mathbb{R}^{N_{\mathrm{h}} N_{\mathrm{v}} \times N_{\mathrm{g}}}, \tag{3.4}$$

where $[\boldsymbol{P}]_{\bar{i},\bar{j}} = \mathbb{E}\left\{ \left\| [\mathbf{G}]_{\bar{i},\bar{j}} \right\|^2 \right\}$. We would like to point out that whenever the angle-delay resolutions is very high, in the limit for N_{g}, N_{v} and $N_{\mathrm{h}} \to \infty$, the ADCPM becomes sparse and its elements $[\boldsymbol{P}]_{\bar{i},\bar{j}}$ match the average channel power of the $\bar{i}$-th AoA and the $\bar{j}$-th ToF as [102]:

$$\lim_{N_{\mathrm{h}}, N_{\mathrm{v}}, N_{\mathrm{g}} \to \infty} [\boldsymbol{P}]_{\bar{i},\bar{j}} = \sum_{p=1}^{N_{\mathrm{p}}} \sigma_p^2\, \delta\left[\bar{i} - m_p N_{\mathrm{v}} - n_p\right] \delta\left[\bar{j} - r_p\right], \tag{3.5}$$

where $m_p = \frac{N_{\mathrm{h}}}{2} + \frac{N_{\mathrm{h}} d_{\mathrm{h}}}{\lambda_c} \sin\theta_p^{\mathrm{ze}} \cos\varphi_p^{\mathrm{az}}$, $n_p = \frac{N_{\mathrm{v}}}{2} + \frac{N_{\mathrm{v}} d_{\mathrm{v}}}{\lambda_c} \cos\theta_p^{\mathrm{ze}}$, and $r_p = \lfloor \frac{\tau_p}{T_s} \rfloor$ is the resolvable delay corresponding to the p-th path. Consequently, the statistical characteristics of the ADCPM enhance the DL model's capability to detect location-specific attributes, offering consistent and reliable fingerprints for accurate location estimation.

3.3 Input and DL Model

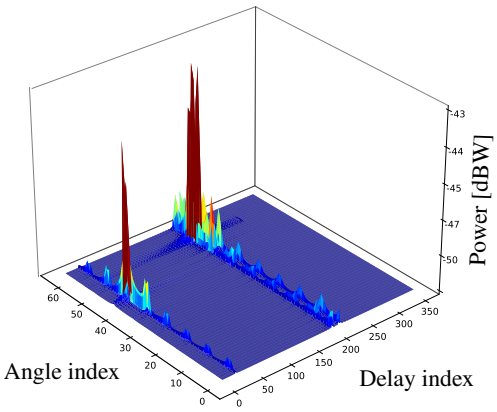
We suggest employing the ADCPM as the fundamental measurement for estimating the UE location. This sparse matrix effectively represents the multipath environment in the power-delay-angle domain, enabling a DL model like a CNN to identify crucial location-centric features. Notably, the initial, often sparse, layers of the CNN are suitable for extracting highly discriminative features from the sparse ADCPM channel matrix [161]. Additionally, the ADCPM encapsulates critical information including RSS, AoA, and ToF for each path, while keeping memory and computational demands minimal due to the channel's sparsity. To represent this characteristic, Fig. 3.2a presents an example of ADCPM with $N_{\mathrm{g}} = 352$ delay samples and $N_{\mathrm{h}}N_{\mathrm{v}} = 64$ angular directions. Figs. 3.2b and 3.2c display the ray-tracing paths along with their azimuth and zenith AoA, which produced the ADCPM fingerprint. Notably, the matrix's sparsity remains clear even without a large array of antennas or high-resolution sampling. In our simulations for inference in next-generation cellular networks, we thus use the ADCPM $\boldsymbol{P}$ as the input $\mathbf{x}$ for the DL model, serving as the basis for positioning and subsequent tracking.

Given the high-dimensionality of the ADCPM input, it is natural to try compressing it to a more compact representation that is absent of redundant information. One of the most popular and effective DL models to perform this operation is the AE, trained to reconstruct the input ADCPM matrix with minimal loss, capturing the most critical features in the compressed domain. The way the AE is trained, i.e., how to tune the parameters to obtain predetermined latent features $\mathbf{h}$, highly relies on the typology of the task (e.g., unsupervised, semi-supervised, and supervised ML). After compressing the input into the latent features $\mathbf{h}$, several downstream tasks become feasible. For instance, we may want to estimate their PDF, assess the likelihood of observing a predetermined input, or use them as input to subsequent models.

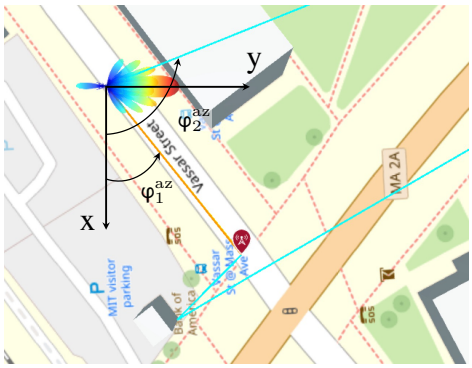
3.4 Efficient Distribution Modelling with Variational Inference

Let's define the encoder and decoder parts as $p_{\boldsymbol{\phi}}(\boldsymbol{h}|\boldsymbol{x})$ and $p_{\boldsymbol{\theta}}(\boldsymbol{x}|\boldsymbol{h})$, parameterized by the non-random parameters $\boldsymbol{\phi}$ and $\boldsymbol{\theta}$, respectively. The objective is to approximate an intractable posterior PDF $p_{\boldsymbol{\theta}}(\boldsymbol{h}|\boldsymbol{x})$, from which the latent features are sampled given an input $\boldsymbol{x}$, with the variational approximation $p_{\boldsymbol{\phi}}(\boldsymbol{h}|\boldsymbol{x})$. For this reason, this method is called VI. We make the hypothesis that the latent features $\boldsymbol{h}$ have a prior distribution $p_{\boldsymbol{\theta}}(\boldsymbol{h})$, also considered intractable, and that the PDFs $p_{\boldsymbol{\theta}}(\boldsymbol{h}|\boldsymbol{x})$ and $p_{\boldsymbol{\theta}}(\boldsymbol{h})$ are differentiable with respect to $\boldsymbol{h}$. To approximate the intractable posterior PDF $p_{\boldsymbol{\theta}}(\boldsymbol{h}|\boldsymbol{x})$, we estimate the parameters $\boldsymbol{\theta}$ and $\boldsymbol{\phi}$ by MLE. Given an input $\boldsymbol{x}$, the log-likelihood function of the parameters can be expressed as:

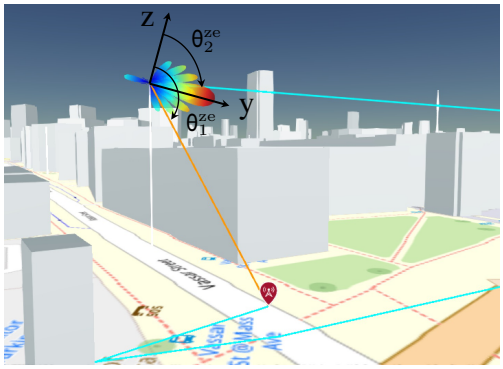
$$\begin{aligned}
\log p_{\boldsymbol{\theta}}(\boldsymbol{x}) &= \log \int p_{\boldsymbol{\theta}}(\boldsymbol{h}, \boldsymbol{x}) d\boldsymbol{h} \\
&= \log \int p_{\boldsymbol{\phi}}(\boldsymbol{h}|\boldsymbol{x}) \frac{p_{\boldsymbol{\theta}}(\boldsymbol{h}, \boldsymbol{x})}{p_{\boldsymbol{\phi}}(\boldsymbol{h}|\boldsymbol{x})} d\boldsymbol{h} \\
&\geq \int p_{\boldsymbol{\phi}}(\boldsymbol{h}|\boldsymbol{x}) \log \frac{p_{\boldsymbol{\theta}}(\boldsymbol{h}) p_{\boldsymbol{\theta}}(\boldsymbol{x}|\boldsymbol{h})}{p_{\boldsymbol{\phi}}(\boldsymbol{h}|\boldsymbol{x})} d\boldsymbol{h}
\end{aligned} \tag{3.6}$$



(a) ADCPM channel input



(b) Azimuth AoA



(c) Zenith AoA

Figure 3.2: *(a) Example of the ADCPM fingerprint with $N_{\mathrm{h}}N_{\mathrm{v}} = 64$ spatial samples (i.e., angle indexes) and $N_{\mathrm{g}} = 352$ temporal samples (i.e., delay indexes). The corresponding azimuth and zenith angles for each of the two main clusters of arrives are represented in (b) and (c), respectively.*

$$= \mathbb{E}_{\boldsymbol{h}\sim p_{\boldsymbol{\phi}}(\boldsymbol{h}|\boldsymbol{x})}\Big\{\log p_{\boldsymbol{\theta}}(\boldsymbol{x}|\boldsymbol{h})\Big\} - \mathrm{KL}\Big(p_{\boldsymbol{\phi}}(\boldsymbol{h}|\boldsymbol{x}) \,\Big\|\, p_{\boldsymbol{\theta}}(\boldsymbol{h})\Big), \tag{3.7}$$

where the inequalities in (3.6) derives from the Jensen's inequality and $\mathrm{KL}(\cdot\|\cdot)$ is the Kullback-Leibler (KL) divergence. It should be noted that the first term in (3.7) represents the expected log-likelihood, which encourages the model to accurately reconstruct the input samples from the latent representation. Conversely, the KL divergence term serves as a regularizer, pushing the learned latent distribution to approximate the prior distribution. Equation (3.7) is used as objective function to be maximized in VAE methods [162], where the aim is to learn a compressed representation of the latent features by imposing a Gaussian prior PDF in $p_{\boldsymbol{\theta}}(\boldsymbol{h})$. This is very useful for anomaly detection tasks, where we want to learn the most precise representation of normal samples, such as to distinguish them from anomalous samples. However, the main drawback with VAE is that they approximate the integral in (3.6) by using MC sampling, both in the training and testing phases. Thus, they have important limitations in real-time applications.

To avoid explicit sampling, many works adopted a conventional AE for compressing the input and then a probability density estimate to approximate the real latent feature distribution. An example is the KDE method [163], that, given a set of training samples

$\{\boldsymbol{h}_j\}_{j=1}^{N_\mathrm{s}}$ from $p(\boldsymbol{h})$ and a test sample $\boldsymbol{h}_i$, returns the probability density estimate as:

$$K\left(\boldsymbol{h}_i|\{\boldsymbol{h}_j\}_{j=1}^{N_\mathrm{s}}\right) = \frac{1}{N_\mathrm{s}}\sum_{j=1}^{N_\mathrm{s}} k_{bw}(\boldsymbol{h}_i - \boldsymbol{h}_j), \tag{3.8}$$

where $k_{bw} : \mathbb{R}^m \to \mathbb{R}$ is a kernel function with bandwidth bw, with m being the latent dimension. However, the main issue with an AE-KDE architecture is that we need to store all the training datasets (i.e., related latent feature samples) to estimate the density function in the inference phase. An alternative, employed in [91], is to adopt a DAGMM to simultaneously optimize the AE weights, and the latent distribution by means of a GMM, whose parameters (i.e., means $\widehat{\boldsymbol{\mu}}_k$, covariances $\widehat{\boldsymbol{\Sigma}}_k$, and mixture coefficients $\widehat{\phi}_k$) are computed with an estimation NN. The estimation network is trained by minimizing the sample energy, i.e., negative log-likelihood as:

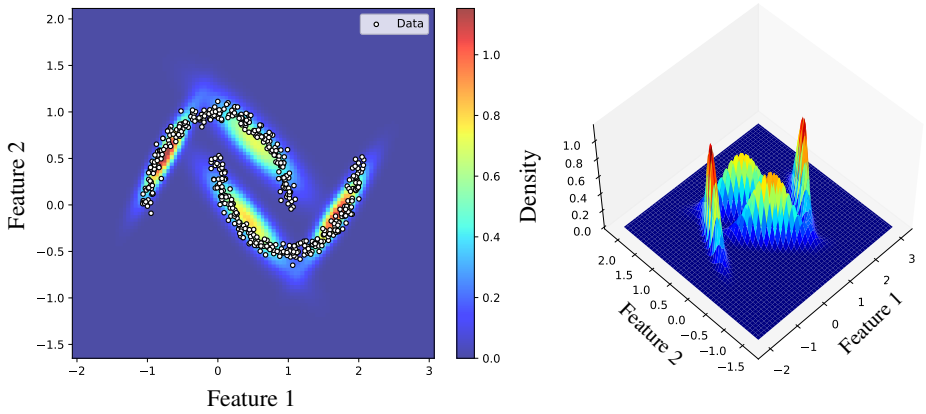
$$J_\mathrm{E}(\boldsymbol{h}) = -\log\left(\sum_{k=1}^{N_\mathrm{K}} \widehat{\phi}_k \exp\left(-\frac{1}{2}(\boldsymbol{h}-\widehat{\boldsymbol{\mu}}_k)^\top \widehat{\boldsymbol{\Sigma}}_k^{-1}(\boldsymbol{h}-\widehat{\boldsymbol{\mu}}_k)\right)\frac{1}{\sqrt{\det\left(2\pi\widehat{\boldsymbol{\Sigma}}_k\right)}}\right), \tag{3.9}$$

where N_K is the number of components. With the GMM we are able to approximate the latent distribution with the most important components, thus avoiding an inefficient storing of training samples.

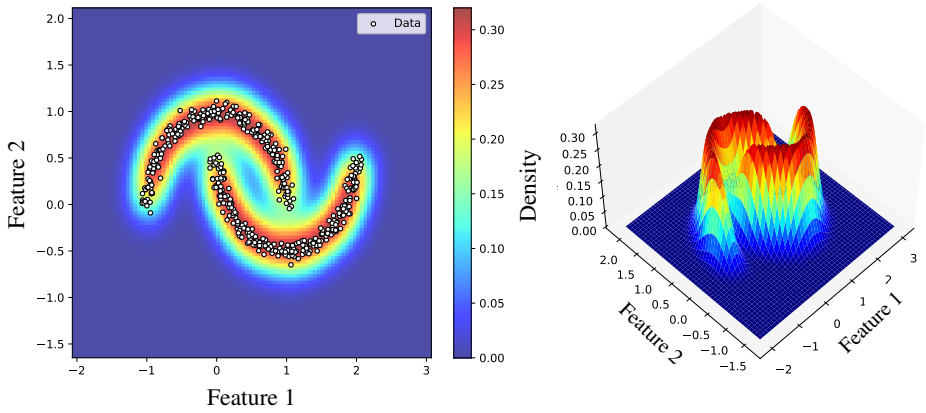
3.4.1 Discussion and Contributions

Even though the GMM approximation is simple and efficient, it may not be suitable to approximate complex distributions that exhibit multimodality, heavy tails, or intricate structures, as the KDE can. For example, when dealing with high-dimensional data or distributions with a large number of modes that are not well-separated, GMMs may require an impractically large number of components to achieve a good approximation, which increases computational complexity [164]. Moreover, GMMs assume that each component follows a Gaussian distribution, which may not capture the true nature of the underlying data if it significantly deviates from Gaussianity [165]. In contrast, KDE provides an estimation that adapts to the actual data distribution without assuming a specific parametric form [163]. Fig. 3.3 illustrates this comparison, highlighting how KDE can capture more intricate structures in data. For example, as noted in [166], real-world data may lack a clear predefined distribution and exhibit arbitrary patterns in latent space, posing challenges for GMM-based methods. Additionally, GMMs often require manual parameter adjustments when modeling the density distribution of input data, which can significantly impact detection performance. To show an example of realistic high-dimensional data, in Fig. 3.4, we report the t-distributed stochastic neighbor embedding (t-SNE) visualization of the ADCPM fingerprint and the related density estimation. Note that despite using many components (12 in this case), the GMM is not able to approximate precisely shapes that are not strictly Gaussian or that exhibit complex structures.

Despite the authors in [91] did not employed a VI-based objective function for training DAGMM, the negative log-likelihood function of a latent sample $\boldsymbol{h}$ can be



(a) GMM density estimate



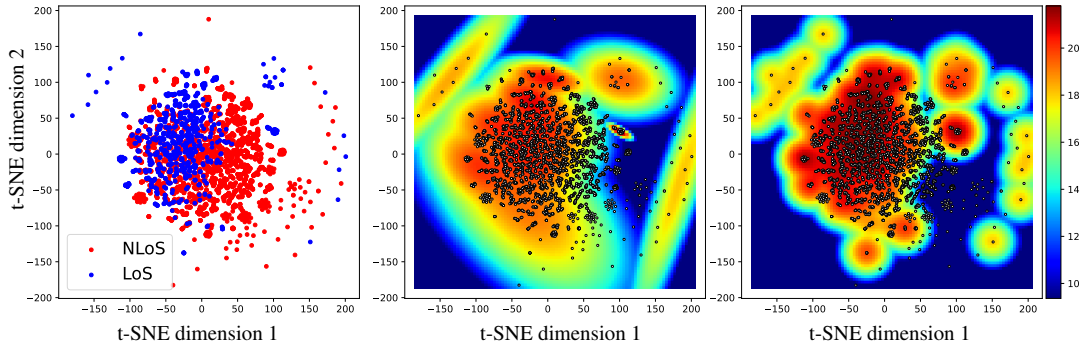
(b) KDE density estimate

Figure 3.3: *Comparison between (a) the GMM and (b) the KDE techniques for estimating the density of a two-dimensional dataset.*

written as:

$$
\begin{aligned}
-\log p_{\boldsymbol{\theta}}(\boldsymbol{h}) &= -\log \sum_{k=1}^{N_{\mathrm{K}}} p_{\boldsymbol{\theta}}(k, \boldsymbol{h}) \\
&= -\log \sum_{k=1}^{N_{\mathrm{K}}} p_{\boldsymbol{\phi}}(k|\boldsymbol{h}) \, \frac{p_{\boldsymbol{\theta}}(k, \boldsymbol{h})}{p_{\boldsymbol{\phi}}(k|\boldsymbol{h})} \\
&\leq -\sum_{k=1}^{N_{\mathrm{K}}} p_{\boldsymbol{\phi}}(k|\boldsymbol{h}) \log \frac{p_{\boldsymbol{\theta}}(\boldsymbol{h}) p_{\boldsymbol{\theta}}(k|\boldsymbol{h})}{p_{\boldsymbol{\phi}}(k|\boldsymbol{h})} \\
&\backsimeq J_{\mathrm{E}}(\boldsymbol{h}) + \mathrm{KL}\Big(p_{\boldsymbol{\phi}}(k|\boldsymbol{h}) \,\Big\|\, p_{\boldsymbol{\theta}}(k|\boldsymbol{h})\Big),
\end{aligned}
\tag{3.10}
$$

where the approximation in (3.10) comes from $J_{\mathrm{E}}(\boldsymbol{h}) \backsimeq \mathbb{E}_{k \sim p_{\boldsymbol{\phi}}(k|\boldsymbol{h})}\Big\{-\log p_{\boldsymbol{\theta}}(\boldsymbol{h}|k)\Big\}$, and where $p_{\boldsymbol{\phi}}(k|\boldsymbol{h})$ is the estimation NN that predicts the parameters of the k-th Gaussian component. Note the similarity between (3.7) and (3.10), where the input sample $\boldsymbol{x}$ now becomes the latent feature $\boldsymbol{h}$. Inspired by (3.10) and to improve the GMM approximation, in Paper [J10], we propose an AE-based model for NLoS identification



(a) t-SNE (b) GMM density estimate (c) KDE density estimate

Figure 3.4: *(a) 2D visualization of the ADCPM samples (LoS and NLoS) obtained with t-SNE dimensionality reduction method. (b) and (c) are the corresponding density estimations for the LoS distribution with GMM and KDE algorithms, respectively.*

where the channel latent features are learned through a KDE only at training time and just exploiting the previous mini-batch, as opposed to the whole dataset. The estimation NN of DAGMM is replaced by a likelihood NN which directly approximates the KDE's output. At inference time, the decoder, as well as the KDE, are discarded, leaving the anomaly score prediction to the likelihood NN.

Part II

Cooperative Learning

CHAPTER 4

Graph-aware Learning

In this chapter, we present two works that perform cooperative learning directly on graphs. Specifically, within the framework of vehicular networks, we propose the integration and employment of MPNN into C-ML systems for the tasks of DA and CP. In the first paper, the objective is to associate measurements obtained with lidar object detection models at different vehicles. To tackle the problem, we present a logical graph mapping based on the measurements, i.e., detections, and the vehicles' positions. We then associate different measurements by means of a modified MPNN model trained on edge-classification task on the logical graph. Finally, we compare the performances of the proposed approach with MPA-based algorithms under different noise conditions and graph dimensions. In the second paper, the objective becomes performing CP in a distributed network of agents that move in the space and gather state and inter-agent measurements at each timestep. Inspired by the MPA for CP, we propose a two-block model, namely LSTM-MPNN, that employs the LSTM for next state prediction and the MPNN for measurements processing and state update. The results indicate that the developed LSTM-MPNN model surpasses the corresponding MPA in CP in both complexity and accuracy. This improvement is achieved by directly learning the state-transition PDF and the distributed state-update from trajectory data.

Cooperative Lidar Sensing for Pedestrian Detection: Data Association Based on Message Passing Neural Networks

Bernardo Camajori Tedeschini, *Graduate Student Member, IEEE*, Mattia Brambilla, *Member, IEEE*, Luca Barbieri, *Member, IEEE*, Gabriele Balducci, and Monica Nicoli, *Senior Member, IEEE*

***Abstract*—This paper considers the problem of cooperative lidar sensing in vehicular networks. We focus on the task of associating the vehicle-generated measurements by lidars to enable a cooperative detection of vulnerable road users. The considered measurements are the three-dimensional bounding boxes extracted from the lidar point cloud. Focusing on a centralized architecture which aggregates and processes all the sensing information, we design a graph formulation of the association problem and we propose a novel solution based on Message Passing Neural Networks (MPNNs). The method has the advantage of accurately learning the associations and the measurement statistics directly from data. We validate the proposed approach on a cooperative sensing scenario simulated by CARLA, an open-source high-fidelity simulator for automated driving scenarios. For the generation of bounding boxes related to pedestrian detections, we consider both artificially-generated and realistic measurements obtained by employing the PointPillars model. We validate the performance by comparing the proposed MPNN model with the Sum-Product Algorithm for Data Association (SPADA), a common approach for data association in multisensor systems. The proposed data-driven MPNN model achieves an association accuracy above 99% and outperforms SPADA in case of moderate sensing errors, as foreseen by automated driving scenarios. We also assess the efficacy of data association in case of mis-modeling between training and testing datasets, observing good generalization capabilities when dealing with untrained conditions.**

***Index Terms*—Cooperative lidar, pedestrian detection, data association, MPNN, SPADA, CARLA simulator.**

Manuscript received 8 November 2022; revised 17 April 2023 and 31 July 2023; accepted 7 August 2023. Date of publication 14 August 2023; date of current version 5 September 2023. This work was supported by the project Centro Nazionale per la Mobilità Sostenibile (MOST), funded by the Italian Ministry of University and Research under the PNRR funding program. The associate editor coordinating the review of this manuscript and approving it for publication was Prof. Mojtaba Soltanalian. *(Corresponding author: Bernardo Camajori Tedeschini.)*

Bernardo Camajori Tedeschini, Mattia Brambilla, Luca Barbieri, and Gabriele Balducci are with the Dipartimento di Elettronica, Informazione e Bioingegneria (DEIB), Politecnico di Milano, 20133 Milan, Italy (e-mail: bernardo.camajori@polimi.it; mattia.brambilla@polimi.it; luca1.barbieri@polimi.it; gabriele.balducci@mail.polimi.it).

Monica Nicoli is with the Dipartimento di Ingegneria Gestionale (DIG), Politecnico di Milano, 20133 Milan, Italy (e-mail: monica.nicoli@polimi.it).

Digital Object Identifier 10.1109/TSP.2023.3304002

I. Introduction

A. Contextualization and Background

In the last two decades, driving automation functionalities have advanced at an incredible rate, allowing an accurate perception of the environment for enhancing vehicle safety [1], [2]. At the same time, the development of cellular communications for the automotive vertical (e.g., 5G and beyond) is driving a new connectivity paradigm for mobility [3], [4], [5]. Vehicle-to-Everything (V2X) communications enable a seamless information sharing among vehicles, road infrastructures and any other road entity over Vehicle-to-Vehicle (V2V) or Vehicle-to-Infrastructure (V2I) links. Examples of exchanged information in V2X networks include sensor data, driving intents and planned trajectories, or safety-related messages [6]. Moreover, V2X communications allow to extend the ego-sensing capabilities beyond the immediate field of view of on-board sensors, enabling the cooperation across sensing systems of different vehicles. The aggregation of data from spatially distributed sensors (both on vehicles and road infrastructure) through V2X links fosters the deployment of the so called Cooperative Localization (CL) systems [7], [8], [9], [10], [11], [12]. A relevant use case for CL in V2X networks is related to Vulnerable Road User (VRU) detection [13], [14], where cooperation can significantly improve the detection capability.

Regardless of whether these sensors are located in the same vehicle [15], [16] or across different units [17], [18], [19], CL heavily relies on the correct association of sensor measurements, i.e., data association [20], [21], [22]. While data association may appear as a simple task, numerous studies have emphasized the importance of addressing this problem due to the limitations of naive solutions that simply associate closely detected objects [23], [24], [25], [26]. These solutions only yield meaningful results if all vehicles detect an identical number of objects, which is an unrealistic assumption due to varying sensor hardware and fields of view, and do not have false alarms due to clutter. Consequently, it is essential to associate multiple sets of measurements related to distinct detected objects that are only partially in common among vehicles.

In the literature, classical approaches for data association were developed for solving the Multiple Object Tracking (MOT) problem, with the ultimate goal of estimating the trajectories of unknown and time-varyingobjects. Differently

from the localization of active devices, passive targets produce unknown measurement-to-target connections, which have to be associated before running any CL algorithm. The fusion of multiple sensors' measurements can take place under centralized [27], [28], [29], [30], [31], [32], [33], [34], [35], [36], distributed [37], [38], [39], [40], [41], [42] or hybrid network architectures [43], [44].

In this paper, we focus on a centralized solution in V2X networks where a central processing unit is in charge of combining the raw data (or derived characteristics) from all connected vehicles. Specifically, the aggregated measurements refer to bounding boxes extracted from lidar sensors at the vehicles. Centralized solutions in the literature mainly rely on probability-based methods such as Belief Propagation (BP), also known as Sum-Product Algorithm (SPA), which gives a systematic approximation of optimal Bayesian inference with an appealing performance-complexity trade-off [45]. BP uses an iterative message passing exchange of information over a suitable graph characterizing the specific problem. BP-based techniques are optimal in case of linear and Gaussian models, but provide only an approximation in case of loopy graphs or statistical distributions arising from real systems [21], [22]. Differently from BP-based solutions, we here propose to use Message Passing Neural Networks (MPNNs), that allow to improve performances upon SPA by directly learning the correct associations and noise distributions from data.

B. Related Works

First works of MOT task were developed in the domain of radar and sonar tracking [30], [31], [32], [33], [34]. Traditional MOT methods, such as Joint Probabilistic Data Association (JPDA) [46], Linear Joint Integrated PDA (LJIPDA) [47] and Multiple Hypothesis Tracker (MHT) [48], assume that the number of targets is known and jointly estimate the target states and association variables. These approaches have been later extended to consider also multi-sensor scenarios [49], [50] as in Linear Multitarget IPDA (LMIPDA) [51]. Recent studies, including probability hypothesis density (PHD) filters [52], [53], adopt finite set statistics to predict the number of targets and target states without directly estimating the association variables. Other studies addressing probabilistic data association can be found in [54], [55], [56], [57], [58], [59]. However, most MOT approaches have limited scalability as the number of sensors and targets grows. Improvements from this point of view have been introduced by BP techniques that are able to achieve high scalability [45]. BP approaches have been investigated for both centralized [27], [28], [29], [36] and distributed [37], [38] solutions.

As far as the data association is concerned, one of the most prevalent approach is to use a graph formulation, which facilitates the description of relationships among multiple measurements on a same set of detected targets. Many solutions that use graphs to solve data association take into account all feasible assignments at the same time, yielding to an NP-hard combinatorial problem [60], [61]. To reduce the complexity, sub-optimal (greedy) methods have been proposed, casting the problem as a linear one and addressing it through minimum-cost [62] and maximum-flow algorithms [63]. These methods, however, do not guarantee satisfying performance, especially in cluttered and occluded environments [64], [65].

Another possibility to successfully manage data association is to build Machine Learning (ML) models that directly learn from data. ML, and in particular Deep Neural Networks (DNNs), have been embedded inside graphs thanks to the rise in popularity of Graph Neural Networks (GNNs) [66], [67]. GNNs, and more specifically MPNNs, inherit the message passing structure of SPA to produce the desired output from a set of inputs. Indeed, they have been jointly used with the SPA to improve the overall performances by correcting errors created by cycles and model mismatch [68], [69]. Compared to DNNs, MPNNs have fewer parameters but they can still catch the linear and non-linear relationships between input data and output, being at the same time scalable [70]. Moreover, MPNNs have been shown to outperform BP on loopy graphs, provided that enough training data is available [71].

C. Contribution

To the best of our knowledge, GNNs have never been explored for cooperative sensing nor for vehicular networks. Drawing inspiration from [72], where the detections obtained by a single camera system were associated over consecutive time frames, we here modify and extend the approach to a cooperative (i.e., multi-vehicle) scenario. We consider a centralized network of vehicles, each with a single lidar sensor, with overlapped Field-of-Views (FOVs) allowing a cooperative detection (at the same time instant) of pedestrians through the association of multiple bounding boxes extracted from the lidar point cloud. We selected pedestrians as they are passive elements of the environment and are extremely relevant for safety-related applications (e.g., vulnerable road user protection) as well as they are popularly present in urban areas. Alternatively, vehicles could also be used, but they are typically equipped with active devices in Cooperative Intelligent Transport Systems (C-ITS) (following V2X paradigms), thus notifying their presence in the near surroundings. On the other hand, passive targets, such as pedestrians, traffic signals or poles [73], are not univocally identified and, therefore, data association is needed for their recognition and cooperative detection by multiple sensors.

We assume that the lidar detection system does not incur false detections (i.e., incorrect bounding boxes), which would require a tracking over time to resolve the ambiguity; here we focus on a snapshot-based data association. This assumption may not always hold in real-world scenarios, especially when objects are partially occluded. However, we employ a filtering strategy that is widely adopted in deep learning object tracking and discards unlikely bounding boxes with low detection confidence (see e.g., [74] for a more complete discussion). This helps limiting the false positives as very unlikely bounding boxes are automatically removed by the detector[1]. We also assume the noise statistics as invariant across all sensors, a condition which in practice might not be fulfilled due todifferent hardware and

[1]Complete removal of false positives may be accomplished by not only taking into account the detection confidence but also the temporal dependencies of detections across adjacent time instants. This extension, not considered here, requires solving the data association problem over multiple graphs relating to adjacent time instants.

lidar processing techniques. Including the variation over time of uncertainties and designing an MPNN-based tracker with data association are non-trivial issues that require deeper research relying on this first activity as a starting point.

We propose an ML approach based on GNNs that exploits the availability of training data, today largely accessible in most applications. In particular, we address the data association problem through MPNNs in a V2X network where vehicles share the detections of lidar sensors in a common infrastructure (e.g., cloud-based). We choose to focus on data association using MPNNs for two main reasons. First, data association is a crucial component in cooperative sensing algorithms, as accurate assignment of detections to tracks significantly impacts the tracking performance: our goal is to provide a solid foundation for more robust tracking solutions. Second, using MPNNs for data association is an innovative approach with great potentials for learning complex relationships in graph-structured data, providing valuable insights and future research opportunities.

A preliminary version of the proposed method has been presented in [75], where we developed a unique graph representation of the data association problem which is handled by an MPNN model that captures the measurements' characteristics and produces a compact and effective feature representation. While in [75] we focused on a simple proof-of-work implementation and stand-alone validation in a vehicular scenario, in this paper we extend the work with the main following contributions:

- proposal of MPNNs models for the cooperative association of 3D bounding boxes from lidar sensing in vehicular networks;
- analysis of the generalization capabilities of the MPNN association model over a number of different and realistic measurement statistics;
- validation of the suggested approach in a realistic cooperative vehicular environment simulated with CARLA [76] where a central unit fuses the bounding boxes obtained by multiple vehicles from on-board lidar data using the PointPillars [77] model;
- comparison with the conventional Sum-Product Algorithm for Data Association (SPADA) [21], [22], with particular focus on association performances and generalization properties.

Note that the assessment on a synthetic cooperative dataset considers the use of an efficient 3D object detector, which has been demonstrated to provide accurate performances in challenging real-world datasets [77]. Since the primary focus of this work is on the fusion over the V2X networks of bounding boxes from multiple vehicles and not on the processing of raw lidar point clouds, any signal losses or adverse weather conditions are not affecting the proposed MPNN as they only reduce the performance of the 3D object detector operating over the lidar point cloud.

Numerical results show that the proposed method is able to efficiently address the data association issue in cooperative connected multi-vehicle systems, and to correctly learn extremely complex (e.g., multi-modal) distributions, such as the realistic PointPillars outputs. Moreover, with respect to SPADA, the proposed MPNN model can achieve higher performances across different noise statistics and intensities in several circumstances.

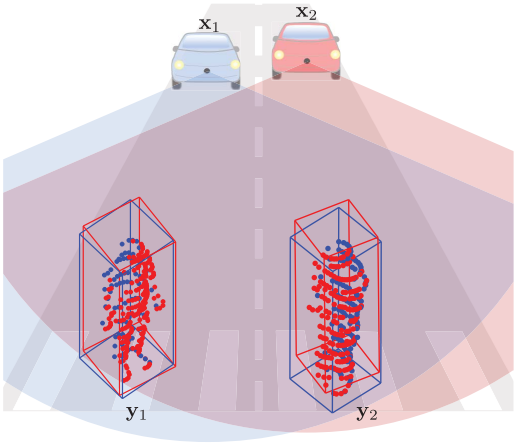
D. Paper Organization

This paper is organized as follows: Section II introduces the system model of the cooperative sensing scenario and its graph representation. Section III firstly provides an introduction on the working principle of GNNs, and then defines the proposed MPNN solution. Section IV is devoted to performance analysis in a cooperative vehicular scenario with lidar-based pedestrian detection and to the comparison with SPADA. Finally, Section V draws the conclusion.

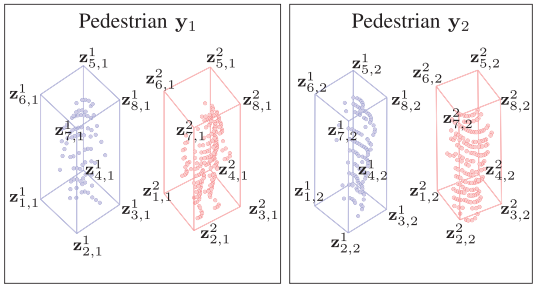
II. System Model

Let us denote with $\mathcal{S}_n = \{1, ..., S_n\}$ a set of connected vehicles at time n. A vehicle $s \in \mathcal{S}_n$ is described by the state vector $\mathbf{x}_{s,n}$, which can include kinematic (e.g., position, velocity, etc.) and non-kinematic (e.g., identification number, category, dimension, etc.) parameters. All vehicles are connected to a central processing unit (e.g., a road side unit or a mobile edge cloud) in charge of aggregating the vehicle-generated information and providing a cooperative detection system. We assume an always-available connectivity: model and effects of the communication protocol are out of the scopes of this paper. Each vehicle has a lidar sensing system embedding an ML algorithm for detecting non-cooperative vulnerable road users, here pedestrians, referred to as targets. The k-th target is described by the state vector $\mathbf{y}_{k,n}$, while the set $\mathcal{Y}_{s,n}$ includes all the pedestrians detectable by vehicle s (i.e., within its FOV) at time n. By processing the lidar point clouds gathered at the vehicles via 3D object detection methods, such as [78], [79], each target falling within the lidar sensing range can be recognized and represented by a bounding box encoding its location, extension and rotation. Each target is assumed to generate at most one bounding box at a vehicle per each time step. This assumption, known as "data association assumption" [80], is common in object detection models for lidar point clouds, and more in general in MOT algorithms, as it helps to simplify the detection and tracking process, reduce ambiguities, and improve the overall tracking performance. The m-th bounding box at vehicle s at time n is $\mathbf{z}^s_{m,n}$ and the associated target is unknown. As such, at time n, a sensor has a set of unpaired (to the originating target) bounding boxes $\mathbf{z}^s_n = \{\mathbf{z}^s_{1,n} \cdots \mathbf{z}^s_{M,n}\}$. Note that the set $\mathbf{z}^s_n$ could even be empty. The union set of all bounding boxes of all vehicles at time n is $\mathcal{Z}_n = \bigcup_{s=1}^{S_n} \mathbf{z}^s_n$.

To visualize the considered vehicular scenario, in Fig. 1(a) we report the case of two vehicles, $\mathbf{x}_{1,n}$ and $\mathbf{x}_{2,n}$, jointly detecting two pedestrians, $\mathbf{y}_{1,n}$ and $\mathbf{y}_{2,n}$, through the bounding boxes $\mathbf{z}^1_n = \{\mathbf{z}^1_{1,n}\, \mathbf{z}^1_{2,n}\}$ and $\mathbf{z}^2_n = \{\mathbf{z}^2_{1,n}\, \mathbf{z}^2_{2,n}\}$ for vehicles $s = 1$ and $s = 2$, respectively. The measurement $\mathbf{z}^s_{m,n}$ is described by the 3D coordinates of its eight corners, i.e., $\mathbf{z}^s_{m,n} = \left[\mathbf{z}^s_{i,m,n}\right]_{i=1}^{8}$, as shown in Fig. 1(b), which take into account the overall footprint and orientation of the target. To correctly associate the bounding



(a) Cooperative detection of pedestrians through lidar sensing



(b) Definition of lidar detections (bounding boxes)

Fig. 1. (a) Cooperative scenario with two vehicles $\mathbf{x}_1$ and $\mathbf{x}_2$ detecting pedestrians $\mathbf{y}_1$ and $\mathbf{y}_2$ by means of lidar technology. (b) Bounding boxes extracted from the lidar point cloud with corner definition. The subscript n is removed for visualization purposes.

boxes, an absolute (fixed) Cartesian spatial reference system has to be used for the identification of the corners. In this case, we choose to label as $\mathbf{z}^s_{1,m,n}$ the bottom-north-est corner and $\mathbf{z}^s_{8,m,n}$ the top-south-west one.

In the proposed GNN solution, the union set $\mathcal{Z}_n$ is modeled as a direct graph $\mathcal{G} = (\mathcal{V}, \mathcal{E})$, where each node $i \in \mathcal{V}$ corresponds to a single measurement, while the edge (i, j), with $i \neq j$, indicates a candidate association. To univocally map the node $i \in \mathcal{V}$ with the measurement m of vehicle s at time n, we define the mapping function $\Phi_n : \mathcal{V} \to \mathcal{Z}_n \times \mathcal{S}_n$. The function $\Phi_n(i) = \{m, s\}$ cannot inherently prevent the association of two distinct measurements of a same vehicle. For this reason, we also introduce the association-related variable $y_{i\to j} \in \{0, 1\}$ which denotes the presence/absence of the edge (i, j), i.e., the two bounding boxes embodied in nodes i and j refer to a same target. The goal of the data association algorithm (here addressed with MPNN) is to estimate the association variable $\widehat{y}_{i\to j} \in \{0, 1\}$ by considering all possible pairings of bounding boxes, with the constraint of $\widehat{y}_{i\to j} = 0$ if the mappings $\Phi_n(i)$ and $\Phi_n(j)$ refer to a same vehicle s.

TABLE I
SUMMARY TABLE OF NOTATION

Description	Symbol
Set of vehicles at time n	$\mathcal{S}_n$
State of vehicle s at time n	$\mathbf{x}_{s,n}$
Set of detectable targets by vehicle s at time n	$\mathcal{Y}_{s,n}$
State of target k at time n	$\mathbf{y}_{k,n}$
Set of bounding boxes of all vehicles at time n	$\mathcal{Z}_n$
Set of bounding boxes of vehicle s at time n	$\mathbf{z}^s_n$
m-th bounding box of vehicle s at time n described by the coordinates of its corners	$\mathbf{z}^s_{m,n} = \left[\mathbf{z}^s_{i,m,n}\right]_{i=1}^{8}$
GNN directed graph with vertex and edges	$\mathcal{G} = (\mathcal{V}, \mathcal{E})$
Mapping function from node $i \in \mathcal{V}$ to measurement m of vehicle s at time n	$\Phi_n : \mathcal{V} \to \mathcal{Z}_n \times \mathcal{S}_n$
Association-related variable	$y_{i\to j} \in \{0, 1\}$

As an example of graph construction, we refer to the vehicular scenario shown in Fig. 2(a) where vehicle $\mathbf{x}_{2,n}$ detects only $\mathbf{y}_{1,n}$ through measurement $\mathbf{z}^2_{1,n}$, while the other two vehicles $\mathbf{x}_{1,n}$ and $\mathbf{x}_{3,n}$ can detect both targets $\mathbf{y}_{1,n}$ and $\mathbf{y}_{2,n}$, respectively. It follows that the graph with true measurements association for such scenario is the one indicated in Fig. 2(b), which has to be reconstructed from the fully-connected graph in Fig. 2(c) that includes all possible pairings. In next section, we detail the proposed algorithm for estimating the connections from all possible associations, i.e., how to get the graph in Fig. 2(b) from the one in Fig. 2(c).

A summary of the main notation variables introduced in this section and their description is provided in Table I.

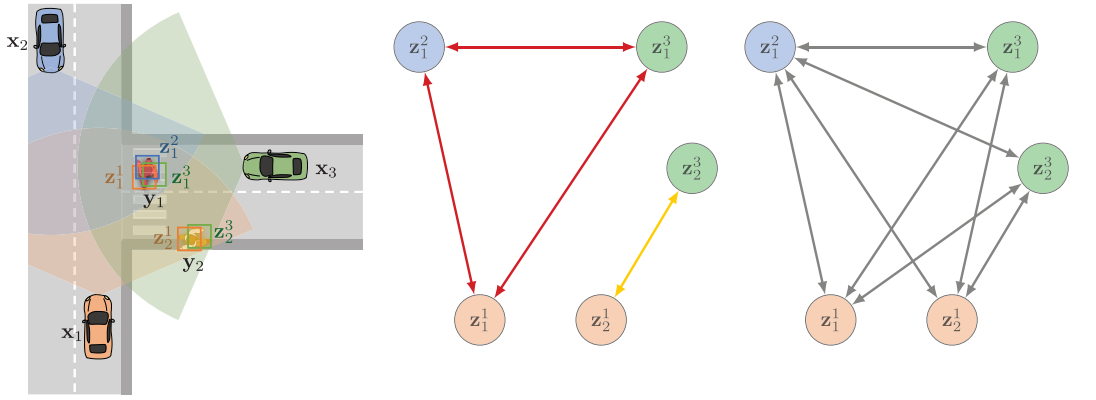
III. Addressing Data Association With MPNN

In this section, we first introduce the general concept of GNN and more specifically MPNN (Section III-A), which is the base for the proposed model. Then, we define the proposed MPNN model with an insight on possible classification strategies. Finally, we describe the loss function used to train the model, as well as the performance metrics.

A. Introduction to GNNs

Neural networks acting on graphs have been investigated for more than a decade, being originally referred to as GNNs [66], [67] and successively extended to many variants such as MPNNs [81]. A complete generalization of GNNs is formulated in [82] under the name of Graph Networks (GNs). Models in this ML family have been studied in supervised, semi-supervised, unsupervised, and reinforcement learning contexts across a wide range of problem domains. They have been used to learn the dynamics of physical systems [83], predict the chemical properties of molecules [84], optimize the communication in multi-agent networks [85], or even employed in machine translation [86]. A further domain of applications includes vehicular environments, where GNN are used to predict road traffic [87], [88] or classify and segment 3D meshes and point clouds [89].

We here consider an MPNN that iteratively performs a message passing procedure over a graph $\mathcal{G}$. Iterations are indexed



(a) Scenario (b) True graph representation (c) Fully connected graph of unknown pairings

Fig. 2. (a) Top view of an exemplary cooperative localization scenario with three vehicles detecting one or two pedestrians each. (b) Virtual graph representing the pairings measurement at multiple vehicles. Each subgraph includes all measurements of a same target. (c) Graph with unknown associations among all measurements. Each edge embodies a potential association, which has to be probabilistically computed to get the result in (b). The color of nodes refers to the color of originating vehicle, while colors of edge indicate the two detected pedestrians.

with t, the maximum number of message passing steps (a design parameter) as T, while $\mathcal{N}_i = \{j \in \mathcal{V} | (i,j) \in \mathcal{E}\}$ is the set of neighbors of node $i \in \mathcal{V}$. We also identify the so called embeddings, i.e., attributes, of node i and edge (i,j) with variables $\mathbf{h}_i^{(t)}$ and $\mathbf{m}_{i\to j}^{(t)}$, respectively.

The purpose of the MPNN is to train a function that propagates information from node and edge embeddings/attributes throughout $\mathcal{G}$. The more message passing steps are performed, the more the node and edge embeddings contain elaborated information, just like the receptive field of a Convolutional Neural Network (CNN). To this extent, a Neural Network (NN) is present at each node and edge of the graph. The NN at node is indicated with $g_{\text{n}}(\cdot)$, while the one over the edge by $g_{\text{e}}(\cdot)$. Considering that $g_{\text{n}}(\cdot)$ and $g_{\text{e}}(\cdot)$ have the same parameters, respectively across each node and each edge, they may be trained on small-scale graphs before being applied to large-scale problems.

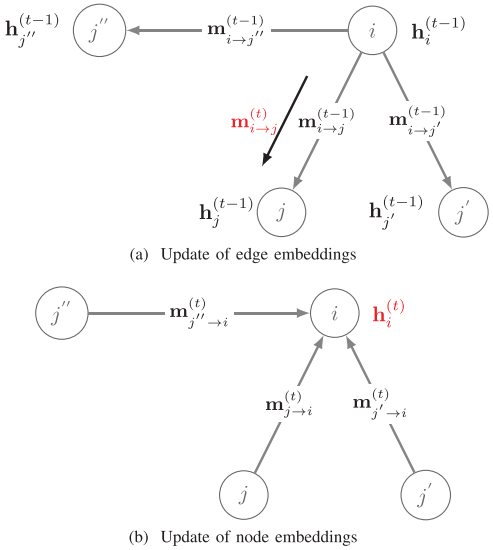
For each iteration $t = 1, \ldots, T$, each node $i \in \mathcal{V}$ sends the following message to its neighbors $\mathcal{N}_i$

$$\mathbf{m}_{i\to j}^{(t)} = g_{\text{e}}\left(\mathbf{h}_i^{(t-1)}, \mathbf{h}_j^{(t-1)}, \mathbf{m}_{i\to j}^{(t-1)}\right), \quad \forall j \in \mathcal{N}_i, \tag{1}$$

with

$$\mathbf{h}_i^{(t)} = g_{\text{n}}\left(\mathbf{h}_i^{(t-1)}, \Phi(\{\mathbf{m}_{j\to i}^{(t)}\}_{j\in\mathcal{N}_i})\right). \tag{2}$$

Function $\Phi(\cdot)$ is called *aggregation* function and it is invariant to permutations of its inputs (e.g., element-wise summation, mean, maximum). Concisely, and referring to Fig. 3, the message $\mathbf{m}_{i\to j}^{(t)}$ sent from node i over an edge (i,j) updates the previously sent message $\mathbf{m}_{i\to j}^{(t-1)}$ over the same edge with the available attributes $\mathbf{h}_i^{(t-1)}$ and $\mathbf{h}_j^{(t-1)}$ of the involved nodes, respectively. The attribute $\mathbf{h}_i^{(t)}$ is obtained by combining together all the incoming messages at the node i, i.e., $\mathbf{m}_{j\to i}^{(t)} \forall j \in \mathcal{N}_i$ (through function Φ) and the previously available information $\mathbf{h}_i^{(t-1)}$ (computed at previous iteration).



(a) Update of edge embeddings

(b) Update of node embeddings

Fig. 3. MPNN working principle: (a) update of edge embeddings, (b) update of node embeddings. The updated elements are in red. For each MPNN step t, first compute the edge embeddings from node i over all edges (i.e., toward all neighbors $\mathcal{N}_i$) according to (1). Then, compute the node embeddings depending on the previously updated edge embeddings as in (2).

B. MPNN Model for Data Association

The proposed model consists of two parts: an MPNN and an edge classifier. The role of the MPNN is to process the input graph $\mathcal{G}$ derived from the measurements of all vehicles at a given time n, i.e., $\mathcal{Z}_n$. On the other hand, the edge classifier is a binary classifier with the role of determining the pairings of all the measurements referring to the same target, i.e., finding the association variable $\widehat{y}_{i\to j}$ based on association probabilities.

As a consequence, at the output of the classifier we have a set of multiple disjoint subgraphs (as in Fig. 2(b)), each of them grouping all the measurements that are hypothesized to be originated from the same target.

The MPNN model is composed of four multi-layer perceptrons (MLPs): $g_{\mathrm{e}}(\cdot)$ at each edge and $g_{\mathrm{n}}^{\mathrm{in}}(\cdot)$, $g_{\mathrm{n}}^{\mathrm{out}}(\cdot)$, $g_{\mathrm{n}}(\cdot)$ at each node. The role of MLPs $g_{\mathrm{e}}(\cdot)$ and $g_{\mathrm{n}}(\cdot)$ is to update the edge and node embeddings, respectively, in a similar way as the conventional MPNN in (1) and (2). On the other side, $g_{\mathrm{n}}^{\mathrm{in}}(\cdot)$ and $g_{\mathrm{n}}^{\mathrm{out}}(\cdot)$ are introduced to better encode the structure of incoming and outgoing edges. In this way, we can split the problem into two parts and individually manage the incoming and outgoing edges in each node.

The message passing over the graph works as follows. First, we update the edges embeddings as in (1), while the node embeddings are updated taking into account both the incoming $\mathbf{m}_{j\to i}^{\mathrm{in},(t)}$ and outgoing $\mathbf{m}_{i\to j}^{\mathrm{out},(t)}$ edge embeddings as

$$\mathbf{h}_i^{(t)} = g_{\mathrm{n}}\left(\sum_{j\,\in\,\mathcal{N}_i} \mathbf{m}_{j\to i}^{\mathrm{in},(t)}, \sum_{j\,\in\,\mathcal{N}_i} \mathbf{m}_{i\to j}^{\mathrm{out},(t)}\right), \tag{3}$$

where $\mathbf{m}_{j\to i}^{\mathrm{in},(t)}$ and $\mathbf{m}_{i\to j}^{\mathrm{out},(t)}$ are defined as

$$\mathbf{m}_{j\to i}^{\mathrm{in},(t)} = g_{\mathrm{n}}^{\mathrm{in}}\left(\mathbf{h}_i^{(t-1)}, \mathbf{m}_{j\to i}^{(t)}\right) \;,\; \forall j \in \mathcal{N}_i, \tag{4}$$

$$\mathbf{m}_{i\to j}^{\mathrm{out},(t)} = g_{\mathrm{n}}^{\mathrm{out}}\left(\mathbf{h}_i^{(t-1)}, \mathbf{m}_{i\to j}^{(t)}\right) \;,\; \forall j \in \mathcal{N}_i. \tag{5}$$

We remark that this is done to divide the problem into two parts, as the constructed graph for solving the data association is bi-directed (i.e., undirected), which is common in most graphs used by MPNNs. However, our approach also needs to ensure the unique constraints of our data association problem, i.e., that the association-edge between two measurements is conceptually the same in both directions.

After T message passing steps, the edge embeddings $\mathbf{m}_{i\to j}^{(T)}$ are fed into an MLP edge classifier $g_{\mathrm{e}}^{\mathrm{class}}(\cdot)$ which evaluates the association probabilities $\widehat{y}_{i\to j}^{(T)}$ as

$$\widehat{y}_{i\to j}^{(T)} = g_{\mathrm{e}}^{\mathrm{class}}\left(\mathbf{m}_{i\to j}^{(T)}\right), \forall (i,j) \in \mathcal{E}. \tag{6}$$

The association variables $\widehat{y}_{i\to j}$ are then obtained with a thresholding operation, with threshold Γ, to pair nodes i and j. Two nodes are associated (i.e., two bounding boxes at distinct vehicles refer to a same pedestrian) if

$$\widehat{y}_{i\to j}^{(T)} \geq \Gamma\,, \tag{7}$$

which implies $\widehat{y}_{i\to j} \triangleq 1$. However, it may happen that one measurement of a vehicle is associated to multiple measurements of another vehicle. To avoid this issue, a constraint is enforced such that a bounding box of a vehicle can be associated to *at most* one bounding box of another vehicle.

C. Loss and Performance Metrics

For computing the training loss and performing back-propagation, we employ the weighted binary cross-entropy that is estimated at the end of each message passing iteration t after the edge classifier's prediction $\widehat{y}_{i\to j}^{(t)}$ as

$$\mathcal{L} = -\frac{1}{|\mathcal{E}|}\sum_{t=1}^{T}\sum_{(i,j)\in\mathcal{E}}\Big\{(1-y_{i\to j})\log\left(1-\widehat{y}_{i\to j}^{(t)}\right) + w\, y_{i\to j}\log\left(\widehat{y}_{i\to j}^{(t)}\right)\Big\}, \tag{8}$$

where w is a weight given to the positive class in order to compensate the class unbalances and it is computed as

$$w = \frac{\sum_{(i,j)\,\in\,\mathcal{E}} \mathbf{1}(y_{i\to j}=0)}{\sum_{(i,j)\,\in\,\mathcal{E}} \mathbf{1}(y_{i\to j}=1)}\,, \tag{9}$$

where $\mathbf{1}(\cdot)$ is an indicator function that returns 1 if the condition is true and 0 otherwise. Concerning the performance metrics, we adopt the accuracy measure defined as

$$\text{Accuracy} = \frac{\text{TP}+\text{TN}}{\text{TP}+\text{FP}+\ \text{TN}+\text{FN}}\,, \tag{10}$$

where the terms TP, TN, FP and FN indicate the number of True Positive (TP), True Negative (TN), False Positive (FP) and False Negative (FN), respectively.

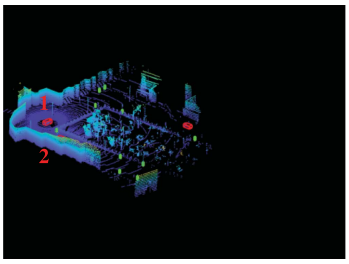
IV. Simulation Experiments

To evaluate the proposed MPNN model for data association we consider a network of vehicles localizing pedestrians through lidar sensing. We dedicate Section IV-A to the simulation scenario and dataset, while Section IV-B reports the results of performed simulations.

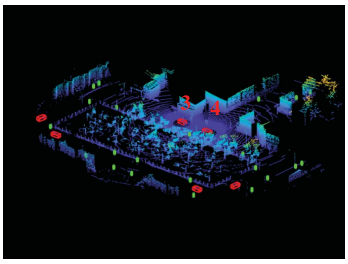
A. Simulation Scenario

Due to the unavailability of real-world cooperative perception datasets, i.e., collected by multiple and synchronous lidar-equipped vehicles, we here employ a simulator of automated driving systems that allows us to generate lidar readings at multiple vehicles moving in a synthetic, yet realistic, mobility environment. Similarly to [90], we use the CARLA simulator [76], an extremely advanced software that integrates trajectory planning and sensing. The considered scenario is referred to as $Town02$ in the simulator, which spans over an area of roughly 200 m $\times$ 200 m. Twenty vehicles with lidar and fifty pedestrians populate the scene, unless otherwise specified. The state $\mathbf{x}_{s,n}$ of each vehicle refers to its 3D position. A snapshot restricted to seven vehicles with associated point clouds of the simulator is show in Fig. 4, where we represent the effect of cooperative sensing by merging seven lidar point clouds. Specifically, for visualization purposes we group the vehicles into three subgroups and we show the partial point cloud in Figs. 4(a), 4(b) and 4(c), respectively, while the cooperative perception obtained by merging all the seven point clouds is in Fig. 4(d).

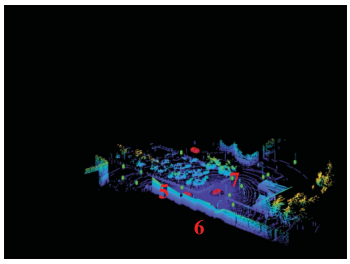
The duration of simulation is 300 s, with sampling time of 0.2 s. This results in 1500 snapshots of the scene, each one described by vehicles and pedestrians' positions and lidar detections. A top-view image of the simulation in a fixed time instant is shown in Fig. 5, where we include both vehicles (red squares) and detected pedestrians (blue triangles) as well as the



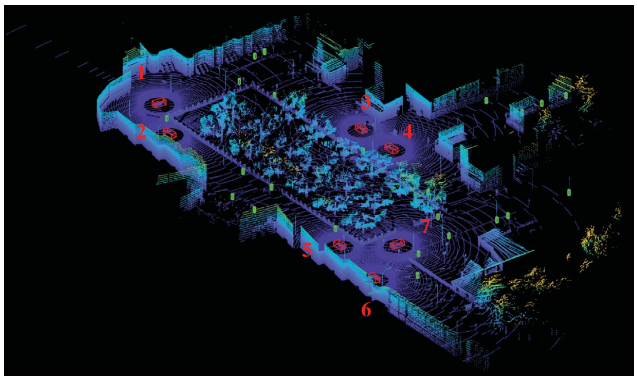
(a) Lidar sensing from vehicles 1-2 in the left



(b) Lidar sensing from vehicles 3-4 in the center



(c) Lidar sensing from vehicles 5-6-7 at the bottom



(d) Cooperative lidar sensing from all vehicles

Fig. 4. Snapshot of lidar sensing simulated by CARLA: seven vehicles (red bounding boxes) detect pedestrians (green bounding boxes). (a) Partial lidar sensing from the two vehicles in the left. (b) Partial lidar sensing from the two vehicles in the center. (c) Partial lidar sensing from three vehicles at the bottom. (d) Cooperative sensing as a combination of seven lidar point clouds.

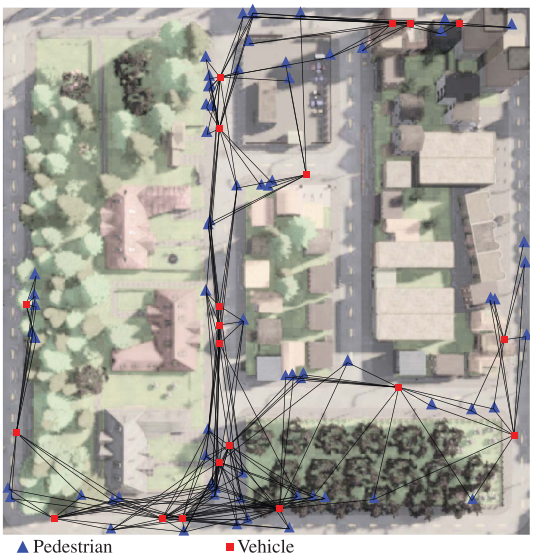


Fig. 5. Top-view of the simulation scenario with twenty vehicles (red squares), detected pedestrians (blue triangles) and detections (black lines).

associated detections (black link). A vehicle $s \in \mathcal{S}$ can detect a pedestrian $k \in \mathcal{F}_{v,n}$ if it falls in its field of view. All lidar sensors are configured to run at a 5 Hz update frequency, with an FOV of 360 deg in azimuth and $[-30, +10]$ deg in elevation. The number of channels supported is 64, corresponding to a spatial resolution of 0.625 deg. The sensing range is limited to 70 m and the number of points of the cloud cannot exceed 1 million per second. The single point has an accuracy of ± 2 cm. To simulate realistic operating conditions, 20% of the points are randomly dropped during every simulation frame.

The ground truth information provided by the simulator includes the true positions of vehicles, i.e., $\mathbf{x}_{s,n}, \forall s \in \mathcal{S}_n$, and the true bounding boxes around the pedestrians, defined by its eight corners, i.e., $\mathbf{y}_{k,n} = [\mathbf{y}_{i,k,n}]_{i=1}^{8}$. Localization errors are introduced as an additive measurement error $\mathbf{w}_{i,m,n}^{s}$, which directly translates over the 3D corners of the measured bounding boxes. The resulting noisy measurement of a bounding box corner is thus defined as:

$$\mathbf{z}_{i,m,n}^{s} = \mathbf{y}_{i,k,n} + \mathbf{w}_{i,m,n}^{s}\,, \quad \forall k \in \mathcal{Y}_{s,n}, \forall i \in \{1, \ldots, 8\}\,. \tag{11}$$

Note that the measurement error distribution is the same for all corners, for all time instants and across all vehicles. Unless otherwise specified, the artificial noise $\mathbf{w}_{i,m,n}^{s}$ follows an isotropic

Gaussian distribution with standard deviation $\sigma = 10$ cm. Note that the additive noise is absent in case of using a ML model for the automatic extraction of bounding boxes, as in the case of PointPillars [77], since the error is embedded in the model itself.

In the simulations, we validate the systems for a variety of noise intensities ranging from extremely accurate detections up to inefficient systems (errors in the order of meters) for the considered vehicular context targeting the automation of mobility. The former case can be considered as a condition in which the vehicle position is assumed to be perfectly known and the only source of error is attributed to lidar sensing and bounding boxes extraction algorithm. The latter case, instead, embeds both vehicle uncertainty and the errors in the generation of bounding boxes. We do not consider separate effects as we aim to assess the aggregated model robustness.

We divide the overall dataset into training (700 samples) and validation (800 samples) parts, with dimensions optimized as discussed in Section IV.B.2. Moreover, in order to assess the generalization of the method, we increase the number of validation samples by applying a random flip along the x and y axes of the bounding box positions, thus obtaining a total of 1600 validation samples. We remark that a sample is a snapshot of the scene at a given time instant n and it is fully represented by the graph of unknown measurement-pairings (Fig. 2(c)). To avoid the computational burden of dealing with too many edges, we introduce a gating which a-priori discards unlikely associations, i.e., ignoring edges related to centroids whose distance is greater than 10 m. As optimizer, we use the Adam optimizer with tuned learning rate of 10^{-3} and hyper-parameters $\beta_1 = 0.9$ and $\beta_2 = 0.999$ [91]. The performance metrics are computed using the thresholding in (7) with $\Gamma = 0.5$.

B. Simulation Results

1) Initialization of Node and Edge Embeddings: To initialize the node and edge embeddings, we adopt a strategy that learns how to extract feature embeddings directly from measurements. This is done by using an MLP at each node and edge, called $g_{\text{n}}^{\text{enc}}(\cdot)$ and $g_{\text{e}}^{\text{enc}}(\cdot)$, respectively. For the considered cooperative lidar sensing scenario, we use the geometric characteristics of the bounding boxes as input to the two neural networks to obtain $\mathbf{m}_{i\to j}^{(0)}$ and $\mathbf{h}_i^{(0)}$ as

$$\mathbf{m}_{i\to j}^{(0)} = g_{\text{e}}^{\text{enc}}\left(\mathbf{z}_{1,m,n}^{s} - \mathbf{z}_{1,m',n}^{s'}, \mathbf{z}_{8,m,n}^{s} - \mathbf{z}_{8,m',n}^{s'}\right), \quad \forall (i,j) \in \mathcal{E} : \Phi_n(i) = \{m,s\} \wedge \Phi_n(j) = \{m',s'\}, s \neq s', \tag{12}$$

$$\mathbf{h}_i^{(0)} = g_{\text{n}}^{\text{enc}}\left(\mathbf{z}_{m,n}^{s}\right) \quad \forall i \in \mathcal{V} : \Phi_n(i) = \{m,s\}. \tag{13}$$

This allows the MPNN to discriminate not only the position of the detected object, but also its dimension and rotation. We find this approach to be highly effective and efficient, as it uses a minimal amount of information for data association, limiting the data exchange among vehicles.

Incorporating additional features, such as individual point cloud positions, into the current feature encoding could be beneficial and would require only modifying the encoding neural networks $g_{\text{n}}^{\text{enc}}(\cdot)$ and $g_{\text{e}}^{\text{enc}}(\cdot)$. While this could potentially enhance performance, there are two primary drawbacks to consider. First, the volume of information that would need to be exchanged with the central entity responsible for data association via MPNN could become unmanageable and unsustainable, given that a lidar sensor typically outputs more than 1 million point clouds per second. Second, increasing the number of input features might inadvertently introduce unrelated or redundant features that may not be beneficial or could even negatively impact the inference process due to the multi-dimensionality problem in machine learning.

TABLE II
IMPACT OF TRAINING DATASET SIZE (NUMBER OF SAMPLES) ON ACCURACY, PRECISION AND RECALL METRICS

# of samples	Accuracy	Precision	Recall
16	0.869	0.852	0.859
32	0.919	0.891	0.912
64	0.972	0.953	0.980
128	0.993	0.989	0.992
300	0.999	0.998	0.998
700	0.999	0.999	0.999

2) Impact of Training Dataset Dimension: We first analyze the impact of the training dataset size on the model's performance. This is crucial in determining whether the model exhibits high or low bias. In essence, expanding the dataset size decreases the model's variance, meaning the residual error is predominantly due to bias. In Table II, we present the validation accuracy, precision, and recall after 100 epochs for varying training dataset size. We note that by increasing the number of samples, the model improves the performance metrics and reaches an upper bound on the accuracy after 700 samples, representing the best accuracy reachable by the model, i.e., its bias. It is noteworthy that the recall typically overcomes the precision, implying a larger number of false positives than false negatives. This is because the ground-truth graph retains a predominant number of zeroed edges, thus the model is more prone to mistake on edges that are labeled as zeros, despite the loss function employed (8) for unbalanced classes.

3) Impact of MPNN Iterations: This assessment aims to verify the role of the number of message passing iterations T, a fundamental parameter to tune the amount of information extracted and elaborated from the data. In Fig. 6, we show the accuracy (and associated confidence) metric in the validation dataset over the number of epochs for $T = \{1, 2, 4, 8, 12\}$. We notice that increasing T leads to a higher accuracy and a faster convergence, at the cost of increasing computational complexity. However, a saturation condition occurs for $T > 4$, leading us to select $T = 4$ as a good trade-off between accuracy, convergence and complexity. This value will be used for the following analyses.

4) Impact of the Measurement Statistics: This assessment has the goal of verifying how the MPNN model handles unobserved noises (for which it has never been trained on). This is extremely useful in case the model is trained in a

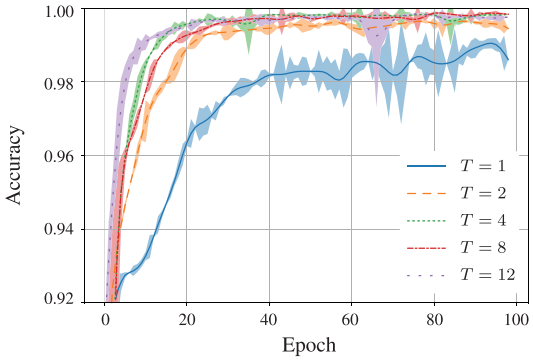


Fig. 6. Accuracy of the MPNN for different values of message passing iterations $T = [1, 2, 4, 8, 12]$ over the epochs for the validation dataset. The mean value (solid line) is plotted together with the associated uncertainty (shaded area) computed using the maximum and minimum values of accuracy as boundaries.

TABLE III
DEFINITION OF MEASUREMENT NOISE DISTRIBUTIONS

Distribution	$f_{\mathbf{w}}(\mathbf{w})$
Isotropic Gaussian	$(\sqrt{2\pi\sigma^2})^{-1}\exp(-0.5\,\sigma^{-2}\mathbf{w}^2)$
Non-isotropic Gaussian	$(\sqrt{\det(2\pi\boldsymbol{\Sigma})})^{-1}\exp(-0.5\,\mathbf{w}^{\mathrm{T}}\boldsymbol{\Sigma}^{-1}\mathbf{w})$
Laplace	$(\sqrt{2\sigma^2})^{-1}\exp(-\sqrt{2}\,\sigma^{-1}\lvert\mathbf{w}\rvert)$
Uniform	$\begin{cases} 0.75\,(\pi\sigma^3)^{-1}, & \text{if } \lVert\mathbf{w}\rVert_2 < \sigma, \\ 0, & \text{otherwise.} \end{cases}$
Discrete	$\begin{cases} 0.4, & \text{if } \mathbf{w} = \mathbf{0}_3, \\ 0.1, & \text{if } \mathbf{w} = \{\mathbf{w}_1, \mathbf{w}_2, \mathbf{w}_3\}, \\ 0, & \text{otherwise.} \end{cases}$

simulated and controlled environment, and then deployed in real systems typically characterized by different measurement statistics. Since there is almost no literature detailing the error characteristics of real ML-based 3D object detectors, we investigate the types of noise that are currently considered in point cloud denoising algorithms. As suggested in [92], we explore five different noise statistics: the already introduced isotropic Gaussian, the non-isotropic Gaussian, the Laplace, the uniform, and the discrete one, which are defined by the distributions $f_{\mathbf{w}}(\mathbf{w}) \triangleq f_{\mathbf{w}}(\mathbf{w}^s_{i,m,n})$ as reported in Table III, where σ denotes the standard deviation, $\boldsymbol{\Sigma} = \sigma^2\left[[1, -0.5, -0.25]^{\mathrm{T}}, [-0.5, 1, -0.25]^{\mathrm{T}}, [-0.25, -0.25, 1]^{\mathrm{T}}\right]^{\mathrm{T}}$, while $\mathbf{w}_1 = (\pm\sigma, 0, 0)$, $\mathbf{w}_2 = (0, \pm\sigma, 0)$ and $\mathbf{w}_3 = (0, 0, \pm\sigma)$.

We also implement the ML model PointPillars [77] to process the lidar point clouds and derive the associated bounding boxes. This method allows us to assess the performance of the MPNN without resorting to artificially generated measurements in (11), leading to a detection system that closely resembles practical scenarios. By using the detections produced by PointPillars as inputs to our data association system, we maintain a noise distribution that mirrors realistic conditions, which is essential for evaluating the effectiveness of the proposed MPNN-based data association strategy in various real-world situations. The

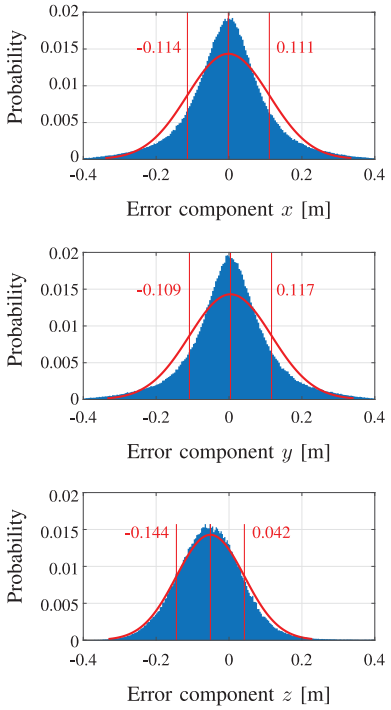


Fig. 7. PointPillars localization error on the corners of bounding boxes along x, y and z components. The three histograms are approximated with a Gaussian distribution whose standard deviation is highlighted in red.

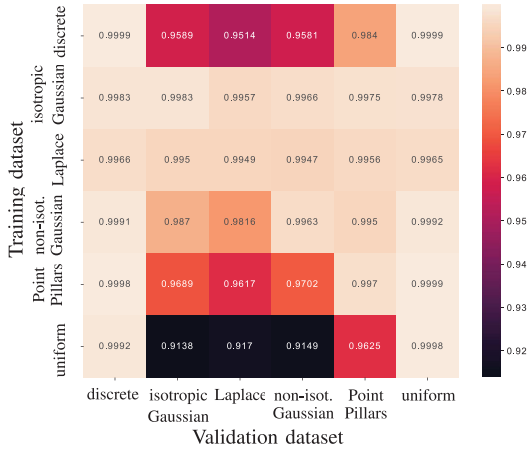
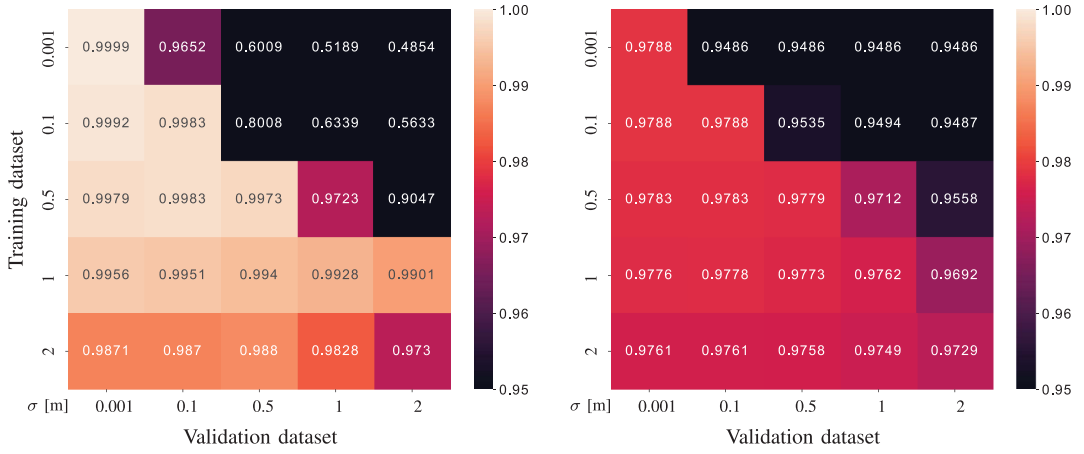


Fig. 8. Accuracy of the MPNN model (after 25 epochs) for training/validation mismatch on measurement noise.

statistics of the 3D localization error (computed as the difference between the true and estimated bounding boxes over x, y and z axes) of PointPillars are reported in Fig. 7, showing that they can be well approximated by a zero-mean Gaussian



(a) MPNN (b) SPADA

Fig. 9. Comparison on data association accuracy between (a) MPNN and (b) SPADA. For MPNN we show the accuracy (after 25 epochs) for training/validation mismatch on measurement noise, whereas for SPADA we validate each standard deviation of the likelihood with different noise intensities.

distribution over the horizontal (x, y) space. On the other hand, for the vertical dimension z, the model is more likely to predict boxes in higher positions (i.e., above the road) instead of the opposite, so the error distribution is slightly biased. Furthermore, we notice that the error statistics do not vary significantly over the three axes, suggesting that they (almost) follow an isotropic Gaussian distribution with standard deviation $\sigma = 10$ cm.

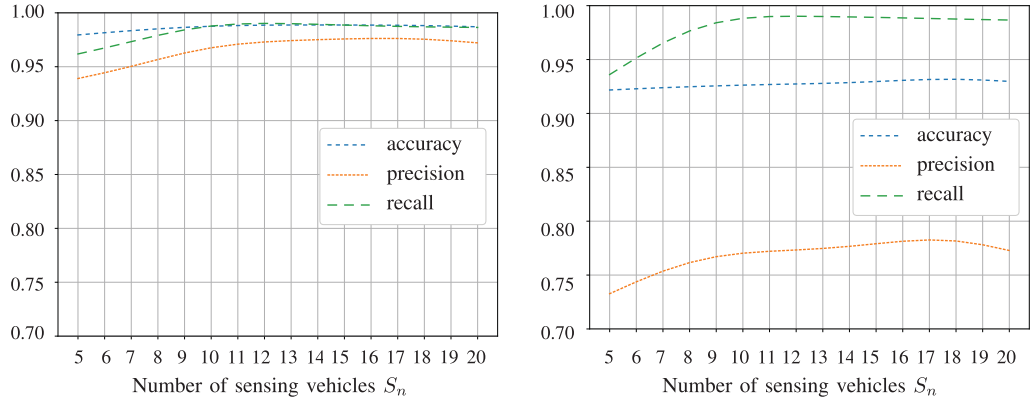
In Fig. 8 we analyze the mixed impact of training and testing with different noise statistics. The value of σ for each noise distribution has been set to 10 cm as closely matching the standard deviation used for fitting the error statistics of PointPillars. Analyzing the results, it is apparent that training and validating the MPNN model on the same noise distributions lead to optimal performances. This shows that the model is able to obtain good accuracy regardless of the noise type, provided that the same noise is experienced for both training and validation phases. Focusing now on the different combinations of training/validation noises, results detail that training under the isotropic Gaussian or Laplace noise allows the model to generalize well over all noise types, suggesting that these distributions may be employed in real-life applications where noise statistics are not known beforehand. On the other hand, training considering discrete and/or uniform leads to poor generalization results during validation, most probably due to the simplistic noise distributions compared to all other noise types. Finally, training on realistic data, i.e., over the noise generated by PointPillars, does not allow the MPNN to generalize well over other distributions, particularly for Laplace and isotropic/non-isotropic Gaussian noises.

5) Impact of the Different Scenarios and False Positives: This experiment aims at verifying the validation performances of the proposed model in a brand new scenario where false positives, i.e., false alarms, are present. This allows us to assess the robustness and adaptability of the model in more realistic conditions, demonstrating its potential for practical implementation.

To this purpose, in Fig. 10, we report the results of performance validation in $Town10$ scenario of CARLA simulator, where we vary the number of cooperating vehicles from 5 to 20. The proposed MPNN association strategy is evaluated considering both the absence (Fig. 10(a)) and presence (Fig. 10(b)) of false alarms, which are obtained from the Pointpillars detector. We would like to highlight that the model has been trained in the map $Town2$ illustrated in Fig. 5 neglecting any false positive, thus $Town10$ and the presence of false alarms are unseen conditions. Starting from the scenario without false alarms in Fig. 10(a), we note that increasing the number of vehicles leads to better accuracy, up to a plateau around 97%, which is just 2% below the results in the original scenario with Pointpillars (see Fig. 8). Accounting for the false positives, we notice in Fig. 10(b) a decrease of the accuracy to 93%. Even more relevant is the precision which falls to 78% due to the fact that each false positive introduces new nodes and edges in the graph which will be associated with real detections, leading to lower performances.

6) MPNN vs SPADA - Generalization Capabilities: This experiment compares the performance of the proposed MPNN association model against a conventional SPADA over different combinations of Gaussian noise intensities used in the training and validation datasets. For the SPADA, a training phase is not needed, but we can embed prior knowledge on the noise intensity by calibrating the standard deviation used for computing the measurement likelihood function. To do so, we process the training dataset and extract a single standard deviation value that characterizes the considered noise intensity.

The comparison is reported in Fig. 9. Regarding the MPNN, we show, for different training and validation datasets, the validation accuracy reached after 25 epochs, while for the



(a) Unseen scenario without false alarms

(b) Unseen scenario with false alarms

Fig. 10. Analysis of validation accuracy, precision and recall in an unseen scenario, for different number of cooperative vehicles: (a) absence of false alarms, (b) presence of false alarms.

SPADA we represent the validation accuracy after convergence using different a-priori noise statistics (in terms of standard deviation). First, we can clearly observe that, in both algorithms, the bottom-left part of the matrix has higher values of accuracy if compared with the top-right part. This is due to the fact that, generally, overestimating the noise (i.e., bottom-left part) leads to a more robust model that can handle noises with lower intensity. On the contrary, underestimating the noise (i.e., top-right part) can incur into problematic situations, especially in the case of MPNN (Fig. 9(a)). From this point of view, SPADA (Fig. 9(b)) is more solid and can better handle different noise values. Under overestimating conditions, on the other hand, the MPNN is able to achieve superior performances compared to SPADA, reaching an accuracy of 99% against 97%, respectively.

7) MPNN vs SPADA - Performances on Different Noise Statistics: This experiment has the aim of comparing the peak or absolute performances of MPNN and SPADA in case we have a training dataset with same statistics of the validation dataset. Understanding the maximum performances is fundamental to have an upper-bound on a real deployment and to know the learning capabilities of the algorithm/model.

In Fig. 11 we report the validation accuracy reached by MPNN and SPADA varying the adopted dataset and for different standard deviations of the noise. For the MPNN we use training and validation datasets with the same value of σ, while the standard deviation of the likelihood in the SPADA is the same as in the validation dataset. We notice that the absolute performances of MPNN outperform the classic SPADA for both $\sigma = 0.1$ m and $\sigma = 0.5$ m and for all datasets. Therefore, the proposed method is able to fully solve the problem and learn synthetic or realistic noise representations. Clearly, for the dataset obtained with PointPillars, we cannot tune the quantity of noise introduced by the ML model and consequently the red and blue circles for the dataset PointPillars coincide. Lastly, we can observe that the degradation of performances passing from $\sigma = 0.1$ m to $\sigma = 0.5$ m are worst for the MPNN. This behaviour is further investigated in the next experiment.

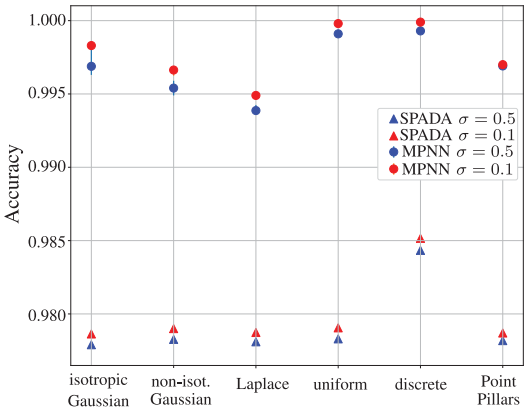
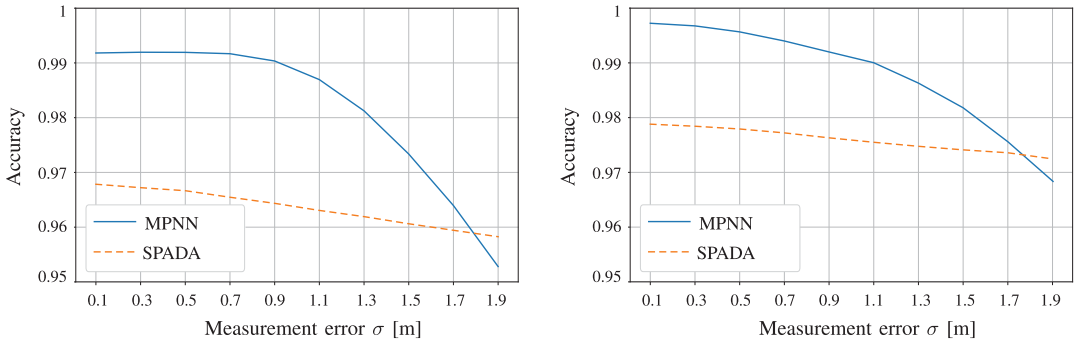


Fig. 11. Comparison of reached accuracy using MPNN (circles) and SPADA (triangles) in different validation datasets. Red markers represent an artificial measurement noise in the validation dataset with standard deviation 0.1 m, while the blue markers describe a standard deviation of 0.5 m. For the MPNN, the noise statistics of the training match those of the validation, whereas for SPADA the standard deviation of the likelihood matches the standard deviation of the noise in the validation dataset.

8) MPNN vs SPADA - Performances on Different Noise Intensity: In this last assessment, we study how the MPNN and SPADA perform over different levels of detection accuracies. This is useful to understand if there are conditions in which one method outperforms the other.

To this aim, we consider different standard deviations of the Gaussian measurement error. The results of this analysis are in Fig. 12, where we report the validation accuracy of the MPNN and SPADA in a scenario with 100 (Fig. 12(a)) or 50 (Fig. 12(b)) pedestrians to be detected. First, we observe that the performances for the scenario in Fig. 12(b) are generally higher than for the scenario in Fig. 12(a). This is due to the fact that with a higher number of pedestrians, the uncertainty on the data association increases and the data association becomes



(a) 100 pedestrians

(b) 50 pedestrians

Fig. 12. Comparison of the impact of measurement error in terms of accuracy between MPNN and SPADA for a scenario with (a) 100 pedestrians or (b) 50 pedestrians. An isotropic Gaussian distribution with standard deviation σ is considered for the additive noise statistics.

more challenging. Second, comparing the two methods across different noise intensities, we note that the MPNN is preferable when the standard deviation of the noise is below 1.8 m in both scenarios. We believe that this behaviour is caused by the fact that the MPNN has difficulties in learning a noise with high variance with respect to a low power noise. On the contrary, the SPADA depends on the standard deviation of the likelihood that in this case is known a-priori and equal to the standard deviation of the validation dataset. Therefore, with high noise intensities, it is preferable to use SPADA as we would need too many samples to learn the noise directly from data.

V. Conclusion

This paper addressed the problem of data association in a cooperative vehicular sensing scenario with multiple vehicles detecting pedestrians through lidar sensors. To solve the problem, we proposed an MPNN model based on a novel graph representation encoding node and edge feature attributes to express the detection knowledge. The validation was carried out in a vehicular environment simulated by CARLA software, which allows to reproduce realistic cooperative lidar sensing scenarios. We considered the PointPillars model for the extraction of bounding boxes from the lidar point cloud, obtaining realistic statistics of bounding boxes measurements. Furthermore, we compared the proposed method with the conventional SPADA to investigate the generalization capabilities and peak performances.

Results showed that the proposed MPNN model is able to learn the correct associations under several realistic measurement statistics and handles good generalization capabilities when it comes to dealing with untrained conditions, such as different measurement error statistics, noise intensities, number of vehicles and new scenarios. The lidar detection error introduced by PointPillars has been found to be well approximated by a Gaussian distribution with standard deviation equal to 10 cm. Under this condition, very high accuracy can be reached by training the model on artificial noises, e.g., Laplace or Gaussian, and then validate the model on the field with realistic noise distribution produced by PointPillars. Concerning the comparison with the classic SPADA, we found that, under overestimation of noise intensity, the proposed method achieved higher performances. Moreover, regarding peak performances, MPNN completely outperforms SPADA up to a noise standard deviation of 1.8 m.

In the incoming years, the relevance of cooperative perception is expected to grow rapidly, particularly in the context of automated and connected mobility, where the new-generation V2X communication technologies bring opportunities for the development of new services. It follows that an efficient management of data association is a fundamental and crucial step for enabling cooperative sensing. As a result, we expect our work to be extended and applied to different contexts. By enhancing the data association performance, our method provides a solid foundation for more accurate and robust object tracking when combined with existing tracking algorithms which exploit the information shared by the vehicles to perform cooperative positioning or sensing of the surrounding environment.

A natural extension of the work would be to manage and account for possible false and/or missed detections through intra-temporal association and non fully-connected vehicular networks. Future developments could also embrace the area of distributed sensor networks in which the flood of information over sensors demands fast interactions of locally-available data but guarantees higher resilience compared to centralized architecture, overcoming the problem of single point of failure. On the other hand, hop-by-hop transport might introduce a non-negligible time delay before the same full information is available at all nodes. In addition, we plan to evaluate our method on real-world cooperative data which would help further validate and refine our approach, ensuring its effectiveness in addressing real-world object detection and tracking challenges.

References

[1] A. Mahdavian, A. Shojaei, S. Mccormick, T. Papandreou, N. Eluru, and A. A. Oloufa, "Drivers and barriers to implementation of connected, automated, shared, and electric vehicles: An agenda for future research," *IEEE Access*, vol. 9, pp. 22195–22213, Feb. 2021.

[2] S. Kuutti, S. Fallah, K. Katsaros, M. Dianati, F. Mccullough, and A. Mouzakitis, "A survey of the state-of-the-art localization techniques and their potentials for autonomous vehicle applications," *IEEE Internet Things J.*, vol. 5, no. 2, pp. 829–846, Apr. 2018.

[3] A. Ghosh, A. Maeder, M. Baker, and D. Chandramouli, "5G evolution: A view on 5G cellular technology beyond 3GPP release 15," *IEEE Access*, vol. 7, pp. 127639–127651, Sep. 2019.

[4] C. De Lima et al., "Convergent communication, sensing and localization in 6G systems: An overview of technologies, opportunities and challenges," *IEEE Access*, vol. 9, pp. 26902–26925, Jan. 2021.

[5] I. F. Akyildiz, A. Kak, and S. Nie, "6G and beyond: The future of wireless communications systems," *IEEE Access*, vol. 8, pp. 133995–134030, Jul. 2020.

[6] H. Bagheri et al., "5G NR-V2X: Toward connected and cooperative autonomous driving," *IEEE Commun. Standards Mag.*, vol. 5, no. 1, pp. 48–54, Mar. 2021.

[7] H. Wymeersch, J. Lien, and M. Z. Win, "Cooperative localization in wireless networks," *Proc. IEEE*, vol. 97, no. 2, pp. 427–450, Feb. 2009.

[8] A. Conti, M. Guerra, D. Dardari, N. Decarli, and M. Z. Win, "Network experimentation for cooperative localization," *IEEE J. Sel. Areas Commun.*, vol. 30, no. 2, pp. 467–475, Feb. 2012.

[9] A. Ihler, J. Fisher, R. Moses, and A. Willsky, "Nonparametric belief propagation for self-localization of sensor networks," *IEEE J. Sel. Areas Commun.*, vol. 23, no. 4, pp. 809–819, Apr. 2005.

[10] N. Patwari, J. Ash, S. Kyperountas, A. Hero, R. Moses, and N. Correal, "Locating the nodes: Cooperative localization in wireless sensor networks," *IEEE Signal Process. Mag.*, vol. 22, no. 4, pp. 54–69, Jul. 2005.

[11] Y. Zheng, N. Cao, T. Wimalajeewa, and P. K. Varshney, "Compressive sensing based probabilistic sensor management for target tracking in wireless sensor networks," *IEEE Trans. Signal Process.*, vol. 63, no. 22, pp. 6049–6060, Nov. 2015.

[12] N. Cao, S. Choi, E. Masazade, and P. K. Varshney, "Sensor selection for target tracking in wireless sensor networks with uncertainty," *IEEE Trans. Signal Process.*, vol. 64, no. 20, pp. 5191–5204, Oct. 2016.

[13] D. Caveney, "Cooperative vehicular safety applications," *IEEE Control Syst. Mag.*, vol. 30, no. 4, pp. 38–53, Aug. 2010.

[14] G. Pocovi, M. Lauridsen, B. Soret, K. I. Pedersen, and P. Mogensen, "Automation for on-road vehicles: Use cases and requirements for radio design," in *Proc. IEEE 82nd Veh. Technol. Conf. (VTC)*, Sep. 2015, pp. 1–5.

[15] A. Milan, S. H. Rezatofighi, A. Dick, I. Reid, and K. Schindler, "Online multi-target tracking using recurrent neural networks," in *Proc. 31st AAAI Conf. Artif. Intell.*, Feb. 2017, pp. 4225–4232.

[16] A. Sadeghian, A. Alahi, and S. Savarese, "Tracking the untrackable: Learning to track multiple cues with long-term dependencies," in *Proc. IEEE Int. Conf. Comput. Vision*, Dec. 2017, pp. 300–311.

[17] S. Zhang et al., "Distributed direct localization suitable for dense networks," *IEEE Trans. Aerosp. Electron. Syst.*, vol. 56, no. 2, pp. 1209–1227, Apr. 2020.

[18] S. Safavi, U. A. Khan, S. Kar, and J. M. F. Moura, "Distributed localization: A linear theory," *Proc. IEEE*, vol. 106, no. 7, pp. 1204–1223, Jul. 2018.

[19] A. Conti, S. Mazuelas, S. Bartoletti, W. C. Lindsey, and M. Z. Win, "Soft information for localization-of-things," *Proc. IEEE*, vol. 107, no. 11, pp. 2240–2264, Nov. 2019.

[20] D. Hall and J. Llinas, "An introduction to multisensor data fusion," *Proc. IEEE*, vol. 85, no. 1, pp. 6–23, Jan. 1997.

[21] F. Meyer et al., "Message passing algorithms for scalable multitarget tracking," *Proc. IEEE*, vol. 106, no. 2, pp. 221–259, Feb. 2018.

[22] J. Williams and R. Lau, "Approximate evaluation of marginal association probabilities with belief propagation," *IEEE Trans. Aerosp. Electron. Syst.*, vol. 50, no. 4, pp. 2942–2959, Oct. 2014.

[23] F. Meyer and M. Z. Win, "Scalable data association for extended object tracking," *IEEE Trans. Signal Inf. Process. Netw.*, vol. 6, pp. 491–507, May 2020.

[24] D. Gaglione, G. Soldi, P. Braca, G. De Magistris, F. Meyer, and F. Hlawatsch, "Classification-aided multitarget tracking using the sum-product algorithm," *IEEE Signal Process. Lett.*, vol. 27, pp. 1710–1714, Sep. 2020.

[25] F. Meyer and M. Z. Win, "Data association for tracking extended targets," in *Proc. IEEE Mil. Commun. Conf. (MILCOM)*, Nov. 2019, pp. 337–342.

[26] F. Meyer, Z. Liu, and M. Z. Win, "Scalable probabilistic data association with extended objects," in *Proc. IEEE Int. Conf. Commun. Workshops (ICC Workshops)*, May 2019, pp. 1–6.

[27] M. Brambilla et al., "Cooperative localization and multitarget tracking in agent networks with the sum-product algorithm," *IEEE Open J. Signal Process.*, vol. 3, pp. 169–195, Mar. 2022.

[28] H.-A. Loeliger, J. Dauwels, J. Hu, S. Korl, L. Ping, and F. R. Kschischang, "The factor graph approach to model-based signal processing," *Proc. IEEE*, vol. 95, no. 6, pp. 1295–1322, Jul. 2007.

[29] F. Meyer, P. Braca, P. Willett, and F. Hlawatsch, "A scalable algorithm for tracking an unknown number of targets using multiple sensors," *IEEE Trans. Signal Process.*, vol. 65, no. 13, pp. 3478–3493, Mar. 2017.

[30] N. Gordon, D. Salmond, and A. Smith, "Novel approach to nonlinear/non-Gaussian Bayesian state estimation," *Proc. Inst. Elect. Eng. F Radar Signal Process.*, vol. 140, no. 2, pp. 107–113, Apr. 1993.

[31] T. Fortmann, Y. Bar-Shalom, and M. Scheffe, "Sonar tracking of multiple targets using joint probabilistic data association," *IEEE J. Ocean. Eng.*, vol. OE8, no. 3, pp. 173–184, Jul. 1983.

[32] R. W. Sittler, "An optimal data association problem in surveillance theory," *IEEE Trans. Mil. Electron.*, vol. MIL-8, no. 2, pp. 125–139, Apr. 1964.

[33] R. Singer, R. Sea, and K. Housewright, "Derivation and evaluation of improved tracking filter for use in dense multitarget environments," *IEEE Trans. Inf. Theory*, vol. IT-20, no. 4, pp. 423–432, Jul. 1974.

[34] Y. Bar-Shalom, "Tracking methods in a multitarget environment," *IEEE Trans. Autom. Control*, vol. AC-23, no. 4, pp. 618–626, Aug. 1978.

[35] S. Bartoletti, A. Giorgetti, M. Z. Win, and A. Conti, "Blind selection of representative observations for sensor radar networks," *IEEE Trans. Veh. Technol.*, vol. 64, no. 4, pp. 1388–1400, Apr. 2015.

[36] M. Brambilla et al., "Sensors localization and target tracking in underwater environment via belief propagation," in *Proc. IEEE Mil. Commun. Conf. (MILCOM)*, Dec. 2021, pp. 641–646.

[37] M. Brambilla, M. Nicoli, G. Soatti, and F. Deflorio, "Augmenting vehicle localization by cooperative sensing of the driving environment: Insight on data association in urban traffic scenarios," *IEEE Trans. Intell. Transp. Syst.*, vol. 21, no. 4, pp. 1646–1663, Apr. 2020.

[38] P. Sharma, A.-A. Saucan, D. J. Bucci, and P. K. Varshney, "Decentralized Gaussian filters for cooperative self-localization and multi-target tracking," *IEEE Trans. Signal Process.*, vol. 67, no. 22, pp. 5896–5911, Nov. 2019.

[39] S. Stankovic, N. Ilic, and M. S. Stankovic, "Adaptive consensus-based distributed system for multisensor multitarget tracking," *IEEE Trans. Aerosp. Electron. Syst.*, vol. 58, no. 3, pp. 2164–2179, Dec. 2022.

[40] R. Olfati-Saber, J. A. Fax, and R. M. Murray, "Consensus and cooperation in networked multi-agent systems," *Proc. IEEE*, vol. 95, no. 1, pp. 215–233, Mar. 2007.

[41] G. Papa, R. Repp, F. Meyer, P. Braca, and F. Hlawatsch, "Distributed Bernoulli filtering using likelihood consensus," *IEEE Trans. Signal Inf. Process. Netw.*, vol. 5, no. 2, pp. 218–233, Jun. 2019.

[42] F. Meyer, O. Hlinka, H. Wymeersch, E. Riegler, and F. Hlawatsch, "Distributed localization and tracking of mobile networks including noncooperative objects," *IEEE Trans. Signal Inf. Process. Netw.*, vol. 2, no. 1, pp. 57–71, Dec. 2016.

[43] H. Winner, S. Hakuli, F. Lotz, and C. Singer, *Handbook of Driver Assistance Systems*. Amsterdam, The Netherlands: Springer International Publishing, 2014.

[44] D. Gruyer, V. Magnier, K. Hamdi, L. Claussmann, O. Orfila, and A. Rakotonirainy, "Perception, information processing and modeling: Critical stages for autonomous driving applications," *Annu. Rev. Contr.*, vol. 44, pp. 323–341, Oct. 2017.

[45] F. Kschischang, B. Frey, and H.-A. Loeliger, "Factor graphs and the sum-product algorithm," *IEEE Trans. Inf. Theory*, vol. 47, no. 2, pp. 498–519, Feb. 2001.

[46] T. E. Fortmann, Y. Bar-Shalom, and M. Scheffe, "Multi-target tracking using joint probabilistic data association," in *Proc. 19th IEEE Conf. Decis. Contr. Symp. Adaptive Processes*, Dec. 1980, pp. 807–812.

[47] D. Musicki and R. Evans, "Linear joint integrated probabilistic data association - LJIPDA," in *Proc. 41st IEEE Conf. Decis. Contr.*, Dec. 2002, vol. 3, pp. 2415–2420.

[48] D. Reid, "An algorithm for tracking multiple targets," *IEEE Trans. Autom. Control*, vol. AC-24, no. 6, pp. 843–854, Dec. 1979.

[49] R. Mahler, "'Statistics 102' for multisource-multitarget detection and tracking," *IEEE J. Sel. Topics Signal Process.*, vol. 7, no. 3, pp. 376–389, Jun. 2013.

[50] M. Kalandros and L. Pao, "Multisensor covariance control strategies for reducing bias effects in interacting target scenarios," *IEEE Trans. Aerosp. Electron. Syst.*, vol. 41, no. 1, pp. 153–173, Jan. 2005.

Chapter 4. Graph-aware Learning

[51] D. Musicki and B. La Scala, "Multi-target tracking in clutter without measurement assignment," *IEEE Trans. Aerosp. Electron. Syst.*, vol. 44, no. 3, pp. 877–896, Jul. 2008.

[52] S. Nagappa and D. E. Clark, "On the ordering of the sensors in the iterated-corrector probability hypothesis density (PHD) filter," in *Proc. Signal Process., Sens. Fusion, Target Recognit. XX*, I. Kadar, Ed., Bellingham, WA, USA: SPIE, May 2011, vol. 8050, pp. 275–280.

[53] R. Mahler, "Approximate multisensor CPHD and PHD filters," in *Proc. 13th Int. Conf. Inf. Fusion*, Jul. 2010, pp. 1–8.

[54] M. Jiang, W. Yi, R. Hoseinnezhad, and L. Kong, "Distributed multi-sensor fusion using generalized multi-Bernoulli densities," in *Proc. 19th Int. Conf. Inf. Fusion (FUSION)*, Jul. 2016, pp. 1332–1339.

[55] A.-A. Saucan, M. J. Coates, and M. Rabbat, "A multisensor multi-Bernoulli filter," *IEEE Trans. Signal Process.*, vol. 65, no. 20, pp. 5495–5509, Oct. 2017.

[56] L. Gao, G. Battistelli, and L. Chisci, "Event-triggered distributed multitarget tracking," *IEEE Trans. Signal Inf. Process. Netw.*, vol. 5, no. 3, pp. 570–584, Sep. 2019.

[57] M. Frohle, C. Lindberg, K. Granstrom, and H. Wymeersch, "Multisensor poisson multi-Bernoulli filter for joint target-sensor state tracking," *IEEE Trans. Intell. Veh.*, vol. 4, no. 4, pp. 609–621, Dec. 2019.

[58] S. He, H.-S. Shin, and A. Tsourdos, "Distributed joint probabilistic data association filter with hybrid fusion strategy," *IEEE Trans. Instrum. Meas.*, vol. 69, no. 1, pp. 286–300, Jan. 2020.

[59] A. K. Gostar et al., "Centralized cooperative sensor fusion for dynamic sensor network with limited field-of-view via labeled multi-Bernoulli filter," *IEEE Trans. Signal Process.*, vol. 69, pp. 878–891, Dec. 2021.

[60] A. Perera, C. Srinivas, A. Hoogs, G. Brooksby, and W. Hu, "Multi-object tracking through simultaneous long occlusions and split-merge conditions," in *Proc. IEEE Comput. Soc. Conf. Comput. Vision Pattern Recognit. (CVPR)*, Jul. 2006, vol. 1, pp. 666–673.

[61] C. Huang, B. Wu, and R. Nevatia, "Robust object tracking by hierarchical association of detection responses," in *Proc. Eur. Conf. Comput. Vision*., Marseille, France: Springer, Oct. 2008, pp. 788–801.

[62] L. Zhang, Y. Li, and R. Nevatia, "Global data association for multi-object tracking using network flows," in *Proc. IEEE Conf. Comput. Vision Pattern Recognit.*, Aug. 2008, pp. 1–8.

[63] J. Berclaz, F. Fleuret, E. Turetken, and P. Fua, "Multiple object tracking using k-shortest paths optimization," *IEEE Trans. Pattern Anal. Mach. Intell.*, vol. 33, no. 9, pp. 1806–1819, Feb. 2011.

[64] B. Wu and R. Nevatia, "Detection and tracking of multiple, partially occluded humans by Bayesian combination of edgelet based part detectors," *Int. J. Comput. Vision*, vol. 75, no. 2, pp. 247–266, Aug. 2007.

[65] G. Shu, A. Dehghan, O. Oreifej, E. Hand, and M. Shah, "Part-based multiple-person tracking with partial occlusion handling," in *Proc. IEEE Conf. Comput. Vision Pattern Recognit.*, Jul. 2012, pp. 1815–1821.

[66] F. Scarselli, M. Gori, A. C. Tsoi, M. Hagenbuchner, and G. Monfardini, "Computational capabilities of graph neural networks," *IEEE Trans. Neural Netw.*, vol. 20, no. 1, pp. 81–102, Jan. 2009.

[67] F. Scarselli, M. Gori, A. C. Tsoi, M. Hagenbuchner, and G. Monfardini, "The graph neural network model," *IEEE Trans. Neural Netw.*, vol. 20, no. 1, pp. 61–80, Dec. 2009.

[68] V. Garcia Satorras and M. Welling, "Neural enhanced belief propagation on factor graphs," in *Proc. 24th Int. Conf. Artif. Intell. Statist.*, A. Banerjee and K. Fukumizu, Eds., San Diego, CA, USA: PMLR, Apr. 2021, vol. 130, pp. 685–693.

[69] M. Liang and F. Meyer, "Neural enhanced belief propagation for cooperative localization," in *Proc. IEEE Statist. Signal Process. Workshop (SSP)*, Aug. 2021, pp. 326–330.

[70] J. Zhou et al., "Graph neural networks: A review of methods and applications," *AI Open*, vol. 1, pp. 57–81, Sep. 2020.

[71] K. Yoon et al., "Inference in probabilistic graphical models by graph neural networks," in *Proc. 53rd Asilomar Conf. Signals, Syst., Comput.* Pacific Grove, CA, USA: IEEE, Nov. 2019, pp. 868–875.

[72] G. Braso and L. Leal-Taixe, "Learning a neural solver for multiple object tracking," in *Proc. IEEE/CVF Conf. Comput. Vision Pattern Recognit.*, Jun. 2020, pp. 6246–6256.

[73] L. Barbieri, B. C. Tedeschini, M. Brambilla, and M. Nicoli, "Implicit vehicle positioning with cooperative lidar sensing," in *Proc. IEEE Int. Conf. Acoust., Speech Signal Process. (ICASSP)*, Jun. 2023, pp. 1–5.

[74] Y. Zhang et al., "ByteTrack: Multi-object tracking by associating every detection box," in *Proc. Eur. Conf. Comput. Vision., Tel Aviv, Israel:* Springer, 2022, Oct. pp. 1–21.

[75] B. C. Tedeschini, M. Brambilla, L. Barbieri, and M. Nicoli, "Addressing data association by message passing over graph neural networks," in *Proc. 25th Int. Conf. Inf. Fusion (FUSION)*, Jul. 2022, pp. 1–7.

[76] A. Dosovitskiy, G. Ros, F. Codevilla, A. Lopez, and V. Koltun, "CARLA: An open urban driving simulator," in *Proc. Conf. Robot Learn.*, Mountain View, CA, USA: PMLR, Nov. 2017, pp. 1–16.

[77] A. H. Lang, S. Vora, H. Caesar, L. Zhou, J. Yang, and O. Beijbom, "Pointpillars: Fast encoders for object detection from point clouds," in *Proc. IEEE/CVF Conf. Comput. Vision Pattern Recognit. (CVPR)*, Jun. 2019, pp. 12689–12697.

[78] H. Su et al., "SPLATNet: Sparse lattice networks for point cloud processing," in *Proc. IEEE Conf. Comput. Vision Pattern Recognit. (CVPR)*, Jun. 2018, pp. 2530–2539.

[79] Z. Yang, Y. Sun, S. Liu, and J. Jia, "3DSSD: Point-based 3D single stage object detector," in *Proc. IEEE/CVF Conf. Comput. Vision Pattern Recognit. (CVPR)*, Jun. 2020, pp. 11037–11045.

[80] M. Liang and F. Meyer, "Neural enhanced belief propagation for multiobject tracking," Dec. 2022, *arXiv:2212.08340*.

[81] J. Gilmer, S. S. Schoenholz, P. F. Riley, O. Vinyals, and G. E. Dahl, "Neural message passing for quantum chemistry," in *Proc. Int. Conf. Mach. Learn.* Sydney, Australia: PMLR, Aug. 2017, pp. 1263–1272.

[82] P. Battaglia et al., "Relational inductive biases, deep learning, and graph networks," Oct. 2018, *arXiv:1806.01261*.

[83] P. Battaglia et al., "Interaction networks for learning about objects, relations and physics," in *Proc. Adv. Neural Inf. Process. Syst. (NIPS Conf.)*, Oct. 2016, vol. 29, pp. 4502–4510.

[84] D. K. Duvenaud et al., "Convolutional networks on graphs for learning molecular fingerprints," in *Proc. Adv. Neural Inf. Process. Syst. (NIPS Conf.)*, Dec. 2015, vol. 28, pp. 2224–2232.

[85] S. Sukhbaatar, A. Szlam, and R. Fergus, "Learning multiagent communication with backpropagation," in *Proc. Adv. Neural Inf. Process. Syst. (NIPS Conf.)*, Oct. 2016, vol. 29, pp. 2244–2252.

[86] A. Vaswani et al., "Attention is all you need," in *Proc. Adv. Neural Inf. Process. Syst. (NIPS Conf.)*, Dec. 2017, vol. 30, pp. 5998–6008.

[87] Y. Li, R. Yu, C. Shahabi, and Y. Liu, "Diffusion convolutional recurrent neural network: Data-driven traffic forecasting," Feb. 2017, *arXiv:1707.01926*.

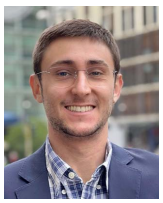
[88] Z. Cui, K. Henrickson, R. Ke, and Y. Wang, "Traffic graph convolutional recurrent neural network: A deep learning framework for network-scale traffic learning and forecasting," *IEEE Trans. Intell. Transp. Syst.*, vol. 21, no. 11, pp. 4883–4894, Nov. 2020.

[89] Y. Wang, Y. Sun, Z. Liu, S. E. Sarma, M. M. Bronstein, and J. M. Solomon, "Dynamic graph CNN for learning on point clouds," *ACM Trans. Graph.*, vol. 38, no. 5, pp. 1–12, Oct. 2019.

[90] E. Arnold, S. Mozaffari, and M. Dianati, "Fast and robust registration of partially overlapping point clouds," *IEEE Robot. Autom. Lett.*, vol. 7, no. 2, pp. 1502–1509, Apr. 2022.

[91] D. P. Kingma and J. Ba, "Adam: A method for stochastic optimization," Jan. 2017, *arXiv:1412.6980*.

[92] S. Luo and W. Hu, "Score-based point cloud denoising," in *Proc. IEEE/CVF Conf. Comput. Vision Pattern Recognit.*, Oct. 2021, pp. 4583–4592.



Bernardo Camajori Tedeschini (Graduate Student Member, IEEE) received the B.Sc. (Hons.) in computer science and M.Sc. (Hons.) degrees in telecommunications engineering from the Politecnico di Milano, Italy, in 2019 and 2021, respectively. From November 2021, he started as a Ph.D. fellow in information technology with the Dipartimento di Elettronica, Informazione e Bioingegneria (DEIB), Politecnico di Milano. He is currently a Visiting Researcher with the Laboratory for Information & Decision Systems at the Massachusetts Institute of Technology (MIT), Cambridge, MA, USA. His research interests include federated learning, machine learning, and localization methods. He was a recipient of the Ph.D. grant from the ministry of the Italian government Ministero dell'Istruzione, dell'Università e della Ricerca (MIUR) and the Roberto Rocca Doctoral Fellowship granted by MIT and Politecnico di Milano.

Mattia Brambilla (Member, IEEE) received the B.Sc. and M.Sc. degrees in telecommunication engineering, and the Ph.D. degree (*cum laude*) in information technology from the Politecnico di Milano, in 2015, 2017, and 2021, respectively. He was a Visiting Researcher with the NATO Centre for Maritime Research and Experimentation (CMRE), La Spezia, Italy, in 2019. In 2021, he joined as Research Fellow with the Faculty of Dipartimento di Elettronica, Informazione e Bioingegneria (DEIB) at the Politecnico di Milano. His research interests include signal processing, statistical learning, and data fusion for cooperative localization and communication. He was the recipient of the Best Student Paper Award at the 2018 IEEE Statistical Signal Processing Workshop.

Luca Barbieri (Member, IEEE) received the B.Sc. and M.Sc. (*cum laude*) degrees in telecommunication engineering and the Ph.D. degree (*cum laude*) in information technology from the Politecnico di Milano, in 2017, 2019, and 2023, respectively. He was a Visiting Researcher with the King's Communications, Learning & Information Processing (KCLIP) laboratory at King's College London, London, U.K., in 2022. He is currently a Research Assistant with the Dipartimento di Elettronica, Informazione e Bioingegneria (DEIB), Politecnico di Milano. His current research interests focus on machine learning, federated learning, and localization methods for vehicular and industrial networks.

Gabriele Balducci received the M.Sc. degree (*cum laude*) in telecommunication engineering with specialization in Signal and Data Analysis, from Politecnico di Milano, in 2022. In the same year, he concluded his path as an Alta Scuola Politecnica student, XVII cycle. Currently, he is an employee in R&D at Nokia Italia cultivating his passion for Statistical Signal Processing.

Monica Nicoli (Senior Member, IEEE) received the M.Sc. (Hons.) and Ph.D. degrees in communication engineering from the Politecnico di Milano, Milan, Italy, in 1998 and 2002, respectively. She was a Visiting Researcher with ENI Agip, from 1998 to 1999, and Uppsala University, in 2001. In 2002, she joined as a Faculty Member with the Politecnico di Milano. She is currently an Associate Professor in telecommunications with the Department of Management, Economics and Industrial Engineering. Her research interests include signal processing, machine learning, and wireless communications, with emphasis on smart mobility and Internet of Things (IoT) applications. She was a recipient of the Marisa Bellisario Award, in 1999, and a corecipient of the best paper awards of the IEEE Symposium on Joint Communications and Sensing, in 2021, the IEEE Statistical Signal Processing Workshop, in 2018, and the *IET Intelligent Transport Systems* journal, in 2014. She is an Associate Editor of the IEEE TRANSACTIONS ON INTELLIGENT TRANSPORTATION SYSTEMS. She has also served as an Associate Editor for the *EURASIP Journal on Wireless Communications and Networking*, from 2010 to 2017, and a Lead Guest Editor for the Special Issue on Localization in Mobile Wireless and Sensor Networks, in 2011.

Open Access funding provided by 'Politecnico di Milano' within the CRUI CARE Agreement.

Message Passing Neural Network Versus Message Passing Algorithm for Cooperative Positioning

Bernardo Camajori Tedeschini, *Graduate Student Member, IEEE*, Mattia Brambilla, *Member, IEEE*, and Monica Nicoli, *Senior Member, IEEE*

***Abstract*—Cooperative Positioning (CP) relies on a network of connected agents equipped with sensing and communication technologies to improve the positioning performance of standalone solutions. In this paper, we develop a completely data-driven model combining Long Short-Term Memory (LSTM) and Message Passing Neural Network (MPNN) for CP, where agents estimate their states from inter-agent and ego-agent measurements. The proposed LSTM-MPNN model is derived by exploiting the analogy with the probability-based Message Passing Algorithm (MPA), from which the graph-based structure of the problem and message passing scheme are inherited. In our solution, the LSTM block predicts the motion of the agents, while the MPNN elaborates the node and edge embeddings for an effective inference of the agents' state. We present numerical evidence that our approach can enhance position estimation, while being at the same time an order of magnitude less complex than typical particle-based implementations of MPA for non-linear problems. In particular, the presented LSTM-MPNN model can reduce the error on agents' positioning to one third compared to MPA-based CP, it holds a higher convergence speed and better exploits cooperation among agents.**

***Index Terms*—Message passing neural network, message passing algorithm, belief propagation, cooperative positioning, LSTM, message passing.**

I. Introduction

A. Contextualization and Background

SIGNAL processing techniques operating over centralized or distributed network architectures have been largely studied in the past, especially for Situation Awareness (SA) applications [1], [2], [3], [4]. The main application domains include Internet of Things (IoT) [5], Connected Autonomous Vehicles (CAVs) [6], [7] and Maritime Situational Awareness (MSA) [8], [9]. These applications are critical as they require sensors (hereafter generally referred as agents) monitoring and perceiving their surroundings and making informed decisions based on the perceived information. The key aspect is the cooperation among agents which enables Cooperative Positioning (CP) techniques and enhances the perception of the environment.

The Message Passing Algorithm (MPA), also known as Belief Propagation (BP) or Sum-Product Algorithm (SPA) [10], [11], is a probabilistic iterative technique which has gained a lot of interest in the field of CP [12] given its ability of linearly scaling with the number of agents [13]. MPA has been largely employed in a different number of SA frameworks, mainly addressing the Multiple Object Tracking (MOT) problem with static or mobile sensing agents [14], [15], [16], [17], [18], [19], [20], [21], [22], [23], [24], embedding or not the measurement to target association problem [25], [26], [27], [28].

B. Related Works

MPA attains optimal performances in case of linear models and Gaussian processes, where the iteratively computed marginal posterior belief converges to the exact marginal posterior distribution. When the conditions of linearity and Gaussianity are not met, particle-based MPA can be employed, although this typically results in a notable increase of computational and communication expenses (i.e., due to particles' sharing and aggregation). Some works tried to improve performances of particle-based MPA implementations by reducing the particle degeneracy in dense and large networks [29], [30] or by auto-tuning the parameters of time-varying system models [31]. However, they did not resolve the main issue of MPA, which is related to the convergence of the beliefs.

Since MPA involves a repeated exchange of information (i.e., an iterative message passing) over a graph that is representative of the considered problem, the intrinsic cyclic structure of graphs leads the MPA's outcome to be only an approximation of the true marginal posterior distribution as the algorithm converges to a local optimum [32], [33], [34], [35]. Specifically, the approximation of beliefs can be considered satisfactory if the optimization problem is locally convex. To improve the performances, Neural Enhanced Belief Propagation (NEBP) have been recently proposed [36], [37], [38], [39], wherein MPA and Message Passing Neural Network (MPNN) are combined to rectify errors caused by cycles and model mismatch.

The MPNN [40], [41] is an extension of Neural Network (NN) customized to work on graph structures. Indeed, in conventional MPNN, a NN is present in each node and edge of the graph, elaborating the input features through an iterative

Manuscript received 16 April 2023; revised 27 June 2023; accepted 18 August 2023. Date of publication 23 August 2023; date of current version 8 December 2023. The associate editor coordinating the review of this article and approving it for publication was G. Ding. *(Corresponding author: Bernardo Camajori Tedeschini.)*

Bernardo Camajori Tedeschini and Mattia Brambilla are with the Dipartimento di Elettronica, Informazione e Bioingegneria, Politecnico di Milano, 20133 Milan, Italy (e-mail: bernardo.camajori@polimi.it; mattia.brambilla@polimi.it).

Monica Nicoli is with the Dipartimento di Ingegneria Gestionale, Politecnico di Milano, 20133 Milan, Italy (e-mail: monica.nicoli@polimi.it).

Digital Object Identifier 10.1109/TCCN.2023.3307953

message passing scheme. The elaborated features, i.e., node and edge embeddings, are usually taken as input to perform a specific task, like node/edge regression or classification. Given their similarity with the message passing in MPA, they have been used within the NEBP framework to address the problems of Data Association (DA) [39], CP [37] and also MOT [38], as well as with the implicit cooperative positioning framework [42]. However, NEBP approaches require performing both iterations of MPNN and MPA, increasing the already high computational time of particle-based methods. Furthermore, it has been demonstrated that in cases where sufficient training data are available, MPNN exhibit superior performance to MPA on cyclic graphs [43], while at the same time being scalable and able to learn non-linear dependencies.

C. Contribution

In this paper, we propose an MPNN solution that can be used as an efficient alternative of MPA in high-complexity problems. We made a first attempt in this direction in [44] where we employed MPNN for solving DA in sensor networks. Here we generalize the analysis by investigating the parallelism between MPA and MPNN, and comparing their performances in the challenging context of dynamic CP.

Mobile CP systems rely on a dynamic model for describing the temporal evolution of the agent locations and a graph model for modeling the inter-agent measurements. Here we propose an NN architecture that combines a Long Short-Term Memory (LSTM) for dynamics modeling and an MPNN for the computation of the marginal likelihoods. The LSTM learns the motion model of agents in time, while the iterative update of estimates based on measurements is obtained with the MPNN.

The main contributions of this paper are as follows:

- definition of a theoretical framework based on the analogy between MPA and MPNN, with focus on the definition of exchanged messages, iterative processing steps and inference prediction;
- proposal of an LSTM-MPNN model which completely replaces MPA for the task of CP. The model is trained using a centralized approach, while it is able to perform a completely distributed inference after deployment;
- comparison with the conventional particle-based MPA, with particular focus on positioning performances and generalization properties.

D. Paper Organization

This paper is organized as follows. Section II is devoted to the description of the adopted system model. Section III first describes the MPA for CP, giving the main steps of the algorithm, and then defines the proposed LSTM-MPNN model with a one-to-one parallelism with MPA. Lastly, it provides insights on distributed inference and centralized training procedures. Section IV presents the simulation scenario and implementation details, followed by simulation results, while Section V draws the conclusions.

II. System Model

We denote with $\mathcal{I}_n = \{1, \dots, I_n\}$ a set of connected agents at timestep n. The connectivity graph between agents is denoted with $\mathcal{G}_n = (\mathcal{V}_n, \mathcal{E}_n)$, where each node $i \in \mathcal{V}_n$ corresponds to an agent, while the edge (i, j), with $i \neq j$, indicates the presence of a communication link from agent i to agent j. Note that the graph is directed, i.e., edges (i, j) and (j, i) differ, and might not necessarily be contemporary present. Each agent $i \in \mathcal{I}_n$ communicates with the set $\mathcal{N}_{i,n}$ of its neighbors and it is described by the state $\mathbf{x}_{i,n}$, which includes kinematic parameters such as posiDon and velocity.. The motion model of agent i from time $n-1$ to time n is described by:

$$\mathbf{x}_{i,n} = f^{(\mathrm{x})}\left(\mathbf{x}_{i,n-1}, \mathbf{w}^{(\mathrm{x})}_{i,n-1}\right), \tag{1}$$

where $\mathbf{w}^{(\mathrm{x})}_{i,n-1}$ is the driving noise process that accounts for motion uncertainty. The derived state-transition probability density function (pdf) is indicated with $p(\mathbf{x}_{i,n}|\mathbf{x}_{i,n-1})$, which, at time $n = 0$, coincides with the prior pdf $p(\mathbf{x}_{i,0})$.

Each agent has access to two types of measurements: a partial and noisy observation $\mathbf{z}^{(\mathrm{A})}_{i,n} = f^{(\mathrm{A})}(\mathbf{x}_{i,n}, \mathbf{w}^{(\mathrm{A})}_{i,n})$ of its own state vector, and an inter-agent measurement $\mathbf{z}^{(\mathrm{A2A})}_{j\to i,n} = f^{(\mathrm{A2A})}(\mathbf{x}_{j,n}, \mathbf{x}_{i,n}, \mathbf{w}^{(\mathrm{A2A})}_{i,n})$, $\forall j \in \mathcal{N}_{i,n}$, where $\mathbf{w}^{(\mathrm{A})}_{i,n}$ and $\mathbf{w}^{(\mathrm{A2A})}_{i,n}$ are the state and inter-agent measurement noises, respectively. The functions $f^{(\mathrm{A})}(\cdot)$ and $f^{(\mathrm{A2A})}(\cdot)$, jointly with the statistics of noises $\mathbf{w}^{(\mathrm{A})}_{i,n}$ and $\mathbf{w}^{(\mathrm{A2A})}_{i,n}$, define the likelihood functions $p(\mathbf{z}^{(\mathrm{A})}_{i,n}|\mathbf{x}_{i,n})$ and $p(\mathbf{z}^{(\mathrm{A2A})}_{j\to i,n}|\mathbf{x}_{j,n}, \mathbf{x}_{i,n})$, respectively. The driving processes and measurement noises are assumed to be independent across agent pairs (i, j) and over time n. We indicate with $\mathbf{x}_n = \{\mathbf{x}_{i,n}\}_{i=1}^{I_n}$ the set of state vectors of all agents at time n, while the two set of measurements are indicated with $\mathbf{z}^{(\mathrm{A})}_n = \{\mathbf{z}^{(\mathrm{A})}_{i,n}\}_{i\in\mathcal{I}_n}$ and $\mathbf{z}^{(\mathrm{A2A})}_n = \{\mathbf{z}^{(\mathrm{A2A})}_{j\to i,n}\}_{i\in\mathcal{I}_n, j\in\mathcal{N}_{i,n}}$. The overall set of measurements at time n is $\mathbf{z}_n = \{\mathbf{z}^{(\mathrm{A})}_{i,n}, \mathbf{z}^{(\mathrm{A2A})}_{i,n}\}$.

CP aims at estimating the states of agents from all the aggregated measurements up to time n, i.e., $\mathbf{z}^{(\mathrm{A})}_{1:n}$ and $\mathbf{z}^{(\mathrm{A2A})}_{1:n}$. The estimated state is indicated with $\widehat{\mathbf{x}}_n$. Probabilistic Bayesian methods, such as MPA, use the marginal posterior pdf $p(\mathbf{x}_{i,n}|\mathbf{z}^{(\mathrm{A})}_{1:n}, \mathbf{z}^{(\mathrm{A2A})}_{1:n})$ to estimate $\widehat{\mathbf{x}}_n$, e.g., through the Minimum Mean Square Error (MMSE) estimator $\widehat{\mathbf{x}}^{(\mathrm{MMSE})}_{i,n} = \int \mathbf{x}_{i,n}\, p(\mathbf{x}_{i,n}|\mathbf{z}^{(\mathrm{A})}_{1:n}, \mathbf{z}^{(\mathrm{A2A})}_{1:n})\, \mathrm{d}\mathbf{x}_{i,n}$ [18]. On the other hand, discriminative probabilistic approaches, like Deep Learning (DL), directly define the posterior pdf with parametric model, i.e., $p(\mathbf{x}_{i,n}|\mathbf{z}^{(\mathrm{A})}_{1:n}, \mathbf{z}^{(\mathrm{A2A})}_{1:n}) = p(\mathbf{x}_{i,n}|\mathbf{z}^{(\mathrm{A})}_{1:n}, \mathbf{z}^{(\mathrm{A2A})}_{1:n}, \boldsymbol{\theta})$, and try to find the parameter vector $\boldsymbol{\theta}$ that maximizes $\widehat{\mathbf{x}}_{i,n} = \mathbb{E}_{\mathbf{x}_{i,n}}[p(\mathbf{x}_{i,n}|\mathbf{z}^{(\mathrm{A})}_{1:n}, \mathbf{z}^{(\mathrm{A2A})}_{1:n}, \boldsymbol{\theta})]$ [45]. This is done using as input a training dataset $\mathcal{S}^{\mathrm{train}} = \{(\mathbf{x}_n, \mathbf{z}^{(\mathrm{A})}_n, \mathbf{z}^{(\mathrm{A2A})}_n)\}_{n=1}^{N_{\mathrm{train}}}$ and minimizing the negative log-likelihood, i.e., $\boldsymbol{\theta} = \arg\min_{\boldsymbol{\theta}} [-\log(p(\mathbf{x}_{i,n}|\mathbf{z}^{(\mathrm{A})}_{1:n}, \mathbf{z}^{(\mathrm{A2A})}_{1:n}, \boldsymbol{\theta}))]$.

A compact representation of the temporal evolution of the system model is reported in Fig. 1, where two different network topologies (i.e., different measurement availability) at time n and $n+1$ are illustrated. The purpose of the figure is to highlight the temporal sequence of CP and visualize different combinations of the graph $\mathcal{G}_n$.

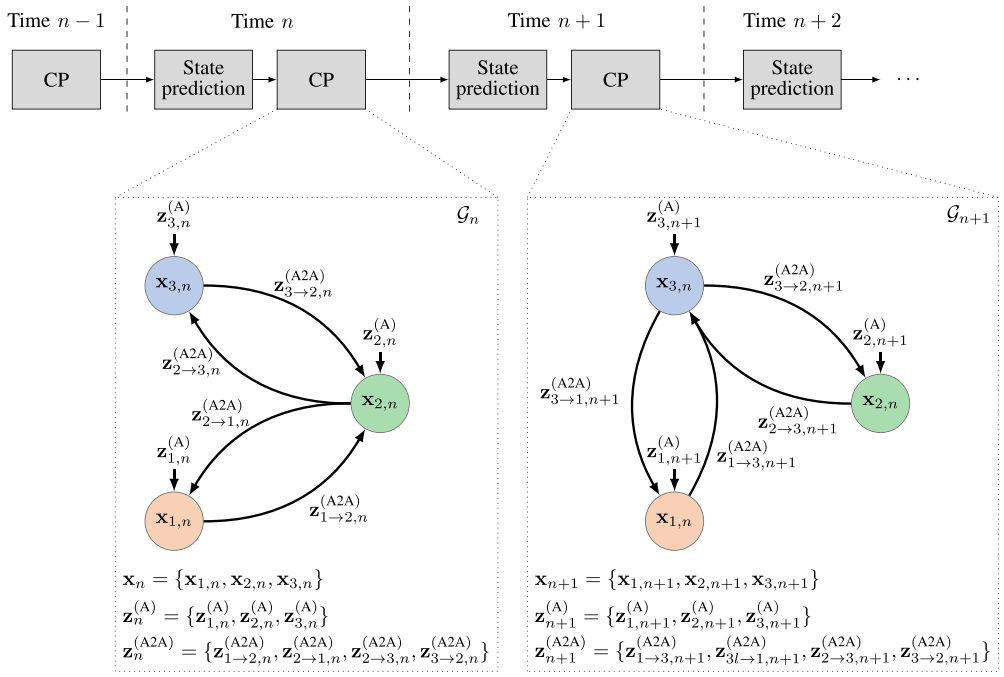


Fig. 1. Illustration of the working principle of CP, with highlighted state vectors and measurement sets for two consecutive time instants. The figure highlights the variation of the graph $\mathcal{G}_n$ due to varied network topology and sets of measurements.

III. Cooperative Positioning Methods

In this section, we first review the MPA Bayesian solution for CP and then we perform a one-to-one comparison with our newly proposed LSTM-MPNN model. Lastly, a description of the inference and training procedure is given.

A. MPA-Based CP

The agent's marginal posterior probability $p(\mathbf{x}_{i,n}|\mathbf{z}_{1:n}^{(\mathrm{A})}, \mathbf{z}_{1:n}^{(\mathrm{A2A})})$ can be obtained by marginalizing the joint posterior pdf $p(\mathbf{x}_{0:n}|\mathbf{z}_{1:n}^{(\mathrm{A})}, \mathbf{z}_{1:n}^{(\mathrm{A2A})})$, where $\mathbf{x}_{0:n} = \{\mathbf{x}_{n'}\}_{n'=0}^{n}$. Assuming statistical independence across agents at timestep $n = 0$ and adopting Bayes' rule, the joint posterior pdf is:

$$p\left(\mathbf{x}_{0:n}|\mathbf{z}_{1:n}^{(\mathrm{A})}, \mathbf{z}_{1:n}^{(\mathrm{A2A})}\right) \propto \prod_{i=1}^{I_n} p\left(\mathbf{x}_{i,0}\right) \prod_{n'=1}^{n} p\left(\mathbf{x}_{i,n'}|\mathbf{x}_{i,n'-1}\right) p\left(\mathbf{z}_{i,n'}^{(\mathrm{A})}|\mathbf{x}_{i,n'}\right) \prod_{j\in\mathcal{N}_{i,n'}} p\left(\mathbf{z}_{j\to i,n'}^{(\mathrm{A2A})}|\mathbf{x}_{j,n'}, \mathbf{x}_{i,n'}\right). \quad (2)$$

Since computing the marginalization of (2) can be unfeasible or extremely complex, the MPA addresses this issue by approximating the marginal posterior with an iterative message passing scheme over a factor graph which factorizes the joint posterior pdf in (2). Denoting the beliefs of agent i at timestep n and message passing iteration $t \in \{1, \ldots, T\}$ with $\mathbf{b}_{i,n}^{(t)} \triangleq \mathbf{b}_i^{(t)}(\mathbf{x}_{i,n}) \approx p(\mathbf{x}_{i,n}|\mathbf{z}_{1:n}^{(\mathrm{A})}, \mathbf{z}_{1:n}^{(\mathrm{A2A})})$, the MPA-based CP performs the following operations in parallel for each agent.

1) *Prediction message:* The predicted state of agent i is represented by the message:

$$\mu_{i,\overrightarrow{n}}(\mathbf{x}_{i,n}) \propto \int p(\mathbf{x}_{i,n}|\mathbf{x}_{i,n-1})\, \mathbf{b}_{i,n-1}^{(T)} \mathrm{d}\mathbf{x}_{i,n-1}, \quad (3)$$

where $\mathbf{b}_{i,n-1}^{(T)}$ is the agent's belief computed at previous time $n-1$ after T message passing steps. Note that the beliefs are initialized at time $n = 0$ as $\mathbf{b}_{i,0}^{(T)} \triangleq p(\mathbf{x}_{i,0})$.

2) *Beliefs exchange:* During message passing iteration $t \in \{1, \ldots, T\}$, each agent i broadcasts $\mathbf{b}_{i,n}^{(t-1)}$ and receives $\mathbf{b}_{j,n}^{(t-1)}$ from its neighbors $j \in \mathcal{N}_{i,n}$. At $t = 1$, the exchanged beliefs are $\mathbf{b}_{i,n}^{(0)} = \mu_{i,\overrightarrow{n}}(\mathbf{x}_{i,n})$.

3) *Measurement messages computation:* During message passing iteration $t \in \{1, \ldots, T\}$, each agent i computes two measurements messages (one for each type of measurement) as:

$$\mu_{i,n}^{(t)(\mathrm{A})}(\mathbf{x}_{i,n}) \triangleq p\left(\mathbf{z}_{i,n}^{(\mathrm{A})}|\mathbf{x}_{i,n}\right), \quad (4)$$

$$\mu_{j\to i,n}^{(t)(\mathrm{A2A})}(\mathbf{x}_{i,n}) \propto \int p\left(\mathbf{z}_{j\to i,n}^{(\mathrm{A2A})}|\mathbf{x}_{j,n}, \mathbf{x}_{i,n}\right) \mathbf{b}_{j,n}^{(t-1)} \mathrm{d}\mathbf{x}_{j,n} \quad \forall j \in \mathcal{N}_{i,n}. \quad (5)$$

4) *Beliefs update:* At message passing iteration $t \in \{1, \dots, T\}$, the beliefs are updated as:

$$\mathbf{b}_{i,n}^{(t)} \propto \mu_{i,\overline{n}}(\mathbf{x}_{i,n})\, \mu_{i,n}^{(t)(\mathrm{A})}(\mathbf{x}_{i,n}) \prod_{j \in \mathcal{N}_{i,n}} \mu_{j \to i,n}^{(t)(\mathrm{A2A})}(\mathbf{x}_{i,n}). \quad (6)$$

5) *State inference*: Lastly, after T message passing steps, the state of agent i is estimated with the MMSE estimator as:

$$\widehat{\mathbf{x}}_{i,n} = \mathbb{E}\Big[\mathbf{b}_{i,n}^{(t)}\Big]. \quad (7)$$

Step 1) is indicated as *prediction step* and it is computed once per timestep n. On the contrary, steps 2), 3) and 4) are called *update steps* as they involve the measurements available at current timestep n and they are performed for all T message passing iterations per each timestep n.

For graphs with a tree structure, the MPA provides exact approximation of the beliefs, which coincide with the true marginal posterior pdf [10]. However, for cyclic graphs, MPA only provides a reasonably accurate approximation of the marginal posterior with a computational complexity that linearly scales with the number of agents I_n and message passing iterations T. Moreover, in case of non-linear motion or measurement models, particle-based methods are recommended, despite incurring in a significant increase of communication and computational costs.

In comparison, MPNN holds the same time scalability [46], it has fewer parameters and it is able to catch any linear or non-linear relationship between input-output data, outperforming BP on loopy graphs if there is a sufficient amount of training data [43]. However, MPNN does not have the knowledge of features relation between time instants, i.e., each message passing iteration t at timestep n is completely independent with respect to the previous timestep $n-1$. To solve this issue, we propose an LSTM-MPNN model which combines the time-dependent capabilities of the recurrent network as well as the flexibility and scalability of the message passing over NNs.

B. LSTM-MPNN-Based CP

The idea behind the proposed solution is to build an equivalent DL-based model of the MPA-based CP described in Section III-A. We start describing the overall model structure, shown in Fig. 2, and then we analyze each single model block. The proposed architecture is composed of two main components, an LSTM block and an MPNN block. Adopting the same logic of the MPA at prediction step, the LSTM at time n receives in input the output of the MPNN $\widehat{\mathbf{x}}_{i,n-1}$ and predicts the most likely change of feature state according to the learned motion model of the agent. This is done by forwarding the hidden states of the LSTM throughout the timesteps. Therefore, the LSTM represents the equivalent block of (3) in the MPA. On the other hand, the MPNN block is performed over T message passing steps, exactly as the message passing in the MPA, and, at last iteration T, it returns the update of feature states, i.e., $\widehat{\mathbf{x}}_{i,n}$. We remark that, by analogy with MPA, we adopt the MPNN in place of a Graph Neural Network (GNN) since the final prediction in the inference step (7) is a direct function of only the beliefs.

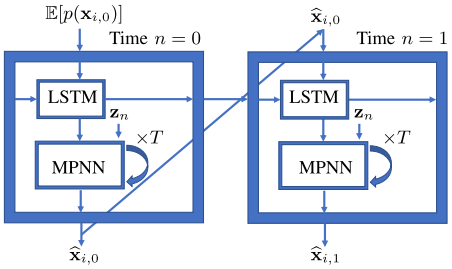


Fig. 2. Block representation of the proposed LSTM-MPNN model.

The MPNN runs on the same physical graph of the agent network, i.e., $\mathcal{G}_n$. It does not create a different graph abstraction, thus it can be computed among the physically connected agents. An MPNN considers two types of features: node embeddings, i.e., $\mathbf{v}_{i,n}^{(t)}$, and edge embeddings, i.e., $\mathbf{e}_{j \to i,n}^{(t)}$. The embeddings, also called attributes, contain elaborated latent information that propagates throughout $\mathcal{G}_n$ at each message passing step t. We can see an analogy between MPA update step and MPNN if we consider the node embeddings $\mathbf{v}_{i,n}^{(t)}$ as elaborated versions of the beliefs $\mathbf{b}_{i,n}^{(t)}$, and the edge embeddings $\mathbf{e}_{j \to i,n}^{(t)}$ as the corresponding measurement messages between agents $\mu_{j \to i,n}^{(t)(\mathrm{A2A})}$.

The proposed MPNN model is composed of NNs for three different functions, encoding of input features ($g_v^{(\mathrm{A})}(\cdot)$ and $g_e^{(\mathrm{A2A})}(\cdot)$), update of node and edge embeddings ($g_v(\cdot)$ and $g_e(\cdot)$) and inference regression ($g_v^{(\mathrm{regres})}$). The encoding of input features is used to extract the most effective representation of measurements $\mathbf{z}_{i,n}^{(\mathrm{A})}$ and $\mathbf{z}_{j \to i,n}^{(\mathrm{A2A})}$ to accomplish the regression task, i.e., agent state estimation. The update of the node and edge embeddings takes the role of (4), (5) and (6), preparing the node embeddings $\mathbf{v}_{i,n}^{(t)}$ for the inference prediction computed by the regressor $g_v^{(\mathrm{regres})}$.

The complete proposed LSTM-MPNN algorithm is shown in Fig. 3 and it is computed by each agent i in parallel.

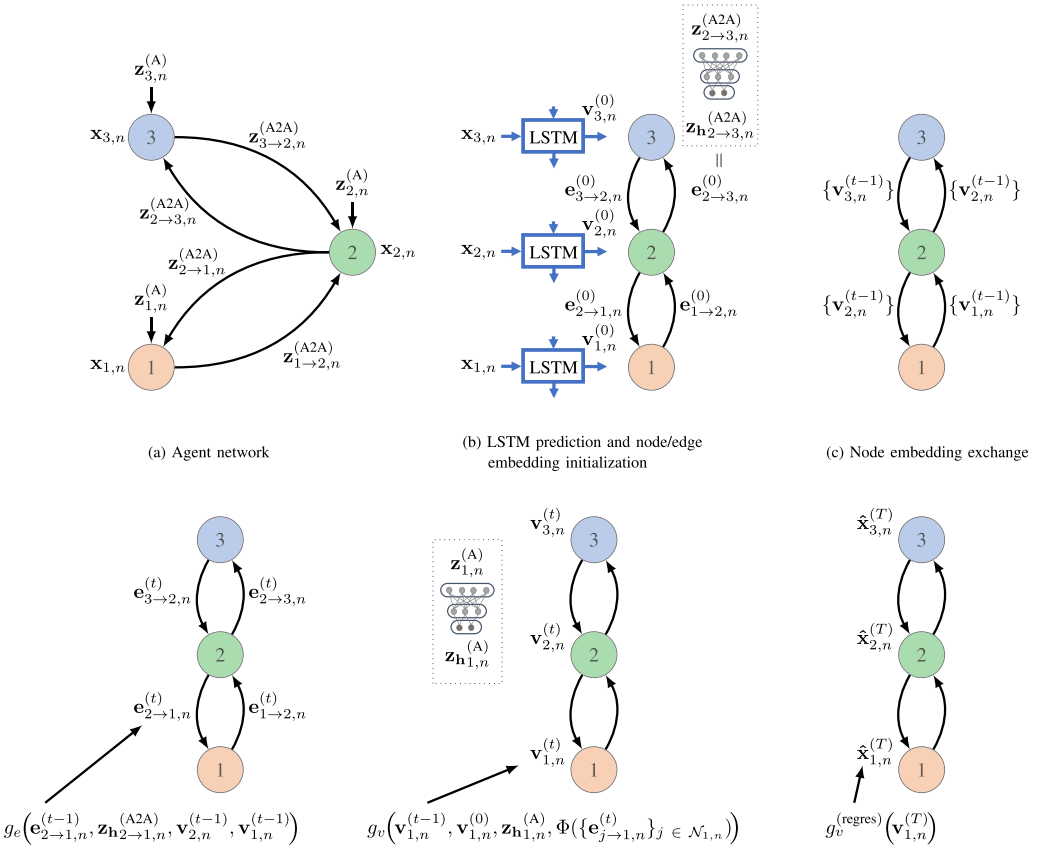
1) *Prediction LSTM:* The LSTM model in agent i predicts the node embeddings $\mathbf{v}_{i,n}^{(t)}$ at time n as:

$$\mathbf{v}_{i,n}^{(0)} = g_v^{(\mathrm{LSTM})}\big(\widehat{\mathbf{x}}_{i,n-1}\big), \quad (8)$$

where $g_v^{(\mathrm{LSTM})}$ is the LSTM model. At $n=0$, the inference is initialized as $\widehat{\mathbf{x}}_{i,n-1} \triangleq \mathbb{E}[p(\mathbf{x}_{i,0})]$. Note that the output of the LSTM coincides with the initialization of the node embeddings at message passing iteration $t=0$. Observing the parallelism with MPA, the belief estimate $\mathbf{b}_{i,n-1}^{(T)}$ is replaced by the state estimate $\widehat{\mathbf{x}}_{i,n-1}$, while the state-transition probability pdf $p(\mathbf{x}_{i,n}|\mathbf{x}_{i,n-1})$ is learned by the LSTM.

2) *Measurements encoding:* At each time n, before starting the message passing, the agent and inter-agent measurements are encoded as:

$$\mathbf{z}_{\mathbf{h}\,i,n}^{(\mathrm{A})} = g_v^{(\mathrm{A})}\Big(\mathbf{z}_{i,n}^{(\mathrm{A})}\Big), \quad (9)$$



(a) Agent network

(b) LSTM prediction and node/edge embedding initialization

(c) Node embedding exchange

(d) Edge embedding update

(e) Node embedding update

(f) State inference

Fig. 3. LSTM-MPNN algorithm for CP. (a) Graph representation of the agent network with agent states and measurements. (b) LSTM prediction at time n and initialization of node and edge embeddings at message passing iteration $t = 0$. (c) Exchange of node embeddings among agents. (d) Update of edge embeddings according to (11). (e) Update of node embeddings according to (12). (f) State inference at time n after T message passing iteration according to (13).

$$\mathbf{z}_{\mathbf{h}_{j\to i,n}}^{(\mathrm{A2A})} = g_e^{(\mathrm{A2A})}\left(\mathbf{z}_{j\to i,n}^{(\mathrm{A2A})}\right), \quad \forall j \in \mathcal{N}_{i,n}. \quad (10)$$

The encoding is necessary to elaborate the input features, it transforms the input measurements into a hidden representation. This is important since all features within the message passing should not belong to the original feature space, but to the hidden space for data privacy reasons. At message passing iteration $t = 1$, the edge embeddings are initialized as: $\mathbf{e}_{j\to i,n}^{(0)} = \mathbf{z}_{\mathbf{h}_{j\to i,n}}^{(\mathrm{A2A})}$.

3) *Node embeddings exchange:* At message passing iteration $t \in \{1, \ldots, T\}$, each agent i broadcasts $\mathbf{v}_{i,n}^{(t-1)}$ and receives $\mathbf{v}_{j,n}^{(t-1)}$ from its neighbors $j \in \mathcal{N}_{i,n}$. Here, the analogy with MPA is straightforward if we compare the beliefs exchange with the node embeddings exchange.

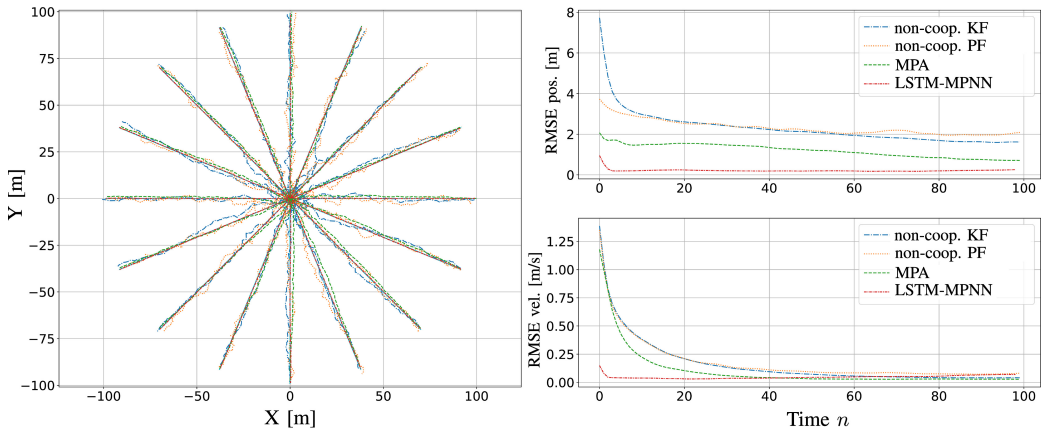
4) *Edge and node embeddings update:* At message passing iteration $t \in \{1, \ldots, T\}$, the edge embeddings are updated as:

$$\mathbf{e}_{j\to i,n}^{(t)} = g_e\left(\mathbf{e}_{j\to i,n}^{(t-1)}, \mathbf{z}_{\mathbf{h}_{j\to i,n}}^{(\mathrm{A2A})}, \mathbf{v}_{j,n}^{(t-1)}, \mathbf{v}_{i,n}^{(t-1)}\right), \quad \forall j \in \mathcal{N}_{i,n}. \quad (11)$$

Note that (11) is the analogous of (5). Subsequently, the node embeddings are updated as:

$$\mathbf{v}_{i,n}^{(t)} = g_v\left(\mathbf{v}_{i,n}^{(t-1)}, \mathbf{v}_{i,n}^{(0)}, \mathbf{z}_{\mathbf{h}_{i,n}}^{(\mathrm{A})}, \Phi\left(\left\{\mathbf{e}_{j\to i,n}^{(t)}\right\}_{j\in\mathcal{N}_{i,n}}\right)\right), \quad (12)$$

where $\Phi(\cdot)$ is called aggregation function, i.e., a function invariant to permutations of its inputs (e.g., element-wise summation, mean, maximum). In the node embeddings update, exactly as in the beliefs update in (6), the initial node embeddings $\mathbf{v}_{i,n}^{(0)}$ are used as a short-connection from the output of the LSTM, i.e., prediction step.



(a) Agent network scenario

(b) Position and velocity RMSE

Fig. 4. Performance evaluation of the proposed LSTM-MPNN for CP. (a) Scenario with 16 moving agents. (b) RMSE of position and velocity over time for the non-cooperative Kalman and particle filters, the cooperative MPA and the proposed LSTM-MPNN.

5) *State inference:* Lastly, after T message passing steps, the regressor NN predicts the state of agent i as:

$$\widehat{\mathbf{x}}_{i,n} = \widehat{\mathbf{x}}_{i,n}^{(T)} = g_v^{(\text{regres})}\left(\mathbf{v}_{i,n}^{(T)}\right). \tag{13}$$

The MMSE estimator in (7) is substituted here by the node regressor $g_v^{(\text{regres})}(\cdot)$ which has the objective of extracting the state prediction from the compact node embeddings.

An interesting fact to point out is that the dimension of the node and edge embeddings, as well as the dimension of the encoded measurements, can be changed according to the problem. As an example, for the case of a state vector described in terms of 2D position and 2D velocity, we need a dimension of eight for the encoding of the node vector, i.e., corresponding to a propagation of a Gaussian belief distribution which holds only two parameters (mean and variance). Increasing the latent feature size leads to a higher complexity of the model which becomes able to learn more complex non-linear dependencies. On the contrary, in particle-based BP, each agent has to exchange a number of parameters equal to the number of adopted particles, each of them with a dimension of the state space, which overall is order of magnitudes higher than the dimension of the latent features in MPNN.

C. Inference and Training Procedure

The proposed LSTM-MPNN model for CP, as the MPA-based CP, is suited for distributed inference as each agent i can rely on a local NNs, i.e., $g_v^{(\text{LSTM})}(\cdot)$, $g_v^{(\text{A})}(\cdot)$, $g_e^{(\text{A2A})}(\cdot)$, $g_v(\cdot)$, $g_e(\cdot)$ and $g_v^{(\text{regres})}(\cdot)$. The physical exchange of embeddings only happens at step 3) of the message passing algorithm (iteration t) and each agent predicts its own state update according to (13). However, for convergence, each NN at each agent should retain the same parameters, as in classical MPNN. This permits a scalable solution to a non-predetermined number of edges, i.e., measurements, and nodes, i.e., agents.

To this aim, we propose a centralized training procedure in which the NNs are firstly trained to learn the CP task and then deployed in an agent network. To compute the training loss and perform back-propagation, we employ the Residual Sum of Squares (RSS) that is estimated at each timestep n and at the end of each message passing iteration t after the regressor prediction $\widehat{\mathbf{x}}_{i,n}^{(t)}$ as:

$$\mathcal{L} = \frac{1}{N}\sum_{n=1}^{N}\frac{1}{|\mathcal{V}_n|}\sum_{t=1}^{T}\sum_{i\ \in\ \mathcal{V}_n}\left\|\widehat{\mathbf{x}}_{i,n}^{(t)} - \mathbf{x}_{i,n}\right\|_2^2, \tag{14}$$

where N is the time sequence length on which the LSTM is trained for tracking. For performance evaluation, we analyze the Root Mean Square Error (RMSE) on the position and velocity of agents.

IV. Simulation Experiments

A. Dataset

We consider a 2D scenario in which $I_n = 16$ connected agents move in an area of 200×200 m for 100 timesteps of 1 s. The agent trajectories form a star shape moving from the origin outwards the area (see Fig. 4a), and the graph $\mathcal{G}_n$ is fully connected. The state of the agents is $\mathbf{x}_{i,n} = [\mathbf{p}_{i,n}^{\mathrm{T}}\dot{\mathbf{p}}_{i,n}^{\mathrm{T}}]^{\mathrm{T}}$, where $\mathbf{p}_{i,n} \in \mathbb{R}^2$ and $\dot{\mathbf{p}}_{i,n} \in \mathbb{R}^2$ are the 2D position and velocity, respectively. The measurements are defined as $\mathbf{z}_{i,n}^{(\text{A})} = \mathbf{x}_{i,n} + \mathbf{w}_{i,n}^{(\text{A})}$ and $\mathbf{z}_{j\to i,n}^{(\text{A2A})} = \|\mathbf{p}_{j,n} - \mathbf{p}_{i,n}\|_2 + \mathbf{w}_{i,n}^{(\text{A2A})}$. Unless otherwise specified, a constant velocity model is used, while the state measurements and inter-agent measurements are zero-mean Gaussian distributed, i.e., $\mathbf{w}_{i,n}^{(\text{A})} \sim \mathcal{N}(\mathbf{0}_4, \mathbf{C}_{\mathbf{w}^{(\text{A})}})$, with $\mathbf{C}_{\mathbf{w}^{(\text{A})}} = \text{diag}(\sigma^2_{\mathbf{p},\mathbf{w}^{(\text{A})}}, \sigma^2_{\mathbf{p},\mathbf{w}^{(\text{A})}}, \sigma^2_{\dot{\mathbf{p}},\mathbf{w}^{(\text{A})}}, \sigma^2_{\dot{\mathbf{p}},\mathbf{w}^{(\text{A})}})$, and

TABLE I
DETAILS ABOUT THE LAYER STRUCTURE OF LSTM AND MLP MODELS

LSTM	
Type	**Output**
Input	4×1
LSTM layer	128×1
LSTM layer	256×1
Maxout	128×1
Linear	64×1
Linear	16×1

$g_v^{(\mathrm{A})}$		$g_v^{(\mathrm{A2A})}$		g_v		g_e		$g_v^{(\mathrm{regres})}$	
Type	**Output**	**Type**	**Output**	**Type**	**Output**	**Type**	**Output**	**Type**	**Output**
Input	4×1	Input	1×1	Input	64×1	Input	64×1	Input	16×1
Linear+GELU	72×1	Linear+GELU	18×1	Linear+GELU	18×1	Linear+GELU	80×1	Linear+GELU	16×1
Linear+GELU	16×1	Linear+GELU	18×1			Linear+GELU	16×1	Linear+GELU	256×1
		Linear+GELU	16×1			Linear+GELU	16×1	Linear+GELU	128×1
								Linear	4×1

$\mathbf{w}_{i,n}^{(\mathrm{A2A})} \sim \mathcal{N}(0, \sigma^2_{\mathbf{w}^{(\mathrm{A2A})}})$, with standard deviations $\sigma_{\mathbf{p},\mathbf{w}^{(\mathrm{A})}} = 5$ m, $\sigma_{\dot{\mathbf{p}},\mathbf{w}^{(\mathrm{A})}} = 1$ m/s and $\sigma_{\mathbf{w}^{(\mathrm{A2A})}} = 2$ m.

For both MPA and MPNN, we consider $T = 10$ message passing iterations. The proposed LSTM-MPNN model has been trained on 10000 instances of constant velocity trajectories, varying $\dot{\mathbf{p}}_{i,n} \in [-10, 10]$ m/s. In order to enhance model convergence and prevent biases, we standardized all the samples by performing a min-max scaler so that each feature lies in [0, 1]. This is done by having a prior knowledge on the agent position, i.e., $\mathbf{p}_{i,n} \in [-100, 100]$ m, and velocity, i.e., $\dot{\mathbf{p}}_{i,n} \in [-10, 10]$ m/s. We highlight that this is not a strong assumption, since to cover higher areas we just need to enlarge the maximum range of the state features. We trained the LSTM-MPNN model for a total of 300 epochs, using a batch size of 32 samples and randomizing the order of the dataset at the beginning of each epoch. Here a sample refers to an instance of trajectories composed of $N = 10$ timesteps, i.e., the training sequence length of the LSTM model.

For the training and testing phases of the model, we used PyTorch version 1.12 and Python version 3.7.11. These operations were conducted on a workstation equipped with an Intel Xeon Silver 4210R CPU, which operates at a frequency of 2.40 GHz. The workstation was also supported by 96 GB of RAM and a Quadro RTX 6000 GPU with 24 GB of memory. For what concerns the optimizer, we used the Adam optimization algorithm [47] with an initial learning rate of 0.0001, and momentum values of 0.9 and 0.999 for β_1 and β_2, respectively.

B. Model and Implementation Details

The LSTM architecture has been inspired by [48], but here we reduced its complexity such that it is constituted by two LSTM layers and a hidden output dimension, i.e., node embeddings, of 16. The complexity reduction is motivated by considering that the state estimation in CP comprises two steps (i.e., prediction and update). For the measurement encoding, update of node and edge embeddings, and state inference, we use Multi-Layer Perceptrons (MLPs) with linear layers and Gaussian Error Linear Unitss (GELUs) activation functions [49]. The complete LSTM and Multi-Layer Perceptrons (MLPs) model structures are reported in Table I.

The selected final architecture of our model was derived upon experimentation, including varying the number of layers and neurons. However, the main rationale behind the general structures is the following. First, the NN encoders $g_v^{(\mathrm{A})}(\cdot)$ and $g_e^{(\mathrm{A2A})}(\cdot)$, despite their small input sizes of 1×1 and 4×1, are characterized by a higher computational complexity when normalized by input size in comparison to the node and edge embedding updates. Second, between $g_v(\cdot)$ and $g_e(\cdot)$, the latter is more complex given its primary role at the initial step of each iteration and the need of processing non-linear inter-agent measurements $\mathbf{z}_{j\to i,n}^{(\mathrm{A2A})}$. Finally, the state inference regressor $g_v^{(\mathrm{regres})}(\cdot)$ is the most challenging task and thus it requires an additional linear layer (4 in total) to effectively predict the state.

C. Simulation Results

1) Tracking Performances: The first test aims at assessing the performances of the proposed LSTM-MPNN model and highlighting the advantages of adopting a data-driven solution. The comparison includes two non cooperative algorithms, i.e., a Kalman Filter (KF) and a Particle Filter (PF), which only use the agent state measurements $\mathbf{z}_{i,n}^{(\mathrm{A})}$, and the cooperative MPA described in Section III-A, which uses the agent state measurements $\mathbf{z}_{i,n}^{(\mathrm{A})}$ and the inter-agent ones $\mathbf{z}_{j\to i,n}^{(\mathrm{A2A})}$, and it is implemented following a particle based approach.

For the particle-based methods, the number of particles is set to $N_{\mathrm{PF}} = 1000$. We would like to point out that the KF represents the optimal non-cooperative case since all noises are Gaussian and all models, i.e., motion and agent state measurements, are linear. On the contrary, the MPA results to be sub-optimal given the non-linearity of inter-agent measurements and the full connectivity of the agent graph.

The results of the comparison are reported in Fig. 4, where we show a realization of the scenario (Fig. 4a) and the RMSE of the position and velocity for each timestep (Fig. 4b) (averaged over 30 simulations). Starting from non-cooperative methods, we notice that the KF is well approximated by the particle-based MPA and reaches a positioning error of 1.62 m while tracking. The cooperative MPA permits to increase the

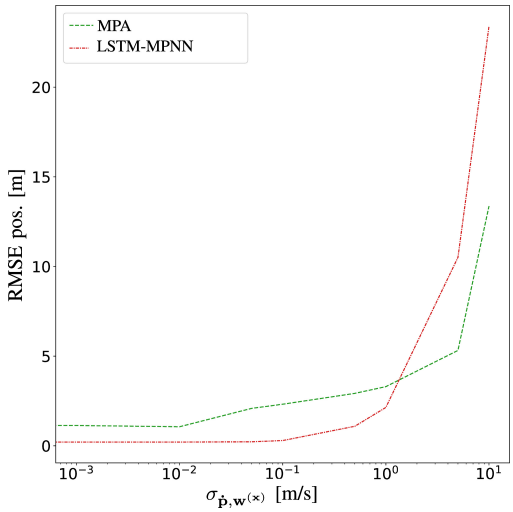


Fig. 5. Analysis of the impact of driving noise standard deviation on the position accuracy for MPA and LSTM-MPNN.

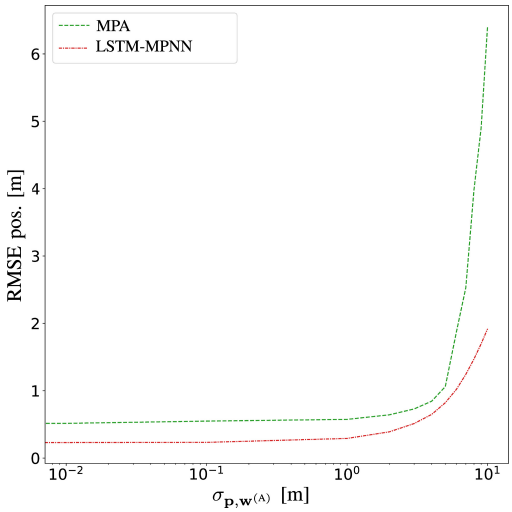


Fig. 6. Comparison of the impact of state-measurement noise error in terms of RMSE of the position between MPA and LSTM-MPNN.

performances by reaching an RMSE on position of 89 cm at convergence. Lastly, the proposed LSTM-MPNN method outperforms all the other methods, achieving an RMSE of 21 cm on the position. Concerning the velocities, all the methods converge at about 0.05 m/s of RMSE. Apart from regime performances, an additional important aspect to consider is the model convergence. Indeed, the LSTM-MPNN method is able to converge after few timesteps, while BP-based algorithms require more time. This feature allows the LSTM-MPNN model to fast react in case of track initialization and recovery after a sudden trajectory variation as it rapidly forgets the previous estimates, updating the state knowledge through LSTM hidden states.

2) Generalization Capabilities: This experiment compares the performances of MPA and LSTM-MPNN under different validation conditions. In particular, we test different intensities of driving process and state-measurement noises. The MPA retains inside the true value of the motion and measurement noises, while the LSTM-MPNN has been trained with noise-free driving processes and measurement models. This is done in order to prove the efficacy of the method with a full-calibrated MPA and a completely miscalibrated LSTM-MPNN.

In a first test, we consider a zero-mean Gaussian-distributed driving noise, i.e., $\mathbf{w}_{i,n}^{(\mathrm{x})} \sim \mathcal{N}(\mathbf{0}_4, \mathbf{C}_{\mathbf{w}^{(\mathrm{x})}})$, with $\mathbf{C}_{\mathbf{w}^{(\mathrm{x})}} = \mathrm{diag}(\sigma^2_{\mathbf{p},\mathbf{w}^{(\mathrm{x})}}, \sigma^2_{\mathbf{p},\mathbf{w}^{(\mathrm{x})}}, \sigma^2_{\dot{\mathbf{p}},\mathbf{w}^{(\mathrm{x})}}, \sigma^2_{\dot{\mathbf{p}},\mathbf{w}^{(\mathrm{x})}})$. In Fig. 5, we compare the MPA and LSTM-MPNN in terms of RMSE on position, with $\sigma_{\mathbf{p},\mathbf{w}^{(\mathrm{x})}} = 0$ m and varying $\sigma_{\dot{\mathbf{p}},\mathbf{w}^{(\mathrm{x})}} \in [0, 10]$ m/s. From the results, we notice that when $\sigma_{\dot{\mathbf{p}},\mathbf{w}^{(\mathrm{x})}} < 0.5$ m/s, the proposed LSTM-MPNN outperforms the particle-based MPA. On the contrary, increasing the noise intensity leads to a faster degradation of performances with respect to the MPA. This is justified by two main factors. Firstly, the model has been trained using error-free trajectories, leading it to anticipate motion models that adhere to the distribution of the training trajectories. Secondly, the increased noise raises the likelihood of encountering an agent with a speed beyond the training range of $[-10, 10]$ m/s, potentially leading to inaccurate predictions.

In a second test, we consider a constant motion model and a varying state-measurement noise, i.e., $\sigma_{\mathbf{p},\mathbf{w}^{(\mathrm{A})}} \in [0, 10]$ m. This time, analyzing the results in Fig. 6, we observe that the LSTM-MPNN achieves a lower RMSE across all considered values of state measurement noise. This confirms the trend that on peak performances, i.e., with same noises and within the same area of cooperation, the proposed LSTM-MPNN model outperforms the cooperative MPA method by reducing the error to one third. Moreover, even with unfavorable conditions, i.e., training on absence of noise, the LSTM-MPNN model better generalizes against noisy state-measurements.

3) Impact on Different Number of Agents: For this last assessment, we evaluate how the different number of cooperative agents affects the performances of the two methods. To this aim, in Fig. 7, we plot the RMSE on the position varying the number of connected agents $I_n \in [2, 22]$. As expected, we observe that, for a low number of agents, the two methods tend to converge to the RMSE achieved for the non-cooperative case, i.e., about 1.5 m. This confirms that with a decreasing number of agents, the LSTM-MPNN model converges to the optimal case of single-agent KF. Increasing I_n, the cooperation plays a crucial role in improving CP, especially for the proposed LSTM-MPNN model. As a matter of fact, in LSTM-MPNN with only 6 cooperative agents the same RMSE of 20 agents for the MPA method is achieved.

4) Computational Complexity: Given the same graph structure and same number of message passing iterations between

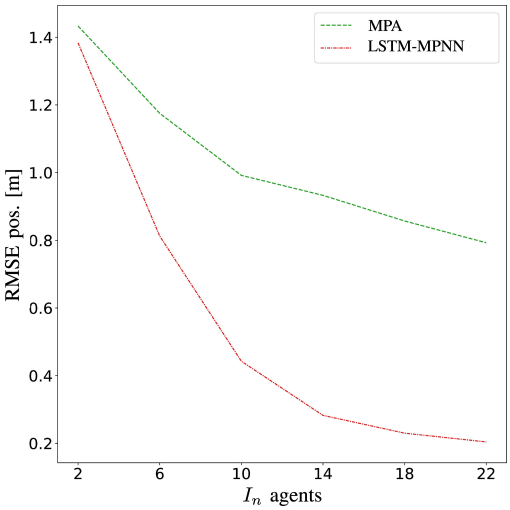


Fig. 7. Comparison of the impact of varying number of cooperative agents in terms of RMSE of the position between MPA and LSTM-MPNN.

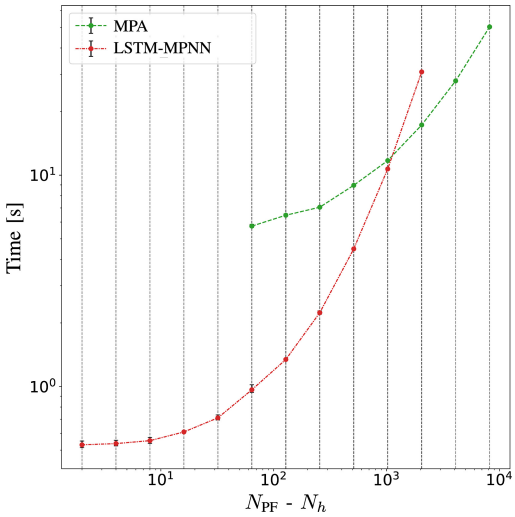


Fig. 8. Comparison of the impact of varying number of particles N_{PF} and node embedding dimension N_h in terms of inference time between MPA and LSTM-MPNN.

MPA and LSTM-MPNN models, the major difference in computational complexity lies in the computation of the prediction and update steps. In order to compare one-to-one the two methods, we define with N_{PF} and N_h the number of particles in MPA and the dimension of the node and edge embeddings, respectively. These variables drive the computational complexity since they tune the trade-off between performances and efficiency. Indeed, N_{PF} and N_h are the dimension of the messages exchanged during each message passing step. Moreover, in MPA, N_{PF} regulates the capability of the model of approximating the distributions according to the importance sampling principle. In LSTM-MPNN, N_h has the same function of N_{PF} in MPA, but with the fundamental difference that here the exchanged vector, i.e., node embedding, does not represent an approximation of the distributions using a sampling mechanism. On the contrary, it represents an effective combination of distribution parameters, e.g., moments, in order to accomplish the CP task.

To this aim, in Fig. 8 we show the whole prediction time of an instance of agent trajectories, i.e., 16 agents moving as shown in Fig. 4a, varying N_{PF} or N_h according to the model. Note that here the time required to exchange the particles and the node embeddings are not considered. Moreover, for a fair comparison, all agent predictions are computed on CPU and in a sequential manner. Observing the results, we notice that the LSTM-MPNN is very efficient for a number of latent dimension $N_h < 100$, performing the whole inference in less than 1 s. On the contrary, the MPA is slower even with $N_{\mathrm{PF}} = 100$ particles. Comparing the two methods for $N_{\mathrm{PF}} = N_h$, we note that, from a pure inference time point of view, it is more convenient to adopt the LSTM-MPNN if $N_{\mathrm{PF}} = N_h < 1000$. However, we would like to point out that, comparing the two methods with the previously adopted $N_{\mathrm{PF}} = 1000$ and $N_h = 16$, we obtain an inference time of 600 ms and 11 s for the LSTM-MPNN and MPA, respectively. Thus, with the proposed method, we reach one third of the error at 1/18 of the time.

V. Conclusion

This paper addressed the problem of CP by proposing an innovative LSTM-MPNN model that can be considered as a promising alternative to conventional probabilistic MPA. Besides providing for the first time a one-to-one parallelism with respect to MPA, we demonstrated the improved performance of a fully DL-based model. We detailed each part of the proposed model, starting from the need of temporal-dependence solved using an LSTM block, up to the message passing structure. The MPNN runs on the same physical graph created by the network of connected agents and it is able to perform inference in a completely distributed way. Mirroring the MPA, the messages, i.e., node embeddings, are exchanged between agents until convergence. Finally, as opposed to the MMSE estimator in MPA, the state inference is carried out through a NN at the node.

We validated the proposed approach in a synthetic network of cooperative agents moving in a scenario over straight trajectories. Numerical results showed that the proposed approach is able to address the problem of CP in an efficient and effective way by outperforming particle-based MPA in a different number of aspects. First, under peak performances point of view, the LSTM-MPNN model reaches a lower RMSE on the position by a factor of 3. Second, the LSTM-MPNN model holds a much higher speed of convergence, an order of magnitude lower computational complexity. As an example, in our experiments, the dimension of the messages exchanged by the MPNN is 16, while the number of particles exchanged by

the BP is 1000. Moreover, the proposed model better handles different state-measurement noises, as well as driving noises if trained on all ranges of state feature values. Finally, the LSTM-MPNN model better exploits the power of cooperation, giving a huge improvement even with small number of cooperating agents.

The value of cooperative positioning is foreseen to dramatically grow over the next several years, especially in the context of automated and connected mobility, where dense networks of agents have to handle complex and dynamic environments. It results that an effective data-driven approach is of paramount importance to enhance positioning capabilities. Our method makes a step toward this direction, by enabling distributed and efficient cooperative inference. Future developments could be implementing not only a distributed inference but also a distributed training, maintaining at the same time the agent's local data privacy. Moreover, applications of fully DL-based methods are foreseen for the major fields of target detection and tracking.

Code Availability Statement

The GitHub repository with the dataset and the Python code for the model, training and inference is available upon request to the corresponding author.

References

[1] M. Z. Win, Y. Shen, and W. Dai, "A theoretical foundation of network localization and navigation," *Proc. IEEE*, vol. 106, no. 7, pp. 1136–1165, Jul. 2018. [Online]. Available: https://ieeexplore.ieee.org/document/8421288/

[2] M. Z. Win, W. Dai, Y. Shen, G. Chrisikos, and H. V. Poor, "Network operation strategies for efficient localization and navigation," *Proc. IEEE*, vol. 106, no. 7, pp. 1224–1254, Jul. 2018. [Online]. Available: https://ieeexplore.ieee.org/document/8421291/

[3] M. Win et al., "Network localization and navigation via cooperation," *IEEE Wireless Commun. Mag.*, vol. 49, no. 5, pp. 56–62, May 2011. [Online]. Available: http://ieeexplore.ieee.org/document/5762798/

[4] D. Gaglione et al., "Bayesian information fusion and multitarget tracking for maritime situational awareness," *IET Radar Sonar Navigat.*, vol. 14, no. 12, pp. 1845–1857, Dec. 2020. [Online]. Available: https://onlinelibrary.wiley.com/doi/10.1049/iet-rsn.2019.0508

[5] A. A. Saucan and M. Z. Win, "Information-seeking sensor selection for Ocean-of-Things," *IEEE Internet Things J.*, vol. 7, no. 10, pp. 10072–10088, Oct. 2020. [Online]. Available: https://ieeexplore.ieee.org/document/9086780/

[6] M. Brambilla, M. Nicoli, G. Soatti, and F. Deflorio, "Augmenting vehicle localization by cooperative sensing of the driving environment: Insight on data association in urban traffic scenarios," *IEEE Trans. Intell. Transp. Syst.*, vol. 21, no. 4, pp. 1646–1663, Apr. 2020. [Online]. Available: https://ieeexplore.ieee.org/document/8848842/

[7] S. Zhang et al., "Distributed direct localization suitable for dense networks," *IEEE Trans. Aerosp. Electron. Syst.*, vol. 56, no. 2, pp. 1209–1227, Apr. 2020. [Online]. Available: https://ieeexplore.ieee.org/document/8762217/

[8] G. Ferri et al., "Cooperative robotic networks for underwater surveillance: An overview," *IET Radar Sonar Navigat.*, vol. 11, no. 12, pp. 1740–1761, Dec. 2017. [Online]. Available: https://onlinelibrary.wiley.com/doi/10.1049/iet-rsn.2017.0074

[9] P. Braca, P. Willett, K. LePage, S. Marano, and V. Matta, "Bayesian tracking in underwater wireless sensor networks with port-starboard ambiguity," *IEEE Trans. Signal Process.*, vol. 62, no. 7, pp. 1864–1878, Apr. 2014. [Online]. Available: http://ieeexplore.ieee.org/document/6737314/

[10] F. Kschischang, B. Frey, and H.-A. Loeliger, "Factor graphs and the sum-product algorithm," *IEEE Trans. Inf. Theory*, vol. 47, no. 2, pp. 498–519, Feb. 2001. [Online]. Available: http://ieeexplore.ieee.org/document/910572/

[11] D. Bickson, O. Shental, and D. Dolev, "Distributed Kalman filter via Gaussian belief propagation," in *Proc. 46th Annu. Allerton Conf. Commun. Control Comput.*, Sep. 2008, pp. 628–635.

[12] H. Wymeersch, J. Lien, and M. Z. Win, "Cooperative localization in wireless networks," *Proc. IEEE*, vol. 97, no. 2, pp. 427–450, Feb. 2009. [Online]. Available: http://ieeexplore.ieee.org/document/4802193/

[13] R. Sánchez-Caucem I. París, and F. J. Díez, "Sum-product networks: A survey," *IEEE Trans. Pattern Anal. Mach. Intell.*, vol. 44, no. 7, pp. 3821–3839, Jul. 2022.

[14] W. Zhang and F. Meyer. "Multisensor multiobject tracking with high-dimensional object states." Dec. 2022. [Online]. Available: https://arxiv.org/abs/2212.14556.

[15] D. Gaglione, P. Braca, G. Soldi, F. Meyer, F. Hlawatsch, and M. Z. Win, "Fusion of sensor measurements and target-provided information in multitarget tracking," *IEEE Trans. Signal Process.*, vol. 70, pp. 322–336, Dec. 2022. [Online]. Available: https://ieeexplore.ieee.org/document/9633194/

[16] F. Meyer and J. Williams, "Scalable detection and tracking of geometric extended objects," *IEEE Trans. Signal Process.*, vol. 69, pp. 6283–6298, Oct. 2021. [Online]. Available: https://ieeexplore.ieee.org/document/9585528/

[17] R. Mendrzik et al., "Joint multitarget tracking and dynamic network localization in the underwater domain," in *Proc. IEEE Int. Conf. Acoust. Speech Signal Process. (ICASSP)*. Barcelona, Spain, May 2020, pp. 4890–4894. [Online]. Available: https://ieeexplore.ieee.org/document/9054047/

[18] F. Meyer et al., "Message passing algorithms for scalable multitarget tracking," *Proc. IEEE*, vol. 106, no. 2, pp. 221–259, Feb. 2018. [Online]. Available: http://ieeexplore.ieee.org/document/8290605/

[19] F. Meyer, P. Braca, P. Willett, and F. Hlawatsch, "A scalable algorithm for tracking an unknown number of targets using multiple sensors," *IEEE Trans. Signal Process.*, vol. 65, no. 13, pp. 3478–3493, Jul. 2017. [Online]. Available: http://ieeexplore.ieee.org/document/7889057/

[20] M. Brambilla et al., "Cooperative localization and multitarget tracking in agent networks with the sum-product algorithm," *IEEE Open J. Signal Process.*, vol. 3, pp. 169–195, Mar. 2022. [Online]. Available: https://ieeexplore.ieee.org/document/9729221/

[21] B. Teague, Z. Liu, F. Meyer, A. Conti, and M. Z. Win, "Network localization and navigation with scalable inference and efficient operation," *IEEE Trans. Mobile Comput.*, vol. 21, no. 6, pp. 2072–2087, Jun. 2022. [Online]. Available: https://ieeexplore.ieee.org/document/9247273/

[22] F. Meyer and M. Z. Win, "Joint navigation and multitarget tracking in networks," in *Proc. IEEE Int. Conf. Commun. Workshops (ICC Workshops)*, Kansas City, MO, USA, May 2018, pp. 1–6. [Online]. Available: https://ieeexplore.ieee.org/document/8403679/

[23] F. Meyer, E. Riegler, O. Hlinka, and F. Hlawatsch, "Simultaneous distributed sensor self-localization and target tracking using belief propagation and likelihood consensus," in *Proc. Conf. Rec. 46th Asilomar Conf. Signals Syst. Comput. (ASILOMAR)*, Nov. 2012, pp. 1212–1216.

[24] E. Leitinger, S. Grebien, and K. Witrisal, "Multipath-based SLAM using belief propagation with interacting multiple dynamic models," in *Proc. 15th Eur. Conf. Antennas Propagat. (EuCAP)*, Dusseldorf, Germany, Mar. 2021, pp. 1–5. [Online]. Available: https://ieeexplore.ieee.org/document/9411320/

[25] F. Meyer and M. Z. Win, "Scalable data association for extended object tracking," *IEEE Trans. Signal Inf. Process. Netw.*, vol. 6, pp. 491–507, May 2020. [Online]. Available: https://ieeexplore.ieee.org/document/9098068/

[26] D. Gaglione, G. Soldi, P. Braca, G. De Magistris, F. Meyer, and F. Hlawatsch, "Classification-aided multitarget tracking using the sum-product algorithm," *IEEE Signal Process. Lett.*, vol. 27, pp. 1710–1714, Sep. 2020. [Online]. Available: https://ieeexplore.ieee.org/document/9206541/

[27] F. Meyer and M. Z. Win, "Data association for tracking extended targets," in *Proc. IEEE Military Commun. Conf. (MILCOM)*, Norfolk, VA, USA, Nov. 2019, pp. 337–342. [Online]. Available: https://ieeexplore.ieee.org/document/9020858/

[28] F. Meyer, Z. Liu, and M. Z. Win, "Scalable probabilistic data association with extended objects," in *Proc. IEEE Int. Conf. Commun. Workshops (ICC Workshops)*, Shanghai, China, May 2019, pp. 1–6. [Online]. Available: https://ieeexplore.ieee.org/document/8757014/

[29] L. Wielandner, E. Leitinger, F. Meyer, B. Teague, and K. Witrisal, "Message passing-based cooperative localization with embedded particle flow," in *Proc. IEEE Int. Conf. Acoust. Speech Signal Process. (ICASSP)*, Singapore, May 2022, pp. 5652–5656. [Online]. Available: https://ieeexplore.ieee.org/document/9747585/

[30] W. Zhang and F. Meyer, "Graph-based multiobject tracking with embedded particle flow," in *Proc. IEEE Radar Conf. (RadarConf21)*. Atlanta, GA, USA, May 2021, pp. 1–6. [Online]. Available: https://ieeexplore.ieee.org/document/9455151/

[31] G. Soldi, F. Meyer, P. Braca, and F. Hlawatsch, "Self-tuning algorithms for multisensor-multitarget tracking using belief propagation," *IEEE Trans. Signal Process.*, vol. 67, no. 15, pp. 3922–3937, Aug. 2019. [Online]. Available: https://ieeexplore.ieee.org/document/8714043/

[32] Y. Weiss and W. T. Freeman, "Correctness of belief propagation in Gaussian graphical models of arbitrary topology," *Neural Comput.*, vol. 13, no. 10, pp. 2173–2200, Oct. 2001. [Online]. Available: https://direct.mit.edu/neco/article/13/10/2173--2200/6465

[33] J. Yedidia, W. Freeman, and Y. Weiss, "Constructing free-energy approximations and generalized belief propagation algorithms," *IEEE Trans. Inf. Theory*, vol. 51, no. 7, pp. 2282–2312, Jul. 2005. [Online]. Available: http://ieeexplore.ieee.org/document/1459044/

[34] M. J. Wainwright and M. I. Jordan, "Graphical models, exponential families, and variational inference," *Found. Trends Mach. Learn.*, vol. 1, nos. 1–2, pp. 1–305, Nov. 2007. [Online]. Available: http://www.nowpublishers.com/article/Details/MAL-001

[35] E. Riegler, G. E. Kirkelund, C. N. Manchon, M.-A. Badiu, and B. H. Fleury, "Merging belief propagation and the mean field approximation: A free energy approach," *IEEE Trans. Inf. Theory*, vol. 59, no. 1, pp. 588–602, Jan. 2013. [Online]. Available: http://ieeexplore.ieee.org/document/6301723/

[36] V. G. Satorras and M. Welling, "Neural enhanced belief propagation on factor graphs," in *Proc. 24th Int. Conf. Artificial Intell. Statist.*, Mar. 2021, pp. 685–693. [Online]. Available: https://proceedings.mlr.press/v130/garcia-satorras21a.html

[37] M. Liang and F. Meyer, "Neural enhanced belief propagation for cooperative localization," in *Proc. IEEE Statist. Signal Process. Workshop (SSP)*, Rio de Janeiro, Brazil, Jul. 2021, pp. 326–330. [Online]. Available: https://ieeexplore.ieee.org/document/9513853/

[38] M. Liang and F. Meyer. "Neural enhanced belief propagation for multiobject tracking." Dec. 2022. [Online]. Available: http://arxiv.org/abs/2212.08340.

[39] M. Liang and F. Meyer, "Neural enhanced belief propagation for data association in multiobject tracking," in *Proc. 25th Int. Conf. Inf. Fusion (FUSION)*, Jul. 2022, pp. 1–7.

[40] F. Scarselli, M. Gori, A. C. Tsoi, M. Hagenbuchner, and G. Monfardini, "Computational capabilities of graph neural networks," *IEEE Trans. Neural Netw.*, vol. 20, no. 1, pp. 81–102, Jan. 2009. [Online]. Available: http://ieeexplore.ieee.org/document/4703190/

[41] F. Scarselli, M. Gori, A. C. Tsoi, M. Hagenbuchner, and G. Monfardini, "The graph neural network model," *IEEE Trans. Neural Netw.*, vol. 20, no. 1, pp. 61–80, Jan. 2009. [Online]. Available: http://ieeexplore.ieee.org/document/4700287/

[42] L. Barbieri, B. Camajori Tedeschini, M. Brambilla, and M. Nicoli, "Implicit vehicle positioning with cooperative lidar sensing," in *Proc. IEEE Int. Conf. Acoust. Speech Signal Process. (ICASSP)*, Jun. 2023, pp. 1–5.

[43] K. Yoon et al., "Inference in probabilistic graphical models by graph neural networks," in *Proc. 53rd Asilomar Conf. Signals Syst. Comput.*, Pacific Grove, CA, USA, Nov. 2019, pp. 868–875.

[44] B. C. Tedeschini, M. Brambilla, L. Barbieri, G. Balducci, and M. Nicoli, "Cooperative lidar sensing for pedestrian detection: Data association based on message passing neural networks," *IEEE Trans. Signal Process.*, early access, Aug. 2023, doi: 10.1109/TSP.2023.3304002.

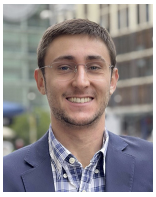
[45] C. M. Bishop, *Pattern Recognition and Machine Learning*. New York, NY, USA: Springer, 2006. [Online]. Available: https://link.springer.com/book/9780387310732

[46] J. Zhou et al., "Graph neural networks: A review of methods and applications," *AI Open*, vol. 1, pp. 57–81, Apr. 2020. [Online]. Available: https://linkinghub.elsevier.com/retrieve/pii/S2666651021000012

[47] D. P. Kingma and J. Ba. "Adam: A method for stochastic optimization." Jan. 2017. [Online]. Available: http://arxiv.org/abs/1412.6980.

[48] J. Liu, Z. Wang, and M. Xu, "Deepmtt: A deep learning maneuvering target-tracking algorithm based on bidirectional LSTM network," *Information. Fusion*, vol. 53, pp. 289–304, Jan. 2020. [Online]. Available: https://linkinghub.elsevier.com/retrieve/pii/S1566253518306122

[49] D. Hendrycks and K. Gimpel. "Gaussian error linear units (GELUs)." Jul. 2020. [Online]. Available: http://arxiv.org/abs/1606.08415.



Bernardo Camajori Tedeschini (Graduate Student Member, IEEE) received the B.Sc. degree (Hons.) in computer science and the M.Sc. degree (Hons.) in telecommunications engineering from the Politecnico di Milano, Italy, in 2019 and 2021, respectively, where he is currently pursuing the Ph.D. degree in information technology with the Dipartimento di Elettronica, Informazione e Bioingegneria.

He is currently a Visiting Researcher with the Laboratory for Information and Decision Systems, Massachusetts Institute of Technology, Cambridge, MA, USA. His research interests include federated learning, machine learning, and localization methods. He was a recipient of the Ph.D. Grant from the Ministry of the Italian Government Ministero dell'Istruzione, dell'Università e della Ricerca and the Roberto Rocca Doctoral Fellowship granted by MIT and the Politecnico di Milano.

Mattia Brambilla (Member, IEEE) received the B.Sc. and M.Sc. degrees in telecommunication engineering and the Ph.D. degree (cum laude) in information technology from the Politecnico di Milano in 2015, 2017, and 2021, respectively. He was a Visiting Researcher with the NATO Centre for Maritime Research and Experimentation, La Spezia, Italy, in 2019. In 2021, he joined the faculty of the Dipartimento di Elettronica, Informazione e Bioingegneria, Politecnico di Milano as a Research Fellow. His research interests include signal processing, statistical learning, and data fusion for cooperative localization and communication. He was the recipient of the Best Student Paper Award at the 2018 IEEE Statistical Signal Processing Workshop.

Monica Nicoli (Senior Member, IEEE) received the M.Sc. (Hons.) and Ph.D. degrees in communication engineering from the Politecnico di Milano, Milan, Italy, in 1998 and 2002, respectively. She was a Visiting Researcher with ENI Agip from 1998 to 1999 and Uppsala University in 2001. In 2002, she joined the Politecnico di Milano as a Faculty Member. She is currently an Associate Professor of Telecommunications with the Department of Management, Economics and Industrial Engineering, Politecnico di Milano.

Her research interests include signal processing, machine learning, and wireless communications, with emphasis on smart mobility and Internet of Things. She was a recipient of the Marisa Bellisario Award in 1999 and a co-recipient of the Best Paper Awards of the EuMA Mediterranean Microwave Symposium in 2022, the IEEE Symposium on Joint Communications and Sensing in 2021, the IEEE Statistical Signal Processing Workshop in 2018, and the IET Intelligent Transport Systems journal in 2014. She is an Associate Editor of the IEEE TRANSACTIONS ON INTELLIGENT TRANSPORTATION SYSTEMS. She has also served as an Associate Editor for the *EURASIP Journal on Wireless Communications and Networking* from 2010 to 2017 and a Lead Guest Editor for the Special Issue on Localization in Mobile Wireless and Sensor Networks in 2011.

Open Access funding provided by 'Politecnico di Milano' within the CRUI CARE Agreement

CHAPTER 5

Federated and Split Learning

In this chapter, we present four works on distributed cooperative learning, namely D-ML and FD-ML. The distributed algorithms are studied in a real FL platform developed in the first paper for performing privacy-preserving brain tumor segmentation in medical networks. The platform is based on the MQTT protocol, and it is flexible to different models, i.e., architectures and sizes, as well as different FL algorithms, i.e., centralized and decentralized FL. Motivated by the open issue of synchronization in FL processes, in the second paper we propose guidelines for designing asynchronous PS orchestration under heterogeneous IoT devices. In particular, we tune the intervals between consecutive global model updates based on sample distributions and computational capabilities, enhancing the accuracy of the system. Then, in the third paper, we address the problem of non-IID data distributions in decentralized FL processes by proposing WAC schemes applied to consensus-based FL. Specifically, we evolve the centralized FedAdp method and introduce three distinct WAC schemes, named CFAdp, tailored for heterogeneous client populations. Results on real IoT devices show an improvement in both convergence and performances under both label and sample data skewness. Finally, in the last paper, the objective was to create a new class of FD-ML algorithms for both distributed learning and inferences under resource-constrained devices. Inspired by the recent introduction of SFL algorithms, we propose a novel server-less SCFL framework that combines the advantages of both SL and consensus-based FL algorithms. SCFL is based on an innovative distributed version of MPNN, which enables privacy-preserving and fully-decentralized learning, as well as low-model complexity for IoT devices and parallel training and testing among agents.

IEEE Access
Multidisciplinary ⋮ Rapid Review ⋮ Open Access Journal

Received December 24, 2021, accepted January 6, 2022, date of publication January 11, 2022, date of current version January 25, 2022.

Digital Object Identifier 10.1109/ACCESS.2022.3141913

Decentralized Federated Learning for Healthcare Networks: A Case Study on Tumor Segmentation

BERNARDO CAMAJORI TEDESCHINI[1], (Graduate Student Member, IEEE), STEFANO SAVAZZI[2], (Member, IEEE), ROMAN STOKLASA[3], LUCA BARBIERI[1], (Graduate Student Member, IEEE), IOANNIS STATHOPOULOS[3,4], MONICA NICOLI[5], (Member, IEEE), AND LUIGI SERIO[3]

[1]Dipartimento di Elettronica, Informazione e Bioingegneria, Politecnico di Milano, 20133 Milano, Italy
[2]Institute of Electronics, Information Engineering and Telecommunication (IEIIT), Consiglio Nazionale delle Ricerche, 20133 Milano, Italy
[3]European Organization for Nuclear Physics (CERN), Technology Department, 1211 Geneve, Switzerland
[4]2nd Department of Radiology, Medical School of Athens, Attiko University Hospital, 12462 Chaidari, Greece
[5]Department of Management, Economics and Industrial Engineering, Politecnico di Milano, 20156 Milano, Italy

Corresponding author: Bernardo Camajori Tedeschini (bernardo.camajori@polimi.it)

This work was supported in part by the European Organization for Nuclear Physics (CERN) Budget for Knowledge Transfer for the Benefit of Medical Applications.

ABSTRACT Smart healthcare relies on artificial intelligence (AI) functions for learning and analysis of patient data. Since large and diverse datasets for training of Machine Learning (ML) models can rarely be found in individual medical centers, classical centralized AI requires moving privacy-sensitive data from medical institutions to data centers that process the fused information. Training on data centers thus requires higher communication resource/energy demands while violating privacy. This is considered today as a significant bottleneck in pursuing scientific collaboration across trans-national clinical medical research centers. Recently, federated learning (FL) has emerged as a distributed AI approach that enables the cooperative training of ML models, without the need of sharing patient data. This paper dives into the analysis of different FL methods and proposes a real-time distributed networking framework based on the Message Queuing Telemetry Transport (MQTT) protocol. In particular, we design a number of solutions for ML over networks, based on FL tools relying on a parameter server (PS) and fully decentralized paradigms driven by consensus methods. The proposed approach is validated in the context of brain tumor segmentation, using a modified version of the popular U-NET model with representative clinical datasets obtained from the daily clinical workflow. The FL process is implemented on multiple physically separated machines located in different countries and communicating over the Internet. The real-time test-bed is used to obtain measurements of training accuracy vs. latency trade-offs, and to highlight key operational conditions that affect the performance in real deployments.

INDEX TERMS Federated learning, learning over networks, medical imaging, healthcare networks, network architectures, machine learning.

I. INTRODUCTION

Deep learning (DL) and Artificial Intelligence (AI) have great potential in clinical research as a means for integrating complex imaging data into personalized indices of diagnosis and prognosis. Combined with the human pathologist's inputs, AI systems have contributed to significantly reduce the human error rate [1]. On the other hand, the increasing volume of data, the widespread adoption of Internet-of-Medical-Things (IoMT) [2] and AI-enabled devices with high computing capabilities, have made conventional centralized (Big-Data) learning solutions inefficient in terms of latency and scalability due to the need of moving, often periodically, large datasets. Besides, regulatory authorities as well as patient organizations are proposing stringent limitations to AI-driven data processing, to ensure that private data are not shared or transferred to third parties, even in anonymized format [3]. In such a dynamic context, Federated Learning (FL) technology [4] has been emerging as a viable solution [5]–[7]. The technology enables the distributed training of Machine Learning (ML) models over remote devices, namely the Medical Nodes (MNs), or clients,

The associate editor coordinating the review of this manuscript and approving it for publication was Larbi Boubchir.

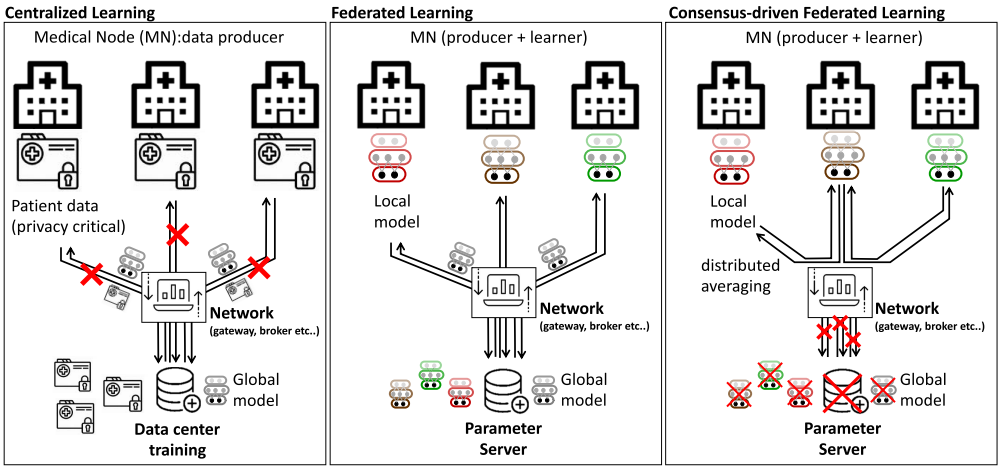


FIGURE 1. From left to right: Centralized Learning (CL), Federated Learning (FL) coordinated by the Parameter Server (PS), namely Federated Averaging (FA), and Consensus-driven learning with fully decentralized implementation (i.e., without PS).

without requiring the same devices to disclose their training data, possible containing privacy sensitive information about patients.

As shown in Figure 1, vanilla FL algorithms, such as Federated Averaging (FA) [4], [8], allow the MNs to learn a shared ML model under the orchestration of a Parameter Server (PS). Typically, the PS interacts with the medical devices through a network, i.e., using a provider or gateway, to collect (and store) the received local models. These are aggregated to obtain a global model that is then fed back to the edge devices for validation and inference. Each Medical Node (MN) thus participates in training the shared model using its own dataset. However, in contrast to classical Centralized Learning (CL), privacy-sensitive data are kept on the device, while cooperation is based on local model exchange.

FL uses the data as and when they are received or available at the MN and it thus supports flexible training processes, such as continual and incremental learning. First implementations of FL leveraged on a server-client architecture [6], [7] where the PS coordinates the learning process. On the other hand, these classical FL techniques are often considered not always resilient against model inversion attacks on the PS, where privacy-critical data can be recreated using the local models stored by the PS [9], [10].

A. RELATED WORKS

Different FL implementations have emerged in the past few years [11] targeting several application scenarios [12]–[16] and technology enablers [17]–[19]. Focusing on the popular brain tumor image segmentation challenge [20], first steps towards the integration of a privacy-preserving FL system into a medical image analysis framework are in [6], [13]. It was demonstrated that the FL model quality is comparable to that of a model trained using CL on a data fusion center. The dataset therein used to build and test the AI system is the multimodal Brain Tumor Segmentation (BraTS) set [21]–[23]. The FL process typically utilizes the same training pipeline designed for centralized training: current state of the art models for 3D brain Magnetic resonance imaging (MRI) processing are based on an auto-encoder regularization tool [24], or the U-NET model [25], [26] with hyperparameter structure described in [27].

The above approaches have two main limits. First, they only used datasets prepared ad-hoc for testing; they did not consider how real data coming from hospitals affect the training process or how to generalize the model. Second, they rely on a central fusion center for model aggregation which could lead to privacy leaks. Decentralized training on incomplete and heterogeneous image datasets (i.e., different scan modalities) poses new challenges to FL and is the main focus of this paper. For what concerns the algorithms, the training process can be implemented via vanilla FL tools that rely on the PS for distributed coordination. However, trusted PS designs are needed [28]. As an alternative, fully distributed learning tools have been recently proposed to replace, or minimize the use of the PS functions, enabling server-less training. These techniques have roots in consensus [15] and distributed ledger [29] enablers, as they let the local models be consensually shared and synchronized across multiple MNs. They rely solely on in-network processing, via consensus, diffusion [18], [30], [31] or gossip [32] tools. Fully decentralized FL policies have been considered for training on low-power devices (robots, drones) in several industrial verticals [15], [33] such as robotics, connected automated vehicles [14], [34] and medical diagnosis [35]. Network scalability/connectivity aspects are however not considered or discussed.

Though FL approaches are promising, they are often simulated on virtual frameworks [11] where (virtual) clients act as independent threads and run on the same physical

machine. With the exception of [12], [36], [37], pilot demonstration of FL platforms featuring geographically distributed devices and real-time training over the Internet are currently overlooked. In line with the road-map towards native (in-network) AI designs [38], in this paper we propose a real-time platform to support network and federated learning functions integration, validating the proposed FL solution in a real-world deployment.

B. CONTRIBUTIONS

The paper proposes the application of decentralized FL methods in the context of cancer diagnosis, focusing in particular on brain tumor segmentation. To demonstrate the system in a real environment, a novel Message Queuing Telemetry Transport (MQTT) based architecture has been developed. The platform is employed to verify the performance of FL over real geographical distributed MNs characterized by non-uniform computing capabilities and heterogeneous datasets. Both classical FL based on PS designs and fully decentralized architectures are evaluated, discussing for each case their impact on the MQTT publishing/subscription operations. The proposed networking architecture and tools are designed to optimize the FL process, weaving together synchronous and asynchronous operations, as well as taking into account the training time of the individual clients to avoid performance penalties caused by slower MNs, namely the straggler effect [11], [39]. The algorithms and MQTT real-time network have been demonstrated by combining for the first time, in a decentralized FL approach, public (BraTS) and private clinical data obtained from the clinical workflow (with no pre-filtering).

The main contributions of the paper are further summarized as follows:

- Vanilla and fully decentralized FL algorithms are integrated into a novel network architecture that adopts the MQTT transport protocol to orchestrate the deep ML model parameters exchange. We propose an optimized set of information to be embedded into the MQTT payload and to characterize the real-time learning process on each epoch, discussing also model parameters compression, serialization and Quality-of-Service (QoS) mechanisms.
- Implementing FL tools on top of the MQTT protocol brings novel challenges that are discussed here for the first time. In particular, 4 mechanisms for ML parameter exchange are proposed: these account for synchronous and asynchronous operations on the MNs and the PS, respectively, as well as decentralized FL, where the clients, rather than the PS, self-organize to coordinate the FL process. For all the considered cases, the MQTT broker is configured to support the client authentication, authorization as well as to control the access to FL resources (global/aggregated models, training statistics and timing). All the proposed architectures are compared to quantify the latency/model quality trade-offs for synchronous and asynchronous FL processes.
- Validation of the FL tools is based on a federation of 5 MNs distributed in different institutions across the Europe and communicating over the Internet. Focusing on brain tumor segmentation as case study, the experiments are conducted in real clinical settings and with medical MRI images obtained from the daily clinical workflow. The proposed real-time testbed thus provides a unique opportunity to quantify the improvements of the FL process in terms of practical metrics, namely the Dice Similarity Coefficient (DSC), and on heterogeneous datasets without any pre-filtering and not prepared for testing purposes (in contrast to public and widely available data).

The paper is organized as follows. Section II introduces the FL algorithms, namely vanilla and fully decentralized tools based on consensus and analyze them with respect to medical imaging problems and privacy considerations. Section III discusses the proposed brain tumor segmentation tasks and the necessary adaptations for FL system deployment. Section IV describes the specifications of the proposed networking architecture and MQTT protocol integrated designs. Targeting tumor segmentation, section V highlights a case study with an extensive database of results obtained from public and private patient datasets. Finally, conclusions and open issues are summarized in Section VI.

II. FEDERATED LEARNING METHODS

The algorithms analyzed in this section range from vanilla FL tools, such as Federated Averaging (FA), relying on the orchestration of the PS, to fully decentralized FL, namely Consensus-driven Federated Averaging (CFA), based on distributed coordination. In server-based FL systems, the data owners (i.e., the MN clients) and the global model owner (i.e., the PS) are the two major entities. On the other hand, in fully decentralized tools the PS is replaced by a consensus over the clients, namely the local model owners. For all cases, the data are distributed among N clients rather than being kept centrally, so each data owner $i = 1, \ldots, N$, has a private dataset $\mathcal{D}_i$ of size $S_i = |\mathcal{D}_i|$.

A. VANILLA FL

The FL process generally aims to obtain an optimized global model $\mathbf{w}_G$ that minimizes a global loss function $\mathcal{L}(\cdot)$ decomposed into the sum of local losses as:

$$\mathbf{w}_G = \underset{\mathbf{w}}{\operatorname{argmin}}\, \mathcal{L}(\mathbf{w}) = \underset{\mathbf{w}}{\operatorname{argmin}} \left[\frac{1}{N} \sum_{i=1}^{N} \mathcal{L}_i(\mathbf{w}) \right], \qquad (1)$$

with $\mathcal{L}_i(\cdot)$ being the local loss function observed by client i. Problem (1) is solved iteratively by alternating the optimization of a *local model* at each client, i.e., using a gradient-based method, with a round of communication with the PS to obtain an updated *global model*. In particular, the FL process is characterized by three main steps: *task initialization* (executed only once at beginning), *local model optimization* and *aggregation*. The *task initialization* is implemented

during the first iteration, $t = 0$: the server determines the target task (i.e., the application and the data requirements), as well as the key parameters of the global model and the training process, such as the learning rate or the number of local epochs. The server then broadcasts the initialized global model $\mathbf{w}_{G,t}$ and task to the chosen participants. In the *local model optimization* phase, at iteration $t > 0$ each participant utilizes the local data $\mathcal{D}_i$ and processing capacity to update the local model parameters $\mathbf{w}_{i,t}$ based on the global model $\mathbf{w}_{G,t}$. The aim of participant i is thus to minimize the local loss function, $\mathbf{w}_{i,t} = \text{argmin}_{\mathbf{w}} \mathcal{L}_i(\mathbf{w})$. This is solved via gradient methods, such as Stochastic Gradient Descent (SGD) [40]:

$$\mathbf{w}_{i,t} \leftarrow \mathbf{w}_{i,t} - \eta \nabla \mathcal{L}_i(\mathbf{w}_{i,t}; b_i), \tag{2}$$

where η is the learning rate while $\nabla \mathcal{L}_i(\mathbf{w}_{i,t}; b_i)$ represents the gradient of the loss function with respect to the model $\mathbf{w}_{i,t}$ and it is measured on a data mini-batch $b_i \subseteq \mathcal{D}_i$. The optimized local parameters $\mathbf{w}_{i,t}$ are sent to the PS to be aggregated. In the following *aggregation* step, the PS collects the local models from the clients and feds back an updated version of global model parameters for the next iteration $t+1$, namely $\mathbf{w}_{G,t+1}$. Main aggregation policies are reviewed in [11, Chapter 3]. Finally the clients use the updated global model to update the local optimization (2). The *training rounds*, consisting of the above described local model optimization and aggregation steps, are repeated until each model $\mathbf{w}_{i,t}$ converges to $\mathbf{w}_G$, or a desired training accuracy is obtained.

B. FEDERATED AVERAGING WITH TRUSTED PS

The main parameters to control the computational effort of FL are: the percentage (C) of clients who take part in an update cycle, the number (E) of local epochs executed by each client and the mini batch b_i size (B) used for each local update. The latter one, considering the different resources that each MN may have, can be relaxed and optimized differently in each node. In what follows, the Federated Averaging (FA) algorithm introduced by [41] is tuned for the medical imaging problem.

In the *local model optimization* step, the client runs, for a number of local epochs E, the Adaptive Moment Estimation (Adam) optimizer [42], that exploits first and second order moments to overcome local minima:

$$\mathbf{w}_{i,t} \leftarrow \mathbf{w}_{i,t} - \eta \frac{\sqrt{1-\beta_2^n}}{1-\beta_1^n} \frac{\mathbf{m}_{i,n}}{\sqrt{\mathbf{v}_{i,n}} + \sigma}. \tag{3}$$

β_1 and β_2 are two hyperparameters and the decaying averages $\mathbf{m}_{i,n}$ and $\mathbf{v}_{i,n}$ are computed respectively as follows:

$$\mathbf{m}_{i,n} \leftarrow \beta_1 \mathbf{m}_{i,n-1} + (1-\beta_1) \nabla \mathcal{L}_i(\mathbf{w}_{i,t}; b_i) \tag{4}$$

$$\mathbf{v}_{i,n} \leftarrow \beta_2 \mathbf{v}_{i,n-1} + (1-\beta_2) \nabla^2 \mathcal{L}_i(\mathbf{w}_{i,t}; b_i), \tag{5}$$

where n is the timestep index of the Adam optimizer. Notice that SGD is not recommended on complex models as it needs careful tuning of the learning rate as the training progresses. For image segmentation problems, i.e., tumor segmentation, the number of local epochs E is usually kept small ($E = 1, 2$), as already verified in other works [13], while the batch size B is increased as much as possible to exploit all the parallel computations given by the Graphics Processing Units (GPUs).

The *aggregation* step at round t is performed through a weighted average according to the number of samples $S_i = |\mathcal{D}_i|$ of each client:

$$\mathbf{w}_{G,t+1} = \frac{\epsilon}{\sum_{j=1}^{N} S_j} \sum_{i=1}^{N} S_i \mathbf{w}_{i,t} + (1-\epsilon) \mathbf{w}_{G,t}, \tag{6}$$

where ϵ regulates the memory of the past models in order to have smoothed, or less rapid, changes of the weights. The *aggregation* step is a crucial part of the algorithm and affects the performances. Studies on the adaptive weight function, that regulates the importance of the client's contribution in the global model, have been performed in [19]. However, in the context of medical imaging, this method is less effective since the datasets in the MNs present few variations of the brightness and/or the noise figure.

Much attention instead should be paid to the characterization or customization of the models in each client. For example, FedPer, proposed in [43] splits the layers of the deep learning model into baseline and personalized ones. While the basic layers are collaboratively learned using the traditional FL technique, the personalized ones are learned locally and not shared, i.e. opportunistically, allowing more flexible training of multiple tasks. Adaptation of this technique to more complex models employed for brain tumor segmentation (Section III) should focus on the *encoder* part, which is generally a pre-trained classification network like VGG [44]/ResNet [45]. On the other hand, the *decoder* part could host the personalization layers [46].

C. CONSENSUS-DRIVEN FEDERATED AVERAGING (CFA)

The FA policy discussed in Section II-B relies on the PS orchestration and may be subject to privacy concerns, especially if the PS infrastructure is vulnerable (e.g., untrusted). In these scenarios, medical sites might avoid joining the collaborative training process to protect their privacy-sensitive data despite the benefits introduced by the cooperation. An alternative approach explored in the following is the consensus-driven FA strategy, namely CFA, that provides a solution to the FL problem (1) using a fully distributed and adaptive approach.

In particular, the CFA consists again of an *aggregation* and a *local model optimization* step: however, differently from the server-based FL, both steps are implemented by the MNs on each learning iteration. CFA is thus serverless as the MNs can cooperate with one another without the coordination of the PS. Rather than sending the model updates to the PS, the medical sites can directly forward the ML model parameters to neighboring participants, provided that they are authenticated as members of the pool of FL learners, and using peer-to-peer communication

Algorithm 1 Consensus-Driven FA

procedure CFA($\mathcal{N}_{i,t}$) ▷ Run on client i
  authentication with network broker
  receive parameters (η, E, B) ▷ RX from broker
  initialize $\mathbf{w}_{i,0} \leftarrow$ device i
  initialize $\mathbf{m}_{i,0} \leftarrow 0$
  initialize $\mathbf{v}_{i,0} \leftarrow 0$
  initialize $n \leftarrow 0$ ▷ Adam timestep
  for each round $t = 1, 2, \ldots$ **do** ▷ Training loop
    receive$\{\mathbf{w}_{k,t}\}_{k\in\mathcal{N}_{i,t}}$ ▷ RX from broker
    $Dec\{\mathbf{w}_{k,t}\}_{k\in\mathcal{N}_{i,t}}$ ▷ Decipher weights
    equation (7) ▷ Aggregation step
    $\mathbf{w}_{k,t} = \text{ModelUpdate}(\boldsymbol{\psi}_{i,t})$
    send $Enc\left(\mathbf{w}_{i,t}\right)$ ▷ Encrypt and TX to broker
  end for
end procedure
procedure ModelUpdate($\boldsymbol{\psi}_{i,t}$) ▷ Model opt. step
  $\mathcal{B} \leftarrow$ mini-batches of size B
  for each local epoch $j = 1, 2, \ldots, E$ **do**
    for batch $b \in \mathcal{B}$ **do** ▷ Local Adam
      $n \leftarrow n + 1$
      $\mathbf{m}_{i,n} \leftarrow \beta_1 \mathbf{m}_{i,n-1} + (1-\beta_1)\nabla\mathcal{L}_{i,t}(\boldsymbol{\psi}_{i,t})$
      $\mathbf{v}_{i,n} \leftarrow \beta_2 \mathbf{v}_{i,n-1} + (1-\beta_2)\nabla^2\mathcal{L}_{i,t}(\boldsymbol{\psi}_{i,t})$
      $\boldsymbol{\psi}_{i,t} \leftarrow \boldsymbol{\psi}_{i,t} - \eta\frac{\sqrt{1-\beta_2^n}}{1-\beta_1^n} \cdot \frac{\mathbf{m}_{i,n}}{\sqrt{\mathbf{v}_{i,n}}+\sigma}$
    end for
  end for
end procedure

links. The MNs implement an ad-hoc aggregation step that incorporates into the local model adaptation the information collected from the local neighborhoods. Such aggregation is typically based on consensus [18], [47] or gossip methodologies [48], [49].

The pseudo-code of the CFA can be found in Algorithm 1. First, local model optimization (3) is performed using local data $\mathcal{D}_i$ over a number E of local rounds/epochs which can be tuned depending on the MN computing and energy requirements. The updated model is then sent to neighbors. In the aggregation step, each MN client i implements the average consensus policy to obtain the aggregated model $\boldsymbol{\psi}_{i,t}$ with the help of the neighbors:

$$\boldsymbol{\psi}_{i,t} = \mathbf{w}_{i,t} + \frac{\epsilon_t}{\sum_{j\in\mathcal{N}_{i,t}} S_j} \sum_{k\in\mathcal{N}_{i,t}} S_k \left(\mathbf{w}_{k,t} - \mathbf{w}_{i,t}\right), \qquad (7)$$

where ϵ_t controls the stability of the update and $\mathcal{N}_{i,t}$ contains the neighbors of client i at round t. Notice that similarly as for server-based FL, each client might either defer the model aggregation until the neighbors complete their local model optimization (synchronous implementation) or rather apply the model aggregation as soon as they complete their model optimization, regardless of neighbors status (asynchronous implementation). These aspects are analyzed in more detail in the next sections.

D. COMPARATIVE ANALYSIS AND PRIVACY CONSIDERATIONS

Vanilla FL typically assumes that the PS and the clients are all honest, meaning that the clients are training with their own private data in a good faith, and send true local models to the PS. However, since the PS aggregates the models from all clients, it is an appealing target for possible attackers and thereby a single point of failure in the distributed platform. So, despite the fact that FL can prevent user privacy leaks, the parameter server is nevertheless subject to threats such as server-side training sample reconstructions [9]. Therefore, trusted implementations of the PS are of particular relevance in order to be suitable for the medical imaging problem. For example, differential privacy techniques add a noise term to each local model in order to prevent information from being exposed during the model exchange [50], [51]. The problem of untrusted PS is tackled in this paper at network layer using encrypted and authenticated communications on each learning iteration, in exchange for larger computation/communication overhead.

Besides vanilla FL tools, fully decentralized CFA replaces the PS with consensus and in-network processing directly between the clients (Sect. II-C). Differently from server-based FL, where the PS stores the local models of *all* the participating clients, in CFA the clients implementing consensus are owners of a (small) subset of the local models, namely the ones shared by the neighborhood $\mathcal{N}_{i,t}$. Thereby, it is unlikely that a model inversion attack targeting an individual client could reconstruct the training samples of all the learners. Furthermore, although the CFA architectural approach solves the untrusted PS issues, the untrusted client problem still needs to be carefully considered. For example, distributed ledger technologies, such as blockchain, provide an effective approach for removing the PS, that is vulnerable to attacks [28]. Moreover, it can be also exploited to address the problem of untrusted clients [52], ensuring security of local model updates obtained from authenticated clients. The development of robust decentralized FL designs against adversarial manipulations or data poisoning is however an open problem [53].

III. BRAIN TUMOR SEGMENTATION: FL MODELS AND METRICS

This section describes the brain tumor segmentation and classification task. In particular, we highlight the ML model selected for FL processing as well as relevant loss and accuracy metrics. Tumor segmentation is one of the fundamental tasks in medical diagnosis to support radiologists and clinicians, and also to reduce idle times for evaluating potential treatments. Given a set of MRI slices, the goal of brain tumor segmentation is to extract regions of interest that capture the tumor extent and its shape. This is accomplished by assigning a class label for every pixel (or voxel) in the images, indicating the presence/absence and/or the morphology of the tumor. As an example, Fig. 2 reports

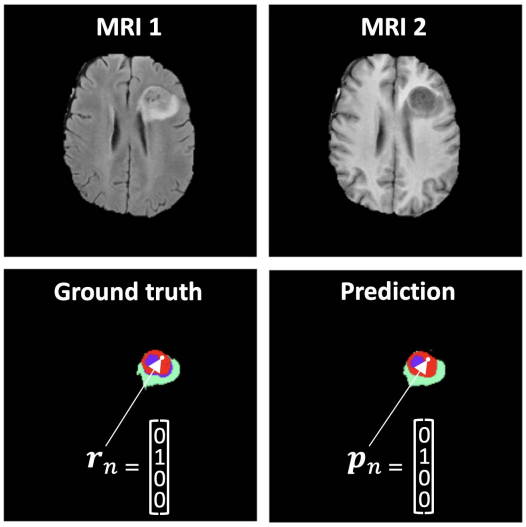


FIGURE 2. Brain tumor segmentation with 2 input layers and 4 segmentation levels: MRI typology 1 (up-left), MRI typology 2 (up-right), ground-truth $r_n = [r_{\ell,n}]_{\ell=1}^{L}$ (down-left), prediction $p_n = [p_{\ell,n}]_{\ell=1}^{L}$ (down-right).

the two input MRI modalities on the top, while the expected ground-truth and predicted tumor segmentation labels are in the bottom-left and bottom-right, respectively. Achieving accurate segmentation labels in this context requires the definition of complex processing systems capable of handling MRI scans with different spatial resolutions, modalities, and levels of noise, according to the specific medical equipment employed.

A. U-NET MODEL AND FL ADAPTATIONS

Current state-of-the-art ML systems for brain tumor segmentation heavily rely on Convolutional Neural Networks (CNN) architectures. Most notably, the U-Net model [25] has been gaining popularity in recent years thanks to its outstanding performances, especially for medical segmentation tasks. The U-Net architecture is composed by two parts: the *encoder*, which extracts the context from the images, and the *decoder*, whose task is to determine the segmentation region. The encoder uses several convolutional blocks, followed by max-pooling operations, for encoding the input image into intermediate representations at different spatial resolutions. On the other hand, the decoder is a symmetric network, composed by the same structural form of the encoder, that performs upsampling and concatenation operations for extracting the final segmentation from the input image.

In this paper, we employ the U-Net model [27] that modifies the original one [25] and it is better suited to the considered FL medical context. Compared to [25], the encoder introduces dropout layers for allowing better model generalization and prevent overfitting. Moreover, the model has been adapted to receive input images with different number of channels, depending on the available MRI modalities, while also providing up to four segmentation levels. In total, it presents about 7.8 millions of parameters and weights of size 30 MB.

B. DICE LOSS AND SIMILARITY METRICS

The ML model is trained using a combination of two losses, namely the Generalized Dice Loss (GDL) [54] and the Cross Entropy (CE). The GDL $\mathcal{L}_{GDL}$ is a generalization of the conventional Dice Loss that takes into account multiple class segmentation problems, rather than binary ones. Considering L segmentation labels and K image elements, the GDL can be computed as:

$$\mathcal{L}_{GDL} = 1 - 2\frac{\sum_{\ell=1}^{L} w_D^{(\ell)} \sum_{n=1}^{K} p_{\ell,n} r_{\ell,n}}{\sum_{\ell=1}^{L} w_D^{(\ell)} \sum_{n=1}^{K} p_{\ell,n} + r_{\ell,n}}, \tag{8}$$

where $r_{\ell,n} \in \{0,1\}$ and $p_{\ell,n} \in \{0,1\}$ are respectively the voxel values of the reference foreground segmentation (ground truth) and the values of the predicted map for the foreground label $\ell = 1, .., L$ (an example is given in Figure 2 with $L = 4$). $w_D^{(\ell)}$ is the weight to model the contribution of each label, and it is defined as $w_D^{(\ell)} = \frac{1}{(\sum_{n=1}^{K} r_{\ell,n})^2}$. Finally, the CE loss $\mathcal{L}_{CE}$ is:

$$\mathcal{L}_{CE} = \sum_{n=1}^{K} \sum_{\ell=1}^{L} r_{\ell,n} \log(p_{\ell,n}). \tag{9}$$

Note that the formulations of (8), (9) consider the ground-truth $r_{\ell,n}$ and the label $p_{\ell,n}$ segmentations encoded as one-hot representations. Finally, the total loss in (1) can be computed as:

$$\mathcal{L} = \lambda \mathcal{L}_{GDL} + (1 - \lambda)\mathcal{L}_{CE}, \tag{10}$$

with $\lambda = 0.85$ and is used to update the weights of the Neural Network (NN).

For assessing the quality of the trained models for brain tumor segmentation, we use the Dice Similarity Coefficient (DSC) metric [55]. Given a pair of ground-truth $\mathcal{T}$ and predicted segmentation patches $\mathcal{P}$, the DSC ranges from 0 to 1 and it is computed as:

$$\text{DSC} = \frac{2|\mathcal{P} \cap \mathcal{T}| + 1}{|\mathcal{P}| + |\mathcal{T}| + 1} \tag{11}$$

where $|\mathcal{P} \cap \mathcal{T}|$ denotes the intersection between the predicted and ground-truth patches, while $|\mathcal{P}|$ and $|\mathcal{T}|$ are the cardinalities of the predicted and true segmentation, respectively. High quality models should possess a DSC value close to 1, indicating a near-optimal match between ground-truths and predictions. Of course, since the DSC works with binary masks, a coefficient will be generated for each of the segmented parts of the tumor (e.g., tumor-core and not-tumor-core).

IV. NETWORK ARCHITECTURE AND MQTT PROTOCOL

In this section, a network architecture is proposed to integrate the FL tools described previously. The proposed system adopts the MQTT transport protocol to coordinate the real-time exchange of the U-NET model parameters through MQTT-compliant *publish* and *subscribe* operations. We discuss the benefits of the chosen transport layer, as compared with HTTP based representational state transfer (REST) services, and design an optimized set of information to be embedded into the MQTT payload, as well as model parameters compression, serialization and Quality-of-Service (QoS) mechanisms. The proposed architecture is validated in Sect.V targeting the brain tumor segmentation task (Sect. III). Nevertheless, the proposed framework is general enough for application to distributed training of deep neural network models.

The aggregation and local model optimization steps of the FL process can be implemented via *synchronous* or *asynchronous* policies: when the training process adopts the synchronous orchestration, all the network entities (clients and PS, if any) share the same time reference t: therefore, the model aggregation should wait for all the scheduled clients to complete their local model optimization. Notice that the aggregation steps on the PS (6) and on the client, in fully decentralized training (7), can be adopted as they are. On the contrary, in the asynchronous orchestration, the clients, or the PS, aggregate the available local models without any regard of their relative temporal alignments. More specifically, asynchronous model aggregation (6) implemented on the PS at time t becomes

$$\mathbf{w}_{G,t} = \frac{\epsilon}{\sum_{j=1}^{N} S_j} \sum_{i=1}^{N} S_i \mathbf{w}_{i,t-\tau_i} + (1-\epsilon)\mathbf{w}_{G,t-T_{PS}}, \quad (12)$$

with $\mathbf{w}_{i,t-\tau_i}$ being the local model from client i available at time $t - \tau_i$, $\tau_i \neq 0$ the relative timing mismatch with the PS, and T_{PS} regulating the time span between two global model updates. Similarly, in the asynchronous implementation of the CFA algorithm, once a medical center finishes its local model optimization, it immediately switches to the aggregation step, regardless of whether its neighbors have already finished their local model optimization or not. Therefore, each MN i will now receive from neighbors k the last uploaded models of rounds $t - \tau_k$, $\forall k \in \mathcal{N}_{i,t}$, namely, from (13)

$$\boldsymbol{\psi}_{i,t} = \mathbf{w}_{i,t} + \frac{\epsilon_t}{\sum_{j\in\mathcal{N}_{i,t}} S_j} \sum_{k\in\mathcal{N}_{i,t}} S_k \left(\mathbf{w}_{k,t-\tau_k} - \mathbf{w}_{i,t}\right). \quad (13)$$

In what follows, and based on the above considerations, we analyze in detail four different network architectures and related MQTT orchestration mechanisms weaving together synchronous and asynchronous operations, i.e., on the clients and/or the PS.

A. MQTT MESSAGING AND FL PROCESS ORCHESTRATION

The choice of the MQTT protocol [56], over for example the HTTP RESTful, was dictated by many factors, starting with the superior bandwidth efficiency of MQTT and lower latency [57]. Another important aspect is the low overhead of the protocol, designed for low-power machine-to-machine transactions, which is crucial given the size of the messages exchanged during the federated training phase. The MQTT protocol is also suited for one-to-many communications as needed i.e., during the distribution of the global model from the PS to the MN. Finally, the architecture can be easily extended to the Internet-of-Things (IoT) field, i.e., on embedded devices, or smartphones, where the implementation is practically constrained by low computational capabilities and battery usage requirements.

1) MQTT MESSAGING

In the proposed FL architecture, both the PS and the medical nodes/centers members of the federation act as MQTT client devices and support publish and subscribe operations. Devices thus share the layers of the deep learning model by encapsulating the parameters into the MQTT standard payload. In particular, the basic subset of information included in the payload are:

- *i)* the updated weights $\mathbf{w}_{i,t}$ of the U-NET model trainable layers (Sect. III.A),
- *ii)* tunable parameters for monitoring the convergence, namely: the number E of local rounds to be executed in each client, the learning rate η to be used in the local model optimization step, the target DSC performance (Sect. III.B) and the patience (to apply the early stopping procedure),
- *iii)* training statistics: the client identification number, the federated round indicator and the performance metrics, i.e., DSC in (11), obtained from the validation dataset.

2) PUBLISH-SUBSCRIBE OPERATIONS AND QoS

MQTT publishing and subscription operations are organized into a number of topics. Considering both FA and CFA algorithms, the main topics are the ones related to the MN and PS exchange of the model parameters. In case the model size exceeds the maximum dimension of the MQTT messages (by default set to 250 MByte), the model can be automatically decomposed in different fragments, each representing one or multiple layers of the model, and published separately. Other topics are related to the configuration parameters, to the number of samples of each medical center and to a timestamp that indicates the last time τ_k the k medical center has been seen. Before being sent to the broker, all messages are first serialized into binary objects with the *cPickle* module. We chose to use this module rather than the classic JSON serialization method because of its higher speed and flexibility [58]. Messages are further compressed using the *zlib* module, in order to occupy less space and bandwidth (about 10% less), and are sent to the MQTT broker with Quality of Service (QoS) 2. In particular, QoS 2 implements a 4-way handshake mechanism that is especially effective over communication links with poor quality, while adding a

negligible delay (100 ms) if compared with the time required for transferring the message payload (>1 s). Minimizing packet losses is critical in FL while, as also shown in [59], QoS 2 is the most effective choice for large payload MQTT publishing operations. For all cases, the TLS protocol is adopted to encrypt the exchanged messages.

3) MQTT BROKER

Considering FA, the clients publish their data to a *MQTT broker* that is in charge of maintaining the model parameters and forwarding them to the PS whenever a change is detected. Multiple copies of the messages, as well as packet losses, are handled directly by the QoS 2 specification. The MQTT broker service thus acts as sink node for local models collection and it is thus maintained until the end of the training process. According to the type of broker, the memory size and capabilities can change: in our case, we used a single-threaded broker but other options are also possible. On each round, the broker accepts subscriptions from the active clients that publish their model parameters. Considering server-based FA, all clients also subscribe to the same broker service, i.e., to download the updated global model. On the other hand, for CFA, each client subscribes to its neighbors' topics to retrieve their last available models and publishes the updated model on its related weights topic.

The software that implements the FL process and enables the MQTT transactions for the client (and the PS) is available online [60]. Once installed, it is completely self-sustaining while all network entities maintain an idle mode state until the training process is initiated or updated. The FL process begins with a Command&Control (C&C) tool that uploads the main parameters on the MQTT broker and starts the training. Once the process is started, the PS or the clients, in case of fully decentralized implementation, are in charge of maintaining the training. Furthermore, new authenticated MNs that aim to join the federation, are accepted from any geographical location. The number of resources consumed are chosen a priori according to the complexity of the task and the dataset, the MN computing capabilities and to the desired training speed. To stop the process many possibilities are accepted: reaching a target number of federated rounds or performance, or a direct message from the C&C tool.

B. SYNCHRONOUS AND ASYNCHRONOUS FL NETWORK ARCHITECTURES

The development of a network architecture designed for native FL support over the Internet needs to face two critical challenges. First, slower clients, i.e., retaining large datasets (high number of MRI images) or characterized by low computational resources, experience a longer training time and might penalize faster clients (straggler effect). Second, global model updates issued by the PS, if used, or by the clients, when implementing consensus, should avoid possible deadlock situations caused by packet losses, i.e., links with poor quality as well as delayed model updates from neighboring clients.

To overcome these issues, we introduce and analyze different network architectures that leverage distinct levels of a/synchronicity on the client or the PS (if used), respectively. We show that introducing temporal variations and asynchronicity over some of the FL network entities, namely the clients and the PS, can lead to more efficient training in the presence of slow/heterogeneous FL learners. In particular, for the proposed implementation, a client is considered asynchronous if it can perform more than two local rounds without stopping or waiting for global/local model updates from the PS or neighboring clients. The time span of one local round implemented on client k is defined as $T_{round}(k)$, which is the sum of the time required to download the weights from the MQTT broker, training the new model (local model optimization step), encrypt, compress and upload the weights to the broker:

$$T_{round}(k) = T_{download} + T_{training} + T_{upload}. \tag{14}$$

Considering PS based FL, we implemented a timer that fires every server sleep time, namely T_{PS}. As shown in (12), when this happens, the PS decides whether to update the global model or not, depending on the backlog of local models retained by the MQTT broker. In particular, the Retain Flag of the MQTT protocol is set to true so that the last message sent by the MN is stored into the MQTT broker: when the PS (or another client) subscribes to the same topic, the broker delivers the message. In what follows we highlight 4 selected architectures: notice that three of them are based on the PS a/synchronous orchestration, while the last one supports the fully decentralized FL tools and the consensus process.

1) PS SYNCH., CLIENT SYNCH. (PS-S/C-S)

The architecture, represented in Figure 3b corresponds to the vanilla FL implementation proposed in [4], [8]: both the PS and the clients are synchronous (S) with respect to the training process, therefore all network elements share a common sense of time, while the FL process is supervised by the MQTT broker. The PS waits until all the clients complete their model optimization steps, and monitors the PS weights topic on the MQTT broker. Once all clients have published their weights, the PS is unlocked and implements the aggregation step as in (6). After the PS has published the updated global model, the clients are unlocked and the PS returns to wait.

2) PS SYNCH., CLIENT ASYNCH. (PS-S/C-A)

As described in Figure 3c, the PS keeps the same synchronous behavior as in the PS-S/C-S architecture. On the other hand, the clients adopt asynchronous (A) actions: when their model optimization step is completed, they might continue the training using local data unless the PS global model is updated. In such case, they stop the local model optimization step and replace the local weights $\mathbf{w}_{k,t}$ with the updated global model $\mathbf{w}_{G,t}$.

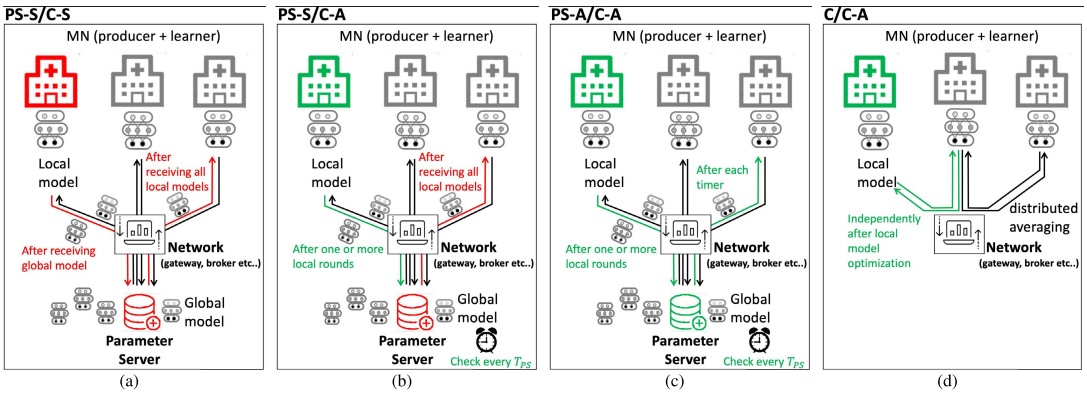


FIGURE 3. FL architectures: from left to right PS-S/C-S, PS-S/C-A, PS-A/C-A and C/C-A. Red color indicates a synchronous element (client or PS) and its implications. On the contrary the asynchronous element is depicted with green color.

3) PS ASYNCH., CLIENT ASYNCH. (PS-A/C-A)

The last type of PS-based architecture is pictured in Figure 3d. In this case, both the PS and the clients are asynchronous (A). On every T_{PS} sec., the PS collects the local model weights from the MQTT broker and updates the global model even though not all clients have finished their model optimization steps. The clients are also asynchronous and act similarly as in the PS-S/C-A architecture.

4) CONSENSUS-DRIVEN (NO PS), CLIENT ASYNCH. (C/C-A)

The architecture supports the consensus-based (C) FL and coordinate the local model exchange among the clients. As illustrated in Figure 3d, the implemented architecture is fully decentralized while every client is asynchronous. The clients can communicate directly with each other uploading and downloading the updated local model weights from the MQTT broker through the specific topics of the neighbors. In particular, the MQTT broker acts as a bridge allowing the communication among interconnected MN, i.e., possibly located in different countries, and according to an assigned connectivity matrix. When a client has finished its round, it downloads the weights of its neighbors, updates the local model according to a specified algorithm (i.e., the CFA described in Sect. II), and in turn publishes the updated local model on its weights topic.

V. CASE STUDY: DIAGNOSTIC IMAGING FOR BRAIN TUMOR SEGMENTATION

The study and validation of the proposed architecture and FL tools is performed on a federation of 5 MNs distributed across Europe. As detailed in Sect. III, we selected the task of binary segmentation of brain tumors (and tumor-like pathologies) in the axial slice-based single-channel FLAIR MR images to concentrate our effort on the challenge of implementing a networking and computing environment in a realistic clinical setting. The FL process and model quality are verified first using the BraTS public data repository (BraTS 2018), next we analyze a more realistic set up featuring an additional real-world clinical data set of images not prepared ad-hoc for testing. Such new dataset is referred to as "Athens set" and it is used to complement the federation with additional MNs. Finally, validation on the BraTS 2020 set is also considered. For all setups, the model quality, here also referred to as performance level, is measured in terms of DSC metric defined in (11).

The approach we follow for the analysis is threefold. First, the data pre-processing pipeline (Sect. V-A) targets the harmonization of BraTS and Athens sets. All the proposed FL network orchestration mechanisms, namely the PS-S/C-S, PS-S/C-A, PS-A/C-A, C/C-A, are then compared with the benchmark centralized ML (Sect. V-B) with respect to the DSC metric. In particular, the parameter server sleep time T_{PS} is optimized for asynchronous FL, targeting the PS-A/C-A architecture. Finally, we analyze the performance in a practical scenario where FL is implemented on heterogeneous medical nodes located in Italy and Switzerland (Sect. V-C). For these last experiments, we consider different combinations of training and validation sets quantifying for each case the benefits of federation.

A. DATASETS AND DATA PREPARATION

To verify the performance of the proposed FL tools and architectures we first populated 4 MNs, located in different countries, with shards from publicly available BraTS 2018 and BraTS 2020 datasets [21]–[23]. The division of BraTS dataset into training and validation sets was performed on a per-examination basis, so all slices from one examination ended up in the same set. The number of examinations and slices per node can be seen in Table 1. In particular, BraTS 2018 dataset was splitted into three shards (namely MNs 1, 2, and 3), and the new examinations from BraTS 2020 (added on top of BraTS 2018), were used for the last client (MN 4).

TABLE 1. Distribution of BraTS datasets across three different FL clients.

	Exam.	Abnorm. slices	Normal slices
Training - MN 1 (BraTS 18)	67	4421	5964
Training - MN 2 (BraTS 18)	67	4458	5927
Training - MN 3 (BraTS 18)	66	4296	5934
Training - MN 4 (BraTS 20)	59	3817	5328
Validation - MN 1 (BraTS 18)	19	1309	1636
Validation - MN 2 (BraTS 18)	19	1272	1673
Validation - MN 3 (BraTS 18)	19	1207	1738
Validation - MN 4 (BraTS 20)	25	1681	2194
Total	341	22461	30394

TABLE 2. Size and distribution of Athens' dataset into training, validation and testing sets.

	Abnormal		Normal	
	Exam.	Slices	Exam.	Slices
Training - MN 5 (Athens)	21	1173	18	1005
Validation - MN 5 (Athens)	5	233	4	186
Total	26	1406	22	1191

As previously mentioned, to complement the federation with additional nodes and training data, we considered a new private dataset (Athens dataset) consisting of 48 MRI examinations, out of which 26 contain abnormalities, and the rest 22 are normal, serving as negative control in the experiments. Similarly as for BraTS, the examinations were divided into training and validation sets on the per-examination basis. The setup has been designed purposely to cope with harmonized, but significantly different, data across nodes. The new dataset is hosted in the MN labelled as 5 while the total size of the dataset and the portions belonging to training and validation sets are depicted in Table 2. Hardware specifics and computing capabilities of the 5 deployed clients are further detailed in Table 3.

1) BRATS VS. ATHENS SET

Besides the fact that images in Athens and BraTS datasets come from different sources, they differ also in few other critical aspects. First, Athens data consist of raw samples as they come out of the MRI machine, without any skull-stripping or transformations. On the other hand, BraTS images are already pre-processed: they are all registered to the common atlas, interpolated and subsampled to an uniform resolution 1 mm^3 in all three main axes in 3D. Furthermore, BraTS collection contains only examinations with abnormalities, while in the Athens dataset there are also normal examinations (healthy patients) and a wider range of oncological cases w.r.t. the shape, position and typology. Finally, BraTS dataset contains only high- and low grade gliomas (HGG and LGG), while Athens examples feature also metastatic and smaller tumor-like lesions (see the examples in Fig. 4).

2) DATA HARMONIZATION AND AUGMENTATION

To harmonize the BraTS and Athens inputs, so that the data of all nodes can contribute to model training, we employed few normalization steps in the data processing pipeline.

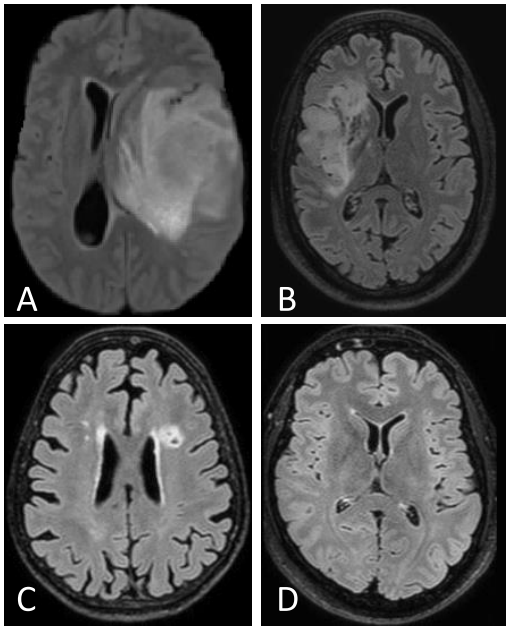


FIGURE 4. Representative images from BraTS (A) and Athens' (B, C, D) datasets: A) large detection with surrounding edema in the left hemisphere from BraTS 2020; B) similar case in right symmetrical position from Athens' data set; C) small detection on the left semi-oval centrum, and D) normal examination.

First, Athens images are resampled to the spatial resolution of 1 mm^2, so that the spatial dimensions are compatible with BraTS samples. Then, the intensities of each slice are standardized (i.e., transformed such that the mean intensity is 0 with standard deviation 1). All slices are then clipped or padded to obtain images with uniform dimensions 240×240 pixels. Because the BraTS dataset contains more fine-grained segmentation labels (marking individual parts of the tumors) compared to what we needed, we merged all labels together to produce a *whole-tumor* segmentation mask. On top of the harmonization of the input data, we employed also a data *randomization* step that consists of randomized transformations like image flipping, rotations and elastic deformations. Input images are also altered by adding Gaussian noise. This process helps to de-correlate the training samples and in turn improve the robustness of the model against overfitting. It is worth to mention that higher spatial resolution MRIs provide critical anatomical features that help to better detect illness and make diagnoses. Unfortunately, High Resolution (HR) MRIs are hampered by extended scan times and low signal-to-noise ratio (SNR), especially when hardware capacity is restricted. Consequently, often Low Resolution (LR) images are taken. Recent research has shown that using CNNs and single image super-resolution (SISR) techniques, HR images may be reconstructed from LR ones [61].

TABLE 3. Medical Nodes hardware and computing capabilities.

	CPU	RAM	GPU
Milan site 1: Desktop PC	Intel(R) Core(TM) i7-3930K @ 3.2 GHz	32 GB	NVIDIA GeForce RTX 3090 with 24 GB
Milan site 2: Laptop PC	i7-10750H @ 2.6 GHz	32 GB	NVIDIA GeForce GTX 1650 Ti with 4 GB
Geneva site 1: Desktop PC1	Intel(R) Core(TM) i7-6700 @ 3.4 GHz	32 GB	NVIDIA Quadro K2200 with 4 GB
Geneva site 2: Desktop PC2	Intel(R) Xeon(R) E5-1630 v3 @ 3.70 GHz	32 GB	NVIDIA GeForce RTX 3090 with 24 GB
Geneva site 3: Desktop PC3	Intel(R) Xeon(R) E5-1630 v3 @ 3.70 GHz	32 GB	NVIDIA GeForce RTX 3090 with 24 GB

B. ASSESSMENT OF NETWORK ARCHITECTURES

In the following initial tests, we deployed 4 clients co-located with the Milan site 1 (hardware specifics are detailed in Table 3). Three clients use training data from the BraTS 2018 while the remaining one uses the Athens dataset. In addition, the FL model quality is validated with both Athens and BraTS validation sets and assessed using the DSC metric (11). This is obtained by averaging the DSC metric over the full validation dataset in each MN. For real-time evaluation of the FL process, the hardware platform hosting each client has been adapted to use 4 GB of GPU memory and about 3.5/4 GB of RAM. To optimize memory usage, we adopted the TFRecord binary format and only a suited number of training samples were kept in memory: about 1000 training slices for shuffling reasons. Regarding the training parameters, we set the number E of local epochs to 1, for the reasons described in Section II-B, and the batch size B to 16. The learning rate of the Adam optimizer was set to 10^{-4}, while the β_1 and β_2 hyperparameters are fixed respectively to 0.9 and 0.999. The validation phase is performed by each MN on its validation set after receiving the global model by the PS. This validation step provides for a more accurate tracking of model quality improvements over time.

In Figure 5 we compare the DSC of PS-S/C-A and the PS-A/C-A architectures at different clock times and by varying the server sleep time T_{PS}. The optimal choice of the sleep time depends on the minimum/maximum local round time between each client, defined as:

$$\begin{aligned} T_{MAX} &= \max_k T_{round}(k) \\ T_{MIN} &= \min_k T_{round}(k). \end{aligned} \tag{15}$$

As highlighted in the corresponding scenarios, setting $T_{PS} < T_{MIN}$ improves the training time of the PS-A/C-A architecture: in particular, it reaches a target DSC of 0.85 while saving the 20% of the training time compared with the synchronous PS-S/C-A option. On the contrary, performance drops are observed when $T_{PS} >= T_{MIN}$, specifically regarding the final DSC that worsens from 0.878 to 0.865. For what concerns the PS-S/C-A architecture, setting $T_{PS} <= 2\ T_{MAX}$ (blue area), gives the worst performance. This result can be due to the fact that in the PS-S/C-A architecture, the clients perform too many local rounds before the aggregation step and this can lead to a bias in the training process. Finally, for the same reason, we can observe that increasing too much T_{PS} (red line) is detrimental and not useful.

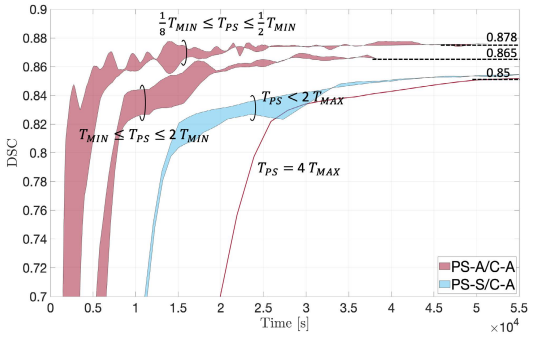


FIGURE 5. Comparison between PS-S/C-A (blue areas) and PS-A/C-A (red areas) architectures for varying T_{PS}. The plots report the observed DSC at different clock times. Highlighted scores of 0.878, 0.865 and 0.85 produce different prediction images (as analyzed in Figure 6).

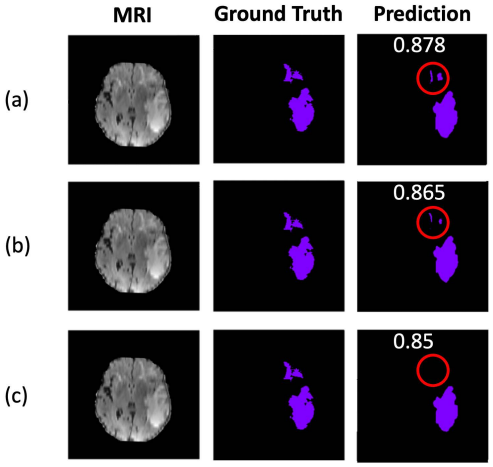


FIGURE 6. Qualitative representation of performance at convergence in Figure 5. From top to bottom, the predicted images reached a DSC metric of respectively 0.878, 0.865 and 0.85. These values correspond to prediction by, respectively, PS-A/C-A with $T_{PS} < T_{MIN}$ (image a), PS-A/C-A with $T_{PS} >= T_{MIN}$ (image b) and PS-S/C-A (image c). Red circle highlights the main detection difference in the upper part of the tumor.

Besides performances as measured by DSC and training time, in Figure 6 we analyze the tumor segmentation quality by visual inspection and considering the final DSC reached by the FL architectures PS-S/C-A and PS-A/C-A after $4 \cdot 10^4$ seconds of training. Although the DSC metric is above 0.85 for all cases, the trained models in Figure 6a

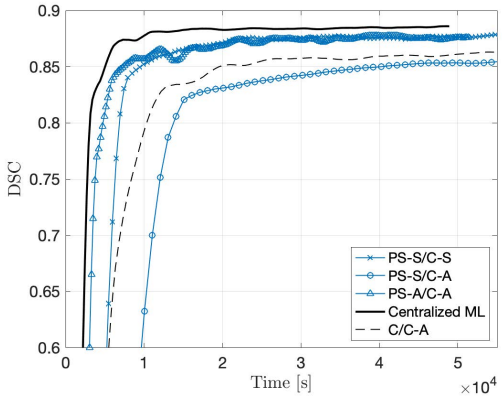


FIGURE 7. Comparison among all architectures. The centralized ML benchmark is highlighted with a thick solid black line, while the fully decentralized FL architecture (C/C-A) is in dashed line. The three server-based FL architectures are in blue lines with different markers.

and 6b are able to detect the upper part of the tumor as they achieve a DSC of 0.878 and 0.865, respectively. On the other hand, the model in Figure 6c cannot, since the corresponding DSC is only 0.85. This example is critical to understand how even a small increase on DSC can significantly improve the predicted image, as well as the segmentation precision/quality.

Considering the same settings described previously, in Figure 7 we investigated the differences among all the proposed FL architectures when compared with the benchmark centralized ML case. The C/C-A architecture replaces the PS in exchange for slower convergence time compared with server-based FL policies (i.e., PS-A/C-A). However, it reaches comparable dice whole metric above 0.85. The performance of the PS-S/C-A scheme is worse than the fully decentralized C/C-A: the heterogeneity of the datasets suggests the use of an asynchronous PS with $T_{PS} < 2\ T_{MAX}$. Compared with synchronous FL, the asynchronous PS-A/C-A architecture, with $T_{PS} < T_{MIN}$, obtains the best tradeoff between performances and training time.

C. IMPACT OF HETEROGENEOUS NODES AND VARYING DATA SETS

In this section we target practical setups where the MNs are located in different countries. Nodes are also equipped with varying computing capabilities, according to Table 3, and communicating over the Internet. The goal is to highlight the robustness of the FL process in handling both data and client heterogeneity. The proposed setups also verify how trained models via FL can be updated and transferred to newcomer MNs bringing new data to the process. In what follows, we adopted the PS-A/C-A architecture by keeping the same server sleep time $T_{PS} = 127$ seconds, as optimized in the previous section. The local round time of the clients varies depending on the physical machine, namely the hardware and

TABLE 4. Results of the real-time remote experiments with the PS-A/C-A FL architecture and $T_{PS} = 127$ s. MN are described in Tables 1 and 2, sites (s.) in Milan (M.) and Geneva (G.) and corresponding hardware are in Table 3. For each of the three experiments (Tables a-b-c), we describe in the upper sub-table the different compositions of the MNs according to their physical sites and the round time T_{round}. The lower sub-table reports T_{MAX} and T_{MIN} values along with the time required to achieve a DSC of 0.85 and the DSC reached after $5 \cdot 10^4$ s.

(a)

Test 1	MN1	MN2	MN3	MN5
Hardware	M. s. 1	M. s. 1	M. s. 1	G. s. 1
T_{round}[s]	441	441	441	332

T_{MAX}[s]	441
T_{MIN}[s]	332
Time [s] to DSC 0.85	12000
DSC after $5 \cdot 10^4$ s	0.878-0.880

(b)

Test 2	MN1	MN2	MN3	MN5
Hardware	M. s. 1	M. s. 1	M. s. 2	G. s. 1
T_{round}[s]	127	127	747	332

T_{MAX}[s]	747
T_{MIN}[s]	127
Time [s] to DSC 0.85	7000
DSC after $5 \cdot 10^4$ s	0.866-0.870

(c)

Test 3	MN1	MN2	MN3	MN5
Hardware	M. s. 1	G. s. 2	G. s. 3	G. s. 1
T_{round}[s]	80	72	70	359

T_{MAX}[s]	359
T_{MIN}[s]	70
Time [s] to DSC 0.85	6000
DSC after $5 \cdot 10^4$ s	0.8707-0.8715

the computing capability. We also used the same number of clients and dataset distributions.

As described in Tables 4a-4b-4c, we performed 3 tests using MNs 1, 2, 3 and 5 in different sites. A visual representation of test number 3 (Table 4c) indicating the specific MNs locations and corresponding dataset distributions can be found in Figure 8. Each MN is characterized by a different time round T_{round}, while the corresponding T_{MAX} and T_{MIN} (15) values are reported for simplicity. For each test, we also reported the time required to achieve a target DSC of 0.85, and the DSC reached after a training period of $5 \cdot 10^4$ seconds (corresponding to about 800 rounds). Considering the time required to reach a DSC of 0.85, we can notice that having a low T_{MIN} leads to higher performances especially in the first part of the training process. This is due to the fact that the MNs with large computing capability can perform several local rounds between two global model updates, thus refining the local model. On the other hand, the higher DSC reached after a period of $5 \cdot 10^4$ seconds is achieved only by test 1 (Table 4a) that has the optimal server sleep time: $T_{PS} < T_{MIN}$.

In the last experiment, we analyzed a further setup where the MN labelled as 4 in Table 1 joins the federation. In particular, the new node contains examples drawn from the BraTS 2020 set and is located in the Milan site 1 (Table 3). The purpose of these tests is to verify the robustness

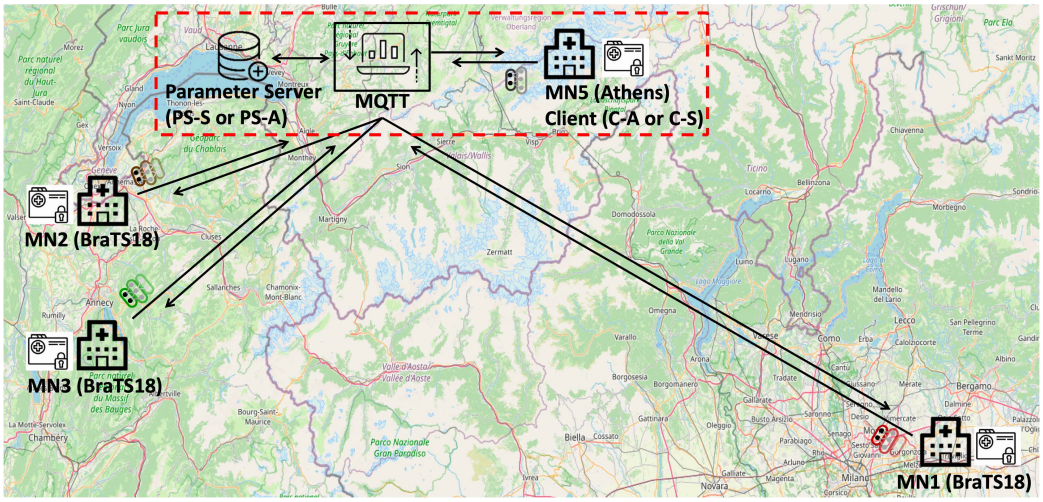


FIGURE 8. Geographical representation of Test 3 of Table 4. The MNs (i.e., the datasets) are also described in Tables 1.

TABLE 5. FL performances with 12 different training/validation set combinations ranging from BraTS 2018, BraTS 2020 and the new Athens dataset. Cross-markers indicate, for each case, the training sets and the corresponding validation examples on which the DSC (second-last column) is computed. The dimension of the total MRI training datasets is reported in the last column in terms of number of slices (of size 115 KB). Max and min values of validation DSC for each dataset are highlighted in green and red, respectively. Notice that the MN 1-2-3 (Table 1) are grouped together in BraTS 18 for brevity.

		Training			Validation					
	# MNs	BraTS 18	BraTS 20	Athens	BraTS 18	BraTS 20	Athens	DSC	# MRI slices	
1	1			X			X	0.72-0.76	2178	one dataset
2	1	X			X			0.860-0.870	31000	
3	1		X			X		0.874-0.878	9145	
4	4	X		X			X	0.730-0.750	33178	two datasets
5	4	X		X	X			0.872-0.876	33178	
6	4	X	X		X			0.877-0.879	40145	
7	4	X	X			X		0.881-0.883	40145	
8	2		X	X		X		0.878-0.882	11323	
9	2		X	X			X	0.740-0.760	11323	
10	5	X	X	X	X			0.879-0.881	42323	three datasets
11	5	X	X	X		X		0.888-0.898	42323	
12	5	X	X	X			X	0.730-0.750	42323	

of FL, in terms of DSC, for varying combinations of training and validation sets chosen from the BraTS 2018, the BraTS 2020 and the Athens data previously described. In Table 5, we compared the observed DSC after 800 FL rounds (corresponding to approx. $5 \cdot 10^4$ seconds) considering all possible combinations of training and validation sets, as indicated by the corresponding markers. In particular, the first three cases (rows 1, 2 and 3) correspond to single node training where the MN uses only its local data for learning (namely BraTS 18, BraTS 20 or Athens), without joining the federation. The following six cases (rows from 4 to 9) highlight the training performance observed when 2 to 4 MNs share their model parameters in the federation. Finally, the last three cases (rows 10, 11 and 12) show the performance of a federation of 5 MNs using all the available datasets (42323 MRI slices) and different validation examples. We can notice that in general, if a MN joins a federation (of 2 to 5 MNs), the DSC increases as expected. Although improvements are numerically small, they provide a significant increase of the predicted image segmentation resolution, as closer to the ground-truth (see also the examples of Figure 6). With respect to single node training, the DSC increases on average by 1% considering a federation of 4 MN, and scales up to 1.6% when all the training sets are considered (5 MNs). Notice that further improvements are expected by replacing the current U-NET model with more complex options [24], [26], as well as increasing the number of deployed MNs, in exchange for larger training time. Looking now at the training datasets, we can observe that the major performance improvements occur when the MNs retaining BraTS data join a federation with other MNs holding training samples with similar characteristics (BraTS 2018 or 2020): see cases 2, 6 and 3, 7. Interestingly, a MN that joins the federation and brings samples from the Athens set provides further

improvements, estimated as 0.6% on average, see cases 6, 10, and 7, 11. This last observation is noteworthy because even with all the differences described in Section V-A and with the fewer examples/slices of Athens dataset, the increase of DSC is significant. FL performance improvements with increasing dataset size, i.e., from one to three datasets as in cases 2, 10 and 3, 11, can reach up to 1.7%, while the benefits of joining the federation are evident. From a different perspective, excluding MNs from the federation, even though they exhibit straggler-like behaviors, is not recommended since the global model might lose its ability to generalize, i.e., being less adaptive to new data. Finally, it is worth noticing that the DSC on the Athens validation dataset seems to be little influenced by the MNs holding the BraTS sets (cases 1, 4, 9 and 12). A cause of this could be the fewer validation slices or the different initial resolution of the Athens dataset.

VI. CONCLUSION: OPEN CHALLENGES AND FUTURE ACTIVITIES

The paper proposed federated and decentralized learning tools to support smart healthcare networks and medical diagnosis. An example of implementation was given in the context of brain tumor segmentation. Parameter server (PS) based federated learning (FL) and fully decentralized FL tools relying upon average consensus have been implemented on top of the MQTT transport protocol, while different network architectures and related designs were proposed to exploit synchronous and/or asynchronous coordination among the clients and the PS, when used. In particular, leveraging distinct levels of asynchronicity on the clients was found as a more efficient option in the presence of heterogeneous nodes and data, as letting the medical nodes to fully explore their local examples before publishing their updates. Asynchronous consensus is also a good-compromise between resource utilization and performances. The proposed architectures were extensively tested on medical data composed of heterogeneous datasets from public and private sources, demonstrating first of all the advantage of the FL system on the centralized training, and furthermore, the main benefits of the MQTT protocol when it comes to reliability, bandwidth efficiency and scalability.

The applications of FL for healthcare and automatic diagnosis are expected to quickly mature in the coming years targeting robust, scalable and privacy-preserving health services. In particular, distributed learning is expected to realize larger-scale and collaborative healthcare systems possibly opening to fully decentralized diagnosis operations, as opposed to centralized analytics on data centers. Besides the promising opportunities highlighted by the paper, future developments of FL are expected to address the robustness of classical and decentralized tools against adversarial manipulations and data poisoning. In addition, the trained models typically observed in medical applications have very large size, ranging from 30 and up to 150 MB [62] per medical node and learning round: when the FL process is implemented over wireless channels, the communication bottleneck should be carefully addressed, possibly via quantization, compression and model updates sparsification. As highlighted in the case study, excluding slow medical nodes (stragglers) from the federation is also not recommended in healthcare networks as these might be fundamental to improve model generalization. Trade-off solutions should be therefore investigated. A properly designed FL system can obtain performances meeting the needs of healthcare professionals. Nevertheless, a quantitative evaluation of trustworthiness metrics is also of fundamental importance to ensure the robustness of the FL platform.

ACKNOWLEDGMENT

Part of this work is performed using anonymized medical images from the 2nd Department of Radiology at Attikon University Hospital in Athens in the framework of a joint CERN/ Attikon University Hospital thesis. The authors would like to thank Prof E. Efstathopoulos and Prof. E. Karavasilis and their teams for their valuable support and guidance in the clinical interpretation of the results.

REFERENCES

[1] D. Wang, A. Khosla, R. Gargeya, H. Irshad, and A. H. Beck, "Deep learning for identifying metastatic breast cancer," Jun. 2016, *arXiv:1606.05718*.

[2] A. Ghubaish, T. Salman, M. Zolanvari, D. Unal, A. Al-Ali, and R. Jain, "Recent advances in the Internet-of-Medical-Things (IoMT) systems security," *IEEE Internet Things J.*, vol. 8, no. 11, pp. 8707–8718, Jun. 2021.

[3] C. Xu, N. Wang, L. Zhu, K. Sharif, and C. Zhang, "Achieving searchable and privacy-preserving data sharing for cloud-assisted E-healthcare system," *IEEE Internet Things J.*, vol. 6, no. 5, pp. 8345–8356, Oct. 2019.

[4] J. Konečný, B. McMahan, and D. Ramage, "Federated optimization: Distributed optimization beyond the datacenter," Nov. 2015, *arXiv:1511.03575*.

[5] N. Rieke, J. Hancox, W. Li, F. Milletarì, H. R. Roth, S. Albarqouni, S. Bakas, M. N. Galtier, B. A. Landman, K. Maier-Hein, S. Ourselin, M. Sheller, R. M. Summers, A. Trask, D. Xu, M. Baust, and M. J. Cardoso, "The future of digital health with federated learning," *npj Digit. Med.*, vol. 3, no. 1, pp. 1–7, Dec. 2020.

[6] M. J. Sheller, B. Edwards, G. A. Reina, J. Martin, S. Pati, A. Kotrotsou, M. Milchenko, W. Xu, D. Marcus, R. R. Colen, and S. Bakas, "Federated learning in medicine: Facilitating multi-institutional collaborations without sharing patient data," *Sci. Rep.*, vol. 10, no. 1, p. 12598, Dec. 2020.

[7] S. Warnat-Herresthal, H. Schultze, K. L. Shastry, S. Manamohan, S. Mukherjee, V. Garg, R. Sarveswara, K. Händler, P. Pickkers, N. A. Aziz, and S. Ktena, "Swarm learning for decentralized and confidential clinical machine learning," *Nature*, vol. 594, no. 7862, pp. 265–270, Jun. 2021, doi: 10.1038/s41586-021-03583-3.

[8] H. B. McMahan, E. Moore, D. Ramage, and B. A. Y. Arcas, "Federated learning of deep networks using model averaging," Feb. 2016, *arXiv:1602.05629*.

[9] D. C. Nguyen, M. Ding, Q.-V. Pham, P. N. Pathirana, L. B. Le, A. Seneviratne, J. Li, D. Niyato, and H. V. Poor, "Federated learning meets blockchain in edge computing: Opportunities and challenges," *IEEE Internet Things J.*, vol. 8, no. 16, pp. 12806–12825, Aug. 2021.

[10] C. Ma, J. Li, M. Ding, H. Yang, F. Shu, T. Suek, and H. V. Poor, "On safeguarding privacy and security in the framework of federated learning," *IEEE Netw.*, vol. 34, no. 4, pp. 242–248, Jul./Aug. 2020.

[11] P. Kairouz *et al.*, "Advances and open problems in federated learning," 2019, *arXiv:1912.04977*.

[12] A. Hard, K. Rao, R. Mathews, S. Ramaswamy, F. Beaufays, S. Augenstein, H. Eichner, C. Kiddon, and D. Ramage, "Federated learning for mobile keyboard prediction," Nov. 2018, *arXiv:1811.03604*.

[13] M. J. Sheller, G. A. Reina, B. Edwards, J. Martin, and S. Bakas, "Multi-institutional deep learning modeling without sharing patient data: A feasibility study on brain tumor segmentation," in *Brainlesion: Glioma, Multiple Sclerosis, Stroke and Traumatic Brain Injuries*. Cham, Switzerland: Springer, Jan. 2019, pp. 92–104.

[14] S. Samarakoon, M. Bennis, W. Saad, and M. Debbah, "Distributed federated learning for ultra-reliable low-latency vehicular communications," *IEEE Trans. Commun.*, vol. 68, no. 2, pp. 1146–1159, Nov. 2020.

[15] S. Savazzi, M. Nicoli, M. Bennis, S. Kianoush, and L. Barbieri, "Opportunities of federated learning in connected, cooperative, and automated industrial systems," *IEEE Commun. Mag.*, vol. 59, no. 2, pp. 16–21, Feb. 2021.

[16] S. R. Pokhrel and J. Choi, "Federated learning with blockchain for autonomous vehicles: Analysis and design challenges," *IEEE Trans. Commun.*, vol. 68, no. 8, pp. 4734–4746, Aug. 2020.

[17] Q. Yang, Y. Liu, T. Chen, and Y. Tong, "Federated machine learning: Concept and applications," *ACM Trans. Intell. Syst. Technol.*, vol. 10, no. 2, pp. 1–19, Jan. 2019, doi: 10.1145/3298981.

[18] S. Savazzi, M. Nicoli, and V. Rampa, "Federated learning with cooperating devices: A consensus approach for massive IoT networks," *IEEE Internet Things J.*, vol. 7, no. 5, pp. 4641–4654, May 2020, doi: 10.1109/JIOT.2020.2964162.

[19] H. Wu and P. Wang, "Fast-convergent federated learning with adaptive weighting," *IEEE Trans. Cognit. Commun. Netw.*, vol. 7, no. 4, pp. 1078–1088, Dec. 2021.

[20] M. Havaei, F. Dutil, C. Pal, H. Larochelle, and P.-M. Jodoin, "A convolutional neural network approach to brain tumor segmentation," in *Brainlesion: Glioma, Multiple Sclerosis, Stroke and Traumatic Brain Injuries*, A. Crimi, B. Menze, O. Maier, M. Reyes, and H. Handels, Eds. Cham, Switzerland: Springer, Jan. 2016, pp. 195–208.

[21] B. H. Menze *et al.*, "The multimodal brain tumor image segmentation benchmark (BRATS)," *IEEE Trans. Med. Imag.*, vol. 34, no. 10, pp. 1993–2024, Oct. 2015.

[22] S. Bakas, H. Akbari, A. Sotiras, M. Bilello, M. Rozycki, J. S. Kirby, J. B. Freymann, K. Farahani, and C. Davatzikos, "Advancing the cancer genome atlas glioma MRI collections with expert segmentation labels and radiomic features," *Sci. Data*, vol. 4, no. 1, Dec. 2017, Art. no. 170117.

[23] S. Bakas *et al.*, "Identifying the best machine learning algorithms for brain tumor segmentation, progression assessment, and overall survival prediction in the brats challenge," Mar. 2019, *arXiv:1811.02629*.

[24] A. Myronenko, "3D MRI brain tumor segmentation using autoencoder regularization," in *Brainlesion: Glioma, Multiple Sclerosis, Stroke and Traumatic Brain Injuries*, A. Crimi, S. Bakas, H. Kuijf, F. Keyvan, M. Reyes, and T. van Walsum, Eds. Cham, Switzerland: Springer, Jan. 2019, pp. 311–320.

[25] O. Ronneberger, P. Fischer, and T. Brox, "U-net: Convolutional networks for biomedical image segmentation," in *Medical Image Computing and Computer-Assisted Intervention—MICCAI*, N. Navab, J. Hornegger, W. M. Wells, and A. F. Frangi, Eds. Cham, Switzerland: Springer, Nov. 2015, pp. 234–241.

[26] Z. Jiang, C. Ding, M. Liu, and D. Tao, "Two-stage cascaded U-Net: 1st place solution to brats challenge 2019 segmentation task," in *Brainlesion: Glioma, Multiple Sclerosis, Stroke and Traumatic Brain Injuries*, A. Crimi and S. Bakas, Eds. Cham, Switzerland: Springer, May 2020, pp. 231–241.

[27] IntelAI. *UNet*. Accessed: Nov. 2021. [Online]. Available: https://github.com/IntelAI/unet/tree/master/2D

[28] M. A. Ferrag, O. Friha, L. Maglaras, H. Janicke, and L. Shu, "Federated deep learning for cyber security in the Internet of Things: Concepts, applications, and experimental analysis," *IEEE Access*, vol. 9, pp. 138509–138542, 2021.

[29] H. Kim, J. Park, M. Bennis, and S.-L. Kim, "Blockchained on-device federated learning," *IEEE Commun. Lett.*, vol. 24, no. 6, pp. 1279–1283, Jun. 2020.

[30] Z. Chen and E. G. Larsson, "Consensus-based distributed computation of link-based network metrics," *IEEE Signal Process. Lett.*, vol. 28, pp. 249–253, 2021.

[31] M. M. Amiri and D. Gündüz, "Machine learning at the wireless edge: Distributed stochastic gradient descent over-the-air," *IEEE Trans. Signal Process.*, vol. 68, pp. 2155–2169, Mar. 2020.

[32] M. Blot, D. Picard, M. Cord, and N. Thome, "Gossip training for deep learning," 2016, *arXiv:1611.09726*.

[33] Y. Qu, S. R. Pokhrel, S. Garg, L. Gao, and Y. Xiang, "A blockchained federated learning framework for cognitive computing in industry 4.0 networks," *IEEE Trans. Ind. Informat.*, vol. 17, no. 4, pp. 2964–2973, Apr. 2021.

[34] L. Barbieri, S. Savazzi, M. Brambilla, and M. Nicoli, "Decentralized federated learning for extended sensing in 6G connected vehicles," *Veh. Commun.*, vol. 33, Jan. 2022, Art. no. 100396. [Online]. Available: https://www.sciencedirect.com/science/article/pii/S2214209621000656

[35] A. Guha Roy, S. Siddiqui, S. Pölsterl, N. Navab, and C. Wachinger, "BrainTorrent: A peer-to-peer environment for decentralized federated learning," May 2019, *arXiv:1905.06731*.

[36] *Federated Learning Brings AI With Privacy to Hospitals*. Accessed: Nov. 2021. [Online]. Available: https://healthcare-in-europe.com/en/news/federated-learning-brings-ai-with-privacy-to-hospitals.html

[37] D. Yang, Z. Xu, W. Li, A. Myronenko, H. R. Roth, S. Harmon, S. Xu, B. Turkbey, E. Turkbey, X. Wang, W. Zhu, G. Carrafiello, F. Patella, M. Cariati, H. Obinata, H. Mori, K. Tamura, P. An, B. J. Wood, and D. Xu, "Federated semi-supervised learning for COVID region segmentation in chest CT using multi-national data from China, Italy, Japan," *Med. Image Anal.*, vol. 70, May 2021, Art. no. 101992.

[38] J. Hoydis, F. A. Aoudia, A. Valcarce, and H. Viswanathan, "Toward a 6G AI-native air interface," *IEEE Commun. Mag.*, vol. 59, no. 5, pp. 76–81, May 2021.

[39] T. Li, A. K. Sahu, A. Talwalkar, and V. Smith, "Federated learning: Challenges, methods, and future directions," *IEEE Signal Process. Mag.*, vol. 37, no. 3, pp. 50–60, May 2020, doi: 10.1109/MSP.2020.2975749.

[40] L. Bottou, "Online algorithms and stochastic approximations," in *Online Learning and Neural Networks*, D. Saad, Ed. Cambridge, U.K.: Cambridge Univ. Press, Oct. 2012. [Online]. Available: http://leon.bottou.org/papers/bottou-98x

[41] H. B. McMahan, E. Moore, D. Ramage, S. Hampson, and B. A. Y. Arcas, "Communication-efficient learning of deep networks from decentralized data," in *Proc. AISTATS*, Apr. 2017, pp. 1273–1282.

[42] D. P. Kingma and J. Ba, "Adam: A method for stochastic optimization," May 2015, *arXiv:1412.6980*.

[43] M. G. Arivazhagan, V. Aggarwal, A. K. Singh, and S. Choudhary, "Federated learning with personalization layers," Dec. 2019, *arXiv:1912.00818*.

[44] K. Simonyan and A. Zisserman, "Very deep convolutional networks for large-scale image recognition," in *Proc. 3rd Int. Conf. Learn. Represent. (ICLR)*, Y. Bengio and Y. LeCun, Eds., San Diego, CA, USA, May 2015.

[45] K. He, X. Zhang, S. Ren, and J. Sun, "Deep residual learning for image recognition," in *Proc. IEEE Conf. Comput. Vis. Pattern Recognit. (CVPR)*, Feb. 2016, pp. 770–778.

[46] D. Cheng and E. Y. Lam, "Transfer learning U-Net deep learning for lung ultrasound segmentation," Oct. 2021, *arXiv:2110.02196*.

[47] G. Soatti, M. Nicoli, S. Savazzi, and U. Spagnolini, "Consensus-based algorithms for distributed network-state estimation and localization," *IEEE Trans. Signal Inf. Process. Netw.*, vol. 3, no. 2, pp. 430–444, Mar. 2017.

[48] M. Blot, D. Picard, M. Cord, and N. Thome, "Gossip training for deep learning," Nov. 2019, *arXiv:1611.09726*.

[49] A. Lalitha, O. C. Kilinc, T. Javidi, and F. Koushanfar, "Peer-to-peer federated learning on graphs," Jan. 2019, *arXiv:1901.11173*.

[50] W. Li, F. Milletarì, D. Xu, N. Rieke, J. Hancox, W. Zhu, M. Baust, Y. Cheng, S. Ourselin, M. J. Cardoso, and A. Feng, "Privacy-preserving federated brain tumour segmentation," in *Machine Learning in Medical Imaging*. Cham, Switzerland: Springer, Oct. 2019, pp. 133–141.

[51] R. Shokri and V. Shmatikov, "Privacy-preserving deep learning," in *Proc. 53rd Annu. Allerton Conf. Commun., Control, Comput. (Allerton)*, Sep. 2015, pp. 909–910.

[52] D. Połap, G. Srivastava, A. Jolfaei, and R. M. Parizi, "Blockchain technology and neural networks for the Internet of Medical Things," in *Proc. IEEE Conf. Comput. Commun. Workshops (INFOCOM WKSHPS)*, Jul. 2020, pp. 508–513.

[53] M. Aledhari, R. Razzak, R. M. Parizi, and F. Saeed, "Federated learning: A survey on enabling technologies, protocols, and applications," *IEEE Access*, vol. 8, pp. 140699–140725, 2020.

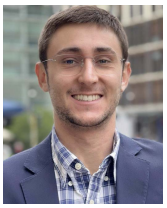
[54] C. H. Sudre, W. Li, T. Vercauteren, S. Ourselin, and M. J. Cardoso, "Generalised dice overlap as a deep learning loss function for highly unbalanced segmentations," in *Proc. Int. Workshop Deep Learn. Med. Image Anal.*, in Lecture Notes in Computer Science, Sep. 2017, pp. 240–248, doi: 10.1007/978-3-319-67558-9_28.

[55] L. R. Dice, "Measures of the amount of ecologic association between species," *Ecology*, vol. 26, no. 3, pp. 297–302, Jul. 1945.

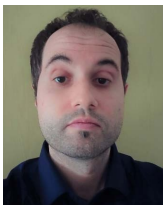
[56] *MQTT V3.1 Protocol Specification*. Accessed: Nov. 2021. [Online]. Available: https://mqtt.org

[57] J. Dizdarević, F. Carpio, A. Jukan, and X. Masip-Bruin, "A survey of communication protocols for Internet of Things and related challenges of fog and cloud computing integration," *ACM Comput. Surv.*, vol. 51, no. 6, pp. 1–29, Jan. 2019, doi: 10.1145/3292674.

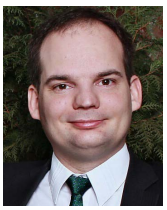
[58] *Pickle vs JSON—Which is Faster in Python3?* Accessed: May 4, 2020. [Online]. Available: https://tinyurl.com/3yznvvy3
[59] S. Lee, H. Kim, D.-K. Hong, and H. Ju, "Correlation analysis of MQTT loss and delay according to QoS level," in *Proc. Int. Conf. Inf. Netw. (ICOIN)*, Jan. 2013, pp. 714–717.
[60] (2021). *Software Repository*. Accessed: Sep. 2021. [Online]. Available: https://github.com/BernardoCama/FL_MQTT
[61] X. Zhu, K. Guo, S. Ren, B. Hu, M. Hu, and H. Fang, "Lightweight image super-resolution with expectation-maximization attention mechanism," *IEEE Trans. Circuits Syst. Video Technol.*, early access, May 10, 2021, doi: 10.1109/TCSVT.2021.3078436.
[62] F. Isensee, P. F. Jäger, P. M. Full, P. Vollmuth, and K. H. Maier-Hein, "nnU-Net for brain tumor segmentation," in *Brainlesion: Glioma, Multiple Sclerosis, Stroke and Traumatic Brain Injuries*, A. Crimi and S. Bakas, Eds. Cham, Switzerland: Springer, Mar. 2021, pp. 118–132.



BERNARDO CAMAJORI TEDESCHINI (Graduate Student Member, IEEE) received the bachelor's degree (Hons.) in computer science and the master's degree (Hons.) in telecommunications, specializing in communication networks and internet from the Politecnico di Milano, in July 2019 and October 2021, respectively, where he is currently pursuing the Ph.D. degree in information technology with the Dipartimento di Elettronica, Informazione e Bioingegneria (DEIB). His research interests include federated learning, machine learning, and localization methods.



STEFANO SAVAZZI (Member, IEEE) received the M.Sc. and Ph.D. degrees (Hons.) in ICT from the Politecnico di Milano, Italy, in 2004 and 2008, respectively. In 2012, he joined the Institute of Electronics, Computer and Telecommunication Engineering (IEIIT), Consiglio Nazionale delle Ricerche (CNR), as a Researcher. He was a Visiting Researcher with Uppsala University, in 2005, and the University of California at San Diego, in 2007. He has coauthored over 100 scientific publications (Scopus). His current research interests include distributed signal processing, machine learning and networking aspects for the Internet of Things, radio localization, and vision technologies. He won the Dimitris N. Chorafas Foundation Award, in 2008. He is serving as an Associate Editor for *Frontiers in Communications and Networks* and *Wireless Communications and Mobile Computing* and Lead Guest Editor for the Special Issue on Radio Sensing and Sensor Networks in Sensors (MDPI).



ROMAN STOKLASA received the M.Sc. (Hons.) and Ph.D. degrees in informatics, image processing, computer vision and machine learning from the Faculty of Informatics, Masaryk University, Brno, Czech Republic, in 2010 and 2016, respectively. In 2013, he was on a visiting internship position with the School of Digital Media Technology, Birmingham City University, U.K. Since 2016, he has been working as a Postdoctoral Researcher with the Centre for Biomedical Image Analysis, Masaryk University. Since 2020, he has been a Senior Fellow with the Technology Department, CERN, Geneva, Switzerland, working on projects related to data and image analysis, machine learning, and predictive maintenance. His research interests include deep learning, image and signal processing, medical and biomedical image analysis, computer vision, and image perception.

LUCA BARBIERI (Graduate Student Member, IEEE) received the B.Sc. and M.Sc. *(cum laude)* degrees in telecommunication engineering from the Politecnico di Milano, in 2017 and 2019, respectively, where he is currently pursuing the Ph.D. degree in information technology with the Dipartimento di Elettronica, Informazione e Bioingegneria (DEIB). His current research interests include machine learning and localization techniques for vehicular and industrial networks.

IOANNIS STATHOPOULOS received the degree in physics and the master's degree in medical physics from the University of Patras. He is currently pursuing the Ph.D. degree with the Department of Technology, European Organization for Nuclear Physics (CERN) and the Medical School of National and the Kapodistrian University of Athens. The subject of his dissertation is the use of machine learning algorithms for detection and classification of brain abnormalities from magnetic resonance images. His research interests include the usage of artificial intelligence, machine and deep learning methods to support medical diagnosis, and pattern recognition analysis using heterogeneous medical data and medical image analysis.

MONICA NICOLI (Member, IEEE) received the M.Sc. (Hons.) and Ph.D. degrees in communication engineering from the Politecnico di Milano, Milan, Italy, in 1998 and 2002, respectively. She was a Visiting Researcher with ENI Agip, from 1998 to 1999, and Uppsala University, in 2001. In 2002, she joined the Politecnico di Milano as a Faculty Member. She is currently an Associate Professor in telecommunications with the Department of Management, Economics and Industrial Engineering. She has coauthored around 150 scientific publications. Her research interests include signal processing, machine learning, and wireless communications, with emphasis on smart mobility and the Internet of Things (IoT) applications. She was a recipient of the Marisa Bellisario Award, in 1999, and a co-recipient of the best paper awards of the IEEE Symposium on Joint Communications and Sensing, in 2021, the IEEE Statistical Signal Processing Workshop, in 2018, and the *IET Intelligent Transport Systems* journal, in 2014. She has served as an Associate Editor for the *EURASIP Journal on Wireless Communications and Networking*, from 2010 to 2017, and a Lead Guest Editor for the Special Issue on Localization in Mobile Wireless and Sensor Networks, in 2011. She is also an Associate Editor of the IEEE TRANSACTIONS ON INTELLIGENT TRANSPORTATION SYSTEMS.

LUIGI SERIO received the M.Sc. degree in nuclear engineering from the Politecnico di Milano, Italy, and the Ph.D. degree in mechanical engineering from Cranfield University, U.K. He is currently a Senior Staff, a Principal Investigator, and the Project Leader with the Technology Department, CERN, Geneva, Switzerland. He is also a Chartered Nuclear Engineer, a Radiation Protection Adviser, and a PMP. He has published more than 100 papers in the field of nuclear and cryogenic engineering and technology. He has an international patent and two theses. His field of expertise is in cryogenics, mechanical, instrumentation, nuclear and thermo-hydraulic engineering, availability and reliability, machine learning and artificial intelligence. He has also worked and collaborated as an Advisor and a Review Committee Member with international projects in the field of nuclear fusion and particle accelerators at JET Joint Undertaking, U.K., SLAC, Stanford, USA, FERMILAB, USA, and ITER Organization, Cadarache, France.

A Traffic Model Based Approach to Parameter Server Design in Federated Learning Processes

Bernardo Camajori Tedeschini, *Graduate Student Member, IEEE*, Stefano Savazzi, *Member, IEEE*, and Monica Nicoli, *Senior Member, IEEE*

***Abstract*— This letter proposes a model to describe the data traffic generated by a Federated Learning (FL) process in a wireless network with asynchronous Parameter Server (PS) orchestration and heterogeneous clients. The model accounts for the local update processes implemented by individual clients and it is used to enforce requirements on the PS design, namely to regulate the interval among consecutive global model updates. PS requirements are validated on realistic pools of resource-constrained wireless edge devices, typically found in Internet-of-Things (IoT) setups. Numerical results show that the proposed policy is effective when devices have unbalanced resources, namely, different sample distributions and computational capabilities. It permits an accuracy gain of up to 15-17% on average with respect to typical asynchronous PS designs.**

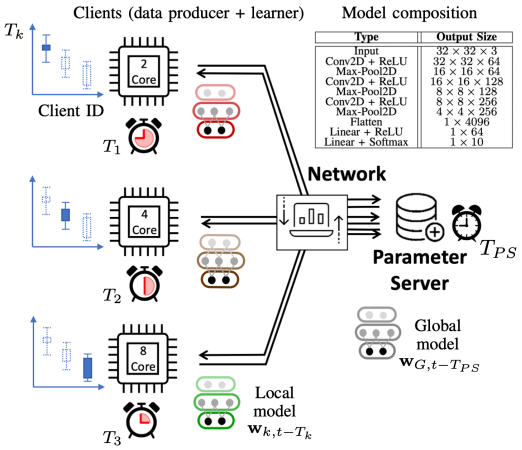
***Index Terms*— Federated learning over networks, traffic modelling, edge devices, computing.**

I. Introduction

FEDERATED Learning (FL) enables resource-constrained edge devices to cooperate over a network for training a shared Machine Learning (ML) model. It protects data ownership by ensuring that the observations used for training never leave the device responsible for its production. As depicted in Fig. 1, FL alternates the computation at each device of local model parameters, i.e., the weights of deep neural network layers, with the communication to a Parameter Server (PS) that fuses the local models and returns a global model [1]. Different FL implementations [2] and enablers [3] emerged in the past few years. Most applications call for geographically distributed [4] and heterogeneous clients with different temporal alignments. In many cases, an asynchronous orchestration of the FL process is also a prerequisite, especially in next generation networks.

Current state-of-the-art on asynchronous FL strategies mainly have the following limitations. First, in vanilla algorithms, the PS updates the global model as soon as a local model is received [5], with no regard to client-specific resource constraints. This can lead, for example, to biased updates from faster clients. Secondly, the update at the clients is not optimized as the number of local epochs is usually fixed and not tuned according to the type of traffic or the quantity/quality of the data [6]. The letter proposes a moment-matching approximation to represent the traffic generated by clients engaged in an asynchronous FL process. The model is validated through a real FL prototype consisting of physically separated clients implementing distributed training over a wireless network, using the Message Queuing Telemetry Transport (MQTT) protocol. Besides adapting and formalizing the moment matching technique to the context of FL, the letter provides the necessary requirements on the time interval among consecutive global model updates, namely the server response time T_{PS}. This is used by the PS to decide whether to update the global model or not, depending on the backlog of local models retained by the clients. The proposed requirements account for the traffic type, the local data size, quality, and the channel impairments affecting the FL local round time.

The letter is organized as follows. Sect. II introduces the proposed traffic model for asynchronous FL. The model uses the moment-matching approximation and permits to categorize the traffic of each client using the dispersion index (D) metric. Requirements in Sect. III exploit the proposed model to define operational points that regulate the clients and the PS behavior, while Sec. IV describes a practical policy for T_{PS} selection that fulfills the proposed requirements. The policy is validated through a real-time FL platform prototype.



Type	Output Size
Input	$32 \times 32 \times 3$
Conv2D + ReLU	$32 \times 32 \times 64$
Max-Pool2D	$16 \times 16 \times 64$
Conv2D + ReLU	$16 \times 16 \times 128$
Max-Pool2D	$8 \times 8 \times 128$
Conv2D + ReLU	$8 \times 8 \times 256$
Max-Pool2D	$4 \times 4 \times 256$
Flatten	1×4096
Linear + ReLU	1×64
Linear + Softmax	1×10

Fig. 1. FL system with heterogeneous and asynchronous clients. Optimized T_{PS} takes into account the local model update completion time of the clients. Top-right corner: model composition for the experiments.

Manuscript received 15 October 2022; revised 6 December 2022, 23 January 2023, and 14 March 2023; accepted 18 April 2023. Date of publication 3 May 2023; date of current version 12 July 2023. The associate editor coordinating the review of this letter and approving it for publication was Z. Yang. *(Corresponding author: Bernardo Camajori Tedeschini.)*

Bernardo Camajori Tedeschini is with the Dipartimento di Elettronica, Informazione e Bioingegneria (DEIB), Politecnico di Milano, 20133 Milan, Italy (e-mail: bernardo.camajori@polimi.it).

Stefano Savazzi is with IEIIT Institute, Consiglio Nazionale delle Ricerche (CNR), 20133 Milan, Italy (e-mail: stefano.savazzi@ieiit.cnr.it).

Monica Nicoli is with the Dipartimento di Ingegneria Gestionale (DIG), Politecnico di Milano, 20133 Milan, Italy (e-mail: monica.nicoli@polimi.it).

Digital Object Identifier 10.1109/LCOMM.2023.3272844

II. FL Data Traffic Model

We consider a FL system composed of one PS and a set of K clients $\mathcal{K} = \{1, \dots, K\}$, each with its own private dataset $\mathcal{S}_k$ of size $S_k = |\mathcal{S}_k|$. As depicted in Fig. 1, the aim of the FL process is to obtain a global ML model, of size G, that minimizes a loss function $\mathbf{w}_G = \text{argmin}_{\mathbf{w}} \mathcal{L}(\mathbf{w})$ with $\mathcal{L} = \frac{1}{K}\sum_{k=1}^{K} \mathcal{L}_k(\mathbf{w}, \mathcal{S}_k)$ and $\mathcal{L}_k$ being the local costs measured by clients using the data batches $\mathcal{S}_k$. The FL process requires the clients to produce local models through optimization, typically via supervised and gradient-based methods. Each client k performs $M(k)$ local epochs before exchanging the local model with the PS, which is in charge of the global model update.

In asynchronous FL, the PS produces a new instance of the global model $\mathbf{w}_{G,t}$ at time $t = nT_{PS}, n = 1, \dots, N_{FL}$ [4]:

$$\mathbf{w}_{G,t} = (1-\epsilon_t)\mathbf{w}_{G,t-T_{PS}} + \frac{\epsilon_t}{\sum_{l=1}^{K} S_l} \sum_{k=1}^{K} S_k \mathbf{w}_{k,t-T_k}, \quad (1)$$

where $\mathbf{w}_{k,t-T_k}$ represents the k-th local model available at time $t-T_k$, $T_k \geq 0$ is the time interval required by the client k to produce an updated local model, while T_{PS} regulates the time span between two global model updates. $n \leq N_{FL}$ is the index of the federated rounds. Finally, ϵ_t controls the stability of the update. Considering that clients may have different computing capabilities and datasets, the associated network traffic can vary significantly depending on T_k. A client-specific characterization is thus proposed to model the local model update process and identify the corresponding traffic pattern.

For the exchange of the NN model parameters among the clients and the PS, we propose to employ the MQTT protocol [7] as it enables a real-time exchange of the model parameters and allows the monitoring of the client training time required for T_{PS} tuning. The time required by a client k to implement a local round can be broken down into the time span to download the weights (T_{down}), train the new model for $M(k)$ epochs using local data batches (T_{train}), encrypt, compress and upload the weights, i.e., to the MQTT broker, (T_{up}):

$$T_k = T_{\text{down}}(k) + M(k) \cdot T_{\text{train}}(k) + T_{\text{up}}(k). \quad (2)$$

These quantities can be computed locally by each client, through standard time measurement functions, and permit to separate the contribution of computing capabilities (T_{train}) from possible channel disturbances affecting uplink (UL) and downlink (DL) communications ($T_{\text{up}}, T_{\text{down}}$).

Based on the above assumption, we introduce a model to approximate the probability density function $p_{T_k}(T_k)$ of the traffic pattern generated by each client k. A moment-matching approximation is employed which divides the process into three categories: Bernoulli, Poisson, and Pascal [8]. We classify the traffic into one of these categories by matching the first two moments defined respectively as:

$$\begin{aligned} &A(k) = \mathbb{E}_n[T_k], \\ &\sigma^2(k) = \mathbb{E}_n[(T_k - A(k))^2],\ n \in \{1, \dots, N_{FL}\}. \end{aligned} \quad (3)$$

The Dispersion Index (D), also called Variance to Mean Ratio (VMR), is:

$$D(k) = \frac{\sigma^2(k)}{A(k)}, \quad \forall k \in \mathcal{K}. \quad (4)$$

According to the moment-matching technique, we can obtain a Poisson traffic by setting $D(k) = 1$, i.e., by imposing a regular traffic pattern. On the contrary, burst-traffic, i.e., Pascal, and smooth traffic, i.e., Bernoulli, are obtained with $D(k) > 1$ and $D(k) < 1$, respectively. Based on the above metrics, in the following, we give upper and lower bounds on the characteristics of the PS, especially regarding the T_{PS}.

III. Minimal Requirements on Clients and T_{PS}

The choice of the server response time T_{PS} is underpinned by the local model update process running on each client, therefore by the number $M(k)$ of epochs that directly reflects on the dispersion index $D(k)$ in (4). Low values of $M(k)$ correspond to frequent contributions of the clients to the global model, at the expense of an increased communication overhead, and possibly non-informative local model updates. Conversely, large $M(k)$ forces the client to implement many local epochs and possibly produce a biased local model (penalized by overfitting).

Optimal $M(k)$ should be bounded as $M_L(k) < M(k) < M_U(k)$. The lower bound $M_L(k)$ sets the minimum $M(k)$ such that the client local model can improve the FL process while satisfying the communication overhead constraints. The upper bound $M_U(k)$ is the maximum $M(k)$ before the client starts overfitting. Note also that $M_L(k)$ is limited by UL and DL maximal communication efficiency η_{MAX} [bit/sec/Hz] dedicated to the link between the PS and the clients, with bandwidth B. Being $T_{FL} = N_{FL}T_{PS}$ the FL training duration, it is:

$$\frac{T_{FL}}{A(k)} G \leq \eta_{MAX} B T_{FL}. \quad (5)$$

This leads to the following:

$$M_L(k) = \mathbb{E}_n \left[\frac{1}{T_{\text{train}}(k)} \left(\frac{G}{\eta_{MAX} B} - (T_{\text{down}}(k) + T_{\text{up}}(k)) \right) \right]. \quad (6)$$

Note that $M_U(k)$ depends mainly on the size of training data and the local model, since more data (or small sized models) require more local epochs for overfitting. For client k, and $\mathbf{w}_{k,m}$ being the local model observed at local epoch $m \in \{1, \dots, M(k)\}$, $M_U(k)$ is assigned as:

$$M_U(k) = \underset{m}{\text{argmin}}\, \mathcal{L}_k(\mathbf{w}_{k,m}, \mathcal{S}_k^{\text{val}}), \quad (7)$$

where $\mathcal{L}_k(\mathbf{w}_{k,m}, \mathcal{S}_k^{\text{val}})$ is the validation loss computed by client k on the validation dataset $\mathcal{S}_k^{\text{val}}$ at epoch m.

As shown in the next section, the choice of $M(k)$ affects T_{PS} and can be used to set practical bounds on global model updates. On one hand, small $M(k)$ such that $M(k) \ll M_L(k)$ might result in $T_{PS} \gg A(k)$ thus exceeding the constraint on the spectral efficiency, with negligible effect on the FL process and accuracy. On the other hand, performing sporadic

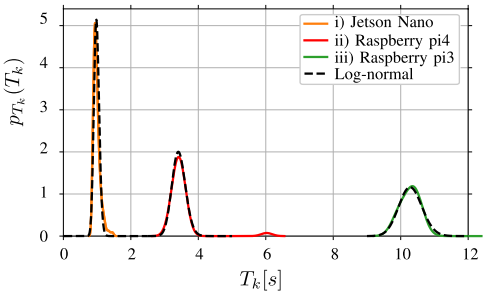


Fig. 2. In orange, red and green the probability density function $p_{T_k}(T_k)$ of a client with hardware ARM-Cortex-A57 SoC, GPU: 128-core Maxwell (*Jetson Nano* model, i), ARM-Cortex-A72 SoC (*Raspberry pi4*, ii), ARMv8-Cortex-A53 SoC (*Raspberry pi3*, iii), respectively. With dotted black line we represent the log-normal distribution that fits the real probability density function of T_k. $M(k)$ is set to 2 and the model size S is 51 KB.

global model updates, namely $M(k) > M_U(k)$, might produce biased local models when $T_{PS} \ll A(k)$. These could negatively contribute to the FL process by either slowing down convergence, reducing the accuracy [9] or possibly preventing the device to complete the local round [10].

IV. PS Design Principles and Validation

This section proposes a policy to regulate the PS response time T_{PS} based on the knowledge of the client dispersion index $D(k)$, the dataset S_k size and possible conditions on local overfitting. The proposed policy is validated in two scenarios where clients are characterized by different traffic patterns, namely varying computing capabilities, and non Independent and Identical Distributed (non-IID) datasets. Validation is based on a FL platform prototype.

A. FL Network Platform and Traffic Modelling

Fig. 2 provides a validation of the proposed client-specific traffic modelling approach based on moment matching. We consider a realistic pool of resource-constrained devices equipped with: i) CPU ARM-Cortex-A57 and GPU 128-core Maxwell (*Jetson Nano* model [11], orange), ii) CPU ARM-Cortex-A72 SoC (*Raspberry pi4*, red) and iii) CPU ARMv8-Cortex-A53 SoC (*Raspberry pi3*, green). For each client, we collected measurements of local round times T_k to obtain the sample probability functions $p_{T_k}(T_k)$. Notice that each client is connected via WLAN to a router which forwards the MQTT packets to the PS. The traffic parameters T_k and $D(k)$, are computed directly by clients at the end of each local round and then sent through a dedicated connection to the PS. The measured statistics $p_{T_k}(T_k)$ are thus reliable and realistic as they are independent from the PS hardware or from the FL processing. As evident from Fig. 2, the local round time distributions are well approximated by log-normal (dashed lines) with mean and standard deviation of 1/0.07, 3.4/0.2 and 10/0.3 for clients i), ii) and iii), respectively.

The goal of the following tests is to analyze the impact of client heterogeneity on PS response time T_{PS}. Based on experiments in Fig. 2, we simulate different execution times of the local rounds according to the log-normal model.

Algorithm 1 T_{PS} Policy

procedure POLICY($\mathcal{S}_k^{\text{train}}, \mathcal{S}_k^{\text{val}}$) ▷ Run on client k
  Initialize $\mathbf{w}_{k,0}$, $M(k) \leftarrow 1$ ▷ Epoch 0
  Train local model using $\mathcal{S}_k^{\text{train}}$
  Compute $D(k)$ with (4)
  Compute $M_L(k)$ with (6), $M_U(k)$ with (7)
  Compute performance metric: $P = \mathcal{P}(\mathbf{w}_{k,M_U(k)}, \mathcal{S}_k^{\text{val}})$
  $M^*(k) \leftarrow \max(\mathcal{F}[D(k), P], M_L(k))$
  $T_k^* \leftarrow T_{\text{down}}(k) + M^*(k)T_{\text{train}}(k) + T_{\text{up}}(k)$
  Return $T_{PS}^*(k) \leftarrow A^*(k) = \mathbb{E}_n[T_k^*]$
end procedure

B. Client-Specific Policy for the PS Response Time

The choice of the PS response time T_{PS} must take into account both the traffic model of Sec. II and the requirements of Sec. III. The optimal server time T_{PS}^* corresponds to a value $M^*(k)$ bounded by $M_L(k)$ and $M_U(k)$. The main idea, shown in Algorithm 1, is that each client computes its own optimal $M^*(k)$:

$$M^*(k) = \max(\mathcal{F}[D(k), P], M_L(k)), \tag{8}$$

where $\mathcal{F}$ is a policy function. Function $\mathcal{F}$ takes as input the local accuracy P and the traffic type $D(k)$. It can be written analytically as:

$$\mathcal{F}[D(k), P] = Q_k(\gamma P) - C \cdot D(k), \tag{9}$$

where $Q_k(\gamma P) = \{m : \mathcal{P}(\mathbf{w}_{k,m}, \mathcal{S}_k^{\text{val}}) = \gamma P\}$ is the number of epochs that corresponds to a validation accuracy of γP, and $\mathcal{P}(.)$ is the cross-entropy accuracy function. $P = \mathcal{P}(\mathbf{w}_{k,M_U(k)}, \mathcal{S}_k^{\text{val}})$ is obtained at local epoch $M_U(k)$, $C > 0$ is a constant (see Sec. IV-C) and $0 < \gamma < 1$ is a hyper-parameter. Optimal $M^*(k)$ is bounded as $M^*(k) \geq M_L(k)$ from (8), and $M^*(k) \leq M_U(k)$ which follows from (9), since $C \cdot D(k)$ is a positive term and $Q_k \leq M_U(k)$ as $\gamma P < P$.

By replacing $M^*(k)$ in (2), each client derives the PS time $T_{PS}^*(k) = A^*(k)$ using the device-specific parameters $M_U(k)$, $M_L(k)$ and $D(k)$, as well as the training $\mathcal{S}_k^{\text{train}}$ and validation datasets $\mathcal{S}_k^{\text{val}}$, respectively, as inputs. The device returns the $T_{PS}^*(k)$ value to the PS which makes a final decision for T_{PS}. The traffic statistics, the upper and lower bounds, $M_U(k)$ and $M_L(k)$, are obtained independently by each client during an initial training stage using $M(k) = 1$. The log-normal model parameters (3) are computed by means of consecutive time measurements T_k, that account for global model download, local training and model upload steps as in (2). After the training stage, we obtain the performance metric P and apply the policy function $\mathcal{F}$ in (8) using P and VMR $D(k)$.

Fig. 3 shows an example of local training, with loss and validation accuracy P for varying local epochs. Notice that few epochs are typically sufficient to improve the local model without incurring in overfitting. Cross-entropy function $\mathcal{P}$ in FL also follows a negative exponential behavior, namely $\mathcal{P} \approx a - e^{-bm}$, for $m < M_U$, while for such case $\gamma = 0.5$ is found as reasonable (see Sec. IV-C). With the proposed policy we avoid the overfitting region, transferring at the same time a great portion of local information. The term $C \cdot D(k)$ in (9)

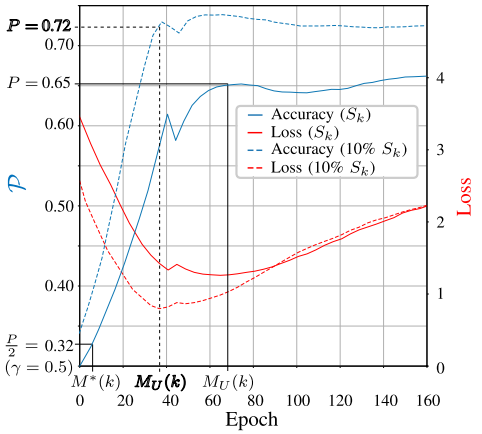


Fig. 3. Example of validation accuracy (blue) and loss (red) during local model training on a client. $M_U(k)$ and $M^*(k)$ values are highlighted together with accuracy P and γP ($\gamma = 0.5$) respectively. Solid and dotted lines are obtained with 100% and 10% of the training dataset (of size S_k). Note that overfitting is expected, as the local training process uses few training samples.

accounts for the traffic variance. Intuitively, a high variance, as a result of a client with varying computing resources, might increase the probability of observing high local training intervals (slow client). For such scenario, choosing a low value of $M(k)$ allows the PS to process the client local model updates more often and compensate for this effect.

The PS collects the optimal $T^*_{PS}(k)$ computed by Algorithm 1 and derives the response time to be used for all clients. Considering the previous analysis, this is obtained as:

$$T^*_{PS} = \min_k T^*_{PS}(k), \tag{10}$$

with the requirements on the efficiency already satisfied by $T^*_{PS}(k)$ since $M_L(k) \leq M^*(k) \leq M_U(k), \forall k$.

In the following, we explore two scenarios in detail. In the first one, the clients are homogeneous as featuring the same VMR $D(k)$, $M^*(k)$ and $T_{\text{train}}(k)$. In the second scenario, the clients are heterogeneous and organized into two clusters (C_1 and C_2). Clients in each cluster $k \in C_i$ have similar computing power/capabilities, namely $A(k) = A_i$, $\sigma^2(k) = \sigma^2_i$ and VMR $D(k) = D_i$, $\forall k \in C_i$ corresponding to the same processing unit, and TPU, if any. Accuracy improvements obtained by following the policy (10) are assessed in both cases.

C. Experimental Assessment

For the experiments, we consider the CIFAR10 [12] dataset using the full validation data and a local training set of $S_k = 500$ images for each of the $K = 9$ clients. Policy validation is based on a real-time FL platform prototype featuring physically separated clients (here *Jetson Nano* devices). The approach adopted for modelling the client heterogeneity is twofold. First, we used an additive delay whose log-normal distribution fits the observed completion times in Fig. 2. Second, the training data is non-IID distributed. To simplify the analysis, the data size is the same for all clients (as typical in FL). The prototype consists of devices connected via WLAN to the PS, thus permits to quantify the impact

TABLE I

HOMOGENEOUS CLIENTS AND NON-IID DATA. OPTIMAL T^*_{PS} AND $M^*(k)$ FOR VARYING $A(k)$, $D(k)$, η/η_{MAX} (%), AND CORRESPONDING ACCURACY IMPROVEMENT (%) W.R.T. VANILLA ASYNCHRONOUS FL

	$A(k)/D(k)$	T^*_{PS}	η/η_{MAX}[%]	$M^*(k)$	[%]
Bernoulli	2/0.03	12	84	6.00	7.1
	10/0.0064	59	17	5.90	0.7
	20/0.0033	136	7	6.80	12.5
	30/0.0021	90	11	3.00	6.0
	40/0.0016	6	168	0.15	3.0
	80/0.0008	6	168	0.07	1.8

(a)

Poisson	2/4.45	9	112	4.50	10.6
	10/0.89	42	24	4.20	9.6
	20/0.44	84	12	4.20	11.3
	30/0.29	76	13	2.53	3.5
	40/0.22	52	19	1.30	1.7
	80/0.11	49	20	0.60	0.6

(b)

Pascal	2/20.53	3	337	1.50	7.9
	10/4.10	15	67	1.50	6.3
	20/2.05	28	36	1.40	3.3
	30/1.36	48	21	1.60	1.8
	40/1.02	66	15	1.65	0.7
	80/0.51	132	7	1.65	2.5

(c)

of communication impairments. The adopted FL algorithm is FedAvg [9] while Adam [13] is used as local optimizer with default hyper-parameters. The ML model is described in Fig. 1 and consists of 10^5 parameters with size $G = 2.53$ MB. Loss and performance metrics are the categorical cross-entropy and categorical accuracy, respectively. For each considered case, the validation accuracy is evaluated after $T_{FL} = 800$ seconds. Considering the communication with the PS, the bandwidth is $B = 40$ MHz while the max. efficiency dedicated to FL is set to $\eta_{MAX} = 0.05$ bit/sec/Hz. MQTT publishing uses Quality of Service (QoS) level 2 as this permits re-transmissions in exchange for larger $T_{\text{up}}(k)$ and $T_{\text{down}}(k)$.

As previously described, the FL process starts with a client local training to find $M_U(k)$ and subsequently $M^*(k)$ and $T^*_{PS}(k)$ according to Algorithm 1, with $\gamma = 0.5$. $C = 0.2$ adapts the VMR $D(k)$ contribution in (9) to the considered traffic types. Fig. 3 shows the validation loss/accuracy versus the local epochs that are used to retrieve $M_U(k)$ and $M^*(k)$ from (7) and (9). For clients with training data size S_k, the optimal $M^*(k)$ is 6 epochs. Setting now the training size to 10%, we observe shifts in the overfitting region (around $M_U(k)$) according to the bias-variance of the model: now $M^*(k) = 4$.

In Table I we consider at first homogeneous clients, i.e., clients with the same computing power and traffic distribution, but with non-IID data (80% of the local samples are drawn from the same class, chosen randomly). Traffic parameters are set to vary within the set $A(k) \in [2, 80]$ s and $\sigma^2(k) \in [0.001, 100]$. To evaluate the proposed policy, we choose $T_{PS} \in [1, 400]$ s and obtain the empirical optimal T_{PS} for each type of traffic. Notice that for the proposed example, setting $\eta_{MAX} = 0.05$, a client with $A(k) \leq 80$ s would require $M_L(k) = 1$. Table I summarizes the results for all experiments

TABLE II

CLIENTS ORGANIZED IN TWO CLUSTERS (C_1 AND C_2) AND NON-IID DATA (SEE TABLE I). THE RATIO η/η_{MAX} (%) REFERS TO THE WORST CLUSTER CASE, I.E., LOWER $M^*(k)$

		A_i/D_i	T^*_{PS}	η/η_{MAX}[%]	$M^*(k)$	[%]
Bernoulli	C_1 C_2	8/0.0084 10/0.0050	55	18	6.87 5.50	7.4
	C_1 C_2	4/0.014 20/0.0022	25	40	6.25 1.25	15.9
	C_1 C_2	2/0.033 40/0.0016	14	72	7.00 0.35	6.0
	C_1 C_2	1/0.061 80/0.0008	6	168	6.00 0.07	27.1

(a)

Poisson	C_1 C_2	8/1.13 10/0.91	37	27	4.62 3.70	14.4
	C_1 C_2	4/2.15 20/0.41	20	50	5.00 1.00	17.5
	C_1 C_2	2/4.22 40/0.19	7	144	3.50 0.17	3.3
	C_1 C_2	1/8.83 80/0.11	4	253	4.00 0.05	26.4

(b)

Pascal	C_1 C_2	8/5.11 10/4.23	17	59	2.12 1.70	7.7
	C_1 C_2	4/11.12 20/2.15	9	112	2.25 0.45	3.8
	C_1 C_2	2/22.07 40/1.18	4	253	2.00 0.10	21.2
	C_1 C_2	1/43.92 80/0.54	3	337	3.00 0.03	22.9

(c)

grouped based on the traffic type (Bernoulli, Poisson, Pascal), mainly determined by the value of $\sigma^2(k)$. For each experiment, we highlight the traffic parameters $A(k)$, $D(k)$ (obtained for $M(k) = 1$), the resulting optimal T^*_{PS}, the corresponding $M^*(k)$ and the fraction (%) of utilized bandwidth w.r.t η_{MAX}. Last column quantifies the accuracy improvement w.r.t. a vanilla asynchronous strategy where the PS updates the global model as soon as a local model is available.

Considering the Bernoulli distribution (Table I.a) and $D(k) \approx 0$, the optimal values of T_{PS} and $M(k)$, obtained empirically, are in-line with the model (9), that gives $M^*(k) = 6$ for $\gamma = 0.5$ and $C = 0.2$. On the other hand, when $A(k) > 20$, namely the clients being all very slow, it is more convenient to keep the T_{PS} as low as possible, rather than following the policy (waiting the end of the optimized round). Increasing $D(k)$, namely using Poisson and Pascal traffic, this behaviour is less evident as the optimal $M^*(k)$ (4 and 1, respectively) is now in line with the policy and maintained for all $A(k)$. To summarize, the policy is effective for the prediction of the optimal values of $M^*(k)$ and can be used to tune the T^*_{PS} when $A(k) \ll T_{FL}$ (see cases highlighted in green).

Table II analyzes a more general case where the clients belong to clusters C_1 and C_2 and have different local learning completion times. Cluster C_2 contains much slower clients compared with C_1: the example is thus useful to verify the proposed policy for T_{PS} and whether the PS should specifically follow any cluster C_i, or not. To achieve that, we vary $A(k) = A_{i=1,2}$ of both clusters within the set $[1, 80]$ s, $T_{PS} \in [1, 400]$ s and $\sigma^2_{i=1,2} \in [0.001, 100]$ s^2. Cluster C_1 contains 5 clients while the remaining 4 clients belong to C_2. The results highlighted in blue show that the optimized T_{PS} should be set to follow the optimal number of local rounds $M^*(k)$ of the faster clients. For example, this can be seen in the extreme case of $(A_1 = 1, A_2 = 80)$ where the cluster C_2 almost does not affect the choice of T^*_{PS}. For all the considered cases, the use of an underestimated value of $M^*(k)$ should be preferred to prevent overfitted local models. In other words, it is more beneficial to update more often the global model following the faster clients, $k \in C_1$, as opposed to wait for the slower clients, $k \in C_2$, to complete their round. To conclude, we observe that designing T_{PS} based on the knowledge of the client-specific traffic is able to outperform asynchronous FL strategies. Optimization of T_{PS} is particularly critical for the case of two clusters. Observed accuracy gain increases from 5.1% (single cluster), to 15-17% on average.

V. CONCLUSION

The letter proposed a stochastic traffic model to describe the clients' behavior in FL processes. The model is based on the moment-matching approximation and it is verified with practical resource-constrained devices communicating with a Parameter Server (PS) using MQTT transport. Traffic characterization is used to develop a policy for the selection of the PS response time T_{PS} in asynchronous FL. The policy satisfies spectral efficiency constraints and avoids FL overfitting impairments. Results obtained in real setups show that an accuracy increase of up to 15-17% is possible when clients exhibit different local model completion times.

REFERENCES

[1] P. Kairouz et al., "Advances and open problems in federated learning," 2019, *arXiv:1912.04977*.

[2] W. Y. B. Lim et al., "Federated learning in mobile edge networks: A comprehensive survey," *IEEE Commun. Surveys Tuts.*, vol. 22, no. 3, pp. 2031–2063, 3rd Quart., 2020.

[3] M. M. Amiri and D. Gündüz, "Machine learning at the wireless edge: Distributed stochastic gradient descent over-the-air," *IEEE Trans. Signal Process.*, vol. 68, pp. 2155–2169, 2020.

[4] B. C. Tedeschini et al., "Decentralized federated learning for healthcare networks: A case study on tumor segmentation," *IEEE Access*, vol. 10, pp. 8693–8708, 2022.

[5] Y. Chen, Y. Ning, M. Slawski, and H. Rangwala, "Asynchronous online federated learning for edge devices with non-IID data," in *Proc. IEEE Int. Conf. Big Data (Big Data)*, Dec. 2020, pp. 15–24.

[6] Z. Wang et al., "Asynchronous federated learning over wireless communication networks," *IEEE Trans. Wireless Commun.*, vol. 21, no. 9, pp. 6961–6978, Sep. 2022.

[7] (2021). *MQTT V3.1 Protocol Specification*. [Online]. Available: https://mqtt.org

[8] R. I. Wilkinson, "Theories for toll traffic engineering in the USA," *Bell Syst. Tech. J.*, vol. 35, no. 2, pp. 421–514, Mar. 1956.

[9] H. B. McMahan et al., "Communication-efficient learning of deep networks from decentralized data," in *Proc. 20th Int. Conf. Artif. Intell. Statist.*, Apr. 2017, pp. 1273–1282.

[10] K. Bonawitz et al., "Towards federated learning at scale: System design," in *Proc. Mach. Learn. Syst.*, vol. 1, 2019, pp. 374–388. [Online]. Available: https://tinyurl.com/34bwmd5m

[11] *Jetson Nano Developer Kit*. Accessed: Mar. 1, 2021. [Online]. Available: https://tinyurl.com/52mdbjzh

[12] A. Krizhevsky and G. Hinton, "Learning multiple layers of features from tiny images," M.S. thesis, Dept. Comput. Sci., Univ. Toronto, Toronto, ON, Canada, 2009.

[13] D. P. Kingma et al., "Adam: A method for stochastic optimization," May 2015, *arXiv:1412.6980*.

Open Access funding provided by 'Politecnico di Milano' within the CRUI CARE Agreement

Weighted Average Consensus Algorithms in Distributed and Federated Learning

Bernardo Camajori Tedeschini, *Graduate Student Member, IEEE*, Stefano Savazzi, *Member, IEEE*, Monica Nicoli, *Senior Member, IEEE*

***Abstract*—The exponential growth of the Internet of Things (IoT) has created an essential demand for Distributed Machine Learning (DML) systems. In this context, Federated Learning (FL) allows IoT devices to collaboratively train models while maintaining data ownership and privacy. Despite the evident advantages, FL faces practical challenges such as client selection and adaptation to heterogeneous data distributions. Recently, consensus-driven algorithms have been proposed to enable efficient and scalable FL without a central coordinating entity. Weighted Average Consensus (WAC) tools, primarily used in distributed signal processing, fail to address FL-specific challenges. The paper proposes a new family of server-less FL algorithms optimized to exploit WAC techniques. In particular, we propose an evolution of the centralized Federated Adaptive Weighting (FedAdp) method and present three distinct WAC schemes specifically designed for non-Independent and Identical Distributed (IID) data. Each scheme has a unique aggregation part that optimizes the weights of the clients' local models. The performances are evaluated in a real-world IoT system, analyzing their convergence properties in the context of heterogeneous client populations. Results show that the proposed algorithms outperform vanilla consensus FL up to 56% of accuracy and they are resilient to both label and sample data skewness.**

***Index Terms*—FL over networks, weighted average consensus, non-independent and identical distributed, edge computing.**

I. Introduction

THE rapid growth of the Internet of Things (IoT) and the diffusion of devices with increasing computational capabilities created a significant need for Distributed Machine Learning (DML) [1]. Federated Learning (FL) is a DML approach that allows devices to collaboratively train models while keeping their data local, thus preserving privacy and security [2]–[5]. In vanilla FL framework, a central entity, i.e., Parameter Server (PS), coordinates the learning process among the participating devices, or clients, by aggregating their locally computed model updates. This technique is particularly relevant in 5G/6G networks [6]–[8], where devices can efficiently communicate and share data, enabling a wide range of applications such as autonomous driving, smart cities, and advanced healthcare services [9]–[12].

Manuscript received XX YYYY 2024; revised XX YYYY 2024; accepted XX YYYY 2025. Date of publication XX YYYY 2025; date of current version XX YYYY 2025. (Corresponding author: B. Camajori Tedeschini.)

B. Camajori Tedeschini and M. Nicoli are with Politecnico di Milano, Milan, Italy (e-mail: bernardo.camajori@polimi.it, monica.nicoli@polimi.it).

S. Savazzi is with Consiglio Nazionale delle Ricerche (CNR), Milan, Italy (e-mail: stefano.savazzi@cnr.it).

Color versions of one or more figures in this article are available at https://doi.org/10.1109/TNSE.2025.XXXXXXX.

Digital Object Identifier 10.1109/TNSE.2025.XXXXXXX

While FL offers undoubted advantages, it also faces several challenges that must be addressed to ensure robustness, efficiency, and security across various application domains [13]–[15]. These challenges include client-selection [16]–[20], which involves determining the optimal set of clients to participate in the training process; energy consumption [21]–[24], as IoT devices often have limited battery life, calling for the design of energy-efficient FL algorithms; and incentive mechanisms [25], which foster cooperation among clients by rewarding them for their contributions. Additionally, Over-The-Air (OTA) computations and communications [26]–[28] require efficient techniques for data transmission and model aggregation while minimizing latency and bandwidth.

Alongside the aforementioned issues, security remains a major concern in FL [29]–[31], as the system is vulnerable to various attacks such as data poisoning and model manipulation. Another aspect to consider is online learning [32]–[34], which necessitates algorithms capable of adapting to dynamic and evolving data distributions. Furthermore, achieving fast-convergence [35], [36] and time-efficient asynchronous FL [37]–[40] is crucial for practical implementation, particularly in scenarios with limited connectivity or highly dynamic environments. Addressing these various aspects is essential for unlocking the full potential of FL and ensuring the successful deployment of FL-based systems across a wide range of applications.

A. Related Works

One of the main concerns in FL is the non-Independent and Identical Distributed (IID) data distribution among clients [41]. Indeed, in case of data heterogeneity, traditional methods like Federated Averaging (FedAvg), which rely on local Stochastic Gradient Descent (SGD), may struggle to achieve convergence when the participating devices execute an excessive number of local updates or have skewed data distributions. To overcome this limitation, adaptive learning rate strategies [42]–[44] and smart clustering or pooling techniques [45]–[47] has been introduced in the last years.

Of particular interest are the works of FedProx [48] and its variants [36], [49] which employ an inexact proximal point update for local optimization, i.e., penalizing the deviation of the local model from the PS global one. While FedProx is performed individually by each client, other algorithms focused on developing ad-hoc PS aggregation weighting to regulate the impact of each local model in the global update. An example can be found in [50] where the authors developed

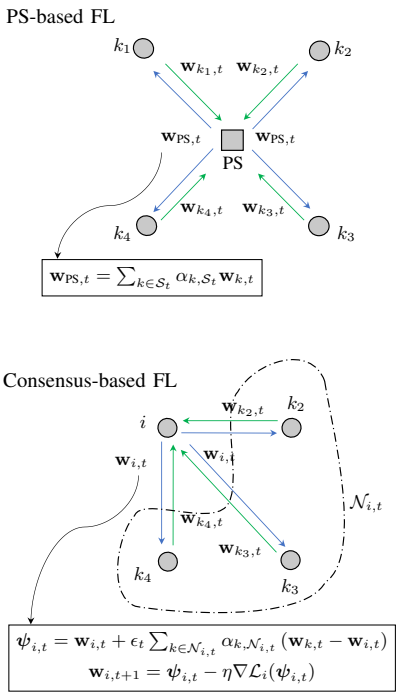


Fig. 1. Comparison of PS and consensus-based architecture in a network with four clients. The aggregation weights $\alpha_{k,\mathcal{S}_t}$ and $\alpha_{k,\mathcal{N}_{i,t}}$ are computed and applied in the first case by the PS and in the second case by each individual client (node i in the figure).

a Federated Adaptive Weighting (FedAdp) algorithm which employs PS aggregation weights based on the inner product between local gradients and the global gradient. Indeed, this metric can be used as a dis/similarity measure to estimate the contribution of the local model. Intuitively, the more the local and global gradients are orthogonal, the less the local model will positively contribute to the global aggregation.

Recently, a distributed version of FL, known as consensus-based FL [51]–[54], has been proposed to address some of the challenges associated with PS-based FL, i.e., with centralized approaches [55]. As depicted in Fig. 1, differently from PS-based FL, in consensus-FL, clients collaboratively train models while also reaching a consensus on the model updates without a central coordinating entity, which leads to a more efficient and scalable learning. These techniques have evolved from conventional distributed maximum likelihood estimation based on consensus [56], where individual nodes depend exclusively on their local data and the information shared via connections with nearby nodes to update their local approximations. In the simplest version of consensus, i.e., Averaged Consensus (AC) [57], the model parameters are updated in a synchronized fashion and with constant or absence of weighting aggregation. Similar to AC, consensus-FL faces challenges such as asynchronous and fast convergence [58], [59], as well as reducing the carbon footprint [60]. To address these issues, consensus-FL algorithms, such as Gossip FL [61], Consensus-driven Federated Averaging (CFA) [62], [63], and dynamic layer selection [64], [65], have been proposed.

In distributed estimation, smarter Weighted AC (WAC) has been proposed [66] to extend the continuous-time approach [67] to a general discrete-time vector-parameter estimation problem. Recently, attempts have been made to apply WAC algorithms to FL under non-IID data distributions and network dynamics. For example, in [68], the authors used simulated clients with label-skewness (each client possessing data from only one or a few classes) but did not explore sample-skewness scenarios or conduct experiments involving real devices. Similarly, in [69], the focus was on label-skewness with simulations on time-varying topologies, but again, sample-skewness and real-world network dynamics were not considered. Moreover, these approaches typically employ conventional CFA algorithms, where the weighting is based solely on the number of samples in the local datasets. Such methods do not account for real network dynamics and do not exploit the learning contribution of each node when determining the appropriate weights for model aggregation. Given the relevance of applications like massive-IoT networks [53], vehicular communications [70], and industrial networks [71], where non-IID data distribution and network dynamics are common, the adoption of WAC in FL systems to handle these challenges is clearly a research direction to explore.

B. Contribution

The design of adaptive weighting algorithms for PS-based FL has been extensively studied in the literature and main challenges can be considered well understood. On the other hand, transferring algorithms optimized for conventional FL architectures to a fully decentralized platform (with no physical PS server) is challenging and partially addressed. The current literature shows a lack of consensus-based algorithms specifically designed for non-IID frameworks, since solutions developed for centralized FL contexts can not be directly applied to fully-distributed setups. To move a step forward in this direction, this paper proposes WAC algorithms designed for decentralized FL setups and under heterogeneous client populations assumptions. The proposed solutions extend popular FL techniques, such as FedAdp, broadening their scope of applicability in distributed contexts, where the adaptive aggregation of model parameters and the optimization strategy are performed client-side without a coordinating PS-part.

In summary, the main contributions are as follows.

- We make a comparative analysis on PS and consensus-based FL, highlighting the key similarities and differences, and investigating the main schemes for centralized weighted FL;
- We propose three different solutions for achieving weighted FL in decentralized network architectures, which mainly differ for the aggregation part according to the local contribution of the neighbors (based on a

virtual PS, on a selected client or on the local retained model). To the best of our knowledge, this is the first attempt to extend conventional WAC techniques to fully-decentralized FL with consensus;
- We analyze the convergence properties and evaluate the FL performance on a real platform consisting of heterogeneous IoT devices. The heterogeneity is taken into account by introducing asymmetries in both samples and label distributions, namely sample and label skewness, and assessed through different models and datasets complexities.

C. Paper Organization

The structure of this paper is as follows. Section II provides an overview of PS and consensus-based FL, together with a description of FL for non-IID local distributions. In Section III, we describe the weighted consensus algorithms, i.e., WAC for FL, detailing the main steps and convergence properties of the proposed algorithms. Section IV presents details about the dataset and FL platform, both simulated and with real IoT devices, followed by a discussion of the numerical results obtained with different non-IID data characterizations. Finally, Section V concludes the paper.

II. System Model

In this section, we first describe the vanilla PS-based FL tools and the consensus-based solutions for server-less architectures. Next, we discuss the main existing categories of FL algorithms for non-IID data.

A. From PS-based to Server-less FL Driven by Consensus

In the framework of FL, we consider a network that includes one PS and a collection of K clients denoted as $\mathcal{K} = \{1, \dots, K\}$. For notation purposes, throughout the paper, we will indicate with subscripts i and k the indices for the clients and neighbors, respectively. Each client possesses its own distinct dataset $\mathcal{D}_i$ with a size $D_i = |\mathcal{D}_i|$. The ultimate loss of the FL procedure is to achieve a global Deep Learning (DL) model that minimizes a loss function $\mathbf{w}_{\text{PS}} = \text{argmin}_{\mathbf{w}} \mathcal{L}(\mathbf{w})$, where $\mathcal{L}(\mathbf{w}) = \frac{1}{K}\sum_{i=1}^{K} \frac{D_i}{\sum_{i' \in \mathcal{K}} D_{i'}} \mathcal{L}_i(\mathbf{w}, \mathcal{D}_i)$, $\mathcal{L}_i$ represents the local cost determined by client i utilizing the data batches $\mathcal{D}_i$. An iterative process, involving a *local model optimization* step followed by an *aggregation* step executed on the PS, is used to obtain the global model.

At time (i.e., federated round) $t = 1, \dots, N_{\text{FL}}$, a set $\mathcal{S}_t \subseteq \mathcal{K}$ of clients is chosen to carry out the training procedure. Clients are required to generate local models via optimization in the FL process, typically employing supervised and gradient-based techniques, e.g., Adam optimizer [72], with mini-batch $\mathcal{B}$ of size B and learning rate η. Each client $i \in \mathcal{S}_t$ performs E local epochs prior to exchanging the local model with the PS, which is responsible for updating the global model. In the vanilla FL using PS, i.e., FedAvg, the *aggregation* step is conducted using a weighted average based on the number of samples D_i from each client:

$$\mathbf{w}_{\text{PS},t} = \sum_{i \in \mathcal{S}_t} \alpha_{i,\mathcal{S}_t} \mathbf{w}_{i,t}, \tag{1}$$

where $\mathbf{w}_{\text{PS},t}$ is the PS global model, $\mathbf{w}_{i,t}$ is the local model of client i and $\alpha_{i,\mathcal{S}_t} = \frac{D_i}{\sum_{i' \in \mathcal{S}_t} D_{i'}}$ are the mixing weights.

Decentralized FL architectures do not employ the PS but rather share their local model(s) repeatedly over Device-to-Device (D2D) links so as to reach a consensus on a global model (consensus-based FL): the clients form a graph $\mathcal{G}_t = (\mathcal{V}_t, \mathcal{E}_t)$, where each node $i \in \mathcal{V}_t$ corresponds to a client/learner, while the edge $(i, j) \in \mathcal{E}_t$, with $i \neq j$, signifies the presence of a communication link from client i to client j. The consensus-based algorithm, referred to as CFA, operates as follows. At round t, each client $i \in \mathcal{S}_t$ performs a *local model optimization* step and then exchanges the model parameters with its neighbors $\mathcal{N}_{i,t}$. Subsequently, an *aggregation* step is executed, similar to the PS [53]:

$$\boldsymbol{\psi}_{i,t} = \mathbf{w}_{i,t} + \epsilon_t \sum_{k \in \mathcal{N}_{i,t}} \alpha_{k,\mathcal{N}_{i,t}} \left(\mathbf{w}_{k,t} - \mathbf{w}_{i,t}\right), \tag{2}$$

where $\boldsymbol{\psi}_{i,t}$ represents the aggregated model, ϵ_t is the consensus step-size which modulates the memory of previous models and $\alpha_{k,\mathcal{N}_{i,t}} = \frac{D_k}{\sum_{k' \in \mathcal{N}_{i,t}} D_{k'}}$ are the mixing weights related to client k, based on the number of samples retained in each client. While the weights $\alpha_{k,\mathcal{N}_{i,t}}$ enable CFA to partially cope with sample unbalances, they do not fully capture the information gain of each local model in case of different data qualities or other types of non-IID data imbalances. Note that the aggregated model $\boldsymbol{\psi}_{i,t}$ represents an estimate of the global model as seen by client i and its neighborhood $\mathcal{N}_{i,t}$. However, as opposed to (1), here the aggregated model is obtained by taking into account the error between the local model and the neighbor ones. It is worth noting that the algorithm operates in the same manner if clients exchange gradients of the local model update instead of model parameters.

B. FL for non-IID

The importance of IID sampling in training data lies in the fact that it ensures the stochastic gradient is an unbiased estimate of the full gradient. FedAvg and CFA are known to be effective when data distribution across different nodes is the same as for the centrally collected data. However, in practice, data distribution across local nodes is typically non-IID, resulting in local losses $\mathcal{L}_i(\mathbf{w}_{\text{PS},t})$ to be closely related to data distribution $\mathcal{D}_i$ and local updates to gravitate towards the optima of its local loss $\mathcal{L}_i(\mathbf{w}_{\text{PS},t})$ rather than the global loss $\mathcal{L}(\mathbf{w}_{\text{PS},t})$. The inconsistency between local models $\mathbf{w}_{i,t}$ and the global model $\mathbf{w}_{\text{PS},t}$ accumulates during local training, necessitating more communication rounds for convergence. Consequently, multiple local updates during local training can potentially harm convergence and even cause divergence in the presence of non-IID data [3].

When it comes to design PS-based FL algorithms for non-IID data, two major categories can be distinguished: either based on the local model optimization (e.g., adding a penalizing loss term) or either based on the global model aggregation step (e.g., weighting the local model contributions). Examples of the first category include FedProx [48], Distributed Approximate NEwton (DANE) [73], and Federated Curvature (FedCurv) [74]. FedProx uses parameter stiffness, i.e., an isotropic penalty term in the local loss, to

avoid diverging from the global model. DANE builds upon FedProx by adding a gradient correction term to accelerate convergence, while FedCurv exploits the Fisher information matrix to protect parameters that are important to each task.

The second category of algorithms, on which we focus on, leaves unaltered the local model optimization step, while targeting the optimal aggregation weights that the PS should employ. FedAdp [50] is one of the most relevant works in this direction which aims at designing the aggregation weights $\widetilde{\alpha}_{i,\mathcal{S}_t}$, such that $\sum_{i\in\mathcal{S}_t}\widetilde{\alpha}_{i,\mathcal{S}_t}=1$, in order to increase the convergence rate especially in non-IID settings. The main idea is that the greater the difference between the global gradient (i.e., of the global model) and the local gradients of each client, the higher is the weight that should be assigned to the local update. To this aim, the PS, after receiving the local updates, computes the global gradient starting from the gradient descent update:

$$\mathbf{w}_{\text{PS},t}=\mathbf{w}_{\text{PS},t-1}-\eta\nabla\mathcal{L}(\mathbf{w}_{\text{PS},t}) \tag{3}$$

with

$$\nabla\mathcal{L}(\mathbf{w}_{\text{PS},t})=\sum_{i\in\mathcal{S}_t}\alpha_{i,\mathcal{S}_t}\nabla\mathcal{L}_i(\mathbf{w}_{\text{PS},t}), \tag{4}$$

and $\nabla\mathcal{L}_i(\mathbf{w}_{\text{PS},t})=-\frac{\Delta_i}{\eta}$ represents the approximated local gradients of client i, with $\Delta_i=\mathbf{w}_{i,t}-\mathbf{w}_{\text{PS},t-1}$. A measure of the distance between the gradients can be obtained, as proposed in [50], by using the instantaneous angle $\theta_{i,t}$:

$$\theta_{i,t}=\arccos\frac{\nabla\mathcal{L}(\mathbf{w}_{\text{PS},t})^{\text{T}}\cdot\nabla\mathcal{L}_i(\mathbf{w}_{\text{PS},t})}{\|\nabla\mathcal{L}(\mathbf{w}_{\text{PS},t})\|\|\nabla\mathcal{L}_i(\mathbf{w}_{\text{PS},t})\|}, \tag{5}$$

where $\|\cdot\|$ represents the l2 norm. Furthermore, for numerical stability purposes, a smoothed angle $\widetilde{\theta}_{i,t}$ is used:

$$\widetilde{\theta}_{i,t}=\begin{cases}\theta_{i,t} & t=1\\ \frac{t-1}{t}\widetilde{\theta}_{i,t-1}+\frac{1}{t}\theta_{i,t} & t>1.\end{cases} \tag{6}$$

Finally, the aggregation weights are obtained as:

$$\widetilde{\alpha}_{i,\mathcal{S}_t}=\sum_{i\in\mathcal{S}_t}\alpha_{i,\mathcal{S}_t}e^{f(\widetilde{\theta}_{i,t})}, \tag{7}$$

where $f(\widetilde{\theta}_{i,t})=\alpha_{\text{G}}(1-e^{-e^{-\alpha_{\text{G}}(\widetilde{\theta}_{i,t}-1)}})$ is a variant of the Gompertz function [75] and α_{G} is an hyper-parameter. We highlight that α_{G} regulates the sensitivity with respect to the smoothed angle between the gradients. The higher α_{G}, the more sensitive the output is to the smoothed angle, potentially increasing the difference in contributions from the participating clients.

III. Weighted Consensus Algorithms

In this section, we first describe the proposed weighted consensus algorithm, specifically designed to handle heterogeneous clients. Next, based on the analysis of [50], we discuss the convergence properties which exploit the weighted consensus scheme.

In fully decentralized scenarios where the PS is not available, each client member of the federation must depend on the local updates from its neighbors to compute the aggregated model: the goal is to determine the optimal aggregation weights $\widetilde{\alpha}_{i,\mathcal{S}_t}$ to be used in the aggregation step. Following the FedAdp paradigm, the main challenge is to identify the equivalent global model on which the similarity metrics should be calculated. We propose that each client hosts virtual PS functions, where $\boldsymbol{\psi}_{i,t}$ in (2) can be now interpreted as an instance of the PS global model which is observed by the local client i using the available neighbors. In what follows, we refer to this solution as Consensus-driven FedAdp virtual PS (CFAdp-vPS). An alternative approach that will be explored in the next sections, involves selecting one specific neighbor local model as the reference instance (i.e., reference model) of the global model, and computing the various gradient distances concerning it. In the subsequent sections, we provide a detailed description of the two versions of weighted consensus algorithms, emphasizing their key characteristic steps.

A. CFAdp with Virtual PS

The first CFAdp algorithm is the dual version of the vanilla PS-based FedAdp method adapted to implement linear distributed average consensus among peer clients. The method is described in Algorithm 1. Here, each client i, after receiving the neighbors local models $\mathbf{w}_{k,t}\,\forall k\in\mathcal{N}_{i,t}$, computes the aggregated gradient as:

$$\nabla\mathcal{L}(\boldsymbol{\psi}_{i,t})=\sum_{k\in\mathcal{S}_t}\alpha_{k,\mathcal{S}_t}\nabla\mathcal{L}_k(\boldsymbol{\psi}_{i,t}), \tag{8}$$

where $\nabla\mathcal{L}_k(\boldsymbol{\psi}_{i,t})=-\frac{\Delta_i}{\eta}$ are the approximated local gradients of client k and $\Delta_k=\mathbf{w}_{k,t}-\boldsymbol{\psi}_{i,t-1}$. Note that (8) is the analogous of (4) where the global model $\mathbf{w}_{\text{PS},t}$ is substituted with its *local* representation $\boldsymbol{\psi}_{i,t}$. Here, the aggregated gradients serve as an approximation of the gradients gathered and combined by a PS, connected with the neighbors $\mathcal{N}_{i,t}$ and the node i itself. The instantaneous angles are computed similarly to (5) in lines 13 of Algorithm 1:

$$\theta_{k,\mathcal{S}_t}=\arccos\frac{\nabla\mathcal{L}(\boldsymbol{\psi}_{i,t})^{\text{T}}\cdot\nabla\mathcal{L}_k(\boldsymbol{\psi}_{i,t})}{\|\nabla\mathcal{L}(\boldsymbol{\psi}_{i,t})\|\|\nabla\mathcal{L}_k(\boldsymbol{\psi}_{i,t})\|}, \tag{9}$$

where the global gradients are substituted with the aggregated gradients, i.e., $\nabla\mathcal{L}(\boldsymbol{\psi}_{i,t})$ with respect to the aggregated model $\boldsymbol{\psi}_{i,t}$ in client i. The instantaneous angle directly relates to the amount of information that can be injected from client k into the learning system. Note that client i has no advantage with respect to all the other clients and that its model contribution can be made negligible according to the angles. Consequently, the new aggregated model is estimated with:

$$\boldsymbol{\psi}_{i,t}=\sum_{k\in\mathcal{S}_t}\widetilde{\alpha}_{k,\mathcal{S}_t}\mathbf{w}_{k,t}, \tag{10}$$

where the mixing weights $\widetilde{\alpha}_{k,\mathcal{S}_t}$ are obtained in line 16 as:

$$\widetilde{\alpha}_{k,\mathcal{S}_t}=\frac{D_k e^{f(\widetilde{\theta}_{k,\mathcal{S}_t})}}{\sum_{k'\in\mathcal{S}_t}D_{k'}e^{f(\widetilde{\theta}_{k',\mathcal{S}_t})}}\quad\forall k\in\mathcal{S}_t. \tag{11}$$

The mixing weights in (11) are obtained similarly to a softmax function, but with the usage of the Gompertz function. This function enables a slow and gradual variation of the weights for big angles, ensuring that major differences between clients do not lead to abrupt changes in the aggregation weights. As the angles decrease, the function allows for a more significant adjustment in the weights, enabling clients with similar, i.e., smaller angles, to have a more substantial impact on the consensus. Finally, the local model optimization step is performed starting from $\boldsymbol{\psi}_{i,t}$ as in CFA.

Algorithm 1 Consensus-driven FedAdp virtual PS:

procedure CFADP-VPS($\mathcal{N}_{i,t}, \alpha_{k,\mathcal{S}_t}$) ▷ Run on client i
  authentication with network broker
  receive parameters (E, B) ▷ RX from broker
  initialize $\mathbf{w}_{i,0} \leftarrow$ device i
  for each round $t = 1, 2, \ldots$ **do** ▷ Training loop
    receive$\{\mathbf{w}_{k,t}\}_{k\in\mathcal{N}_{i,t}}$ ▷ RX from broker
    $Dec\{\mathbf{w}_{k,t}\}_{k\in\mathcal{N}_{i,t}}$ ▷ Decipher weights
    $\nabla\mathcal{L}_k(\boldsymbol{\psi}_{i,t}) = -\frac{\Delta_k}{\eta} \quad \forall k \in \mathcal{S}_t$
    $\nabla\mathcal{L}(\boldsymbol{\psi}_{i,t}) = \sum_{k\in\mathcal{S}_t} \alpha_{k,\mathcal{S}_t} \nabla\mathcal{L}_k(\boldsymbol{\psi}_{i,t})$
    $\theta_{k,\mathcal{S}_t} = \arccos \frac{\nabla\mathcal{L}(\boldsymbol{\psi}_{i,t})^{\mathrm{T}} \cdot \nabla\mathcal{L}_k(\boldsymbol{\psi}_{i,t})}{\|\nabla\mathcal{L}(\boldsymbol{\psi}_{i,t})\|\|\nabla\mathcal{L}_k(\boldsymbol{\psi}_{i,t})\|} \quad \forall k \in \mathcal{S}_t$
    $\widetilde{\theta}_{k,\mathcal{S}_t} = \begin{cases} \theta_{k,\mathcal{S}_t} & t = 1 \\ \frac{t-1}{t}\widetilde{\theta}_{k,\mathcal{S}_{t-1}} + \frac{1}{t}\theta_{k,\mathcal{S}_t} & t > 1 \end{cases} \quad \forall k \in \mathcal{S}_t$
    $f(\widetilde{\theta}_{k,\mathcal{S}_t}) = \alpha_{\mathrm{G}}(1 - e^{-e^{-\alpha_{\mathrm{G}}(\widetilde{\theta}_{k,\mathcal{S}_t}-1)}}) \quad \forall k \in \mathcal{S}_t$
    $\widetilde{\alpha}_{k,\mathcal{S}_t} = \frac{D_k e^{f(\widetilde{\theta}_{k,\mathcal{S}_t})}}{\sum_{k'\in\mathcal{S}_t} D_{k'} e^{f(\widetilde{\theta}_{k',\mathcal{S}_t})}} \quad \forall k \in \mathcal{S}_t$
    $\boldsymbol{\psi}_{i,t} = \sum_{k\in\mathcal{S}_t} \widetilde{\alpha}_{k,\mathcal{S}_t} \mathbf{w}_{k,t}$
    $\mathbf{w}_{i,t+1} = \boldsymbol{\psi}_{i,t} - \eta\nabla\mathcal{L}_i(\boldsymbol{\psi}_{i,t})$ ▷ Model update
    send $Enc(\mathbf{w}_{i,t+1})$ ▷ Encrypt and TX to broker
  end for
end procedure

We want to point out that the CFAdp algorithm considers both the volume of data and the contribution of each node in the neighborhood (i.e., the correlation between the local and aggregated gradients obtained from neighbors) when determining the appropriate weights for model aggregation. As described in Sec. III-C, this allows to infer an upperbound to the rate at which the overall FL loss may decline, under specific assumptions on loss function.

B. CFAdp with Client Selection

As clarified in the analysis of Sect IV, in some cases, computing the aggregated gradients as in (8) may not be the best way to combine the local gradients observed in the neighborhood. For example, this might happen when a neighbor's local model gives a limited or negative contribution to the FL process. The proposed algorithm, referred to as CFAdp with Client Selection (CFAdp-CS), allows every client to proactively select the best neighbor model which is used to represent the new aggregated model, namely, a new reference $\boldsymbol{\psi}_{i,t}$ for the next round t.

The comprehensive pseudo-code is outlined in Algorithm 2, and it works in the following way. After receiving the neighbor models, each client i computes the instantaneous and smoothed angles, and Gompertz function for each couple of neighbors, obtaining the matrices $\theta_{k_1,k_2,t}$, $\widetilde{\theta}_{k_1,k_1,t}$ and $f(\widetilde{\theta}_{k_1,k_2,t})\,\forall k_1, k_2 \in \mathcal{S}_t, k_1 \neq k_2$, respectively. In particular, the instantaneous angles are estimated as:

$$\theta_{k_1,k_2,t} = \arccos \frac{\nabla\mathcal{L}_{k_1}(\boldsymbol{\psi}_{i,t})^{\mathrm{T}} \cdot \nabla\mathcal{L}_{k_2}(\boldsymbol{\psi}_{i,t})}{\|\nabla\mathcal{L}_{k_1}(\boldsymbol{\psi}_{i,t})\|\|\nabla\mathcal{L}_{k_2}(\boldsymbol{\psi}_{i,t})\|}. \tag{12}$$

This permits the evaluation of the similarities between clients' updates, holding a method to select the best reference clients

Algorithm 2 Consensus-driven FedAdp with Client Selection:

procedure CFADP-CS($\mathcal{N}_{i,t}, \epsilon_t$) ▷ Run on client i
  authentication with network broker
  receive parameters (E, B) ▷ RX from broker
  initialize $\mathbf{w}_{i,0} \leftarrow$ device i
  for each round $t = 1, 2, \ldots$ **do** ▷ Training loop
    receive$\{\mathbf{w}_{k,t}\}_{k\in\mathcal{N}_{i,t}}$ ▷ RX from broker
    $Dec\{\mathbf{w}_{k,t}\}_{k\in\mathcal{N}_{i,t}}$ ▷ Decipher weights
    $\nabla\mathcal{L}_k(\boldsymbol{\psi}_{i,t}) = -\frac{\Delta_k}{\eta} \quad \forall k \in \mathcal{S}_t$
    $\theta_{k_1,k_2,t} = \arccos \frac{\nabla\mathcal{L}_{k_1}(\boldsymbol{\psi}_{i,t})^{\mathrm{T}} \cdot \nabla\mathcal{L}_{k_2}(\boldsymbol{\psi}_{i,t})}{\|\nabla\mathcal{L}_{k_1}(\boldsymbol{\psi}_{i,t})\|\|\nabla\mathcal{L}_{k_2}(\boldsymbol{\psi}_{i,t})\|} \quad \forall k_1, k_2 \in \mathcal{S}_t, k_1 \neq k_2$
    $\widetilde{\theta}_{k_1,k_1,t} = \begin{cases} \theta_{k_1,k_2,t} & t = 1 \\ \frac{t-1}{t}\widetilde{\theta}_{k_1,k_2,t-1} + \frac{1}{t}\theta_{k_1,k_2,t} & t > 1 \end{cases} \quad \forall k_1, k_2 \in \mathcal{S}_t, k_1 \neq k_2$
    $f(\widetilde{\theta}_{k_1,k_2,t}) = \alpha_{\mathrm{G}}(1 - e^{-e^{-\alpha_{\mathrm{G}}(\widetilde{\theta}_{k_1,k_2,t}-1)}}) \quad \forall k_1, k_2 \in \mathcal{S}_t, k_1 \neq k_2$
    $k_t^* = \operatorname{argmax}_k \sum_{k'\in\mathcal{S}_t, k'\neq k} f(\widetilde{\theta}_{k,k',t})$
    $\widetilde{\alpha}_{k,k_t^*} = \frac{D_k e^{f(\widetilde{\theta}_{k,k_t^*})}}{\sum_{k'\in\mathcal{N}_{k_t^*}} D_{k'} e^{f(\widetilde{\theta}_{k',k_t^*})}} \quad \forall k \in \mathcal{N}_{k_t^*}$
    $\boldsymbol{\psi}_{i,t} = \mathbf{w}_{k_t^*} + \epsilon_t \sum_{k\in\mathcal{N}_{k_t^*}} \widetilde{\alpha}_{k,k_t^*} \left(\mathbf{w}_{k,t} - \mathbf{w}_{k_t^*}\right)$
    $\mathbf{w}_{i,t+1} = \boldsymbol{\psi}_{i,t} - \eta\nabla\mathcal{L}_i(\boldsymbol{\psi}_{i,t})$ ▷ Model update
    send $Enc(\mathbf{w}_{i,t+1})$ ▷ Encrypt and TX to broker
  end for
end procedure

with respect to the current update. Then, the reference neighbor model is selected as:

$$k_t^* = \operatorname*{argmax}_k \sum_{k'\in\mathcal{S}_t, k'\neq k} f(\widetilde{\theta}_{k,k',t}). \tag{13}$$

The reference neighbor in (13) is chosen by taking among the summations of the Gompertz functions since this helps in identifying the client whose local model is most similar to the majority of the neighboring models, thus maximizing the potential contribution to the consensus. The aggregation weights are computed excluding the reference client as:

$$\widetilde{\alpha}_{k,k_t^*} = \frac{D_k e^{f(\widetilde{\theta}_{k,k_t^*})}}{\sum_{k'\in\mathcal{N}_{k_t^*}} D_{k'} e^{f(\widetilde{\theta}_{k',k_t^*})}} \quad \forall k \in \mathcal{N}_{k_t^*}. \tag{14}$$

As opposed to (10), the aggregated model is now obtained using the selected client k_t^* as reference and implementing an exponential moving average:

$$\boldsymbol{\psi}_{i,t} = \mathbf{w}_{k_t^*} + \epsilon_t \sum_{k\in\mathcal{N}_{k_t^*}} \widetilde{\alpha}_{k,k_t^*} \left(\mathbf{w}_{k,t} - \mathbf{w}_{k_t^*}\right), \tag{15}$$

where ϵ_t modulates the memory of previous models. The benefit of this approach is that, in a fully-connected network, all clients will select the same reference client during each federated round, leading to a more stable and seamless convergence. This is especially advantageous when one client possesses a highly representative model, as it can serve as a reference to accelerate the convergence process. Notice that (15) can be rewritten as (10) as shown in Appendix A. Therefore, CFAdp-CS can be interpreted as a special case of CFAdp-vPS for which the same convergence properties as described in Sec. III-C hold.

As a last remark, note that through Algorithm 2 it is always possible to force each client i to choose its own local model i as the reference, so that $k_t^* = i, \forall t$, and regardless of the reliability of the local data and update. This particular *degenerate* case, referred to as CFAdp Egocentric (CFAdp-Ego), will further analyzed in Sect. IV. In this case, the model aggregation (15) reduces to:

$$\boldsymbol{\psi}_{i,t} = \mathbf{w}_{i,t} + \epsilon_t \sum_{k \in \mathcal{N}_{i,t}} \widetilde{\alpha}_{k,i} \left(\mathbf{w}_{k,t} - \mathbf{w}_{i,t} \right). \tag{16}$$

In the egocentric approach, each client uses its own local model as the reference. In other words, the instantaneous angles in (12) are computed between the local gradients $\nabla \mathcal{L}_i$ and the neighbors' gradients $\nabla \mathcal{L}_k$. If the local model is highly biased, such as in the scenario depicted in Figure 6 with 1 IID client and 9 non-IID clients, relying on a biased local reference can hinder optimal aggregation.

The three proposed strategies, namely CFAdp-vPS, CFAdp-CS and CFAdp-Ego, are visually represented in Fig. 2 to highlight their mutual differences. In Fig. 2(a), representing CFAdp-vPS, the reference clients (highlighted in red) are all the available ones in the subset $\mathcal{S}_t$, as described in (8). Therefore, each client acts as a virtual PS, which collects and updates the model independently from its index.

In Fig. 2(b), illustrating CFAdp-CS, the reference client changes at each round according to the model with the highest contribution, as determined by (13). Note that the selected client is the same for all clients in the network during that round, leading to a more stable and seamless convergence.

Finally, in Fig. 2(c), we show the CFAdp-Ego, where each client adopts its local model as the reference. This means that the aggregation is centered around each client's own model, and the similarities are computed between the local gradients and those of the neighbors. As previously discussed, if the local model is highly biased, its dependence can lead to suboptimal aggregation and affect the convergence of the algorithm.

C. Convergence Analysis

We now analyze the theoretical convergence of the proposed CFAdp algorithm by adopting typical FL assumptions [36], [41], [48], [50], [52].

Assumption 1. ***γ-Lipschitz smoothness.***
Considering a generic client i member of the federation, we let $\mathcal{L}_k(\boldsymbol{\psi}_{i,t})$ $\forall k \in \mathcal{S}_t$ be γ-Lipschitz smooth i.e., $\|\nabla \mathcal{L}_k(\boldsymbol{\psi}_{i,t}) - \nabla \mathcal{L}_k(\boldsymbol{\psi}_{i,t+q})\| \leq \gamma \|\boldsymbol{\psi}_{i,t} - \boldsymbol{\psi}_{i,t+q}\|$ for any two parameter vectors $\boldsymbol{\psi}_{i,t}, \boldsymbol{\psi}_{i,t+q}$.

Based on Assumption 1, the local representation of the global objective, defined as $\nabla \mathcal{L}(\boldsymbol{\psi}_{i,t}) = \sum_{k \in \mathcal{S}_t} \alpha_{k,\mathcal{S}_t} \nabla \mathcal{L}_k(\boldsymbol{\psi}_{i,t})$, can also be assumed as γ-Lipschitz smooth since $\sum_{k \in \mathcal{S}_t} \alpha_{k,\mathcal{S}_t} = 1$ for each subset $\mathcal{S}_t$.

Assumption 2. ***Bounded Local Dissimilarity.***
For any client i, the dissimilarity between local loss of client k and the local representation of the global objective at $\boldsymbol{\psi}_{i,t}$ is bounded by A and B, i.e., $A\|\nabla \mathcal{L}(\boldsymbol{\psi}_{i,t})\| \leq \|\nabla \mathcal{L}_k(\boldsymbol{\psi}_{i,t})\| \leq B\|\nabla \mathcal{L}(\boldsymbol{\psi}_{i,t})\|$.

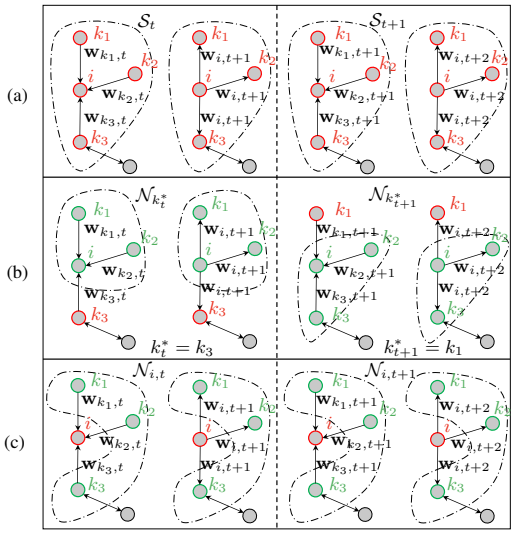


Fig. 2. Three Consensus-driven Federated Adaptive Weighting (CFAdp) strategies, where reference models and neighbors are highlighted in red and green, respectively. From top to bottom: CFAdp with virtual PS (CFAdp-vPS), with client selection (CFAdp-CS) and egocentric (CFAdp-Ego).

Notice that when all the local data samples are the same, it is $A = B = 1$, therefore the local dissimilarity $|A - B|$ in Assumption 2 might be an indicator of the data heterogeneity among clients, under the assumption of same training configuration.

Assumption 3. ***Stationarity.***
For any client i, we assume that the subsets $\mathcal{S}_t$ and $\mathcal{N}_{i,t}$ $\forall i \in \mathcal{S}_t$ are stationary over time.

In the context of PS-based FL, the same assumptions have been made in several works [17], [36], [48], [52], [76]–[80] to ensure stability and successful convergence of the learning process across distributed datasets. Smoothness [76]–[78] is a safeguard to ensure that the learning process is stable, meaning small changes in the model parameters don't result in large variations in learning loss, while the bounded dissimilarity [17], [79] ensures that learning can be effectively coordinated across different clients to converge towards a useful model that generalizes across all participating nodes.

Finally, by assuming stationary neighbors (as done in [80]), the client selection policy remains consistent over time, which is crucial for establishing the convergence guarantees of the WAC strategy. This practical assumption acknowledges that not all nodes participate in every round due to factors like network connectivity issues or device availability, but maintains that the selection of participating clients does not change drastically over time, thus providing a stable and predictable learning environment. However, the algorithm remains operational even in non-stationary environments. In scenarios where the network topology changes, the algorithm can still work effectively if the client selection policy adheres

to a consistent random and uniform mechanism. This ensures ergodicity, meaning every client has a fair chance of being selected over time, regardless of dynamic network changes.

It is important to note that the consensus-based FL inherently complements the mesh network structure, where each node not only captures and disseminates its own data, but also serves as a relay for other nodes. In the case of a mesh network, even if the mobility of the clients is partially present, the consensus-based FL model would still work effectively, assuming that the set of neighboring clients remains stationary. This essentially implies that even if a subset of clients changes its position or connection, as long as the overall structure of the neighboring set of clients remains consistent over time, the learning process can proceed uninterrupted.

Theorem 1. *With loss function $\mathcal{L}_k(\boldsymbol{\psi}_{i,t})$ satisfying Assumptions 1-2-3 and supposing $\boldsymbol{\psi}_{i,t}$ is not a stationary solution, the expected decrease in the global loss function on client i-th and between two consecutive consensus rounds satisfies,*

$$\begin{aligned}\mathcal{L}(\boldsymbol{\psi}_{i,t+1}) &\leq \mathcal{L}(\boldsymbol{\psi}_{i,t}) \\ &- \eta\mathbb{E}_{k\in\mathcal{S}_t}\Bigg[\Bigg(\frac{\nabla\mathcal{L}(\boldsymbol{\psi}_{i,t})^{\mathrm{T}}\cdot\nabla\mathcal{L}_k(\boldsymbol{\psi}_{i,t})}{\|\nabla\mathcal{L}(\boldsymbol{\psi}_{i,t})\|\|\nabla\mathcal{L}_k(\boldsymbol{\psi}_{i,t})\|} \\ &- \frac{B\gamma\eta}{2}\Bigg)\frac{A^2}{2}\|\nabla\mathcal{L}(\boldsymbol{\psi}_{i,t})\|^2\Bigg], \end{aligned} \tag{17}$$

where the expectation $\mathbb{E}_{k|t}$ refers to the weighting strategy of the client $k\in\mathcal{S}_t$ for global model aggregation.

The proof of Theorem 1 builds upon [50] while to guarantee paper self-consistency, it is discussed in Appendix B. Theorem 1 provides a bound on how rapid the decrease of the FL loss can be expected on the generic client i. It is straightforward to verify that the convergence upper bound of decentralized FL tool after N_{FL} consensus rounds is given by

$$\begin{aligned}\mathcal{L}(\boldsymbol{\psi}_{i,N_{\mathrm{FL}}}) &\leq \mathcal{L}(\boldsymbol{\psi}_{i,1}) \\ &- \eta\sum_{t=1}^{N_{\mathrm{FL}}}\mathbb{E}_{k\in\mathcal{S}_t}\Bigg[\Bigg(\frac{\nabla\mathcal{L}(\boldsymbol{\psi}_{i,t})^{\mathrm{T}}\cdot\nabla\mathcal{L}_k(\boldsymbol{\psi}_{i,t})}{\|\nabla\mathcal{L}(\boldsymbol{\psi}_{i,t})\|\|\nabla\mathcal{L}_k(\boldsymbol{\psi}_{i,t})\|} \\ &- \frac{B\gamma\eta}{2}\Bigg)\frac{A^2}{B}\|\nabla\mathcal{L}(\boldsymbol{\psi}_{i,t})\|^2\Bigg]. \end{aligned} \tag{18}$$

Based on Theorem 1, we have the following remarks.

Remark 1. *The decrease of FL loss on the client i and between two consecutive learning rounds shows the same dependencies as in the FedAdp algorithm [50] including the bound gap $[A, B]$. The correlation $\frac{\nabla\mathcal{L}(\boldsymbol{\psi}_{i,t})^{\mathrm{T}}\cdot\nabla\mathcal{L}_k(\boldsymbol{\psi}_{i,t})}{\|\nabla\mathcal{L}(\boldsymbol{\psi}_{i,t})\|\|\nabla\mathcal{L}_k(\boldsymbol{\psi}_{i,t})\|}$ between the local gradient and the representation of the global gradient, obtained from the received neighbor models, is a local metric to measure their alignment level.*

Remark 2. *Similarly as for PS-based FedAdp, increasing $\mathbb{E}_{k\in\mathcal{S}_t}[\cdot]$ in each global round improves the convergence of decentralized FL. Contributions of each individual neighbor can be measured quantitatively through the correlation $\frac{\nabla\mathcal{L}(\boldsymbol{\psi}_{i,t})^{\mathrm{T}}\cdot\nabla\mathcal{L}_k(\boldsymbol{\psi}_{i,t})}{\|\nabla\mathcal{L}(\boldsymbol{\psi}_{i,t})\|\|\nabla\mathcal{L}_k(\boldsymbol{\psi}_{i,t})\|}$ between the local gradient $\nabla\mathcal{L}_k(\boldsymbol{\psi}_{i,t})$ and the local representation of the global gradient $\nabla\mathcal{L}(\boldsymbol{\psi}_{i,t})$ obtained from the neighbors, and assign larger weights to the nodes with higher contribution to enlarge the expected decrease of FL loss in each global round.*

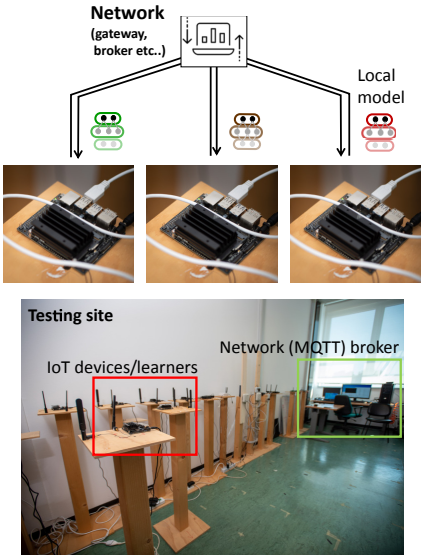


Fig. 3. Real FL-platform composed of IoT devices, i.e., *Jetson Nano*, connected via WLAN to a workstation.

IV. Simulation Experiments

In this section, we first describe the real-world FL network platform, and then clarify the main networking characteristics and tools that underpin the CFAdp-CS, CFAdp-vPS and CFAdp-Ego consensus processes. Finally, we present the results on convergence properties and performances with highly skewed non-IID data distributions.

A. FL Networking Characteristics and Platform

In order to validate the three proposed CFAdp strategies, we adopted both simulated and real fully-distributed clients connected via Wireless Local Area Network (WLAN) and communicating, i.e., exchanging neural network model parameters, through the broker-based Message Queuing Telemetry Transport (MQTT) protocol [81]. The simulated network of clients was implemented in a workstation featuring an Intel(R) Xeon(R) Silver 4210R CPU operating at 2.40 GHz, 96 GB of RAM, and a Quadro RTX 6000 24 GB GPU. This allowed us to have more flexibility in defining the computational capabilities and the number of clients. On the contrary, the real-FL platform prototype was composed of 6 *Jetson Nano* devices [82] equipped with CPU ARM-Cortex-A57 and GPU 128-core Maxwell. The laboratory comprising the IoT devices and the workstation is shown in Fig. 3.

Communications among the clients in the consensus scheme are managed by an MQTT broker, which receives and

forwards model updates. Specifically, each client subscribes to the topics related to the model parameters of its neighbors. Upon completing a local training round, a client pushes its updated local model to its designated topic, e.g., `/fl_session_ID/client_ID`, and pulls the updated models from its neighbors. This pulling operation can be performed asynchronously and automatically, allowing the clients to continuously listen for new updates on their neighbors' topics. The MQTT protocol ensures reliable communication by handling possible packet losses and retransmissions, guaranteeing exactly-once packet delivery through Quality-of-Service (QoS) level 2. Notice that, as observed during the experimental tests, a high QoS level might introduce transmission delays. The development of strategies to mitigate them has not been considered in this paper, as our primary objective is to design a WAC learning strategy tailored for non-IID local data. Despite the observed delays in the tests, the use of MQTT transport combined with the proposed WAC strategies demonstrated robustness and resilience. For more insights into the potential effects of transmission delays on FL optimization procedures in the presence of a PS, we refer to our previous work [40].

B. FL Dataset and Implementation

Regarding the datasets, we employed a simple dataset widely used for classification tasks, i.e., the Modified National Institute of Standards and Technology (MNIST) [83] dataset using the full validation data, and a more complex scenario with Canadian Institute For Advanced Research (CIFAR)-100 dataset [84]. The corresponding adopted DL models are a Convolutional Neural Network (CNN) (LeNet architecture [85]) and a Convolutional Vision Transformers (CVT) model [86]. For training procedures, we considered the Adam optimization algorithm [72] with an initial learning rate of 0.0001 and momentum values of $m_1 = 0.9$ and $m_2 = 0.999$. To choose the hyper-parameter α_{G}, we tested the values in the range $[1, 10]$ and we reported the results in Fig. 4. Note that increasing α_{G} improved the accuracy up to a certain point, beyond which the performance gains saturated. We found that $\alpha_{\mathrm{G}} = 4$ provided a good trade-off between sensitivity to the smoothed angles and preserving the distinguishability of contributions from nodes with smaller angle differences. Therefore, for the experiments, we assigned α_{G} equal to 4. The consensus step-size ϵ_t was set to 0.3. Finally, we adopted a number of local epochs $E = 1$ and a maximum number of federated rounds $N_{\mathrm{FL}} = 100$.

In order to accurately regulate the degree of non-IIDness between clients, we considered two cases. First case adopts a Dirichlet distribution to assign the number of local samples, namely the *quantity skew*. For the second case, the Dirichlet density is used to regulate the label distributions, or *label skew* [41], [87], [88]. In particular, for non-IID sample distributions, the same percentage of labels is retained on each client, while the number of samples in client i is $D_i = p_i^{(\mathrm{S})} D$, where D is the total number of training samples and $\boldsymbol{p}^{(\mathrm{S})} = [p_i^{(\mathrm{S})}]_{i=1}^{K} \sim Dir(\boldsymbol{\beta}^{(\mathrm{S})})$ are the random samples of a Dirichlet distribution with concentration parameters $\boldsymbol{\beta}^{(\mathrm{S})}$. Here, for simplicity, we consider $\boldsymbol{\beta}^{(\mathrm{S})} = [\beta_i^{(\mathrm{S})}]_{i=1}^{K} = [\beta^{(\mathrm{S})}]_{i=1}^{K}$. On the contrary, for non-IID label distributions, the same quantity of samples is kept on each client, while the distribution of labels across clients follows the Dirichlet distribution. In particular, for a client i, the proportion of label ℓ in the local dataset is $p_{i\ell}^{(\mathrm{L})}$, where $\boldsymbol{p}_i^{(\mathrm{L})} = [p_{i\ell}^{(\mathrm{L})}]_{\ell=0}^{9} \sim Dir(\boldsymbol{\beta}^{(\mathrm{L})})$ and $\boldsymbol{\beta}^{(\mathrm{L})} = [\beta_\ell^{(\mathrm{L})}]_{\ell=0}^{9} = [\beta^{(\mathrm{L})}]_{\ell=0}^{9}$.

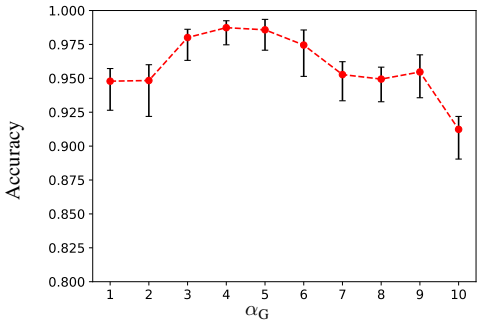
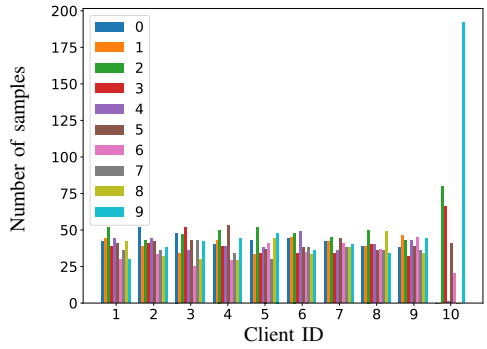


Fig. 4. Tuning of the Gompertz function hyper-parameter α_{G}.

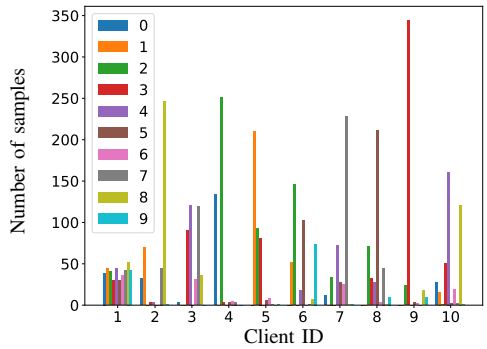
C. Convergence Analysis

In this first assessment, we aim at verifying the convergence capabilities of the proposed algorithms within a specific client, e.g., $i = 10$, whose local data distribution is much different from those of its neighbors. This is done in order to compare the different methods (i.e., CFAdp methods and non-adaptive baseline strategy CFA) in the worst-case scenario where non-IID clients usually struggle to converge. To this aim, we simulated a network of $K = 10$ clients in two scenarios: (a) 9 clients out of 10 hold a uniform distribution of labels, while a single client holds a non-IID label distributions with $\beta^{(\mathrm{L})} = 0.2$, and (b) only 1 client out of 10 holds a IID label distribution. For an example showing how labels are assigned, we refer to Fig. 5, where we represent the histogram of the labels for each client. Note that the client $i = 10$ under consideration has a very imbalanced distribution of labels, e.g., in Fig. 5(a) digits 7 and 8 are missing.

In Fig. 6, we show the average validation accuracy computed by $i = 10$ for each federated round and varying the consensus algorithm, including both scenarios (a) and (b). This is done to establish lower and upper bounds for each consensus algorithm's performance in the presence of non-IID local data. Further intermediate non-IID cases are tested in Sec. IV-D. The confidence bounds are obtained using the standard deviation as uncertainties. Additionally, we have included the centralized (i.e., PS-based) FedAdp algorithm in our comparisons as an upper bound to the consensus CFAdp versions. This inclusion allows us to benchmark the performance of our proposed algorithms against the best possible scenario in a centralized setting. A common behaviour in the two scenarios is that with CFA, due to the highly imbalanced distribution, the client struggles to converge and presents drops of performances



(a) 9 IID - 1 non-IID scenario



(b) 1 IID - 9 non-IID scenario

Fig. 5. Examples of label distributions in a federation of 10 clients. (a) Clients 1-9 retain a uniform distribution of labels (i.e., colors 0-9), while client 10 holds an imbalanced distribution. (b) Only client 1 has IID local distribution.

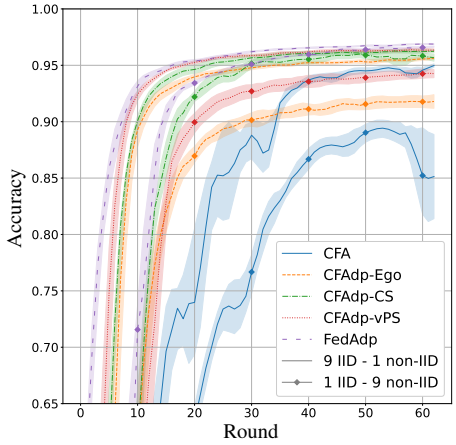


Fig. 6. Validation accuracy in a non-IID client varying the number of training FL rounds, for different consensus algorithms (i.e., baseline CFA and proposed CFAdp-Ego, CFAdp-CS and CFAdp-vPS). Two scenarios are represented: (a) 9 IID - 1 non-IID, (b) 1 IID - 9 non-IID clients.

caused by local overfitting. On the contrary, adopting a FL WAC strategy as CFAdp-Ego, we notice a much higher speed of convergence and stability on the training. However, given the fact that the reference local model is highly biased towards the skewed distribution, the performances are inferior to the CFAdp-CS which automatically selects the best neighbors' model as reference. Focusing on scenario (b), with CFAdp-CS we also avoid using the unbalanced models as references and outperform even the CFAdp-vPS. On the contrary, whenever the vast majority of clients have IID distributions as in scenario (a), it is more convenient to adopt the CFAdp-vPS since it represents the equivalent PS-based version.

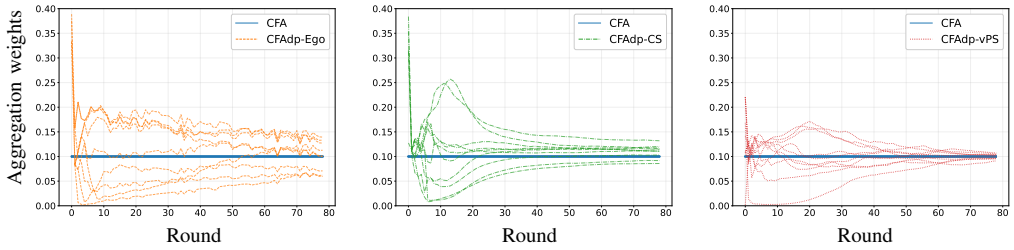
To further analyze the convergence of the proposed algorithms, in Fig. 7, we represent the aggregation weights in client $i = 10$ and scenario (a) at different training epochs for every neighbor k, i.e., $\alpha_{k,\mathcal{S}_t}$, $\widetilde{\alpha}_{k,i}$, $\widetilde{\alpha}_{k,k_t^*}$ and $\widetilde{\alpha}_{k,\mathcal{S}_t}$ for CFA, CFAdp-Ego, CFAdp-CS and CFAdp-vPS, respectively. Notice that the baseline strategy maintains all the weights $\alpha_{k,\mathcal{S}_t} = \frac{D_k}{\sum_{k' \in \mathcal{S}_t} D_{k'}} = 1/K = 0.1$ since all clients retain the same number of samples. On the contrary, the WAC strategies modulate the weights in order to compensate the unbalance between label distributions. Here the consensus step-size ϵ_t is fixed to 0.3 for all epochs in order not to alter the convergence of the aggregation weights. Between the WAC strategies, we can see that the CFAdp-Ego presents periodical changes of weights values due to tendency on overfitting. Moreover, the convergence times are much slower if compared with CFAdp-CS and CFAdp-vPS. Indeed, selecting the best neighbor dramatically smooths the convergence process and ultimately leads to a faster approximation towards the CFAdp-vPS solution.

D. Quantity and Label Skweness

In this section we evaluate two different datasets, i.e., MNIST and CIFAR100, using two DL models, i.e., CNN and CVT, respectively, tested on both simulated and real devices.

In this first experiment, we assess the performances of the proposed methods when the number of samples in each client varies significantly according to a Dirichlet distribution described in Sec. IV-B. This is important in order to measure the capabilities of coping with important or negligible local updates, as well as with generalized or overfitted models. To this aim, we simulated three different FL scenarios with $K = \{3, 10, 30\}$ clients and we plot in Fig. 8 the mean and standard deviations of the reached validation accuracy after 20 rounds, varying $\beta^{(\mathrm{S})} \in [0.01, 1]$.

From the results, we note that the observed tendency in the previous experiments is found again among all $\beta^{(\mathrm{S})}$ values, i.e., the simple weight aggregation strategy of CFA, based solely on the number of samples, struggles with very imbalanced distributions. Starting with the three-client scenario, we notice that if the distributions are the same among all clients,



(a) CFA vs CFAdp-Ego (b) CFA vs CFAdp-CS (c) CFA vs CFAdp-vPS

Fig. 7. Average aggregation weights after 5 different runs of CFA, CFAdp-Ego (a), CFAdp-CS (b) and CFAdp-vPS (c), at varying training federated rounds. We refer to (16), (14) and (11) for the aggregation weights $\widetilde{\alpha}_{k,i}$, $\widetilde{\alpha}_{k,k_t^*}$ and $\widetilde{\alpha}_{k,\mathcal{S}_t}$ of CFAdp-Ego, CFAdp-CS and CFAdp-vPS, respectively. The baseline non-adaptive CFA has all the weights $\alpha_{k,\mathcal{S}_t} = 1/K = 0.1$.

i.e., high $\beta^{(\mathrm{S})}$, the proposed CFAdp strategies reach the same level of accuracy. This is intuitive since whenever all models bring the same contributions, there is not advantage in choosing a specific one. Still, the WAC, with adaptive aggregation weights, outperforms the conventional CFA from 7 to 56%. With a higher number of clients, the convergence time increases and the differences between the proposed WAC strategies become more distinct.

In the last assessment, we employ the real FL-prototype composed of 6 IoT devices where each client experiences non-IID label distribution with $\beta^{(\mathrm{L})}$ varying in $[0.05, 100]$. Different from before, here we employ the much more complex CVT model and CIFAR100 dataset. In Fig. 9, we report the validation accuracy reached after $N_{\mathrm{FL}} = 100$ federated round, for each consensus algorithm. Note that with different $\beta^{(\mathrm{L})}$ and bigger models, the CFAdp-CS and CFAdp-vPS have almost the same performances. This is due to the fact that an increase in the number of model parameters, coupled with the use of a larger number of classes (100), worsens the overfitting of local models. Ultimately, it results in an equivalent choice between using all clients or the best-one as a reference. On the contrary, CFAdp-Ego, reduces the maximum achievable performance since it always relies on the local model which used as a reference for all the rounds. Finally, the CFA struggles to achieve 80% of accuracy.

V. Conclusion

This paper addressed the challenges associated with non-IID data distribution in fully-distributed, server-less networked learning systems by introducing a new family of algorithms with roots in WAC tools and adapted for FL processes, namely Consensus-driven FedAdp (CFAdp). Evolved from WAC schemes, the proposed tools have been optimized and adapted specifically for FL, each one employing a unique strategy for calculating the global model. Specifically, we developed three CFAdp algorithms, named CFAdp-Ego, CFAdp-CS, and CFAdp-vPS, where the reference local model is within the client itself, within the best selected neighbor and across all clients, respectively. They are then evaluated in terms of their convergence properties and resilience against non-IID data distribution. The evaluation included both simulated and real experiments using a FL platform implemented over a WLAN network, testing with varying model complexities, i.e., CNN and CVT, and datasets, i.e., MNIST and CIFAR100. Specifically, the tests aimed to simulate complexity and data heterogeneity typically encountered in IoT deployments, including various degrees of sample and label skewness modelled with Dirichlet distributions.

The derived key takeaways are the following. The weighted consensus schemes, i.e., CFAdp, outperform the vanilla tools, such as CFA, up to 56% thanks to the dynamic adjustment of the aggregation weights, disregarding negative client contributions. Whenever the federation of clients has IID or non-IID local distributions, meaning that each client has the same level of non-IID (quantity or label skewness), the CFAdp-vPS, which represents the equivalent PS-based version, achieves the best performances in terms of convergence time and asymptotic accuracy. Conversely, whenever one or few clients retain very high-quality and evenly distributed data, the CFAdp-CS permits to take advantage of the good local model contribution by taking the best client as a global reference.

References

[1] P. Vepakomma *et al.*, "No peek: A survey of private distributed deep learning," Dec. 2018, arXiv:1812.03288.

[2] J. Konečný *et al.*, "Federated optimization: Distributed optimization beyond the datacenter," Nov. 2015, arXiv:1511.03575.

[3] H. B. McMahan *et al.*, "Communication-efficient learning of deep networks from decentralized data," Jan. 2016, arXiv:1602.05629.

[4] J. Konečný *et al.*, "Federated optimization: Distributed machine learning for on-device intelligence," Oct. 2016, arXiv:1610.02527.

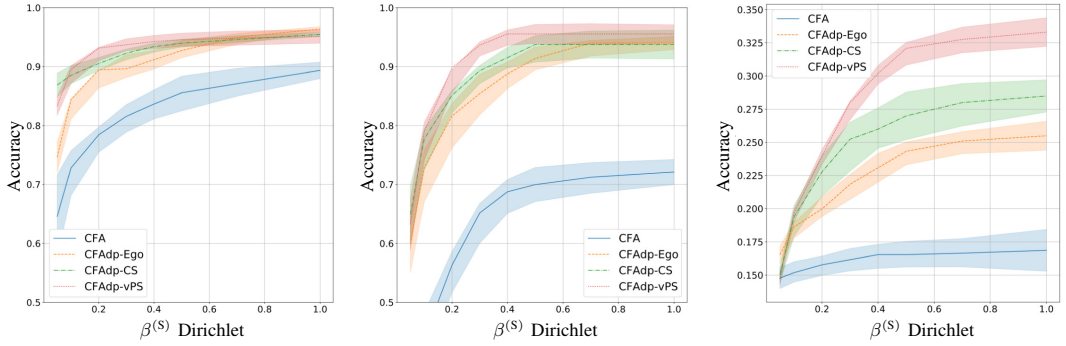
[5] B. Camajori Tedeschini *et al.*, "Split consensus federated learning: an approach for distributed training and inference," *IEEE Access*, vol. 12, pp. 119 535–119 549, Aug. 2024.

[6] L. Italiano *et al.*, "A tutorial on 5G positioning," *IEEE Commun. Surveys & Tuts.*, pp. 1–1, Aug. 2024.

[7] S. Niknam *et al.*, "Federated learning for wireless communications: Motivation, opportunities, and challenges," *IEEE Commun. Mag.*, vol. 58, no. 6, pp. 46–51, Jun. 2020.

[8] B. Camajori Tedeschini *et al.*, "Real-time Bayesian neural networks for 6G cooperative positioning and tracking," *IEEE J. Sel. Areas Commun.*, vol. 42, no. 9, pp. 2322–2338, Sep. 2024.

[9] L. Pu *et al.*, "Cost-efficient and skew-aware data scheduling for incremental learning in 5G networks," *IEEE J. Sel. Areas Commun.*, vol. 40, no. 2, pp. 578–595, Feb. 2022.



(a) $K = 3$ clients scenario (b) $K = 10$ clients scenario (c) $K = 30$ clients scenario

Fig. 8. Mean reached validation accuracy after 20 rounds of training for varying concentration parameters $\beta^{(S)}$ on the number of samples. The confidence bounds are obtained using the standard deviation as uncertainties.

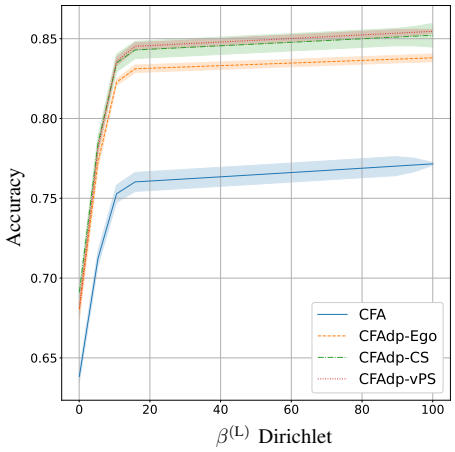


Fig. 9. Mean reached validation accuracy after 100 rounds of training for varying concentration parameters $\beta^{(L)}$ on the label skewness. The confidence bounds are obtained using the standard deviation as uncertainties.

[10] H. Gu *et al.*, "Fedaux: An efficient framework for hybrid federated learning," in *ICC 2022 - IEEE Int. Conf. Commun.* Seoul, Korea, Republic of: IEEE, May 2022, pp. 195–200.

[11] B. Camajori Tedeschini *et al.*, "Decentralized federated learning for healthcare networks: A case study on tumor segmentation," *IEEE Access*, vol. 10, pp. 8693–8708, 2022.

[12] U. Milasheuski *et al.*, "On the impact of data heterogeneity in federated learning environments with application to healthcare networks," in *IEEE Conf. Artif. Intell.* IEEE, Jun. 2024, pp. 1017–1023.

[13] P. Kairouz *et al.*, "Advances and open problems in federated learning," *Found. Trends® Mach. Learn.*, vol. 14, no. 1-2, pp. 1–210, 2021.

[14] S. Niknam *et al.*, "Federated learning for wireless communications: Motivation, opportunities, and challenges," *IEEE Commun. Mag.*, vol. 58, no. 6, pp. 46–51, Jun. 2020.

[15] H. H. Yang *et al.*, "Scheduling policies for federated learning in wireless networks," *IEEE Trans. Commun.*, vol. 68, no. 1, pp. 317–333, Jan. 2020.

[16] J. Perazzone *et al.*, "Communication-efficient device scheduling for federated learning using stochastic optimization," in *IEEE INFOCOM 2022 - IEEE Conf. Comput. Commun.* London, United Kingdom: IEEE, May 2022, pp. 1449–1458.

[17] H. Wu *et al.*, "Node selection toward faster convergence for federated learning on non-IID data," *IEEE Trans. Netw. Sci. Eng.*, vol. 9, no. 5, pp. 3099–3111, Sep. 2022.

[18] W. Zhang *et al.*, "Client selection for federated learning with non-IID data in mobile edge computing," *IEEE Access*, vol. 9, pp. 24 462–24 474, Feb. 2021.

[19] W. Guo *et al.*, "Joint device selection and power control for wireless federated learning," *IEEE J. Sel. Areas Commun.*, vol. 40, no. 8, pp. 2395–2410, Aug. 2022.

[20] T. Nishio *et al.*, "Client selection for federated learning with heterogeneous resources in mobile edge," in *ICC 2019 - 2019 IEEE Int. Conf. Commun.* Shanghai, China: IEEE, May 2019, pp. 1–7.

[21] J. Xu *et al.*, "Bandwidth allocation for multiple federated learning services in wireless edge networks," *IEEE Trans. Wireless Commun.*, vol. 21, no. 4, pp. 2534–2546, Apr. 2022.

[22] B. Luo *et al.*, "Cost-effective federated learning design," in *IEEE INFOCOM 2021 - IEEE Conf. Comput. Commun.* Vancouver, BC, Canada: IEEE, May 2021, pp. 1–10.

[23] B. Luo *et al.*, "Cost-effective federated learning in mobile edge networks," *IEEE J. Sel. Areas Commun.*, vol. 39, no. 12, pp. 3606–3621, Dec. 2021.

[24] N. H. Tran *et al.*, "Federated learning over wireless networks: Optimization model design and analysis," in *IEEE INFOCOM 2019 - IEEE Conf. Comput. Commun.* IEEE, Apr. 2019, pp. 1387–1395.

[25] X. Ling *et al.*, "Time is gold: A time-dependent incentive mechanism design for fast federated learning," in *2022 IEEE/CIC Int. Conf. Commun. China*. IEEE, Aug. 2022, pp. 1038–1043.

[26] K. Yang *et al.*, "Federated learning via over-the-air computation," *IEEE Trans. Wireless Commun.*, vol. 19, no. 3, pp. 2022–2035, Mar. 2020.

[27] M. Mohammadi Amiri *et al.*, "Machine learning at the wireless edge: Distributed stochastic gradient descent over-the-air," *IEEE Trans. Signal Process.*, vol. 68, pp. 2155–2169, Mar. 2020.

[28] M. M. Amiri *et al.*, "Federated learning over wireless fading channels," *IEEE Trans. Wireless Commun.*, vol. 19, no. 5, pp. 3546–3557, May 2020.

[29] Z. Gao *et al.*, "Fedim: An anti-attack federated learning based on agent importance aggregation," in *2021 IEEE 20th Int. Conf. Trust, Security Privacy Comput. Commun.* IEEE, Oct. 2021, pp. 1445–1451.

[30] A. Bhowmick *et al.*, "Protection against reconstruction and its applications in private federated learning," Jun. 2019, arXiv:1812.00984.

[31] Y. Jiang *et al.*, "Improving federated learning personalization via model agnostic meta learning," Jan. 2023, arXiv:1909.12488.

[32] V. C. Gogineni *et al.*, "Communication-efficient online federated learning framework for nonlinear regression," in *ICASSP 2022 - 2022 IEEE Int. Conf. Acoust., Speech Signal Process. (ICASSP)*. Singapore, Singapore: IEEE, May 2022, pp. 5228–5232.

[33] Y. Chen *et al.*, "Asynchronous online federated learning for edge devices

with non-IID data," in *2020 IEEE Int. Conf. Big Data*, Dec. 2020, pp. 15–24.

[34] P. Han *et al.*, "Adaptive gradient sparsification for efficient federated learning: An online learning approach," in *2020 IEEE 40th Int. Conf. Distrib. Comput. Syst.*, Nov. 2020, pp. 300–310, iSSN: 2575-8411.

[35] C. T. Dinh *et al.*, "DONE: Distributed approximate newton-type method for federated edge learning," *IEEE Trans. Parallel Distrib. Syst.*, pp. 1–1, Jan. 2022.

[36] H. T. Nguyen *et al.*, "Fast-convergent federated learning," *IEEE J. Sel. Areas Commun.*, vol. 39, no. 1, pp. 201–218, Jan. 2021.

[37] C. Zhou *et al.*, "TEA-fed: time-efficient asynchronous federated learning for edge computing," in *Proc. 18th ACM Int. Conf. Comput. Frontiers*. Virtual Event Italy: ACM, May 2021, pp. 30–37.

[38] S. Chen *et al.*, "Heterogeneous semi-asynchronous federated learning in internet of things: A multi-armed bandit approach," *IEEE Trans. Emerg. Topics Comput. Intell.*, vol. 6, no. 5, pp. 1113–1124, Oct. 2022.

[39] Z. Wang *et al.*, "Asynchronous federated learning over wireless communication networks," *IEEE Trans. Wireless Commun.*, vol. 21, no. 9, pp. 6961–6978, Sep. 2022.

[40] B. Camajori Tedeschini *et al.*, "A traffic model based approach to parameter server design in federated learning processes," *IEEE Commun. Lett.*, vol. 27, no. 7, pp. 1774–1778, Jul. 2023.

[41] Y. Zhao *et al.*, "Federated learning with non-IID data," *ArXiv*, 2018.

[42] J. Zhang *et al.*, "Fedada: Fast-convergent adaptive federated learning in heterogeneous mobile edge computing environment," *World Wide Web*, vol. 25, no. 5, pp. 1971–1998, Sep. 2022.

[43] K. Tu *et al.*, "Adaptive federated learning via mean field approach," in *2022 IEEE Int. Conf. Internet Things (iThings) IEEE Green Comput. & Commun. (GreenCom) IEEE Cyber, Phys. & Social Comput. (CPSCom) IEEE Smart Data (SmartData) IEEE Congr. Cybermatics (Cybermatics)*. Espoo, Finland: IEEE, Aug. 2022, pp. 168–175.

[44] K. Mo *et al.*, "Two-dimensional learning rate decay: Towards accurate federated learning with non-IID data," in *2021 Int. Joint Conf. Neural Netw. (IJCNN)*. Shenzhen, China: IEEE, Jul. 2021, pp. 1–7.

[45] N. Mhaisen *et al.*, "Optimal user-edge assignment in hierarchical federated learning based on statistical properties and network topology constraints," *IEEE Trans. Netw. Sci. Eng.*, vol. 9, no. 1, pp. 55–66, Jan. 2022.

[46] P. Tian *et al.*, "WSCC: A weight-similarity-based client clustering approach for non-IID federated learning," *IEEE Internet Things J.*, vol. 9, no. 20, pp. 20 243–20 256, Oct. 2022.

[47] N. Yoshida *et al.*, "Hybrid-FL for wireless networks: Cooperative learning mechanism using non-IID data," in *ICC 2020 - 2020 IEEE Int. Conf. Commun.* Dublin, Ireland: IEEE, Jun. 2020, pp. 1–7.

[48] T. Li *et al.*, "Federated optimization in heterogeneous networks," Apr. 2020, arXiv:1812.06127.

[49] R. Pathak *et al.*, "Fedsplit: an algorithmic framework for fast federated optimization," in *Advances Neural Inform. Process. Syst.*, H. Larochelle *et al.*, Eds., vol. 33. Curran Associates, Inc., Mar. 2020, pp. 7057–7066.

[50] H. Wu *et al.*, "Fast-convergent federated learning with adaptive weighting," *IEEE Trans. Cogn. Commun. Netw.*, vol. 7, no. 4, pp. 1078–1088, Dec. 2021.

[51] S. Wang *et al.*, "When edge meets learning: Adaptive control for resource-constrained distributed machine learning," in *IEEE INFOCOM 2018 - IEEE Conf. Comput. Commun.* Honolulu, HI: IEEE, Apr. 2018, pp. 63–71.

[52] S. Wang *et al.*, "Adaptive federated learning in resource constrained edge computing systems," *IEEE J. Sel. Areas Commun.*, vol. 37, no. 6, pp. 1205–1221, Jun. 2019.

[53] S. Savazzi *et al.*, "Federated learning with cooperating devices: A consensus approach for massive iot networks," *IEEE Internet Things J.*, vol. 7, no. 5, pp. 4641–4654, May 2020.

[54] F. P.-C. Lin *et al.*, "Semi-decentralized federated learning with cooperative D2D local model aggregations," *IEEE J. Sel. Areas Commun.*, vol. 39, no. 12, pp. 3851–3869, Dec. 2021.

[55] S. Savazzi *et al.*, "Opportunities of federated learning in connected, cooperative, and automated industrial systems," *IEEE Commun. Mag.*, vol. 59, no. 2, pp. 16–21, Feb. 2021.

[56] G. Soatti *et al.*, "Distributed signal processing for dense 5G iot platforms: Networking, synchronization, interference detection and radio sensing," *Ad Hoc Netw.*, vol. 89, pp. 9–21, Jun. 2019.

[57] R. Olfati-Saber *et al.*, "Consensus and cooperation in networked multi-agent systems," *Proc. IEEE*, vol. 95, no. 1, pp. 215–233, Jan. 2007.

[58] Y. Lu *et al.*, "Differentially private asynchronous federated learning for mobile edge computing in urban informatics," *IEEE Trans. Ind. Inform.*, vol. 16, no. 3, pp. 2134–2143, Mar. 2020.

[59] G. Zhu *et al.*, "Broadband analog aggregation for low-latency federated edge learning," *IEEE Trans. Wireless Commun.*, vol. 19, no. 1, pp. 491–506, Jan. 2020.

[60] S. Savazzi *et al.*, "An energy and carbon footprint analysis of distributed and federated learning," *IEEE Trans. Green Commun. Netw.*, vol. 7, no. 1, pp. 248–264, Mar. 2023.

[61] I. Hegedűs *et al.*, "Gossip learning as a decentralized alternative to federated learning," in *Distrib. Appl. Interoperable Syst.*, J. Pereira *et al.*, Eds. Cham: Springer International Publishing, Jun. 2019, vol. 11534, pp. 74–90.

[62] S. Savazzi *et al.*, "Federated learning with mutually cooperating devices: A consensus approach towards server-less model optimization," in *ICASSP 2020 - 2020 IEEE Int. Conf. Acoust., Speech Signal Process. (ICASSP)*. Barcelona, Spain: IEEE, May 2020, pp. 3937–3941.

[63] J. Mills *et al.*, "Communication-efficient federated learning for wireless edge intelligence in iot," *IEEE Internet Things J.*, vol. 7, no. 7, pp. 5986–5994, Jul. 2020.

[64] L. Barbieri *et al.*, "A layer selection optimizer for communication-efficient decentralized federated deep learning," *IEEE Access*, vol. 11, pp. 22 155–22 173, Mar. 2023.

[65] Y. Chen *et al.*, "Communication-efficient federated deep learning with layerwise asynchronous model update and temporally weighted aggregation," *IEEE Trans. Neural Netw. Learn. Syst.*, vol. 31, no. 10, pp. 4229–4238, Oct. 2020.

[66] G. Soatti *et al.*, "Consensus-based algorithms for distributed network-state estimation and localization," *IEEE Trans. Signal Inform. Process. over Netw.*, vol. 3, no. 2, pp. 430–444, Jun. 2017.

[67] D. P. Spanos *et al.*, "Distributed sensor fusion using dynamic consensus," in *IFAC World Congr.* Citeseer, 2005, pp. 1–6.

[68] A. Giuseppi *et al.*, "A weighted average consensus approach for decentralized federated learning," *Mach. Intell. Res.*, vol. 19, no. 4, pp. 319–330, Jul. 2022.

[69] Z. Chen *et al.*, "DACFL: Dynamic average consensus-based federated learning in decentralized sensors network," *Sens.*, vol. 22, no. 9, p. 3317, Apr. 2022.

[70] S. Samarakoon *et al.*, "Distributed federated learning for ultra-reliable low-latency vehicular communications," *IEEE Trans. Commun.*, vol. 68, no. 2, pp. 1146–1159, Feb. 2020.

[71] S. Savazzi *et al.*, "A joint decentralized federated learning and communications framework for industrial networks," in *2020 IEEE 25th Int. Workshop Comput. Aided Modeling Des. Commun. Links Netw. (CAMAD)*. Pisa, Italy: IEEE, Sep. 2020, pp. 1–7.

[72] D. P. Kingma *et al.*, "Adam: A method for stochastic optimization," Jan. 2017, arXiv:1412.6980.

[73] O. Shamir *et al.*, "Communication efficient distributed optimization using an approximate newton-type method," *ArXiv*, May 2013.

[74] N. Shoham *et al.*, "Overcoming forgetting in federated learning on non-IID data," *ArXiv*, Oct. 2019.

[75] D. Mackay *et al.*, "Variational Gaussian process classifiers," *IEEE Trans. Neural Netw.*, vol. 11, no. 6, pp. 1458–1464, Nov. 2000.

[76] C. Chen *et al.*, "GIFT: Toward accurate and efficient federated learning with gradient-instructed frequency tuning," *IEEE J. Sel. Areas Commun.*, vol. 41, no. 4, pp. 902–914, Apr. 2023.

[77] S. Park *et al.*, "Regulated subspace projection based local model update compression for communication-efficient federated learning," *IEEE J. Sel. Areas Commun.*, vol. 41, no. 4, pp. 964–976, Apr. 2023.

[78] C.-H. Hu *et al.*, "Scheduling and aggregation design for asynchronous federated learning over wireless networks," *IEEE J. Sel. Areas Commun.*, vol. 41, no. 4, pp. 874–886, Apr. 2023.

[79] J. Jin *et al.*, "Accelerated federated learning with decoupled adaptive optimization," in *Proc. 39th Int. Conf. Mach. Learn.*, ser. Proceedings of Machine Learning Research, K. Chaudhuri *et al.*, Eds., vol. 162. PMLR, 17-23 Jul. 2022, pp. 10 298–10 322.

[80] T. Jahani-Nezhad *et al.*, "Swiftagg+: Achieving asymptotically optimal communication loads in secure aggregation for federated learning," *IEEE J. Sel. Areas Commun.*, vol. 41, no. 4, pp. 977–989, Apr. 2023.

[81] "Mqtt v3.1 protocol specification," https://mqtt.org.

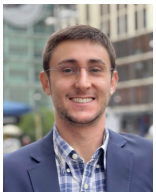
[82] "Jetson nano developer kit," https://tinyurl.com/52mdbjzh, accessed: 2021-03-01.

[83] Y. LeCun *et al.*, "MNIST handwritten digit database," *ATT Labs [Online]. Available: http://yann.lecun.com/exdb/mnist*, vol. 2, 2010.

[84] A. Krizhevsky *et al.*, "Learning multiple layers of features from tiny images," *Master's thesis, Department of Computer Science, University of Toronto*, 2009.

[85] Y. LeCun *et al.*, "Backpropagation Applied to Handwritten Zip Code Recognition," *Neural Comput.*, vol. 1, no. 4, pp. 541–551, Dec. 1989.

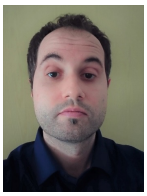
[86] H. Wu *et al.*, "Cvt: Introducing convolutions to vision transformers," in *2021 IEEE/CVF Int. Conf. Comput. Vis. (ICCV)*. Montreal, QC, Canada: IEEE, Oct. 2021, pp. 22–31.
[87] M. Luo *et al.*, "No Fear of Heterogeneity: Classifier Calibration for Federated Learning with Non-IID Data," in *Advances Neural Inform. Process. Syst.*, M. Ranzato *et al.*, Eds., vol. 34. Curran Associates, Inc., 2021, pp. 5972–5984.
[88] T. Lin *et al.*, "Ensemble Distillation for Robust Model Fusion in Federated Learning," in *Advances Neural Inform. Process. Syst.*, H. Larochelle *et al.*, Eds., vol. 33. Curran Associates, Inc., 2020, pp. 2351–2363.



Bernardo Camajori Tedeschini (Graduate Student Member, IEEE) received the B.Sc. (Hons.) in Computer Science and M.Sc. (Hons.) degrees in Telecommunications Engineering from the Politecnico di Milano, Italy, in 2019 and 2021, respectively. From November 2021 he started as PhD fellow in Information Technology at Dipartimento di Elettronica, Informazione e Bioingegneria (DEIB), Politecnico di Milano.

He is currently a visiting researcher with the Laboratory for Information & Decision Systems at the Massachusetts Institute of Technology (MIT), Cambridge, MA. His research interests include federated learning, machine learning and localization methods.

He was a recipient of the Ph.D. grant from the Ministry of the Italian government Ministero dell'Istruzione, dell'Università e della Ricerca (MUR) and the Roberto Rocca Doctoral Fellowship granted by MIT and Politecnico di Milano.



Stefano Savazzi (Member, IEEE) received the M.Sc. degree and the Ph.D. degree (Hons.) in ICT from the Politecnico di Milano, Italy, in 2004 and 2008, respectively. In 2012, he joined the Institute of Electronics, Computer and Telecommunication Engineering (IEIIT), Consiglio Nazionale delle Ricerche (CNR), as a Researcher, and as a Senior Researcher from 2023. He was a Visiting Researcher with Uppsala University, in 2005 and University of California at San Diego in 2007. He has coauthored over 110 scientific publications (Scopus). His current research interests include distributed signal processing, machine learning and networking aspects for the Internet of Things, radio localization and vision technologies. Dr. Savazzi won the Dimitris N. Chorafas Foundation Award in 2008. He is principal investigator for CNR in Horizon EU projects Holden and TRUSTroke. He is also serving as Associate Editor for Frontiers in Communications and Networks, Wireless Communications and Mobile Computing and Lead Guest Editor for the Special Issue on Radio Sensing and Sensor Networks in Sensors (MDPI).

Monica Nicoli (Senior Member, IEEE) received the M.Sc. (Hons.) and Ph.D. degrees in communication engineering from Politecnico di Milano, Milan, Italy, in 1998 and 2002, respectively. She was a Visiting Researcher with ENI Agip, from 1998 to 1999, and Uppsala University, in 2001. In 2002, she joined Politecnico di Milano as a Faculty Member. She is currently an Associate Professor in telecommunications with the Department of Management, Economics and Industrial Engineering. Her research interests include signal processing, machine learning, and wireless communications, with emphasis on smart mobility and Internet of Things (IoT) applications. She was a recipient of the Marisa Bellisario Award, in 1999, and a co-recipient of the best paper awards of the IEEE Symposium on Joint Communications and Sensing, in 2021, the IEEE Statistical Signal Processing Workshop, in 2018, and the IET Intelligent Transport Systems journal, in 2014. She is an Associate Editor of the IEEE Transactions on Intelligent Transportation Systems. She has also served as an Associate Editor for the EURASIP Journal on Wireless Communications and Networking, from 2010 to 2017, and a Lead Guest Editor for the Special Issue on Localization in Mobile Wireless and Sensor Networks, in 2011.

APPENDIX

A. CFAdp-CS vs CFAdp-vPS: Client Selection Process

In this section we discuss Algorithm 2 which implements CFAdp by selecting a client as opposed to Algorithm 1 that implements weighting average. Algorithm 2 averaging operation in (15) can be rewritten as follows:

$$\boldsymbol{\psi}_{i,t}=\left(1-\epsilon_t \sum_{k\in\mathcal{N}_{k_t^*}} \widetilde{\alpha}_{k,k_t^*}\right)\mathbf{w}_{k_t^*}+\sum_{k\in\mathcal{N}_{k_t^*}} \epsilon_t\, \widetilde{\alpha}_{k,k_t^*}\mathbf{w}_{k,t}, \tag{A1}$$

Compared with Algorithm 2 the consensus weights $\widetilde{\alpha}_{k,\mathcal{S}_t}$ are thus modified as follows:

$$\widetilde{\alpha}_{k,\mathcal{N}_{k_t^*}}=\begin{cases}\sum_{k\in\mathcal{N}_{k_t^*}}(1-\epsilon_t\,\widetilde{\alpha}_{k,k_t^*}) & k=k_t^*\\ \epsilon_t\,\widetilde{\alpha}_{k,k_t^*} & k\neq k_t^*.\end{cases} \tag{A2}$$

In Sect. IV we provide a numerical comparison between CFAdp weights $\widetilde{\alpha}_{k,\mathcal{S}_t}$ and CFAdp-CS ones $\widetilde{\alpha}_{k,k_t^*}$ as obtained during FL training.

B. CFAdp-vPS: Proof of Theorem 1

From the γ-Lipschitz smoothness of $\mathcal{L}(\boldsymbol{\psi}_{i,t})$ in Assumption 1 and Taylor expansion, we have:

$$\begin{aligned}\mathcal{L}(\boldsymbol{\psi}_{i,t+1}) \leq& \mathcal{L}(\boldsymbol{\psi}_{i,t}) + \nabla\mathcal{L}(\boldsymbol{\psi}_{i,t})^{\mathrm{T}}\cdot(\boldsymbol{\psi}_{i,t+1}-\boldsymbol{\psi}_{i,t})\\ &+\frac{\gamma}{2}\|\boldsymbol{\psi}_{i,t+1}-\boldsymbol{\psi}_{i,t}\|^2.\end{aligned} \tag{A3}$$

The last two terms on the right-hand side of the above inequality are bounded respectively as:

- *Bounding* $\|\boldsymbol{\psi}_{i,t+1}-\boldsymbol{\psi}_{i,t}\|^2$: By the definition of the global aggregation for $\boldsymbol{\psi}_{i,t+1}$, we have:
$$\|\boldsymbol{\psi}_{i,t+1}-\boldsymbol{\psi}_{i,t}\| = \mathbb{E}_{k\in\mathcal{S}_t}[\|\mathbf{w}_{i,t+1}-\boldsymbol{\psi}_{i,t}\|]. \tag{A4}$$
By following SGD optimization, for each term within the expectation in the right hand side of A4, we have:
$$\mathbf{w}_{i,t+1}=\boldsymbol{\psi}_{i,t}-\eta\nabla\mathcal{L}_k(\boldsymbol{\psi}_{i,t}). \tag{A5}$$
Therefore,
$$\begin{aligned}\|\boldsymbol{\psi}_{i,t+1}-\boldsymbol{\psi}_{i,t}\|^2 &= (\mathbb{E}_{k\in\mathcal{S}_t}[\|\mathbf{w}_{i,t+1}-\boldsymbol{\psi}_{i,t}\|])^2\\ &=\eta^2(\mathbb{E}_{k\in\mathcal{S}_t}[\|\nabla\mathcal{L}_k(\boldsymbol{\psi}_{i,t})\|])^2\\ &\leq \eta^2\mathbb{E}_{k\in\mathcal{S}_t}[\|\nabla\mathcal{L}_k(\boldsymbol{\psi}_{i,t})\|^2],\end{aligned} \tag{A6}$$
where inequality holds because of Cauchy-Schwarz inequality.
- *Bounding* $\nabla\mathcal{L}(\boldsymbol{\psi}_{i,t})^{\mathrm{T}}\cdot(\boldsymbol{\psi}_{i,t+1}-\boldsymbol{\psi}_{i,t})$: Again, by the definition of the global aggregation for $\boldsymbol{\psi}_{i,t+1}$ and A5 we have:
$$\begin{aligned}&\nabla\mathcal{L}(\boldsymbol{\psi}_{i,t})^{\mathrm{T}}\cdot(\boldsymbol{\psi}_{i,t+1}-\boldsymbol{\psi}_{i,t}) =\\ &=-\eta\mathbb{E}_{k\in\mathcal{S}_t}[\nabla\mathcal{L}(\boldsymbol{\psi}_{i,t})^{\mathrm{T}}\cdot\nabla\mathcal{L}_k(\boldsymbol{\psi}_{i,t})].\end{aligned} \tag{A7}$$
The expectation term in A7 can be further rewritten as shown in [50] with the following substitutions: $\mathbf{w}_{\mathrm{PS},t}$ becomes $\boldsymbol{\psi}_{i,t}$ and the expectation is defined over all clients $k\in\mathcal{S}_t$.

IEEE Access
Multidisciplinary ⋮ Rapid Review ⋮ Open Access Journal

Received 7 June 2024, accepted 11 August 2024, date of publication 20 August 2024, date of current version 4 September 2024.

Digital Object Identifier 10.1109/ACCESS.2024.3446577

RESEARCH ARTICLE

Split Consensus Federated Learning: An Approach for Distributed Training and Inference

BERNARDO CAMAJORI TEDESCHINI[1], (Graduate Student Member, IEEE), MATTIA BRAMBILLA[1], (Member, IEEE), AND MONICA NICOLI[2], (Senior Member, IEEE)

[1]Dipartimento di Elettronica, Informazione e Bioingegneria, Politecnico di Milano, 20133 Milan, Italy
[2]Department of Management, Economics and Industrial Engineering, Politecnico di Milano, 20156 Milan, Italy

Corresponding author: Bernardo Camajori Tedeschini (bernardo.camajori@polimi.it)

ABSTRACT Distributed Machine Learning (D-ML), such as Federated Learning (FL) and Split Learning (SL), aims at resolving the limitations of Centralized Machine Learning (C-ML) by enhancing scalability and efficiency. D-ML relies on the client-Parameter Server (PS) paradigm, in which clients collaboratively train ML models while keeping their data locally, reducing the need for central data storage and preserving data privacy. In this paper, we propose a new fully-distributed method, named Split Consensus Federated Learning (SCFL), which combines the characteristics of FL and SL into a network of clients that cooperate in learning a shared model. Inspired by the iterative approach of Message Passing Neural Networks (MPNN), the proposed SCFL framework allows to decentralize the training and inference tasks of the neural networks at the clients, preserving the privacy of locally stored data. The proposed SCFL framework removes the need for a coordinating central entity, i.e., the PS, resulting into a fully-decentralized solution where both the training and inference procedures are distributed over the clients. We present three different strategies for SCFL implementation and we validate them in a cooperative positioning use case where clients use D-ML for network localization. Results show that the proposed SCFL method is able to combine the computational power (and data) of all clients to train local models which closely approximate the global C-ML solution at convergence.

INDEX TERMS Split consensus federated learning, split learning, federated learning, message passing neural network, consensus, cooperative positioning.

I. INTRODUCTION

In recent years, Machine Learning (ML) has contaminated several fields, including healthcare [1], [2], finance [3], [4], and transportation [5], [6]. Most of the applications rely on a Centralized Machine Learning (C-ML) architecture where data collection and processing from multiple clients is performed at a single central server, which raises concerns about data privacy and security. Furthermore, with the rapid increase in volume and complexity of data available at different locations and machines, C-ML may face difficulties in terms of scalability and efficiency. Consequently, there is a growing demand for alternative methodologies that can efficiently address the complexity and security issues while retaining the benefits of conventional ML techniques.

Distributed Machine Learning (D-ML) [7], [8] has emerged as a viable solution for avoiding data aggregation at a single central entity. A popular D-ML mechanism is Federated Learning (FL) [9], [10], which allows spatially distributed clients to collaboratively train a global ML model without the need of sharing their raw data. In FL, each client keeps a local copy of the model, e.g., a Deep Learning (DL) model, and trains it using local data, while a coordinating Parameter Server (PS) aggregates the locally

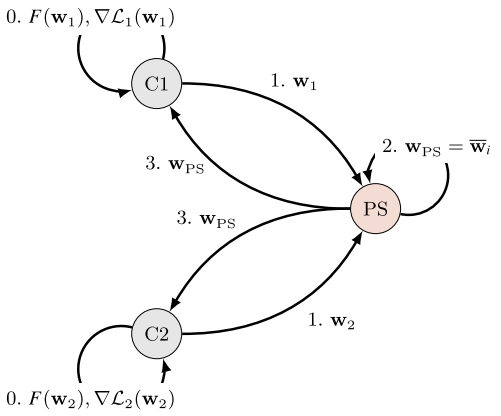
The associate editor coordinating the review of this manuscript and approving it for publication was Jolanta Mizera-Pietraszko.

trained models to produce a global model shared among all clients. This approach not only preserves data privacy by keeping data within local boundaries but also enhances computational efficiency and scalability in various distributed environments [11], [12], [13], [14], [15], [16]. As a drawback, the PS-based structure still relies on a central coordinating entity for the aggregation of models, potentially leading to a single point of failure and increased communication overhead.

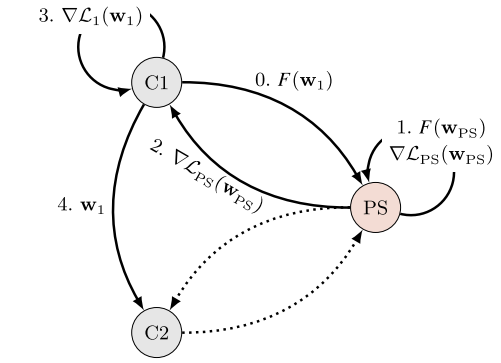
An alternative to FL is Split Learning (SL) [17], a D-ML approach that has been recently designed to address the challenges of resource-constrained setups, such as Internet of Things (IoT) networks where clients may have limited computing capabilities and energy resources. In SL, the model training and validation processes are divided between the clients and a PS, each having only a partial access/visibility to/of a specific portion of the model [18]. This characteristic ensures both model and data privacy while improving communication efficiency and convergence speed compared to FL [19]. Specifically, the Neural Network (NN) to be trained is split into two sub-networks at a specific layer, named split or cut layer, and the upper and lower layers are assigned to the clients and the PS, respectively. Clients perform forward propagation and send the output, called smashed data, to the PS, which computes the final output. Gradients are then back-propagated, with the PS sending the cut layer's gradient back to the clients. This process is repeated until new training data is obtained.

A conceptual comparison of the two D-ML methods is reported in Figure 1, in which we present the FL (Figure 1a) and SL (Figure 1b) frameworks, indicating their main steps indexed in chronological order. Specifically, the steps for FL are: 0) local model optimization, 1) aggregation, 2) broadcast of the updated global model; while the steps for SL are: 0) client forward pass and exchange of smashed data, 1) PS forward pass and back-propagation, 2) exchange of PS gradients, 3) client back-propagation, 4) client model exchange with next neighbors.

Although SL has received significant research attention, a number of open problems are still to be addressed, such as leakage reduction [20], [21], [22], [23], [24], non-Independent and Identical Distributed (IID) data distribution among clients [25], [26], [27], and communication costs. Tackling these challenges is crucial for unlocking the full potential of SL for D-ML applications.



(a) Federated Learning



(b) Split Learning

FIGURE 1. Schematic example of (a) FL and (b) SL in a network of two clients and a PS. The client model parameters and gradients are indicated with $\mathbf{w}_i$ and $\nabla\mathcal{L}_i(\mathbf{w}_i)$, respectively. On the contrary, the global or PS model parameters and gradients are indicated with $\mathbf{w}_{\mathrm{PS}}$ and $\nabla\mathcal{L}_{\mathrm{PS}}(\mathbf{w}_{\mathrm{PS}})$, respectively. The weighted average of the FL is indicated with a thick bar. Finally, the forward pass is indicated with $F(\cdot)$, back-propagation is indicated with the model gradients inside a self-loop, and dashed lines represent the next timestamp.

A. RELATED WORKS

In this section, we discuss how the limitations of FL and SL have been addressed so far in the literature and our proposal to solve remaining open problems.

Vanilla FL architectures present two main drawbacks related to centralization at the PS and limited computational capabilities of clients. Decentralization has been proposed by replacing the PS with a consensus procedure that performs model fusion through iterative inter-client exchanges of model parameters. First works in this direction are represented by fully-distributed gossip FL [28] and Consensus-driven Federated Averaging (CFA) [29], [30]. In gossip FL, local updates are propagated in a peer-to-peer manner where each client shares its own local model update to the immediate neighbors. CFA extends gossip approaches to include average consensus by exploiting all or a subset of neighbors at each round. Regarding FL with resource-constrained devices, state-of-the-art approaches mainly focus on optimized versions of SL that split the computations between clients and PS, usually located in the cloud [31], [32], [33]. However, SL does not exploit parallelization

of training and validation procedures and still relies on a centralized architecture with a PS.

More specifically, in SL, clients interact with the PS sequentially, causing other clients' resources to remain idle during the relay-based training process. This results in increased training overhead and latency, especially when a large number of devices are involved in the learning process. To solve this issue, some authors proposed to impose differences in the training order and adjust the data size inside the nodes [34]. A full parallelization has been introduced with the Split Federated Learning (SFL) framework by the pioneer works in [35], [36], and [37]. SFL integrates the primary benefit of FL, i.e., parallel processing among distributed clients, with the core advantage of SL, i.e., partitioning the network into client-side and server-side sub-networks throughout training. Unlike SL, SFL enables all clients to carry out computations concurrently while engaging with both a split PS and a federated PS. Contrary to SL, SFL allows all clients to interact with the federated PS and the split PS simultaneously while doing calculations.

A further problem of SL relies on the mandatory usage of a PS whose main operation is to distribute the computational complexity of model training and inference. However, in massive IoT networks, a PS may not be available or may be prohibitive from a communication point of view. A first step in this direction is taken by the works [38], [39] which introduce a split version of Recurrent Neural Networks (RNNs) with a continuous exchange of smashed data among clients. However, these architectures still rely on a PS, thus lacking from a full decentralization.

B. ADDRESSING FL AND SL OPEN PROBLEMS

Unavailability of a fully-distributed methodology for both training and inference is a major issue for adoption of FL and SL in resource-constrained networks of devices, and it is thus the topic we aim to address in this work.

For the design of a completely decentralized architecture, we proposed the following. We first observe that in SL, the key aspect is that clients alone are not able to perform a complete inference (and back-propagation) of the whole model, as the model is split between clients and PS to lower the computational complexity of each node. Therefore, for distributed SL, we conceive to train an overall model whose intermediate outputs, i.e., smashed data, are computed and exchanged between clients. A DL framework which satisfies the aforementioned characteristics can be found in Graph Neural Network (GNN) [40], more specifically in the variant of Message Passing Neural Network (MPNN) [41]. Indeed, in MPNN, the final inference is the result of sequential intermediate outputs, obtained with a message passing procedure [42], [43]. However, in vanilla MPNN, both the training and inference procedures are centralized since the exchange of node and edge embeddings happens in the same physical machine. Moreover, the built computational graph permits to back-propagate gradients in a unique and parallel way such that at the end each nodes will have the same NN parameters. In this paper, we propose to exploit the MPNN methodology to fill the literature gap and design a new fully decentralized SL architecture for distributed inference and training, as outlined in the following section.

C. CONTRIBUTION

In this work, we propose to incorporate the message passing scheme inside the MPNN as a sequence of smashed data exchanged among clients, i.e., the nodes of the graph. According to this scheme, at each timestep the clients do not complete the inference of the whole model, but they just perform one iteration of the message passing at a time. Thus, the complete model is composed of many small models retained by individual clients. The information available at each client does not explicitly describe the available data, as it is a hidden representation incorporated by the so-called node and edge embeddings of the MPNN. This aspect ensures data privacy, just as in vanilla SL, since each client holds private labels or outputs. To accelerate the entire training process, similarly to SFL, a consensus scheme is executed after the message passing iterations. Depending on the number of operations within the message passing iteration and by the type of consensus scheme, i.e., full model exchange or gradient average, we can distinguish between three main training procedures, namely, 3-Steps Strategy (3SS), 2-Steps Strategy (2SS) and Distributed-MPNN (D-MPNN). We refer to this new fully-distributed framework as Split Consensus Federated Learning (SCFL).

To summarize, the main contributions of the paper are the following:

- A comprehensive review and comparative analysis of FL, SL and SFL, with focus on the iterative processing steps for training;
- The design of a fully-decentralized SFL architecture, namely SCFL, which exploits the above analysis and extends the centralized MPNN to perform distributed training and inference procedures between physically separated clients;
- The proposal and validation of three decentralized training procedures for SCFL which can be effectively adopted in fully-distributed agent networks.

Compared to FL, SL, and SFL, the key distinctive advantages of the proposed SCFL method are the following:

- The introduction of a SL paradigm within the FL framework, enabling complex DL models to be trained on distributed resource-constrained devices through an efficient message-passing mechanism inspired by MPNN. This approach reduces the model bias and enhances the model's ability to accurately capture underlying data patterns.
- The scalability with respect to the number of nodes, allowing SCFL to be trained and scaled over complex network topologies without procedural alterations. This property is crucial for applications in dynamic network environments with large number of connected devices.

TABLE 1. Comparison of SL, FL, SFL and the proposed SCFL method.

	SL	FL	SFL	SCFL (proposed)
Privacy-preserving	Yes	Yes	Yes	Yes
Fully-distributed	No	Yes	No	Yes
Parallel train/test	No	Yes	Yes	Yes
Low complexity	Yes	No	Yes	Yes

- The unique capability of training DL models on physically separated clients, where the inference of each client is dependent on its neighbors. This feature is particularly beneficial for tasks requiring collaborative information sharing, such as cooperative positioning in networks of agents.
- The preservation of privacy, the fully-decentralized architecture, and the low complexity training, which are fundamental for deployment in privacy-sensitive, resource-constrained environments.

A summary of the main differences between the proposed SCFL method and SL, FL and SFL is provided in Table 1.

The proposed SCFL solution is suitable for operating conditions where distributed cooperative training is the only viable option, and it is here validated for the illustrative use case of Cooperative Positioning (CP) in agent networks [44], [45], [46], [47], [48]. Examples of possible domains of application are within the fields of vehicular networks [49], [50], [51], [52], IoT [53], [54], maritime surveillance [55], [56] and drones [57], [58].

D. PAPER ORGANIZATION

This paper is organized as follows. Section II presents the distributed machine learning context, comprised of FL and SL. Section III first introduces the proposed SCFL framework and its relationship with the MPNNs, and then it presents the distributed training and inference strategies, as well as the innovation aspects of SCFL that allow to overcome the limitations of current FL architectures. Section IV discusses the CP use case of SCFL and its main experimental results. Lastly, Section V draws the conclusions.

II. FUNDAMENTALS OF DISTRIBUTED ML

This section is designed to provide the reader the basic understandings of D-ML, concentrating on architectural and algorithmic aspects. The contents of this section are indeed functional to contextualize and introduce the core principles of the proposed SCFL method, described later in Section III, as well as to highlight the differences with respect to our solution. To this extent, we first describe the framework of FL, specifically focusing on consensus-based algorithms. Then, we describe the vanilla SL framework and the related parallelized SFL version with PS. The focus on such state-of-the-art notions is required since the proposed SCFL paradigm inherits both the FL and SL features and extends them to accommodate for a fully-distributed scheme.

A. FEDERATED LEARNING

In the context of FL, we consider a network which includes one PS and a set of I clients denoted as $\mathcal{I} = \{1, \ldots, I\}$. Each client has its own dataset $\mathcal{S}_i$ of size $S_i = |\mathcal{S}_i|$. The objective of the FL procedure is to obtain a global DL model defined by the parameters $\mathbf{w}$ by minimizing:

$$\mathbf{w}_{\text{PS}} = \underset{\mathbf{w}}{\operatorname{argmin}}\, \mathcal{L}(\mathbf{w}), \tag{1}$$

where the loss function $\mathcal{L}(\mathbf{w})$ is given by:

$$\mathcal{L}(\mathbf{w}) = \frac{1}{I}\sum_{i=1}^{I} \mathcal{L}_i(\mathbf{w}, \mathcal{S}_i), \tag{2}$$

with $\mathcal{L}_i$ representing the loss determined by client i utilizing the local data batches $\mathcal{S}_i$. To obtain the global model $\mathbf{w}_{\text{PS}}$, an iterative process is carried out with each iteration involving a local model optimization step performed by the client and followed by an aggregation step executed on the PS.

Clients generate local models typically employing supervised and gradient-based optimization techniques, e.g., Stochastic Gradient Descent (SGD) or Adam optimizers [59], with mini-batch $\mathcal{B}$ of size B and learning rate η. Each client i carries out E local epochs prior to exchanging the local model with the PS, which is responsible for updating the global model.

In vanilla FL, i.e., Federated Averaging (FedAvg), the aggregation step during federated round $n = 1, \ldots, N$ is conducted at the PS using a weighted average accounting for the number of samples S_i from each client as:

$$\mathbf{w}_{\text{PS},n+1} = \frac{\epsilon}{\sum_{j=1}^{I} S_j}\sum_{i=1}^{I} S_i\, \mathbf{w}_{i,n} + (1-\epsilon)\, \mathbf{w}_{\text{PS},n}, \tag{3}$$

where $\mathbf{w}_{i,n}$ are the local model parameters and ϵ modulates the memory of previous models.

On the contrary, when no PS is available, the consensus-based FL regime applies. In this framework, a network of clients constitutes a graph $\mathcal{G} = (\mathcal{V}, \mathcal{E})$, where each node $i \in \mathcal{V}$ represents a client, while the edge $(i, j) \in \mathcal{E}$, with $i \neq j$, indicates the presence of a communication link from client i to client j. Observe that edges (i, j) and (j, i) are distinct, i.e., $(i, j) \neq (j, i)$, and they may not necessarily exist concurrently. From the graph $\mathcal{G}$, we define the set of neighbors of client i as $\mathcal{N}_i = \{j \in \mathcal{V} | (i, j) \in \mathcal{E}\}$ and $\mathcal{N}_{i^*} = \mathcal{N}_i \cup \{i\}$.

The consensus-based algorithm, called CFA, works in the following way. At round n, each client i, after performing a local model optimization step, exchanges the model parameters $\mathbf{w}_{i,n}$ with its neighbors $\mathcal{N}_i$ and subsequently performs an aggregation step, similarly to the PS [29], as:

$$\boldsymbol{\psi}_{i,n} = \mathbf{w}_{i,n} + \frac{\epsilon}{\sum_{j\in\mathcal{N}_i} S_j}\sum_{j'\in\mathcal{N}_i} S_{j'}\left(\mathbf{w}_{j',n} - \mathbf{w}_{i,n}\right), \tag{4}$$

where $\boldsymbol{\psi}_{i,n}$ is the aggregated model. This aggregation step is needed to let the local model converge to a consensus global

Algorithm 1 Consensus-Driven Federated Averaging

procedure CFA($\mathcal{N}_i, \epsilon, \eta$) ▷ Run on client i
  initialize $\mathbf{w}_{i,0} \leftarrow$ client i
  for each round $n = 1, \ldots, N$ **do** ▷ Training loop
    broadcast $\mathbf{w}_{i,n}$
    receive $\{\mathbf{w}_{j,n}\}_{j \in \mathcal{N}_i}$
    eq. (4)
    $\mathbf{w}_{i,n} = \text{ModelUpdate}(\boldsymbol{\psi}_{i,n})$
  end for
end procedure
procedure ModelUpdate($\boldsymbol{\psi}_{i,n}$) ▷ Model opt. step
  compute $F(\boldsymbol{\psi}_{i,n})$ ▷ Forward-pass
  compute $\nabla \mathcal{L}_{i,n}(\boldsymbol{\psi}_{i,n})$ ▷ Backward-pass
  $\boldsymbol{\psi}_{i,n} \leftarrow \boldsymbol{\psi}_{i,n} - \eta \nabla \mathcal{L}_{i,n}(\boldsymbol{\psi}_{i,n})$ ▷ Local SGD
end procedure

model. The complete pseudo-code for CFA algorithm is reported in Algorithm 1, where the local model optimization step has been represented as a single-batch model update. For simplicity of notation, we do not consider n-dependence on hyper-parameters and graph structure. Note that the algorithm works in the same way if the clients send the gradients of the local model update, instead of exchanging the model parameters.

B. SPLIT LEARNING

In the simplest SL framework, the model parameters $\mathbf{w}$ are split into two parts (one for the PS and for the clients), i.e., $\mathbf{w}_{\text{PS},n}$ and $\mathbf{w}_{i,n}$. Note that, here, differently from FL, the model structures of $\mathbf{w}_{\text{PS},n}$ and $\mathbf{w}_{i,n}$ are distinct. Thus, to complete inference and back-propagation procedures, an exchange of smashed data and gradients must be carried out between PS and clients. The pseudo-code for SL algorithm is given in Algorithm 2, reporting for simplicity only the synchronization of the learning process in peer-to-peer mode [17]. At the end of the training, SL permits to achieve identical results to a traditional (i.e., centralized) training procedure, where all layers are available at the same entity, since it involves the same steps and processes (forward propagation and back-propagating gradients), just applied in a different order.

A drawback of the SL procedure is that it must be executed sequentially by each client. To address this issue, SFL algorithms remove the constraint on the sequentiality of inter-client model exchange, performing forward propagation of the client-side model in parallel. The PS processes the forward propagation and back-propagation on its server-side model using each client's transformed data separately, allowing a high degree of parallelism. After sending the gradients back to the respective clients for their own back-propagation, a step of FedAvg is performed by the PS and by the clients through an additional PS for the federated part, i.e., FPS [35]. We refer to Figure 2 for a representation of the SFL workflow.

Algorithm 2 Split Learning

procedure SL(η)
  initialize $\mathbf{w}_{i,0} \quad \forall\, i \in \mathcal{I}$
  for each round $n = 1, \ldots, N$ **do** ▷ Training loop
    for client $i = 1, \ldots, I$ **do** ▷ Run on client i
      compute $F(\mathbf{w}_{i,n})$ ▷ Client Forward-pass
      send $F(\mathbf{w}_{i,n})$ to PS
      receive
        $\nabla \mathcal{L}_{\text{PS},n}(\mathbf{w}_{\text{PS},n}) = \text{PSUpdate}(F(\mathbf{w}_{i,n}))$
      $\mathbf{w}_{i,n} = \text{ClientUpdate}(\mathbf{w}_{i,n})$
      send $\mathbf{w}_{i,n}$ to client $i+1$
    end for
  end for
end procedure
procedure ClientUpdate($\mathbf{w}_{i,n}$) ▷ Model opt. step
  compute $\nabla \mathcal{L}_{i,n}(\mathbf{w}_{i,n})$ ▷ Client Backward-pass
  $\mathbf{w}_{i,n} \leftarrow \mathbf{w}_{i,n} - \eta \nabla \mathcal{L}_{i,n}(\mathbf{w}_{i,n})$ ▷ Local SGD
end procedure
procedure PSUpdate($F(\mathbf{w}_{i,n})$) ▷ Model opt. step
  compute $F(\mathbf{w}_{\text{PS},n})$ ▷ PS Forward-pass
  compute $\nabla \mathcal{L}_{\text{PS},n}(\mathbf{w}_{\text{PS},n})$ ▷ PS Backward-pass
  $\mathbf{w}_{\text{PS},n} \leftarrow \mathbf{w}_{\text{PS},n} - \eta \nabla \mathcal{L}_{\text{PS},n}(\mathbf{w}_{\text{PS},n})$ ▷ Local SGD
end procedure

III. DESIGN OF SPLIT CONSENSUS FEDERATED LEARNING

In D-ML frameworks where the PS is absent, the distribution among clients of training and inference tasks becomes challenging, especially for training. Indeed, the absence of a PS prevents the direct coordination and consolidation of local models into a global version. We here overcome such limitations by proposing the SCFL approach.

SCFL differs from vanilla FL as it does not require a PS coordinating the clients and aggregating the local models for the convergence to a global version (exactly as in distributed consensus-FL). The final goal of SCFL is still to achieve the same global model in each client at the end of training, but it is achieved with direct Device-to-Device (D2D) communications. The fundamental difference from consensus-FL is that, in SCFL, the clients need to perform in a distributed way both training and inference procedure. Indeed, they cannot complete a whole inference autonomously for model complexity reasons, as in PS-based SL. On the other hand, SCFL differs from PS-based SL as it does not exist a PS-version of the model structure, since each client is fundamentally equivalent to the others.

The main assumptions that we make for the design of SCFL are the following:

A1) As in SL, FL and SFL, each client has the same model structure;
A2) As in SL, FL and SFL, each client needs the forwarding procedure result, i.e., smashed data, of its neighbors to complete the inference. Thus, it results that both

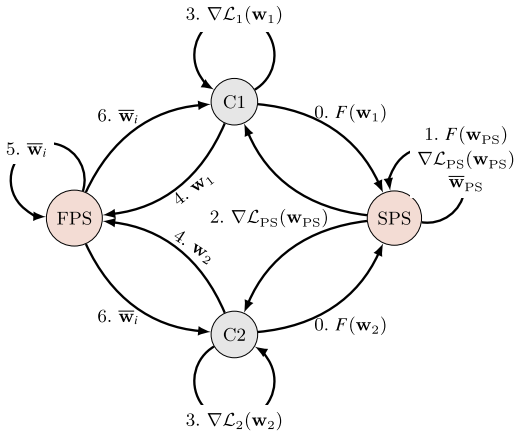


FIGURE 2. SFL framework with vanilla architecture composed of a split PS, i.e., SPS, and a federated PS, i.e., FPS. For ease of notation, step 2) is represented in the same way for both the clients.

training and prediction have to be performed in a fully-distributed way;

A3) Each client has the ability of exchanging different types of messages that span from direct model parameters or gradients, up to smashed data. In any case, the body of the messages must not disclose any private information about the local retained data inside the clients.

These assumptions do not limit the usage or applicability of SCFL, they rather give a performance advantage in cooperative contexts, e.g., CP, where the exchange of smashed data dramatically improves the performances. This claim is analyzed in Section IV-C3, where we compare the proposed solution with the CFA approach.

The above assumptions highlight a similarity with the vanilla MPNNs approach, where nodes are represented by distributed clients. Indeed, the inference message passing procedure of MPNNs can be seen as multiple forwards passes between clients which exchange smashed data, i.e., intermediate outputs of a bigger model. In the same manner, back-propagation is computed by taking into account all the predictions during message passing. However, while in vanilla MPNNs the forward and backward passes are computed within the same computational graph (i.e., centralized procedure), for performing distributed operations, especially training, we need to carefully design the strategy to follow. We thereby propose to exploit this synergy by first revising the centralized MPNN (Section III-A) and then designing an extension to a distributed framework by incorporating the MPNN into the proposed SCFL approach. This allows to combine the benefits of both SL and FL. The distributed features of the proposed SCFL solution are detailed for both inference (Section III-B) and training (Section III-C) procedures.

A. REVIEW OF CENTRALIZED MPNN

NNs operating on graphs have been investigated only in the recent years, initially as GNNs [40], [41], and subsequently expanded to include variations such as MPNNs [60]. Their goal is to train, in a centralized way, a function that disseminates information throughout a graph $\mathcal{G}$. The information is diffused by message passing using node and edge latent features, called embeddings, and denoted as $\mathbf{v}_{i,n}^{(t)}$ and $\mathbf{e}_{j\to i,n}^{(t)}$, respectively, where t is the message passing iteration index. For the encoding of the embeddings, a NN is placed at each node and edge of the graph. The NN at the node is denoted by $g_v(\cdot)$, while the one at the edge is represented by $g_e(\cdot)$. Then, according to the specific task, e.g., regression or classification, an additional *global* NN is present.

Let us consider the node regression task with a specific NN at the node $g_v^{(\text{regres})}(\cdot)$. Given that $g_v(\cdot)$, $g_v^{(\text{regres})}(\cdot)$ and $g_e(\cdot)$ maintain the same parameters, across each node and each edge respectively, they can be centrally trained on small-scale graphs before being utilized in large-scale problems. The final node prediction is performed independently by each node after T message passing iterations in which node and edge embeddings are updated through the NN models according to the specific message passing structure.

We here recall the centralized vanilla MPNN inference and training procedure. The inference procedure starts with the node and edge embedding initialization, i.e., $\mathbf{v}_{i,n}^{(0)}, \quad \forall j \in \mathcal{V}$, and $\mathbf{e}_{j\to i,n}^{(0)}, \quad \forall j \in \mathcal{N}_i$, through a feature extraction mechanism, e.g., an encoding NN. Then, at message passing iteration $t = 1, \ldots, T$, each node $i \in \mathcal{V}$ sends the following message to its neighbors $\mathcal{N}_i$:

$$\mathbf{e}_{j\to i,n}^{(t)} = g_e\left(\mathbf{v}_{i,n}^{(t-1)}, \mathbf{v}_{j,n}^{(t-1)}, \mathbf{e}_{j\to i,n}^{(t-1)}\right), \quad \forall j \in \mathcal{N}_i, \tag{5}$$

with

$$\mathbf{v}_{i,n}^{(t)} = g_v\left(\mathbf{v}_{i,n}^{(t-1)}, \Phi\left(\{\mathbf{e}_{j\to i,n}^{(t)}\}_{j\in\mathcal{N}_i}\right)\right), \tag{6}$$

where $\Phi(\cdot)$ is called aggregation function and it can be chosen among whatever function which is invariant to permutations of its inputs, e.g., element-wise summation. After T message passing iterations, the prediction is performed as:

$$\widehat{\mathbf{y}}_{i,n} = \widehat{\mathbf{y}}_{i,n}^{(T)} = g_v^{(\text{regres})}\left(\mathbf{v}_{i,n}^{(T)}\right), \tag{7}$$

where $\widehat{\mathbf{y}}_{i,n}$ is the estimate of the true target variable $\mathbf{y}_{i,n}$. Since the exchange of messages is centralized, the inference and forward-pass procedure can be performed in parallel considering all the edge and node embeddings as samples of a batch that is given as input to the edge and node NNs, respectively. Hence, what is carried out is the training, in this case, of only three distinct and individual NNs that will subsequently be distributed across various network topologies. In order to compute the training loss and perform back-propagation, the Residual Sum of Squares (RSS) estimated at each epoch n and at the end of each message passing iteration t after the regressor prediction $\widehat{\mathbf{y}}_{i,n}^{(t)}$

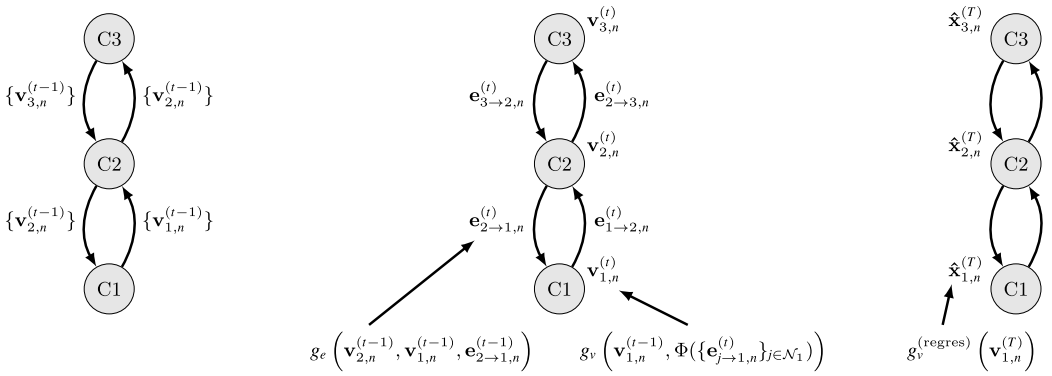


FIGURE 3. Distribute inference procedure. a) node embedding exchange. b) edge and node embedding update. c) prediction.

is considered. It is defined as:

$$\mathcal{L} = \frac{1}{N\,|\mathcal{V}|} \sum_{n=1}^{N} \sum_{t=1}^{T} \sum_{i\,\in\,\mathcal{V}} \left\| \widehat{\mathbf{y}}_{i,n}^{(t)} - \mathbf{y}_{i,n} \right\|_2^2 . \tag{8}$$

In centralized MPNN, a key (limiting) aspect is that each node cannot proceed with the next message passing without the output of its neighbors at previous message passing iteration. In the proposed SCFL framework, we overcome such limitation by considering each node as an independent and physically separated client that needs the neighbors smashed data, i.e., node embeddings, to proceed with the inference and, ultimately, perform prediction. The proposal is detailed in the next section.

B. DESIGN OF DISTRIBUTED INFERENCE IN SCFL

In the proposed SCFL framework, each physically separated client has to carefully choose the type of information to be exchanged with its neighbors, avoiding unfeasible communication costs. Therefore, we design an equivalent distributed inference procedure where clients exchange node embeddings $\mathbf{v}_{i,n}^{(t)}$, i.e., smashed data, to their neighbors. Here, as in CFA, the set of neighbors is built starting from the communication links between agents, which may vary according to the environment. However, in this work, we focus on the proposal of a new distributed architecture, rather than focusing on tackling specific non-IID distributions among agents.

The inference starts with the initialization by each individual client i of the node embeddings $\mathbf{v}_{i,n}^{(0)}, \quad \forall j \in \mathcal{V}$, and incoming edge embeddings, $\mathbf{e}_{j\to i,n}^{(0)}, \quad \forall j \in \mathcal{N}_i$. Note that this initialization can be performed with whatever local NN which is not required to retain the same parameters across all clients. Then, for T message passing iterations, the following steps are performed by each client:

1) *Node embeddings exchange*: at message passing iteration $t = 1, \ldots, T$, each client i broadcasts $\mathbf{v}_{i,n}^{(t-1)}$ and receives $\mathbf{v}_{j,n}^{(t-1)}$ from its neighbors $j \in \mathcal{N}_i$. Note that, as in SL, the messages exchanged (smashed data) do not disclose private information.
2) *Edge and node embeddings update*: at message passing iteration $t = 1, \ldots, T$, the edge embeddings are updated as:
$$\mathbf{e}_{j\to i,n}^{(t)} = g_e\left(\mathbf{v}_{j,n}^{(t-1)}, \mathbf{v}_{i,n}^{(t-1)}, \mathbf{e}_{j\to i,n}^{(t-1)}\right), \quad \forall j \in \mathcal{N}_i. \tag{9}$$
Subsequently, the node embeddings are updated as:
$$\mathbf{v}_{i,n}^{(t)} = g_v\left(\mathbf{v}_{i,n}^{(t-1)}, \Phi\left(\{\mathbf{e}_{j\to i,n}^{(t)}\}_{j\,\in\,\mathcal{N}_i}\right)\right). \tag{10}$$
We would like to point out here that (9) and (10) can be modified accordingly to the type of task, input features or particular requirements, without altering the inference structure.
3) *State inference*: lastly, after T message passing steps, each client i predicts the estimate:
$$\widehat{\mathbf{y}}_{i,n} = \widehat{\mathbf{y}}_{i,n}^{(T)} = g_v^{(\text{regres})}\left(\mathbf{v}_{i,n}^{(T)}\right). \tag{11}$$

A graphical representation of the steps is reported in Figure 3.

C. DESIGN OF DISTRIBUTED TRAINING IN SCFL

To design the fully-distributed training procedure peculiar of SCFL, we first observe that the distributed inference relies on the fact that each client needs the same parameters for $g_v(\cdot)$, $g_v^{(\text{regres})}(\cdot)$ and $g_e(\cdot)$, as in centralized MPNN. To enforce this behaviour, we propose three distributed training procedures which are derived from SL and consensus-FL. In particular, SL is adopted for performing a complete distributed inference and back-propagation, while consensus-FL is exploited for convergence to a unique, globally aggregated model. This is especially pertinent, as convergence to a global model has been demonstrated to be attainable with as few as two neighbors, affirming the efficiency of the process [29]. Given the fact that the SL is implemented with a message passing procedure, we can choose the number and type of operations within each message passing iteration. This results in three distinct procedures that we denote as 3SS, 2SS and D-MPNN.

Algorithm 3 3-Steps Strategy

procedure 3SS($\mathcal{N}_i$, η) ▷ Run on client i
  initialize $\mathbf{w}_{i,0} \leftarrow$ client i
  for each round $n = 1, \ldots, N$ **do** ▷ Training loop
    for each message passing iter $t = 1, \ldots, T$ **do**
      compute $F^{(t)}(\mathbf{w}_{i,n})$ ▷ Forward-pass
      broadcast $F^{(t)}(\mathbf{w}_{i,n})$
      receive $\{F^{(t)}(\mathbf{w}_{j,n})\}_{j\in\mathcal{N}_i}$
      compute $\nabla\mathcal{L}^{(t)}_{i,n}(\mathbf{w}_{i,n})$ ▷ Backward-pass
      broadcast $\nabla\mathcal{L}^{(t)}_{i,n}(\mathbf{w}_{i,n})$
      receive $\{\nabla\mathcal{L}^{(t)}_{j,n}(\mathbf{w}_{j,n})\}_{j\in\mathcal{N}_i}$
      $\nabla\overline{\mathcal{L}}^{(t)}_{i,n}(\mathbf{w}_{i,n}) \leftarrow \sum_{j'\in\mathcal{N}_{i^*}} S_{j'}\nabla\mathcal{L}^{(t)}_{j',n} / \sum_{j\in\mathcal{N}_{i^*}} S_j$
      $\mathbf{w}_{i,n} \leftarrow \mathbf{w}_{i,n} - \eta\nabla\overline{\mathcal{L}}^{(t)}_{i,n}(\mathbf{w}_{i,n})$ ▷ Local SGD
    end for
  end for
end procedure

Algorithm 4 2-Steps Strategy

procedure 2SS($\mathcal{N}_i$, η) ▷ Run on client i
  initialize $\mathbf{w}_{i,0} \leftarrow$ client i
  for each round $n = 1, \ldots, N$ **do** ▷ Training loop
    for each message passing iter $t = 1, \ldots, T$ **do**
      compute $F^{(t)}(\mathbf{w}_{i,n})$ ▷ Forward-pass
      broadcast $F^{(t)}(\mathbf{w}_{i,n})$
      receive $\{F^{(t)}(\mathbf{w}_{j,n})\}_{j\in\mathcal{N}_i}$
      compute $\nabla\mathcal{L}^{(t)}_{i,n}(\mathbf{w}_{i,n})$ ▷ Backward-pass
      $\mathbf{w}_{i,n} \leftarrow \mathbf{w}_{i,n} - \eta\nabla\mathcal{L}^{(t)}_{i,n}(\mathbf{w}_{i,n})$ ▷ Local SGD
    end for
    broadcast $\mathbf{w}_{i,n}$
    receive $\{\mathbf{w}_{j,n}\}_{j\in\mathcal{N}_i}$
    $\mathbf{w}_{i,n} \leftarrow \sum_{j'\in\mathcal{N}_{i^*}} S_{j'}\mathbf{w}_{j',n} / \sum_{j\in\mathcal{N}_{i^*}} S_j$
  end for
end procedure

Algorithm 5 Distributed-MPNN Strategy

procedure D-MPNN($\mathcal{N}_i$, η) ▷ Run on client i
  initialize $\mathbf{w}_{i,0} \leftarrow$ client i
  for each round $n = 1, \ldots, N$ **do** ▷ Training loop
    for each message passing iter $t = 1, \ldots, T$ **do**
      compute $F^{(t)}(\mathbf{w}_{i,n})$ ▷ Forward-pass
      broadcast $F^{(t)}(\mathbf{w}_{i,n})$
      receive $\{F^{(t)}(\mathbf{w}_{j,n})\}_{j\in\mathcal{N}_i}$
    end for
    compute $\nabla\mathcal{L}_{i,n}(\mathbf{w}_{i,n})$ ▷ Backward-pass
    $\mathbf{w}_{i,n} \leftarrow \mathbf{w}_{i,n} - \eta\nabla\mathcal{L}_{i,n}(\mathbf{w}_{i,n})$ ▷ Local SGD
    broadcast $\mathbf{w}_{i,n}$
    receive $\{\mathbf{w}_{j,n}\}_{j\in\mathcal{N}_i}$
    $\mathbf{w}_{i,n} \leftarrow \sum_{j'\in\mathcal{N}_{i^*}} S_{j'}\mathbf{w}_{j',n} / \sum_{j\in\mathcal{N}_{i^*}} S_j$
  end for
end procedure

For the sake of notation consistency, we adhere to the notation used in Figure 2, i.e., representing the NN models $g_v(\cdot)$, $g_v^{(\text{regres})}(\cdot)$ and $g_e(\cdot)$ of client i as $\mathbf{w}_i$ and the exchange of node embedding $\mathbf{v}_{i,n}^{(t)}$ as $F(\mathbf{w}_i)$.

In the following, we describe the three proposed SCFL distributed training strategies, i.e., 3SS, 2SS and D-MPNN, highlighting their distinct features for the distributed training of a single epoch. Note that each of them requires T message passing iterations and a consensus-FL step.

1) 3SS

It is constituted by three operations within each message passing iterations, i.e., forward-step, a gradient exchange, and a consensus-FL step, which closely resembles SFL in terms of logical steps. The distinction with respect to SFL is that, due to the absence of a PS, each client already holds all the gradients from its neighbors, without needing an additional exchange of model parameters. Consequently, during the consensus-FL step, the back-propagation is computed using a weighted average of all received gradients, similar to what occurs in CFA with gradient exchange. A graphical representation of the 3SS distributed training strategy for SCFL is given in Figure 4a, while its pseudo-code is highlighted in Algorithm 3.

2) 2SS

It consists of two steps to be performed for T iterations, i.e., a forward-step and a back-propagation step, which are independently computed by each client after the exchange of smashed data. If 2SS is halted after the second step, each client would have distinct model parameters since it retains different private local data for computing the loss function. It follows that, to ensure convergence to a single global model, i.e., identical parameters for $g_v(\cdot)$, $g_v^{(\text{regres})}(\cdot)$, and $g_e(\cdot)$ across all devices, a final step of CFA with model exchange is performed at the end of each training epoch.

A graphical representation of the 2SS distributed training strategy for SCFL is given in Figure 4b, while its pseudo-code is highlighted in Algorithm 4.

3) D-MPNN

This strategy resembles a centralized MPNN training, with the key innovative aspect that it is constituted by fully-distributed steps. This is because first it performs T steps of forward propagation, thus following (5), (6) and (7), and then back-propagation is computed. The key difference here is that after T forward propagations, an individual client back-propagation and a consensus-FL step are executed. Since these last two steps precisely represent the CFA algorithm, we are assured that, under specific conditions, the solution converges to the centralized MPNN outcome, i.e., adopting the centralized loss in (8). A graphical representation of the D-MPNN distributed training strategy for SCFL is given in Figure 4, while its pseudo-code is highlighted in Algorithm 5.

D. INNOVATION ASPECTS OF SCFL OVERCOMING LIMITATIONS OF CURRENT FL ARCHITECTURES

The proposed SCFL framework has different peculiar aspects that overcome the limitations of existing FL approaches. First, it embodies the SL paradigm which permits to train complex DL models within resource-constrained devices by exploiting a message passing procedure derived from MPNN. As a result, the overall bias of the whole derived model is reduced compared to individual client models, consequently enhancing its ability to more accurately identify the true underlying patterns within the data. Second, given the MPNN properties of scalability with the number of nodes, the SCFL framework can be trained on whatever number of clients and, more importantly, can be tested and scaled over complex network topologies without altering the procedure. Lastly, the proposed SCFL is the only method that permits to train DL models in physically separated clients where the inference procedure of each client is strictly dependent on the inference of its neighbors. This fully-distributed training features is of remarkable importance in scenarios where the information retained by the neighbors is necessary for the accomplishment of a task, such as CP which is examined in next section.

IV. SIMULATION EXPERIMENTS

In this section, we first describe a practical application for the SCFL framework, which consists in a fully-distributed procedure for CP in a set of connected agents (e.g., vehicles). Then, we detail the simulated scenario and we present a set of numerical results.

A. USE CASE: COOPERATIVE POSITIONING

We consider a cooperative localization scenario where a set of mobile agents aim to estimate their positions (or state) based on ego-agent location measurements, e.g., noisy agent position, and inter-agent measurements, e.g., agent pairwise-distances. Agents cannot rely on a central coordinator (i.e., the PS), only on data exchange with neighbors.

The state of agent i at time n is denoted as $\mathbf{y}_{i,n}$. Note that here, since we consider a dynamic scenario where the agents move during training and inference, the index n denotes both the time-step and the epoch index. Thus, the network graph becomes n-dependent as $\mathcal{G}_n = (\mathcal{V}_n, \mathcal{E}_n)$. We define with $\mathbf{z}_{i,n}^{(\mathrm{A})} = f^{(\mathrm{A})}\left(\mathbf{y}_{i,n}, \mathbf{w}_{i,n}^{(\mathrm{A})}\right)$ and $\mathbf{z}_{j\to i,n}^{(\mathrm{A2A})} = f^{(\mathrm{A2A})}\left(\mathbf{y}_{j,n}, \mathbf{y}_{i,n}, \mathbf{w}_{i,n}^{(\mathrm{A2A})}\right)$, $\forall j \in \mathcal{N}_{i,n}$, the state and inter-agent measurement, respectively. $\mathbf{w}_{i,n}^{(\mathrm{A})}$ and $\mathbf{w}_{i,n}^{(\mathrm{A2A})}$ are the state and inter-agent measurement noises, respectively, while $f^{(\mathrm{A})}(\cdot)$ and $f^{(\mathrm{A2A})}(\cdot)$ are two non-linear functions. We emphasize that the measurements do not depend on the message passing index t. This follows the assumption that each agent has only one measurement per time-step n, and the time interval between two sub-sequent time-steps is significantly larger than the one between message passing iterations. We refer to Figure 5 for a visual representation of the CP scenario with four agents.

To address this specific CP task, we adopt the following models. For node and edge initialization, we adopt three NNs, i.e., $g_v^{(\mathrm{LSTM})}(\cdot)$, $g_v^{(\mathrm{A})}(\cdot)$ and $g_e^{(\mathrm{A2A})}(\cdot)$, whose parameters and gradients are never shared across agents since they are unique and specific for each agent. A Long Short-Term Memory (LSTM) NN is required to introduce n-dependence relations between time-consecutive state-predictions. At message passing iteration $t = 0$, the initialization and measurement encoding is as follows:

$$\mathbf{v}_{i,n}^{(0)} = g_v^{(\mathrm{LSTM})}(\widehat{\mathbf{y}}_{i,n-1}), \tag{12}$$

$$\mathbf{z}\mathbf{h}_{i,n}^{(\mathrm{A})} = g_v^{(\mathrm{A})}\left(\mathbf{z}_{i,n}^{(\mathrm{A})}\right), \tag{13}$$

$$\mathbf{z}\mathbf{h}_{j\to i,n}^{(\mathrm{A2A})} = g_e^{(\mathrm{A2A})}\left(\mathbf{z}_{j\to i,n}^{(\mathrm{A2A})}\right), \quad \forall j \in \mathcal{N}_{i,n}. \tag{14}$$

At $n = 0$, the inference is initialized as $\widehat{\mathbf{y}}_{i,n-1} \triangleq \mathbb{E}[p(\mathbf{y}_{i,0})]$, where $p(\mathbf{y}_{i,0})$ is the prior knowledge of the agent position. On the contrary, the edge and node embedding update are computed according to:

$$\mathbf{e}_{j\to i,n}^{(t)} = g_e\left(\mathbf{e}_{j\to i,n}^{(t-1)}, \mathbf{z}\mathbf{h}_{j\to i,n}^{(\mathrm{A2A})}, \mathbf{v}_{j,n}^{(t-1)}, \mathbf{v}_{i,n}^{(t-1)}\right), \quad \forall j \in \mathcal{N}_{i,n}, \tag{15}$$

$$\mathbf{v}_{i,n}^{(t)} = g_v\left(\mathbf{v}_{i,n}^{(t-1)}, \mathbf{v}_{i,n}^{(0)}, \mathbf{z}\mathbf{h}_{i,n}^{(\mathrm{A})}, \Phi\left(\{\mathbf{e}_{j\to i,n}^{(t)}\}_{j\in\mathcal{N}_{i,n}}\right)\right). \tag{16}$$

Finally, after T message passing steps, the state prediction is performed as in (11).

B. SIMULATION SCENARIO

We consider a 2D localization scenario where $I_n = 16$ connected agents move within a 200 m $\times$ 200 m area for 100 timesteps sampled at 1 s. The agent trajectories create a spiral shape pattern, starting from the origin and moving towards the area's limits as a spiral (see Figure 6). The graph $\mathcal{G}_n$ is fully-connected, i.e., each agent is connected to all the others. The agent's state is $\mathbf{y}_{i,n} = [\mathbf{p}_{i,n}^{\mathrm{T}}\dot{\mathbf{p}}_{i,n}^{\mathrm{T}}]^{\mathrm{T}}$, where $\mathbf{p}_{i,n} \in \mathbb{R}^2$ and $\dot{\mathbf{p}}_{i,n} \in \mathbb{R}^2$ represent the 2D position and velocity, respectively. The measurements are defined as $\mathbf{z}_{i,n}^{(\mathrm{A})} = \mathbf{y}_{i,n} + \mathbf{w}_{i,n}^{(\mathrm{A})}$ and $\mathbf{z}_{j\to i,n}^{(\mathrm{A2A})} = \|\mathbf{p}_{j,n} - \mathbf{p}_{i,n}\|_2 + \mathbf{w}_{i,n}^{(\mathrm{A2A})}$. We model the agent kinematics with a constant velocity motion model, unless stated otherwise, while the state measurements and inter-agent measurements follow zero-mean Gaussian distributions, i.e., $\mathbf{w}_{i,n}^{(\mathrm{A})} \sim \mathcal{N}\left(\mathbf{0}_4, \mathbf{C}_{\mathbf{w}^{(\mathrm{A})}}\right)$, with $\mathbf{C}_{\mathbf{w}^{(\mathrm{A})}} = \mathrm{diag}\left(\sigma^2_{\mathbf{p},\mathbf{w}^{(\mathrm{A})}}, \sigma^2_{\mathbf{p},\mathbf{w}^{(\mathrm{A})}}, \sigma^2_{\dot{\mathbf{p}},\mathbf{w}^{(\mathrm{A})}}, \sigma^2_{\dot{\mathbf{p}},\mathbf{w}^{(\mathrm{A})}}\right)$, and $\mathbf{w}_{i,n}^{(\mathrm{A2A})} \sim \mathcal{N}\left(0, \sigma^2_{\mathbf{w}^{(\mathrm{A2A})}}\right)$, with standard deviations $\sigma_{\mathbf{p},\mathbf{w}^{(\mathrm{A})}} = 5$ m, $\sigma_{\dot{\mathbf{p}},\mathbf{w}^{(\mathrm{A})}} = 1$ m/s, and $\sigma_{\mathbf{w}^{(\mathrm{A2A})}} = 2$ m.

The network of agents is trained on 1,000 instances of constant velocity linear trajectories, with $\dot{\mathbf{p}}_{i,n} \in [-10, 10]$ m/s, where each instance is composed of $I_n = 16$ connected agents. To enhance model convergence and prevent biases, we standardized all samples by applying a min-max scaler, ensuring each feature falls within the [0, 1] range. This is done with prior knowledge of agent position,

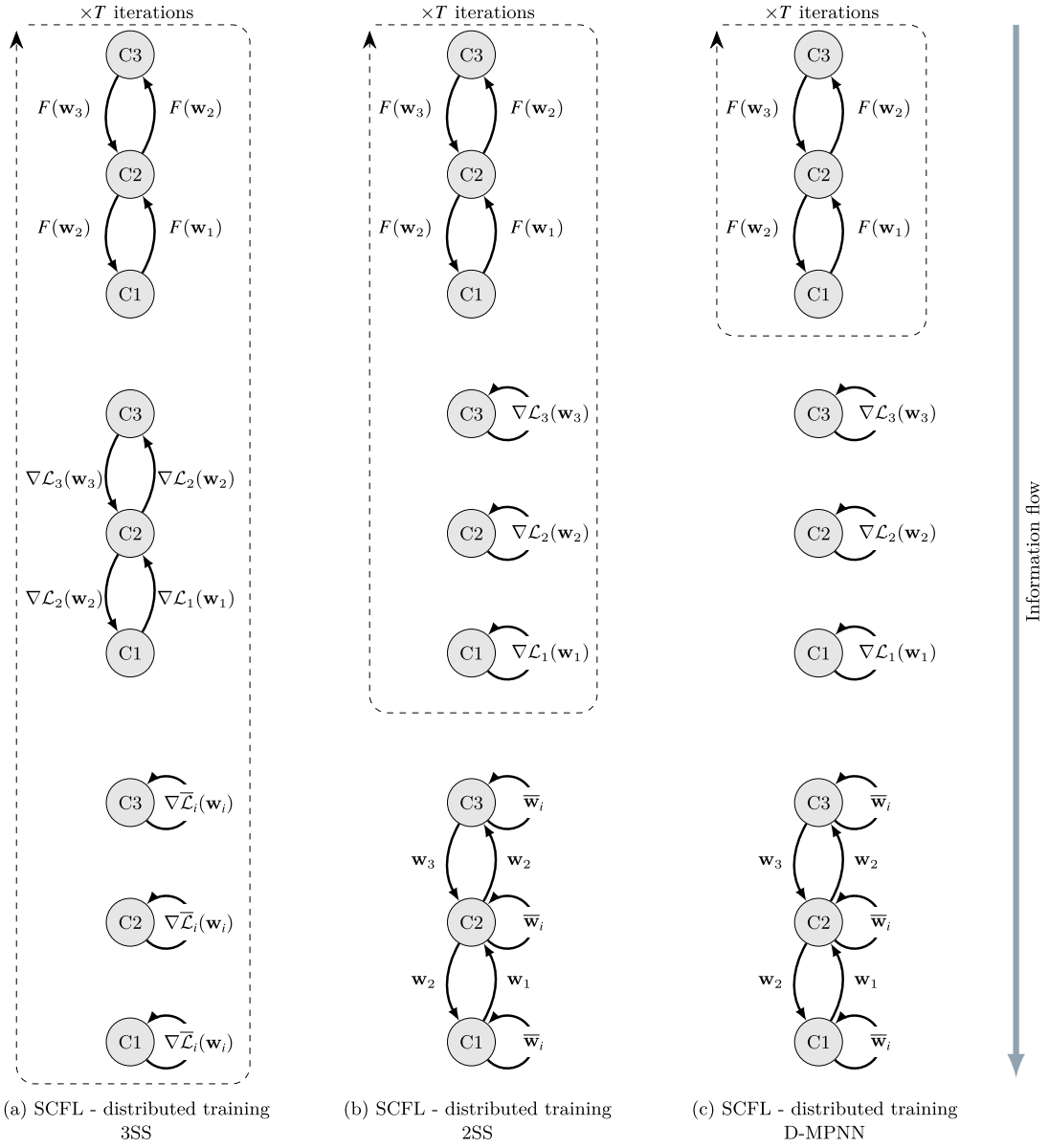


FIGURE 4. Visualization of SCFL distributed training procedure according to three types of strategies. a) 3SS; b) 2SS and c) D-MPNN. The right arrow highlights the flow of information, i.e., the sequential order of operations to be performed for updating the model parameters during training. Note that in all the SCFL strategies, the PS is missing, being the training achieved in a fully-distributed manner.

i.e., $\mathbf{p}_{i,n} \in [-100, 100]$ m, and velocity, i.e., $\dot{\mathbf{p}}_{i,n} \in [-10, 10]$ m/s. We trained both the centralized and distributed models for a total of 100 epochs, using a batch size of 32 samples and randomizing the dataset order at the beginning of each epoch. Here, a sample refers to a trajectory instance composed of $N = 10$ timesteps, i.e., the training length sequence of the LSTM model.

The LSTM has been modified from [61], employing only two layers and a hidden output dimension, or node embeddings, of 16. The NNs of the MPNN model are

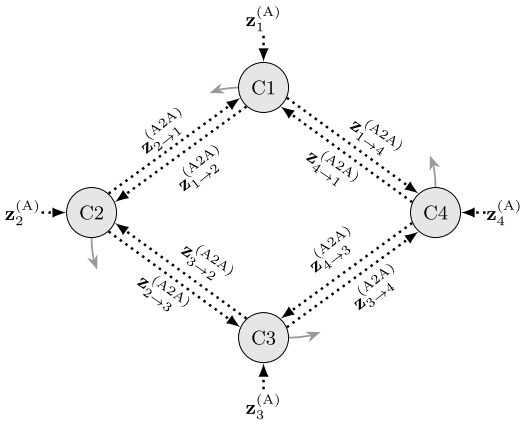


FIGURE 5. CP task with four agents, represented by circles. The measurements are represented as black dotted arrows, while trajectory paths are indicated with colored solid arrows. For ease of notation, we omit the epoch index *n*.

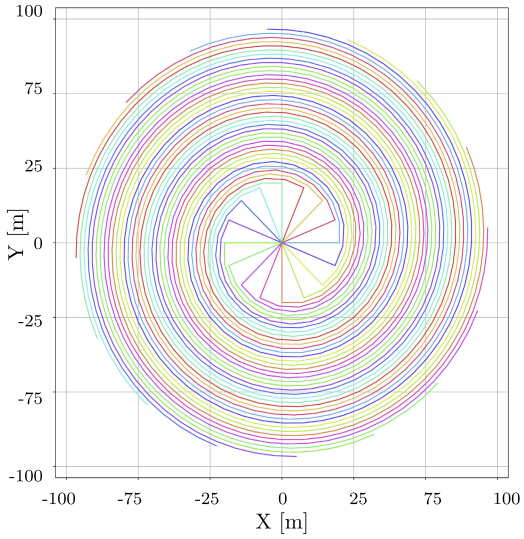


FIGURE 6. Example of spiral scenario composed of 16 cooperating agents represented by different colors.

Multi-Layer Perceptrons (MLPs) with two hidden layers and a neuron count of [80, 16] with Gaussian Error Linear Units (GELU) activation functions. The number of message passing steps is $T = 10$. Lastly, we consider dimension of 16 for edge embeddings, state, and inter-agent measurements.

For model training and testing, we used PyTorch version 1.12 and Python version 3.7.11, while simulations were executed on a workstation featuring an Intel(R) Xeon(R) Silver 4210R CPU operating at 2.40 GHz, 96 GB of RAM, and a Quadro RTX 6000 24 GB GPU. Regarding the optimizer, we employed the Adam optimization algorithm [59] with an initial learning rate of 0.0001 and momentum values of 0.9 and 0.999 for β_1 and β_2, respectively.

C. NUMERICAL RESULTS

1) TRAINING

In a first assessment, we compare the performances of the three proposed SCFL strategies (i.e., 3SS, 2SS and D-MPNN) with respect to a centralized MPNN solution where all clients, i.e., nodes of the graph, lie in a single machine. In Figure 7 we show the Root Mean Square Error (RMSE) of the position and velocity estimates evaluated on the validation dataset for each epoch of the training (Figure 7a) and corresponding wall clock time (Figure 7b).

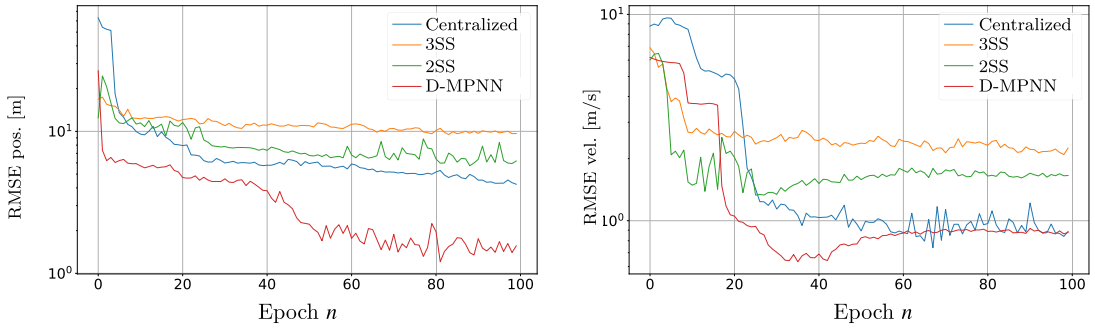
Observing the results, we note that at corresponding epoch, the best performances are reached by the distributed D-MPNN and the centralized solution, followed by 2SS and 3SS implementations. This result suggests that the cooperative forwarding pass of the training should be performed for all message passing iterations in the first stage of the training (see Figure 4c). In contrast, increasing the number of operations in message passing, as in 3SS and 2SS, leads to worse performances. We believe that this is due to the fact that performing all forwarding iterations at the beginning allows client models to fully exploit the potential of message passing operations, without interrupting the flow of refined information within the node and edge embeddings.

An interesting observation is that the proposed SCFL method with D-MPNN outperforms also the centralized solution. To explain this behavior, we note in Figure 7b that the centralized solution completes training in much less time, and its RMSE is similar to the one of D-MPNN, only with less machine time spent on training. Therefore, we assert that the centralized solution and distributed D-MPNN exhibit practically the same performance with the same computational resources. The reason why D-MPNN converges faster in terms of epochs is that step 2 of D-MPNN, which involves gradient computation and back-propagation, is performed individually by each client, thereby increasing the computations linearly with the number of clients. In fully-distributed networks of agents, this feature can lead a significant advantage in speeding up the training process.

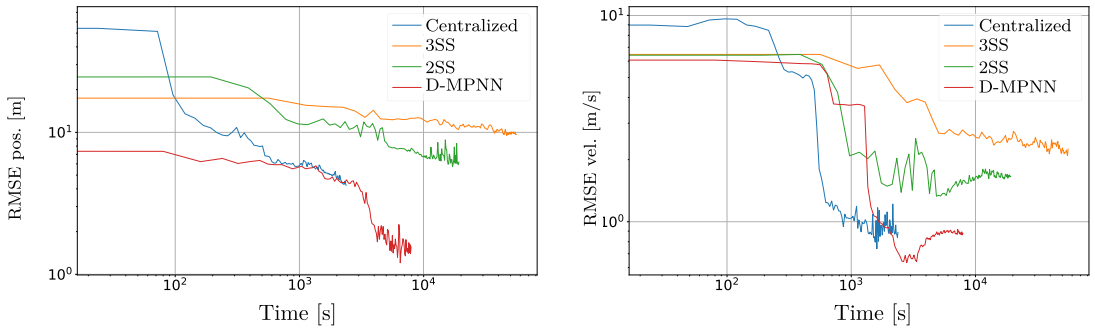
2) CONVERGENCE

This experiment aims at verifying the convergence of the three proposed fully-distributed SCFL strategies with respect to centralized training. In other words, the objective is to assess whether or not the distributed training among clients is equivalent or very well approximated to the centralized architecture in terms of client local model parameters. To this aim, we studied how much the distributions of $g_v(\cdot)$, $g_v^{(\text{regres})}(\cdot)$ and $g_e(\cdot)$ vary between the centralized and distributed strategies.

In Figure 8 we show the Kullback-Leibler (KL) divergence of the model parameter distributions with respect to the



(a) Validation results for varying training epochs.



(b) Validation results with respect to wall clock time.

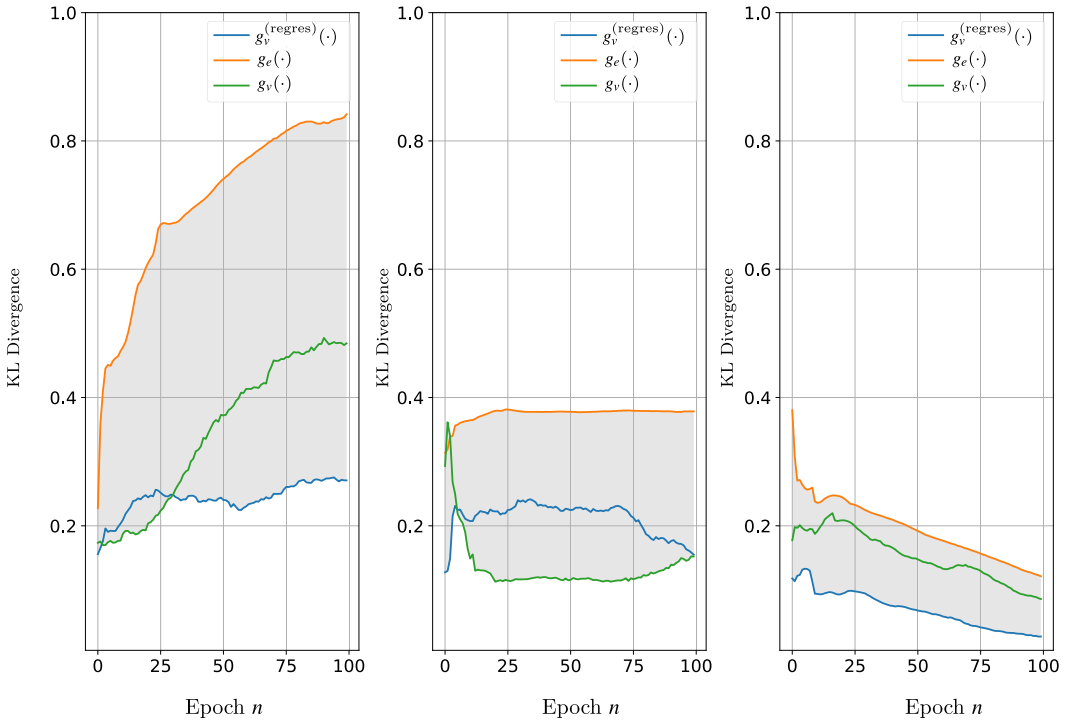
FIGURE 7. Comparison of the three proposed SCFL strategies with respect to a centralized solution. a) RMSE of position and velocity on the validation set for each training epoch. b) RMSE of position and velocity on the validation set with respect to the wall clock time of the training.

centralized model parameters for each epoch of training. We start noticing that, in 3SS (Figure 8a), the KL divergence seems somehow to diverge from the centralized solution for all the three local models. This confirms that performing many operations in a single message passing iteration, i.e., a forward-step, a gradient exchange, and a consensus-FL step, is not beneficial as it does not fully exploit the message passing elaboration of latent features. On the contrary, in 2SS (Figure 8b), we observe a neutral behavior with local parameter distributions that tend to hold the same distance with respect to the centralized solution. This is due to the fact that steps 1 and 2 (see Figure 4b) lead to a more biased local model, since each client adopts its private data to compute the gradients and perform back-propagation, while step 3 drives to a common global model which resembles the centralized solution exactly as in FL. Finally, D-MPNN (Figure 8c), which holds the best performances, clearly converges to the centralized approach, as the logical steps of the distributed algorithm match the classical MPNN training.

3) BASELINE COMPARISON

In this assessment, we compare the proposed D-MPNN method with the current state-of-the-art D-ML algorithm, i.e., CFA, both in terms of performances and communication efficiency. Since in CFA there is no exchange of smashed data, we train the same NNs present in D-MPNN without the message passing procedure. We can consider the CFA as a contraction of D-MPNN, where the T iterations are performed within each agent and where the cooperation is only present in the last step of weights exchange.

In Figure 9 we show the validation results of D-MPNN ad CFA varying the number of training epochs. We also connect the points in the two curves that correspond to the same training time, allowing to evaluate the trade-off between performances and training efficiency. From the results, we notice that the D-MPNN outperforms CFA at every epoch, both in terms of convergence speed and achieved RMSE. This is mainly due to the message passing procedure, which permits to further elaborate the input measurements by exploiting the neighbors smashed data. Despite the longer epoch duration due to the message passing, we observe that, at the same time instant, the D-MPNN constantly achieves better performances. This demonstrates that the message passing procedure can indeed reduce the biases of the simpler independent models within the agents, in the same way as in conventional centralized MPNN.



(a) SCFL - distributed training 3SS (b) SCFL - distributed training 2SS (c) SCFL - distributed training D-MPNN

FIGURE 8. KL divergence, at varying training epochs, of the local model parameter distributions between the centralized and the SCFL distributed training strategies, i.e., a) 3SS; b) 2SS and c) D-MPNN.

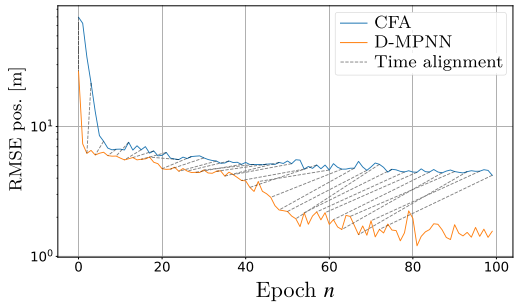


FIGURE 9. Comparison of the proposed SCFL D-MPNN method with respect to the CFA algorithm in terms of RMSE. The points related to the same training time are connected by dashed lines.

V. CONCLUSION

This paper addressed the problem of distributed inference and model training in a network of physically separated clients. We started by reviewing the parallelism between PS-based FL, SL and SFL methods which is used as a starting point for the design of a fully-distributed consensus-based SL, named SCFL. In this framework, clients have to forward the smashed data to their neighbors for completing the inference. We developed three distributed training strategies, namely 3SS, 2SS and D-MPNN, which take inspiration from centralized MPNN where the message passing iterations resemble the split forwarding, while preserving local data privacy. These strategies mainly differ for the type of operations within each single message passing iteration, thus obtaining different performances under the same conditions. The main advantages are the local data privacy preservation, as in FL and SL, since only smashed data, gradients and model parameters are exchanged, and the complete independence on a centralized entity (i.e., a PS) to perform the training procedure.

We proved the efficacy of the proposed SCFL paradigm in a CP use-case where distributed moving clients (i.e., agents) have to self-localize based on local ego-agent and inter-agents measurements. For the specific task, we developed a custom model structure composed of client-specific models, i.e., encoding NNs and an LSTM model to learn the agent mobility, and shared models, i.e., edge, node and regressor NNs. Comparing the three distributed strategies with respect to a centralized solution, we found that the distributed strategies highly differ in terms of performance and convergence capabilities. Specifically, the best learning strategy, i.e., D-MPNN, is the one that takes the most

advantage from message passing by first performing the cooperative inference through node embeddings exchange, and then applying individual back-propagation with a final consensus-FL step. This strategy efficiently combines the computational power of all clients as they were on a single machine and strives towards the centralized model parameters at convergence.

We expect distributed training and inference to play a crucial role in forthcoming years, especially in applications such as cooperative sensing in connected automated vehicles, distributed manufacturing control, and autonomous multi-agent robot systems, where collaboration is requested to achieve a shared objective. Relevant challenges are represented by the requirement of fast convergence in time-sensitive tasks and the management of non-IID data in heterogeneous settings.

REFERENCES

[1] A. Qayyum, J. Qadir, M. Bilal, and A. Al-Fuqaha, "Secure and robust machine learning for healthcare: A survey," *IEEE Rev. Biomed. Eng.*, vol. 14, pp. 156–180, 2021.

[2] M. Chen, Y. Hao, K. Hwang, L. Wang, and L. Wang, "Disease prediction by machine learning over big data from healthcare communities," *IEEE Access*, vol. 5, pp. 8869–8879, 2017.

[3] M. Rizinski, H. Peshov, K. Mishev, L. T. Chitkushev, I. Vodenska, and D. Trajanov, "Ethically responsible machine learning in fintech," *IEEE Access*, vol. 10, pp. 97531–97554, 2022.

[4] M. F. Dixon, I. Halperin, and P. Bilokon, *Machine Learning in Finance: From Theory to Practice*. Switzerland: Springer, Jul. 2020.

[5] F. Zantalis, G. Koulouras, S. Karabetsos, and D. Kandris, "A review of machine learning and IoT in smart transportation," *Future Internet*, vol. 11, no. 4, p. 94, Apr. 2019.

[6] L. Barbieri, S. Savazzi, M. Brambilla, and M. Nicoli, "Decentralized federated learning for extended sensing in 6G connected vehicles," *Veh. Commun.*, vol. 33, Jan. 2022, Art. no. 100396.

[7] P. Vepakomma, T. Swedish, R. Raskar, O. Gupta, and A. Dubey, "No peek: A survey of private distributed deep learning," 2018, *arXiv:1812.03288*.

[8] C. Thapa, M. A. P. Chamikara, and S. A. Camtepe, "Advancements of federated learning towards privacy preservation: From federated learning to split learning," in *Federated Learning Systems*, vol. 965, M. H. U. Rehman and M. M. Gaber, Eds. Switzerland: Springer, Jun. 2021, pp. 79–109.

[9] J. Koneč ný, B. McMahan, and D. Ramage, "Federated optimization: Distributed optimization beyond the datacenter," 2015, *arXiv:1511.03575*.

[10] H. B. McMahan, E. Moore, D. Ramage, S. Hampson, and B. A. Y. Arcas, "Communication-efficient learning of deep networks from decentralized data," in *Proc. 20th Int. Conf. Artif. Intell. Statist. (AISTATS)*, 2016, pp. 1273–1282.

[11] Q. Yang, Y. Liu, T. Chen, and Y. Tong, "Federated machine learning: Concept and applications," *ACM Trans. Intell. Syst. Technol.*, vol. 10, no. 2, pp. 1–19, Mar. 2019.

[12] T. Li, A. K. Sahu, A. Talwalkar, and V. Smith, "Federated learning: Challenges, methods, and future directions," *IEEE Signal Process. Mag.*, vol. 37, no. 3, pp. 50–60, May 2020.

[13] B. Camajori Tedeschini, S. Savazzi, R. Stoklasa, L. Barbieri, I. Stathopoulos, M. Nicoli, and L. Serio, "Decentralized federated learning for healthcare networks: A case study on tumor segmentation," *IEEE Access*, vol. 10, pp. 8693–8708, 2022.

[14] B. C. Tedeschini, S. Savazzi, and M. Nicoli, "A traffic model based approach to parameter server design in federated learning processes," *IEEE Commun. Lett.*, vol. 27, no. 7, pp. 1774–1778, Jul. 2023.

[15] L. Barbieri, S. Savazzi, and M. Nicoli, "A layer selection optimizer for communication-efficient decentralized federated deep learning," *IEEE Access*, vol. 11, pp. 22155–22173, 2023.

[16] L. Barbieri, S. Savazzi, and M. Nicoli, "Communication-efficient distributed learning in V2X networks: Parameter selection and quantization," in *Proc. IEEE Global Commun. Conf. (GLOBECOM)*, Dec. 2022, pp. 603–608.

[17] O. Gupta and R. Raskar, "Distributed learning of deep neural network over multiple agents," *J. Netw. Comput. Appl.*, vol. 116, pp. 1–8, Aug. 2018.

[18] K. Palanisamy, V. Khimani, M. H. Moti, and D. Chatzopoulos, "SplitEasy: A practical approach for training ML models on mobile devices," in *Proc. 22nd Int. Workshop Mobile Comput. Syst. Appl.*, Feb. 2021, pp. 37–43.

[19] A. Singh, P. Vepakomma, O. Gupta, and R. Raskar, "Detailed comparison of communication efficiency of split learning and federated learning," 2019, *arXiv:1909.09145*.

[20] T. Titcombe, A. J. Hall, P. Papadopoulos, and D. Romanini, "Practical defences against model inversion attacks for split neural networks," 2021, *arXiv:2104.05743*.

[21] O. Li, J. Sun, X. Yang, W. Gao, H. Zhang, J. Xie, V. Smith, and C. Wang, "Label leakage and protection in two-party split learning," 2021, *arXiv:2102.08504*.

[22] E. Erdogan, A. Küpçü, and A. E. Çiçek, "UnSplit: Data-oblivious model inversion, model stealing, and label inference attacks against split learning," in *Proc. 21st Workshop Privacy Electron. Soc.*, Nov. 2022, pp. 115–124.

[23] H. Madaan, M. G. V. Kulkarni, and A. Pant, "Vulnerability due to training order in split learning," in *ICT Systems and Sustainability*, vol. 321, M. Tuba, S. Akashe, and A. Joshi, Eds. Singapore: Springer, Jan. 2022, pp. 103–112.

[24] D. Pasquini, G. Ateniese, and M. Bernaschi, "Unleashing the tiger: Inference attacks on split learning," in *Proc. ACM SIGSAC Conf. Comput. Commun. Secur.*, Nov. 2021, pp. 2113–2129.

[25] Y. Li and X. Lyu, "On the Convergence of FedAvg on Non-IID Data," Jun. 2023, *arXiv:1907.02189*.

[26] Y. Gao, M. Kim, C. Thapa, A. Abuadbba, Z. Zhang, S. Camtepe, H. Kim, and S. Nepal, "Evaluation and optimization of distributed machine learning techniques for Internet of Things," *IEEE Trans. Comput.*, vol. 71, no. 10, pp. 2538–2552, Oct. 2022.

[27] Y. Gao, M. Kim, S. Abuadbba, Y. Kim, C. Thapa, K. Kim, S. A. Camtep, H. Kim, and S. Nepal, "End-to-end evaluation of federated learning and split learning for Internet of Things," in *Proc. Int. Symp. Reliable Distrib. Syst. (SRDS)*, Sep. 2020, pp. 91–100.

[28] I. Hegedus, G. Danner, and M. Jelasity, "Gossip learning as a decentralized alternative to federated learning," in *Distributed Applications and Interoperable Systems*, vol. 11534, J. Pereira and L. Ricci, Eds. Cham, Switzerland: Springer, Jun. 2019, pp. 74–90.

[29] S. Savazzi, M. Nicoli, and V. Rampa, "Federated learning with cooperating devices: A consensus approach for massive IoT networks," *IEEE Internet Things J.*, vol. 7, no. 5, pp. 4641–4654, May 2020.

[30] J. Mills, J. Hu, and G. Min, "Communication-efficient federated learning for wireless edge intelligence in IoT," *IEEE Internet Things J.*, vol. 7, no. 7, pp. 5986–5994, Jul. 2020.

[31] H. Choi and I. V. Bajic, "Deep feature compression for collaborative object detection," in *Proc. 25th IEEE Int. Conf. Image Process. (ICIP)*, Oct. 2018, pp. 3743–3747.

[32] Y. Kang, J. Hauswald, C. Gao, A. Rovinski, T. Mudge, J. Mars, and L. Tang, "Neurosurgeon: Collaborative intelligence between the cloud and mobile edge," *ACM SIGARCH Comput. Archit. News*, vol. 45, no. 1, pp. 615–629, Apr. 2017.

[33] S. Teerapittayanon, B. McDanel, and H. T. Kung, "Distributed deep neural networks over the cloud, the edge and end devices," in *Proc. IEEE 37th Int. Conf. Distrib. Comput. Syst. (ICDCS)*, Jun. 2017, pp. 328–339.

[34] J. Jeon and J. Kim, "Privacy-sensitive parallel split learning," in *Proc. Int. Conf. Inf. Netw. (ICOIN)*, Jan. 2020, pp. 7–9.

[35] C. Thapa, P. C. Mahawaga Arachchige, S. Camtepe, and L. Sun, "SplitFed: When federated learning meets split learning," in *Proc. AAAI Conf. Artif. Intell.*, Jun. 2022, vol. 36, no. 8, pp. 8485–8493.

[36] W. Wu, M. Li, K. Qu, C. Zhou, X. Shen, W. Zhuang, X. Li, and W. Shi, "Split learning over wireless networks: Parallel design and resource management," *IEEE J. Sel. Areas Commun.*, vol. 41, no. 4, pp. 1051–1066, Apr. 2023.

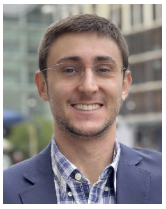
[37] S. Oh, J. Park, P. Vepakomma, S. Baek, R. Raskar, M. Bennis, and S. L. Kim, "LocFedMix-SL: Localize, federate, and mix for improved scalability, convergence, and latency in split learning," in *Proc. ACM Web Conf.*, Apr. 2022, pp. 3347–3357.

[38] A. Abedi and S. S. Khan, "FedSL: Federated split learning on distributed sequential data in recurrent neural networks," *Multimedia Tools Appl.*, vol. 83, no. 10, pp. 28891–28911, Sep. 2024.

[39] W. Zhou, Z. Qu, Y. Zhao, B. Tang, and B. Ye, "An efficient split learning framework for recurrent neural network in mobile edge environment," in *Proc. Conf. Res. Adapt. Convergent Syst.*, Oct. 2022, pp. 131–138.

[40] F. Scarselli, M. Gori, A. C. Tsoi, M. Hagenbuchner, and G. Monfardini, "Computational capabilities of graph neural networks," *IEEE Trans. Neural Netw.*, vol. 20, no. 1, pp. 81–102, Jan. 2009.

[41] F. Scarselli, M. Gori, A. C. Tsoi, M. Hagenbuchner, and G. Monfardini, "The graph neural network model," *IEEE Trans. Neural Netw.*, vol. 20, no. 1, pp. 61–80, Jan. 2009.
[42] B. C. Tedeschini, M. Brambilla, L. Barbieri, and M. Nicoli, "Addressing data association by message passing over graph neural networks," in *Proc. 25th Int. Conf. Inf. Fusion (FUSION)*, Jul. 2022, pp. 1–7.
[43] B. Camajori Tedeschini, M. Brambilla, L. Barbieri, G. Balducci, and M. Nicoli, "Cooperative lidar sensing for pedestrian detection: Data association based on message passing neural networks," *IEEE Trans. Signal Process.*, vol. 71, pp. 3028–3042, 2023.
[44] B. Camajori Tedeschini, M. Brambilla, and M. Nicoli, "Message passing neural network versus message passing algorithm for cooperative positioning," *IEEE Trans. Cogn. Commun. Netw.*, vol. 9, no. 6, pp. 1666–1676, Dec. 2023.
[45] M. Brambilla, D. Gaglione, G. Soldi, R. Mendrzik, G. Ferri, K. D. LePage, M. Nicoli, P. Willett, P. Braca, and M. Z. Win, "Cooperative localization and multitarget tracking in agent networks with the sum-product algorithm," *IEEE Open J. Signal Process.*, vol. 3, pp. 169–195, 2022.
[46] N. Patwari, J. Ash, S. Kyperountas, A. Hero, R. Moses, and N. Correal, "Locating the nodes: Cooperative localization in wireless sensor networks," *IEEE Signal Process. Mag.*, vol. 22, no. 4, pp. 54–69, Jul. 2005.
[47] M. Z. Win, A. Conti, S. Mazuelas, Y. Shen, W. M. Gifford, D. Dardari, and M. Chiani, "Network localization and navigation via cooperation," *IEEE Commun. Mag.*, vol. 49, no. 5, pp. 56–62, May 2011.
[48] M. Z. Win, Y. Shen, and W. Dai, "A theoretical foundation of network localization and navigation," *Proc. IEEE*, vol. 106, no. 7, pp. 1136–1165, Jul. 2018.
[49] G. Soatti, M. Nicoli, N. Garcia, B. Denis, R. Raulefs, and H. Wymeersch, "Implicit cooperative positioning in vehicular networks," *IEEE Trans. Intell. Transp. Syst.*, vol. 19, no. 12, pp. 3964–3980, Dec. 2018.
[50] M. Brambilla, M. Nicoli, G. Soatti, and F. Deflorio, "Augmenting vehicle localization by cooperative sensing of the driving environment: Insight on data association in urban traffic scenarios," *IEEE Trans. Intell. Transp. Syst.*, vol. 21, no. 4, pp. 1646–1663, Apr. 2020.
[51] L. Barbieri, B. Camajori Tedeschini, M. Brambilla, and M. Nicoli, "Deep learning-based cooperative LiDAR sensing for improved vehicle positioning," *IEEE Trans. Signal Process.*, vol. 72, pp. 1666–1682, 2024.
[52] B. Camajori Tedeschini and M. Nicoli, "Cooperative deep-learning positioning in mmWave 5G-advanced networks," *IEEE J. Sel. Areas Commun.*, vol. 41, no. 12, pp. 3799–3815, Dec. 2023.
[53] Y. Ma, C. Tian, and Y. Jiang, "A multitag cooperative localization algorithm based on weighted multidimensional scaling for passive UHF RFID," *IEEE Internet Things J.*, vol. 6, no. 4, pp. 6548–6555, Aug. 2019.
[54] A. Conti, S. Mazuelas, S. Bartoletti, W. C. Lindsey, and M. Z. Win, "Soft information for Localization-of-Things," *Proc. IEEE*, vol. 107, no. 11, pp. 2240–2264, Nov. 2019.
[55] N. Forti, E. d'Afflisio, P. Braca, L. M. Millefiori, S. Carniel, and P. Willett, "Next-gen intelligent situational awareness systems for maritime surveillance and autonomous navigation [point of view]," *Proc. IEEE*, vol. 110, no. 10, pp. 1532–1537, Oct. 2022.
[56] Y. Li, Y. Wang, W. Yu, and X. Guan, "Multiple autonomous underwater vehicle cooperative localization in anchor-free environments," *IEEE J. Ocean. Eng.*, vol. 44, no. 4, pp. 895–911, Oct. 2019.
[57] J. Wang, C. Jiang, Z. Han, Y. Ren, R. G. Maunder, and L. Hanzo, "Taking drones to the next level: Cooperative distributed Unmanned-Aerial-Vehicular networks for small and mini drones," *IEEE Veh. Technol. Mag.*, vol. 12, no. 3, pp. 73–82, Sep. 2017.
[58] A. Guerra, F. Guidi, D. Dardari, and P. M. Djuric, "Networks of UAVs of low complexity for time-critical localization," *IEEE Aerosp. Electron. Syst. Mag.*, vol. 37, no. 10, pp. 22–38, Oct. 2022.
[59] D. P. Kingma and J. Ba, "Adam: A method for stochastic optimization," 2014, *arXiv:1412.6980*.
[60] J. Gilmer, S. S. Schoenholz, P. F. Riley, O. Vinyals, and G. E. Dahl, "Neural message passing for quantum chemistry," in *Proc. 34th Int. Conf. Mach. Learn.*, Jul. 2017, pp. 1263–1272.
[61] J. Liu, Z. Wang, and M. Xu, "DeepMTT: A deep learning maneuvering target-tracking algorithm based on bidirectional LSTM network," *Inf. Fusion*, vol. 53, pp. 289–304, Jan. 2020.



BERNARDO CAMAJORI TEDESCHINI (Graduate Student Member, IEEE) received the B.Sc. (Hons.) in Computer Science and M.Sc. (Hons.) degrees in Telecommunications Engineering from the Politecnico di Milano, Italy, in 2019 and 2021, respectively. From November 2021 he started as PhD fellow in Information Technology at Dipartimento di Elettronica, Informazione e Bioingegneria (DEIB), Politecnico di Milano. He is currently a visiting researcher with the Laboratory for Information & Decision Systems at the Massachusetts Institute of Technology (MIT), Cambridge, MA.

His research interests include federated learning, machine learning and localization methods. He was a recipient of the Ph.D. grant from the ministry of the Italian government Ministero dell'Istruzione, dell'Università e della Ricerca (MIUR) and the Roberto Rocca Doctoral Fellowship granted by MIT and Politecnico di Milano.

MATTIA BRAMBILLA (Member, IEEE) received the B.Sc. and M.Sc. degrees in telecommunication engineering and the Ph.D. degree (cum laude) in information technology from the Politecnico di Milano, in 2015, 2017, and 2021, respectively. He was a Visiting Researcher with the NATO Centre for Maritime Research and Experimentation (CMRE), La Spezia, Italy, in 2019. In 2021 he joined the faculty of Dipartimento di Elettronica, Informazione e Bioingegneria (DEIB) at the Politecnico di Milano as Research Fellow. His research interests include signal processing, statistical learning, and data fusion for cooperative localization and communication. He was a recipient of the Best Student Paper Award at the 2018 IEEE Statistical Signal Processing Workshop.

MONICA NICOLI (Senior Member, IEEE) received the M.Sc. (Hons.) and Ph.D. degrees in communication engineering from Politecnico di Milano, Milan, Italy, in 1998 and 2002, respectively. She was a Visiting Researcher with ENI Agip, from 1998 to 1999, and Uppsala University, in 2001. In 2002, she joined Politecnico di Milano as a Faculty Member. She is currently an Associate Professor in telecommunications with the Department of Management, Economics and Industrial Engineering.

Her research interests include signal processing, machine learning, and wireless communications, with emphasis on smart mobility and Internet of Things (IoT). She was a recipient of the Marisa Bellisario Award, in 1999, and a co-recipient of the best paper awards of the EuMA Mediterranean Microwave Symposium, in 2022, the IEEE Symposium on Joint Communications and Sensing, in 2021, the IEEE Statistical Signal Processing Workshop, in 2018, and the IET Intelligent Transport Systems journal, in 2014. She is an Associate Editor of the IEEE TRANSACTIONS ON INTELLIGENT TRANSPORTATION SYSTEMS. She has also served as an Associate Editor for the EURASIP *Journal on Wireless Communications and Networking*, from 2010 to 2017, and a Lead Guest Editor for the Special Issue on Localization in Mobile Wireless and Sensor Networks, in 2011.

Open Access funding provided by 'Politecnico di Milano' within the CRUI CARE Agreement

CHAPTER 6

Multi-Agent Reinforcement Learning

In this chapter, we present a data-driven approach for extending the ICP framework and overcoming its main limitations due to the cyclicity of the factor graph. In particular, we propose a MARL algorithm, namely ICP-MAPPO, tailored for highly non-stationary learning where the agent network, i.e., a vehicular network, continuously varies in time. We model the problem as a Dec-POMDP where agents aim at performing CP by exploiting exchanged measurements, comprising detection of passive objects that act as reference points to refine the predictions. The agents predict the state evolution by beliefs learning and follow a policy that dictates which links deactivate to simultaneously improve performances and achieve high communication efficiency. To solve the issue of partial observability of the state, we employ the C-ML paradigm for training, with subsequent deployment of the models for decentralized execution. Results show that this new scheme, named centralized-training and dynamic-decentralized-execution, is able to outperform the ICP algorithm when it comes to both positioning accuracy, speed of convergence and efficiency of neighbors' cooperation.

Multi-agent Reinforcement Learning for Distributed Cooperative Vehicular Positioning

Bernardo Camajori Tedeschini, *Student Member, IEEE*, Mattia Brambilla, *Member, IEEE*, Monica Nicoli, *Senior Member, IEEE* and Moe Z. Win, *Fellow, IEEE*

***Abstract*—With the advent of cooperative intelligent transport systems (C-ITS) and vehicle-to-everything (V2X) communications, cooperative positioning based on V2X sharing of location information has been emerging as a promising augmentation system for conventional satellite navigation. An example is implicit cooperative positioning (ICP) which relies on Bayesian filtering for cooperative sensing of targets that are used as reference points for improving vehicle positioning. However, ICP methods rely on pre-determined models which makes them sub-optimal in case of non-Gaussian non-linear models or complex cooperation graphs. To address these limitations, the paper proposes a decentralized-partially observable Markov decision process (Dec-POMDP) framework, paired with deep multi-agent reinforcement learning (MARL) algorithms. We introduce a novel ICP-multi-agent proximal policy optimization (MAPPO) algorithm where distributed agents (i.e., vehicles) dynamically activate/deactivate the radio links with the neighbors to optimize the communication efficiency, still guaranteeing accurate positioning. We reproduce a realistic C-ITS scenario with CARLA simulator, where vehicles move according to real-world dynamics and communicate with each other. Results show that the proposed ICP-MAPPO algorithm, with its dynamic-decentralized-execution and centralized-training schemes, outperforms state-of-the-art ICP methods by 21% in terms of positioning accuracy, and it can reduce the communication overhead by following the optimal learned policy.**

***Index Terms*—Multi-agent reinforcement learning, Dec-POMDP, implicit cooperative positioning, Bayesian-filtering, message passing algorithm.**

Manuscript received 21 June 2024; revised 7 September 2024; accepted 19 September 2024. Date of publication XX YYYY 2024; date of current version XX YYYY 2024. The fundamental research described in this paper was supported, in part, by the Roberto Rocca Doctoral Fellowship granted by the Massachusetts Institute of Technology and Politecnico di Milano, by MOST – Sustainable Mobility National Research Center and received funding from the European Union Next-GenerationEU (PIANO NAZIONALE DI RIPRESA E RESILIENZA (PNRR) – MISSIONE 4 COMPONENTE 2, INVESTIMENTO 1.4 – D.D. 1033 17/06/2022, CN00000023), by the National Science Foundation under Grant CNS-2148251, and by the federal agency and industry partners in the RINGS Program. The material in this paper has been presented in part to the International Conference on Information Fusion (FUSION), Venice, Italy, July 2024. (Corresponding author: Moe Z. Win.)

B. Camajori Tedeschini is with Dipartimento di Elettronica, Informazione e Bioingegneria (DEIB), Politecnico di Milano, 20133 Milan, Italy and with the Wireless Information and Network Sciences Laboratory, Massachusetts Institute of Technology, Cambridge, MA 02139 USA (e-mail: bernardo.camajori@polimi.it, berni97@mit.edu).

M. Brambilla is with Dipartimento di Elettronica, Informazione e Bioingegneria (DEIB), Politecnico di Milano, 20133 Milan, Italy (e-mail: mattia.brambilla@polimi.it).

M. Nicoli is with Dipartimento di Ingegneria Gestionale (DIG), Politecnico di Milano, 20156 Milan, Italy (e-mail: monica.nicoli@polimi.it).

M. Z. Win is with the Laboratory for Information and Decision Systems (LIDS), Massachusetts Institute of Technology, Cambridge, MA 02139 USA (e-mail: moewin@mit.edu).

Color versions of one or more figures in this article are available at https://doi.org/10.1109/YYYY.2024.XXXXXXX.

Digital Object Identifier 10.1109/YYYY.2024.XXXXXXX

I. Introduction

COOPERATIVE POSITIONING (CP) represents a key enabling feature for future automated mobility services [1]–[8]. Modern vehicles leverage an on-board sensor suite including global navigation satellite systems (GNSS), light detection and ranging (LIDAR), radio detection and ranging (RADAR), and stereo cameras to perceive the surrounding environment and perform automated maneuvers [9]–[13]. However, these sensors are not yet able to meet the location accuracy requirements of autonomous driving in harsh environments such as dense urban areas or canyons [14]. Recently, methods have been proposed to combine localization sensors with the latest 5th generation (5G) of cellular communications [15]–[20], depicting a new horizon for mobile connectivity and positioning services [21]–[24]. 5G vehicle-to-everything (V2X) communications are envisioned as crucial in the evolution towards cooperative intelligent transport systems (C-ITS) [25]–[28] by enabling simultaneous communication and localization functionalities [29]–[31]. CP among vehicles, by means of sidelink V2X communications, can be used to overcome the GNSS performance degradation and guarantee a seamless high-accuracy positioning (HAP) service [32]–[36]. The complexity lies in the resource-intensive nature of CP [37], which involves vehicles interacting with each other repeatedly to determine positions. In particular, this process demands significant power and bandwidth [38]–[40], while also facing challenges in scheduling transmissions due to the intricate measurement and information fusion processes [41]–[43]. These factors contribute to larger delays and scalability issues in the optimization problems related to cooperative localization [44], [45].

Conventional approaches, such as implicit cooperative positioning (ICP) [32], [46], integrate GNSS and passive sensor data through Bayesian-filtering, e.g., centralized extended Kalman filter (EKF) or message passing algorithm (MPA), to coherently fuse the measurements at different vehicles. In particular, in ICP, passive objects such as poles, road signs, or traffic lights, are cooperatively detected by multiple vehicles and exploited as noisy anchor points to enhance the vehicle location accuracy. On the one hand, in case of a centralized data-processing architecture gathering all vehicles' measurements, convergence can be achieved, but at the expense of high computational complexity. On the other hand, standard MPA algorithms are optimal only in case of Gaussian-linear models and acyclic factor graphs [47]–[50]. Recent solutions tried to limit the aforementioned problems by either performing fully-distributed particle-based MPA between vehicles [34] or

auto-adjusting the parameters of time-varying models [51]. Still, particle-based solutions require high communication and computational loads which limit their scalability.

In recent years, there has been a growing reliance on machine learning (ML) tools to overcome the limits of conventional approaches' issues, especially regarding scalability and non-linear models [52]–[55]. In particular, the reinforcement learning (RL) paradigm [56]–[58] and its deep learning (DL)-based version [59]–[61] are notably effective in challenging single-agent Markov decision processes (MDPs) where labeled data are scarce or costly. They also excel in environments where the agent's actions directly impact the state of the environment and long-term rewards are prioritized [62]–[64]. Indeed, RL can be seen as a generalization of Bayesian filtering where the agents do not just predict the state through belief computation but also make decisions to maximize long-term rewards, with a policy guiding the decision from state to action. RL is especially well-suited for complex scenarios with extensive state and action spaces, where deep neural networks (DNNs) can efficiently approximate the high-dimensional, nonlinear functions that represent such policies [59], [65]. This approach has been successfully applied in several fields, varying from rate and power control [66]–[69] to dynamic spectrum access in multi-user scenarios and efficient scheduling in vehicular networks [70]–[73].

In case more than one agent acts in the environment and the state is not directly observable, we categorize the framework as multi-agent RL (MARL) [74] and the system as decentralized-partially observable MDP (Dec-POMDP) [75]–[77]. MARL involves independent agents whose actions influence each other's perception of the environment, and it is often solved with the usage of recurrent neural network (RNN), exploiting histories of observations and actions [78]. MARL algorithms, similarly to RL methods, can be divided into two categories: Q-learning and policy optimization (PO) (which comprises actor-critic methods) [79]–[81]. Q-learning focuses on estimating the long-term reward (i.e., Q-value) of each action, selecting the action with the highest Q-value and indirectly (i.e., not explicitly) formulating the policy [82]–[84]. On the other hand, PO directly optimizes the policy through the gradient of the total reward relative to policy parameters [85]–[88]. Multi-agent PO algorithms, especially when combined with a centralized agent learning and a decentralized execution of the policies (e.g., multi-agent proximal policy optimization (MAPPO) [85]), have shown remarkable performances with respect to Q-learning algorithms. This is mainly due to their lack of learning biases and improved sample efficiency thanks to training guidelines like parameter sharing [89]–[91].

Concerning MARL for CP, most of the literature works focus on intelligent unmanned aerial vehicles (UAVs) for target tracking [92] or agent scheduling aided for improving CP [93]. In [92], the objective was to maneuver the agents to track passive objects. However, they considered the state of the agents as known, while the main challenge is to estimate from the measurements their state jointly with target sensing. In [93], the state was estimated with conventional MPA, while the objective was to activate links between agents to optimize cooperative positioning performances (i.e., by improving the positioning error bound (PEB)). The drawbacks of this method are that RL is not actively used for positioning but rather as an assistance method to MPA, and that they consider one agent only, i.e., a single link, at the time instead of exploiting the full potential of multi-agent systems (MASs).

Considering the above review, the fundamental unresolved questions related to CP are as follows: i) how to design a decentralized MARL algorithm that simultaneously performs the computation of the agent state beliefs and the scheduling of the agent-to-agent communication resources, optimizing both location accuracy and communication efficiency; ii) what positioning accuracy improvement can be achieved with respect to state-of-the-art Bayesian approaches like ICP that exploit passive object detections between multiple agents; iii) what are the main trade-offs between positioning improvement and communication resource optimization. Addressing these questions is mandatory for enabling the employment in connected automated vehicles (CAVs), in particular the ability to scale and handle real-word impairments encountered in vehicular scenarios. In this perspective, the goals of this paper are to develop agent-specific policies for selecting communication scheduling between neighbors and, at the same time, learning a representation of the world dynamics that takes advantage of the selected neighbors' measurements. We advocate MARL-based ICP, a new paradigm in which PO RL algorithms are exploited to extend the conventional Bayesian-filtering approach incorporating the actions of the agents. The main idea is to learn from data the relation between agents' states and passive feature observations (see Fig. 1 for a visualization of the cooperative scenario) by selecting only those links to the neighbors that can provide a significant gain to the positioning accuracy. This approach not only improves the localization performance but also enhances the communication efficiency.

In this paper, we propose a new MARL algorithm, namely ICP-MAPPO, expressly designed for performing efficient distributed positioning through the MARL framework and extending the conventional Bayesian-filtering ICP to data-driven approaches. The key contributions are as follows:

- We revise the ICP Bayesian-filtering approach analyzing the current limitations and investigating more general frameworks for solution, drawing from the Dec-POMDP system model and MARL methods.
- We reformulate the ICP methodology into a MARL problem and we design the new ICP-MAPPO solution, relying on dynamic-decentralized-execution and training schemes to simultaneously optimize the Bayesian-filtering and MARL objectives.
- We validate the proposed ICP-MAPPO approach in a realistic C-ITS scenario simulated with CARLA [94], where CAVs perform CP by exploiting passive targets, i.e., poles, distributed over the scene.
- We perform a comparison with the state-of-the-art ICP algorithm [32] and single-agent-based algorithms. We prove the superior performances of the proposed algorithm both in terms of positioning error and communication efficiency.

For easy reference, Table I lists the main abbreviations used throughout the paper.

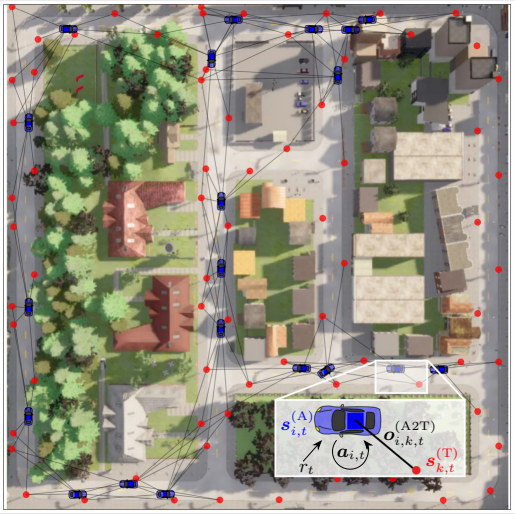


Fig. 1. Top-view of the cooperative scenario with twenty vehicles (blue vehicle icons), detected poles (red circles) and detections (black lines).

TABLE I
LIST OF MAIN ABBREVIATIONS

Acronym	Definition
A2A	Agent-to-agent
A2T	Agent-to-target
Dec-POMDP	Decentralized-partially observable Markov decision process
EKF	Extended Kalman Filter
ICP	Implicit cooperative positioning
LSTM	Long short-term memory
MLP	Multi-layer perceptron
MAPPO	Multi-agent proximal policy optimization
MARL	Multi-agent reinforcement learning
MPA	Message passing algorithm

The rest of this paper is structured as follows. Sec. II describes the system model of cooperative agents. Sec. III reviews the ICP Bayesian-filtering. Sec. IV presents the MARL framework and the proposed ICP-MAPPO execution and training schemes. Sec. V provides information about the simulated scenario and the results. Finally, Sec. VI draws the conclusions.

Notations

Random variables are displayed in sans serif, upright fonts; their realizations in serif, italic fonts. Vectors and matrices are denoted by bold lowercase and uppercase letters, respectively. For example, a random variable and its realization are denoted by x and x; a random vector and its realization are denoted by $\mathbf{x}$ and $\boldsymbol{x}$; a random matrix and its realization are denoted by $\mathbf{X}$ and $\boldsymbol{X}$, respectively. Random sets and their realizations are denoted by up-right sans serif and calligraphic font, respectively. For example, a random set and its realization are denoted by X and $\mathcal{X}$, respectively. The function $p_{\mathsf{x}}(x)$, and simply $p(x)$ when there is no ambiguity, denotes the probability density function (PDF) of x. Notations $\mathbf{X}^\top$, $\mathbf{X}^*$ and $\mathbf{X}^{\mathrm{H}}$ indicate the matrix transposition, conjugation and conjugate transposition. With the notation $\mathsf{x} \sim \mathcal{N}(\mu, \sigma^2)$ we indicate a Gaussian random variable x with mean μ and standard deviation σ, whose PDF is denoted by $\mathcal{N}(x; \mu, \sigma^2)$. We use $\mathbb{E}\{\cdot\}$ and $\mathbb{V}\{\cdot\}$ to denote the expectation and the variance of a random variable, respectively. $\mathbb{R}$ and $\mathbb{C}$ stand for the set of real and complex numbers, respectively. Finally, we define with $\mathrm{blockdiag}(\cdot)$ the block diagonal matrix whose diagonal contains the input blocks matrices.

Notations and definitions of important quantities used in the paper are summarized in Table II.

TABLE II
LIST OF NOTATIONS

Notation	Definition
N, K	Number of agents and passive objects
$\mathbf{s}_{i,t}$, $\mathbf{a}_{i,t}$, $\mathbf{o}_{i,t}$	State, action and observation of agent i at time t
$\mathbf{h}_{i,t}^{\mathrm{b}}$, $\mathbf{h}_{i,t}^{\mathrm{V}}$	History in belief and critic NNs of agent i at time t
$\boldsymbol{\tau}$, $\boldsymbol{\tau}_t$	Trajectory and transition at time t
r_t, R_t	Reward and reward-to-go at time t
$\pi_{\boldsymbol{\theta}}$, $V_{\boldsymbol{\phi}}$, $b_{\boldsymbol{\psi}}$	Actor, critic and beliefs NNs
H, L_τ	Horizon and trajectory length
$A_{i,t}$	Advantage function of agent i at time t
α, β, ϵ	Entropy, reward and clipping coefficients
γ, μ	Discount factor and learning rate

II. System Model

We consider a vehicular network where a set of N vehicles engage in cooperative localization as depicted in Fig. 1. The connectivity graph for vehicle cooperation at time t is $\mathcal{G}_t = (\mathcal{V}, \mathcal{E}_t)$, with $\mathcal{V} = \{1, 2, \ldots, N\}$ representing the set of agents (vehicles), and $\mathcal{E}_t$ the edges (communication links) among them. Each agent $i \in \mathcal{V}$ in the network at time t has a set of neighbors $\mathcal{N}_{i,t}$, and it is assigned a state $\mathbf{s}_{i,t}^{(\mathrm{A})} = \left[\mathbf{u}_{i,t}^{(\mathrm{A})^\top} \mathbf{v}_{i,t}^{(\mathrm{A})^\top}\right]^\top$, where $\mathbf{u}_{i,t}^{(\mathrm{A})}$ and $\mathbf{v}_{i,t}^{(\mathrm{A})}$ are the 2D position and velocity vectors, respectively, defined in a global coordinate system. We denote with $\mathbf{s}_t^{(\mathrm{A})} = \left[\mathbf{s}_{i,t}^{(\mathrm{A})}\right]_{i=1}^{N}$ the aggregate state of all the vehicles at time t. The kinematic state transition of vehicle i at time t is modelled as

$$\mathbf{s}_{i,t}^{(\mathrm{A})} = f^{(\mathrm{A})}\left(\mathbf{s}_{i,t-1}^{(\mathrm{A})}, \mathbf{w}_{i,t-1}^{(\mathrm{A})}\right) \tag{1}$$

where $f^{(\mathrm{A})}(\cdot)$ is a nonlinear function that governs the dynamics of the vehicle's state and $\mathbf{w}_{i,t-1}^{(\mathrm{A})}$ represents the driving noise process, incorporating the uncertainty in motion. The model in (1) is associated to a state-transition PDF denoted as $T\left(\boldsymbol{s}_{i,t}^{(\mathrm{A})} | \boldsymbol{s}_{i,t-1}^{(\mathrm{A})}\right) \triangleq p\left(\boldsymbol{s}_{i,t}^{(\mathrm{A})} | \boldsymbol{s}_{i,t-1}^{(\mathrm{A})}\right)$.

The scenario includes a set $\mathcal{F} = \{1, 2, \ldots, K\}$ of K static and passive objects (or targets, denoted as red circles in Fig. 1) that vehicles can detect and localize by on-board sensors. To facilitate detection by vehicle sensors, specific objects easily identifiable and suitable for the purpose should be used. In

this study, poles have been selected due to their ubiquity (especially in urban areas), ease of recognition, and fixed nature. Each pole k is described by a 2D position state $\mathbf{s}_{k,t}^{(\mathrm{T})}$, which is assumed to be constant over time. As before, we denote with $\mathbf{s}_t^{(\mathrm{T})} = \left[\mathbf{s}_{k,t}^{(\mathrm{T})}\right]_{k\in\mathcal{F}}$ the aggregate state of all passive objects at time t.

Vehicles are equipped with three distinct types of sensors. The first is a GNSS receiver, providing an estimate of the vehicle's state $\mathbf{s}_{i,t}^{(\mathrm{A})}$, modelled as

$$\mathbf{o}_{i,t}^{(\mathrm{GNSS})} = \boldsymbol{H}^{(\mathrm{GNSS})}\,\mathbf{s}_{i,t}^{(\mathrm{A})} + \mathbf{n}_{i,t}^{(\mathrm{GNSS})} \tag{2}$$

where $\mathbf{n}_{i,t}^{(\mathrm{GNSS})} \sim \mathcal{N}\big(\mathbf{0}_{2\times2}, \boldsymbol{R}_{i,t}^{(\mathrm{GNSS})}\big) \in \mathbb{R}^{2\times1}$ is a zero-mean Gaussian noise with covariance $\boldsymbol{R}_{i,t}^{(\mathrm{GNSS})} = {\sigma^{(\mathrm{GNSS})}}^2\boldsymbol{I}_2$, and $\boldsymbol{H}^{(\mathrm{GNSS})} = [\boldsymbol{I}_2\,\mathbf{0}_{2\times2}] \in \mathbb{R}^{2\times4}$. From (2), we define the GNSS likelihood function as $p\big(\boldsymbol{o}_{i,t}^{(\mathrm{GNSS})}|\boldsymbol{s}_{i,t}^{(\mathrm{A})}\big)$, and with $\mathbf{o}_t^{(\mathrm{GNSS})} = \big[\mathbf{o}_{i,t}^{(\mathrm{GNSS})}\big]_{i=1}^{N}$ the aggregate GNSS measurements of all the vehicles at time t.

The second sensor refers to an active sensing technology for sidelink positioning offering relative agent-to-agent (A2A) location measurements for any pair of vehicles $(i,j) \in \mathcal{E}_t$

$$\mathbf{o}_{i,j,t}^{(\mathrm{A2A})} = \boldsymbol{H}^{(\mathrm{A2A})}\big(\mathbf{s}_{i,t}^{(\mathrm{A})} - \mathbf{s}_{j,t}^{(\mathrm{A})}\big) + \mathbf{n}_{i,j,t}^{(\mathrm{A2A})} \tag{3}$$

where $\boldsymbol{H}^{(\mathrm{A2A})} = [\boldsymbol{I}_2\,\mathbf{0}_{2\times2}] \in \mathbb{R}^{2\times4}$ and $\mathbf{n}_{i,j,t}^{(\mathrm{A2A})} \sim \mathcal{N}\big(\mathbf{0}_{2\times2}, \boldsymbol{R}_{i,j,t}^{(\mathrm{A2A})}\big)$ is a zero-mean Gaussian noise with covariance $\boldsymbol{R}_{i,j,t}^{(\mathrm{A2A})} = {\sigma^{(\mathrm{A2A})}}^2\boldsymbol{I}_2$. Additionally, agents have the capability to communicate with their neighbors to share location-related data.

The third sensor type is a passive technology (e.g., RADAR, LIDAR, camera, or any combination), used by vehicle i to detect a set of passive objects $\mathcal{F}_{i,t} \subseteq \mathcal{F}$ in proximity at time t, and produce agent-to-target (A2T) measurements for each object $k \in \mathcal{F}_{i,t}$ as

$$\mathbf{o}_{i,k,t}^{(\mathrm{A2T})} = \boldsymbol{H}^{(\mathrm{A2T})}\mathbf{s}_{i,t}^{(\mathrm{A})} - \mathbf{s}_{k,t}^{(\mathrm{T})} + \mathbf{n}_{i,k,t}^{(\mathrm{A2T})} \tag{4}$$

where $\boldsymbol{H}^{(\mathrm{A2T})} = [\boldsymbol{I}_2\,\mathbf{0}_{2\times2}] \in \mathbb{R}^{2\times4}$ and $\mathbf{n}_{i,k,t}^{(\mathrm{A2T})} \sim \mathcal{N}\big(\mathbf{0}_{2\times2}, \boldsymbol{R}_{i,k,t}^{(\mathrm{A2T})}\big)$ is a zero-mean Gaussian noise with covariance $\boldsymbol{R}_{i,k,t}^{(\mathrm{A2T})} = {\sigma^{(\mathrm{A2T})}}^2\boldsymbol{I}_2$.

We denote with $p\big(\boldsymbol{o}_{i,j,t}^{(\mathrm{A2A})}|\boldsymbol{s}_{i,t}^{(\mathrm{A})}, \boldsymbol{s}_{j,t}^{(\mathrm{A})}\big)$ and $p\big(\boldsymbol{o}_{i,k,t}^{(\mathrm{A2T})}|\boldsymbol{s}_{i,t}^{(\mathrm{A})}, \boldsymbol{s}_{k,t}^{(\mathrm{T})}\big)$ the A2A and A2T likelihoods, respectively. Moreover, we denote with $\mathbf{o}_{i,t} = \big[\mathbf{o}_{i,t}^{(\mathrm{GNSS})\top}\,\mathbf{o}_{i,t}^{(\mathrm{A2A})\top}\,\mathbf{o}_{i,t}^{(\mathrm{A2T})\top}\big]^\top$ the vector of all available measurements of vehicle i at time t, where $\mathbf{o}_{i,t}^{(\mathrm{A2A})} = \big[\mathbf{o}_{i,j,t}^{(\mathrm{A2A})}\big]_{j\in\mathcal{N}_{i,t}}$ and $\mathbf{o}_{i,t}^{(\mathrm{A2T})} = \big[\mathbf{o}_{i,k,t}^{(\mathrm{A2T})}\big]_{k\in\mathcal{F}_{i,t}}$. The total number of unique A2A and A2T measurements at time t is defined as $N_t^{(\mathrm{A2A})} = \sum_{i=1}^{N}|\mathcal{N}_{i,t}|$ and $N_t^{(\mathrm{A2T})} = \sum_{i=1}^{N}|\mathcal{F}_{i,t}|$, respectively. Note that the A2A measurements are not subject to measurement-origin uncertainty, i.e., it is not requested to perform any data association algorithm for pairing them, as the enabling technology is assumed to be active. On the other hand, the A2T observations are unlabelled, as it is unknown which object gives rise to a measurement, being them produced by a passive sensing technology. In this work, we assume that data association has already been performed at the vehicles (using, e.g., methods [53]) and that each A2T measurement has been correctly labeled with the originating target. We consider perfect data association as we aim to derive the best-case performances on the achievable accuracy of data-driven ICP and compare it with conventional Bayesian ICP in the same conditions. Interested readers can refer to [46] for details on data association and their impact on inference algorithms.

III. Bayesian Filtering

In this section, we describe the Bayesian filtering solution, under the ICP framework, and then we highlight its main drawbacks and improvements.

A. Centralized Implicit Cooperative Positioning

The objective of ICP is to concurrently estimate the state of all vehicles and passive objects in the network. To this aim, we define the set of all available measurements at time t as

$$\mathbf{o}_t = \boldsymbol{H}\,\mathbf{s}_t + \mathbf{n}_t \tag{5}$$

where $\mathbf{o}_t = \big[\mathbf{o}_{i,t}\big]_{i\in\mathcal{V}} \in \mathbb{R}^{\left(2N+2N_t^{(\mathrm{A2A})}+2N_t^{(\mathrm{A2T})}\right)\times1}$, $\boldsymbol{H}$ is the matrix modeling the relation to the states, defined as in [32], and $\mathbf{s}_t = \big[\mathbf{s}_t^{(\mathrm{A})\top}\,\mathbf{s}_t^{(\mathrm{T})\top}\big]^\top \in \mathbb{R}^{\left(4N+2K\right)\times1}$ is the aggregated state of the system. $\mathbf{n}_t \sim \mathcal{N}\big(\mathbf{0}, \boldsymbol{R}_t\big)$ is the overall measurement noise with covariance $\boldsymbol{R}_t = \mathrm{blockdiag}\big(\boldsymbol{R}_t^{(\mathrm{GNSS})}, \boldsymbol{R}_t^{(\mathrm{A2A})}, \boldsymbol{R}_t^{(\mathrm{A2T})}\big)$, where $\boldsymbol{R}_t^{(\mathrm{GNSS})} = \mathrm{blockdiag}\big(\boldsymbol{R}_{1,t}^{(\mathrm{GNSS})}, \ldots, \boldsymbol{R}_{N,t}^{(\mathrm{GNSS})}\big)$, $\boldsymbol{R}_t^{(\mathrm{A2A})} = \mathrm{blockdiag}\Big(\boldsymbol{R}_{1,t}^{(\mathrm{A2A})}, \ldots, \boldsymbol{R}_{N_t^{(\mathrm{A2A})},t}^{(\mathrm{A2A})}\Big)$ with the ℓ-th entry given by $\boldsymbol{R}_{\ell,t}^{(\mathrm{A2A})} = \boldsymbol{R}_{i_\ell,j_\ell,t}^{(\mathrm{A2A})}$, and $\boldsymbol{R}_t^{(\mathrm{A2T})} = \mathrm{blockdiag}\Big(\boldsymbol{R}_{1,t}^{(\mathrm{A2T})}, \ldots, \boldsymbol{R}_{N_t^{(\mathrm{A2T})},t}^{(\mathrm{A2T})}\Big)$ with $\boldsymbol{R}_{\ell,t}^{(\mathrm{A2T})} = \boldsymbol{R}_{i_\ell,k_\ell,t}^{(\mathrm{A2T})}$.

The overall state estimate $\widehat{\boldsymbol{s}}_t$ is obtained through the minimum mean square error (MMSE) estimator as

$$\widehat{\boldsymbol{s}}_t = \mathbb{E}\{\mathbf{s}_t|\mathbf{o}_{1:t}\} = \int \boldsymbol{s}_t\,p\big(\boldsymbol{s}_t|\boldsymbol{o}_{1:t}\big)\,d\boldsymbol{s}_t \tag{6}$$

where $\boldsymbol{o}_{1:t} = \big[\boldsymbol{o}_{t'}\big]_{t'=1}^{t}$ is the set of all aggregated measurements up to time t and $p\big(\boldsymbol{s}_t|\boldsymbol{o}_{1:t}\big)$ is the posterior PDF defined as [95]

$$p\big(\boldsymbol{s}_t|\boldsymbol{o}_{1:t}\big) \propto p\big(\boldsymbol{o}_t|\boldsymbol{s}_t\big)\int p\big(\boldsymbol{s}_t|\boldsymbol{s}_{t-1}\big)\,p\big(\boldsymbol{s}_{t-1}|\boldsymbol{o}_{1:t-1}\big)\,d\boldsymbol{s}_{t-1}\,. \tag{7}$$

We denote with $b\big(\boldsymbol{s}_{i,t}|\boldsymbol{o}_{1:t}\big) \triangleq p\big(\boldsymbol{s}_{i,t}|\boldsymbol{o}_{1:t}\big)$ the marginal posterior PDF, also called *belief* of agent i. Given that all the measurements are mutually independent, the likelihood function of $\boldsymbol{s}_t$ is computed as

$$p\big(\boldsymbol{o}_t|\boldsymbol{s}_t\big) = p\big(\boldsymbol{o}_t^{(\mathrm{GNSS})}|\boldsymbol{s}_t^{(\mathrm{A})}\big)\prod_{i=1}^{N}\prod_{j\in\mathcal{N}_{i,t}} p\big(\boldsymbol{o}_{i,j,t}^{(\mathrm{A2A})}|\boldsymbol{s}_{i,t}^{(\mathrm{A})}, \boldsymbol{s}_{j,t}^{(\mathrm{A})}\big) \times \prod_{i=1}^{N}\prod_{k\in\mathcal{F}_{i,t}} p\big(\boldsymbol{o}_{i,k,t}^{(\mathrm{A2T})}|\boldsymbol{s}_{i,t}^{(\mathrm{A})}, \boldsymbol{s}_{k,t}^{(\mathrm{T})}\big)\,. \tag{8}$$

For notation purposes, we will denote the likelihood function also as $O\big(\boldsymbol{o}_t|\boldsymbol{s}_t\big) \triangleq p\big(\boldsymbol{o}_t|\boldsymbol{s}_t\big)$. In case the dynamic and measurements models in (1) and (5), respectively, are linear and with a Gaussian noise, the state estimate in (6) reduces to a Kalman filter (KF) as described in [32], [46], with efficient resolution in matrix form.

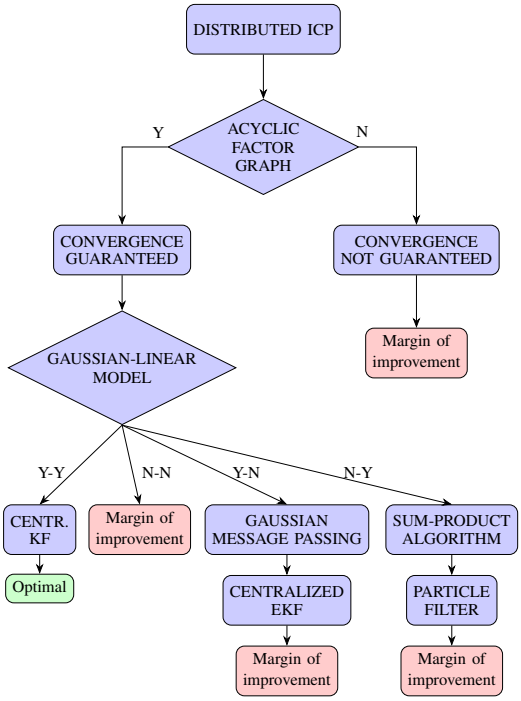


Fig. 2. Convergence diagram flow of ICP methods.

B. Limitations of Bayesian ICP Methods

The centralized ICP approach is impractical for extensive networks due to two major limitations: the single central computing unit representing a point of failure, and its computational complexity growing cubically with the number of vehicles and passive objects [32]. To overcome such limitations, distributed or consensus-based ICP algorithms have been studied in the past [34]. However, their convergence to the centralized solution is guaranteed only in acyclic (i.e., tree-structured) factor graphs. Moreover, even in case of convergence, the result would be optimal only with Gaussian and linear models (i.e., in (1) and (5)). In all the other cases, optimality is not guaranteed. In Fig. 2 we summarized all cases and highlighted those where improvements could be provided by new data-driven designs. We would like to point out that in real-world dynamics, the factor graph is usually not acyclic and the models are typically neither Gaussian nor linear.

The aim of this paper is to address the gap by proposing a new decentralized data-driven solution to the ICP problem suited for non-linear non-Gaussian models, overcoming the limits of parametric Bayesian implementations based on EKF or particle filter (PF) highlighted in Fig. 2. The proposed distributed method also incorporates a data-driven optimization of the cooperation graph by making the agents actively and opportunistically select the cooperating neighbors so as to minimize the communication signaling. In particular, to address the limitations of conventional ICP solutions, we adopt neural networks (NNs)-based models, which are able to learn whatever non-linear function is hidden in the data thanks to the universal approximation theorem. Specifically, a RNN learns the non-linear motion and measurement models, whereas a multi-layer perceptron (MLP) learns the non-linear relation between link activation and state estimate. Moreover, NNs have proven effective even in non-Gaussian settings [53], given their ability to model complex probability distributions without assuming any specific form. The centralized ICP method reviewed in this section will be used as a benchmark to assess the proposed method.

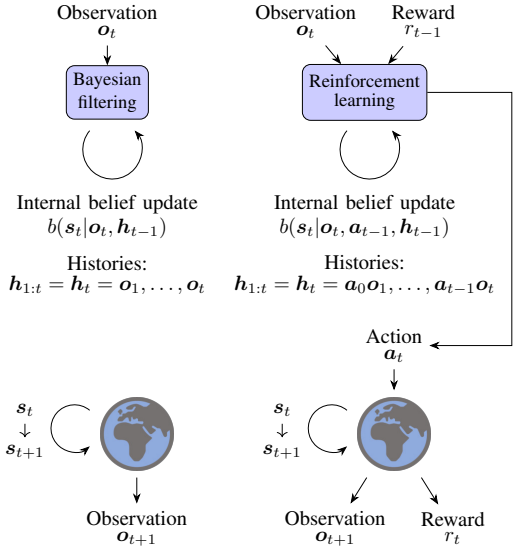
IV. MARL for Cooperative Positioning

In this section, we first introduce the MARL framework (Sec. IV-A) that will be used later for the design of the ICP-MAPPO solution (Sec. IV-B). The ICP-MAPPO execution and training schemes are reported in Sec. IV-C and IV-D, respectively.

A. MARL Framework

We model the cooperative MAS as a finite-horizon Dec-POMDP [75] defined by the tuple $\langle \mathcal{V}, \mathcal{S}, \mathcal{A}, T_0, T, \mathcal{O}, O, R, \gamma, H \rangle$. We recall that the set $\mathcal{V}$ refers to the cooperative agents, while the sets $\mathcal{S}$ and $\mathcal{A}$ denote the state and action spaces, respectively. T_0 is the initial state distribution at time $t = 0$, while $T(\boldsymbol{s}_t|\boldsymbol{s}_{t-1}, \boldsymbol{a}_t) \triangleq p(\boldsymbol{s}_t|\boldsymbol{s}_{t-1}, \boldsymbol{a}_t)$ is the state transition PDF that, differently from the Bayesian-filtering system model in Sec. II, now also includes the joint action realization $\boldsymbol{a}_t = \left[\boldsymbol{a}_{i,t}\right]_{i\in\mathcal{V}} \in \mathcal{A}$ and the joint state $\boldsymbol{s}_t \in \mathcal{S}$. At each time t, the agents receive the joint observations or measurements $\boldsymbol{o}_t \in \mathcal{O}$ which are sampled from the distribution $O(\boldsymbol{o}_t|\boldsymbol{a}_{t-1}, \boldsymbol{s}_t) \triangleq p(\boldsymbol{o}_t|\boldsymbol{a}_{t-1}, \boldsymbol{s}_t)$. Note that here, (8) is also function of the previous joint action of the agents $\boldsymbol{a}_{t-1}$, thus generalizing the concept of Bayesian-filtering. $R(\mathbf{s}_t, \mathbf{a}_t) = \mathsf{r}_t \in \mathbb{R}$ denotes the instantaneous shared reward at time t obtained from the reward function R, while $\gamma \in [0, 1)$ and H are the discount factor and time horizon of each episode, respectively.

Since the states and rewards are not directly observable by the agents (partially observable MDP), each agent i keeps track of the so-called *histories* defined as $\boldsymbol{h}_{i,1:t} = \boldsymbol{h}_{i,t} = \left[(\boldsymbol{a}_{i,t'-1}, \boldsymbol{o}_{i,t'})\right]_{t'=1}^{t}$. Note that the histories are a generalization of the aggregated measurements up to time t in (6). Given a new observation $\boldsymbol{o}_{i,t}$, the state estimates $\widehat{\boldsymbol{s}}_{i,t}$ are produced by MMSE criterion from the belief PDF $b_{\boldsymbol{\psi}}(\boldsymbol{s}_{i,t}|\boldsymbol{o}_{i,t}, \boldsymbol{a}_{i,t-1}, \boldsymbol{h}_{i,t-1}) = p_{\boldsymbol{\psi}}(\boldsymbol{s}_{i,t}|\boldsymbol{o}_{i,t}, \boldsymbol{a}_{i,t-1}, \boldsymbol{h}_{i,t-1})$ parameterized by $\boldsymbol{\psi}$. Moreover, agents adopt a policy $\pi_{\boldsymbol{\theta}}(\boldsymbol{a}_{i,t}|\boldsymbol{h}_{i,t}) = p_{\boldsymbol{\theta}}(\boldsymbol{a}_{i,t}|\boldsymbol{h}_{i,t})$ defined by $\boldsymbol{\theta}$ to obtain the action $\boldsymbol{a}_{i,t}$ from histories $\boldsymbol{h}_{i,t}$. A full comparison between Bayesian filtering and RL (i.e., its generalized version) can be found in Fig. 3. By defining the *reward-to-go* $\mathsf{R}_t = \sum_{t'=t}^{H-1} \gamma^{t'-t}\, \mathsf{r}_{t'}$ as the cumulative discounted reward from time t to the end of the episode, the objective of the MARL problem is to maximize,



(a) Bayesian filtering (b) Reinforcement learning

Fig. 3. Comparison between Bayesian filtering and RL.

over the policy π, the expected cumulative discounted reward from the beginning of the episode

$$\max_{\pi} J(\pi) = \max_{\pi} \mathbb{E}\{\mathsf{R}_0\} \tag{9}$$

which usually translates into optimizing the parameters of the policy as $\boldsymbol{\theta}^* = \text{argmax}_{\boldsymbol{\theta}} J(\pi_{\boldsymbol{\theta}})$, with $\pi^*_{\boldsymbol{\theta}}$ representing the optimal policy.

B. MARL Solution to the ICP Problem

In standard Dec-POMDP, each agent only knows its local actions and observations, thus resulting in possible non-stationary learning problems from each agent's perspective [96]. By training independent learners to optimize the team reward (i.e., concurrent learning), we induce a change in the dynamics of the environment as teammates continuously adapt their behaviours throughout learning. On the contrary, whenever a fully connected graph with communications is present, the Dec-POMDP collapses to a centralized POMDP, resulting in higher complexity and communication inefficiencies [89], [97], exactly as in centralized ICP. To solve the issues of independent and centralized training-execution, the state-of-the-art works exploit the so called centralized-training and decentralized-execution paradigm. This framework permits to learn the policies in a centralized way and then deploy them in the network graph for decentralized execution [85], [87], [98].

While this approach solves the problem in standard MARL algorithms, in the context of ICP, having access to the neighbors' measurements would allow the positioning accuracy to be significantly improved. Indeed, the objective of ICP is to minimize over the belief b the error on the state estimate as

$$\min_{b} J(b) = \min_{b} \mathbb{E}\left\{ \sum_{t} \left\| \mathbf{s}_t - \widehat{\boldsymbol{s}}_t \right\|_2^2 \right\}. \tag{10}$$

Therefore, we here propose to define as actions the agent's selection of the communication links to the neighbors to cooperate with. This allows to optimize the communication efficiency with respect to the centralized solution. Formally, we define the following Dec-POMDP:

1) Agents: The agent is identified by vehicle $i \in \mathcal{V}$ that composes the connected network.

2) Actions: The action of agent i at time t is $\mathbf{a}_{i,t} = \left[\mathsf{a}_{i,j,t}\right]_{j=1}^{N}$, where $\mathsf{a}_{i,j,t} \in \{0, 1\}$ represents the Boolean decision of agent i to communicate with agent j.

3) States: Only the states of the vehicles $\mathbf{s}_t^{(\mathrm{A})}$ are considered, while the target states $\mathbf{s}_t^{(\mathrm{T})}$ are implicitly learned by the NNs through the hidden features. Indeed, the system does not output or keep track of the states of the targets, since they are not needed as in the ICP Bayesian filtering formulation. In other words, the ICP-MAPPO model just outputs the predicted states of the agents, while the targets' states are contained in the hidden space, i.e., histories. Therefore, from now on, we indicate with $\mathbf{s}_t$ the state of the agents $\mathbf{s}_t^{(\mathrm{A})}$.

4) Observations: GNSS, A2A, and A2T measurements described in Sec. II are the observations used in the Dec-POMDP modeling, as they are the only output returned by the world at inference time.

During the centralized training, the agents learn the relation between histories-actions, i.e., policy optimization, and histories-states, i.e., belief optimization, while having access to the full observable state $\mathbf{s}_t$ and measurements $\mathbf{o}_t$. Conversely, during the decentralized execution, the agents decide how to modify the network graph to achieve the best trade-off between positioning accuracy and communication efficiency. We call this approach *centralized-training and dynamic-decentralized-execution*, as during execution, according to the agents' actions, the coordination graph may vary, passing from fully-connected to fully-decentralized according to the agent's decisions.

C. ICP-MAPPO Execution Scheme

For the belief and action prediction, we propose to employ a long short-term memory (LSTM) and MLP, respectively. In Fig. 4, we show a compact representation of the execution within each agent. In particular, the NN functions are defined as

$$\widehat{\boldsymbol{s}}_{i,t}, \boldsymbol{h}_{i,t}^{\mathrm{b}} = b_{\boldsymbol{\psi}}(\boldsymbol{s}_{i,t} | \boldsymbol{o}_{i,t}, \bar{\boldsymbol{a}}_{i,t-1}, \bar{\boldsymbol{h}}_{i,t-1}^{\mathrm{b}}) \tag{11}$$

$$\boldsymbol{a}_{i,t} \sim \pi_{\boldsymbol{\theta}}(\boldsymbol{a}_{i,t} | \boldsymbol{h}_{i,t}^{\mathrm{b}}) \tag{12}$$

where $\boldsymbol{o}_{i,t}$ is the ordered vector of all measurements of agent i at time t defined as in Sec. II, $\bar{\boldsymbol{a}}_{i,t} = \left[\bar{a}_{i,j,t}\right]_{j=1}^{N}$ includes the sampled actions from the policy distribution adjusted with the feasibility of the network connectivity as

$$\bar{a}_{i,j,t} = \begin{cases} a_{i,j,t} & \text{if} \quad j \in \mathcal{N}_{i,t} \\ -1 & \text{otherwise} \end{cases} \tag{13}$$

and $\bar{\boldsymbol{h}}^{\mathrm{b}}_{i,t}$ are the hidden features of the belief LSTM which contain a compressed representation of the histories of agent i and all selected neighbors at the previous timestep

$$\bar{\boldsymbol{h}}^{\mathrm{b}}_{i,t} = \frac{\boldsymbol{h}^{\mathrm{b}}_{i,t} + \sum_{j\in\mathcal{V}} \boldsymbol{h}^{\mathrm{b}}_{j,t}\, \mathbb{1}(\bar{a}_{i,j,t} == 1)}{1 + \sum_{j\in\mathcal{V}} \mathbb{1}(\bar{a}_{i,j,t} == 1)} \tag{14}$$

where $\mathbb{1}(\cdot)$ is the indicator function that returns 1 if the condition is true and 0 otherwise. We point out that the hidden features $\boldsymbol{h}^{\mathrm{b}}_{i,t}$ include not only past actions and measurements but also the implicit state estimates of the targets $\widehat{\boldsymbol{s}}^{(\mathrm{T})}_t$, which are never explicitly predicted by the system for output space complexity reduction.

The key rationale behind the execution scheme is the following. We employ the average operation in (14) to avoid gradient divergence over the timesteps. We would like to notice that the action decision at time t in (12) is mainly based on the previous timestep information $\bar{\boldsymbol{h}}^{\mathrm{b}}_{i,t-1}$, as there is no way for agent i to know a priori the measurements of its neighbors $\boldsymbol{h}^{\mathrm{b}}_{j,t}, \forall j \in \mathcal{V}$, in order to activate the communications between them. Moreover, the actions $\bar{\boldsymbol{a}}_{i,t}$ are given as input to the belief LSTM for two main reasons. First, the information about which agents were selected for measurements fusion is necessary to coherently predict the state estimate. Second, the negative action values imposed by the lack of possible connectivity permit each agent to implicitly learn its index or identification. In this way, the scalable and efficient parameter sharing approach for training one single NN [89], instead of agent-specific NNs, can be combined with agent differentiation by index learning.

D. ICP-MAPPO Training Scheme

For the reward definition, we propose to use a function that, looking at the future timestep, rewards the actions that gave a predetermined improvement β on the positioning accuracy. In other words, each agent i tries to answer the following question: if I had chosen agent j' instead of agent j, would the performances have improved? This is formalized as

$$r_t = \begin{cases} -1 & \text{if} \quad \left\|\boldsymbol{s}_t - \widehat{\boldsymbol{s}}_t\right\|_2^2 - \left\|\boldsymbol{s}_{t+1} - \widehat{\boldsymbol{s}}_{t+1}\right\|_2^2 \leq -\beta \\ +1 & \text{if} \quad \left\|\boldsymbol{s}_t - \widehat{\boldsymbol{s}}_t\right\|_2^2 - \left\|\boldsymbol{s}_{t+1} - \widehat{\boldsymbol{s}}_{t+1}\right\|_2^2 > \beta \\ +2 & \text{if} \quad -\beta < \left\|\boldsymbol{s}_t - \widehat{\boldsymbol{s}}_t\right\|_2^2 - \left\|\boldsymbol{s}_{t+1} - \widehat{\boldsymbol{s}}_{t+1}\right\|_2^2 \leq \beta \end{cases} \tag{15}$$

where β is a hyper-parameter which regulates the improvement step. At the beginning of the learning, if the improvement is negative and bigger than β, the reward is negative as the actual agent selection worsen the positioning accuracy. On the other hand, if the improvement is positive and greater than β, the reward is +1. Finally, when the learning starts converging and the improvements become smaller, we introduce a long-term reward of +2. Note that, while in conventional Dec-POMDPs the reward directly depends on the actions, in the proposed system the effect of the actions' choice can be assessed only at the next timestamp and only by measuring the positioning error.

Regarding the type of MARL algorithm, we opted for PO over Q-learning-based methods. This is because Q-learning algorithms combined with DL have no guarantees of convergence and retain a lot of bias (i.e., inaccurate state-action value or Q-value). On the contrary, PO algorithms retain very low bias since they directly optimize the objective function in (9) and have been proven to outperform Q-learning methods in MARL systems [87]. Moreover, while off-policy RL algorithms use historical data to learn the policy, in the context of CP, where state estimation is crucial, it is essential to utilize the most up-to-date policy available since the action sampling (i.e., radio link activation) directly influences the positioning performances. Despite PO algorithms having an intrinsic high variance, i.e., they require a lot of samples to converge, this can be mitigated by the learning of the value function, either $V^{\pi}(\boldsymbol{s}_t)$ or $Q^{\pi}(\boldsymbol{s}_t, \boldsymbol{a}_t)$, which estimates the long-term reward given a specific state or state-action pair, respectively. Specifically, we employ the state value function defined as

$$\begin{aligned} V^{\pi}(\boldsymbol{s}_t) &= \mathbb{E}\{\mathsf{R}_t | \mathsf{s}_t = \boldsymbol{s}_t\} \\ &= \mathbb{E}_{\mathbf{a}_t\sim\pi, \mathbf{s}_{t+1}\sim T}\Big\{ R(\boldsymbol{s}_t, \mathbf{a}_t) + \gamma V^{\pi}(\mathbf{s}_{t+1}) \Big\}. \end{aligned} \tag{16}$$

Usually, $V^{\pi}(\boldsymbol{s}_t)$ cannot be directly computed due to the curse of dimensionality and thus it is estimated by an additional NN $\widehat{V}_{\boldsymbol{\phi}}(\boldsymbol{s}_t) = V_{\boldsymbol{\phi}}(\boldsymbol{s}_t)$, with parameters $\boldsymbol{\phi}$ which are only employed during training.

In standard single-agent RL frameworks, the policy optimization problem is usually defined with the introduction of trajectories $\boldsymbol{\tau} = (\mathbf{s}_0, \mathbf{a}_0, \ldots, \mathbf{s}_H, \mathbf{a}_H)$ by maximizing

$$\begin{aligned} J(\pi_{\boldsymbol{\theta}}) &= \mathbb{E}_{\boldsymbol{\tau}\sim p(\boldsymbol{\tau}|\pi_{\boldsymbol{\theta}})}\Big\{ \widetilde{R}(\boldsymbol{\tau}) \Big\} \\ &= \sum_{t=0}^{H} \mathbb{E}_{\mathbf{s}_t\sim p(\boldsymbol{s}_t|\pi_{\boldsymbol{\theta}}), \mathbf{a}_t\sim\pi_{\boldsymbol{\theta}}(\boldsymbol{a}_t|\boldsymbol{s}_t)}\Big\{ \gamma^t\, R(\mathbf{s}_t, \mathbf{a}_t) \Big\} \end{aligned} \tag{17}$$

where $\widetilde{R}(\boldsymbol{\tau}) = \mathsf{R}_0$ is the reward of trajectory $\boldsymbol{\tau}$, $p(\boldsymbol{\tau}|\pi) = T_0 \prod_{t=0}^{H-1} T(\boldsymbol{s}_{t+1}|\boldsymbol{s}_t, \boldsymbol{a}_t)\, \pi(\boldsymbol{a}_t|\boldsymbol{s}_t)$ is the PDF of an H-step trajectory, and $p(\boldsymbol{s}_t|\pi)$ is the state marginal of the trajectory distribution induced by policy π. Standard REINFORCE PO algorithms [99] update the policy parameters in (17) in the direction of $\nabla_{\boldsymbol{\theta}} J(\pi_{\boldsymbol{\theta}})$, which can be written as (see Appendix A)

$$\nabla_{\boldsymbol{\theta}} J(\pi_{\boldsymbol{\theta}}) = \mathbb{E}_{(\mathbf{s}_t, \mathbf{a}_t)\sim p(\boldsymbol{s}_t, \boldsymbol{a}_t|\pi_{\boldsymbol{\theta}})}\Big\{ \sum_{t=0}^{H-1} \nabla_{\boldsymbol{\theta}} \log\big(\pi_{\boldsymbol{\theta}}(\boldsymbol{a}_t|\boldsymbol{s}_t)\big) A_t \Big\} \tag{18}$$

where $p(\boldsymbol{s}_t, \boldsymbol{a}_t|\pi_{\boldsymbol{\theta}})$ is the state-action marginal of the trajectory distribution induced by policy π and $A_t = A_t(\mathbf{s}_t, \mathbf{a}_t)$ is the generic advantage function at time t [100], which quantifies the convenience of taking a specific action $\mathbf{a}_t$ in a given state $\mathbf{s}_t$, compared to the average action's expected return for that state.

During successive optimization steps of (18) within the same trajectory, where the objective is to maintain proximity between new and old policy parameters, even minor variations in the NN weights can lead to significant differences in performance. Consequently, a single unfavorable optimization step can drastically deteriorate the policy's effectiveness. Recent state-of-the-art methods, e.g., trust region policy optimization (TRPO) [101] and proximal policy optimization (PPO) [102], tried to solve this problem by taking the largest gradient

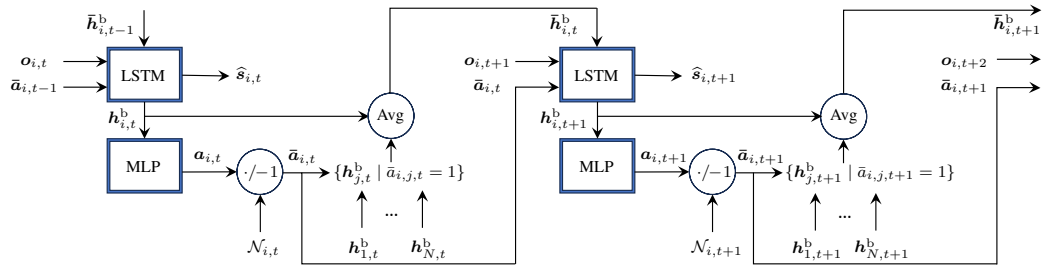


Fig. 4. Dynamic-decentralized-execution scheme of the proposed ICP-MAPPO algorithm.

step size possible to improve performance, while maintaining constraints on how close the new and old policies (i.e., $\pi_{\boldsymbol{\theta}_{\text{old}}}$ at previous train step) are allowed to be. The constraint in TRPO is enforced by Kullback–Leibler (KL) divergence and the parameters are obtained by maximizing the *surrogate* objective function as

$$\begin{aligned}\boldsymbol{\theta} = \underset{\boldsymbol{\theta}}{\operatorname{argmax}}\ & \mathbb{E}_{(\mathbf{s}_t,\mathbf{a}_t)\sim p(\boldsymbol{s}_t,\boldsymbol{a}_t|\pi_{\boldsymbol{\theta}})}\Big\{\frac{\pi_{\boldsymbol{\theta}}(\boldsymbol{a}_t|\boldsymbol{s}_t)}{\pi_{\boldsymbol{\theta}_{\text{old}}}(\boldsymbol{a}_t|\boldsymbol{s}_t)}A_t(\mathbf{s}_t,\mathbf{a}_t)\Big\} \\ \text{s.t. } & \mathbb{E}_{\mathbf{s}_t\sim p(\boldsymbol{s}_t|\pi_{\boldsymbol{\theta}})}\Big\{D_{\text{KL}}\big(\pi_{\boldsymbol{\theta}}(\cdot|\boldsymbol{s}_t)\big\|\pi_{\boldsymbol{\theta}_{\text{old}}}(\cdot|\boldsymbol{s}_t)\big)\Big\} \leq \epsilon\end{aligned} \tag{19}$$

which resulted in a second-order optimization method. On the contrary, PPO and its recent multi-agent version MAPPO use a much more efficient first-order method that exploits clipping to remove incentives for the new policy to get far from the old policy.

In this paper, we adopt three loss functions: $L(\boldsymbol{\phi})$ and $L(\boldsymbol{\theta})$ derived from the MAPPO scheme to train the state-value and policy NNs, respectively, and $L(\boldsymbol{\psi})$ to train the belief NN. $\pi_{\boldsymbol{\theta}}$ and $V_{\boldsymbol{\phi}}$ are called *actor* and *critic*, respectively, since the actor is responsible for selecting actions based on the current policy, and the critic evaluates the quality of these actions by estimating the value function. In Dec-POMDP, the critic $V_{\boldsymbol{\phi}}$ is also dependent on the history of action-observation pairs and thus it is usually modelled with a RNN as

$$\widehat{V}_{\boldsymbol{\phi}}(\boldsymbol{s}_{i,t},\boldsymbol{h}^{\text{V}}_{i,t-1}),\boldsymbol{h}^{\text{V}}_{i,t} = V_{\boldsymbol{\phi}}(\boldsymbol{s}_{i,t},\boldsymbol{h}^{\text{V}}_{i,t-1}) \tag{20}$$

where $\boldsymbol{h}^{\text{V}}_{i,t}$ are the hidden features of the critic. Given a trajectory of length L_τ (subset of the horizon length H), $L(\boldsymbol{\phi})$ is defined to perform regression on the rewards-to-go as

$$\begin{aligned}L(\boldsymbol{\phi}) = \frac{1}{NL_\tau}\sum_{i\in\mathcal{V}}\sum_{\ell=1}^{L_\tau}\Bigg\{\max\Bigg(&\Big[\widehat{V}_{\boldsymbol{\phi}}(\boldsymbol{s}_{i,\ell},\boldsymbol{h}^{\text{V}}_{i,\ell}) - R_\ell\Big]^2, \\ &\Big[\text{clip}\big(\widehat{V}_{\boldsymbol{\phi}}(\boldsymbol{s}_{i,\ell},\boldsymbol{h}^{\text{V}}_{i,\ell-1}),\widehat{V}_{\boldsymbol{\phi}_{\text{old}}}(\boldsymbol{s}_{i,\ell},\boldsymbol{h}^{\text{V}}_{i,\ell-1}),\epsilon\big) - R_\ell\Big]^2\Bigg)\Bigg\}\end{aligned} \tag{21}$$

where the clip prevents the value function from radically changing between iterations, and it is defined as

$$\text{clip}\big(A,B,\epsilon\big) = \min\Big(\max\big(A,B-\epsilon\big),B+\epsilon\Big) \tag{22}$$

where ϵ is the clip coefficient.

The actor $\pi_{\boldsymbol{\theta}}$ is also trained with clipping to discard the KL constraint in (19) by minimizing

$$\begin{aligned}L(\boldsymbol{\theta}) = -\frac{1}{NL_\tau}\sum_{i\in\mathcal{V}}\sum_{\ell=1}^{L_\tau}\Bigg\{\min\Bigg(&\frac{\pi_{\boldsymbol{\theta}}(\boldsymbol{a}_{i,\ell}|\boldsymbol{h}^{\text{b}}_{i,\ell})}{\pi_{\boldsymbol{\theta}_{\text{old}}}(\boldsymbol{a}_{i,\ell}|\boldsymbol{h}^{\text{b}}_{i,\ell})}\widehat{A}_{i,\ell}, \\ &\text{clip}\Big(\frac{\pi_{\boldsymbol{\theta}}(\boldsymbol{a}_{i,\ell}|\boldsymbol{h}^{\text{b}}_{i,\ell})}{\pi_{\boldsymbol{\theta}_{\text{old}}}(\boldsymbol{a}_{i,\ell}|\boldsymbol{h}^{\text{b}}_{i,\ell})},1,\epsilon\Big)\widehat{A}_{i,\ell}\Bigg) + \alpha S\big(\pi_{\boldsymbol{\theta}}(\cdot|\boldsymbol{h}^{\text{b}}_{i,\ell})\big)\Bigg\}\end{aligned} \tag{23}$$

where $\widehat{A}_{i,\ell} = R_\ell - \widehat{V}_{\boldsymbol{\phi}_{\text{old}}}(\boldsymbol{s}_{i,\ell},\boldsymbol{h}^{\text{V}}_{i,\ell-1})$ is the advantage function estimate, $S(p_{\text{x}}) = \mathbb{E}_{\text{x}\sim p_{\text{x}}}\Big\{-\log\big(p_{\text{x}}(x)\big)\Big\}$ is the entropy function which encourages the exploration by inducing stochastic policies, and α is the temperature hyper-parameter which balances the trade-off between exploiting the best actions and exploring new actions. Finally, the beliefs $b_{\boldsymbol{\psi}}$ adopt a MSE loss function to minimize $J(b)$ in (10) as

$$L(\boldsymbol{\psi}) = \frac{1}{NL_\tau}\sum_{i\in\mathcal{V}}\sum_{\ell=1}^{L_\tau}\big\|\widehat{\boldsymbol{s}}_{i,t} - \boldsymbol{s}_{i,t}\big\|_2^2. \tag{24}$$

All the NNs are trained with maximum likelihood estimation (MLE) criterion. However, while $b_{\boldsymbol{\psi}}(\boldsymbol{s}_{i,t}|\boldsymbol{o}_{i,t},\bar{\boldsymbol{a}}_{i,t-1},\bar{\boldsymbol{h}}^{\text{b}}_{i,t-1})$ directly outputs $\widehat{\boldsymbol{s}}_{i,t}$, $\pi_{\boldsymbol{\theta}}(\boldsymbol{a}_{i,t}|\boldsymbol{h}^{\text{b}}_{i,t})$ predicts the probability of communication among agents through sigmoid activation functions, from which actions $\boldsymbol{a}_{i,t}$ are sampled. The full training algorithm can be found in Algorithm 1, where we defined a transition as $\boldsymbol{\tau}_t = (\boldsymbol{s}_t,\boldsymbol{o}_t,\boldsymbol{h}^{\text{b}}_t,\bar{\boldsymbol{h}}^{\text{b}}_t,\boldsymbol{h}^{\text{V}}_t,\boldsymbol{a}_t,\bar{\boldsymbol{a}}_t,r_t,\boldsymbol{s}_{t+1},\boldsymbol{o}_{t+1},\widehat{\boldsymbol{s}}_{t+1})$. Since our approach combines the usage of passive targets to improve the position estimate and MAPPO MARL to perform an efficient agent selection, we call this algorithm ICP-MAPPO.

The main characteristics of ICP-MAPPO are the following. ICP-MAPPO is a low-bias on-policy algorithm since the data used to train the agents are collected from the policy currently being learned or improved. For value regression, we adopted a centralized value function that takes as input extra global information (i.e., the states) not present in the agent's local observation to accurately estimate the values state. The beliefs are computed as in model-based value estimation (MBVE) RL [103], [104], leveraging the learned dynamics to predict the state estimate. This additionally reduces the variance of the PO method without introducing additional biases by avoiding performing rollouts [105]. Finally, as opposed to conventional MARL algorithms, the rewards are not directly dependent on the action, but only implicitly through the beliefs of the next

Algorithm 1 Implicit Cooperative Positioning Multi-Agent Proximal Policy Optimization (ICP-MAPPO)

Input: actor, critic and belief parameters $\boldsymbol{\theta} = \boldsymbol{\theta}_{\text{old}}$, $\boldsymbol{\phi} = \boldsymbol{\phi}_{\text{old}}$, and $\boldsymbol{\psi}$.
for each training step $n = 1, 2, \ldots, N_{\text{step}}$ **do**
  Initialize empty batch $\mathcal{B} = \{\}$ and trajectory $\boldsymbol{\tau} = []$
  Initialize histories $\boldsymbol{h}^{\text{V}}_{i,0}$ and $\boldsymbol{h}^{\text{b}}_{i,1}$ for critic and beliefs
  Initialize state estimate $\widehat{\boldsymbol{s}}_0$
  for $t = 1, 2, \ldots, H$ **do**
    for all agents $i \in \mathcal{V}$ **in parallel do**
      Sample action $\boldsymbol{a}_{i,t} \sim \pi_{\boldsymbol{\theta}_{\text{old}}}(\boldsymbol{a}_{i,t}|\boldsymbol{h}^{\text{b}}_{i,t})$
      Send $\boldsymbol{h}^{\text{b}}_{i,t}$ and receive $\boldsymbol{h}^{\text{b}}_{j,t}$ $\forall j \in \mathcal{N}_{i,t}$
      Get value estimate $\widehat{V}_{\boldsymbol{\phi}_{\text{old}}}(\boldsymbol{s}_{i,t}, \boldsymbol{h}^{\text{V}}_{i,t-1})$ with (20)
      Compute $\bar{\boldsymbol{a}}_{i,t}$ and $\bar{\boldsymbol{h}}^{\text{b}}_{i,t}$ with (13) and (14)
      Observe $\boldsymbol{s}_{i,t+1}, \boldsymbol{o}_{i,t+1}$
      Get state estimate $\widehat{\boldsymbol{s}}_{i,t+1}$ with (11)
    end for
    Observe r_t and store $\boldsymbol{\tau}_t$ in $\boldsymbol{\tau}$
  end for
  Compute advantage estimate $\widehat{A}_{i,t}$ $\forall\, t$ and agent i on $\boldsymbol{\tau}$
  Compute reward-to-go R_t for each $\forall\, t$ on $\boldsymbol{\tau}$
  Split trajectory τ into chunks of length L_τ
  for each $\ell = 0, 1, \ldots, \lfloor H/L_\tau \rfloor$ **do**
    $\mathcal{B} = \mathcal{B} \cup \{\boldsymbol{\tau}_t, \widehat{A}_t, R_t\}_{t=\ell}^{\ell+L_\tau}$
    Adam update of $\boldsymbol{\psi}$ on $L(\boldsymbol{\psi})$ with data $\{\boldsymbol{\tau}_t\}_{t=\ell}^{\ell+L_\tau}$
  end for
  for each mini-batch **do**
    Sample $\{\boldsymbol{\tau}_\ell\}_{\ell=1}^{L_\tau} \sim \mathcal{B}$
    Adam update of $\boldsymbol{\theta}$ on $L(\boldsymbol{\theta})$ with data $\{\boldsymbol{\tau}_\ell\}_{\ell=1}^{L_\tau}$
    Adam update of $\boldsymbol{\phi}$ on $L(\boldsymbol{\phi})$ with data $\{\boldsymbol{\tau}_\ell\}_{\ell=1}^{L_\tau}$
  end for
  $\boldsymbol{\theta}_{\text{old}} \leftarrow \boldsymbol{\theta}$, $\boldsymbol{\phi}_{\text{old}} \leftarrow \boldsymbol{\phi}$
end for

timestep. This permits to effectively decouple the evaluation of actions based on the improvement of state predictions rather than immediate outcomes, focusing on long-term strategic benefits rather than short-term gains.

V. Simulation Experiments

In this section, we first introduce the scenario and the training procedures, and then we describe the baseline methods, and the main simulation results.

A. Simulation Setup

To evaluate the performances of the proposed ICP-MAPPO algorithm, we simulate a C-ITS scenario with the CARLA software [94] in an urban map (i.e., *Town02* of CARLA) that spans an area of $200 \times 200\,\text{m}^2$. Fig. 1 shows a bird-eye-view representation of the map. CARLA takes into account inter-vehicle dynamics, such as acceleration, braking behavior, and collision physics, as well as communication constraints given by the environment. Within the area, 20 CAVs move for 1500 timesteps sampled every 0.2 s, while 72 fixed objects (poles) are detected by the vehicles if in line-of-sight (LoS) and within a sensing range of 70 m. The same coverage area applies to A2A measurements. For the communications, we only consider the direct LoS path, as if the vehicles were equipped with LIDAR technology that could be blocked by obstacles such as buildings or other vehicles. The absolute driving speed adopted in the testing scenario ranges from 0 to about 60 km/h, with a mean and standard deviation speed of 0.2 km/h and 14 km/h, respectively. We point out that the motion models of the vehicles are not linear and that the factor graph to solve the distributed ICP method contains cycles. For the GNSS, A2A, and A2T observations, measurement errors are simulated as additive independent Gaussian noises with standard deviations of 2 m each.

For the training and testing of the ICP-MAPPO algorithm, we create two different simulations each composed of $H = 1500$ timesteps. Model training is performed over $N_{\text{step}} = 2000$ episodes (or training steps), each characterized by a different realization of the measurements. For testing, 40 Monte Carlo (MC) evaluations are considered, unless otherwise specified. During training, we adopt a trajectory length $L_\tau = H/2$ to use at most 2 mini-batches, as suggested by [87], [106]. The entropy, reward and clipping coefficients have been chosen to be $\alpha = 0.01$, $\beta = 0.05$ and $\epsilon = 0.2$, respectively. Note that $\beta = 0.05$ would correspond to an improvement step of the reward function of 5 cm in a non-standardized state scenario. The discount factor is $\gamma = 0.99$, while the Adam [107] learning rate is $\mu = 10^{-5}$ with standard hyper-parameters.

Regarding the NN architectures, we adopt a critic network with three layers: a fully-connected (FC) linear layer with 256 neurons, a gated recurrent unit (GRU) with hidden size of 256 and a final FC linear layer. The actor is an MLP with two hidden linear layers of [128, 64] neurons and rectified linear unit (ReLU) activation functions, and an output layer with sigmoid activation function. Lastly, the belief network employs two bidirectional LSTM layers of 256 hidden neurons each and ReLU activation functions, followed by a Maxout unit with 128 output features and two linear layers of [64, 32] neurons.

B. Computational Complexity and Latency

To access the real-time processing capabilities of the proposed method in fulfilling the CAVs requirements on latency, we here investigate the computational complexities and communication delays of the proposed ICP-MAPPO solution with respect to the ICP algorithm. We specify that the number of floating point operations (FLOPs) for $V_{\boldsymbol{\phi}}$, $\pi_{\boldsymbol{\theta}}$ and $b_{\boldsymbol{\psi}}$ are $0.82 \cdot 10^6$, $0.54 \cdot 10^6$, and $11.3 \cdot 10^6$, respectively. For comparison, the computational complexity of particle-based ICP methods is estimated with $O(N_{\text{mp}} \cdot N \cdot K \cdot N_{\text{p}})$, where N_{mp} and N_{p} are the number of message passing iterations and particles, respectively. The experiments are performed on a workstation machine with Intel(R) Xeon(R) Silver 4210R CPU @ 2.40 GHz, 96 GB RAM, and a Quadro RTX 6000 24 GB GPU, capable of achieving about $16.3 \cdot 10^{12}$ floating point operations per second (FLOPS) with just CPU performances. This implies a maximum latency for sample-inference of

around $1\,\mu$s, which is expected to be truthful and accurate since the computational capabilities of CAVs are planned to far exceed our workstation capabilities with more than $4 \cdot 10^{15}$ FLOPS for L5 SAE level [108].

When considering the communication delays with a hidden LSTM size of 256 bytes for ICP-MAPPO and about $N_{\text{mp}} = 1000$ particles (each with 2 bytes for 2D position and 1 byte for the weight) in the ICP method, the data transmission would require approximately 1 and 10 packets, respectively. This estimate is based on 5G vehicle-to-vehicle (V2V) communications with a typical packet size of 300 bytes. Two communication scenarios are possible: direct V2V [109] or vehicle-to-network-to-vehicle (V2N2V) [110] when under cellular coverage. For direct V2V communication, the end-to-end (E2E) packet latency is around 1 ms [109], resulting in 10 ms for ICP and 1 ms for ICP-MAPPO. In the V2N2V case, assuming the distances and scenarios described in [110], the E2E packet latency is around 4 ms, resulting in 40 ms for ICP and 4 ms for ICP-MAPPO. We note that the ICP E2E communication delay exceeds the 5 ms latency requirements of fully CAVs [111] in both scenarios, especially if a message passing procedure with multiple belief exchanges is considered. On the contrary, the ICP-MAPPO method meets the stringent latency requirements needed for fully CAVs.

C. Baseline Methods

As benchmark algorithms, we consider the following implementations:

1) KF-GNSS: Non-cooperative single-agent GNSS-based KF only using GNSS observations and perfect knowledge of the measurement standard deviation $\sigma^{(\text{GNSS})} = 2\,\text{m}$. For the motion dynamics (1), we adopt a constant velocity model with standard deviation of the Gaussian-distributed velocity driving process calibrated on the data and equal to $0.5\,\text{m/s}^2$.

2) ICP: Centralized ICP method from [32] with known A2A and A2T standard deviations, i.e., $\sigma^{(\text{A2A})} = \sigma^{(\text{A2T})} = 2\,\text{m}$, and same motion model as for the KF-GNSS. Note that the use of the exact measurement statistics in generation and tracking allows to obtain the optimal performance (i.e. with no errors due to mis-modeling). Here the network of agents is fully-connected, i.e., all the agents share the same measurements.

3) Ego ICP-MAPPO: Proposed ICP-MAPPO method, with no-cooperation, i.e., only comprising the belief LSTM and imposing no connectivity with other agents, i.e., $\bar{a}_{i,j,t} = -1 \;\forall t \in \{0, \dots, H-1\}, i \in \mathcal{V}, j \in \mathcal{N}_{i,t}$. In this way, each agent has to rely just on its measurements without performing aggregation of the neighbors' hidden features.

D. Results

1) Training performances: In the first assessment, we aim at verifying the convergence of the proposed ICP-MAPPO algorithm during the training episodes. In Fig. 5, 6 and 7, we report the mean belief LSTM loss, reward, and state value function, respectively, along with the 5-95 percentile as error bounds. The metrics are computed among agents and trajectory over the whole episode. From the figures, we

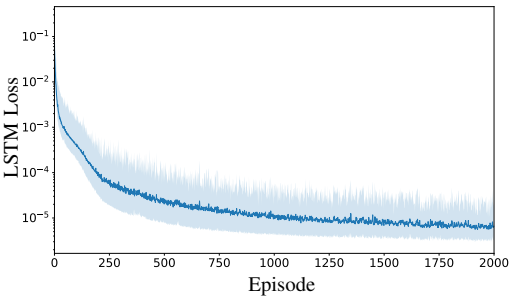


Fig. 5. Belief LSTM loss varying the number of training episodes.

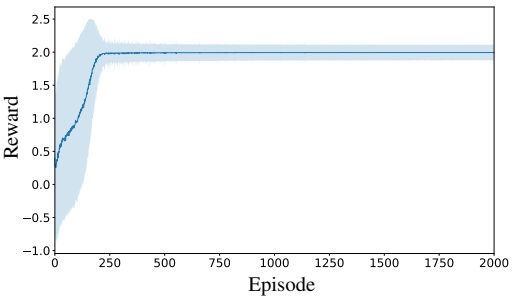


Fig. 6. Achieved reward varying the number of training episodes.

notice two distinct phases of the training: before and after reward convergence. In the first phase, i.e., before episode 250, the exploration is encouraged, leading to a much higher variability of the reward and a very rapid decrease of the LSTM loss function. After passing into the second phase, the positioning improvement becomes smaller, with a consequent convergence of the reward to the value of 2. Notably, also the mean value function converges after about 250 episodes, but with a high variance between agents and trajectories. This may be indicative of a rich and complex environment where the optimal policy may not be static, but rather dynamic and contingent on the interactions between agents and the environment. Indeed, the complexity of the state, e.g., each agent has a different trajectory in the space, can lead to a wide range of value function estimates as different states are visited with varying frequencies.

2) Testing cooperative positioning: This experiment has the objective of comparing the positioning capabilities of ICP-MAPPO with respect to the baselines in an unseen testing trajectory. To this aim, Fig. 8 shows the root mean square error (RMSE) on the vehicle position at each timestep of the trajectory (Fig. 8a) and the corresponding cumulative density function (CDF) of the absolute error (Fig. 8b). The RMSE is computed among the agents at the single timestep, while the mean and error bounds are computed within the MC evaluations. From the results, we observe that the Ego

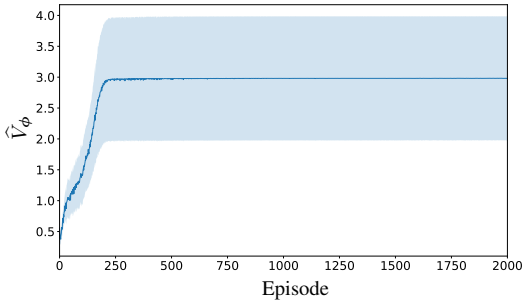
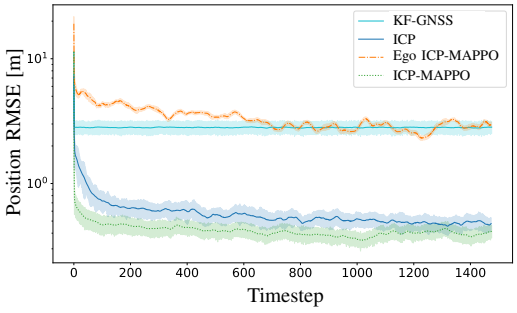
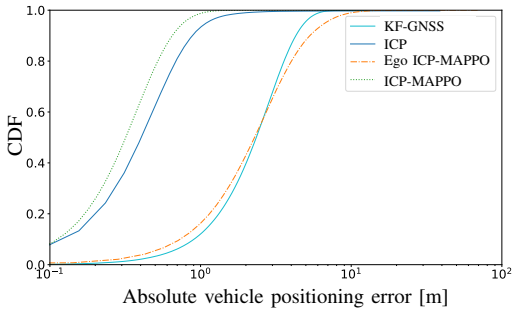


Fig. 7. Mean value function varying the number of training episodes.



(a)



(b)

Fig. 8. Testing performances on the cooperative scenario. (a) RMSE of the position over time for the single-agent KF-GNSS, ICP, proposed single agent and cooperative ICP-MAPPO. (b) CDF of the absolute error.

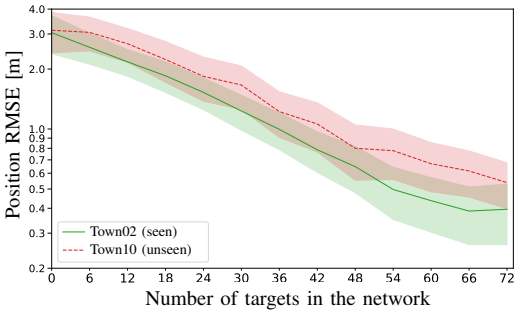


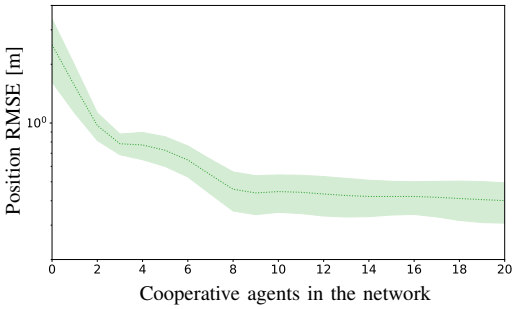
Fig. 9. RMSE on the position achieved by ICP-MAPPO varying the number of targets (i.e., poles) in two distinct environments.

ICP-MAPPO, which only relies on GNSS measurements, converges to the KF-GNSS method, indicating a correct usage of the observations to estimate the position. Passing to the cooperative methods, we notice a higher speed of convergence of ICP-MAPPO with respect to the conventional ICP. This is mainly due to the learned vehicles' dynamics and to the effective combination of neighbors' observations. As a consequence, the ICP-MAPPO algorithm outperforms the ICP method in terms of absolute error by 21%, passing from a median of 42 cm to 33 cm.

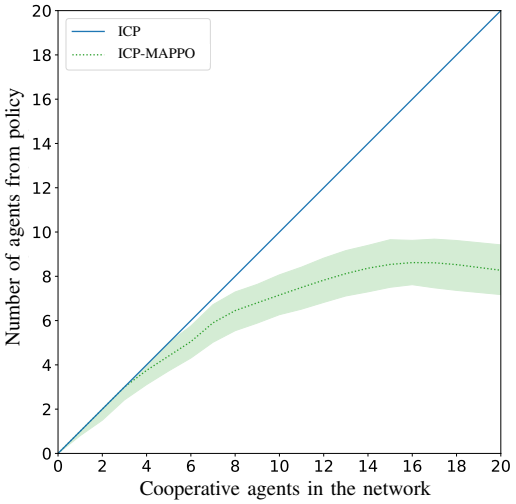
3) Generalization capabilities: This experiment aims at assessing the generalization capabilities of the proposed method in unseen scenarios. To evaluate the environmental dependence of our model, we tested the pre-trained ICP-MAPPO on a different CARLA map, specifically *Town10*. In Fig. 9, we plotted the position RMSE on testing trajectories in both *Town02* (used for training) and *Town10* (unseen environment), varying the number of passive objects in the respective map. We shall notice that the numbers of poles in *Town10* and *Town02* are 146 and 72, respectively. Since ICP-MAPPO was trained with a maximum input size of 72 measurements, we adjusted the number of targets up to 72 for this experiment.

The results in Fig. 9 confirm that, even in the unseen scenario, a higher number of vehicles increases the positioning accuracy thanks to the cooperation among vehicles. Comparing the results on *Town02* and *Town10*, we note that in the limit-case of no measurements shared among agents, the performances in the two scenarios coincide. On the contrary, when the number of features increases, the performances on the unseen scenario are slightly lower (i.e., about 10 cm) despite the completely new environment.

4) Communication efficiency: In this last assessment, we test the effectiveness of the policy choices in terms of cooperation power and communication efficiency. In Fig. 10 we report the position RMSE at convergence (Fig. 10a) and the mean number of selected agents from the policy (Fig. 10b) varying the maximum degree of connectivity allowed in the network. In Fig. 10a we observe an intuitive inverse relation between the maximum cooperative agents and the RMSE, with a rapid decrease under 1 m of RMSE with just 2 agents. Notably, after 8 cooperative agents, the improvement in RMSE is negligible, with convergence to about 40 cm. To study this behaviour, in Fig. 10b we notice that the policy tends to select no more than 9 agents for cooperation. This likely occurs because the marginal benefits of additional cooperation diminish beyond this point, leading agents to prefer collaboration with only their closest neighbors. Indeed, incorporating data from distant agents that do not observe common targets results in only slight enhancements in positional accuracy. Lastly, we highlight that the ICP-MAPPO has higher performance than



(a)



(b)

Fig. 10. Communication efficiency between ICP and the proposed ICP-MAPPO. (a) RMSE on the position varying the maximum number of cooperative agents in the network. (b) Mean number of neighbor agents selected by the policy varying the maximum connectivity of the graph.

the ICP method for the same number of cooperative agents in the network.

To evaluate the trade-off between positioning accuracy and communication overhead, in Fig. 11, we plot the mean number of A2A links, considering varying numbers of cooperative vehicles in $\{2, 6, 10, 15, 20\}$. We observe that with a smaller number of cooperative agents, such as 2, the ICP-MAPPO tends to employ all available agents, leveraging neighbors' measurements to rapidly reduce GNSS uncertainty. Conversely, with a higher number of agents, particularly beyond 10, the benefits of additional cooperation decrease (as shown in Fig. 10a). This is because only the closest neighbors with a significant number of shared targets substantially enhance positioning accuracy. Notably, with 10 and 20 agents, ICP-MAPPO reduces the number of links by 30% and 60%,

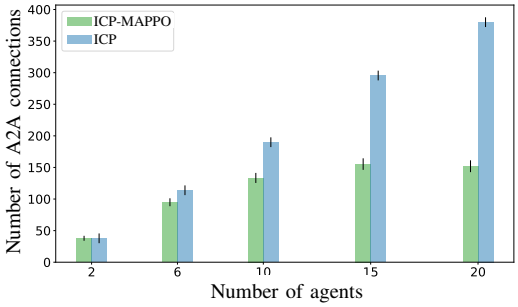


Fig. 11. Mean number of A2A connections in the network graph, for the ICP and the proposed ICP-MAPPO algorithms, and different maximum number of cooperative agents.

respectively, compared to ICP.

VI. Conclusion

In this paper, we addressed the problem of CP in a distributed network of agents that exploit passive detected targets to improve the positioning accuracy according to the ICP framework. We provided a generalization of the Bayesian ICP solution by means of MARL, which enables the dynamic optimization of the A2A links used for cooperation accounting for partial observability of the state. We presented a novel ICP-MAPPO algorithm where the agents actively select the neighbors to communicate with by following their optimized policy. This allows to minimize the communication overhead for cooperation, while improving the positioning accuracy of ego-agent systems. The proposed solution is proven to outperform single and multi-agent conventional approaches thanks to DL-based states' belief and policy models.

Realistic simulations of a C-ITS scenario created with CARLA simulator demonstrate the superior performances of ICP-MAPPO with state-of-the-art ICP methods, both in terms of positioning accuracy and efficiency of communications. The cooperation is indeed intelligently exploited to enhance the performances and, at the same time the communication efficiency, by selecting ad-hoc neighbors that are relevant for the task. The benefits of the approach look promising for applications where groups of agents have a common inference objective and predictions or decisions need to be taken based on incomplete or uncertain data.

As future work, we envision the extension of the proposed method to decentralized frameworks [112], incorporating also data association of the targets to the measurements. Additionally, performances could be enhanced by exploiting a higher dimension of latent features within object detectors, instead of filtering specific objects such as poles. This approach would allow vehicles to exchange much more meaningful information in a compressed manner. Furthermore, including motion planning [113] could enable the system to not only estimate but also modify the vehicles' states according to their destinations. Finally, introducing safe RL [114] by adding safety constraints related to communication resources, such

as maximum available bandwidth, would ensure that the policies learned by the agents remain efficient under real-world communication constraints.

APPENDIX A
PROOF OF (18)

To prove (18), we start by writing the gradient of the RL objective function in (17) as

$$\nabla_{\boldsymbol{\theta}} J(\pi_{\boldsymbol{\theta}}) = \nabla_{\boldsymbol{\theta}} \mathbb{E}_{\boldsymbol{\tau}\sim p(\boldsymbol{\tau}|\pi_{\boldsymbol{\theta}})}\Big\{\widetilde{R}(\boldsymbol{\tau})\Big\} = \nabla_{\boldsymbol{\theta}} \sum_{\boldsymbol{\tau}} p(\boldsymbol{\tau}|\pi_{\boldsymbol{\theta}})\, \widetilde{R}(\boldsymbol{\tau})$$
$$= \sum_{\boldsymbol{\tau}} \nabla_{\boldsymbol{\theta}}\, p(\boldsymbol{\tau}|\pi_{\boldsymbol{\theta}})\, \widetilde{R}(\boldsymbol{\tau}). \tag{A1}$$

Now, we can rewrite the gradient of the trajectory PDF $\nabla_{\boldsymbol{\theta}}\, p(\boldsymbol{\tau}|\pi_{\boldsymbol{\theta}})$ using the log-derivative trick as

$$\nabla_{\boldsymbol{\theta}}\, p(\boldsymbol{\tau}|\pi_{\boldsymbol{\theta}}) = p(\boldsymbol{\tau}|\pi_{\boldsymbol{\theta}})\, \nabla_{\boldsymbol{\theta}} \log\big(p(\boldsymbol{\tau}|\pi_{\boldsymbol{\theta}})\big). \tag{A2}$$

Given that the gradient of the log-trajectory PDF $\nabla_{\boldsymbol{\theta}} \log\big(p(\boldsymbol{\tau}|\pi_{\boldsymbol{\theta}})\big)$ is

$$\nabla_{\boldsymbol{\theta}} \log\big(p(\boldsymbol{\tau}|\pi_{\boldsymbol{\theta}})\big) = \nabla_{\boldsymbol{\theta}} \log\Big(T_0 \prod_{t=0}^{H-1} T(\boldsymbol{s}_{t+1}|\boldsymbol{s}_t, \boldsymbol{a}_t)\, \pi_{\boldsymbol{\theta}}(\boldsymbol{a}_t|\boldsymbol{s}_t)\Big)$$
$$= \sum_{t=0}^{H-1} \nabla_{\boldsymbol{\theta}} \log\big(\pi_{\boldsymbol{\theta}}(\boldsymbol{a}_t|\boldsymbol{s}_t)\big) \tag{A3}$$

we can rewrite (A1) as

$$\nabla_{\boldsymbol{\theta}}\, J(\pi_{\boldsymbol{\theta}}) = \sum_{\boldsymbol{\tau}} p(\boldsymbol{\tau}|\pi_{\boldsymbol{\theta}})\, \nabla_{\boldsymbol{\theta}} \log\big(p(\boldsymbol{\tau}|\pi_{\boldsymbol{\theta}})\big)\widetilde{R}(\boldsymbol{\tau})$$
$$= \mathbb{E}_{\boldsymbol{\tau}\sim p(\boldsymbol{\tau}|\pi_{\boldsymbol{\theta}})}\Big\{\nabla_{\boldsymbol{\theta}} \log\big(p(\boldsymbol{\tau}|\pi_{\boldsymbol{\theta}})\big)\widetilde{R}(\boldsymbol{\tau})\Big\}$$
$$= \mathbb{E}_{(\mathbf{s}_t,\mathbf{a}_t)\sim p(\boldsymbol{s}_t,\boldsymbol{a}_t|\pi_{\boldsymbol{\theta}})}\Big\{\sum_{t=0}^{H-1} \nabla_{\boldsymbol{\theta}} \log\big(\pi_{\boldsymbol{\theta}}(\boldsymbol{a}_t|\boldsymbol{s}_t)\big)$$
$$\times \sum_{t=0}^{H-1} \gamma^t\, R\big(\mathbf{s}_t, \mathbf{a}_t\big)\Big\}. \tag{A4}$$

Since the action $\mathbf{a}_t$ at time t only influences the future rewards and not the past ones, (A4) can be equivalently rewritten as

$$\nabla_{\boldsymbol{\theta}}\, J(\pi_{\boldsymbol{\theta}}) = \mathbb{E}_{(\mathbf{s}_t,\mathbf{a}_t)\sim p(\boldsymbol{s}_t,\boldsymbol{a}_t|\pi_{\boldsymbol{\theta}})}\Big\{\sum_{t=0}^{H-1} \nabla_{\boldsymbol{\theta}} \log\big(\pi_{\boldsymbol{\theta}}(\boldsymbol{a}_t|\boldsymbol{s}_t)\big)\mathsf{R}_t\Big\} \tag{A5}$$

where we used the reward-to-go at time t $\mathsf{R}_t = \sum_{t'=t}^{H-1} \gamma^{t'-t} R\big(\mathbf{s}_{t'}, \mathbf{a}_{t'}\big)$, as opposed to R_0.

Since it can be proven that for any function of the state $B(\mathbf{s}_t)$ called baseline, we have that $\mathbb{E}_{\mathbf{a}_t\sim\pi_{\boldsymbol{\theta}}(\boldsymbol{a}_t|\boldsymbol{s}_t)}\Big\{\nabla_\theta \log\big(\pi_{\boldsymbol{\theta}}(\boldsymbol{a}_t|\boldsymbol{s}_t)\, B(\mathbf{s}_t)\big)\Big\} = 0$, then we can reduce the variance of the PO algorithm, while remaining unbiased, by subtracting the baseline from the reward-to-go as

$$\nabla_{\boldsymbol{\theta}}\, J(\pi_{\boldsymbol{\theta}}) = \mathbb{E}_{(\mathbf{s}_t,\mathbf{a}_t)\sim p(\boldsymbol{s}_t,\boldsymbol{a}_t|\pi_{\boldsymbol{\theta}})}\Big\{\sum_{t=0}^{H-1} \Big[\nabla_{\boldsymbol{\theta}} \log\big(\pi_{\boldsymbol{\theta}}(\boldsymbol{a}_t|\boldsymbol{s}_t)\big)$$
$$\times \big(\mathsf{R}_t - B(\mathbf{s}_t)\big)\Big]\Big\}. \tag{A6}$$

Finally, R_t and $B(\mathbf{s}_t)$ are usually substituted with their estimates $Q^\pi(\mathbf{s}_t, \mathbf{a}_t)$ and $V^\pi(\mathbf{s}_t)$, respectively, leading to the definition of the advantage function $A_t = Q^\pi(\mathbf{s}_t, \mathbf{a}_t) - V^\pi(\mathbf{s}_t)$. Recently, more advanced versions of the advantage function, as the generalized advantage estimator (GAE) function A_t^{GAE} have been proposed in the literature [100] to regulate the bias-variance trade-off, increase stability, efficiency, and obtain faster convergence. We want to point out that usage of the baseline and/or the estimate of R_t are not necessary, and thus any function $F_t \in \big\{\mathsf{R}_t, Q^\pi(\mathbf{s}_t, \mathbf{a}_t), \mathsf{R}_t - V^\pi(\mathbf{s}_t), A_t, A_t^{\mathrm{GAE}}\big\}$ is a valid choice.

REFERENCES

[1] M. Z. Win *et al.*, "Network localization and navigation via cooperation," *IEEE Commun. Mag.*, vol. 49, no. 5, pp. 56–62, May 2011.
[2] L. Gao *et al.*, "Cooperative localization in transportation 5.0," *IEEE Trans. Intell. Veh.*, pp. 1–6, Mar. 2024.
[3] M. Z. Win, Y. Shen, and W. Dai, "A theoretical foundation of network localization and navigation," *Proc. IEEE*, vol. 106, no. 7, pp. 1136–1165, Jul. 2018.
[4] Y. Gao, H. Jing, M. Dianati, C. M. Hancock, and X. Meng, "Performance analysis of robust cooperative positioning based on GPS/UWB integration for connected autonomous vehicles," *IEEE Trans. Intell. Veh.*, vol. 8, no. 1, pp. 790–802, Jan. 2023.
[5] A. Mahmoud, A. Noureldin, and H. S. Hassanein, "Integrated positioning for connected vehicles," *IEEE Trans. Intell. Transp. Syst.*, vol. 21, no. 1, pp. 397–409, Jan. 2020.
[6] A. Conti, S. Mazuelas, S. Bartoletti, W. C. Lindsey, and M. Z. Win, "Soft information for localization-of-things," *Proc. IEEE*, vol. 107, no. 11, pp. 2240–2264, Nov. 2019.
[7] F. Gustafsson and F. Gunnarsson, "Mobile positioning using wireless networks: possibilities and fundamental limitations based on available wireless network measurements," *IEEE Signal Process. Mag.*, vol. 22, no. 4, pp. 41–53, Jul. 2005.
[8] C. A. Gómez-Vega, Z. Liu, C. A. Gutiérrez, M. Z. Win, and A. Conti, "Efficient deployment strategies for network localization with assisting nodes," *IEEE Trans. Mobile Comput.*, vol. 23, no. 5, pp. 6272–6287, May 2024.
[9] L. Chen *et al.*, "Milestones in autonomous driving and intelligent vehicles: Survey of surveys," *IEEE Trans. Intell. Veh.*, vol. 8, no. 2, pp. 1046–1056, Feb. 2023.
[10] D. Cao *et al.*, "Future directions of intelligent vehicles: Potentials, possibilities, and perspectives," *IEEE Trans. Intell. Veh.*, vol. 7, no. 1, pp. 7–10, Mar. 2022.
[11] P. Yang, D. Duan, C. Chen, X. Cheng, and L. Yang, "Multi-sensor multi-vehicle (MSMV) localization and mobility tracking for autonomous driving," *IEEE Trans. Veh. Technol.*, vol. 69, no. 12, pp. 14 355–14 364, Oct. 2020.
[12] J. Ji, A. Khajepour, W. W. Melek, and Y. Huang, "Path planning and tracking for vehicle collision avoidance based on model predictive control with multiconstraints," *IEEE Trans. Veh. Technol.*, vol. 66, no. 2, pp. 952–964, Feb. 2017.
[13] T. G. Reid *et al.*, "Localization requirements for autonomous vehicles," *SAE Int. J. Connected Automated Veh.*, vol. 2, no. 3, pp. 12–02–03–0012, Oct. 2019.
[14] S. Kuutti, S. Fallah, K. Katsaros, M. Dianati, F. Mccullough, and A. Mouzakitis, "A survey of the state-of-the-art localization techniques and their potentials for autonomous vehicle applications," *IEEE Internet Things J.*, vol. 5, no. 2, pp. 829–846, Mar. 2018.
[15] M. H. C. Garcia *et al.*, "A tutorial on 5G NR V2X communications," *IEEE Commun. Surveys & Tuts.*, vol. 23, no. 3, pp. 1972–2026, Feb. 2021.
[16] F. Morselli, S. Modarres Razavi, M. Z. Win, and A. Conti, "Soft information-based localization for 5G networks and beyond," *IEEE Trans. Wireless Commun.*, vol. 22, no. 12, pp. 9923–9938, Dec. 2023.
[17] H. Zhou, W. Xu, J. Chen, and W. Wang, "Evolutionary V2X technologies toward the internet of vehicles: Challenges and opportunities," *Proc. IEEE*, vol. 108, no. 2, pp. 308–323, Feb. 2020.
[18] G. Torsoli, M. Z. Win, and A. Conti, "Blockage intelligence in complex environments for beyond 5G localization," *IEEE J. Sel. Areas Commun.*, vol. 41, no. 6, pp. 1688–1701, Jun. 2023.
[19] A. Conti, G. Torsoli, C. A. Gómez-Vega, A. Vaccari, G. Mazzini, and M. Z. Win, "3GPP-compliant datasets for xG location-aware networks," *IEEE Open J. Veh. Technol.*, vol. 5, pp. 473–484, Dec. 2024.

Chapter 6. Multi-Agent Reinforcement Learning

[20] A. Conti, G. Torsoli, C. A. Gómez-Vega, A. Vaccari, and M. Z. Win, "xG-Loc: 3GPP-compliant datasets for xG location-aware networks," *IEEE Dataport*, Dec. 2023.

[21] L. Italiano, B. Camajori Tedeschini, M. Brambilla, H. Huang, M. Nicoli, and H. Wymeersch, "A tutorial on 5G positioning," *IEEE Commun. Surveys & Tuts.*, pp. 1–48, Aug. 2024.

[22] B. Camajori Tedeschini, G. Kwon, M. Nicoli, and M. Z. Win, "Real-time Bayesian neural networks for 6G cooperative positioning and tracking," *IEEE J. Sel. Areas Commun.*, vol. 42, no. 9, pp. 2322–2338, Aug. 2024.

[23] A. Conti *et al.*, "Location awareness in beyond 5G networks," *IEEE Commun. Mag.*, vol. 59, no. 11, pp. 22–27, Nov. 2021.

[24] B. Camajori Tedeschini and M. Nicoli, "Cooperative deep-learning positioning in mmWave 5G-advanced networks," *IEEE J. Sel. Areas Commun.*, vol. 41, no. 12, pp. 3799–3815, Dec. 2023.

[25] M. A. Javed, S. Zeadally, and E. B. Hamida, "Data analytics for cooperative intelligent transport systems," *Veh. Commun.*, vol. 15, pp. 63–72, Jan. 2019.

[26] A. Alalewi, I. Dayoub, and S. Cherkaoui, "On 5G-V2X use cases and enabling technologies: A comprehensive survey," *IEEE Access*, vol. 9, pp. 107 710–107 737, Jul. 2021.

[27] K. Sehla, T. M. T. Nguyen, G. Pujolle, and P. B. Velloso, "Resource allocation modes in C-V2X: From LTE-V2X to 5G-V2X," *IEEE Internet Things J.*, vol. 9, no. 11, pp. 8291–8314, Mar. 2022.

[28] M. Driusso, C. Marshall, M. Sabathy, F. Knutti, H. Mathis, and F. Babich, "Vehicular position tracking using LTE signals," *IEEE Trans. Veh. Technol.*, vol. 66, no. 4, pp. 3376–3391, Apr. 2017.

[29] G. Kwon, Z. Liu, A. Conti, H. Park, and M. Z. Win, "Integrated localization and communication for efficient millimeter wave networks," *IEEE J. Sel. Areas Commun.*, vol. 41, no. 12, pp. 3925–3941, Dec. 2023.

[30] F. Liu *et al.*, "Integrated sensing and communications: Toward dual-functional wireless networks for 6G and beyond," *IEEE J. Sel. Areas Commun.*, vol. 40, no. 6, pp. 1728–1767, Jun. 2022.

[31] G. Kwon, A. Conti, H. Park, and M. Z. Win, "Joint communication and localization in millimeter wave networks," *IEEE J. Sel. Topics Signal Process.*, vol. 15, no. 6, pp. 1439–1454, Nov. 2021.

[32] G. Soatti, M. Nicoli, N. Garcia, B. Denis, R. Raulefs, and H. Wymeersch, "Implicit cooperative positioning in vehicular networks," *IEEE Trans. Intell. Transp. Syst.*, vol. 19, no. 12, pp. 3964–3980, Dec. 2018.

[33] M. Brambilla, M. Nicoli, G. Soatti, and F. Deflorio, "Augmenting vehicle localization by cooperative sensing of the driving environment: Insight on data association in urban traffic scenarios," *IEEE Trans. Intell. Transp. Syst.*, vol. 21, no. 4, pp. 1646–1663, Apr. 2020.

[34] M. Brambilla *et al.*, "Cooperative localization and multitarget tracking in agent networks with the sum-product algorithm," *IEEE Open J. Signal Process.*, vol. 3, pp. 169–195, Mar. 2022.

[35] F. Jin, K. Liu, C. Liu, T. Cheng, H. Zhang, and V. C. S. Lee, "A cooperative vehicle localization and trajectory prediction framework based on belief propagation and transformer model," *IEEE Trans. Consum. Electron.*, pp. 1–1, Feb. 2024.

[36] M. Z. Win, W. Dai, Y. Shen, G. Chrisikos, and H. Vincent Poor, "Network operation strategies for efficient localization and navigation," *Proc. IEEE*, vol. 106, no. 7, pp. 1224–1254, Jul. 2018.

[37] A. Conti, M. Guerra, D. Dardari, N. Decarli, and M. Z. Win, "Network experimentation for cooperative localization," *IEEE J. Sel. Areas Commun.*, vol. 30, no. 2, pp. 467–475, Feb. 2012.

[38] W. Dai, Y. Shen, and M. Z. Win, "Distributed power allocation for cooperative wireless network localization," *IEEE J. Sel. Areas Commun.*, vol. 33, no. 1, pp. 28–40, Jan. 2015.

[39] P. Yang, C. Xiang, and S. Zhang, "Distributed joint power and bandwidth allocation for multiagent cooperative localization," *IEEE Commun. Lett.*, vol. 26, no. 11, pp. 2601–2605, Aug. 2022.

[40] T. Zhang, A. F. Molisch, Y. Shen, Q. Zhang, H. Feng, and M. Z. Win, "Joint power and bandwidth allocation in wireless cooperative localization networks," *IEEE Trans. Wireless Commun.*, vol. 15, no. 10, pp. 6527–6540, Oct. 2016.

[41] J. Nie, J. Yan, H. Yin, L. Ren, and Q. Meng, "A multimodality fusion deep neural network and safety test strategy for intelligent vehicles," *IEEE Trans. Intell. Veh.*, vol. 6, no. 2, pp. 310–322, Sep. 2020.

[42] Z. Liu *et al.*, "Robust target recognition and tracking of self-driving cars with radar and camera information fusion under severe weather conditions," *IEEE Trans. Intell. Transp. Syst.*, vol. 23, no. 7, pp. 6640–6653, Feb. 2021.

[43] T. Wang, Y. Shen, A. Conti, and M. Z. Win, "Network navigation with scheduling: Error evolution," *IEEE Trans. Inf. Theory*, vol. 63, no. 11, pp. 7509–7534, Nov. 2017.

[44] B. Teague, Z. Liu, F. Meyer, A. Conti, and M. Z. Win, "Network localization and navigation with scalable inference and efficient operation," *IEEE Trans. Mobile Comput.*, vol. 21, no. 6, pp. 2072–2087, Jun. 2022.

[45] S. Dwivedi, D. Zachariah, A. De Angelis, and P. Handel, "Cooperative decentralized localization using scheduled wireless transmissions," *IEEE Commun. Lett.*, vol. 17, no. 6, pp. 1240–1243, Jun. 2013.

[46] L. Barbieri, B. Camajori Tedeschini, M. Brambilla, and M. Nicoli, "Deep learning-based cooperative LiDAR sensing for improved vehicle positioning," *IEEE Trans. Signal Process.*, vol. 72, pp. 1666–1682, Mar. 2024.

[47] Y. Weiss and W. T. Freeman, "Correctness of belief propagation in Gaussian graphical models of arbitrary topology," *Neural Comput.*, vol. 13, no. 10, pp. 2173–2200, Oct. 2001.

[48] J. Yedidia, W. Freeman, and Y. Weiss, "Constructing free-energy approximations and generalized belief propagation algorithms," *IEEE Trans. Inf. Theory*, vol. 51, no. 7, pp. 2282–2312, Jul. 2005.

[49] M. J. Wainwright and M. I. Jordan, "Graphical models, exponential families, and variational inference," *Found. Trends® Mach. Learn.*, vol. 1, no. 1-2, pp. 1–305, Nov. 2008.

[50] F. Meyer *et al.*, "Message passing algorithms for scalable multitarget tracking," *Proc. IEEE*, vol. 106, no. 2, pp. 221–259, Feb. 2018.

[51] G. Soldi, F. Meyer, P. Braca, and F. Hlawatsch, "Self-tuning algorithms for multisensor-multitarget tracking using belief propagation," *IEEE Trans. Signal Process.*, vol. 67, no. 15, pp. 3922–3937, Aug. 2019.

[52] M. G. Kibria, K. Nguyen, G. P. Villardi, O. Zhao, K. Ishizu, and F. Kojima, "Big data analytics, machine learning, and artificial intelligence in next-generation wireless networks," *IEEE Access*, vol. 6, pp. 32 328–32 338, May 2018.

[53] B. Camajori Tedeschini, M. Brambilla, L. Barbieri, G. Balducci, and M. Nicoli, "Cooperative lidar sensing for pedestrian detection: Data association based on message passing neural networks," *IEEE Trans. Signal Process.*, vol. 71, pp. 3028–3042, Aug. 2023.

[54] B. Camajori Tedeschini, M. Brambilla, and M. Nicoli, "Message passing neural network versus message passing algorithm for cooperative positioning," *IEEE Trans. Cogn. Commun. Netw.*, vol. 9, no. 6, pp. 1666–1676, Dec. 2023.

[55] B. Camajori Tedeschini, M. Brambilla, and M. Nicoli, "Split consensus federated learning: an approach for distributed training and inference," *IEEE Access*, vol. 12, pp. 119 535–119 549, Aug. 2024.

[56] R. S. Sutton and A. G. Barto, *Reinforcement Learning: An Introduction*. A Bradford Book, Oct. 2018.

[57] D. P. Bertsekas, *A Course in Reinforcement Learning*, 1st ed. Athena Scientific, Jun. 2023.

[58] D. P. Bertsekas, *Reinforcement Learning and Optimal Control*, 1st ed. Athena Scientific, Jul. 2019.

[59] V. Mnih *et al.*, "Human-level control through deep reinforcement learning," *Nature*, vol. 518, no. 7540, pp. 529–533, Feb. 2015.

[60] D. P. Bertsekas, *Lessons from AlphaZero for Optimal, Model Predictive, and Adaptive Control*, 1st ed. Athena Scientific, Aug. 2021.

[61] T. P. Lillicrap *et al.*, "Continuous control with deep reinforcement learning," in *Proc. 4th Int. Conf. Learn. Representations*, Sep. 2016, pp. 1–14.

[62] N. R. Ke *et al.*, "Learning dynamics model in reinforcement learning by incorporating the long term future," in *Proc. 7th Int. Conf. Learn. Representations*, Mar. 2019, pp. 1–14.

[63] D. P. Bertsekas, *Dynamic Programming and Optimal Control*, 4th ed. Athena Scientific, Oct. 2000.

[64] D. P. Bertsekas, *Dynamic Programming and Optimal Control, Volume II: Approximate Dynamic Programming*, 4th ed. Athena Scientific, Jun. 2012.

[65] N. C. Luong *et al.*, "Applications of deep reinforcement learning in communications and networking: A survey," *IEEE Commun. Surveys & Tuts.*, vol. 21, no. 4, pp. 3133–3174, Oct. 2019.

[66] H. Zhang, N. Yang, W. Huangfu, K. Long, and V. C. M. Leung, "Power control based on deep reinforcement learning for spectrum sharing," *IEEE Trans. Wireless Commun.*, vol. 19, no. 6, pp. 4209–4219, Mar. 2020.

[67] F. Meng, P. Chen, L. Wu, and J. Cheng, "Power allocation in multi-user cellular networks: Deep reinforcement learning approaches," *IEEE Trans. Wireless Commun.*, vol. 19, no. 10, pp. 6255–6267, Jun. 2020.

[68] X. Li, J. Fang, W. Cheng, H. Duan, Z. Chen, and H. Li, "Intelligent power control for spectrum sharing in cognitive radios: A deep reinforcement learning approach," *IEEE Access*, vol. 6, pp. 25 463–25 473, Apr. 2018.

[69] J. Vlachogiannis and N. Hatziargyriou, "Reinforcement learning for reactive power control," *IEEE Trans. Power Syst.*, vol. 19, no. 3, pp. 1317–1325, Aug. 2004.

Chapter 6. Multi-Agent Reinforcement Learning

[70] L. Liang, H. Ye, and G. Y. Li, "Spectrum sharing in vehicular networks based on multi-agent reinforcement learning," *IEEE J. Sel. Areas Commun.*, vol. 37, no. 10, pp. 2282–2292, Aug. 2019.

[71] O. Naparstek and K. Cohen, "Deep multi-user reinforcement learning for distributed dynamic spectrum access," *IEEE Trans. Wireless Commun.*, vol. 18, no. 1, pp. 310–323, Jan. 2019.

[72] H. Song, L. Liu, J. Ashdown, and Y. Yi, "A deep reinforcement learning framework for spectrum management in dynamic spectrum access," *IEEE Internet Things J.*, vol. 8, no. 14, pp. 11 208–11 218, Jan. 2021.

[73] Q. Luo, C. Li, T. H. Luan, and W. Shi, "Collaborative data scheduling for vehicular edge computing via deep reinforcement learning," *IEEE Internet Things J.*, vol. 7, no. 10, pp. 9637–9650, Mar. 2020.

[74] T. T. Nguyen, N. D. Nguyen, and S. Nahavandi, "Deep reinforcement learning for multiagent systems: A review of challenges, solutions, and applications," *IEEE Trans. Cybern.*, vol. 50, no. 9, pp. 3826–3839, Mar. 2020.

[75] F. A. Oliehoek and C. Amato, *A Concise Introduction to Decentralized POMDPs*, ser. SpringerBriefs in Intelligent Systems. Springer International Publishing, Jun. 2016.

[76] L. Kraemer and B. Banerjee, "Multi-agent reinforcement learning as a rehearsal for decentralized planning," *Neurocomputing*, vol. 190, no. C, p. 82–94, May 2016.

[77] S. Bhattacharya, S. Kailas, S. Badyal, S. Gil, and D. Bertsekas, "Multiagent reinforcement learning: Rollout and policy iteration for POMDP with application to multirobot problems," *IEEE Trans. Robot.*, vol. 40, pp. 2003–2023, Dec. 2024.

[78] S. Omidshafiei, J. Pazis, C. Amato, J. P. How, and J. Vian, "Deep decentralized multi-task multi-agent reinforcement learning under partial observability," in *Proc. 34th Int. Conf. Mach. Learn.*, Aug. 2017, p. 2681–2690.

[79] S. Gronauer and K. Diepold, "Multi-agent deep reinforcement learning: a survey," *Artif. Intell. Rev.*, vol. 55, no. 2, pp. 895–943, Apr. 2021.

[80] K. Zhang, Z. Yang, and T. Başar, *Multi-Agent Reinforcement Learning: A Selective Overview of Theories and Algorithms*. Springer International Publishing, Jun. 2021, pp. 321–384.

[81] L. Busoniu, R. Babuska, and B. De Schutter, "A comprehensive survey of multiagent reinforcement learning," *IEEE Trans. Syst., Man, Cybern., Part C (Appl. Rev.)*, vol. 38, no. 2, p. 156–172, Mar. 2008.

[82] T. Rashid, M. Samvelyan, C. S. de Witt, G. Farquhar, J. Foerster, and S. Whiteson, "Monotonic value function factorisation for deep multi-agent reinforcement learning," *J. Mach. Learn. Res. 21(178):1-51, 2020*, Mar. 2020.

[83] K. Son, D. Kim, W. J. Kang, D. E. Hostallero, and Y. Yi, "QTRAN: Learning to factorize with transformation for cooperative multi-agent reinforcement learning," in *Proc. 36th Int. Conf. Mach. Learn.*, May 2019, pp. 5887–5896.

[84] J. Wang, Z. Ren, T. Liu, Y. Yu, and C. Zhang, "QPLEX: Duplex dueling multi-agent Q-learning," in *Proc. 38th Int. Conf. Mach. Learn.*, Aug. 2021, pp. 1–27.

[85] R. Lowe, Y. Wu, A. Tamar, J. Harb, P. Abbeel, and I. Mordatch, "Multi-agent actor-critic for mixed cooperative-competitive environments," in *Proc. 31th Int. Conf. Neural Inf. Process. Syst.*, Dec. 2017, pp. 6382–6393.

[86] J. Foerster, G. Farquhar, T. Afouras, N. Nardelli, and S. Whiteson, "Counterfactual multi-agent policy gradients," in *Proc. 32nd AAAI Conf. Artif. Intell.*, Feb. 2018, pp. 2974–2982.

[87] C. Yu *et al.*, "The surprising effectiveness of PPO in cooperative, multi-agent games," in *Proc. 36th Int. Conf. Neural Inf. Process. Syst.*, Mar. 2021, pp. 1–14.

[88] V. Mnih *et al.*, "Asynchronous methods for deep reinforcement learning," in *Proc. 33rd Int. Conf. Mach. Learn.*, Feb. 2016, pp. 1928–1937.

[89] J. K. Gupta, M. Egorov, and M. Kochenderfer, "Cooperative multi-agent control using deep reinforcement learning," in *Auton. Agents Multiagent Syst.*, ser. Lecture Notes in Computer Science, G. Sukthankar and J. A. Rodriguez-Aguilar, Eds. Springer International Publishing, Nov. 2017, pp. 66–83.

[90] F. Christianos, G. Papoudakis, M. A. Rahman, and S. V. Albrecht, "Scaling multi-agent reinforcement learning with selective parameter sharing," in *Proc. 38th Int. Conf. Mach. Learn.*, Feb. 2021, pp. 1989–1998.

[91] J. K. Terry, N. Grammel, S. Son, and B. Black, "Parameter sharing for heterogeneous agents in multi-agent reinforcement learning," *ArXiv*, pp. 1–16, May 2020.

[92] Z. Xia *et al.*, "Multi-agent reinforcement learning aided intelligent UAV swarm for target tracking," *IEEE Trans. Veh. Technol.*, vol. 71, no. 1, pp. 931–945, Nov. 2021.

[93] B. Peng, G. Seco-Granados, E. Steinmetz, M. Frohle, and H. W. Wymeersch, "Decentralized scheduling for cooperative localization with deep reinforcement learning," *IEEE Trans. Veh. Technol.*, vol. 68, no. 5, pp. 4295–4305, May 2019.

[94] A. Dosovitskiy, G. Ros, F. Codevilla, A. Lopez, and V. Koltun, "CARLA: An open urban driving simulator," in *Proc. 1st Conf. Robot Learning*, Nov. 2017, pp. 1–16.

[95] M. Arulampalam, S. Maskell, N. Gordon, and T. Clapp, "A tutorial on particle filters for online nonlinear/non-Gaussian Bayesian tracking," *IEEE Trans. Signal Process.*, vol. 50, no. 2, pp. 174–188, Feb. 2002.

[96] A. Tampuu *et al.*, "Multiagent cooperation and competition with deep reinforcement learning," *PLOS ONE*, vol. 12, Apr. 2017.

[97] N. Mehta, P. Tadepalli, and A. Fern, "Multi-agent shared hierarchy reinforcement learning," in *ICML Workshop Rich Representations Reinforcement Learn.*, Aug. 2005, pp. 45–50.

[98] P. Sunehag *et al.*, "Value-decomposition networks for cooperative multi-agent learning," *ArXiv*, pp. 1–17, Jun. 2017.

[99] R. J. Williams, "Simple statistical gradient-following algorithms for connectionist reinforcement learning," *Mach. Learn.*, vol. 8, no. 3-4, pp. 229–256, Jan. 2005.

[100] J. Schulman, P. Moritz, S. Levine, M. Jordan, and P. Abbeel, "High-dimensional continuous control using generalized advantage estimation," in *Proc. 4th Int. Conf. Mach. Learn.*, Jun. 2016, pp. 1–14.

[101] J. Schulman, S. Levine, P. Moritz, M. I. Jordan, and P. Abbeel, "Trust region policy optimization," in *Proc. 31st Int. Conf. Mach. Learn.*, Feb. 2015, pp. 1889–1897.

[102] J. Schulman, F. Wolski, P. Dhariwal, A. Radford, and O. Klimov, "Proximal policy optimization algorithms," *ArXiv*, pp. 1–12, Jul. 2017.

[103] V. Feinberg, A. Wan, I. Stoica, M. I. Jordan, J. E. Gonzalez, and S. Levine, "Model-based value estimation for efficient model-free reinforcement learning," in *Proc. 35st Int. Conf. Mach. Learn.*, Feb. 2018, pp. 1–12.

[104] J. Buckman, D. Hafner, G. Tucker, E. Brevdo, and H. Lee, "Sample-efficient reinforcement learning with stochastic ensemble value expansion," in *Proc. 31th Int. Conf. Neural Inf. Process. Syst.*, Dec. 2018, pp. 8234–8244.

[105] D. P. Bertsekas, *Rollout, Policy Iteration, and Distributed Reinforcement Learning*, 1st ed. Athena Scientific, Aug. 2020.

[106] A. Ilyas *et al.*, "A closer look at deep policy gradients," in *Proc. 8th Int. Conf. Learn. Representations*, Nov. 2020, pp. 1–27.

[107] D. P. Kingma and J. Ba, "Adam: A method for stochastic optimization," in *Proc. 3rd Int. Conf. Learn. Representations*, Dec. 2015, pp. 1–15.

[108] J. Jhung, H. Suk, H. Park, and S. Kim, "Hardware accelerators for autonomous vehicles," in *Artificial Intelligence and Hardware Accelerators*, A. Mishra, J. Cha, H. Park, and S. Kim, Eds. Cham: Springer International Publishing, 2023, pp. 269–317.

[109] M. Mikami, K. Serizawa, Y. Ishida, H. Nishiyori, K. Moto, and H. Yoshino, "Field experimental evaluation on latency and reliability performance of 5G NR V2V direct communication in real express highway environment," in *2020 IEEE 91st Veh. Technol. Conf. (VTC2020-Spring)*. IEEE, May 2020, pp. 1–5.

[110] B. Coll-Perales *et al.*, "End-to-end V2X latency modeling and analysis in 5G networks," *IEEE Trans. Veh. Technol.*, vol. 72, no. 4, pp. 5094–5109, Apr. 2023.

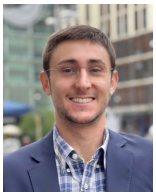
[111] *Study on enhancement of 3GPP Support for 5G V2X Services*, TR 22.886 Version 16.2.0, 3rd Generation Partnership Project (3GPP), Sophia Antipolis, France, 2018, Dec.

[112] D. Chen, K. Zhang, Y. Wang, X. Yin, Z. Li, and D. Filev, "Communication-efficient decentralized multi-agent reinforcement learning for cooperative adaptive cruise control," *IEEE Trans. Intell. Veh.*, pp. 1–14, Feb. 2024.

[113] L. Zhang, R. Zhang, T. Wu, R. Weng, M. Han, and Y. Zhao, "Safe reinforcement learning with stability guarantee for motion planning of autonomous vehicles," *IEEE Trans. Neural Netw. Learn. Syst.*, vol. 32, no. 12, pp. 5435–5444, Dec. 2021.

[114] M. Han, Y. Tian, L. Zhang, J. Wang, and W. Pan, "Reinforcement learning control of constrained dynamic systems with uniformly ultimate boundedness stability guarantee," *Autom.*, vol. 129, p. 109689, May 2021.

Chapter 6. Multi-Agent Reinforcement Learning



Bernardo Camajori Tedeschini (Graduate Student Member, IEEE) is pursuing the Ph.D. degree in Information Technology at the Dipartimento di Elettronica, Informazione e Bioingegneria (DEIB), Politecnico di Milano, Milan, Italy, since November 2021. He received his M.Sc. (Hons.) degree in Telecommunications Engineering and B.Sc. (Hons.) degree in Computer Science from the Politecnico di Milano, Milan, Italy, in 2021 and 2019, respectively.

Currently, he is a Visiting PhD Researcher at the Wireless Information and Network Sciences Laboratory, the Massachusetts Institute of Technology (MIT), Cambridge, MA, USA. In 2021, he has served as a Visiting Research Scientist at CERN, Geneva, Switzerland, where he worked on the CAFEIN project, focusing on the development and deployment of a Federated network platform. His research interests encompass federated learning, machine learning for signal processing and sensing over networks, and localization methods.

Mr. Camajori Tedeschini is a recipient of a Ph.D. grant from Italy's Ministero dell'Istruzione, dell'Università e della Ricerca (MIUR) and the Roberto Rocca Doctoral Fellowship, which was jointly awarded by MIT and Politecnico di Milano. He earned both his Bachelor's and Master's degrees with highest honors and he was honored with the best freshmen prize from Politecnico di Milano in 2017.

Mattia Brambilla (Member, IEEE) received the B.Sc. and M.Sc. degrees in telecommunication engineering and the Ph.D. degree (cum laude) in information technology from the Politecnico di Milano, in 2015, 2017, and 2021, respectively.

He was a Visiting Researcher with the NATO Centre for Maritime Research and Experimentation (CMRE), La Spezia, Italy, in 2019. In 2021 he joined the faculty of Dipartimento di Elettronica, Informazione e Bioingegneria (DEIB) at the Politecnico di Milano as Research Fellow. His research interests include signal processing, statistical learning, and data fusion for cooperative localization and communication.

Dr. Brambilla was the recipient of the Best Student Paper Award at the 2018 IEEE Statistical Signal Processing Workshop.

Monica Nicoli (Senior Member, IEEE) received the M.Sc. (Hons.) and Ph.D. degrees in communication engineering from Politecnico di Milano, Milan, Italy, in 1998 and 2002, respectively. She was a Visiting Researcher with ENI Agip, from 1998 to 1999, and Uppsala University, in 2001. In 2002, she joined Politecnico di Milano as a Faculty Member. She is currently an Associate Professor in telecommunications with the Department of Management, Economics and Industrial Engineering.

Her research interests include signal processing, machine learning, and wireless communications, with emphasis on smart mobility and Internet of Things (IoT). She was a recipient of the Marisa Bellisario Award, in 1999, and a co-recipient of the best paper awards of the EuMA Mediterranean Microwave Symposium, in 2022, the IEEE Symposium on Joint Communications and Sensing, in 2021, the IEEE Statistical Signal Processing Workshop, in 2018, and the IET Intelligent Transport Systems journal, in 2014. She is an Associate Editor of the IEEE Transactions on Intelligent Transportation Systems. She has also served as an Associate Editor for the EURASIP Journal on Wireless Communications and Networking, from 2010 to 2017, and a Lead Guest Editor for the Special Issue on Localization in Mobile Wireless and Sensor Networks, in 2011.

Moe Z. Win (Fellow, IEEE) is a Professor at the Massachusetts Institute of Technology (MIT) and the founding director of the Wireless Information and Network Sciences Laboratory. Prior to joining MIT, he was with AT&T Research Laboratories and with NASA Jet Propulsion Laboratory.

His research encompasses fundamental theories, algorithm design, and network experimentation for a broad range of real-world problems. His current research topics include ultra-wideband systems, network localization and navigation, network interference exploitation, and quantum information science. He has served the IEEE Communications Society as an elected Member-at-Large on the Board of Governors, as elected Chair of the Radio Communications Committee, and as an IEEE Distinguished Lecturer. Over the last two decades, he held various editorial positions for IEEE journals and organized numerous international conferences. He has served on the SIAM Diversity Advisory Committee.

Dr. Win is an elected Fellow of the AAAS, the EURASIP, the IEEE, and the IET. He was honored with two IEEE Technical Field Awards: the IEEE Kiyo Tomiyasu Award (2011) and the IEEE Eric E. Sumner Award (2006, jointly with R. A. Scholtz). His publications, co-authored with students and colleagues, have received several awards. Other recognitions include the MIT Frank E. Perkins Award (2024), the MIT Everett Moore Baker Award (2022), the IEEE Vehicular Technology Society James Evans Avant Garde Award (2022), the IEEE Communications Society Edwin H. Armstrong Achievement Award (2016), the Cristoforo Colombo International Prize for Communications (2013), the Copernicus Fellowship (2011) and the *Laurea Honoris Causa* (2008) from the Università degli Studi di Ferrara, and the U.S. Presidential Early Career Award for Scientists and Engineers (2004).

Part III

Cooperative Inference

CHAPTER 7

Efficient Distribution Sampling for NLoS Identification

In this chapter, we present a work on efficient distribution sampling for NLoS identification in next-generation cellular networks, modelling the problem as an anomaly detection task. In particular, we address the limitations of current semi-supervised learning methods in providing a precise and compact latent feature representation that does not require two-stage training and holding the entire training dataset for prediction. To this aim, we propose a DAKDM composed of an AE, a KDE and a likelihood network. The AE permits to have a compact representation of the input, i.e., ADCPM, by minimizing the reconstruction error. On the contrary, the likelihood network is trained to mimic the KDE output adopting the VI paradigm. At inference phase, the decoder and KDE parts are discarded, and only the encoder and likelihood networks are adopted for anomaly score estimation. The BSs are trained according to the C-ML paradigm by only employing LoS data, i.e., normal data, whereas they are tested to distinguish NLoS samples, i.e., anomalous data, from the learned LoS distribution. Comparisons with statistical and DL methods for anomaly detection show that the proposed DAKDM is able to have similar performances to best state-of-the-art methods while being significantly more efficient in terms of storage requirements and inference time.

IEEE JOURNAL ON SELECTED AREAS IN COMMUNICATIONS, VOL. 41, NO. 6, JUNE 2023 1655

On the Latent Space of mmWave MIMO Channels for NLOS Identification in 5G-Advanced Systems

Bernardo Camajori Tedeschini, *Graduate Student Member, IEEE*, Monica Nicoli, *Senior Member, IEEE*, and Moe Z. Win, *Fellow, IEEE*

***Abstract*— In mission-critical verticals such as automated driving, 5G-advanced networks must provide centimeter-level dynamic positioning along with ultra-reliable low-latency communication services. Massive Multiple-Input Multiple-Output (mMIMO) and millimeter waves (mmWave) are the key enablers, allowing high accuracy angle and delay estimation. Still, extracting such information from highly-dimensional Channel Impulse Responses (CIRs) results in a complex task, due to channel sparsity and intermittent blockage. In this paper we focus on non-line-of-sight (NLOS) identification from CIR data, proposing a Deep Autoencoding Kernel Density Model (DAKDM) to characterize the statistics of the channel latent features. We formulate the problem as a semi-supervised anomaly detection task in which only LOS samples, i.e., normal data, are adopted for training. DAKDM is a single-stage training model that takes as input the full CIR thanks to an AutoEncoder (AE) structure. The proposed method is able to learn the latent distribution by means of a Kernel Density Estimator (KDE) in combination with a deep learning likelihood network. We validate the proposed solution in a 5G Urban micro (UMi) vehicular scenario. Results show that the proposed model can significantly outperform conventional algorithms and obtain similar performances to variational Bayes algorithms at one tenth of the inference time.**

***Index Terms*— Deep autoencoding kernel density model, anomaly detection, CIR, 5G, deep learning, NLOS identification.**

I. Introduction

THE newest release of the 5th generation (5G) of cellular communication systems, namely the 3rd generation partnership project (3GPP) Release 16, also known as new radio (NR), introduces for the first time high-precision positioning functionalities into cellular networks. location services (LCS) are extended from regulatory services to roaming and commercial capabilities [1], [2], [3], [4]. Higher frequencies, bandwidth improvements and massive-multiple-input multiple-output (MIMO) technologies are the key feature enablers for radio access technology (RAT)-dependent dynamic positioning [2], [3], [4], [5] and location awareness of connected nodes [6], [7], [8], [9], [10], [11]. The major fields of application can be found in target tracking [12], [13], [14], internet-of-things (IoT) [15], [16], [17], crowd sensing [18], [19], smart environments [20] and industrial automation [21]. Strict requirements are foreseen for the most critical services such as automated driving [22], [23]. These include a lateral and longitudinal positioning error of 10 and 50 cm [24], respectively, and a latency down to 5 ms for fully autonomous driving vehicles [25]. Next 5G releases, also known as 5G-Advanced and beyond, will have to meet such challenging localization requirements while coping with complex propagation conditions, due to the extreme path-loss and frequent blockage conditions experienced by millimeter waves (mmWave).

These problems have been widely studied in the field of localization and navigation focusing on fundamental performance limits [26], [27], [28], [29], [30], [31], algorithm design [32], [33], [34], [35], [36], [37], [38], network operation [39], [40], [41], [42], and network experimentation [43], [44], [45], [46], [47], [48]. It is clear that legacy solutions for positioning, based on conventional approaches for multi-lateration/angulation, will struggle to deal with rapidly fading channels and intermittent blockage. Geometrical approaches rely in fact on line-of-sight (LOS) condition for estimating directions and ranges of the positioning reference signals. Real-time detection and prediction of non-LOS (NLOS) links is mandatory to mitigate the false localization due to biased range/angle estimates. Since the environment significantly impacts on the propagation, data-driven techniques have so far produced very encouraging outcomes in NLOS detection [48], [49]. Therefore, machine learning (ML) is expected to play a crucial role in future generation networks [50], [51] and standard compliant solutions are foreseen already from Release 17-18 [52].

Solutions for blockage detection should exploit the whole power-delay-angle profile of the channel impulse response (CIR) as this embeds a wide range of geographical data and propagation characteristics [53]. In 5G industrial use-cases, e.g., automated driving, historical CIR data are largely available in roadside units that receive continuous information from geolocalized vehicles [54], [55], [56], [57]. ML algorithms

Manuscript received 13 October 2022; revised 28 March 2023; accepted 13 April 2023. Date of publication 8 May 2023; date of current version 19 June 2023. The fundamental research described in this paper was supported in part by the Ph.D. Grant from the Ministry of the Italian Government Ministero dell'Istruzione, dell'Università e della Ricerca (MIUR), in part by the Roberto Rocca Doctoral Fellowship granted by the Massachusetts Institute of Technology and Politecnico di Milano, in part by the National Science Foundation under Grant CNS-2148251, and in part by the Federal Agency and Industry Partners in the RINGS Program. *(Corresponding author: Bernardo Camajori Tedeschini.)*

Bernardo Camajori Tedeschini is with Dipartimento di Elettronica, Informazione e Bioingegneria (DEIB), Politecnico di Milano, 20133 Milan, Italy (e-mail: bernardo.camajori@polimi.it).

Monica Nicoli is with Dipartimento di Ingegneria Gestionale (DIG), Politecnico di Milano, 20156 Milan, Italy (e-mail: monica.nicoli@polimi.it).

Moe Z. Win is with the Laboratory for Information and Decision Systems (LIDS), Massachusetts Institute of Technology, Cambridge, MA 02139 USA (e-mail: moewin@mit.edu).

Color versions of one or more figures in this article are available at https://doi.org/10.1109/JSAC.2023.3273769.

Digital Object Identifier 10.1109/JSAC.2023.3273769

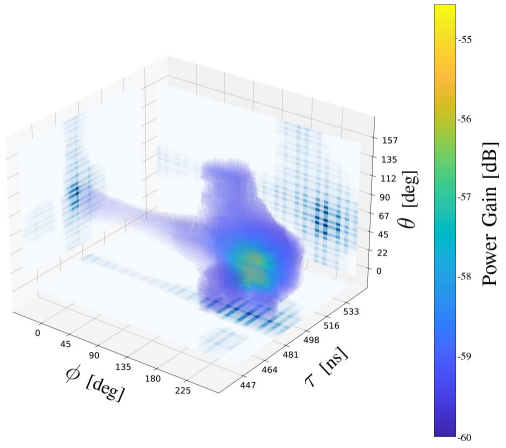


Fig. 1. Sparse channel representation in azimuth ϕ, elevation θ and delay τ domain.

could easily exploit these data for automatic NLOS detection. However, since such signals are highly dependent on the environment, ML approaches for detecting NLOS using CIRs frequently fail to generalize to varied contexts [58]. Moreover, massive MIMO and very high frequencies of advanced-5G networks will produce high dimensional channel responses which may be complex to handle. An example of channel power-delay-angle-profile is shown in Fig. 1, for a 5G urban scenario with carrier frequency 30 GHz, bandwidth 400 MHz and uniform planar antenna array receiver of 64 elements. The sparse power delay-angle profile of the channel is a signature of the user location and should be exploited to infer the visibility conditions of the base station.

In this paper, we propose an innovative strategy to characterize the sparsity of the mmWave MIMO channel and approximate whatever high-dimensional distribution in a fast and compact way. To demonstrate the efficacy of the method, we address the problem of NLOS identification, exclusively employing LOS CIRs for training. This is done because LOS CIRs are easier to extract in training procedures and present more peculiar distributions, i.e., usually the direct path is the dominant factor in a Rician fading channel. In addition, this facilitates the deployment and results in higher generalization compared to other systems that require both classes for training (i.e., LOS and NLOS). Therefore, we treat the problem as an anomaly detection case in which LOS samples are considered as normal samples, while NLOS samples are anomalous.

II. Related Works

In this section, we first review the literature starting from early works on ultra wide-band (UWB) systems (Sec. II-A) and then we extend the analysis to ML-based algorithms (Sec. II-B). Next, we review the state of the art on anomaly detection focusing on neural network (NN) approaches (Sec. II-C) and we discuss the original contributions provided in the paper (Sec. II-D).

A. NLOS Identification

Existing techniques for NLOS identification/detection problem can be mainly divided into three major categories: based on range estimates, based on position estimates and based on channel statistics. The first group of methods, i.e., based on range estimates, measures the running variance of the ranges and applies a threshold using pre-computed variance statistics [59]. The techniques based on position estimates are mainly map-based, i.e., they observe the user equipment (UE) position in relation to the geometry of the environment [60], [61]. These first two categories are either too oversimplified or require perfect knowledge of the UE's position and of the map geometry.

The third class relies on channel statistics, such as amplitude, mean and root-mean-square delay. In case these statistics are known at-priori, a joint-likelihood ratio test can be adopted for hypothesis selection [62], [63] or as soft information in weighted least squares (WLS) algorithms. The limitations of this last class of existing techniques include experiencing delays while gathering channel statistics to create a database and determining the complex combined probability distributions of necessary features for statistical methods. ML-based approaches overcome these drawbacks by avoiding statistical modeling of the input features.

B. ML for NLOS Identification

ML approaches to NLOS identification can be divided into: supervised, unsupervised and semi-supervised learning. First works (i.e., supervised learning) use hand-crafted channel state information (CSI) features such as energy, maximum rise time, kurtosis, root-mean-square-delay spread, maximum amplitude, time of flight (ToF), Ricean-K-factor and mean excess delay [48], [64], [65], [66]. These deterministic features have a solid theoretical basis and capture the differences between LOS and NLOS conditions in terms of power and delay attributes, as well as the strength of the dominant signal component relative to the multipath components. The most popular adopted ML models are support vector machines (SVMs), relevance vector machines (RVMs), random forests (RFs) and Gaussian processes (GP). These methods can also be used to directly mitigate the range bias by applying a regression problem to the ranging-error estimates [49], [67].

Despite achieving good results, these methods highly depend on the pre-selected features which limit their potential. On the other hand, deep learning (DL) approaches can directly learn the most suitable combination of features (typically non-linear) using as input the full CIR and producing as output the desired classification. First works in this direction can be found in [68], [69], [70] using convolutional neural networks (CNN) to perform feature extractions in grid-like data where local patterns and structures are critical. Some recent studies [71], [72] directly exploit the automatic feature extraction of the CNN in order to locate a target by performing a fingerprint training. A main limit is the need of extensive measurement campaigns and time-consuming labeling of data. Moreover, supervised learning approaches require updating the training

database when conditions are changed and need representative samples of all the possible NLOS anomalies.

A solution could be permitting not to have labels at all and manage the problem as an unsupervised one. Authors in [73] fit a Gaussian mixture model (GMM) with two components (one for LOS and one for NLOS) using some key handcrafted features of the channel and output the classification according to the magnitude of the membership weights. While unsupervised techniques are very promising, unfortunately they do not achieve very high performances, due to lack of knowledge or lack of structured data.

The third class of semi-supervised approaches has the advantage of not needing examples of all the possible anomalies as supervised learning. Moreover, powerful DL semi-supervised methods focus on learning one single distribution which, in many cases, is easier than a separating boundary between two distributions [74]. Works that adopt this strategy can be found in [58], [75] which adopt the Pearson correlation coefficient and one-class SVM to perform NLOS classification, respectively. A very recent work [76] employs variational autoencoder (VAE) to perform feature extraction and imposes a Gaussian distribution to the latent features in order to ease learning of distribution of normal samples. The score adopted to define the probability of NLOS is then used to estimate the bias and variance of time-based measurements. Although the idea of using an autoencoder (AE) to have a compact representation of the channel can give very good results, the usage of sampling-based methods to perform the prediction has the main drawback of not being suitable for real-time applications.

C. Neural Networks for Anomaly Detection

Anomaly detection is frequently employed in problems where we have a large amount of data from normal circumstances but little data from abnormal ones. Here, on the other hand, we consider the setting of semi-supervised learning in which normal training data only are provided. In this case, the problem turns out to be locating those samples that do not conform to the normal ones or a model explaining normal ones. Thus, the objective is to learn in a finer way as possible the distribution related to the normal samples.

To this aim, many works focus on end-to-end models to directly produce the classification using one-class neural networks (OC-NN) [77]. On the other hand, generative models are increasing in popularity with generative adversarial network (GAN) and VAE [78]. However, GANs are problematic to control in the training phase [79] and VAEs have the downside of requiring sampling, which is unfeasible under certain use cases, and furthermore experiments have shown that they tend to perform worse than AE [80], [81].

Reconstruction methods, as AE, are the most widely used DL techniques for anomaly detection in images [82]. Usually, they are used in combination with density-based methods, as kernel density estimation (KDE) [83], for score estimation by first performing dimensionality reduction, and then applying density estimation to the latent low-dimensional space. However, these two-steps methods restrict the modification to the dimensionality reduction since fine-tuning is difficult in well-trained AE. To solve this problem, authors in [84] propose a deep autoencoding Gaussian mixture model (DAGMM) to mutually learn the latent feature representations and their density under the GMM framework by mixture membership estimation. Even though their approach is direct and does not require two step-training, GMM may not be able to fully represent the latent distribution of normal samples and are subject to singularity problems. On the other hand, KDE are perfect to represent complex distributions, but they are very slow in evaluation and require storing the whole dataset for inference.

D. Contributions

In this paper, we address the problem of NLOS identification in 5G-advanced cellular systems using an innovative approach that allows to overcome the aforementioned limitations. The main contributions are the following:

- We propose a feature extraction method that exploits the angle-delay channel power matrix (ADCPM) as input data and allows to characterize the distributions of the latent features of the sparse space-time channel in massive MIMO cellular systems using orthogonal frequency division multiplexing (OFDM).
- We design NLOS identification as a semi-supervised anomaly-detection problem by exploiting a deep autoencoding kernel density model (DAKDM). The DL model allows to identify the few key parameters that describe the sparse space-time channel response and to learn the distributions of such latent features from training data. The proposed approach is able to jointly learn the sparse channel representation and approximate the KDE likelihood in a single training stage without storing the dataset.
- We simulate a realistic 5G-advanced MIMO-OFDM vehicular scenario, according to the standard specifications [85], using a Matlab ray-tracing software [86]. The scenario is composed of multiple vehicular UEs created with simulation of urban mobility (SUMO) software [87].

The paper is organized as follows: Sec. III introduces the channel model for a multi-user MIMO-OFDM system and its extracted fingerprinting. Sec. IV provides the context of anomaly detection applied to the NLOS identification and defines the proposed DAKDM solution. Sec. V is devoted to the description of the simulated 5G scenario and to the comparison with state-of-the-art anomaly detection DL methods. Finally, Sec. VI draws the conclusion.

E. Notation

Random variables are displayed in sans serif, upright fonts; their realizations in serif, italic fonts. Vectors and matrices are denoted by bold lowercase and uppercase letters, respectively. For example, a random variable and its realization are denoted by x and x; a random vector and its realization are denoted by $\mathbf{x}$ and $\boldsymbol{x}$; a random matrix and its realization are denoted by $\mathbf{X}$ and $\boldsymbol{X}$, respectively. Random sets and their realizations are denoted by up-right sans serif and calligraphic font, respectively. For example, a random

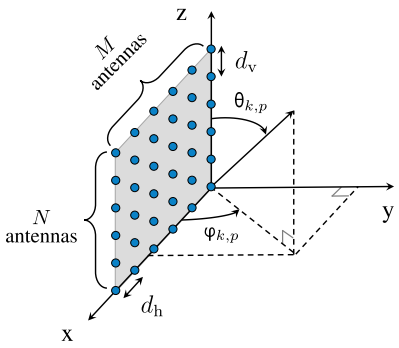


Fig. 2. Uniform planar array with M and N antennas in x and z directions, respectively. The direction of arrival (DoA) is highlighted and decomposed into azimuth $0 \leq \varphi_{k,p} < \pi$ and elevation $0 \leq \theta_{k,p} < \pi$.

set and its realization are denoted by X and $\mathcal{X}$, respectively. The function $p_{\mathsf{x}}(x)$, and simply $p(x)$ when there is no ambiguity, denotes the probability density function (PDF) of x. $j = \sqrt{-1}$ denotes the imaginary unit. The notation $\mathbf{X}^{\mathrm{T}}$, $\mathbf{X}^{*}$ and $\mathbf{X}^{\mathrm{H}}$ indicate the matrix transposition, conjugation and conjugate transposition. The Kronecker and the Hadamard product between two matrices are denoted with the symbols $\otimes$ and $\odot$, respectively. With the notation $\mathsf{x} \sim \mathcal{CN}(\mu, \sigma^2)$ we indicate a complex Gaussian random variable x with mean μ and standard deviation σ. We use $\mathbb{E}\{\cdot\}$ and $\mathbb{V}\{\cdot\}$ to denote the expectation and the variance of random variable, respectively. $\mathbb{R}$ and $\mathbb{C}$ stand for the set of real and complex numbers, respectively. $\mathrm{Re}(x)$ and $\mathrm{Im}(x)$ are the real and complex part of the complex number x, respectively. $\lfloor x \rfloor$ indicates the largest integer not greater than x, while $\delta(\cdot)$ and $\delta[\cdot]$ are the Dirac delta and Kronecker functions, respectively.

III. System Model

A. Channel Model

We consider a multi-user mmWave MIMO-OFDM system in which K UEs transmit in uplink direction over a bandwidth B at carrier wavelength λ_{c}. The base station (BS)'s cell panel is equipped with an uniform planar array (UPA) with $N \times M$ isotropic antennas. The antenna spacings are d_{h} and d_{v}, over the horizontal and vertical dimension, respectively. We assume that the UE transmits using only one logical port and a number of physical antennas unknown at the BS. Between the UE $k = 1, \ldots, K$ and the BS, we consider a multipath channel with N_k paths with ToF $\tau_{k,p}$ for path $p = 1, \ldots, N_k$. The DoAs from the k-th UE and of the p-th path are divided into azimuth $0 \leq \varphi_{k,p} < \pi$ and elevation $0 \leq \theta_{k,p} < \pi$. A picture of the panel array can be found in Fig. 2. We restrict the azimuth up to π since we consider an UPA antenna. For tri-sectorial BSs the angular coverage is reduced to $2\pi/3$.

The OFDM modulation is performed over N_{c} sub-carriers, sampling interval T_{s} and symbol duration $T_{\mathrm{c}} = N_{\mathrm{c}} T_{\mathrm{s}}$. Considering a baseband representation of the signal, we define the frequency at the ℓ-th sub-carrier as $f_\ell = \frac{\ell}{T_{\mathrm{c}}}$, $\ell = 0, \ldots, N_{\mathrm{c}} - 1$. The cyclic-prefix duration is $T_{\mathrm{g}} = N_{\mathrm{g}} T_{\mathrm{s}}$ and it is assumed to be larger than the maximum channel delay for all UEs $\tau_{\mathrm{MAX}} = \max_{k,p} \tau_{k,p}$. Consequently, we define with $\mathsf{r}_{k,p} = \lfloor \frac{\tau_{k,p}}{T_{\mathrm{s}}} \rceil$ the temporally resolvable propagation delay of the p-th path with respect to the k-th UE. Thus, the baseband CIR of user k is modelled as [88]:

$$\mathbf{h}_k(\tau) = \sum_{p=1}^{N_k} \mathsf{a}_{k,p}\, \beta_{k,p}\, \mathbf{e}(\theta_{k,p}, \varphi_{k,p}) e^{-j2\pi \frac{\mathsf{d}_{k,p}}{\lambda_{\mathrm{c}}}} \delta(\tau - \tau_{k,p}), \quad (1)$$

where the p-th path is characterized by a complex path gain $\alpha_{k,p} = \mathsf{a}_{k,p} e^{-j2\pi \frac{\mathsf{d}_{k,p}}{\lambda_{\mathrm{c}}}} \beta_{k,p}$ with $\beta_{k,p} = e^{j2\pi \nu_{k,p} t}$ due to the Doppler frequency shift, a traveled distance $\mathsf{d}_{k,p} = c\tau_{k,p}$, a pulse waveform approximated with a Dirac delta function $\delta(\tau - \tau_{k,p})$ and an array response vector $\mathbf{e}(\theta_{k,p}, \varphi_{k,p}) \in \mathbb{C}^{MN}$. For $p > 1$, the p-th path is $\alpha_{k,p} = \mathsf{a}_{k,p} e^{j\psi_{k,p}}$, with $\psi_{k,p} = 2\pi \nu_{k,p} t - 2\pi \frac{\mathsf{d}_{k,p}}{\lambda_{\mathrm{c}}}$ and $\alpha_{k,p} \sim \mathcal{CN}(0, \sigma^2_{k,p})$. The first path $p = 0$ is $\alpha_{k,0} \sim \mathcal{CN}(s_0 \mathsf{a}_{k,0} e^{j\psi_{k,0}}, \sigma^2_{k,0})$ where it is $s_0 = 1$ for LOS (i.e., with a deterministic direct path contribution and Rician fading) and $s_0 = 0$ for NLOS (i.e., Rayleigh fading). Additionally, we consider the Doppler-related rotation to be almost constant over time interval τ_{MAX} and that the complex amplitudes $\alpha_{k,p}$ associated to different paths as uncorrelated, according to the wide-sense stationary uncorrelated scattering model. At the BS, the array response vector can be decomposed into [89]:

$$\mathbf{e}(\theta_{k,p}, \varphi_{k,p}) = \mathbf{e}_{\mathrm{v}}(\theta_{k,p}) \otimes \mathbf{e}_{\mathrm{h}}(\theta_{k,p}, \varphi_{k,p}), \quad (2)$$

where the $M \times 1$ response vector to the elevation angle is:

$$\mathbf{e}_{\mathrm{v}}(\theta_{k,p}) = \left[e^{-j2\pi(m-1)\frac{d_{\mathrm{v}}}{\lambda_{\mathrm{c}}} \cos(\theta_{k,p})} \right]_{m=1}^{M} \quad (3)$$

and the $N \times 1$ response vector to the azimuth angle is:

$$\mathbf{e}_{\mathrm{h}}(\theta_{k,p}, \varphi_{k,p}) = \left[e^{-j2\pi(n-1)\frac{d_{\mathrm{h}}}{\lambda_{\mathrm{c}}} \sin(\theta_{k,p}) \cos(\varphi_{k,p})} \right]_{n=1}^{N}. \quad (4)$$

Adopting an OFDM modulation with sampling at $t = nT_{\mathrm{s}}$, the channel frequency response (CFR) at the ℓ-th sub-carrier can be written as the discrete Fourier transform (DFT) of the CIR of the different paths [90], [91]:

$$\begin{aligned} \mathbf{H}_{k,\ell} &\approx \sum_{n=0}^{N_{\mathrm{g}}-1} \sum_{p=1}^{N_k} \alpha_{k,p}\, \mathbf{e}(\theta_{k,p}, \varphi_{k,p})\, \delta[n - \mathsf{r}_{k,p}]\, e^{-j2\pi \tau_{k,p} f_\ell} \\ &= \sum_{p=1}^{N_k} \alpha_{k,p}\, \mathbf{e}(\theta_{k,p}, \varphi_{k,p})\, e^{-j2\pi \frac{\ell \mathsf{r}_{k,p}}{N_{\mathrm{c}}}}, \end{aligned} \quad (5)$$

where the approximation holds for ToFs multiple of the sampling interval T_{s} or equivalently for $N_{\mathrm{g}} \to \infty$. Finally, the space-frequency channel response matrix (SFCRM) $\mathbf{H}_k \in \mathbb{C}^{MN \times N_{\mathrm{c}}}$ of the k-th UE is obtained as:

$$\mathbf{H}_k = [\mathbf{H}_{k,0}\ \mathbf{H}_{k,1}\ \ldots\ \mathbf{H}_{k,N_{\mathrm{c}}-1}], \quad (6)$$

which will be used in the next section to extract the channel fingerprints.

B. Channel Fingerprints

To detect the propagation conditions that generated the response (5), classifying them in LOS or NLOS, we propose to analyze the CFR in the angle-delay domain, which eases the recognition of the clustered multipath components associated to the direct (LOS) or secondary (NLOS) macro-paths. We thereby convert the SFCRM (6) into the domain of the angle of arrival (AoA) and the ToF, by introducing the angle-delay channel response matrix (ADCRM). We define with $\boldsymbol{V}_M \in \mathbb{C}^{M\times M}$ and $\boldsymbol{V}_N \in \mathbb{C}^{N\times N}$ the phase-shifted DFT matrices [92] where $[\boldsymbol{V}_M]_{u,v} = \frac{1}{\sqrt{M}} e^{-j2\pi \frac{u(v-\frac{M}{2})}{M}}$ and $[\boldsymbol{V}_N]_{u,v} = \frac{1}{\sqrt{N}} e^{-j2\pi \frac{u(v-\frac{N}{2})}{N}}$. Then, we denote by $\boldsymbol{F} \in \mathbb{C}^{N_\mathrm{c}\times N_\mathrm{g}}$ the matrix formed by the first N_g columns of N_c dimensional unitary DFT matrix where $[\boldsymbol{F}]_{u,v} = \frac{1}{\sqrt{N_\mathrm{c}}} e^{-j2\pi \frac{uv}{N_\mathrm{c}}}$. ADCRM is computed as [93]:

$$\mathbf{G}_k = \frac{1}{\sqrt{MNN_\mathrm{c}}} (\boldsymbol{V}_M^\mathrm{H} \otimes \boldsymbol{V}_N^\mathrm{H})\, \mathbf{H}_k \boldsymbol{F}^* \in \mathbb{C}^{MN\times N_\mathrm{g}}, \quad (7)$$

where $(\boldsymbol{V}_M^\mathrm{H} \otimes \boldsymbol{V}_N^\mathrm{H})$ and $\boldsymbol{F}^*$ project the SFCRM into the angle and delay domain, respectively.

For NLOS identification, we propose to use the ADCRM to compute the average power of the channel components that are collected into the ADCPM defined as:

$$\mathbf{P}_k = \mathbb{E}\{\mathbf{G}_k \odot \mathbf{G}_k^*\} \in \mathbb{C}^{MN\times N_\mathrm{g}}, \quad (8)$$

where $[\mathbf{P}_k]_{i,r} = \mathbb{E}\{|[\mathbf{G}_k]_{i,r}|^2\}$. We recall here that the ADCPM holds some important asymptotic properties, as for N, M and $N_\mathrm{g} \to \infty$, it tends to be a sparse matrix with elements $[\mathbf{P}_k]_{i,r}$ matching the i-th AoA and the r-th ToF [93]:

$$\lim_{M,N,N_\mathrm{g}\to\infty} [\mathbf{P}_k]_{i,r} = \sum_{p=1}^{N_k} \sigma_{k,p}^2 \delta[i - m_{k,p}N - n_{k,p}]\delta[r - \mathsf{r}_{k,p}], \quad (9)$$

where $m_{k,p}N + n_{k,p}$ denotes the index of the i-th AoA and $\mathsf{r}_{k,p}$ the index of the r-th ToF. Note that the angle and delay indexes $m_{k,p}N + n_{k,p}$ and $\mathsf{r}_{k,p}$, are two distinct and discrete quantities which relates to the physical AoA and ToF in the following way. The ToF can be approximated as $\tau_{k,p} = \mathsf{r}_{k,p} T_\mathrm{s}$, while the azimuth $\varphi_{k,p}$ and elevation $\theta_{k,p}$ can be written as $\varphi_{k,p} = \arccos\left(\frac{m_{k,p} - \frac{M}{2}}{M} \frac{\lambda_\mathrm{c}}{d_\mathrm{h}}\right)$ and $\theta_{k,p} = \arccos\left(\frac{n_{k,p} - \frac{N}{2}}{N} \frac{\lambda_\mathrm{c}}{d_\mathrm{v}}\right)$, respectively. Consequently, working in the transformed angle-delay domain allows the DL model to learn the location-dependent features and, therefore, the statistics of LOS data to be used for blockage prediction.

Regarding the complexity overhead due to the ADCPM computation, we observe that $\mathbf{P}_k$ is obtained from the channel matrix $\mathbf{H}_k$ which is always estimated for communications purposes. Therefore the only overhead is the computation of (7), which can be efficiently performed using the 2D-Inverse Fast FT (IFFT), with an overall complexity of $O(MNN_\mathrm{g} \cdot \log(MNN_\mathrm{g}))$.

IV. Blockage Detection Methodology

In this section, we first introduce the problem formulation of the semi-supervised setting which serves for the proposed DL model's foundation. Then, we describe the network input, i.e., the ADCPM fingerprint, followed by the definition of the DAKDM. Finally, we define the loss function used to train the model.

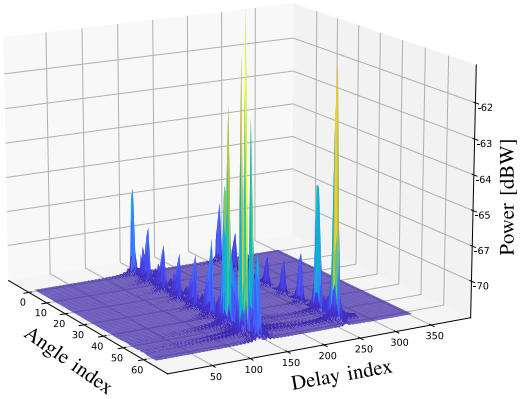


Fig. 3. Example of a sparse ADCPM with $M = N = 8$ antennas at the BS and $N_\mathrm{g} = 352$ cyclic prefix duration in terms of sampling intervals T_s, simulated in an urban road environment with ray-tracing software.

A. Problem Formulation

We consider a semi-supervised setting in which we are given a training dataset $\mathcal{S}^{\text{train}}$ comprising only normal data, i.e., $\boldsymbol{X}_i$ sampled from $p_{\mathbf{X}}$, and a smaller testing data $\mathcal{S}^{\text{test}}$ comprising normal (label $y_i = 0$) and anomalous data (label $y_i = 1$). Here, we refer to LOS samples as normal, while we consider NLOS samples as anomalous. Nevertheless, the choice of normal/anomalous condition is arbitrary and could be customized to the specific scenario, as the proposed model would still be valid in both cases, i.e., LOS or NLOS samples as normal data. Usually, the high-dimensional distribution of normal samples $p_{\mathbf{X}}$ is complex and unknown. Thus, the objective is to first elaborate $\mathcal{S}^{\text{train}}$ such that we can learn its manifold distribution and, subsequently, during inference time, identify the anomalous samples in $\mathcal{S}^{\text{test}}$ as outliers. The mapping of the high-dimensional data is carried out using a DL model $f(\cdot)$ that learns the normal data distribution while also attempting to reduce an anomaly score $A(\boldsymbol{X}_i)$ given as output. The higher the anomaly score of $A(\boldsymbol{X}_i)$ for a test sample $\boldsymbol{X}_i$, the higher probability that $\boldsymbol{X}_i$ is anomalous. For evaluation, a threshold (η) criterion is applied, i.e., $A(\boldsymbol{X}_i) > \eta$ denotes an anomaly, based on a predefined false positive rate (FPR).

B. DL Input

We employ (9) as input to the neural network for NLOS identification, as this matrix represents the clustered multipath structure of the channel and embeds the information on LOS/NLOS propagation conditions that we are interested to extract. Moreover, the sparsity of the matrix helps the CNN in features extraction since the first layers of CNN are usually sparse and they gather the more discriminant features [94]. In Fig. 3 we can see the ADCPM $\mathbf{P}_k$ composed by MN angle

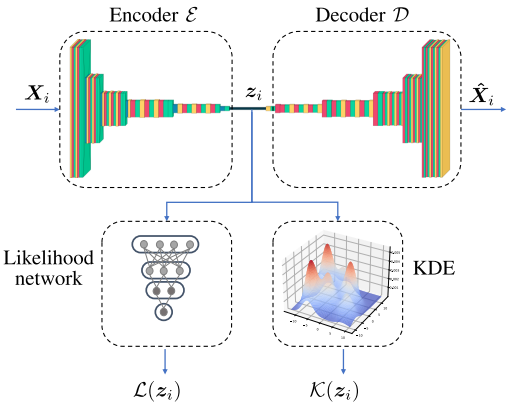


Fig. 4. Structure of the proposed deep autoencoding kernel density model (DAKDM) composed by an autoencoder (AE), a kernel density estimation (KDE) and a likelihood network.

directions and N_{g} delay samples. The sparsity of the matrix is well-visible even without a huge number of antennas or sample's resolution. From now on, for simplicity of notation, we drop the index k related to user k and we denote the i-th input sample as $\boldsymbol{X}_i = \boldsymbol{P}_i$.

C. Deep Autoencoding Kernel Density Model

The proposed DAKDM system for NLOS identification involves three main elements: an AE, a KDE model and a likelihood network. The model can be seen in Fig. 4. The AE comprises an encoder $\mathcal{E}(\cdot)$, that elaborates the i-th input $\boldsymbol{X}_i \in \mathbb{R}^{MN \times N_{\mathrm{g}}}$ into a latent representation $\boldsymbol{z}_i \in \mathbb{R}^m$, and a decoder $\mathcal{D}(\cdot)$, that carries out an inverse transformation to return to the original high-dimensional distribution, obtaining $\hat{\boldsymbol{X}}_i$. The latent distribution $p_{\mathbf{z}}$ may have any form, i.e., it is not constrained to belong to any specific PDF family. This makes the proposed approach general enough to be applied to any channel environment.

The distribution $p_{\mathbf{z}}$ is automatically learned by the KDE block of the system (see Fig. 4). The KDE is a non-parametric method to estimate any distribution directly from a set of samples drawn from it. Given a set of samples $\{\boldsymbol{z}_j\}_{j=1}^{N_{\mathrm{s}}}$ from $p_{\mathbf{z}}$, we define the KDE $\mathcal{K}(\cdot)$ applied to sample $\boldsymbol{z}_i$ as [95]:

$$\mathcal{K}(\boldsymbol{z}_i|\{\boldsymbol{z}_j\}_{j=1}^{N_{\mathrm{s}}}) = \frac{1}{N_{\mathrm{s}}} \sum_{j=1}^{N_{\mathrm{s}}} k_h(\boldsymbol{z}_i - \boldsymbol{z}_j), \tag{10}$$

where $k_h : \mathbb{R}^m \to \mathbb{R}$ is a kernel function with bandwidth h which regulates the balance between the estimator's variance and bias. The kernel employed in this paper is the widely known Gaussian kernel:

$$k_h(\boldsymbol{x}) = e^{-\frac{|\boldsymbol{x}|^2}{2h^2}}. \tag{11}$$

The output of a KDE, trained only with normal latent samples $\{\boldsymbol{z}_j\}_{j=1}^{N_{\mathrm{s}}}$, can be seen as the likelihood of the test sample to belong to the normal distribution. Thus, the derived anomaly score can be obtained as $A_{\mathcal{K}}(\boldsymbol{z}_i) = -\log(\mathcal{K}(\boldsymbol{z}_i|\{\boldsymbol{z}_j\}_{j=1}^{N_{\mathrm{s}}}))$. However, the downsides of KDE lie in the fact that it requires storing all the training dataset to estimate the density function at inference time.

Algorithm 1 Mini-Batch Training Procedure

procedure TRAINING(batch size N_{s}) ▷ Batch number j
  for $i = 1, 2, \ldots, N_{\mathrm{s}}$ **do**
    Encode incoming signal $\boldsymbol{X}_i$: $\boldsymbol{z}_i = \mathcal{E}(\boldsymbol{X}_i)$.
    Compute anomaly score: $A_{\mathcal{L}}(\boldsymbol{z}_i) = -\log(\mathcal{L}(\boldsymbol{z}_i))$.
    Compute KDE prediction: $A_{\mathcal{K}}(\boldsymbol{z}_i) = -\log(\mathcal{K}(\boldsymbol{z}_i^j|\{\boldsymbol{z}_l^{j-1}\}_{l=1}^{N_{\mathrm{s}}}))$.
  end for
  Compute loss function $\mathrm{L}_{\mathrm{tot}}^j$.
  Perform backward-pass.
end procedure

The idea to solve this issue is to first reduce the number of samples N_{s} used to estimate the distribution, and then approximate the output of the KDE with a NN that is much faster in the prediction. We call the NN to estimate the output of the KDE as likelihood network and denote it with $\mathcal{L}(\cdot)$. The logical steps for the training procedure with a batch of N_{s} samples are described in Algorithm 1. First, we encode the input with the encoder. Then, we extract the anomaly score as $A(\boldsymbol{X}_i) \triangleq A_{\mathcal{L}}(\boldsymbol{z}_i) = -\log(\mathcal{L}(\boldsymbol{z}_i))$ and we compute the KDE prediction $A_{\mathcal{K}}(\boldsymbol{z}_i)$. Finally, we compute the loss function which is described in Sec. IV-D and perform the backward pass. The key aspect here is that the KDE output is computed using the previous mini-batch, i.e., $\mathcal{K}(\boldsymbol{z}_i^j|\{\boldsymbol{z}_l^{j-1}\}_{l=1}^{N_{\mathrm{s}}})$. This permits to avoid storing all the training dataset to estimate the density function. The underlying assumption is that the batch size N_{s} is able to give a good representation of the likelihood through the KDE. Formally:

$$\mathrm{KL}\left(\mathcal{K}\left(\boldsymbol{z}_i|\{\boldsymbol{z}_j\}_{j=1}^{N_{\mathrm{s}}}\right) \,\|\, p(\boldsymbol{z}_i)\right) \backsimeq 0, \tag{12}$$

where $\mathrm{KL}(\cdot\|\cdot)$ is the Kullback-Leibler divergence. On the contrary, at inference time, we just check if the anomaly score $A_{\mathcal{L}}(\boldsymbol{z}_i) > \eta$. This implies that, during deployment, we can completely discard both the decoder $\mathcal{D}(\cdot)$ and the KDE $\mathcal{K}(\cdot)$, just relying on the faster prediction of the encoder $\mathcal{E}(\cdot)$ and likelihood network $\mathcal{L}(\cdot)$.

D. Loss Function

The objective of the loss function is to first induce the DAKDM to learn the latent representation of normal data and then to approximate $A_{\mathcal{K}}(\boldsymbol{z}_i)$ with $A_{\mathcal{L}}(\boldsymbol{z}_i)$. To this aim, we consider the training dataset $\mathcal{S}^{\text{train}} = \{\boldsymbol{B}_j\}_{j=1}^{N_{\mathrm{b}}}$, where N_{b} is the number of batches in the training dataset and $\boldsymbol{B}_j = \{\boldsymbol{X}_i^j\}_{i=1}^{N_{\mathrm{s}}}$ is the j-th mini-batch with N_{s} samples. We define the total loss related to mini-batch j as follows:

$$\begin{aligned}\mathrm{L}_{\mathrm{tot}}^j = &\frac{1}{N_{\mathrm{s}}} \sum_{i=1}^{N_{\mathrm{s}}} \mathrm{L}_{\mathrm{rec}}(\boldsymbol{X}_i^j, \hat{\boldsymbol{X}}_i^j) \\ &+ \frac{w_{\mathrm{KL}}}{N_{\mathrm{s}}} \sum_{i=1}^{N_{\mathrm{s}}} \mathrm{KL}_{\mathrm{point}}\left(\mathcal{L}(\boldsymbol{z}_i^j) \,\|\, \mathcal{K}\left(\boldsymbol{z}_i^j|\{\boldsymbol{z}_l^{j-1}\}_{l=1}^{N_{\mathrm{s}}}\right)\right) \\ &+ \frac{w_{\mathrm{lik}}}{N_{\mathrm{s}}} \sum_{i=1}^{N_{\mathrm{s}}} \left(-\log \mathcal{K}\left(\boldsymbol{z}_i^j|\{\boldsymbol{z}_l^{j-1}\}_{l=1}^{N_{\mathrm{s}}}\right)\right),\end{aligned} \tag{13}$$

where $\mathrm{L}_{\mathrm{rec}}(\boldsymbol{X}_i^j, \hat{\boldsymbol{X}}_i^j) = \|\boldsymbol{X}_i^j - \hat{\boldsymbol{X}}_i^j\|_2$ is the loss function that describes the reconstruction error given by the AE, w_{KL} and w_{lik} are the weighting parameters that control how much the single losses affect the objective function as a whole and $\mathrm{KL}_{\mathrm{point}}$ is the pointwise KL-divergence defined as:

$$\mathrm{KL}_{\mathrm{point}}(x\|y) = x \log\left(\frac{x}{y}\right). \tag{14}$$

With the second right-hand side of (13), we exploit the power of the likelihood network to learn the KDE output trained with the previous mini-batch. The choice of the loss function is motivated by the fact that, if assumption (12) holds, then we can write the contribution of $\boldsymbol{z}_i$ to the anomaly score with the following [96]:

$$\begin{aligned} -\log p(\boldsymbol{z}_i) \lesssim & -\log \mathcal{K}\left(\boldsymbol{z}_i | \{\boldsymbol{z}_j\}_{j=1}^{N_{\mathrm{s}}}\right) \\ & + \mathrm{KL}\left(\mathcal{L}(p(\boldsymbol{z}_i)|\boldsymbol{z}_i) \,\|\, p\left(\mathcal{K}\left(\boldsymbol{z}_i | \{\boldsymbol{z}_j\}_{j=1}^{N_{\mathrm{s}}}\right) | \boldsymbol{z}_i\right)\right), \end{aligned} \tag{15}$$

where $\mathcal{L}(p(\boldsymbol{z}_i)|\boldsymbol{z}_i)$ is the likelihood network that provides the probability of $\boldsymbol{z}_i$ given $\boldsymbol{z}_i$. For the proof of (15), please refer to Appendix A. We directly insert the first right-hand side of (15) in the loss function to induce the AE to decrease the anomaly score, thus increasing the likelihood. On the other hand, we do not have a KDE that provides the probability of its predictions, therefore we consider the $p(\mathcal{K}(\boldsymbol{z}_i|\{\boldsymbol{z}_j\}_{j=1}^{N_{\mathrm{s}}})|\boldsymbol{z}_i)$ as a single deterministic value that we approximate through the likelihood network.

V. Simulation Experiments

A. 5G NR Network Simulation

To evaluate the proposed DAKDM method for NLOS identification, we simulate realistic CSI data based on the 5G NR clustered delay line (CDL) channel model [97] which is characterized by a maximum bandwidth of 2 GHz over the whole frequency range of 0.5 GHz to 100 GHz. We simulate the wave propagation using a ray-tracing method [98], [99], [100] provided by Antenna Toolbox Matlab package where the propagation pathways from the UE to the BSs are computed based on the surface geometry from a map file. Ray-tracing uses the shooting and bouncing ray (SBR) method [101], accounting for up to 10 path reflections. The method does not take into account buildings' windows and possible foliage, which would require a high-definition 3D mapping of the environment or a complex simulation with artificially created maps. The channel model is then produced by coupling all the paths taking into account the small-scale fading due to the UE's movement, angle spread and cluster properties. This permits to achieve spatial consistency, meaning that two adjacent positions present similar channel characteristics due to comparable scattered environments.

B. Urban Mobility Scenario

For the experiments, we simulate a 3GPP urban micro (UMi) scenario in an area of 1000 m $\times$ 1000 m, near the Leonardo Campus of Politecnico di Milano, with specific parameters described in [85]. As shown in Fig. 5, the scenario comprises 19 urban sites, placed in an hexagonal layout with Inter-Site Distance (ISD) of 200 m, each equipped with 3 cells. The BS antennas are characterized by an UPA configuration with $M = N = 8$ elements and a downtilt of 15°. The transmission power is 44 dBm and each antenna element was defined using the specifications in [102], providing a front-to-back ratio of about 30 dB and a maximum gain of 8 dBi.

The vehicular UEs move in the area traveling along different trajectories generated with SUMO software which replicates actual traffic patterns on a particular route network. We generate up to 50 trajectories sampled every second, for a total simulation time of 170 s. Each UE is equipped with an omni-directional antenna and transmits the 5G standard compliant sounding reference signals (SRSs) to all the BSs in the area using a carrier frequency $f_{\mathrm{c}} = 30$ GHz and a transmission bandwidth $B = 400$ MHz. The BSs, which can be in either LOS or NLOS condition due to occlusions and reflections, demodulate the OFDM signal and estimate the channel using a least squares (LS) estimator. Subsequently, they obtain the channel fingerprint using the estimated channel response to compute the angle-delay channel structure (7) and then the associated power structure (8).

For the experiments, we do not consider the multi-user interference (MUI), but it is worth mentioning few considerations for possible real implementations of the method. In practice, the BSs can adopt various techniques to manage the inaccuracy of channel estimation due to factors such as the MUI. One common technique is to use channel estimation algorithms that are robust to MUI, such as linear minimum mean-square error (LMMSE) [103] which obtains sub-optimal performance (sub-optimal as it does not use the knowledge of the full CSI) with moderate computational complexity. Additionally, other techniques may rely on non-linear pre-coding schemes which have been found to provide near-optimal performance [104], [105]. In the standard of 5G-NR, codebook-based MIMO precoding techniques have been proposed and they are described in the 3GPP technical specifications (TS) 38-214 [106] and 38-211 [107]. With latest releases, i.e., Rel 16 and 17, MU-MIMO codebook (type II) has been improved with the reduction of the feedback overhead. By implementing these techniques, the MUI is highly reduced and the residual interference resembles to background noise. Moreover, in case the model has been trained in a channel in presence of non-null interference, we would have an even-broader LOS distribution characterization, which would be beneficial in case of single-user transmission.

C. CSI Dataset

In the offline phase, each BS is assured to gather LOS only realizations of the channel, composing a training dataset for the blockage detection. In the online phase, on the other hand, we create the test dataset adopting unobserved positions of the UEs and collecting a balanced number of samples between LOS and NLOS conditions. We saved more than $7.5 \cdot 10^4$ and $8.6 \cdot 10^4$ samples in the training and testing set, respectively. Before the training, all the samples are standardized (i.e., transformed such that the mean intensity is 0 with standard deviation of 1) and shuffled at each epoch.

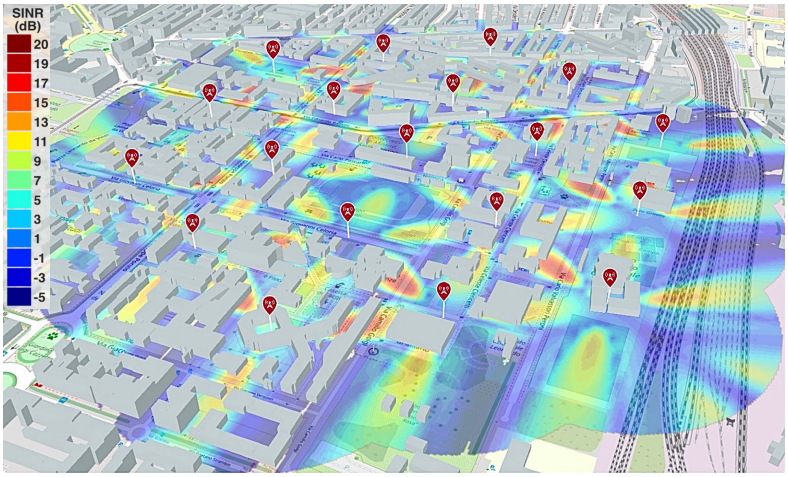


Fig. 5. UMi scenario composed of 19 sites in the area of Politecnico di Milano, Leonardo campus, Italy. The signal-to-interference-plus-noise ratio (SINR) is shown in downlink as a function of the UE position when BSs use trisectorial cells with broadside transmission.

MATLAB 2022a is used to create the channel fingerprints of the data points, while the DL model for training and testing is implemented using Pytorch [108] (v1.12 with Python 3.7.11). We run our simulations on a workstation with an Intel(R) Xeon(R) Silver 4210R CPU @ 2.40 GHz, 96 GB of RAM and a Quadro RTX 6000 24 GB GPU. The testing times, described in Sec. V-F1, only apply to the run time on Pytorch 1.12. Unless otherwise specified, we train the model for a number of epochs $E = 30$ with a batch size $N_s = 64$. w_{KL} and w_{lik} are both empirically set to 0.1. We adopted the Adam [109] with an initial learning rate $lr = 10^{-4}$, and momentum $\beta_1 = 0.9$, $\beta_2 = 0.999$.

D. DL Model Characteristics

For the AE part, we adopt the Segnet architecture [110] with one single channel encoder and decoder. The upsampling layers employ the encoder pool indices to create a sparse feature mapping which is ideal for reproducing the sparse ADCPM input. The AE is the most complex part of the model, however, at testing time, we use only the encoder part, thus halving the inference time if compared with VAE models or in general solutions that adopt the reconstruction error as a monitoring feature.

On the other hand, we develop the likelihood network using a simpler multi-layer perceptron (MLP) which is able to learn whatever non-linear function. The network can be found in Table I. To cope with the overfitting we adopt the dropout technique [111] after each activation function. Furthermore, we insert a single batch normalization layer [112] right before the ReLU function. This is done to avoid that the output of the network will converge to a unique value after a long training.

TABLE I
LIKELIHOOD NETWORK LAYER COMPOSITION

Layer Num.	Type	Output Size
0	Input	8×1
1	Linear + Tanh + Dropout	8×1
2	Linear + Tanh + Dropout	5×1
3	Linear	1×1
4	BatchNorm1d	1×1
5	ReLU	1×1

E. Baselines

To evaluate the performances of the proposed model, we compare it against a number of DL approaches proposed in the literature to solve anomaly detection problems:

- DAGMM [84]. Single-stage training model composed by an AE and a GMM used for learning the latent feature distribution. The membership weights, which represent the probability that a given data point belongs to each component, are usually computed with the expectation (E)-step of the expectation-maximization (EM) algorithm used for the GMM fitting. However, in this case, the membership weights are produced by an estimation DL network.
- AE-KDE [83]. Double-stage training model in which first the AE is trained and then a KDE is used to learn the distribution of latent features from all the training dataset. The bandwidth and the kernel are the same of DAKDM.
- VAE [76]. Auto-encoding variational Bayes applied to NLOS identification. Here, the sampling mechanism is mandatory since we need to sample new latent variables from the learned probability distribution, i.e., in this case a Gaussian distribution. The anomaly score $A(\boldsymbol{X}_i)$ is computed as $A(\boldsymbol{X}_i) = \frac{1}{N_m} \sum_{j=1}^{N_m} L_{rec}(\boldsymbol{X}_i, \hat{\boldsymbol{X}}_i^j)$, where N_m is the number of samples. As suggested by the authors, we draw 10 samples from the latent space representation to derive the anomaly score.

- GANomaly [113]. Deep-generative model composed by an AE, a second encoder and a discriminator. The model minimizes simultaneously the reconstruction error, the encoder loss given by the second encoder and the adversarial loss yielded by the discriminator.

For a fair comparison, we give the same input to each model and we adopt the unchanged architecture for the encoders and decoders with respect to DAKDM. Therefore, we adopt the same number of latent features for all architectures.

In addition to DL model baselines, we compare our method with classical ML and statistical algorithms. In particular, we implement:

- JLRT [63]. Joint-likelihood ratio test considering the statistics of the kurtosis, mean excess delay and root-mean-square delay spread. The PDFs of the statistics are approximated as log-normal distributions and they are considered independent of each other.
- RF [66]. Random forest model with 100 trees and, as input features, the Rician K-factor, root-mean-square delay spread, mean excess delay and dominant channel tap.
- CORR [75]. Pearson correlation coefficient computed using a reference set of LOS ADCPM sliced in the direction of arrival with higher received power. We gathered 100 LOS reference signals and we considered only the samples comprising 10 points before and 100 points after the first peak. The likelihood of a test input is obtained by averaging the correlation coefficient with the reference LOS signals.
- OC-SVM [58]. One-class support vector machine which computes the smallest hyper-sphere containing normal, i.e., LOS, samples. We use the score function as a likelihood measure. As regards the feature selection, we adopt both static channel characteristics as the maximum received power, kurtosis, skewness, rising time, root-mean-square delay spread, Rician K-factor, angular spread of arrival and both time-varying features [114] like the angular variant of arrival.

Note that, while CORR and OC-SVM are semi-supervised learning algorithms, JLRT and RF are supervised learning methods since they require statistics/samples of both classes. The models and algorithms are run independently by each BS, after the UE uplink transmission. The training, if required, is performed before the validation procedure at each BS using the locally collected input samples.

F. Results

1) Inference Timings: In this assessment, we want to measure the time required by each DL model to predict the output of a sample. This is of particular relevance in real-time applications where the inference time must be as low as possible. An example is the vehicular applications where the end-to-end latency must be contained within 100 ms or less [115]. In Fig. 6, we show the boxplot of the inference time for each sample over the whole testing dataset. We notice that the proposed DAKDM is able to predict the anomaly score in half of the time with respect to DAGMM as it does not require

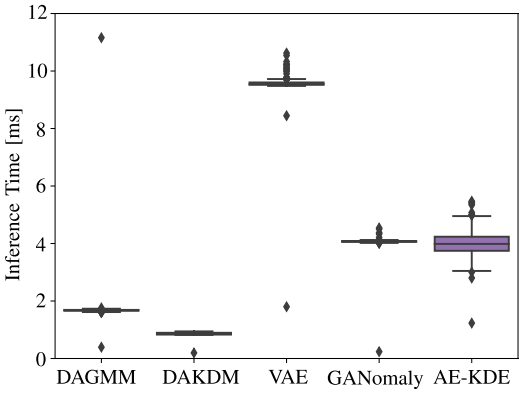
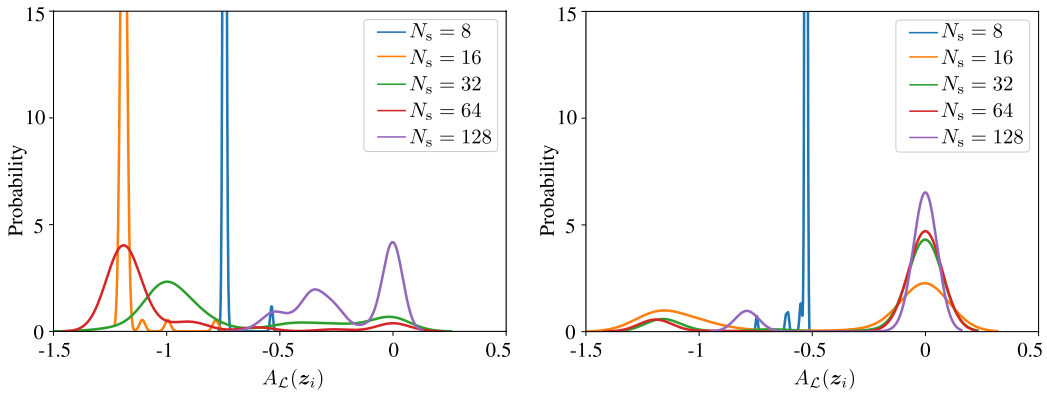


Fig. 6. Boxplot of the distribution of the inference time per sample [ms], varying the adopted DL models.

the decoder part prediction. Moreover, GANomaly and AE-KDE models require up to 4 ms for a single prediction. This is because GANomaly holds a more complex model, while AE-KDE has to pass through the whole training dataset for a single prediction. Finally, VAE takes about ten times more than DAKDM due to the sampling strategy.

2) Batch Size: This assessment has the goal of verifying how the batch size N_s affects the performances of the proposed DAKDM. Theoretically, N_s should be large enough to generate a good representation of the latent features' distribution. To verify this behaviour, in Fig. 7 we analyze the anomaly score $A_{\mathcal{L}}(\boldsymbol{z}_i)$ of normal (Fig. 7a) and anomalous (Fig. 7b) samples in the testing dataset after 30 epochs for $N_s = \{8, 16, 32, 64, 128\}$. To avoid singular issues due to possible zeros values given as input to the logarithm, we shift the likelihood distribution as $A_{\mathcal{L}}(\boldsymbol{z}_i) = -\log(\mathcal{L}(\boldsymbol{z}_i)+1)$. The first thing to notice is that the anomalous score of normal data is lower than the abnormal data and this is because the likelihood network outputs higher values for samples with normal distributions. Second, we observe that decreasing N_s for normal data, will produce lower mean and variances distributions, thus enhancing the NLOS identification capabilities. This is due to the fact that with a large batch size, the model struggles to learn the pointwise KL-divergence since in the loss function we have the contributions of many points. On the contrary, with lower batch sizes, the likelihood network learns exactly which value assign to each latent representation. Reducing N_s has the benefit of being suitable for simpler devices with low computational capabilities, in exchange for higher training times. As a trade-off between performances and training times, we choose $N_s = 16$.

3) Hyper-Parameters Tuning: This experiment aims of tuning the main hyper-parameters related to the density models, i.e., the bandwidth h of the KDE for DAKDM and AE-KDE and the number of GMM components, denoted with g, for DAGMM. In Fig. 8, we report the area under the curve (AUC) obtained in the testing set after 30 training epochs varying the bandwidth $h \in \{0.05, 0.1, 0.2, 0.4, 0.8, 1.6, 3, 6, 12\}$ (Fig. 8a)



(a) Anomaly score of normal samples

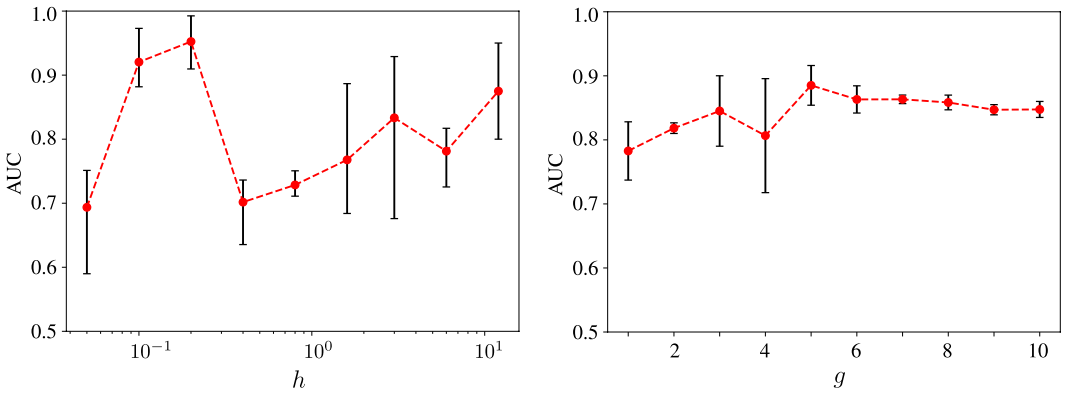
(b) Anomaly score of anomalous samples

Fig. 7. Comparison of the anomaly score $A_{\mathcal{L}}(\boldsymbol{z}_i)$ after 30 epochs between normal and anomalous data, for varying batch size N_s.

for DAKDM and the number of GMM components $g \in \{1, 2, 3, 4, 5, 6, 7, 8, 9, 10\}$ (Fig. 8b) for DAGMM. Starting from Fig. 8a, we notice that the higher AUC is reached by $h = 0.2$ and that for not optimal values, the AUC can differ significantly. This is somehow due to the range values of $\boldsymbol{z}_i$ and to the number of points that we have. Since, in practice, we have few anomalous samples, for tuning the bandwidth we can simply rely on maximizing the likelihood of normal samples varying the bandwidth. Comparing the results with Fig. 8b, we see that DAGMM is not able to achieve high peaks of performances in average, i.e., above 90% of AUC. This means that the latent distribution cannot be well approximated with less than 10 Gaussian distributions. Clearly, increasing g will improve the performance but at the cost of a higher complexity of the DAGMM estimation network.

4) Performance Comparison: In this last assessment, we compare the performances of the proposed DAKDM with the models described in Sec. V-E. In Table II, we report the average AUC after 10 runs and the F-score, Accuracy, Precision and Recall using a threshold on the anomaly score related to 20% of FPR. We notice that the performances of the AE-KDE (highlighted in green) are superior with respect to all the others. The reason behind this is that the AE-KDE represents the perfect unfeasible upper-bound, i.e., it requires storing the whole training dataset for inference and thus it can perfectly reconstruct the latent features distribution. On the other hand, the lower-bound is represented by the statistical JLRT method (highlighted in red), which obtains an AUC of 63%. This method assumes the independence of few hand-crafted features, which may not hold in any situation. The second non ML-based method CORR reaches an AUC of 64%, meaning that the LOS reference signals are not a good representation of the LOS distribution. The OC-SVM, a classical ML method, achieves a slightly higher AUC due to its capabilities of projecting the original features in a higher hyper-space (kernel trick). However, its main limitations lay in the features-choice which, for sparse and high-dimensional spaces, constitutes a non-trivial task. Moreover, we can notice that the precision (94%) is much higher with respect to the recall (60%). This means that OC-SVM tends to classify all test samples as LOS, learning a rough, i.e., too general, LOS distribution. Finally, among classical ML methods, the RF achieves the highest performances by reaching an AUC of 79%. However, we remark that this method requires the knowledge of NLOS samples, thus it holds an advantage with respect to semi-supervised learning methods.

Focusing now on the DL models, numerical results show that they highly outperform the classical ML and statistical algorithms. Indeed, while deterministic feature extraction might be more suitable for low-dimensional or simple channels, using the raw ADCPM as input to the CNN structure allows the DL models to utilize the full potential of automatic feature extraction, which contributes to the superior performance of the DL methods in comparison to classical ML and statistical algorithms. However, this does not exclude the possibility of incorporating deterministic features in future work to further improve the performance of the proposed model. Among the DL methods, DAKDM (highlighted in bold) and VAE hold the highest AUCs if compared with DAGMM and GANomaly. In particular, DAKDM and VAE outperform DAGMM and GANomaly of 7% and 16%, respectively. The reasons behind this are that GANomaly is a very complex network that requires a non-negligible effort in hyper-parameter tuning and optimization, with additional issues in training stability [79]. On the other hand, DAGMM is not able to accurately learn the LOS latent feature representation due to its limited number of Gaussian components. Both DAKDM and VAE achieves the highest performances, i.e., 95% and 96% of AUC, but with two different methods. While VAE imposes a simple latent distribution, DAKDM automatically learns the low-dimensional LOS distribution thanks to the KDE in training phase. However, the main advantage of DAKDM is that it does not require sampling procedures



(a) Tuning of bandwidth for DAKDM and AE-KDE.

(b) Tuning of number of GMM components for DAGMM.

Fig. 8. Comparison of the AUC reached after 30 epochs using (a) DAKDM and (b) DAGMM, for varying bandwidth h and number of GMM components g, respectively. The mean value (red dot) is plotted together with the associated uncertainty (error bar) computed using the maximum and minimum values of AUC as boundaries.

TABLE II

COMPARISON ON MEAN PERFORMANCE INDICATORS AFTER 10 DIFFERENT RUNS BETWEEN THE PROPOSED DAKDM AND THE BASELINES

	AUC	F-score	Accuracy	Precision	Recall
AE-KDE	0.9981	0.9376	0.9196	0.8845	0.9973
VAE	0.9620	0.8654	0.8491	0.9408	0.8013
DAKDM	**0.9535**	**0.8625**	**0.8456**	**0.9406**	**0.7972**
DAGMM	0.8919	0.8131	0.7767	0.8305	0.7964
GANomaly	0.8214	0.8337	0.8068	0.8697	0.8006
RF	0.7979	0.7753	0.7506	0.8495	0.7131
OC-SVM	0.7735	0.7419	0.7435	0.9477	0.6096
CORR	0.6449	0.7153	0.6152	0.6474	0.7992
JLRT	0.6395	0.5636	0.5962	0.7658	0.4459

and necessitates only 10% of the inference time needed by VAE (see Sec. V-F1). This makes it suitable for low-latency and mission-critical applications such as V2X networks for automated driving.

VI. CONCLUSION

This paper addressed the problem of high-dimensional channel distribution characterization for next generation cellular networks. In order to demonstrate the method, we tackle the problem of NLOS identification in a mm-wave MIMO system with sparse space-time channel responses. We model the problem within the semi-supervised anomaly detection framework where LOS samples correspond to normal data with peculiar characteristics and distributions. We propose a deep autoencoding kernel density model (DAKDM) where the manifold distribution of normal data is elaborated with an AE that takes as input the sparse ADCPM which univocally map the position-dependent features of the channel. The AE is jointly learned together with a likelihood network which is trained to learn the output of a KDE that directly estimates the distribution of the latent features. The DAKDM has the main advantage of learning whatever latent distribution without storing the whole training dataset.

We validated the model in a 5G standard compliant UMi scenario simulated with Matlab ray-tracing package, permitting spatial channel consistency between adjacent positions. The UEs are vehicles which move in the area according to dynamics simulated by the SUMO software. Compared with DL state-of-the-art models, results showed that the proposed DAKDM is much more efficient, both in terms of inference time and computational requirements. In particular, DAKDM holds a prediction time per sample which is up to one fourth and one tenth of GANomaly and VAE, respectively. This makes it appropriate for edge devices with strong latency requirements for mission-critical applications. From a performance point-of-view, DAKDM is able to achieve similar performances of the top-performer VAE, outperforming GMM-based method such as DAGMM of about 7%.

In the next years, ML and in particular DL methods are expected to play a crucial role in next generation cellular networks. Communication systems, but also localization techniques, are required to increase performance capabilities and types of services to accomplish increasingly high standards. Thus, DL-based methods as the proposed DAKDM become essential to push further the performances. A natural extension of our work would be to integrate NLOS mitigation into the system in order to compensate the induced error given by lack of visibility or directly integrate DL techniques into positioning algorithms suited for high complexity environments. A further direction of research could be the extension to a cooperative inference framework where BSs exchange mutual-soft information for accuracy enhancement. Moreover, more realistic environments with simulated foliage and dynamic obstructions should be explored. Challenges are represented by NLOS situations, changes in the environment and lack of possible representative samples for each feasible location.

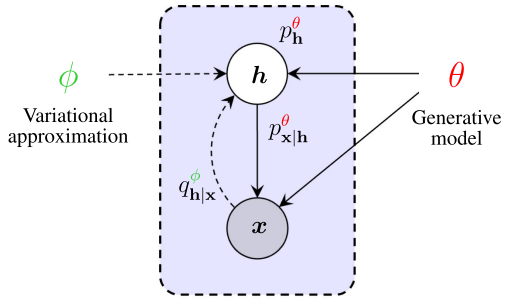


Fig. 9. Variational inference problem as described in [116]. The generative model is indicated by solid lines while variational approximation is reported with dashed lines.

Appendix A
Proof Of (15)

In this Appendix we provide a proof of the anomaly score contribution given by $\boldsymbol{z}_i$. From the variational inference approach [96], we can note that the likelihood network performs the same objective of latent variable inference. To see this parallelism, we recall the variational inference context where we are given an observation $\boldsymbol{h}$ (i.e., latent variable) from the prior distribution $p^{\theta}_{\mathbf{h}}$ with parameters θ. Subsequently, a datapoint $\boldsymbol{x}$ is generated from $p^{\theta}_{\mathbf{x}|\mathbf{h}}$, which is considered intractable. The objective is to estimate the exact posterior $p^{\theta}_{\mathbf{h}|\mathbf{x}}$, also intractable, with a simpler variational posterior $q^{\phi}_{\mathbf{h}|\mathbf{x}}$ with parameters ϕ. For a graphical representation of the problem, please refer to Fig. 9 [116].

We can view the datapoint $\boldsymbol{x}$ as a compact representation of the channel $\boldsymbol{z}_i$ and the latent variable $\boldsymbol{h}$ as the distribution $p(\boldsymbol{z}_i)$. Let us denote the probability as $\mathsf{s}_i = p_{\mathbf{z}}(\boldsymbol{z}_i)$, which can be approximated with the KDE $\hat{\mathsf{s}}_i = \mathcal{K}\big(\boldsymbol{z}_i|\{\boldsymbol{z}_j\}_{j=1}^{N_{\mathrm{s}}}\big)$, the likelihood network $\mathcal{L}^{\phi}(\cdot)$ (with parameters ϕ) will be acting as the variational distribution $q^{\phi}(\cdot)$. Following this parallelism, the contribution of $\boldsymbol{z}_i$ to the anomaly score, i.e., negative log-likelihood, can be written as follows:

$$
\begin{aligned}
-\log p_{\mathbf{z}}(\boldsymbol{z}_i) &= -\log \int_s p_{\mathbf{z},\mathsf{s}_i}(\boldsymbol{z}_i, s)\, ds \qquad (16)\\
&= -\log \int_s \mathcal{L}^{\phi}(s|\boldsymbol{z}_i)\, \frac{p_{\mathbf{z},\mathsf{s}_i}(\boldsymbol{z}_i, s)}{\mathcal{L}^{\phi}(s|\boldsymbol{z}_i)}\, ds\\
&\leq -\int_s \mathcal{L}^{\phi}(s|\boldsymbol{z}_i) \log \frac{p_{\mathbf{z},\mathsf{s}_i}(\boldsymbol{z}_i, s)}{\mathcal{L}^{\phi}(s|\boldsymbol{z}_i)}\, ds \qquad (17)
\end{aligned}
$$

where (17) is the consequence of Jensen's inequality. It follows that

$$
\begin{aligned}
-\log p_{\mathbf{z}}(\boldsymbol{z}_i) &\leq -\mathbb{E}_{\mathcal{L}^{\phi}}\Big\{\log p_{\mathbf{z},\mathsf{s}_i}(\boldsymbol{z}_i, s) - \log \mathcal{L}^{\phi}(s|\boldsymbol{z}_i)\Big\}\\
&= -\log p_{\mathbf{z}}(\boldsymbol{z}_i) + \mathrm{KL}\big(\mathcal{L}^{\phi}(s|\boldsymbol{z}_i) \,\big\|\, p_{\mathsf{s}_i|\mathbf{z}}(s|\boldsymbol{z}_i)\big)\\
&\backsimeq -\log \mathcal{K}\big(\boldsymbol{z}_i|\{\boldsymbol{z}_j\}_{j=1}^{N_{\mathrm{s}}}\big)\\
&\quad + \mathrm{KL}\big(\mathcal{L}^{\phi}(s|\boldsymbol{z}_i) \,\big\|\, p_{\hat{\mathsf{s}}_i|\mathbf{z}}(\hat{s}|\boldsymbol{z}_i)\big), \qquad (18)
\end{aligned}
$$

where the approximation in (18) is due to (12), concluding the proof.

References

[1] M. Win et al., "Network localization and navigation via cooperation," *IEEE Commun. Mag.*, vol. 49, no. 5, pp. 56–62, May 2011. [Online]. Available: http://ieeexplore.ieee.org/document/5762798/

[2] A. Conti et al., "Location awareness in beyond 5G networks," *IEEE Commun. Mag.*, vol. 59, no. 11, pp. 22–27, Nov. 2021. [Online]. Available: https://ieeexplore.ieee.org/document/9665420/

[3] S. Fischer, *5G and Beyond: Fundamentals and Standards*, X. Lin and N. Lee, Eds. Cham, Switzerland: Springer, 2021, doi: 10.1007/978-3-030-58197-8.

[4] F. Mogyorósi et al., "Positioning in 5G and 6G networks—A survey," *Sensors*, vol. 22, no. 13, p. 4757, Jun. 2022. [Online]. Available: https://www.mdpi.com/1424-8220/22/13/4757

[5] O. Kanhere and T. S. Rappaport, "Position location for futuristic cellular communications: 5G and beyond," *IEEE Commun. Mag.*, vol. 59, no. 1, pp. 70–75, Jan. 2021. [Online]. Available: http://ieeexplore.ieee.org/document/9356512/

[6] A. H. Sayed, A. Tarighat, and N. Khajehnouri, "Network-based wireless location: Challenges faced in developing techniques for accurate wireless location information," *IEEE Signal Process. Mag.*, vol. 22, no. 4, pp. 24–40, Jul. 2005. [Online]. Available: http://ieeexplore.ieee.org/document/1458275/

[7] N. Patwari, J. N. Ash, S. Kyperountas, A. O. Hero, R. L. Moses, and N. S. Correal, "Locating the nodes: Cooperative localization in wireless sensor networks," *IEEE Signal Process. Mag.*, vol. 22, no. 4, pp. 54–69, Jul. 2005. [Online]. Available: http://ieeexplore.ieee.org/document/1458287/

[8] M. Z. Win, Z. Wang, Z. Liu, Y. Shen, and A. Conti, "Location awareness via intelligent surfaces: A path toward holographic NLN," *IEEE Veh. Technol. Mag.*, vol. 17, no. 2, pp. 37–45, Jun. 2022. [Online]. Available: https://ieeexplore.ieee.org/document/9781656/

[9] Z. Wang, Z. Liu, Y. Shen, A. Conti, and M. Z. Win, "Location awareness in beyond 5G networks via reconfigurable intelligent surfaces," *IEEE J. Sel. Areas Commun.*, vol. 40, no. 7, pp. 2011–2025, Jul. 2022. [Online]. Available: https://ieeexplore.ieee.org/document/9729782/

[10] A. Shahmansoori, G. E. Garcia, G. Destino, G. Seco-Granados, and H. Wymeersch, "Position and orientation estimation through millimeter-wave MIMO in 5G systems," *IEEE Trans. Wireless Commun.*, vol. 17, no. 3, pp. 1822–1835, Mar. 2018. [Online]. Available: https://ieeexplore.ieee.org/document/8240645/

[11] R. Karlsson and F. Gustafsson, "The future of automotive localization algorithms: Available, reliable, and scalable localization: Anywhere and anytime," *IEEE Signal Process. Mag.*, vol. 34, no. 2, pp. 60–69, Mar. 2017. [Online]. Available: http://ieeexplore.ieee.org/document/7872535/

[12] X. Wang, T. Li, S. Sun, and J. Corchado, "A survey of recent advances in particle filters and remaining challenges for multitarget tracking," *Sensors*, vol. 17, no. 12, p. 2707, Nov. 2017. [Online]. Available: http://www.mdpi.com/1424-8220/17/12/2707

[13] M. Chiani, A. Giorgetti, and E. Paolini, "Sensor radar for object tracking," *Proc. IEEE*, vol. 106, no. 6, pp. 1022–1041, Jun. 2018. [Online]. Available: https://ieeexplore.ieee.org/document/8365918/

[14] N. Wang and C. K. Ahn, "Coordinated trajectory-tracking control of a marine aerial-surface heterogeneous system," *IEEE/ASME Trans. Mechatronics*, vol. 26, no. 6, pp. 3198–3210, Dec. 2021. [Online]. Available: https://ieeexplore.ieee.org/document/9339888/

[15] M. Z. Win, F. Meyer, Z. Liu, W. Dai, S. Bartoletti, and A. Conti, "Efficient multisensor localization for the Internet of Things: Exploring a new class of scalable localization algorithms," *IEEE Signal Process. Mag.*, vol. 35, no. 5, pp. 153–167, Sep. 2018. [Online]. Available: https://ieeexplore.ieee.org/document/8454389/

[16] A. Zanella, N. Bui, A. Castellani, L. Vangelista, and M. Zorzi, "Internet of Things for smart cities," *IEEE Internet Things J.*, vol. 1, no. 1, pp. 22–32, Feb. 2014. [Online]. Available: https://ieeexplore.ieee.org/document/6740844/

[17] A. A. Saucan and M. Z. Win, "Information-seeking sensor selection for Ocean-of-Things," *IEEE Internet Things J.*, vol. 7, no. 10, pp. 10072–10088, Oct. 2020. [Online]. Available: https://ieeexplore.ieee.org/document/9086780/

Chapter 7. Efficient Distribution Sampling for NLoS Identification

[18] B. Guo et al., "Mobile crowd sensing and computing: The review of an emerging human-powered sensing paradigm," *ACM Comput. Surv.*, vol. 48, no. 1, pp. 1–31, Sep. 2015. [Online]. Available: https://dl.acm.org/doi/10.1145/2794400

[19] D. E. Boubiche, M. Imran, A. Maqsood, and M. Shoaib, "Mobile crowd sensing—Taxonomy, applications, challenges, and solutions," *Comput. Hum. Behav.*, vol. 101, pp. 352–370, Dec. 2019. [Online]. Available: https://linkinghub.elsevier.com/retrieve/pii/S0747563218305181

[20] G. Pasolini, P. Toppan, F. Zabini, C. D. Castro, and O. Andrisano, "Design, deployment and evolution of heterogeneous smart public lighting systems," *Appl. Sci.*, vol. 9, no. 16, p. 3281, Aug. 2019. [Online]. Available: https://www.mdpi.com/2076-3417/9/16/3281

[21] A. Eskandarian, C. Wu, and C. Sun, "Research advances and challenges of autonomous and connected ground vehicles," *IEEE Trans. Intell. Transp. Syst.*, vol. 22, no. 2, pp. 683–711, Feb. 2021. [Online]. Available: https://ieeexplore.ieee.org/document/8936542/

[22] B. C. Tedeschini, M. Brambilla, L. Barbieri, and M. Nicoli, "Addressing data association by message passing over graph neural networks," in *Proc. 25th Int. Conf. Inf. Fusion (FUSION)*, Linköping, Sweden, Jul. 2022, pp. 1–7. [Online]. Available: https://ieeexplore.ieee.org/document/9841233/

[23] H. Wymeersch, G. Seco-Granados, G. Destino, D. Dardari, and F. Tufvesson, "5G mmWave positioning for vehicular networks," *IEEE Wireless Commun.*, vol. 24, no. 6, pp. 80–86, Dec. 2017. [Online]. Available: http://ieeexplore.ieee.org/document/8246850/

[24] S. Ko, H. Chae, K. Han, S. Lee, D. Seo, and K. Huang, "V2X-based vehicular positioning: Opportunities, challenges, and future directions," *IEEE Wireless Commun.*, vol. 28, no. 2, pp. 144–151, Apr. 2021. [Online]. Available: http://ieeexplore.ieee.org/document/9382021/

[25] *Study on Enhancement of 3GPP Support for 5G V2X Services*, document TR 22.886, 3GPP, Dec. 2018, version 16.2.0.

[26] M. Z. Win, Y. Shen, and W. Dai, "A theoretical foundation of network localization and navigation," *Proc. IEEE*, vol. 106, no. 7, pp. 1136–1165, Jul. 2018. [Online]. Available: https://ieeexplore.ieee.org/document/8421288/

[27] Z. Liu, A. Conti, S. K. Mitter, and M. Z. Win, "Communication-efficient distributed learning over networks—Part I: Sufficient conditions for accuracy," *IEEE J. Sel. Areas Commun.*, vol. 41, no. 4, pp. 1081–1101, Apr. 2023. [Online]. Available: https://ieeexplore.ieee.org/document/10043782

[28] Z. Liu, A. Conti, S. K. Mitter, and M. Z. Win, "Communication-efficient distributed learning over networks—Part II: Necessary conditions for accuracy," *IEEE J. Sel. Areas Commun.*, vol. 41, no. 4, pp. 1102–1119, Apr. 2023. [Online]. Available: https://ieeexplore.ieee.org/document/10043777

[29] A. A. D'Amico, A. D. J. Torres, L. Sanguinetti, and M. Win, "Cramér–Rao bounds for holographic positioning," *IEEE Trans. Signal Process.*, vol. 70, pp. 5518–5532, 2022. [Online]. Available: https://ieeexplore.ieee.org/abstract/document/9950340

[30] S. Mazuelas, Y. Shen, and M. Z. Win, "Spatiotemporal information coupling in network navigation," *IEEE Trans. Inf. Theory*, vol. 64, no. 12, pp. 7759–7779, Dec. 2018. [Online]. Available: https://ieeexplore.ieee.org/document/8418783/

[31] Y. Shen, S. Mazuelas, and M. Z. Win, "Network navigation: Theory and interpretation," *IEEE J. Sel. Areas Commun.*, vol. 30, no. 9, pp. 1823–1834, Oct. 2012. [Online]. Available: http://ieeexplore.ieee.org/document/6311242/

[32] A. T. Ihler, J. W. Fisher, R. L. Moses, and A. S. Willsky, "Nonparametric belief propagation for self-localization of sensor networks," *IEEE J. Sel. Areas Commun.*, vol. 23, no. 4, pp. 809–819, Apr. 2005. [Online]. Available: http://ieeexplore.ieee.org/document/1413473/

[33] A. Conti, S. Mazuelas, S. Bartoletti, W. C. Lindsey, and M. Z. Win, "Soft information for Localization-of-Things," *Proc. IEEE*, vol. 107, no. 11, pp. 2240–2264, Nov. 2019. [Online]. Available: https://ieeexplore.ieee.org/document/8827486/

[34] F. Meyer et al., "Message passing algorithms for scalable multitarget tracking," *Proc. IEEE*, vol. 106, no. 2, pp. 221–259, Feb. 2018. [Online]. Available: http://ieeexplore.ieee.org/document/8290605/

[35] S. Bartoletti, A. Giorgetti, M. Z. Win, and A. Conti, "Blind selection of representative observations for sensor radar networks," *IEEE Trans. Veh. Technol.*, vol. 64, no. 4, pp. 1388–1400, Apr. 2015. [Online]. Available: http://ieeexplore.ieee.org/document/7024174/

[36] U. A. Khan, S. Kar, and J. M. F. Moura, "DILAND: An algorithm for distributed sensor localization with noisy distance measurements," *IEEE Trans. Signal Process.*, vol. 58, no. 3, pp. 1940–1947, Mar. 2010. [Online]. Available: http://ieeexplore.ieee.org/document/5352246/

[37] S. Bartoletti, W. Dai, A. Conti, and M. Z. Win, "A mathematical model for wideband ranging," *IEEE J. Sel. Topics Signal Process.*, vol. 9, no. 2, pp. 216–228, Mar. 2015. [Online]. Available: http://ieeexplore.ieee.org/document/6955781/

[38] H. Wymeersch, J. Lien, and M. Z. Win, "Cooperative localization in wireless networks," *Proc. IEEE*, vol. 97, no. 2, pp. 427–450, Mar. 2009. [Online]. Available: http://ieeexplore.ieee.org/document/4802193/

[39] M. Z. Win, W. Dai, Y. Shen, G. Chrisikos, and H. V. Poor, "Network operation strategies for efficient localization and navigation," *Proc. IEEE*, vol. 106, no. 7, pp. 1224–1254, Jul. 2018. [Online]. Available: https://ieeexplore.ieee.org/document/8421291/

[40] U. A. Khan, S. Kar, and J. M. F. Moura, "Distributed sensor localization in random environments using minimal number of anchor nodes," *IEEE Trans. Signal Process.*, vol. 57, no. 5, pp. 2000–2016, May 2009. [Online]. Available: http://ieeexplore.ieee.org/document/4776469/

[41] T. Wang, Y. Shen, A. Conti, and M. Z. Win, "Network navigation with scheduling: Error evolution," *IEEE Trans. Inf. Theory*, vol. 63, no. 11, pp. 7509–7534, Nov. 2017. [Online]. Available: http://ieeexplore.ieee.org/document/7953679/

[42] J. Chen, W. Dai, Y. Shen, V. K. N. Lau, and M. Z. Win, "Resource management games for distributed network localization," *IEEE J. Sel. Areas Commun.*, vol. 35, no. 2, pp. 317–329, Feb. 2017. [Online]. Available: http://ieeexplore.ieee.org/document/7835141/

[43] A. Conti, M. Guerra, D. Dardari, N. Decarli, and M. Z. Win, "Network experimentation for cooperative localization," *IEEE J. Sel. Areas Commun.*, vol. 30, no. 2, pp. 467–475, Feb. 2012. [Online]. Available: http://ieeexplore.ieee.org/document/6136832/

[44] A. Conti, D. Dardari, M. Guerra, L. Mucchi, and M. Z. Win, "Experimental characterization of diversity navigation," *IEEE Syst. J.*, vol. 8, no. 1, pp. 115–124, Mar. 2014. [Online]. Available: http://ieeexplore.ieee.org/document/6581875/

[45] Z. Liu, W. Dai, and M. Z. Win, "Mercury: An infrastructure-free system for network localization and navigation," *IEEE Trans. Mobile Comput.*, vol. 17, no. 5, pp. 1119–1133, May 2018. [Online]. Available: https://ieeexplore.ieee.org/document/7983453/

[46] B. Teague, Z. Liu, F. Meyer, A. Conti, and M. Z. Win, "Network localization and navigation with scalable inference and efficient operation," *IEEE Trans. Mobile Comput.*, vol. 21, no. 6, pp. 2072–2087, Jun. 2022. [Online]. Available: https://ieeexplore.ieee.org/document/9247273/

[47] L. Barbieri, M. Brambilla, A. Trabattoni, S. Mervic, and M. Nicoli, "UWB localization in a smart factory: Augmentation methods and experimental assessment," *IEEE Trans. Instrum. Meas.*, vol. 70, pp. 1–18, Apr. 2021. [Online]. Available: https://ieeexplore.ieee.org/document/9409138/

[48] S. Marano, W. Gifford, H. Wymeersch, and M. Win, "NLOS identification and mitigation for localization based on UWB experimental data," *IEEE J. Sel. Areas Commun.*, vol. 28, no. 7, pp. 1026–1035, Sep. 2010. [Online]. Available: http://ieeexplore.ieee.org/document/5555901/

[49] H. Wymeersch, S. Marano, W. M. Gifford, and M. Z. Win, "A machine learning approach to ranging error mitigation for UWB localization," *IEEE Trans. Commun.*, vol. 60, no. 6, pp. 1719–1728, Jun. 2012. [Online]. Available: http://ieeexplore.ieee.org/document/6192275/

[50] A. Bourdoux et al., "6G white paper on localization and sensing," Jun. 2020, *arXiv:2006.01779*.

[51] T. Wild, V. Braun, and H. Viswanathan, "Joint design of communication and sensing for beyond 5G and 6G systems," *IEEE Access*, vol. 9, pp. 30845–30857, 2021. [Online]. Available: http://ieeexplore.ieee.org/document/9354629/

[52] *Moderator's Summary for Discussion [RAN93e-R18Prep-10] Expanded and Improved Positioning*, document TSG-RAN Meeting (RP) 0.0.4, 3GPP, Version 0.0.4, Sep. 2021. [Online]. Available: https://www.3gpp.org/ftp/tsgran/TSGRAN/TSGR93e/Docs/

[53] A. Nessa, B. Adhikari, F. Hussain, and X. N. Fernando, "A survey of machine learning for indoor positioning," *IEEE Access*, vol. 8, pp. 214945–214965, 2020. [Online]. Available: http://ieeexplore.ieee.org/document/9264122/

[54] L. Cazzella et al., "Deep learning of transferable MIMO channel modes for 6G V2X communications," *IEEE Trans. Antennas Propag.*, vol. 70, no. 6, pp. 4127–4139, Jun. 2022. [Online]. Available: https://ieeexplore.ieee.org/document/9765739/

[55] K. Wang et al., "Task offloading with multi-tier computing resources in next generation wireless networks," *IEEE J. Sel. Areas Commun.*, vol. 41, no. 2, pp. 306–319, Feb. 2023. [Online]. Available: https://ieeexplore.ieee.org/document/9978919/

Chapter 7. Efficient Distribution Sampling for NLoS Identification

[56] M. Brambilla, D. Pardo, and M. Nicoli, "Location-assisted subspace-based beam alignment in LOS/NLOS mm-wave V2X communications," in *Proc. IEEE Int. Conf. Commun. (ICC)*, Jun. 2020, pp. 1–6. [Online]. Available: https://ieeexplore.ieee.org/document/9148587/

[57] M. Brambilla, L. Combi, A. Matera, D. Tagliaferri, M. Nicoli, and U. Spagnolini, "Sensor-aided V2X beam tracking for connected automated driving: Distributed architecture and processing algorithms," *Sensors*, vol. 20, no. 12, p. 3573, Jun. 2020. [Online]. Available: https://www.mdpi.com/1424-8220/20/12/3573

[58] Z. Miao, L. Zhao, W. Yuan, and F. Jin, "Application of one-class classification in NLOS identification of UWB positioning," in *Proc. Int. Conf. Inf. Syst. Artif. Intell. (ISAI)*, Jun. 2016, pp. 318–322. [Online]. Available: https://ieeexplore.ieee.org/document/7816727/

[59] J. Borras, P. Hatrack, and N. Mandayam, "Decision theoretic framework for NLOS identification," in *Proc. 48th IEEE Veh. Technol. Conf. Pathway Global Wireless Revolution (VTC)*, vol. 2, May 1998, pp. 1583–1587. [Online]. Available: http://ieeexplore.ieee.org/document/686556/

[60] Y.-H. Jo, J.-Y. Lee, D.-H. Ha, and S.-H. Kang, "Accuracy enhancement for UWB indoor positioning using ray tracing," in *Proc. IEEE/ION PLANS*, Apr. 2006, pp. 565–568. [Online]. Available: http://ieeexplore.ieee.org/document/1650645/

[61] S. Seidel and T. Rappaport, "A ray tracing technique to predict path loss and delay spread inside buildings," in *Proc. Commun. Global Users (GLOBECOM)*, vol. 2, Dec. 1992, pp. 649–653. [Online]. Available: http://ieeexplore.ieee.org/document/276436/

[62] I. Guvenc, C.-C. Chong, and F. Watanabe, "NLOS identification and mitigation for UWB localization systems," in *Proc. IEEE Wireless Commun. Netw. Conf.*, Mar. 2007, pp. 1571–1576. [Online]. Available: http://ieeexplore.ieee.org/document/4224541/

[63] İ. Güvenç, C.-C. Chong, F. Watanabe, and H. Inamura, "NLOS identification and weighted least-squares localization for UWB systems using multipath channel statistics," *EURASIP J. Adv. Signal Process.*, vol. 2008, no. 1, Aug. 2007, Art. no. 271984.

[64] C. Huang et al., "Artificial intelligence enabled radio propagation for communications—Part II: Scenario identification and channel modeling," *IEEE Trans. Antennas Propag.*, vol. 70, no. 6, pp. 3955–3969, Jun. 2022. [Online]. Available: https://ieeexplore.ieee.org/document/9713743/

[65] T. Van Nguyen, Y. Jeong, H. Shin, and M. Z. Win, "Machine learning for wideband localization," *IEEE J. Sel. Areas Commun.*, vol. 33, no. 7, pp. 1357–1380, Jul. 2015. [Online]. Available: http://ieeexplore.ieee.org/document/7102989/

[66] E. Kurniawan, L. Zhiwei, and S. Sun, "Machine learning-based channel classification and its application to IEEE 802.11ad communications," in *Proc. IEEE Global Commun. Conf. (GLOBECOM)*, Dec. 2017, pp. 1–6. [Online]. Available: http://ieeexplore.ieee.org/document/8254052/

[67] X. Yang, "NLOS mitigation for UWB localization based on sparse pseudo-input Gaussian process," *IEEE Sensors J.*, vol. 18, no. 10, pp. 4311–4316, May 2018. [Online]. Available: https://ieeexplore.ieee.org/document/8322223/

[68] Q. Zheng et al., "Channel non-line-of-sight identification based on convolutional neural networks," *IEEE Wireless Commun. Lett.*, vol. 9, no. 9, pp. 1500–1504, Sep. 2020.

[69] C. Jiang, J. Shen, S. Chen, Y. Chen, D. Liu, and Y. Bo, "UWB NLOS/LOS classification using deep learning method," *IEEE Commun. Lett.*, vol. 24, no. 10, pp. 2226–2230, Oct. 2020. [Online]. Available: https://ieeexplore.ieee.org/document/9108193/

[70] Z. Cui, Y. Gao, J. Hu, S. Tian, and J. Cheng, "LOS/NLOS identification for indoor UWB positioning based on Morlet wavelet transform and convolutional neural networks," *IEEE Commun. Lett.*, vol. 25, no. 3, pp. 879–882, Mar. 2021. [Online]. Available: https://ieeexplore.ieee.org/document/9264213/

[71] H. Chen, Y. Zhang, W. Li, X. Tao, and P. Zhang, "ConFi: Convolutional neural networks based indoor Wi-Fi localization using channel state information," *IEEE Access*, vol. 5, pp. 18066–18074, 2017. [Online]. Available: https://ieeexplore.ieee.org/document/8027020/

[72] Y. Jing, J. Hao, and P. Li, "Learning spatiotemporal features of CSI for indoor localization with dual-stream 3D convolutional neural networks," *IEEE Access*, vol. 7, pp. 147571–147585, 2019. [Online]. Available: https://ieeexplore.ieee.org/document/8864050/

[73] J. Fan and A. S. Awan, "Non-line-of-sight identification based on unsupervised machine learning in ultra wideband systems," *IEEE Access*, vol. 7, pp. 32464–32471, 2019. [Online]. Available: https://ieeexplore.ieee.org/document/8666972/

[74] J. E. van Engelen and H. H. Hoos, "A survey on semi-supervised learning," *Mach. Learn.*, vol. 109, no. 2, pp. 373–440, Feb. 2020.

[75] Z. Zeng, R. Bai, L. Wang, and S. Liu, "NLOS identification and mitigation based on CIR with particle filter," in *Proc. IEEE Wireless Commun. Netw. Conf. (WCNC)*, Apr. 2019, pp. 1–6. [Online]. Available: https://ieeexplore.ieee.org/document/8886002/

[76] M. Stahlke, S. Kram, F. Ott, T. Feigl, and C. Mutschler, "Estimating TOA reliability with variational autoencoders," *IEEE Sensors J.*, vol. 22, no. 6, pp. 5133–5140, Mar. 2022. [Online]. Available: https://ieeexplore.ieee.org/document/9503384/

[77] P. Oza and V. M. Patel, "One-class convolutional neural network," *IEEE Signal Process. Lett.*, vol. 26, no. 2, pp. 277–281, Feb. 2019. [Online]. Available: https://ieeexplore.ieee.org/document/8586962/

[78] L. Ruff et al., "A unifying review of deep and shallow anomaly detection," *Proc. IEEE*, vol. 109, no. 5, pp. 756–795, May 2021. [Online]. Available: https://ieeexplore.ieee.org/document/9347460/

[79] M. Arjovsky and L. Bottou, "Towards principled methods for training generative adversarial networks," Jan. 2017, *arXiv:1701.04862*.

[80] H. Xu et al., "Unsupervised anomaly detection via variational auto-encoder for seasonal KPIs in web applications," in *Proc. World Wide Web Conf. (WWW)*, 2018, pp. 187–196, doi: 10.1145/3178876.3185996.

[81] E. Nalisnick, A. Matsukawa, Y. W. Teh, D. Gorur, and B. Lakshminarayanan, "Do deep generative models know what they don't know?" Oct. 2018, *arXiv:1810.09136*.

[82] C. Zhou and R. C. Paffenroth, "Anomaly detection with robust deep autoencoders," in *Proc. 23rd ACM SIGKDD Int. Conf. Knowl. Discovery Data Mining*. New York, NY, USA: Association for Computing Machinery, Aug. 2017, pp. 665–674, doi: 10.1145/3097983.3098052.

[83] V. L. Cao, M. Nicolau, and J. McDermott, "A hybrid autoencoder and density estimation model for anomaly detection," in *Parallel Problem Solving From Nature—PPSN XIV*, J. Handl, E. Hart, P. R. Lewis, M. López-Ibáñez, G. Ochoa, and B. Paechter, Eds. Cham, Switzerland: Springer, Aug. 2016, pp. 717–726.

[84] B. Zong et al., "Deep autoencoding Gaussian mixture model for unsupervised anomaly detection," in *Proc. Int. Conf. Learn. Represent.*, Feb. 2018, pp. 1–19. [Online]. Available: https://openreview.net/forum?id=BJJLHbb0-

[85] *Study on NR Positioning Support*, document TR 38.855, Version 16.0.0, 3GPP, Sep. 2019. [Online]. Available: https://portal.3gpp.org/desktopmodules/Specifications/SpecificationDetails.aspx?specificationId=3501

[86] *MATLAB, R2022a*. Natick, MA, USA: MathWorks, 2022. [Online]. Available: https://it.mathworks.com/help/antenna/ref/rfprop.raytracing.html

[87] P. A. Lopez et al., "Microscopic traffic simulation using SUMO," in *Proc. 21st Int. Conf. Intell. Transp. Syst. (ITSC)*, Nov. 2018, pp. 2575–2582. [Online]. Available: https://elib.dlr.de/124092/

[88] D. Tse and P. Viswanath, *Fundamentals of Wireless Communication*. Cambridge, U.K.: Cambridge Univ. Press, 2005.

[89] H. L. Van Trees and E. Detection, *Modulation Theory, Optimum Array Processing (Part IV)*. New York, NY, USA: Wiley, 2002.

[90] C. Sun, X. Gao, S. Jin, M. Matthaiou, Z. Ding, and C. Xiao, "Beam division multiple access transmission for massive MIMO communications," *IEEE Trans. Commun.*, vol. 63, no. 6, pp. 2170–2184, Jun. 2015. [Online]. Available: https://ieeexplore.ieee.org/document/7093168/

[91] X. Sun, X. Gao, G. Y. Li, and W. Han, "Single-site localization based on a new type of fingerprint for massive MIMO-OFDM systems," *IEEE Trans. Veh. Technol.*, vol. 67, no. 7, pp. 6134–6145, Jul. 2018. [Online]. Available: https://ieeexplore.ieee.org/document/8307353/

[92] Y. Zhou, M. Herdin, A. M. Sayeed, and E. Bonek, "Experimental study of MIMO channel statistics and capacity via the virtual channel representation," Univ. Wisconsin-Madison, Madison, WI, USA, Tech. Rep., Jan. 2007, pp. 10–15, vol. 5.

[93] C. Wu et al., "Learning to localize: A 3D CNN approach to user positioning in massive MIMO-OFDM systems," *IEEE Trans. Wireless Commun.*, vol. 20, no. 7, pp. 4556–4570, Jul. 2021. [Online]. Available: https://ieeexplore.ieee.org/document/9149427/

[94] M. D. Zeiler and R. Fergus, "Visualizing and understanding convolutional networks," in *Computer Vision—ECCV*, D. Fleet, T. Pajdla, B. Schiele, and T. Tuytelaars, Eds. Cham, Switzerland: Springer, 2014, pp. 818–833.

[95] E. Parzen, "On estimation of a probability density function and mode," *Ann. Math. Statist.*, vol. 33, no. 3, pp. 1065–1076, Sep. 1962. [Online]. Available: http://www.jstor.org/stable/2237880

[96] M. I. Jordan, Z. Ghahramani, T. S. Jaakkola, and L. K. Saul, "An introduction to variational methods for graphical models," *Mach. Learn.*, vol. 37, no. 2, pp. 183–233, Nov. 1999.

Chapter 7. Efficient Distribution Sampling for NLoS Identification

[97] *Study on Channel Model for Frequencies From 0.5 to 100 GHz (Rel-16)*, document TR 38.901, Version 16.1.0, 3GPP, Nov. 2020. [Online]. Available: https://www.etsi.org/deliver/etsitr/138900138999/138901/16.01.0060/tr138901v160100p.pdf

[98] C.-F. Yang and C.-J. Ko, "A ray tracing method for modeling indoor wave propagation and penetration," in *IEEE Antennas Propag. Soc. Int. Symp. Dig.*, vol. 1, Jul. 1996, pp. 441–444. [Online]. Available: https://ieeexplore.ieee.org/document/686780/

[99] H.-J. Li, C.-C. Chen, T.-Y. Liu, and H.-C. Lin, "Applicability of ray-tracing technique for the prediction of outdoor channel characteristics," *IEEE Trans. Veh. Technol.*, vol. 49, no. 6, pp. 2336–2349, Nov. 2000. [Online]. Available: https://ieeexplore.ieee.org/document/901902/

[100] A. Hsiao, C. Yang, T. Wang, I. Lin, and W. Liao, "Ray tracing simulations for millimeter wave propagation in 5G wireless communications," in *Proc. IEEE Int. Symp. Antennas Propag., USNC/URSI Nat. Radio Sci. Meeting*, Oct. 2017, pp. 1901–1902. [Online]. Available: https://ieeexplore.ieee.org/document/8072993/

[101] Z. Yun and M. F. Iskander, "Ray tracing for radio propagation modeling: Principles and applications," *IEEE Access*, vol. 3, pp. 1089–1100, 2015. [Online]. Available: https://ieeexplore.ieee.org/document/7152831/

[102] *Guidelines for Evaluation of Radio Interface Technologies for IMT-2020*, document, Report International Telecommunication Union (ITU), 2017. [Online]. Available: https://www.itu.int/dms_pub/itu-r/opb/rep/R-REP-M.2412-2017-PDF-E.pdf

[103] X. He, Q. Guo, J. Tong, J. Xi, and Y. Yu, "Low-complexity approximate iterative LMMSE detection for large-scale MIMO systems," *Digit. Signal Process.*, vol. 60, pp. 134–139, Jan. 2017. [Online]. Available: https://linkinghub.elsevier.com/retrieve/pii/S1051200416301452

[104] L. Gopal, Y. Rong, and Z. Zang, "Tomlinson–Harashima precoding based transceiver design for MIMO relay systems with channel covariance information," *IEEE Trans. Wireless Commun.*, vol. 14, no. 10, pp. 5513–5525, Oct. 2015. [Online]. Available: https://ieeexplore.ieee.org/abstract/document/7115189/

[105] A. Hindy and A. Nosratinia, "Ergodic fading MIMO dirty paper and broadcast channels: Capacity bounds and lattice strategies," *IEEE Trans. Wireless Commun.*, vol. 16, no. 8, pp. 5525–5536, Aug. 2017. [Online]. Available: https://ieeexplore.ieee.org/document/7942023/

[106] *5G; NR; Physical Channels and Modulation*, document TS 38.214, Version 16.12.0 Release 16, 3GPP, Jan. 2023. [Online]. Available: https://portal.3gpp.org/desktopmodules/Specifications/SpecificationDetails.aspx?specificationId=3216

[107] *5G; NR; Physical Channels and Modulation*, document TS 38.211, Version 16.10.0 Release 16, 3GPP, Jun. 2022. [Online]. Available: https://portal.3gpp.org/desktopmodules/Specifications/SpecificationDetails.aspx?specificationId=3213

[108] A. Paszke et al., "Automatic differentiation in PyTorch," Tech. Rep., 2017. [Online]. Available: https://openreview.net/forum?id=BJJsrmfCZ

[109] D. P. Kingma and J. Ba, "Adam: A method for stochastic optimization," Jan. 2014, *arXiv:1412.6980*.

[110] V. Badrinarayanan, A. Kendall, and R. Cipolla, "SegNet: A deep convolutional encoder–decoder architecture for image segmentation," *IEEE Trans. Pattern Anal. Mach. Intell.*, vol. 39, no. 12, pp. 2481–2495, Dec. 2017. [Online]. Available: https://ieeexplore.ieee.org/document/7803544/

[111] N. Srivastava, G. Hinton, A. Krizhevsky, I. Sutskever, and R. Salakhutdinov, "Dropout: A simple way to prevent neural networks from overfitting," *J. Mach. Learn. Res.*, vol. 15, no. 1, pp. 1929–1958, 2014.

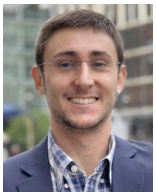
[112] S. Ioffe and C. Szegedy, "Batch normalization: Accelerating deep network training by reducing internal covariate shift," in *Proc. 32nd Int. Conf. Mach. Learn.*, in Proceedings of Machine Learning Research, vol. 37, F. Bach and D. Blei, Eds. Lille, France: PMLR, Jul. 2015, pp. 448–456. https://proceedings.mlr.press/v37/ioffe15.html

[113] S. Akcay, A. Atapour-Abarghouei, and T. P. Breckon, "GANomaly: Semi-supervised anomaly detection via adversarial training," in *Computer Vision—ACCV*, C. V. Jawahar, H. Li, G. Mori, and K. Schindler, Eds. Cham, Switzerland: Springer, May 2019, pp. 622–637.

[114] C. Huang et al., "Machine learning-enabled LOS/NLOS identification for MIMO systems in dynamic environments," *IEEE Trans. Wireless Commun.*, vol. 19, no. 6, pp. 3643–3657, Jun. 2020. [Online]. Available: https://ieeexplore.ieee.org/document/8968748/

[115] S. Sun, J. Hu, Y. Peng, X. Pan, L. Zhao, and J. Fang, "Support for vehicle-to-everything services based on LTE," *IEEE Wireless Commun.*, vol. 23, no. 3, pp. 4–8, Jun. 2016. [Online]. Available: https://ieeexplore.ieee.org/document/7498068/

[116] D. P. Kingma and M. Welling, "Auto-encoding variational Bayes," Dec. 2013, *arXiv:1312.6114*.



Bernardo Camajori Tedeschini (Graduate Student Member, IEEE) received the B.Sc. degree (Hons.) in computer science and M.Sc. degree (Hons.) in telecommunications engineering from the Politecnico di Milano, Italy, in 2019 and 2021, respectively. From November 2021, he started as Ph.D. fellow in Information Technology at Dipartimento di Elettronica, Informazione e Bioingegneria (DEIB), Politecnico di Milano. He is currently a Visiting Researcher with the Laboratory for Information and Decision Systems at the Massachusetts Institute of Technology (MIT), Cambridge, MA.

His research interests include federated learning, machine learning and localization methods. He was a recipient of the Ph.D. grant from the ministry of the Italian government Ministero dell'Istruzione, dell'Università e della Ricerca (MIUR) and the Roberto Rocca Doctoral Fellowship granted by MIT and Politecnico di Milano.

Monica Nicoli (Senior Member, IEEE) received the M.Sc. (Hons.) and Ph.D. degrees in communication engineering from Politecnico di Milano, Milan, Italy, in 1998 and 2002, respectively. She was a Visiting Researcher with ENI Agip, from 1998 to 1999, and Uppsala University, in 2001. In 2002, she joined Politecnico di Milano as a Faculty Member. She is currently an Associate Professor in telecommunications with the Department of Management, Economics and Industrial Engineering.

Her research interests include signal processing, machine learning, and wireless communications, with emphasis on smart mobility and Internet of Things (IoT) applications. She was a recipient of the Marisa Bellisario Award, in 1999, and a co-recipient of the best paper awards of the IEEE Symposium on Joint Communications and Sensing, in 2021, the IEEE Statistical Signal Processing Workshop, in 2018, and the IET *Intelligent Transport Systems journal*, in 2014. She is an Associate Editor of the IEEE TRANSACTIONS ON INTELLIGENT TRANSPORTATION SYSTEMS. She has also served as an Associate Editor for the EURASIP *Journal on Wireless Communications and Networking*, from 2010 to 2017, and a Lead Guest Editor for the Special Issue on Localization in Mobile Wireless and Sensor Networks, in 2011.

Moe Z. Win (Fellow, IEEE) is a Professor at the Massachusetts Institute of Technology (MIT) and the founding director of the Wireless Information and Network Sciences Laboratory. Prior to joining MIT, he was with AT&T Research Laboratories and with NASA Jet Propulsion Laboratory.

His research encompasses fundamental theories, algorithm design, and network experimentation for a broad range of real-world problems. His current research topics include ultra-wideband systems, network localization and navigation, network interference exploitation, and quantum information science. He has served the IEEE Communications Society as an elected Member-at-Large on the Board of Governors, as elected Chair of the Radio Communications Committee, and as an IEEE Distinguished Lecturer. Over the last two decades, he held various editorial positions for IEEE journals and organized numerous international conferences. Recently, he has served on the SIAM Diversity Advisory Committee.

Dr. Win is an elected Fellow of the AAAS, the EURASIP, the IEEE, and the IET. He was honored with two IEEE Technical Field Awards: the IEEE Kiyo Tomiyasu Award (2011) and the IEEE Eric E. Sumner Award (2006, jointly with R. A. Scholtz). His publications, coauthored with students and colleagues, have received several awards. Other recognitions include the MIT Everett Moore Baker Award (2022), the IEEE Vehicular Technology Society James Evans Avant Garde Award (2022), the IEEE Communications Society Edwin H. Armstrong Achievement Award (2016), the Cristoforo Colombo International Prize for Communications (2013), the Copernicus Fellowship (2011) and the *Laurea Honoris Causa* (2008) from the Università degli Studi di Ferrara, and the U.S. Presidential Early Career Award for Scientists and Engineers (2004). He is an ISI Highly Cited Researcher.

Open Access funding provided by 'Politecnico di Milano' within the CRUI CARE Agreement

CHAPTER 8

Efficient Latent Features Combination for Static Positioning

In this chapter, we deal with the task of CP through next-generation BSs which aim at performing static positioning of a UE in an urban environment. The challenges are how to perform real-time positioning in both LoS and NLoS environments by exploiting the neighbors' output in an efficient manner. In the paper, we propose an AE-based model to extract an efficient non-linear representation of channel, comprising all RSS, ToF, and AoA for every path. Subsequently, a NLoS identification model, trained in a supervised way, assigns an estimate of the NLoS probability. In case we are in NLoS, the latent features are used as fingerprint to perform localization with a single-BS, i.e., egocentric mode. On the contrary, in the case of LoS condition, the latent features are exchanged among BSs, carefully combined, and adopted as input in a NN for position estimation. In the simulations, we adopted ray-tracing technology for realistic channel representation, and Simulation of Urban MObility (SUMO) software for creating realistic C-ITS environment in an UMi environment. The results demonstrate that the cooperative architecture particularly enhances the performance over traditional geometric algorithms, and solves the limits of single-BS predictions with DL by dynamically changing strategies based on the scenario.

IEEE JOURNAL ON SELECTED AREAS IN COMMUNICATIONS, VOL. 41, NO. 12, DECEMBER 2023 3799

Cooperative Deep-Learning Positioning in mmWave 5G-Advanced Networks

Bernardo Camajori Tedeschini, *Graduate Student Member, IEEE*, and Monica Nicoli, *Senior Member, IEEE*

***Abstract*— In application verticals that rely on mission-critical control, such as cooperative intelligent transport systems (C-ITS), 5G-Advanced networks must be able to provide dynamic positioning with accuracy down to the centimeter level. To achieve this level of precision, technology enablers, such as massive multiple-input multiple-output (mMIMO), millimeter waves (mmWave), machine learning and cooperation are of paramount importance. In this paper, we propose a cooperative deep learning (DL)-based positioning methodology that combines these key technologies into a new promising solution for precise 5G positioning. Sparse channel impulse response (CIR) data are used by the positioning infrastructure to extract position-dependent features. We model the problem as a joint task composed of non-line-of-sight (NLOS) identification and position estimation which permits to suitably handle geometrical location measurements and channel fingerprints. The network of base stations (BSs) automatically steers between egocentric (in case of NLOS) and cooperative (for LOS) positioning mode. We perform extensive standard-compliant simulations in a 5G urban micro (UMi) vehicular scenario obtained by ray-tracing and simulation of urban mobility (SUMO) software. Results show that the proposed cooperative DL architecture is able to outperform conventional geometrical positioning algorithms operating in LOS by 47%, achieving a median error of 71 cm on unseen trajectories.**

***Index Terms*— Cooperative positioning, deep learning, 5G, channel impulse response, cooperative intelligent transport systems.**

I. Introduction

THE currently deployed release of 5th generation (5G) cellular networks, 3rd generation partnership project (3GPP) Release 16, lays the foundations for future location awareness systems that are foreseen to extend the location services (LCS) beyond regulatory use cases (i.e., emergency and lawful interception) to include roaming and commercial capabilities [1], [2], [3], [4]. These systems are made possible by the use of higher frequencies, i.e., millimeter waves (mmWave), with enlarged bandwidth, and massive multiple-input multiple-output (mMIMO) technologies [5], which enhance radio access technology (RAT)-dependent measurements in accordance with 3GPP standards [6], [7]. However, future releases of 5G and beyond, such as 5G-Advanced from Release 18, will face challenges in achieving the centimeter-level absolute accuracy required by the most demanding 5G use cases due to higher path loss and frequent blockages [8], [9].

Legacy solutions such as least squares (LS) multi-lateration/angulation may struggle to effectively handle situations with low signal-to-noise ratio (SNR), multipath ambiguities, and particularly non-line-of-sight (NLOS) conditions [10], [11]. A potential solution is already foreseen by a novel paradigm called integrated sensing and communication (ISAC) [12], [13], [14]. Specifically, 5G base stations (BSs) (also known as gNodeBs) can natively support ISAC through the use of a joint signal processing framework, allowing for a more efficient utilization of spectrum resources. The integration of communication and sensing features on the same hardware platform is, at present time, not yet been commercialized. Nevertheless, synthetic datasets are emerging [15], [16], [17] with the clear objective of permitting the design of novel artificial intelligence (AI) and machine learning (ML) algorithms which will play a fundamental role in next-generation networks [18], [19], [20], [21], [22].

In the context of cooperative intelligent transport systems (C-ITS), connected automated vehicles (CAV) rely on ML, and more specifically deep learning (DL), for various functions, including identifying and segmenting objects within images, controlling the vehicle and avoiding collisions, and determining the most efficient route [23], [24]. Because of the high complexity of such tasks (including precise positioning), urban areas may install computing units, namely roadside units (RSUs), on busy roads that CAV will be able to use to offload part of the computing activities [25], [26]. Cooperation between nearby RSUs, here referred to as BSs, is of paramount importance for enabling network-based precise localization [27], [28]. A key aspect is that, in 5G industrial applications such as automated driving, BSs often have access to a large amount of historical channel state information (CSI) data that is continuously received from geolocalized vehicles [29], [30].

While having a perfect knowledge of the overall CSI is unfeasible, e.g., accurate delays, angle of arrivals (AoAs) and power gains, the usage of estimates of the channel impulse response (CIR) can be adopted by ML approaches to learn relevant information about the environment and its propagation characteristics [31]. It is important to observe that not only LOS but also NLOS CIRs embed significant location information, though embedded in distinct peculiar distributions. Therefore, CIRs can be exploited in both cases

Manuscript received 15 February 2023; revised 13 August 2023; accepted 13 August 2023. Date of publication 9 October 2023; date of current version 22 November 2023. This work was supported in part by a Ph.D. Grant from the Ministry of the Italian Government Ministero dell'Istruzione, dell'Università e della Ricerca (MIUR), and in part by the project Centro Nazionale per la Mobilità Sostenibile (MOST), funded by the Italian Ministry of University and Research under the PNRR funding program. *(Corresponding author: Bernardo Camajori Tedeschini.)*

Bernardo Camajori Tedeschini is with Dipartimento di Elettronica, Informazione e Bioingegneria (DEIB), Politecnico di Milano, 20133 Milan, Italy (e-mail: bernardo.camajori@polimi.it).

Monica Nicoli is with Dipartimento di Ingegneria Gestionale (DIG), Politecnico di Milano, 20133 Milan, Italy (e-mail: monica.nicoli@polimi.it).

Color versions of one or more figures in this article are available at https://doi.org/10.1109/JSAC.2023.3322795.

Digital Object Identifier 10.1109/JSAC.2023.3322795

for positioning goals. Distinguishing between LOS/NLOS conditions is extremely important since the derived measurements are fundamentally different. In case of LOS condition, the extracted features can be combined by a cooperative set of BSs as in conventional geometric methods in order to enhance satellite positioning. On the contrary, NLOS features represent a real challenge for the system since they are related to a specific environment, acting as area-specific fingerprints. A complete solution for both situations has yet to be found. Therefore, in this paper, we propose a cooperative DL solution for the joint problem of NLOS identification and 3D position estimation that takes as input the high-dimensional sparse CIRs. We model the problem of NLOS position estimation as an egocentric system at each BS, while a cooperative architecture is proposed for LOS conditions.

Table I and II list the main abbreviations and notations used throughout the paper, respectively.

A. Paper Organization

The remainder of this article is structured as follows. Sec. II presents an overview of the state of the art on wireless localization with ML and DL. Sec. III describes the system model, the MIMO-orthogonal frequency division multiplexing (OFDM) channel and related angle-delay channel power matrix (ADCPM) adopted as input for ML-based positioning. In Sec. IV, we discuss the proposed supervised ML setting and the DL model stored in each BS for the joint NLOS detection-positioning task prediction. Sec. V extends the model to be compliant with a cooperative architecture based on a set of collaborative BSs for positioning purposes. Section VI provides information on the simulated 5G scenario and compares the proposed method with conventional positioning algorithms. Finally, in Section VII, we draw the conclusions.

II. Related Works and Contributions

A. Early Works on NLOS Detection and Positionong

First works on NLOS identification and localization with ML were applied to ultra wide-band (UWB) systems. They typically used hand-crafted features of the channel, such as energy, maximum power, rise time, mean excess delay, root-mean-square delay spread, and kurtosis, as inputs [32], [33], [34], [35], [36]. The most commonly used ML models in these studies were Gaussian processess (GPs), support vector machines (SVMs), and relevance vector machines (RVMs). While achieving good results, these methods could not fully exploit the ML potential as they heavily relied on a pre-defined and limited set of features that could not express the whole location information enclosed in the CSI.

Other methods, mainly based on received signal strength (RSS) fingerprinting, were proposed for precise positioning using Wi-Fi technology [37], [38]. With the advent of MIMO-OFDM systems in IEEE 802.11a/n protocol, allowing for the extraction of CSI from commercial Wi-Fi devices, there has been significant research on wireless positioning, behavioral awareness, and target tracking using CSI. The access to channel information over multiple carriers and antennas gave the possibility to extract detailed information about the propagation of the radio signal, and to learn not only the position of the user [39], [40], [41] but also information about the environment that shaped such propagation [42]. DL approaches were employed to directly learn the optimal non-linear combination of features and produce the desired NLOS classification or position estimation as output. Studies in this field can be found in [43], [44], [45], and [46], adopting convolutional neural networks (CNN) for feature extraction.

B. DL for High-Precision Positioning

In outdoor conditions, DL-based positioning is a relatively new concept, but with a great potential of achieving high levels of accuracy. Authors in [16] achieve a mean positioning error of 1.5 m by using a cell-specific neural network (NN) in 5G Rel. 15 networks, but considering LOS environments only. NLOS prediction is handled through statistical tests [47] or directly included into the model's prediction [48], [49]. A recent study [50] adopted a variational autoencoder (VAE) to extract features and impose a Gaussian distribution on the latent features for the purpose of binary classification of samples as LOS or NLOS. While the use of an autoencoder (AE) to obtain a compact representation of the channel can yield good results, the reliance on sampling-based methods for prediction makes this approach not suitable for real-time applications.

The employment of full CIR data, especially stacked into image-shapes, has emerged recently as a promising approach. Authors in [51] adopted both CIR (i.e., path power gain, phase and time of flight (ToF)) as well as geographical information (i.e., AoA and angle of departure (AoD)) to predict the user equipment (UE) location. However, they assumed perfect CIR knowledge by ray-tracing, which is hard to achieve under practical conditions. Authors in [52] employed as input the channel frequency response (CFR) matrix computed by practical channel estimation and augmented with additive noise at training time. Despite achieving good results, the CFR matrix does not explicitly express AoA nor ToF of each path and thus may add complexity in feature extraction. A recent work [53] adopted a 3D CNN with inception modules to directly predict the position of a UE from an ADCPM. This approach, however, highly relies on the fingerprint sampling-distance and it is not able to distinguish between LOS and NLOS conditions, treating each position as equal. In other words, there are no distinctions between geometrical features, useful in LOS environments, and merely NLOS position-dependent fingerprints. Furthermore, no works are available in the literature on DL-based location estimation using data collected by multiple cells, i.e., by the cooperation of BS's. Works mainly rely on centralized processing [54] or vehicle-to-vehicle (V2V) communications [55], [56].

C. Contributions

In this paper, we address the problem of precise cooperative positioning in urban environments covered by 5G mm-wave networks and we propose a new DL-based approach that allows to exploit the full potential of wideband space-time CIR for localization. The main contributions are as follows.

- We propose a new method for the extraction of location-related features from the CIR of 5G mmWave

TABLE I
LIST OF ABBREVIATIONS

ACRONYM	DEFINITION
ADCPM	Angle-delay channel power matrix
ADCRM	Angle-delay channel response matrix
AoA	Angle of arrival
AoD	Angle of departure
BS	Base station
CDF	Cumulative density function
CFR	Channel frequency response
CIR	Channel impulse response
CNN	Convolutional neural networks
CSI	Channel state information
C-ITS	Cooperative intelligent transport systems
DL	Deep learning
GDoP	Geometrical dilution of precision
ISD	Inter-site distance
ML	Machine learning
MLP	Multi-layer perceptron
mMIMO	Massive multiple-input multiple-output
NN	Neural network
NLOS/LOS	Non-/Line-of-sight
NLS	Non-linear least squares
SFCRM	Space-frequency channel response matrix
TDoF	Time difference of flight
ToF	Time of flight
UE	User equipment
UPA	Uniform planar array
VAE/AE	Variational/ Autoencoder

TABLE II
LIST OF NOTATIONS

NOTATION	DEFINITION
K	N. of UEs
$\mathcal{S}_{\text{BS},i}$	Set of BSs detecting a same UE at time i
$\mathcal{N}_i^{(j)}$	Set of neighboring BSs for BS j at time i
λ_c	Carrier wavelength
N, M	N. of antennas in vertical and horizontal directions
$d^{(v)}, d^{(h)}$	Antenna spacing in vertical and horizontal directions
N_k	N. of paths between the BS and user k
$\theta_{k,p}, \phi_{k,p}$	Zenith and azimuth AoA for UE k and path p
$T_\text{s}, T_\text{c}, T_\text{g}$	Sampling, symbol and cyclic prefix interval
N_c	N. of sub-carriers
N_g	Cyclic prefix length in sampling intervals
$r_{k,p}$	Resolvable propagation delay of UE k and path p
$\tau_{k,p}$	Propagation delay for UE k and path p
$\alpha_{k,p}$	Complex channel gain for UE k and path p
$\bar{\alpha}_{k,p}$	Equivalent gain in frequency domain for UE k and path p
$\nu_{k,p}$	Doppler frequency shift for UE k and path p
$\mathbf{q}_k(\tau)$	Baseband CIR of UE k
$\mathbf{h}_{k,\ell}$	CFR for UE k at the sub-carrier l
$\mathbf{H}_k, \mathbf{G}_k, \mathbf{P}_k$	SFCRM, ADCRM and ADCPM, respectively
$\mathbf{x}_i^{(j)}, \hat{\mathbf{x}}_i^{(j)}$	ADCPM input and related estimate at time i for BS j
$\mathbf{z}_i^{(j)}$	Latent features of sample at time i and BS j
$\mathbf{u}_i, \hat{\mathbf{u}}_i$	Real and estimated UE position at time i
$\hat{p}_{s,i}^{(j)}$	LOS probability at time i and BS j
$\mathbf{W}$	NN parameters
N_b	N. of samples per batch

OFDM MIMO links. The channel is parameterized in terms of ADCPM matrices, typically sparse, that can be treated as images and used to capture the location-dependent features. The sparse ADCPM directly includes the AoAs and ToFs and is fully compliant with the 3GPP Release 16 standard, providing CSI data that is representative of real 5G communication.

- We model the problem of NLOS identification and position estimation as a joint task, proposing a novel loss function to simultaneously learn a compact representation of the channel, i.e., latent features, and maximize the log-likelihood of the joint task.
- We develop a DL model for network-based localization that automatically steers between ego-positioning in NLOS environments and cooperative-positioning in LOS situations, exploiting neighbors BSs predictions to enhance the location accuracy.
- We simulate a realistic C-ITS scenario in an urban environment. The simulated network is fully compliant with the 5G standard [57] and provides realistic outdoor conditions through the use of Matlab ray-tracing software. We simulate multiple trajectories of vehicles, i.e., UEs, created with simulation of urban mobility (SUMO) software [58]. Simulations are used to assess the performance of the DL-based method, showing significant gains over conventional techniques.

Notation

We denote with $j = \sqrt{-1}$ the imaginary unit. Columns vectors and matrices are denoted by lower- and upper-case characters, respectively. Matrix conjugation, transposition and conjugate transposition are indicated as $\mathbf{A}^H$, $\mathbf{A}^T$ and $\mathbf{A}^*$, respectively. We indicate the Hadamard and Kronecker product between two matrices with $\odot$ and $\otimes$, respectively. The symbol $\mathbb{E}[\cdot]$ is used for expectation of random variable, whereas $\mathbb{C}$ and $\mathbb{R}$ are the set of complex and real numbers, respectively. We denote with $\mathcal{N}(x; \mu, \sigma^2)$ the distribution of a Gaussian random variable x with mean μ and standard deviation σ. $\delta(\cdot)$ and $\delta[\cdot]$ indicate the Dirac delta and Kronecker functions, respectively, while $\lfloor x \rfloor$ represents the largest integer not greater than x.

III. SYSTEM MODEL

In this section, we define the channel model and the location-related channel fingerprints that will be used by the proposed DL positioning algorithms in Sec. IV. For simplicity of notation, we here report the multi-user single-BS case, leaving the extension to a set of cooperative BSs in Sec. V.

A. Channel Model

Let us consider a wide-bandwidth multi-user MIMO-OFDM system operating at carrier wavelength λ_c where K UEs communicate with a BS in uplink direction. The UE is equipped with an omni-directional antennas, whereas a uniform planar array (UPA) with $N \times M$ isotropic antenna elements (i.e., N and M elements in the vertical and horizontal directions, respectively) is installed in the cell panel of the BS. The antenna elements are separated by a distance $d^{(h)}$ and $d^{(v)}$ in the horizontal and vertical direction, respectively. A multipath channel with N_k paths is present between the BS and the UE k. The overall scenario is represented in Fig. 1, where the generic p-th path of the k-th UE channel is represented, jointly with the direction of arrivals (DoAs) of the signal impinging

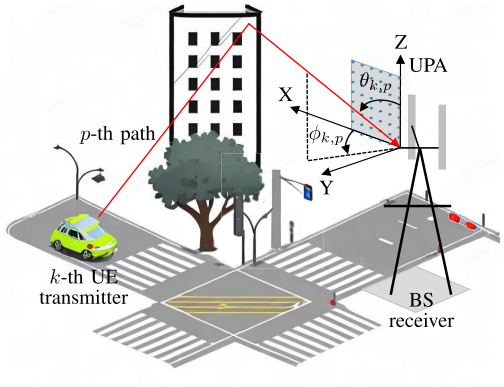


Fig. 1. BS receiving the uplink signal from the k-th UE through an UPA. The DoA of the p-th path is composed by the zenith angle $\theta_{k,p}$ and the azimuth angle $\phi_{k,p}$.

the antenna panel, composed by a zenith angle $0 \leq \theta_{k,p} \leq \pi$ and an azimuth angle $0 \leq \phi_{k,p} \leq \pi$.

We consider an OFDM modulation with sampling interval T_s, N_c sub-carriers and symbol duration $T_\mathrm{c} = N_\mathrm{c}T_\mathrm{s}$. The ℓ-th sub-carrier has frequency $f_\ell = \frac{\ell}{T_\mathrm{c}}$, $\ell = 0, \ldots, N_\mathrm{c}-1$ and we assume that the cyclic-prefix duration $T_\mathrm{g} = N_\mathrm{g}T_\mathrm{s}$ is greater than the maximum channel delay among all UEs, τ_{MAX}, while N_g is the number of sampling interval that composes a guard interval. The temporally resolvable propagation delay related to the p-th path and k-th UE is indicated with $r_{k,p} = \lfloor \frac{\tau_{k,p}}{T_\mathrm{s}} \rfloor$. According to this notation, we model the baseband CIR of user k as [59]:

$$\mathbf{q}_k(\tau) = \sum_{p=1}^{N_k} a_{k,p}\, e^{-j2\pi(\frac{d_{k,p}}{\lambda_\mathrm{c}} - \nu_{k,p}\tau_{k,p})}\, \mathbf{e}(\theta_{k,p}, \phi_{k,p})\delta(\tau - \tau_{k,p}), \tag{1}$$

where $\alpha_{k,p} = a_{k,p}e^{-j2\pi(\frac{d_{k,p}}{\lambda_\mathrm{c}} - \nu_{k,p}\tau_{k,p})}$ is the complex gain of p-th path which also includes the frequency shift due to Doppler $\nu_{k,p}$ and has average power $\sigma^2_{k,p} = \mathbb{E}[|a_{k,p}|^2]$, $d_{k,p} = c\tau_{k,p}$ is the traveled distance (with c being the speed of light in air), $\delta(\tau - \tau_{k,p})$ is the delta Dirac function and $\mathbf{e}(\theta_{k,p}, \phi_{k,p}) \in \mathbb{C}^{MN}$ the array response vector [60]. We assume that over the time interval τ_{MAX}, the rotation due to the Doppler is almost constant.

Considering sampling time with rate $1/T_\mathrm{s}$ and assuming each different path independent and wide sense stationary [59], we can write the CFR at the ℓ-th sub-carrier as [61]:

$$\mathbf{h}_{k,\ell} = \sum_{p=1}^{N_k} \bar{\alpha}_{k,p}\mathbf{e}(\theta_{k,p}, \phi_{k,p}), \tag{2}$$

where $\bar{\alpha}_{k,p} = \alpha_{k,p}e^{-j2\pi\tau_{k,p}f_l}$ are the equivalent channel gains in the frequency domain. Concatenating the different CFRs at each sub-carrier, we obtain the space-frequency channel response matrix (SFCRM):

$$\mathbf{H}_k = [\mathbf{h}_{k,0}\ \mathbf{h}_{k,1}\ \ldots\ \mathbf{h}_{k,N_\mathrm{c}-1}] \in \mathbb{C}^{MN \times N_\mathrm{c}}, \tag{3}$$

which will be used in the next section for ADCPM extraction.

B. ADCPM Location Fingerprints

For location estimation, it is convenient to convert the channel response to the angle-delay domain, where the identification of the LOS component (if present) and secondary NLOS macro-paths is facilitated. In fact, in LOS condition, AoA and ToF can be effectively used to localize a UE thanks to their geometric relationship with the location. At the same time, in NLOS circumstances, different surrounding environments hold different channel parameters, acting as location-dependent features (or fingerprints). Therefore, we transform the SFCRM in (3) into an angle-delay channel response matrix (ADCRM) by introducing the phase-shifted discrete Fourier transform (DFT) matrices $\mathbf{V}_M \in \mathbb{C}^{M\times M}$ and $\mathbf{V}_N \in \mathbb{C}^{N\times N}$, where $[\mathbf{V}_M]_{i,j} = \frac{1}{\sqrt{M}}e^{-j2\pi\frac{i(j-\frac{M}{2})}{M}}$ and $[\mathbf{V}_N]_{i,j} = \frac{1}{\sqrt{N}}e^{-j2\pi\frac{i(j-\frac{N}{2})}{N}}$. We denote with $\mathbf{F} \in \mathbb{C}^{N_\mathrm{c}\times N_\mathrm{g}}$ the matrix formed by the first N_g columns of N_c dimensional unitary DFT matrix where $[\mathbf{F}]_{i,j} = \frac{1}{\sqrt{N_\mathrm{c}}}e^{-j2\pi\frac{ij}{N_\mathrm{c}}}$. Finally, we compute the ADCRM as [53]:

$$\mathbf{G}_k = \frac{1}{\sqrt{MNN_\mathrm{c}}}(\mathbf{V}_M^H \otimes \mathbf{V}_N^H)\mathbf{H}_k\mathbf{F}^* \in \mathbb{C}^{MN\times N_\mathrm{g}}, \tag{4}$$

where $\mathbf{F}^*$ and $(\mathbf{V}_M^H \otimes \mathbf{V}_N^H)$ project the SFCRM into the delay and angle domain, respectively.

For positioning purposes, we propose to use power-angle-delay profile, represented by the ADCPM:

$$\mathbf{P}_k = \mathbb{E}[\mathbf{G}_k \odot \mathbf{G}_k^*] \in \mathbb{C}^{MN\times N_\mathrm{g}}, \tag{5}$$

where $[\mathbf{P}_k]_{i,j} = \mathbb{E}[|[\mathbf{G}_k]_{i,j}|^2]$. It can be shown indeed that for N, M and $N_\mathrm{g} \to \infty$, the ADCPM approximates a sparse matrix with elements $[\mathbf{P}_k]_{i,j}$ matching the i-th AoA and the j-th ToF [53]:

$$\lim_{M,N,N_\mathrm{g}\to\infty}[\mathbf{P}_k]_{i,j} = \sum_{p=1}^{N_k}\sigma^2_{k,p}\delta[i - m_{k,p}N - n_{k,p}]\delta[j - r_{k,p}], \tag{6}$$

where $r_{k,p}$ indicates the index of the j-th ToF and $m_{k,p}N + n_{k,p}$ refers to the index of the i-th AoA. Therefore, the statistical information of the ADCPM enables the learning by the DL model of the location-dependent characteristics, delivering steady and trustworthy fingerprints for positioning.

C. DL Model Input

We propose to employ the ADCPM as a set of measurements for location estimation. This sparse matrix provides indeed a visual image of the multipath configuration in the power-angle-delay domain from which a DL model such as CNN can extract the most representative location-related features. Additionally, since the first layers of CNN are often sparse and collect the highly discriminating features, the CNN eases features extraction from the ADCPM sparse channel matrix [62]. Moreover, the ADCPM embodies all the relevant information (i.e., ToF, AoA and RSS for every path) with small storage and low complexity characteristics thanks to the channel sparsity. To highlight this aspect, in Fig. 2 we show an example of ADCPM $\mathbf{P}_k$ composed by $N_\mathrm{g} = 352$ delay

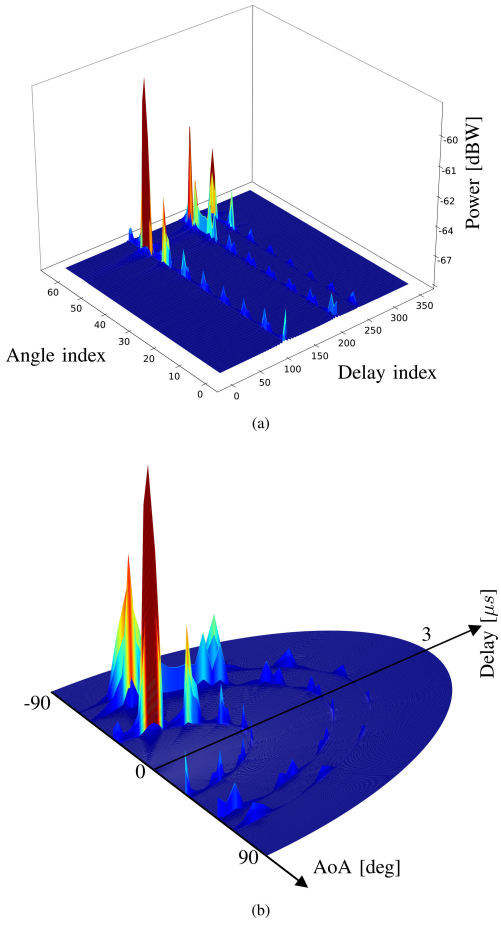


Fig. 2. Example of sparse channel power-delay-angle profile encoding the UE location, represented by an ADCPM with $N_g = 352$ temporal samples and $NM = 64$ spatial samples (resulting from $N = M = 8$ antennas) (a) and in the transformed angle-delay domain (b).

samples and $MN = 64$ angle directions. The actual model input can be seen in Fig. 2a, while the polar representation of the ADCPM with physical AoAs and ToFs is in Fig. 2b. Even in the absence of a large number of antennas or a high sample resolution, the matrix sparsity is clearly visible. Therefore, we propose to use the ADCPM as input to the model and we denote the i-th sample with $\mathbf{x}_i = \mathbf{P}_i$, dropping the index k for ease of notation.

IV. Methodology: Single-BS Localization

In this section, we first tackle the localization problem in a supervised setting in which a single BS has to locate the UE. We propose a DL model (Sec. IV-A) and a loss function for the joint task of position estimation and LOS identification (Sec. IV-B). The approach will then be extended to the multi-BS, i.e., cooperative, case in Sec. V.

A. Deep Learning Model

We assume a supervised ML setting in which both a regression and a classification problem have to be solved by a single BS. The regression problem refers to the estimation of the UE's position, while the classification problem concerns the LOS/NLOS identification. We define the training dataset as $\mathcal{S}^{\text{train}} = \{(\mathbf{x}_i, \mathbf{u}_i, s_i)\}_{i=1}^{N_{\text{train}}}$, where $\mathbf{x}_i = \mathbf{P}_i$ denotes the i-th input sample (i.e., ADCPM channel response), $\mathbf{u}_i$ represents the 3D position and $s_i \in \{0, 1\}$ is the Boolean identifier of the sight condition. To validate the performances, we also hold a similar test dataset $\mathcal{S}^{\text{test}}$ composed of N_{test} samples.

We note that the regression and classification tasks are two interrelated problems that must be addressed accordingly. In fact, if the UE lies in a LOS condition, its position can be directly computed from the geometrical features of the direct path (i.e., ToF, AoA and mean power), while NLOS typically requires a finer training based on more complex multipath fingerprints.

To extract such key features from the ADCPM samples $\mathbf{x}_i$, we propose to employ an AE, with structure represented in Fig. 3. The encoder $\mathcal{E}(\cdot)$ is used to produce the hidden (or latent) features $\mathbf{z}_i$ (including the location-related information embedded in the channel), while the decoder $\mathcal{D}(\cdot)$ tries to reconstruct the input samples obtaining $\hat{\mathbf{x}}_i$. The AE is designed so as to minimize a metric of the reconstruction error $\|\mathbf{x}_i - \hat{\mathbf{x}}_i\|_2^2$ [63], making the model able to reconstruct the input $\mathbf{x}_i$ from the low-dimensional data $\mathbf{z}_i$. This guarantees that $\mathbf{z}_i$ contains all the necessary and sufficient information to accomplish the specific task.

The tasks herein considered are position regression and LOS identification. Therefore, in principle, two NNs with $\mathbf{z}_i$ as input, should be sufficient. However, due to the major difference between position estimation in LOS and NLOS conditions, we propose to use three separate NNs: one for LOS identification, one for position regression in LOS conditions and one for position regression in NLOS conditions. Given the overall model with parameters defined as $\mathbf{W}$, the output of the NNs is respectively: $\hat{p}_{s,i} \in [0, 1]$, $\hat{\mathbf{u}}_{\text{LOS},i} \in \mathbb{R}^3$ and $\hat{\mathbf{u}}_{\text{NLOS},i} \in \mathbb{R}^3$. Specifically, $\hat{p}_{s,i} = p(s_i)$ predicts the probability that the sample $\mathbf{x}_i$ relates to a UE in LOS condition, while $\hat{\mathbf{u}}_{\text{LOS},i}$ and $\hat{\mathbf{u}}_{\text{NLOS},i}$ predict the 3D position of the UE in LOS and NLOS settings, respectively. The overall position estimate, indicated with $\hat{\mathbf{u}}_i$, is obtained by applying a threshold Γ to $\hat{p}_{s,i}$ and considering only $\hat{\mathbf{u}}_{\text{LOS},i}$ or $\hat{\mathbf{u}}_{\text{NLOS},i}$ according to the result:

$$\hat{\mathbf{u}}_i = \Gamma(\hat{p}_{s,i})\hat{\mathbf{u}}_{\text{LOS},i} + \Gamma(1 - \hat{p}_{s,i})\hat{\mathbf{u}}_{\text{NLOS},i}. \tag{7}$$

The position estimates in LOS and NLOS are then adopted in the loss function for the model training, described in the next section.

B. Loss Function for Joint Sight Detection and Localization

In order to train the model, we have to define a loss function whose objective is to enable learning the correct representation of the latent features and, at the same time, jointly estimating the sight condition and the location. To this aim, we first treat separately the classification and regression tasks and then we combine them with an overall objective

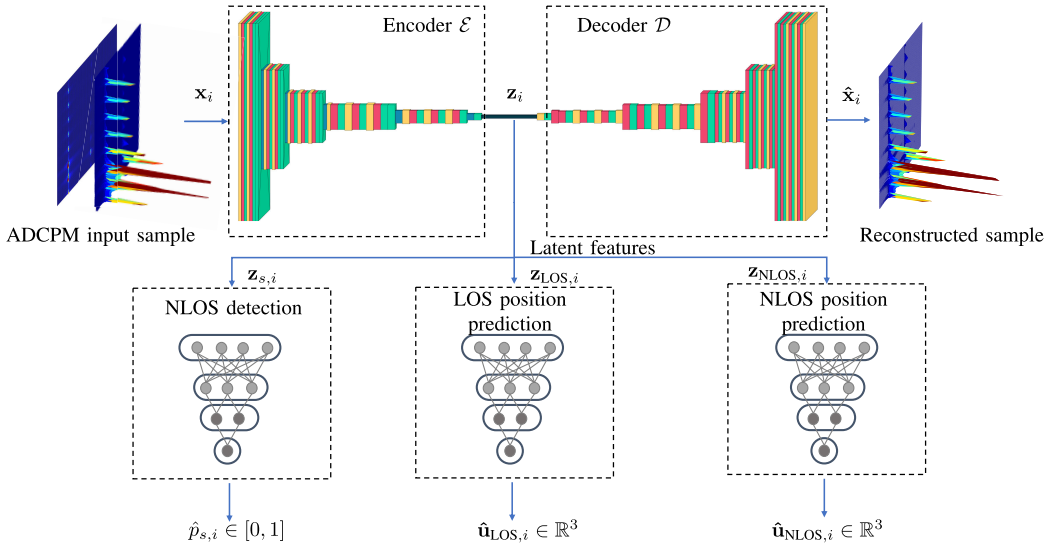


Fig. 3. Overview of the proposed model composed by an autoencoder (AE) for feature extraction, and 3 neural networks (NNs) for NLOS classification and position estimation in both LOS and NLOS conditions.

function. For the classification task, we propose a discriminative probabilistic approach, namely the maximum likelihood [64], where we directly define the posterior conditional probability $p(s_i|\mathbf{x}_i)$ using a parametric model $\mathbf{W}$, i.e., $p(s_i|\mathbf{x}_i) = p(s_i|\mathbf{x}_i, \mathbf{W})$. Subsequently, we maximize the likelihood of the model $p(s_i|\mathbf{x}_i, \mathbf{W})$, by optimizing the parameters $\mathbf{W}$ over the training set. For the specific binary classification problem, the likelihood function is:

$$p(s_i|\mathbf{x}_i, \mathbf{W}) = \hat{p}_{s,i}\delta[s_i - 1] + (1 - \hat{p}_{s,i})\delta[s_i]. \tag{8}$$

For the regression task, we reiterate the discriminative approach with the constraint of belonging to a specific condition, either LOS or NLOS. Again, this is done considering that the two circumstances hold different statistics. We assume that in the LOS case, the target variable $\mathbf{u}_i$ (i.e., the location) is Gaussian distributed with a deterministic mean $\hat{\mathbf{u}}_{\text{LOS},i}(\mathbf{x}_i, \mathbf{W})$:

$$\mathbf{u}_i = \hat{\mathbf{u}}_{\text{LOS},i}(\mathbf{x}_i, \mathbf{W}) + \boldsymbol{\epsilon}_{\text{LOS}}, \tag{9}$$

where $\boldsymbol{\epsilon}_{\text{LOS}}$ is a zero-mean random Gaussian variable with covariance $\mathbf{C}_{\text{LOS}} = \mathbf{I}_3\sigma^2_{\text{LOS}}$, and $\mathbf{I}_3$ denoting the 3×3 identity matrix. The likelihood function is thus given by:

$$\begin{aligned} &p(\mathbf{u}_i|s_i = 1, \mathbf{x}_i, \mathbf{W}, \mathbf{C}_{\text{LOS}}) \\ &\quad = \boldsymbol{\mathcal{N}}(\mathbf{u}_i; \hat{\mathbf{u}}_{\text{LOS},i}(\mathbf{x}_i, \mathbf{W}), \mathbf{C}_{\text{LOS}}). \end{aligned} \tag{10}$$

Similarly, the likelihood function of the regression problem in NLOS conditions can be written as:

$$\begin{aligned} &p(\mathbf{u}_i|s_i = 0, \mathbf{x}_i, \mathbf{W}, \mathbf{C}_{\text{NLOS}}) \\ &\quad = \boldsymbol{\mathcal{N}}(\mathbf{u}_i; \hat{\mathbf{u}}_{\text{NLOS},i}(\mathbf{x}_i, \mathbf{W}), \mathbf{C}_{\text{NLOS}}). \end{aligned} \tag{11}$$

Note that the requirement on the mono-modality of $\mathbf{u}_i$, both in LOS and NLOS, can be easily disregarded by applying for example a *mixture of experts* model [65].

Combining (8), (10) and (11), we define the joint likelihood for the variables $(\mathbf{u}_i, s_i)$ as:

$$\begin{aligned} &p(\mathbf{u}_i, s_i|\mathbf{x}_i, \mathbf{W}, \mathbf{C}_{\text{LOS}}, \mathbf{C}_{\text{NLOS}}) \\ &\quad = \hat{p}_{s,i}\delta[s_i - 1]\boldsymbol{\mathcal{N}}(\mathbf{u}_i; \hat{\mathbf{u}}_{\text{LOS},i}(\mathbf{x}_i, \mathbf{W}), \mathbf{C}_{\text{LOS}}) \\ &\qquad + (1 - \hat{p}_{s,i})\delta[s_i]\boldsymbol{\mathcal{N}}(\mathbf{u}_i; \hat{\mathbf{u}}_{\text{NLOS},i}(\mathbf{x}_i, \mathbf{W}), \mathbf{C}_{\text{NLOS}}). \end{aligned} \tag{12}$$

A representation of the distribution can be found in Fig. 4. In order to maximize the likelihood, we insert the negative log-likelihood in the loss function. It can be shown that the the negative log-likelihood of a batch of independent samples can be written as (see Appendix A):

$$\begin{aligned} \mathrm{L}_{\text{task}} &\backsimeq -\log \prod_{i=1}^{N_\text{b}} p(\mathbf{u}_i, s_i|\mathbf{x}_i, \mathbf{W}, \mathbf{C}_{\text{LOS}}, \mathbf{C}_{\text{NLOS}}) \\ &= \sum_{i=1}^{N_\text{b}} \left\{ s_i \cdot \left[-\log(\hat{p}_{s,i}) + \frac{\|\mathbf{u}_i - \hat{\mathbf{u}}_{\text{LOS},i}\|_2^2}{2\sigma^2_{\text{LOS}}} \right] \right. \\ &\qquad \left. + (1 - s_i) \cdot \left[-\log(1 - \hat{p}_{s,i}) + \frac{\|\mathbf{u}_i - \hat{\mathbf{u}}_{\text{NLOS},i}\|_2^2}{2\sigma^2_{\text{NLOS}}} \right] \right\}, \end{aligned} \tag{13}$$

where N_b is the batch size. Whenever the LOS condition occurs, the second right-hand side of (13) does not contribute to the likelihood, whereas if NLOS holds, only the first right hand side is considered.

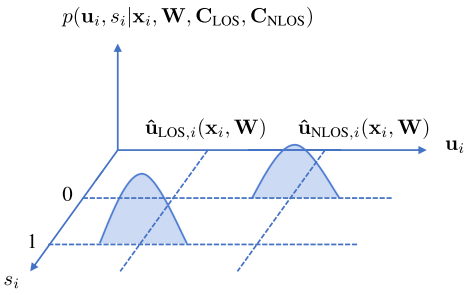


Fig. 4. Joint likelihood distribution of $(\mathbf{u}_i, s_i)$. Here, for sake of simplicity, the multi-dimensional Gaussian distributions are represented as uni-variate.

Finally, the complete adopted loss function is:

$$\begin{aligned}\mathrm{L}_{\text{tot}} = &\frac{w_{\text{rec}}}{N_{\text{b}}}\sum_{i=1}^{N_{\text{b}}}\|\mathbf{x}_i - \hat{\mathbf{x}}_i\|_2^2 \\ &+ \frac{1}{N_{\text{b}}}\sum_{i=1}^{N_{\text{b}}}\Big\{w_s s_i \cdot \left[-\log(\hat{p}_{s,i}) + w_{\text{u}}\|\mathbf{u}_i - \hat{\mathbf{u}}_{\text{LOS},i}\|_2^2\right] \\ &+ (1-s_i)\cdot\left[-\log(1-\hat{p}_{s,i}) + w_{\text{u}}\|\mathbf{u}_i - \hat{\mathbf{u}}_{\text{NLOS},i}\|_2^2\right]\Big\}\end{aligned} \tag{14}$$

where w_{rec} regulates the sample reconstruction, w_s compensates the unbalances between LOS and NLOS samples and w_{u} controls the uncertainty of the model on the position estimation and includes both σ^2_{LOS} and σ^2_{NLOS}.

V. Methodology: Cooperative Localization

In this section, we extend the approach for UE positioning to a multi-BS scenario. We present at first the proposed cooperative AE architecture and then the cooperative training procedure that can be used in practice to deploy the set of BSs composing the localization infrastructure.

A. Cooperative Architecture

We propose a cooperative architecture where each BS adopts a cell-specific DL model. The main assumptions behind the proposed architecture are the following. First, each BS is able to evaluate whether a UE is in LOS or NLOS condition based on the observed ADCPM and the procedure described in Section IV. Second, in case of NLOS, a position estimate is obtained by the BS by means of a previous fingerprint training procedure. Third, the latent geometrical features adopted by the LOS position estimation are combined in order to get a more accurate inference. This is somehow intuitive if we consider the latent features as a sort of non-linear combination of ToF and AoA measurements that each BS can share with the neighbors' cells.

Let us denote by $\mathcal{S}_{\text{BS},i} = \{1, \ldots, S_{\text{BS},i}\}$ the set of BSs which detect a UE at timestep i, and gather a batch of samples $\mathbf{X}_i = \{\mathbf{x}_i^{(j)}\}_{j=1}^{S_{\text{BS},i}}$. Note that the number of detected BSs can vary at every timestep, without being constrained to detect a minimum number of BSs, as opposed to classical geometrical approaches. As illustrated in the pseudo-code in Algorithm 1, the main idea is that, for position inference, each BS j computes its latent features $\mathbf{z}_i^{(j)}$. At this stage, there is no difference between $\mathbf{z}_i^{(j)}$, $\mathbf{z}_{s,i}^{(j)}$, $\mathbf{z}_{\text{LOS},i}^{(j)}$, and $\mathbf{z}_{\text{NLOS},i}^{(j)}$ and the inference proceeds as in the single-BS method. Then, if the BS predicts a NLOS condition, the latent features extracted by that BS are not combined with other cells (e.g., as a multi-lateration) and position estimation continues according to the $\hat{\mathbf{u}}_{\text{NLOS},i}^{(j)}$ prediction. On the contrary, if we are in LOS condition, then the latent LOS features $\mathbf{z}_{s,i}^{(j)}$ are exchanged with the neighbors' cells, defined with $\mathcal{N}_i^{(j)} = \mathcal{S}_{\text{BS},i}\backslash\{j\}$. After averaging the latent LOS features $\mathbf{z}_{s,i}^{(j)}$, the position is estimated with $\hat{\mathbf{u}}_{\text{LOS},i}^{(j)}$.

Algorithm 1 Cooperative Position Inference

Input: sample $\mathbf{x}_i^{(j)}$, neighbors' BS $\mathcal{N}_i^{(j)}$ ▷ Run on BS j at timestep i
Output: estimated position $\hat{\mathbf{u}}_i^{(j)}$

```
Encode sample x_i^(j) through encoder E: z_i^(j) ← E(x_i^(j))
Initialize the latent features
z_{s,i}^(j) ← z_i^(j), z_{LOS,i}^(j) ← z_i^(j), z_{NLOS,i}^(j) ← z_i^(j)
Predict probability of LOS condition p̂_{s,i}^(j) from z_{s,i}^(j)
for j' = 1, 2, ..., N_i^(j) do
    Send {p̂_{s,i}^(j), z_{LOS,i}^(j)} to j'
    Receive {p̂_{s,i}^(j'), z_{LOS,i}^(j')} from j'
end for
Assign ẑ_{PLOS,i}^(j) ← 1/(Σ_{j'=1}^{S_BS,i} p̂_{s,i}^(j')) Σ_{j'=1}^{S_BS,i} p̂_{s,i}^(j') z_{LOS,i}^(j')
Predict LOS position û_{LOS,i}^(j) from ẑ_{PLOS,i}^(j)
Predict NLOS position û_{NLOS,i}^(j) from z_{NLOS,i}^(j)
Estimate position û_i^(j) ← Γ(p̂_{s,i}^(j)) û_{LOS,i}^(j) +
                            Γ(1 − p̂_{s,i}^(j)) û_{NLOS,i}^(j)
```

As an example of cooperative inference, we refer to the scenario shown in Fig. 5 where three BSs detect a UE, here represented by a vehicle. NLOS and LOS links are indicated with red and green dashed arrows, respectively. Since BS j and j'' are in LOS condition with respect to the vehicle, the contribution of $\mathbf{z}_{s,i}^{(j)}$ and $\mathbf{z}_{s,i}^{(j'')}$ to the position estimation will be higher than $\mathbf{z}_{s,i}^{(j')}$. The negligible contribution of NLOS BSs is highlighted with red solid arrows while significant latent features are colored in green. This is due to the fact that the probability of LOS condition of BS j' will be low, i.e., $\hat{p}_{s,i}^{(j')} < \hat{p}_{s,i}^{(j)}$ and $\hat{p}_{s,i}^{(j')} < \hat{p}_{s,i}^{(j'')}$. The outcome is an increased accuracy on the position estimation which is not or little affected by NLOS BSs.

B. Cooperative Training

For the training of the cooperative set of BSs we propose the following procedure. During the data gathering phase, a UE (e.g., a vehicle) moves along specified trajectories (a-priori divided into LOS and NLOS segments) sending uplink signals, i.e., sounding reference signal (SRS), to every BS in its range. The time instant of the transmission, the coarse position obtained from global navigation satellite systems (GNSS) and

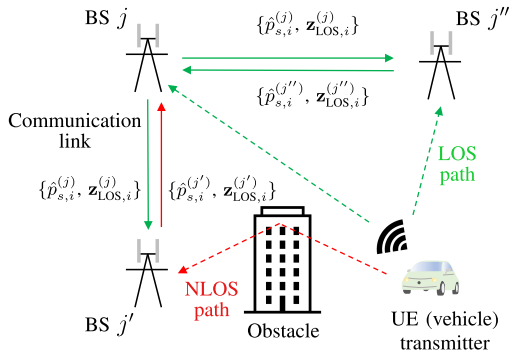


Fig. 5. Cooperative inference scenario composed of three BSs.

the LOS identification are either sent to the BSs as auxiliary data through the communication link, or stored inside the UE for the post-processing. As for the inference, the BSs evaluate the ADCPM samples and store the data. In order to perform back-propagation at BS j, the loss function is computed as:

$$\mathrm{L}_{\text{tot}}^{(j)} = \frac{w_{\text{rec}}}{N_{\text{b}}} \sum_{i=1}^{N_{\text{b}}} \left\| \mathbf{x}_i^{(j)} - \hat{\mathbf{x}}_i^{(j)} \right\|_2^2 + \frac{1}{N_{\text{b}}} \sum_{i=1}^{N_{\text{b}}} \Bigg\{ w_s s_i^{(j)} \left[w_{\text{u}} \left\| \mathbf{u}_i^{(j)} - \hat{\mathbf{u}}_{\text{LOS},i}^{(j)}(\bar{\mathbf{z}}_{\text{LOS},i}^{(j)}) \right\|_2^2 - \log(\hat{p}_{s,i}^{(j)}) \right] + (1 - s_i^{(j)}) \left[w_{\text{u}} \left\| \mathbf{u}_i^{(j)} - \hat{\mathbf{u}}_{\text{NLOS},i}^{(j)}(\mathbf{z}_{\text{NLOS},i}^{(j)}) \right\|_2^2 - \log(1 - \hat{p}_{s,i}^{(j)}) \right] \Bigg\}, \tag{15}$$

where $\bar{\mathbf{z}}_{\text{LOS},i}^{(j)} = \sum_{j'=1}^{S_{\text{BS},i}} s_i^{(j')} \mathbf{z}_{\text{LOS},i}^{(j')} / \sum_{j'=1}^{S_{\text{BS},i}} s_i^{(j')}$. Note that, to speed up the training phase, a centralized loss computation can be performed in a batch-manner, i.e., in parallel way, using $S_{\text{BS},i}$ as batch size. The rational behind this approach is that while the models for NLOS position estimation are trained to be BS-specific, the LOS networks can be shared as they have consistent parameters among all the BSs.

VI. Performance Assessment in a 5G Network

To assess the performance of the proposed cooperative DL positioning system, we employ a data generator based on the 5G new radio (NR) clustered delay line (CDL) channel model [66], which is defined over a bandwidth of 2 GHz in the frequency range from 0.5 GHz to 100 GHz. The radio wave propagation is simulated using a ray-tracing method [67], [68], [69] from the Matlab package, which plots the propagation paths from the UE to the BSs based on the surface geometry from a map file. The ray-tracing method employs the shooting and bouncing rays (SBR) method with up to 10 path reflections [70]. The channel model is then generated by combining all the paths and taking into account the small-scale fading caused by multipath and UE's movement. With this method, adjacent positions will have similar channel characteristics due to the similar scattering environment, ensuring spatial consistency.

TABLE III
Experiment Parameters

Parameter	Value
Carrier frequency f_{c}	30 GHz
Bandwidth B	400 MHz
BS noise figure	7 dB
BS antenna configuration	$M = N = 8$, $d^{(h)} = d^{(v)} = 0.5\lambda_{\text{c}}$
BS antenna radiation pattern	Directional, 8 dBi – (Table 6.1.1-5 of [57])
UE noise figure	13 dB
UE antenna configuration	Omni
UE radiation pattern	0 dBi
UE max. TX power	23 dBm
Layout	Hexagonal grid, 19 macro sites, 3 sectors per site, ISD = 200 m
BS clock offset	0 ns (perfect synchronization)
Penetration loss	0 dB
UE mobility	Max 30 km/h
UE height	1.5 m
Min. BS-UE distance (2D)	10 m
BS antenna height	25 m

A. Simulated Scenario

For the experiments, we simulate a 3GPP urban micro (UMi) scenario in a 1000 × 1000 m area, near the Leonardo's campus of Politecnico di Milano, in Milano, Italy, using the parameters described in [57]. The specific values are summarized in Table III. As shown in Fig. 6, the scenario consists of 19 sites with an inter-site distance (ISD) of 200 m, placed in an hexagonal layout, each equipped with 3 cells and separated by 120 deg in azimuth. Each cell antenna uses an UPA configuration with $N = M = 8$ elements and a downtilt of 15 deg. A macro image of the antenna array pattern can be found in Fig. 7. Each antenna element was defined using specifications in [71], providing a front-to-back ratio of about 30 dB and a maximum gain of 8 dBi.

The UEs move around the region following various routes generated by the SUMO simulator that reproduces the vehicular traffic flow over the considered road network. The simulation runs for 170 seconds and generates up to 50 trajectories, which are sampled every second. Each UE uses a carrier frequency of $f_{\text{c}} = 30$ GHz and a transmission bandwidth of $B = 400$ MHz to transmit 5G standard compliant SRSs to all nearby BSs. Finally, each BS demodulates the signals and obtains a channel estimation, i.e., SFCRM, via the received pilot signals through a LS estimator. The ADCPM is then obtained according to (4) and (5).

B. Positioning Tests

We divided the experiments into offline and online phases. In the offline (training) phase, the UE moves along the trajectories and each BS gathers both LOS and NLOS channel realizations according to the specific UE's position. We considered the training positions as perfectly known (i.e., no error was introduced to the ground truth positions) as we aimed at assessing the lower bound performances of the method. In total, we gathered about $5.9 \cdot 10^4$ samples in 1659 positions only for the training phase. In the online (test) phase, the NLOS position capabilities were verified in the same trajectories but in random positions not adopted in the

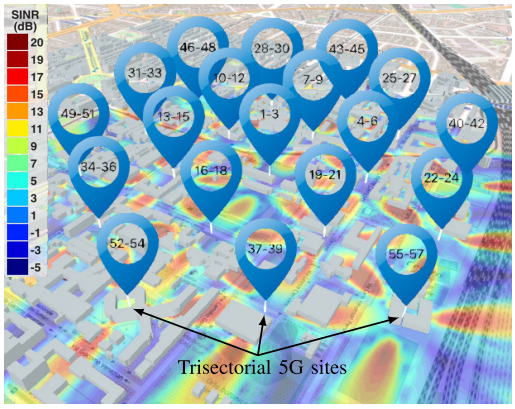


Fig. 6. UMi scenario simulated for performing analysis composed of 19 sites in the area of Politecnico di Milano, Leonardo campus, Milano, Italy.

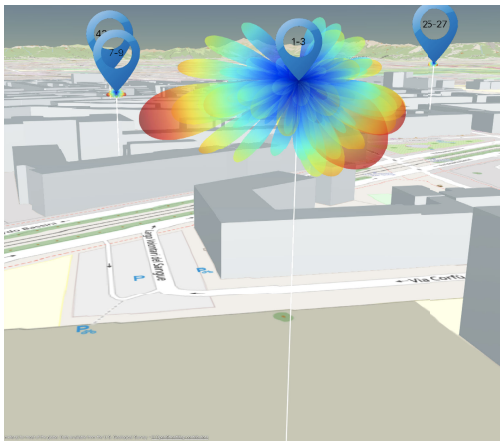


Fig. 7. Antenna array patterns of 3 cells (1-3) forming a site.

training. Specifically, we adopted about $2.5 \cdot 10^4$ samples in 711 positions for NLOS testing. In this way, we assess whether or not, each BS can learn the environment, i.e., the channel characteristics, around it. On the other hand, for LOS positioning, we validated completed different trajectories to analyze the capabilities of estimating the position from learned geometrical features. Here, the total number of LOS tested positions was 867. A representation of the adopted trajectories can be found in Fig. 8. Note that in the training trajectories, i.e., red markers, the number of detected BSs can vary significantly: we measured from 1 up to 13 BSs in the collected samples. The test samples for LOS evaluation, highlighted with purple placeholders, are located in the top-left corner of the map. To avoid biases and improve model convergence, before model training, we standardized with zero mean and unitary standard deviation all the samples, and we shuffled the dataset at each epoch.

The toolbox *Antenna Toolbox* of MATLAB 2022b is used to generate the channel fingerprints for the data points, while the model for training and testing is implemented using Pytorch [72] (v1.12 with Python 3.7.11). The simulations are run on a workstation with an Intel(R) Xeon(R) Silver 4210R CPU @ 2.40 GHz, 96 GB of RAM, and a Quadro RTX 6000 24 GB GPU. The training and testing times refer only to the runtime on Pytorch 1.12. Unless otherwise noted, the model is trained for a total $E = 60$ epochs with a batch size $N_\text{b} = 64$. The Adam optimization algorithm [73] is used with an initial learning rate $lr = 10^{-4}$ and momentum values $\beta_1 = 0.9$, $\beta_2 = 0.999$.

For what concerns the hyper-parameters choice of w_rec, w_s and w_u, we clarify that their values depend on the specific dataset at hand and thereby tuning has been performed as follows. Starting from w_s, since this parameters regulates how much weight is given to the LOS samples class in order to compensate the class unbalances, it is computed as in weighted cross-entropy loss: $w_s = N_\text{LOS}/N_\text{NLOS}$, where N_LOS and N_NLOS are the number of LOS and NLOS samples in the training batch, respectively. On the contrary, w_rec and w_u have been chosen empirically using a grid-approach in $[0.1, 1]$ with step size 0.1 and assigned as: $w_\text{rec} = 0.1$, $w_\text{u} = 0.9$. This can be intuitively explained by two reasons. First, the AE model is much more complex than the MLPs for positioning, leading to a fast drop of the reconstruction error. Second, the number of features in $\mathbf{x}_i$ is larger than the dimension of $\mathbf{u}_i$, which automatically increases the weight of the reconstruction error with respect to the positioning error. In order to balance these two quantities, we suggest values that satisfy $w_\text{rec} < w_\text{u}$.

C. DL Model Design

The adopted DL model architecture is as follows. The AE is the most critical component as a good feature extraction is essential to enable precise positioning. Driven by the necessity of handling sparse data, i.e., ADCPM input, we select the Segnet architecture [74] where the upsampling layers employ the encoder pool indices to create ad-hoc sparse feature mapping. At testing time, we can completely discard the decoder part as the input reconstruction is only adopted in the training phase to learn the latent features representation.

The choice of the NNs for NLOS classification and position estimation is dictated by the specific task to accomplish. A BS must be able to localize a UE regardless of whether it is in ego mode, i.e., only a single BS detects the UE, or in cooperative mode, i.e., the UE is detected by multiple BSs. In the former case, the latent features should be a non-linear composition of ToF and AoA for each of the multipaths. On the other hand, in the latter case, multiple ToF must be exploited to localize the UE. To assess these concepts, we experiment different multi-layer perceptrons (MLPs) architectures trying to localize a UE with only a ToF and AoA or 3 or more ToF as inputs.

In Fig. 9, we show the test results of the MLP described in Table IV which was trained on a synthetic dataset using as input either ToF and AoA only (Fig. 9a), or 3 measurements of ToFs (Fig. 9b) or 10 ToFs (Fig. 9c). This was done in order to verify the positioning capabilities of the models, i.e.,

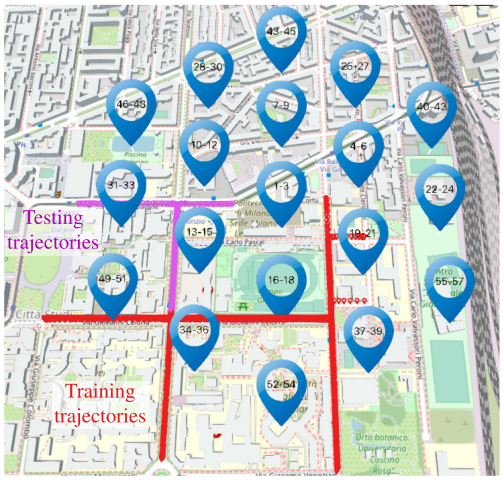


Fig. 8. Scenario comprising training trajectories, i.e., red placeholders, and LOS testing trajectories, i.e., purple placeholders.

to assess the bias of the MLPs for LOS and NLOS position estimation. Since the testing positions (red circles) are never seen in the training phase, we can conclude that the model is able to learn the geometrical meaning of the input features and perform multi-angulation and circular lateration. It is worth noticing that, in the case of cooperative architecture, the BSs' positions (black squares) are never used as input features but they are automatically learned by the training. This can be an advantage in case the coordinates of the BSs are not known or partially known.

Given these results, we employed the MLP in Table IV for the LOS and NLOS position prediction NNs, while the MLP in Table V is adopted for NLOS detection NN. The key difference between these two types of NNs is the size and layer composition. First, we assumed that single supervised NLOS classification is a simpler task if compared with 3D position regression, thus resulting in a different number of layers and neurons. Second, we employed Tanh (instead of ReLu or GeLu) activation functions in the NLOS detection NN since blockage detection should be performed in the fastest and most efficient way possible, as the whole cooperative prediction highly depends on it. On the contrary, GeLu is more computationally expensive but can capture more complex patterns (as needed for regression) due to its smooth and differentiable nature [75]. Finally, both adopt dropout to avoid overfitting.

D. Baselines

For performance assessment, we compare the proposed method with the following algorithms/models:

- Ego DL model: the proposed DL model (Fig. 3) trained with the loss (14) for single-BS positioning. The model does not communicate with any other BS and has to rely only on its prediction.
- Cooperative DL model: the proposed DL-based cooperative architecture described in Sec. V.
- Single-BS ToF-AoA: conventional positioning obtained by a single BS using the ToF estimate obtained from the cross-correlation with the SRS according to 5G NR Rel. 16 and the AoA estimated through multiple signal classification (MUSIC) algorithm [76].
- Multi-BS time difference of flight (TDoF): conventional hyperbolic-multilateration obtained using TDoFs estimated by all the BSs receiving SRS.

E. Simulation Results

1) Training Convergence: This first assessment has the aim of verifying the training convergence of the proposed ego DL model, i.e., the correct behavior of the loss function and of the root mean square error (RMSE) performance metric. In Fig. 10, we report the testing results separately for LOS and NLOS samples in the test set, for varying number of training epochs. We notice that the loss function decreases quite rapidly in just 60 epochs and does not show signs of overfitting. This may be due to the fact that the dataset is very large (about 50 GB) and thus it is difficult to memorize every sample. Furthermore, the model regularization, i.e., dropout, helps with this aspect.

Referring to the performance metric, we can see that the NLOS case holds much higher oscillations with respect to the LOS one. While learning geometrical patterns and associated positions is a more standard task, associating NLOS samples with the UE location is much more difficult. The DL model has not only to understand how the environment is configured but also how it could change between the training positions. Finally, we note that the LOS RMSE is slightly worse that the NLOS. This is due to the LOS performance bounds imposed by the physical layer parameters, i.e., the resolution of the ToF (limited by the signal bandwidth) and AoA (by the number of antennas). On the other hand, the NLOS RMSE highly depends on the training position resolution, i.e., the training spatial sampling, as the denser the training points, the better the performances.

2) Blockage Detection: This experiment has the objective of assessing the steering model capabilities in discerning LOS and NLOS conditions. The steering is taken into account in both the smart weighting of the geometrical latent features and in the final position estimate, i.e., lines 8 and 11 of Algorithm 1, respectively. To this aim, in Fig. 11 we show the testing accuracy, precision and recall for varying number of training epochs. The numerical results show that the model reaches an accuracy of approximately 85%, validating the capability of the proposed approach of learning the multi-task problem within such a realistic environment. In terms of the parameter Γ, we implemented a conservative yet effective approach, as detailed subsequently. We selected $\Gamma = 0.6$ not with the aim of matching the values of precision and recall, but rather to adjust it towards a lower number of false positives compared to false negatives. This approach is driven by the rationale that if the model is uncertain about the visibility condition, it slightly leans towards the NLOS case. As a result,

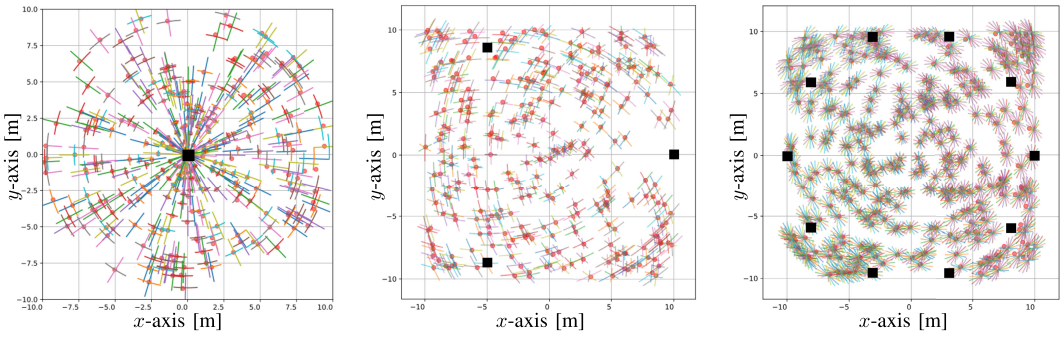


Fig. 9. Testing positioning results of an MLP with input (a) AoA and ToF, (b) 3 ToFs (c) 10 ToFs. The BSs are located in the black squares.

TABLE IV
LOS AND NLOS POSITIONING NETWORK

Layer Num.	Type	Output Size
0	Input	8×1
1	Linear + GeLu + Dropout	8×1
2	Linear + GeLu + Dropout	16×1
3	Linear + GeLu + Dropout	32×1
4	Linear + GeLu + Dropout	64×1
5	Linear + GeLu + Dropout	128×1
6	Linear + GeLu + Dropout	256×1
7	Linear + GeLu + Dropout	128×1
8	Linear + GeLu + Dropout	64×1
9	Linear + GeLu + Dropout	32×1
10	Linear + GeLu + Dropout	16×1
11	Linear	3×1

TABLE V
NLOS CLASSIFICATION NETWORK

Layer Num.	Type	Output Size
0	Input	8×1
1	Linear + Tanh + Dropout	8×1
2	Linear + Tanh + Dropout	5×1
3	Linear	1×1
4	BatchNorm1d	1×1
5	Sigmoid	1×1

substantial geometric errors are circumvented, and only the fingerprint technique is employed.

3) Online Computational Complexity: In this section, we analyze the computational complexity of the proposed localization method by assessing the number of floating point operations per second (FLOPS) and the inference time required for performing positioning. Starting from the computational complexity, we can divide the computational load into measurement extraction and position estimation. For measurement extraction, assuming to have N_{g} signal samples and MN antenna elements, the ADCPM can be efficiently computed adopting a 2D-inverse fast Fourier transform (IFFT), with a complexity of $O(MNN_{\text{g}} \cdot (\log(N_{\text{g}}MN)))$. On the other hand, time-based measurements obtained with cross-correlation, hold a complexity in the order of $O(N_{\text{g}}^2)$ (or $O(N_{\text{g}} \cdot (\log(N_{\text{g}})))$ with efficient methods as FFT). Finally, angle-based measurements obtained by the MUSIC algorithm using N and M scanning directions in the azimuth and zenith domain, respectively, involve an overall complexity of $O((MN)^3)$ [77], considering the eigenvalue decomposition (EVD) on the signal covariance matrix as predominant on the scanning search for source directions. In order to have a reference measure of time required by our system (described in Sec. VI-B) to compute the measurements, we clarify that on average, the ADCPM, ToF and AoA computations took about 0.4, 0.3 and 0.4 ms, respectively. As expected, the complexity of the ADCPM results higher than single time-based or angle-based measurements, as it embodies all the location-related information provided by the propagation channel, i.e., including the ToFs, AoAs and received power of all multipaths.

For what concerns the algorithms for position estimation, we compare non-linear least square (NLS) methods, adopted in Single-BS ToF-AoA and Multi-BS TDoF, and the proposed DL model. Denoting with N_u the unknown location coordinates to be estimated, N_{meas} the number of measurements and N_{iter} the number iterations for NLS convergence, for Gauss-Newton method we hold an overall complexity of $O(N_{\text{iter}}(N_{\text{meas}} \cdot N_{\text{u}} + N_{\text{meas}} \cdot N_{\text{u}}^2 + N_{\text{u}}^3))$ (assuming that the cost of computing the Jacobian is roughly proportional to $O(N_{\text{meas}} \cdot N_{\text{u}})$, the cost of Jacobian matrix multiplication to $O(N_{\text{meas}} \cdot N_{\text{u}}^2)$, and the matrix inversion to $O(N_{\text{u}}^3)$). However, estimating the complexity of the DL models is not straightforward as it would involve a detailed breakdown of the operations performed. To this aim, in Fig. 12, we empirically assess the inference times using the system hardware described in Sec. VI-B and we compare it with NLS solutions, for varying number of measurements. For NLS method, we empirically choose $N_{\text{iter}} = 50$ and $N_{\text{u}} = 3$ (3D position). Results show that the proposed DL model is able to perform the position inference in about 1.5-2 ms which corresponds to a NLS with about 10 measurements. We point out that these results highly depend on the specific hardware and implementation of both the DL and NLS algorithms. Nevertheless, the overall conclusion is that, despite being slightly more complex, the proposed method has the main advantages of having greater accuracy with respect to classical geometrical algorithms and retaining the ability to localize in NLOS conditions (as described in Sec. VI-E.4), while being at the same time compliant with the strict requirements given

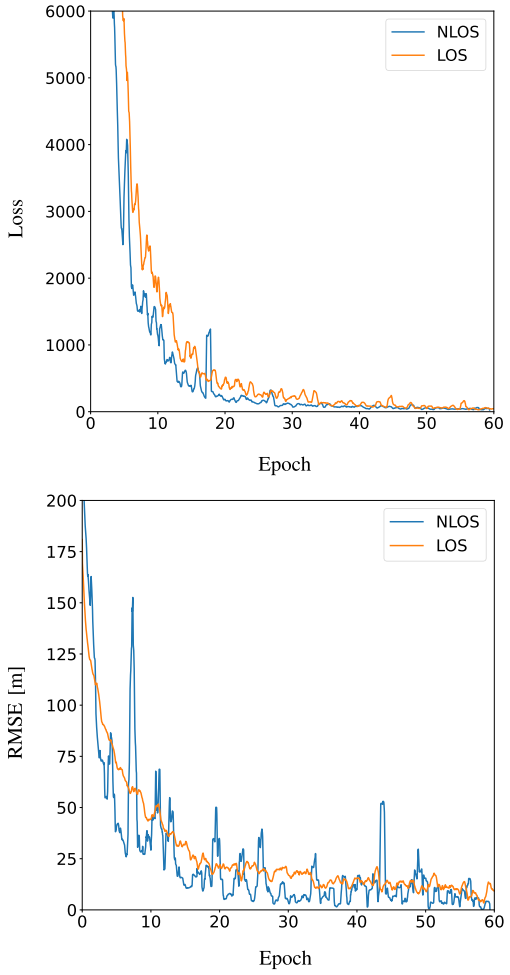


Fig. 10. Testing results, i.e., loss on top and RMSE on bottom, of LOS and NLOS samples varying the number of epochs in the training.

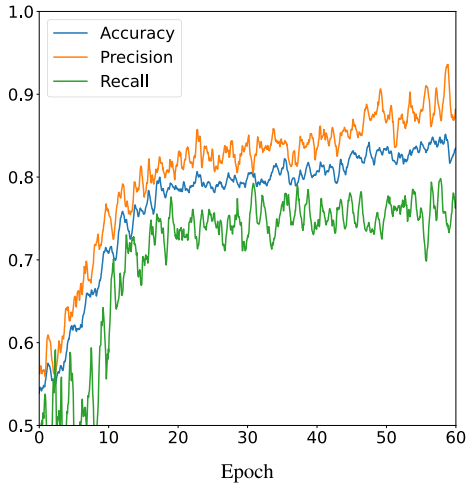


Fig. 11. Testing results of classification accuracy, precision and recall varying the number of epochs in the training.

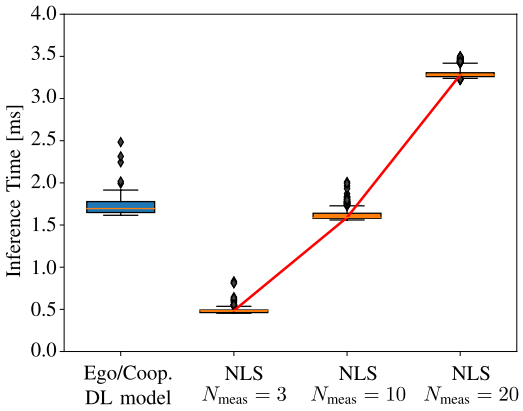


Fig. 12. Boxplot of the distribution of the inference time per sample [ms].

by cooperative, connected, and automated mobility (CCAM) of 5 ms [78].

4) Positioning Accuracy: In this assessment we test the positioning accuracy of the methods described in Sec. VI-D. To this aim, in Fig. 13 and 14, we show the cumulative density functions (CDFs) of the location error in the LOS and complete test trajectory, respectively. For the cooperative methods, we consider a position LOS if all the BSs detecting the UE are in LOS. To better compare the results, in both the figures, we report both the 2D and 3D error for each method.

Starting from the position methods with single BS in the LOS positions, we notice a huge improvement between the Single-BS ToF-AoA and the ego DL model, passing from a mean error of 26.38 m to 5.99 m in 3 dimensions. With the 2D error metric, the performances are slightly better, obtaining 24.76 m and 5.35 m, respectively. The ego DL model automatically extracts a compact non-linear representation of the overall multipath profile (i.e., ToF, AoA and power of all paths in LOS conditions), which is much more informative than the direct path information, thus outperforming the traditional single-BS based positioning. Moreover, in 3D positioning, it outperforms even the Multi-BS TDoF. This is due to the fact that, in classical TDoF approaches, the vertical geometrical dilution of precision (GDoP) is very limited due to the poor geometrical arrangement of the BSs over the vertical dimension, as BSs are usually located at similar heights. On the contrary, the ML approach is able to learn the usual altitude of the user and exploit this information a-priori in the position's computation.

Moving to the cooperative methods in the LOS testing trajectory, we observe that the proposed cooperative DL model

outperforms the Multi-BS TDoF even in the 2D positioning case, i.e., holding 66% of the points with an error of less than 0.81 m and 90% with an error of less than 1.3 m. The mean error decreases from 3.21 m with Multi-BS TDoF, to 1.68 m with the cooperative DL solution, with an improvement of above 47%. The cooperative DL model, in fact, holds a common-sense of where the UE could be, i.e., discarding or not considering possible unfeasible solutions that could be obtained by geometrical algorithms.

This is confirmed by the results on the complete testing trajectory in Fig. 14, composed by both LOS and NLOS positions. Whenever one or more BSs are in NLOS, we suffer a severe degradation of performances. In these conditions, the ego DL model reaches a median 2D error of 3.01 m with respect to an error of 3.74 m in case of Multi-BS TDoF. Comparing the cooperative and ego DL models, we notice an approaching of the two CDFs, mainly due to the higher NLOS performances in case of ego-positioning. This is because, even when LOS positions are inaccurately classified, the positioning is corrected through fingerprint training without being impacted by errors in geometrical features. Essentially, the position estimation solely relies on either the LOS MLP or the NLOS MLP, but never a combination of the two. On the other hand, the cooperative DL model suffers slightly higher errors since in NLOS positions the BSs cannot cooperate.

In case the achieved performances do not satisfy the target accuracy for the specific location-sensitive service, several strategies can be implemented in order to improve the proposed method without changing the structure of the model. First, starting from the physical layer, increasing the bandwidth and number of antennas at the BSs would permit to enhance the space-time system resolution, and thereby the ability of the model to resolve the multipath and extract location information from the ADCPM. Second, from a design point of view, we can increase the DL model dimension (especially the AE part), which reduces the model bias, and simultaneously increase the number of collected data, thus reducing the variance. However, we need to keep in mind that this comes at the price of higher training and inference times, as well as higher costs of dataset creation, resulting in a performance-complexity trade-off.

5) Tracking Performance: This experiment compares the performances of the positioning methods in the testing trajectory where a UE, i.e., a CAV, moves along a road at variable speed. In Fig. 15, we can see in pink the covered ground-truth trajectory both in 3D (a) and in 2D (b). The CAV moves faster in the north part of the trajectory and then slows down in the southern part of the road. Since the objective is to assess the point-position estimation of each method, we do not implement any tracking filter and we rely exclusively on the channel realization at specific samples of the trajectory. Moreover, passing from the 3D to the 2D representation, we discard unlikely estimated positions just for easy-visualization purposes of the 2D trajectory. The Single-BS ToF-AoA and the ego DL model results are obtained from the BS number 48, while the cooperative methods consider only the positions where the CAV detect the BS number 48.

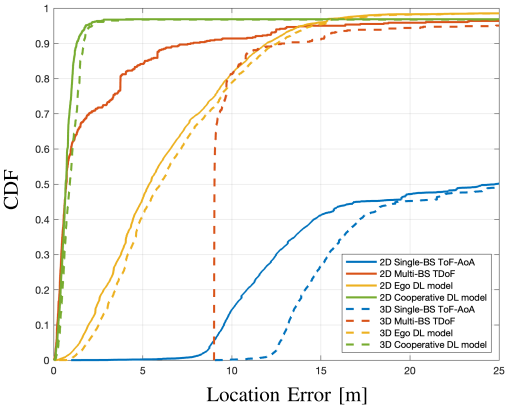


Fig. 13. Positioning performances in terms of CDF of the distance error in the LOS testing trajectory. The solid and dotted lines are the 2D and 3D errors, respectively.

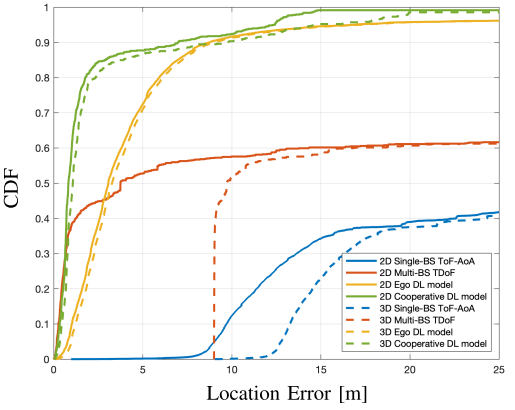
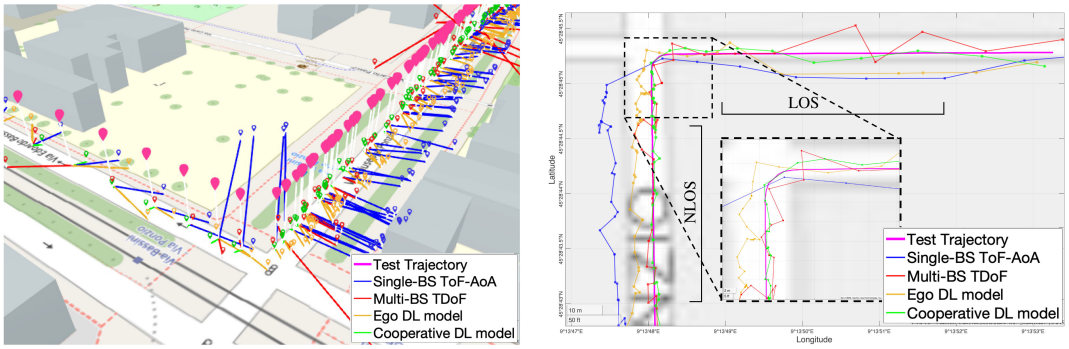


Fig. 14. Positioning performances in terms of CDF of the location error over the whole testing trajectory. The solid and dotted lines are the 2D and 3D errors, respectively.

Observing Fig. 15a, we notice that the Single-BS ToF-AoA method has the worst performances since it locates the CAV outside the road for most of the trajectory. This is mainly due to AoA estimations, which become worst the higher the distance from the BS, and ToF estimations. In LOS positions, i.e., north part of the trajectory, the ToF error is only due to the autocorrelation of SRSs and peak estimation. On the contrary, in NLOS positions, i.e., south part of the trajectory, the major source of error is represented by reflections. The ego DL model improves this aspect by halving the error (see Fig. 15b) especially in NLOS sections. The Multi-BS TDoF struggles in high-speed conditions, i.e., east part of LOS trajectory, and in NLOS positions where the number of cooperative BSs is limited or the range-biases are severe, i.e., top-right corner of Fig. 15a. Finally, the proposed cooperative DL model (green markers) achieves the higher positioning accuracy in almost all conditions, combining



(a) (b)

Fig. 15. Positioning performance in a 5G urban scenario: (a) 3D testing trajectory and related estimate obtained by the positioning methods (represented by different colors), with location errors represented as solid lines. (b) Bird-eye view of the testing and estimated trajectories.

cooperative LOS measurements with egocentric NLOS predictions.

VII. Conclusion and Future Works

Given the paramount importance of providing enhanced solutions for high-precision positioning in next 3GPP Releases, in this paper, we propose a cooperative positioning network architecture based on DL. Each BS composing the localization infrastructure has the same proposed DL model which solves the joint task of NLOS identification and position estimation. Depending on the condition, LOS or NLOS, the task is solved in cooperative or egocentric (ego) mode, respectively. In order to cooperate, the architecture internally exchanges only the compact latent feature representation of the channel obtained with an AE structure, permitting to combine the complete non-linear measurements and enhance positioning accuracy.

The proposed cooperative architecture is suitable and fully-compliant with 5G massive-MIMO OFDM systems, where sparse space-time channel responses, i.e., ADCPM, are adopted as input-images to the DL model. The ADCPM embodies position-dependent features, such as ToF, AoA and RSS of each propagation path, which can be automatically extracted by the proposed DL model. With the use of Matlab ray-tracing and SUMO software, we simulate a complex and realistic C-ITS scenario where some CAVs create multiple trajectories and communicate with a set of BSs, i.e., 3GPP UMi scenario. Results show that the proposed cooperative architecture is able to improve upon classical geometrical algorithms, e.g., TDoF multi-lateration, both in LOS and NLOS conditions, by increasing the accuracy of 47%. Moreover, the cooperation overcomes the limitations of single-BS prediction based on DL by automatically switching between egocentric and cooperative mode.

ML, and more especially DL techniques, are foreseen to have a huge impact on next-generation cellular networks. This work is thereby a first attempt to implement a cooperative high-precision positioning system towards that direction. Further developments could be the integration of different DL models into the architecture or the tracking of many simultaneous targets with automatic data-association.

Appendix A
Joint Log-Likelihood

To prove (13), we start by rewriting the likelihood distribution (12) as:

$$p(\mathbf{u}_i, s_i|\mathbf{x}_i, \mathbf{W}, \mathbf{C}_{\text{LOS}}, \mathbf{C}_{\text{NLOS}}) = [\hat{p}_{s,i}\boldsymbol{\mathcal{N}}(\mathbf{u}_i; \hat{\mathbf{u}}_{\text{LOS},i}(\mathbf{x}_i, \mathbf{W}), \mathbf{C}_{\text{LOS}})]^{s_i} \times [(1-\hat{p}_{s,i})\boldsymbol{\mathcal{N}}(\mathbf{u}_i; \hat{\mathbf{u}}_{\text{NLOS},i}(\mathbf{x}_i, \mathbf{W}), \mathbf{C}_{\text{NLOS}})]^{(1-s_i)}. \quad (16)$$

For simplicity of notation, we drop the dependencies on $\mathbf{x}_i$ and $\mathbf{W}$. The negative log-likelihood of the overall batch of samples, using (16) is:

$$\begin{aligned} \mathrm{L}_{\text{task}} &= -\sum_{i=1}^{N_b} \log\Big\{[\hat{p}_{s,i}\boldsymbol{\mathcal{N}}(\mathbf{u}_i; \hat{\mathbf{u}}_{\text{LOS},i}, \mathbf{C}_{\text{LOS}})]^{s_i} \\ &\quad \times [(1-\hat{p}_{s,i})\boldsymbol{\mathcal{N}}(\mathbf{u}_i; \hat{\mathbf{u}}_{\text{NLOS},i}, \mathbf{C}_{\text{NLOS}})]^{(1-s_i)}\Big\} \quad (17) \\ &= \sum_{i=1}^{N_b}\Big\{s_i\big[-\log(\hat{p}_{s,i}) - \log(\boldsymbol{\mathcal{N}}(\mathbf{u}_i; \hat{\mathbf{u}}_{\text{LOS},i}, \mathbf{C}_{\text{LOS}}))\big] \\ &\quad + (1-s_i)\big[-\log(1-\hat{p}_{s,i}) - \log(\boldsymbol{\mathcal{N}}(\mathbf{u}_i; \hat{\mathbf{u}}_{\text{NLOS},i}, \mathbf{C}_{\text{NLOS}}))\big]\Big\}. \quad (18) \end{aligned}$$

Now we can explicitly compute the logarithm of the multi-variate Gaussian distribution and obtain:

$$\begin{aligned} \mathrm{L}_{\text{task}} = \sum_{i=1}^{N_b}\Big\{ & s_i \cdot \Big[-\log(\hat{p}_{s,i}) + \frac{\log(2\pi|\mathbf{C}_{\text{LOS}}|)}{2} \\ & + \frac{\|\mathbf{u}_i - \hat{\mathbf{u}}_{\text{LOS},i}\|_2^2}{2\sigma_{\text{LOS}}^2}\Big] \\ & + (1-s_i)\cdot\Big[-\log(1-\hat{p}_{s,i}) + \frac{\log(2\pi|\mathbf{C}_{\text{NLOS}}|)}{2} \\ & + \frac{\|\mathbf{u}_i - \hat{\mathbf{u}}_{\text{NLOS},i}\|_2^2}{2\sigma_{\text{NLOS}}^2}\Big]\Big\}. \quad (19) \end{aligned}$$

By discarding the terms that do not depend on $\mathbf{u}_i$ or s_i, we obtain:

$$\mathrm{L}_{\text{task}} \backsimeq \sum_{i=1}^{N_b} \Bigg\{ s_i \cdot \left[-\log(\hat{p}_{s,i}) + \frac{\|\mathbf{u}_i - \hat{\mathbf{u}}_{\text{LOS},i}\|_2^2}{2\sigma_{\text{LOS}}^2} \right] + (1 - s_i) \cdot \left[-\log(1 - \hat{p}_{s,i}) + \frac{\|\mathbf{u}_i - \hat{\mathbf{u}}_{\text{NLOS},i}\|_2^2}{2\sigma_{\text{NLOS}}^2} \right] \Bigg\}, \tag{20}$$

concluding the derivation.

REFERENCES

[1] J. A. del Peral-Rosado, R. Raulefs, J. A. López-Salcedo, and G. Seco-Granados, "Survey of cellular mobile radio localization methods: From 1G to 5G," *IEEE Commun. Surveys Tuts.*, vol. 20, no. 2, pp. 1124–1148, 2nd Quart., 2018. [Online]. Available: https://ieeexplore.ieee.org/document/8226757/

[2] O. Kanhere and T. S. Rappaport, "Position location for futuristic cellular communications: 5G and beyond," *IEEE Commun. Mag.*, vol. 59, no. 1, pp. 70–75, Jan. 2021. [Online]. Available: https://ieeexplore.ieee.org/document/9356512/

[3] A. Nessa, B. Adhikari, F. Hussain, and X. N. Fernando, "A survey of machine learning for indoor positioning," *IEEE Access*, vol. 8, pp. 214945–214965, 2020. [Online]. Available: https://ieeexplore.ieee.org/document/9264122/

[4] R. Di Taranto, S. Muppirisetty, R. Raulefs, D. Slock, T. Svensson, and H. Wymeersch, "Location-aware communications for 5G networks: How location information can improve scalability, latency, and robustness of 5G," *IEEE Signal Process. Mag.*, vol. 31, no. 6, pp. 102–112, Nov. 2014. [Online]. Available: https://ieeexplore.ieee.org/document/6924849

[5] A. M. Nor, S. Halunga, and O. Fratu, "Survey on positioning information assisted mmWave beamforming training," *Ad Hoc Netw.*, vol. 135, Oct. 2022, Art. no. 102947. [Online]. Available: https://linkinghub.elsevier.com/retrieve/pii/S1570870522001287

[6] J. Zhang, E. Björnson, M. Matthaiou, D. W. K. Ng, H. Yang, and D. J. Love, "Prospective multiple antenna technologies for beyond 5G," *IEEE J. Sel. Areas Commun.*, vol. 38, no. 8, pp. 1637–1660, Aug. 2020. [Online]. Available: https://ieeexplore.ieee.org/document/9113273/

[7] L. Yin, Q. Ni, and Z. Deng, "A GNSS/5G integrated positioning methodology in D2D communication networks," *IEEE J. Sel. Areas Commun.*, vol. 36, no. 2, pp. 351–362, Feb. 2018.

[8] F. Mogyorósi et al., "Positioning in 5G and 6G networks—A survey," *Sensors*, vol. 22, no. 13, p. 4757, Jun. 2022. [Online]. Available: https://www.mdpi.com/1424-8220/22/13/4757

[9] A. M. Nor and E. M. Mohamed, "Li-Fi positioning for efficient millimeter wave beamforming training in indoor environment," *Mobile Netw. Appl.*, vol. 24, no. 2, pp. 517–531, Apr. 2019, doi: 10.1007/s11036-018-1154-4.

[10] C. Morelli, M. Nicoli, V. Rampa, and U. Spagnolini, "Hidden Markov models for radio localization in mixed LOS/NLOS conditions," *IEEE Trans. Signal Process.*, vol. 55, no. 4, pp. 1525–1542, Apr. 2007. [Online]. Available: http://ieeexplore.ieee.org/document/4133045/

[11] L. Barbieri, M. Brambilla, A. Trabattoni, S. Mervic, and M. Nicoli, "UWB localization in a smart factory: Augmentation methods and experimental assessment," *IEEE Trans. Instrum. Meas.*, vol. 70, pp. 1–18, 2021. [Online]. Available: https://ieeexplore.ieee.org/document/9409138/

[12] F. Liu et al., "Integrated sensing and communications: Toward dual-functional wireless networks for 6G and beyond," *IEEE J. Sel. Areas Commun.*, vol. 40, no. 6, pp. 1728–1767, Jun. 2022. [Online]. Available: https://ieeexplore.ieee.org/document/9737357/

[13] Q. Zhang, H. Sun, X. Gao, X. Wang, and Z. Feng, "Time-division ISAC enabled connected automated vehicles cooperation algorithm design and performance evaluation," *IEEE J. Sel. Areas Commun.*, vol. 40, no. 7, pp. 2206–2218, Jul. 2022. [Online]. Available: https://ieeexplore.ieee.org/document/9728752/

[14] S. Shi, Z. Cheng, L. Wu, Z. He, and B. Shankar, "Distributed 5G NR-based integrated sensing and communication systems: Frame structure and performance analysis," in *Proc. 30th Eur. Signal Process. Conf. (EUSIPCO)*, Belgrade, Serbia, Aug. 2022, pp. 1062–1066. [Online]. Available: https://ieeexplore.ieee.org/document/9909950/

[15] Z. Yang, C. Wu, and Y. Liu, "Locating in fingerprint space: Wireless indoor localization with little human intervention," in *Proc. 18th Annu. Int. Conf. Mobile Comput. Netw.*, Istanbul Turkey, Aug. 2012, pp. 269–280, doi: 10.1145/2348543.2348578.

[16] M. M. Butt, A. Rao, and D. Yoon, "RF fingerprinting and deep learning assisted UE positioning in 5G," in *Proc. IEEE 91st Veh. Technol. Conf. (VTC-Spring)*, May 2020, pp. 1–7.

[17] J. Sangthong, J. Thongkam, and S. Promwong, "Indoor wireless sensor network localization using RSSI based weighting algorithm method," in *Proc. 6th Int. Conf. Eng., Appl. Sci. Technol.*, Chiang Mai, Thailand, Jul. 2020, pp. 1–4. [Online]. Available: https://ieeexplore.ieee.org/document/9165300/

[18] *Moderator's Summary for Discussion [Ran93E-R18Prep-10] Expanded and Improved Positioning*, document TSG-RAN meeting (RP) 0.0.4, Version 0.0.4, 3GPP, Sep. 2021. [Online]. Available: https://www.3gpp.org/ftp/tsg_ran/TSG_RAN/TSGR_93e/Docs/

[19] A. Bourdoux et al., "6G white paper on localization and sensing," 2020, *arXiv:2006.01779*.

[20] T. Wild, V. Braun, and H. Viswanathan, "Joint design of communication and sensing for beyond 5G and 6G systems," *IEEE Access*, vol. 9, pp. 30845–30857, 2021. [Online]. Available: https://ieeexplore.ieee.org/document/9354629/

[21] S. Rezaie, E. de Carvalho, and C. N. Manchón, "A deep learning approach to location- and orientation-aided 3D beam selection for mmWave communications," *IEEE Trans. Wireless Commun.*, vol. 21, no. 12, pp. 11110–11124, Dec. 2022. [Online]. Available: https://ieeexplore.ieee.org/document/9832551/

[22] M. A. Arfaoui et al., "Invoking deep learning for joint estimation of indoor LiFi user position and orientation," *IEEE J. Sel. Areas Commun.*, vol. 39, no. 9, pp. 2890–2905, Sep. 2021.

[23] S. V. Balkus, H. Wang, B. D. Cornet, C. Mahabal, H. Ngo, and H. Fang, "A survey of collaborative machine learning using 5G vehicular communications," *IEEE Commun. Surveys Tuts.*, vol. 24, no. 2, pp. 1280–1303, 2nd Quart., 2022. [Online]. Available: https://ieeexplore.ieee.org/document/9706268/

[24] L. Barbieri, B. C. Tedeschini, M. Brambilla, and M. Nicoli, "Implicit vehicle positioning with cooperative LiDAR sensing," in *Proc. IEEE Int. Conf. Acoust., Speech Signal Process. (ICASSP)*, Jun. 2023, pp. 1–5.

[25] S. Bartoletti et al., "Positioning and sensing for vehicular safety applications in 5G and beyond," *IEEE Commun. Mag.*, vol. 59, no. 11, pp. 15–21, Nov. 2021. [Online]. Available: https://ieeexplore.ieee.org/document/9665433/

[26] B. C. Tedeschini, M. Brambilla, L. Barbieri, and M. Nicoli, "Addressing data association by message passing over graph neural networks," in *Proc. 25th Int. Conf. Inf. Fusion (FUSION)*, Linkoping, Sweden, Jul. 2022, pp. 1–7. [Online]. Available: https://ieeexplore.ieee.org/document/9841233/

[27] P. Zhang, J. Lu, Y. Wang, and Q. Wang, "Cooperative localization in 5G networks: A survey," *ICT Exp.*, vol. 3, no. 1, pp. 27–32, Mar. 2017. [Online]. Available: https://www.sciencedirect.com/science/article/pii/S2405959517300346

[28] G. Soatti, M. Nicoli, N. Garcia, B. Denis, R. Raulefs, and H. Wymeersch, "Implicit cooperative positioning in vehicular networks," *IEEE Trans. Intell. Transp. Syst.*, vol. 19, no. 12, pp. 3964–3980, Dec. 2018. [Online]. Available: https://ieeexplore.ieee.org/document/8307478/

[29] M. Brambilla, D. Pardo, and M. Nicoli, "Location-assisted subspace-based beam alignment in LOS/NLOS mm-wave V2X communications," in *Proc. IEEE Int. Conf. Commun. (ICC)*, Dublin, Ireland, Jun. 2020, pp. 1–6. [Online]. Available: https://ieeexplore.ieee.org/document/9148587/

[30] M. Brambilla, L. Combi, A. Matera, D. Tagliaferri, M. Nicoli, and U. Spagnolini, "Sensor-aided V2X beam tracking for connected automated driving: Distributed architecture and processing algorithms," *Sensors*, vol. 20, no. 12, p. 3573, Jun. 2020. [Online]. Available: https://www.mdpi.com/1424-8220/20/12/3573

[31] B. C. Tedeschini, M. Nicoli, and M. Z. Win, "On the latent space of mmWave MIMO channels for NLOS identification in 5G-advanced systems," *IEEE J. Sel. Areas Commun.*, vol. 41, no. 6, pp. 1655–1669, May 2023. [Online]. Available: https://ieeexplore.ieee.org/document/10121016/

[32] S. Marano, W. M. Gifford, H. Wymeersch, and M. Z. Win, "NLOS identification and mitigation for localization based on UWB experimental data," *IEEE J. Sel. Areas Commun.*, vol. 28, no. 7, pp. 1026–1035, Sep. 2010. [Online]. Available: http://ieeexplore.ieee.org/document/5555901/

Chapter 8. Efficient Latent Features Combination for Static Positioning

[33] H. Wymeersch, S. Marano, W. M. Gifford, and M. Z. Win, "A machine learning approach to ranging error mitigation for UWB localization," *IEEE Trans. Commun.*, vol. 60, no. 6, pp. 1719–1728, Jun. 2012. [Online]. Available: http://ieeexplore.ieee.org/document/6192275/

[34] T. Van Nguyen, Y. Jeong, H. Shin, and M. Z. Win, "Machine learning for wideband localization," *IEEE J. Sel. Areas Commun.*, vol. 33, no. 7, pp. 1357–1380, Jul. 2015. [Online]. Available: http://ieeexplore.ieee.org/document/7102989/

[35] E. Kurniawan, L. Zhiwei, and S. Sun, "Machine learning-based channel classification and its application to IEEE 802.11ad communications," in *Proc. IEEE Global Commun. Conf.*, Singapore, Dec. 2017, pp. 1–6. [Online]. Available: http://ieeexplore.ieee.org/document/8254052/

[36] X. Yang, "NLOS mitigation for UWB localization based on sparse pseudo-input Gaussian process," *IEEE Sensors J.*, vol. 18, no. 10, pp. 4311–4316, May 2018. [Online]. Available: https://ieeexplore.ieee.org/document/8322223/

[37] P. Bahl and V. N. Padmanabhan, "RADAR: An in-building RF-based user location and tracking system," in *Proc. IEEE INFOCOM. Conf. Comput. Commun., 19th Annu. Joint Conf. IEEE Comput. Commun. Societies*, Tel Aviv, Israel, Mar. 2000, pp. 775–784. [Online]. Available: http://ieeexplore.ieee.org/document/832252/

[38] M. Youssef and A. Agrawala, "The Horus WLAN location determination system," in *Proc. 3rd Int. Conf. Mobile Syst., Appl.*, Jun. 2005, pp. 205–218, doi: 10.1145/1067170.1067193.

[39] X. Wang, L. Gao, S. Mao, and S. Pandey, "DeepFi: Deep learning for indoor fingerprinting using channel state information," in *Proc. IEEE Wireless Commun. Netw. Conf. (WCNC)*, Mar. 2015, pp. 1666–1671.

[40] Y. Chapre, A. Ignjatovic, A. Seneviratne, and S. Jha, "CSI-MIMO: An efficient Wi-Fi fingerprinting using channel state information with MIMO," *Pervas. Mobile Comput.*, vol. 23, pp. 89–103, Oct. 2015. [Online]. Available: https://www.sciencedirect.com/science/article/pii/S1574119215001406

[41] X. Wang, L. Gao, and S. Mao, "PhaseFi: Phase fingerprinting for indoor localization with a deep learning approach," in *Proc. IEEE Global Commun. Conf. (GLOBECOM)*, San Diego, CA, USA, Dec. 2014, pp. 1–6. [Online]. Available: http://ieeexplore.ieee.org/document/7417517/

[42] S. Savazzi, S. Sigg, M. Nicoli, V. Rampa, S. Kianoush, and U. Spagnolini, "Device-free radio vision for assisted living: Leveraging wireless channel quality information for human sensing," *IEEE Signal Process. Mag.*, vol. 33, no. 2, pp. 45–58, Mar. 2016. [Online]. Available: http://ieeexplore.ieee.org/document/7426567/

[43] H. Chen, Y. Zhang, W. Li, X. Tao, and P. Zhang, "ConFi: Convolutional neural networks based indoor Wi-Fi localization using channel state information," *IEEE Access*, vol. 5, pp. 18066–18074, 2017. [Online]. Available: http://ieeexplore.ieee.org/document/8027020/

[44] X. Wang, X. Wang, and S. Mao, "ResLoc: Deep residual sharing learning for indoor localization with CSI tensors," in *Proc. IEEE 28th Annu. Int. Symp. Pers., Indoor, Mobile Radio Commun. (PIMRC)*, Montreal, QC, USA, Oct. 2017, pp. 1–6. [Online]. Available: http://ieeexplore.ieee.org/document/8292236/

[45] X. Wang, X. Wang, and S. Mao, "CiFi: Deep convolutional neural networks for indoor localization with 5 GHz Wi-Fi," in *Proc. IEEE Int. Conf. Commun. (ICC)*, Paris, France, May 2017, pp. 1–6. [Online]. Available: http://ieeexplore.ieee.org/document/7997235/

[46] Y. Jing, J. Hao, and P. Li, "Learning spatiotemporal features of CSI for indoor localization with dual-stream 3D convolutional neural networks," *IEEE Access*, vol. 7, pp. 147571–147585, 2019. [Online]. Available: https://ieeexplore.ieee.org/document/8864050/

[47] M. Malmstrom, I. Skog, S. M. Razavi, Y. Zhao, and F. Gunnarsson, "5G positioning—A machine learning approach," in *Proc. 16th Workshop Positioning, Navigat. Commun. (WPNC)*, Bremen, Germany, Oct. 2019, pp. 1–6. [Online]. Available: https://ieeexplore.ieee.org/document/8970186/

[48] J. Gante, G. Falcão, and L. Sousa, "Deep learning architectures for accurate millimeter wave positioning in 5G," *Neural Process. Lett.*, vol. 51, no. 1, pp. 487–514, Feb. 2020, doi: 10.1007/s11063-019-10073-1.

[49] Y. Li, S. Mazuelas, and Y. Shen, "Deep generative model for simultaneous range error mitigation and environment identification," in *Proc. IEEE Global Commun. Conf. (GLOBECOM)*, Madrid, Spain, Dec. 2021, pp. 1–6. [Online]. Available: https://ieeexplore.ieee.org/document/9685255/

[50] M. Stahlke, S. Kram, F. Ott, T. Feigl, and C. Mutschler, "Estimating TOA reliability with variational autoencoders," *IEEE Sensors J.*, vol. 22, no. 6, pp. 5133–5140, Mar. 2022. [Online]. Available: https://ieeexplore.ieee.org/document/9503384/

[51] N. Lv, F. Wen, Y. Chen, and Z. Wang, "A deep learning-based end-to-end algorithm for 5G positioning," *IEEE Sensors Lett.*, vol. 6, no. 4, pp. 1–4, Apr. 2022. [Online]. Available: https://ieeexplore.ieee.org/document/9706343/

[52] K. Gao, H. Wang, H. Lv, and W. Liu, "Toward 5G NR high-precision indoor positioning via channel frequency response: A new paradigm and dataset generation method," *IEEE J. Sel. Areas Commun.*, vol. 40, no. 7, pp. 2233–2247, Jul. 2022. [Online]. Available: https://ieeexplore.ieee.org/document/9729785/

[53] C. Wu et al., "Learning to localize: A 3D CNN approach to user positioning in massive MIMO-OFDM systems," *IEEE Trans. Wireless Commun.*, vol. 20, no. 7, pp. 4556–4570, Jul. 2021. [Online]. Available: https://ieeexplore.ieee.org/document/9364875/

[54] B. El Boudani et al., "Implementing deep learning techniques in 5G IoT networks for 3D indoor positioning: DELTA (deep learning-based co-operative architecture)," *Sensors*, vol. 20, no. 19, p. 5495, Sep. 2020. [Online]. Available: https://www.mdpi.com/1424-8220/20/19/5495

[55] H. Kim, S. H. Lee, and S. Kim, "Cooperative localization with distributed ADMM over 5G-based VANETs," in *Proc. IEEE Wireless Commun. Netw. Conf. (WCNC)*, Apr. 2018, pp. 1–5. [Online]. Available: https://ieeexplore.ieee.org/document/8377454/

[56] L. Barbieri, S. Savazzi, M. Brambilla, and M. Nicoli, "Decentralized federated learning for extended sensing in 6G connected vehicles," *Veh. Commun.*, vol. 33, Jan. 2022, Art. no. 100396. [Online]. Available: https://linkinghub.elsevier.com/retrieve/pii/S2214209621000656

[57] *Study on NR Positioning Support*, document (TR) 38.855, Version 16.0.0, 3GPP, Sep. 2019. [Online]. Available: https://portal.3gpp.org/desktopmodules/Specifications/SpecificationDetails.aspx?specificationId=3501

[58] P. Alvarez Lopez et al., "Microscopic traffic simulation using SUMO," in *Proc. 21st Int. Conf. Intell. Transp. Syst.*, Nov. 2018, pp. 2575–2582. [Online]. Available: https://elib.dlr.de/124092/

[59] D. Tse and P. Viswanath, *Fundamentals of Wireless Communication*. Cambridge, U.K.: Cambridge Univ. Press, 2005.

[60] H. L. Van Trees, *Detection, Estimation, and Modulation Theory. 4: Optimum Array Processing*. New York, NY, USA: Wiley, 2002.

[61] X. Sun, X. Gao, G. Y. Li, and W. Han, "Single-site localization based on a new type of fingerprint for massive MIMO-OFDM systems," *IEEE Trans. Veh. Technol.*, vol. 67, no. 7, pp. 6134–6145, Jul. 2018. [Online]. Available: https://ieeexplore.ieee.org/document/8307353/

[62] M. D. Zeiler and R. Fergus, "Visualizing and understanding convolutional networks," in *Computer Vision—ECCV* (Lecture Notes in Computer Science), D. Fleet, T. Pajdla, B. Schiele, and T. Tuytelaars, Eds. Cham, Switzerland: Springer, 2014, pp. 818–833.

[63] J. Schmidhuber, "Deep learning in neural networks: An overview," *Neural Netw.*, vol. 61, pp. 85–117, Jan. 2015. [Online]. Available: https://www.sciencedirect.com/science/article/pii/S0893608014002135

[64] C. M. Bishop, *Pattern Recognition and Machine Learning*, vol. 4, no. 4. Berlin, Germany: Springer, Aug. 2006. [Online]. Available: https://link.springer.com/book/9780387310732

[65] R. A. Jacobs, M. I. Jordan, S. J. Nowlan, and G. E. Hinton, "Adaptive mixtures of local experts," *Neural Comput.*, vol. 3, no. 1, pp. 79–87, Mar. 1991. [Online]. Available: http://ieeexplore.ieee.org/document/6797059/

[66] *Study on Channel Model for Frequencies From 0.5 to 100 GHz (Rel-16)*, document TR 38.901, Version 16.1.0, 3GPP, Nov. 2020. [Online]. Available: https://www.etsi.org/deliver/etsi_tr/138900_138999/138901/16.01.00_60/tr_138901v160100p.pdf

[67] C.-F. Yang and C.-J. Ko, "A ray tracing method for modeling indoor wave propagation and penetration," in *Proc. IEEE Antennas Propag. Soc. Int. Symp.*, Baltimore, MD, USA, Jul. 1996, pp. 441–444. [Online]. Available: http://ieeexplore.ieee.org/document/549632/

[68] H.-J. Li, C.-C. Chen, T.-Y. Liu, and H.-C. Lin, "Applicability of ray-tracing technique for the prediction of outdoor channel characteristics," *IEEE Trans. Veh. Technol.*, vol. 49, no. 6, pp. 2336–2349, Nov. 2000. [Online]. Available: http://ieeexplore.ieee.org/document/901902/

[69] A.-Y. Hsiao, C.-F. Yang, T.-S. Wang, I. Lin, and W.-J. Liao, "Ray tracing simulations for millimeter wave propagation in 5G wireless communications," in *Proc. IEEE Int. Symp. Antennas Propag. USNC/URSI Nat. Radio Sci. Meeting*, San Diego, CA, USA, Jul. 2017, pp. 1901–1902. [Online]. Available: http://ieeexplore.ieee.org/document/8072993/

[70] Z. Yun and M. F. Iskander, "Ray tracing for radio propagation modeling: Principles and applications," *IEEE Access*, vol. 3, pp. 1089–1100, 2015. [Online]. Available: http://ieeexplore.ieee.org/document/7152831/

[71] *Guidelines for Evaluation of Radio Interface Technologies for IMT-2020*, document SG05 ContribuQon 57, International Telecommunication Union (ITU)-R M, 2017. [Online]. Available: https://www.itu.int/md/R15-SG05-C-0057

[72] A. Paszke et al., "Automatic differentiation in PyTorch," in *Proc. NIPS*, Oct. 2017, pp. 1–4. [Online]. Available: https://openreview.net/forum?id=BJJsrmfCZ

[73] D. P. Kingma and J. Ba, "Adam: A method for stochastic optimization," 2014, *arXiv:1412.6980*.

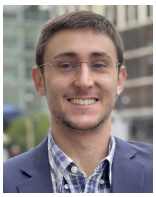
[74] V. Badrinarayanan, A. Kendall, and R. Cipolla, "SegNet: A deep convolutional encoder–decoder architecture for image segmentation," *IEEE Trans. Pattern Anal. Mach. Intell.*, vol. 39, no. 12, pp. 2481–2495, Dec. 2017. [Online]. Available: http://ieeexplore.ieee.org/document/7803544/

[75] D. Hendrycks and K. Gimpel, "Gaussian error linear units (GELUs)," 2016, *arXiv:1606.08415*.

[76] R. Schmidt, "Multiple emitter location and signal parameter estimation," *IEEE Trans. Antennas Propag.*, vol. AP-34, no. 3, pp. 276–280, Mar. 1986. [Online]. Available: http://ieeexplore.ieee.org/document/1143830/

[77] F.-G. Yan, J. Wang, S. Liu, Y. Shen, and M. Jin, "Reduced-complexity direction of arrival estimation using real-valued computation with arbitrary array configurations," *Int. J. Antennas Propag.*, vol. 2018, pp. 1–10, Apr. 2018. [Online]. Available: https://www.hindawi.com/journals/ijap/2018/3284619/

[78] *Study on Enhancement of 3GPP Support for 5G V2X Services*, document (TR) 22.886, Version 16.2.0, 3GPP, Dec. 2018. [Online]. Available: https://portal.3gpp.org/desktopmodules/Specifications/SpecificationDetails.aspx?specificationId=3108



Bernardo Camajori Tedeschini (Graduate Student Member, IEEE) received the B.Sc. (Hons.) in Computer Science and M.Sc. (Hons.) degrees in Telecommunications Engineering from the Politecnico di Milano, Italy, in 2019 and 2021, respectively. From November 2021 he started as PhD fellow in Information Technology at Dipartimento di Eleeronica, Informazione e Bioingegneria (DEIB), Politecnico di Milano. He is currently a visiting researcher with the Laboratory for Information & Decision Systems at the Massachusees Institute of Technology (MIT), Cambridge, MA.

His research interests include federated learning, machine learning and localization methods. He was a recipient of the Ph.D. grant from the ministry of the Italian government Ministero dell'Istruzione, dell'Università e della Ricerca (MIUR) and the Roberto Rocca Doctoral Fellowship granted by MIT and Politecnico di Milano.

Monica Nicoli (Senior Member, IEEE) received the M.Sc. (Hons.) and Ph.D. degrees in communication engineering from Politecnico di Milano, Milan, Italy, in 1998 and 2002, respectively. She was a Visiting Researcher with ENI Agip, from 1998 to 1999, and Uppsala University, in 2001. In 2002, she joined Politecnico di Milano as a Faculty Member. She is currently an Associate Professor in telecommunications with the Department of Management, Economics and Industrial Engineering.

Her research interests include signal processing, machine learning, and wireless communications, with emphasis on smart mobility and Internet of Things (IoT). She was a recipient of the Marisa Bellisario Award, in 1999, and a co-recipient of the best paper awards of the EuMA Mediterranean Microwave Symposium, in 2022, the IEEE Symposium on Joint Communications and Sensing, in 2021, the IEEE Statistical Signal Processing Workshop, in 2018, and the *IET Intelligent Transport Systems journal*, in 2014. She is an Associate Editor of the IEEE TRANSACTIONS ON INTELLIGENT TRANSPORTATION SYSTEMS. She has also served as an Associate Editor for the *EURASIP Journal on Wireless Communications and Networking*, from 2010 to 2017, and a Lead Guest Editor for the Special Issue on Localization in Mobile Wireless and Sensor Networks, in 2011.

Open Access funding provided by 'Politecnico di Milano' within the CRUI CARE Agreement

CHAPTER 9

Efficient Uncertainty Quantification for Mobile Positioning

In this chapter, we tackle the task of UE tracking in urban environments with next-generation cellular networks. Given the presence of high-blockage conditions, the main challenge is how to estimate in real-time the reliability of DL model predictions. In the paper, we propose two complementary contributions to this challenge. First, we propose the integration of BNN into tracking filters by exploiting the predicted uncertainty as likelihood measure, which permits the coherent combination of multiple BSs' position estimates. Second, we introduce a novel BNN algorithm, namely BBK, which exploits the teacher-student paradigm to perform uncertainty quantification without the need of performing sampling procedures at inference phase. Moreover, BBK, as opposed to state-of-the-art real-time BNN methods, is able to fully evaluate the output's uncertainty by distinguishing between aleatoric and epistemic uncertainties. This enables insights behind the uncertainty prediction and can guide for more efficient sample gathering. We test the proposed BBK method with an AE-based DL model which predicts the UE position from ADCPM samples. The results indicate that the BBK approach can accurately estimate both aleatoric and epistemic uncertainties, surpassing the performances of current real-time BNN methods, particularly in OoD scenarios. In terms of mobile positioning, the proposed cooperative tracking method outperforms traditional geometric tracking filters, as well as RNN models thanks to the coherent fusion of multiple BSs predictions.

Real-Time Bayesian Neural Networks for 6G Cooperative Positioning and Tracking

Bernardo Camajori Tedeschini, *Graduate Student Member, IEEE*, Girim Kwon, *Member, IEEE*, Monica Nicoli, *Senior Member, IEEE*, and Moe Z. Win, *Fellow, IEEE*

***Abstract*— In the evolving landscape of 5G new radio and related 6G evolution, achieving centimeter-level dynamic positioning is pivotal, especially in cooperative intelligent transportation system frameworks. With the challenges posed by higher path loss and blockages in the new frequency bands (i.e., millimeter waves), machine learning (ML) offers new approaches to draw location information from space-time wide-bandwidth radio signals and enable enhanced location-based services. This paper presents an approach to real-time 6G location tracking in urban settings with frequent signal blockages. We introduce a novel teacher-student Bayesian neural network (BNN) method, called Bayesian bright knowledge (BBK), that predicts both the location estimate and the associated uncertainty in real-time. Moreover, we propose a seamless integration of BNNs into a cellular multi-base station tracking system, where more complex channel measurements are taken into account. Our method employs a deep learning (DL)-based autoencoder structure that leverages the complete channel impulse response to deduce location-specific attributes in both line-of-sight and non-line-of-sight environments. Testing in 3GPP specification-compliant urban micro (UMi) scenario with ray-tracing and traffic simulations confirms the BBK's superiority in estimating uncertainties and handling out-of-distribution testing positions. In dynamic conditions, our BNN-based tracking system surpasses geometric-based tracking techniques and state-of-the-art DL models, localizing a moving target with a median error of 46 cm.**

***Index Terms*— Bayesian neural networks, tracking, deep learning, channel impulse response, intelligent transportation systems.**

Manuscript received 11 November 2023; revised 19 March 2024; accepted 18 April 2024. Date of current version 21 August 2024. The fundamental research described in this paper was supported, in part, by the Roberto Rocca Doctoral Fellowship granted by Massachusetts Institute of Technology and Politecnico di Milano, by the European Union — NextGenerationEU (National Sustainable Mobility Center CN00000023, Italian Ministry of University and Research Decree n. 1033—17/06/2022, spoke 9), by the National Research Foundation of Korea under Grant 2021R1A6A3A14040142, by the National Science Foundation under Grant CNS-2148251, and by the federal agency and industry partners in the RINGS Program. The material in this paper will be presented in part at the IEEE International Conference on Communications, Denver, CO, USA, in June 2024. *(Corresponding author: Moe Z. Win.)*

Bernardo Camajori Tedeschini is with the Dipartimento di Elettronica, Informazione e Bioingegneria (DEIB), Politecnico di Milano, 20133 Milan, Italy, and also with the Wireless Information and Network Sciences Laboratory, Massachusetts Institute of Technology, Cambridge, MA 02139 USA.

Girim Kwon is with the Wireless Information and Network Sciences Laboratory, Massachusetts Institute of Technology, Cambridge, MA 02139 USA.

Monica Nicoli is with the Department of Management, Economics and Industrial Engineering (DIG), Politecnico di Milano, 20156 Milan, Italy.

Moe Z. Win is with the Laboratory for Information and Decision Systems (LIDS), Massachusetts Institute of Technology, Cambridge, MA 02139 USA (e-mail: moewin@mit.edu).

Color versions of one or more figures in this article are available at https://doi.org/10.1109/JSAC.2024.3413950.

Digital Object Identifier 10.1109/JSAC.2024.3413950

I. Introduction

POSITIONING and tracking capabilities have become increasingly crucial in the evolution of cellular networks, as they provide benefits to the 5th generation (5G) use cases [1], [2], [3]. Since the adoption of 3rd Generation Partnership Project (3GPP) Release 15 in 2018, these networks have not only achieved rapid development [4], [5], [6], but have also expanded to introduce new use cases and services [7]. Notably in 3GPP Releases 16 and 17, location awareness systems have been extended beyond from regulatory applications to commercial and roaming functionalities [8], [9], [10]. Still, the major leap forward in positioning performances is expected with the advent of 5G Advanced in 3GPP Release 18 [11], [12], [13], [14], where the primary goal of centimeter-level absolute accuracy will be achieved thanks to transformative key features enablers, i.e., massive multiple-input multiple-output (mMIMO) [15], larger bandwidths and millimeter waves (mmWave) [16]. The main challenges are the higher path loss and frequent blockages, which limit the potential of conventional and global navigation satellite systems (GNSS)-based solutions. Indeed, under the absence of line-of-sight (LOS) link, GNSS becomes challenging to utilize effectively, even with advanced satellite techniques such as real-time kinematic (RTK) [17].

Extensive research in the domain of localization and navigation has explored various aspects of these challenges, focusing on fundamental limits [18], [19], [20], [21], network operation and experimentation [22], [23], [24], [25], and algorithm design [26], [27], [28], [29]. To solve these challenges, 5G Advanced has pushed interest in leveraging artificial intelligence (AI) and machine learning (ML) for assisted or even direct positioning [30]. Indeed, base stations (BSs) often have access to a large number of historical channel state information (CSI) [31], [32], [33], which can be exploited through deep learning (DL) methods (e.g., autoencoder (AE) structures [34]) as location fingerprints [35]. The advantages of direct artificial intelligence (AI)/machine learning (ML) positioning are the ability to perform both LOS and non-LOS (NLOS) positioning, and single-BS localization, enabled by integrated sensing and communication (ISAC) frameworks [36], [37], [38]. Therefore, we foresee these solutions as a promising long-term answer to advanced positioning methods.

Despite the potential of ML in positioning applications, traditional ML approaches have limitations, especially concerning uncertainty quantification. In critical applications, such as tracking of connected automated vehicles (CAVs) [39],

[40], [41], lack of uncertainty quantification can be a major limiting or even blocking factor as integrity and reliability requirements are very stringent. From this point of view, Bayesian neural networks (BNNs) offer a promising solution to these challenges [42], as they not only provide point estimates but also quantify the uncertainty associated with these estimates [43], allowing for more reliable and robust positioning. In particular, in static positioning, BNNs are more resilient to overfitting during training with respect to neural networks (NNs), they can incorporate prior knowledge on the problem at hand and, more importantly, are able to characterize the uncertainty of the model (i.e., whether it is due to the lack of training samples or to the intrinsic characteristics of the data). Furthermore, in dynamic settings, i.e., tracking, the uncertainty can be cleverly combined in Bayesian tracking solutions.

While BNNs address key limitations of conventional ML algorithms, they also present specific challenges. A main drawback is the need for sampling during inference time, which may not be suitable for real-time applications [44]. Another issue is the computational and storage overhead of maintaining multiple NN configurations for Bayesian inference. Given the lack of a complete solution to these problems and the great potential of BNNs in cellular tracking systems, in this paper, we explore and propose the integration of real-time BNN solutions into future 6G systems. These advancements aim to bring together the best of both worlds: the robustness and uncertainty quantification of BNNs along with the speed and efficiency required for next-generation tracking systems.

The rest of this paper is structured as follows. Sec. II presents the related works including next-generation cellular positioning, ML for static and mobile positioning, and BNNs for real-time inference, along with the paper's main contributions. Sec. III describes the channel model and the channel fingerprint used for positioning. In Sec. IV, we discuss the proposed real-time BNN method and its application to a DL model with AE structure. Sec. V presents the integration of BNN methods in next-generation cellular tracking systems. Sec. VI presents a case study, and Sec. VII draws the conclusions.

II. Related Works and Contributions

A. Next Generation Cellular Positioning

3GPP Release 18 is expected to significantly enhance the existing positioning standards by introducing key methodologies. It introduces the support of carrier phase positioning (CPP), a GNSS-based technology known for its centimeter-level precision, though it is traditionally limited to outdoor use where GNSS signals are not blocked [45]. Moreover, Release 18 is set to fulfill low power high accuracy positioning (LPHAP) standards and introduce positioning features for reduced capability (RedCap) user equipments (UEs), such as wearable medical devices, and augmented reality goggles. Finally, it comprises studies on sidelink (SL) positioning, e.g., the design of SL-positioning reference signal (PRS) [46].

However, in case of non-cooperative and non-RedCap UEs moving in frequent blockage environment, e.g., CAVs in urban scenario, geometric-based positioning methodologies struggle in fulfilling the requirements. For this reason, AI-based solutions have been studied, particularly regarding complexity, positioning performances, and generalizations [11].

B. ML for Static Positioning

Significant works on ML-based wireless positioning were developed with the introduction of MIMO-orthogonal frequency division multiplexing (OFDM) systems in the IEEE 802.11a/n protocol, which enabled the extraction of CSI from commercial Wi-Fi devices. The availability of channel information across multiple carriers and antennas allowed for detailed insights into radio signal propagation, enabling learning of the user's position [47], [48]. DL techniques were utilized to learn the best non-linear combination of features for tasks like NLOS classification or position estimation, with many studies adopting convolutional neural networks (CNNs) for feature extraction [49], [50], [51], [52].

Regarding 5G positioning, gathering the knowledge of Wi-Fi works, two main components started to delineate: the usage of full channel impulse response (CIR) data as input features for positioning and the adoption of AE NN structures. Leveraging full CIR data, especially when organized into image-like structures, is gaining a lot of momentum. Authors in [31] adopted the channel frequency response (CFR) matrix obtained both from ray-tracing simulations and real experiments. However, while being effective for positioning, the CFR does not distinctly represent the angle of arrival (AOA) or time of flight (TOF) for each path, potentially complicating feature extraction. A different study employed a 3D angle-delay channel power matrix (ADCPM) and a CNN with inception modules to predict UE position [53], but with double the inference time and four times the computational storage compared to 2D ADCPM. Dealing with image structure permits employing more complex DL models which compress the channel into a compact and efficient representation, namely AE.

C. ML for Mobile Positioning and Tracking

UE tracking by ML-based methods in the context of 5G networks is a relatively unexplored area since the majority of previous works employed conventional Bayesian techniques, e.g., extended Kalman filter (EKF) [54] or message passing algorithm (MPA) [55], in conjunction with mmWave and MIMO enablers. Authors in [56] adopted state-of-the-art temporal convolutional network (TCN) models to perform NLOS outdoor tracking, reaching a mean absolute error (MAE) of 1.8 m. A similar work has been carried out in indoor conditions [57], with long short-term memory (LSTM) and CNN applied to raw CSI fingerprinting. However, LSTMs and TCNs have two main drawbacks. First, they require a set of training trajectories with highly accurate ground truth positions. While this is practical for static positioning, for

dynamic positioning, especially in outdoor conditions, it is difficult to obtain the ground truth target position while moving, unless using high-precision optical laser positioning systems. Second, the conventional LSTMs and TCNs do not provide an uncertainty measure of their predictions, thus limiting the deployment in safety-critical applications.

D. BNN for Real-Time Inference

In the context of DL-based uncertainty estimation, it is essential to distinguish between aleatoric and epistemic uncertainties [58]. These two types of uncertainties are the roots behind the prediction uncertainty and they are generated by two different phenomena. Aleatoric uncertainty refers to the dispersion of the predicted distribution of our target variable, e.g., UE location, based on the given features, which arises from measurement inaccuracies. Therefore, aleatoric uncertainty remains unchanged even with additional data collection under identical experimental conditions. This specific uncertainty, referred to as *data uncertainty*, can be made data-dependent and learned as an additional output from conventional NNs [59]. Thus, these solutions are important for real-time applications, where no Monte Carlo (MC) sampling procedures are required, but fail to generalize in out-of-distribution (OOD) scenarios with sparse training data.

For this reason, BNNs and epistemic uncertainty evaluation have been studied [60]. The epistemic uncertainty, also known as *model uncertainty*, derives from the uncertainty over the BNN model parameters, which are considered random variables. Contrary to the aleatoric uncertainty, the ambiguity on the model parameters, and thus on the output, can be explained away by providing more training samples. Existing BNN methods, both exact methods like Markov chain Monte Carlo (MCMC) [61], [62] and approximations like variational inference (VI) [63], [64], [65], [66], [67], aim at effectively sampling from the posterior distribution of the weights and predicting the posterior predictive distribution over the output. This is done by having multiple NN parameters instances and by predicting multiple times the same input sample.

Many works tried to tackle the problem of simultaneous learning of aleatoric and epistemic uncertainties. The authors in [59] adopted a BNN derived from Monte Carlo Dropout (MCDropout) method which also predicted the aleatoric uncertainty of data through a specific loss function. Despite achieving good results, this method still relies on a sampling inference procedure, thus not being suitable for real-time applications. A solution to this issue can be found in the so-called *teacher-student* techniques, such as Bayesian dark knowledge (BDK) [68], where a student NN, i.e., non-Bayesian, is trained to mimic the output of a teacher BNN, which is on the other hand Bayesian thus learning both the points estimates and the output uncertainties. During the inference phase, the real-time uncertainty estimation is performed by the student NN without requiring time-consuming sampling procedures. The primary challenge in teacher-student methods is the student's inability to differentiate between the two distinct uncertainties in output. Distinguishing between these uncertainties is crucial, as it provides insights into the reasons for a model's uncertainty regarding a specific test sample, such as insufficient training data points or inherent data noise. Moreover, recognizing these uncertainties can guide where additional training points would be most beneficial (i.e., wherever high epistemic uncertainty and low aleatoric uncertainty are present). At the present day and to the authors' knowledge, no real-time solution to both aleatoric and epistemic uncertainty learning is present in the literature (see Table I), in particular regarding safety-critical applications such as automated driving.

TABLE I
COMPARISON OF METHODS FOR REAL-TIME, ALEATORIC AND EPISTEMIC PREDICTION CAPABILITIES

	Real-time prediction	Aleatoric prediction	Epistemic prediction
MCMC/VI [61]–[67]	No	No	Yes
Bayesian & aleatoric regres. [59]	No	Yes	Yes
BDK [68]	Yes	Yes	No
BBK (proposed)	Yes	Yes	Yes

E. Contributions

In this paper, we address the problem of UE tracking in next-generation networks through the usage of the full CIR and DL-based predictions, whose output uncertainty is obtained through real-time BNN techniques and seamlessly integrated into existing tracking systems. The real-time BNN methodology, namely Bayesian bright knowledge (BBK), is built with a teacher-student paradigm as it is the only methodology of BNN which does not require sampling for producing the uncertainty estimation. The localization system is based on a NN trained on offline gathered data, which are limited by spatial density. To account for both the aleatoric and epistemic uncertainties caused by noisy measurements and limited density of the training points, respectively, we propose a Bayesian tracking approach based on a BNN. This BNN constructs the model that links measurements and positions, optimally predicting and weighing uncertainties in the position calculation.

The main contributions of this paper are summarized in the following.

- We design a teacher-student BNN method, i.e., BBK, that predicts both epistemic and aleatoric uncertainties without requiring a sampling procedure during inference phase. This makes it suitable for real-time and safety-critical use cases.
- We propose the integration of BNNs in cellular tracking systems where a set of cooperative BSs aims at tracking a moving target. The integrated system is easy to be implemented and is compatible with any BNN method.
- We develop a DL model based on an AE structure which exploits the complete CIR, i.e., sparse ADCPM matrices, to extract location-specific features and perform positioning in both LOS and NLOS settings.
- We model a realistic cooperative intelligent transportation system (C-ITS) setting in an urban context. Our simulated network aligns with the 5G standard [69] and offers realistic outdoor conditions using Wireless InSite 3D ray-tracing software and MATLAB software. We emulate

various vehicle trajectories, or UEs, designed with the simulation of urban mobility (SUMO) software [70].

1) Notation: A random variable and its realization are denoted by x and x; a random vector and its realization are denoted by $\mathbf{x}$ and $\boldsymbol{x}$; a random matrix and its realization are denoted by $\mathbf{X}$ and $\boldsymbol{X}$, respectively. The function $p_{\mathsf{x}}(x)$, and simply $p(x)$ when there is no ambiguity, denotes the probability density function (PDF) of x. $j = \sqrt{-1}$ denotes the imaginary unit. The notations $\mathbf{X}^\top$, $\mathbf{X}^*$ and $\mathbf{X}^{\mathrm{H}}$ indicate the matrix transposition, conjugation and conjugate transposition. $\det(\cdot)$ and $\mathrm{Tr}(\cdot)$ denote the determinant and the trace of the matrix argument, respectively. The Kronecker and Hadamard products between two matrices are denoted with $\otimes$ and $\odot$, respectively. With the notation $\mathsf{x} \sim \mathcal{N}(\mu, \sigma^2)$ we indicate a Gaussian random variable x with mean μ and standard deviation σ, whose PDF is denoted by $\mathcal{N}(x; \mu, \sigma^2)$. With the notation $\mathsf{y} \sim \mathcal{U}(a, b)$ we indicate a uniform random variable y with support $[a, b]$. We use $\mathbb{E}\{\cdot\}$ and $\mathbb{V}\{\cdot\}$ to denote the expectation and the variance of random variable, respectively. $\mathbb{R}$ and $\mathbb{C}$ stand for the set of real and complex numbers, respectively. $\lfloor x \rfloor$ indicates the largest integer not greater than x. $|\boldsymbol{x}|$ denotes the length of the vector $\boldsymbol{x}$. $\delta(\cdot)$ is the Kronecker delta function.

III. System Model

A. Channel Model

Consider a mmWave OFDM system where a UE communicates with a BS in uplink direction at carrier wavelength λ_{c}. The BS is equipped with a uniform planar array (UPA) composed of $N_{\mathrm{v}} \times N_{\mathrm{h}}$ isotropic antenna elements (i.e., N_{v} and N_{h} elements in the vertical and horizontal directions with the antenna spacings of d_{v} and d_{h}, respectively). On the contrary, the UE holds a single omni-directional antenna. The channel between the BS and the UE (see Fig. 1) is composed of N_{p} distinct paths, each with a TOF τ_p, a zenith AOA $\theta_p \in [0, \pi]$, and an azimuth AOA $\varphi_p \in [0, \pi]$, for path $p = 1, 2, \ldots, N_{\mathrm{p}}$.

We employ an OFDM scheme with a sampling interval of T_{s}, N_{c} sub-carriers, and a symbol duration given by $T_{\mathrm{c}} = N_{\mathrm{c}} T_{\mathrm{s}}$. For the k-th sub-carrier, the frequency is $f_k = \frac{k}{T_{\mathrm{c}}}$, $k = 0, 1, \ldots, N_{\mathrm{c}} - 1$. We assume that the cyclic-prefix duration $T_{\mathrm{g}} = N_{\mathrm{g}} T_{\mathrm{s}}$ surpasses the maximum channel delay, denoted by τ_{MAX}. Here, N_{g} represents the number of sampling intervals constituting a guard interval.

Assuming a sampling rate of $1/T_{\mathrm{s}}$ and treating each path as independent and wide-sense stationary [71], the CFR for the k-th sub-carrier can be expressed as [72], [73]

$$\mathbf{h}_k = \sum_{p=1}^{N_{\mathrm{p}}} \bar{\alpha}_{p,k}\, \mathbf{e}(\theta_p, \varphi_p) \in \mathbb{C}^{N_{\mathrm{h}} N_{\mathrm{v}}} \tag{1}$$

where $\bar{\alpha}_{p,k} = \alpha_p e^{-j2\pi\tau_p f_k}$ is the channel gain in the frequency domain, $\alpha_p = a_p e^{-j2\pi(\frac{d_p}{\lambda_{\mathrm{c}}} - \nu_p \tau_p)}$ is the complex path gain of which includes the Doppler frequency shift ν_p and has average power $\sigma_p^2 = \mathbb{E}\{\|a_p\|^2\}$ and $d_p = c\tau_p$ is the traveled distance (where c is the speed of light in air), and $\mathbf{e}(\theta_p, \varphi_p) \in \mathbb{C}^{N_{\mathrm{h}} N_{\mathrm{v}} \times 1}$ is the array response vector [71]. Finally, by considering the different CFRs at every sub-carrier, we get the space-frequency channel response matrix (SFCRM) as:

$$\mathbf{H} = [\mathbf{h}_0\ \mathbf{h}_1\ \cdots\ \mathbf{h}_{N_{\mathrm{c}}-1}] \in \mathbb{C}^{N_{\mathrm{h}} N_{\mathrm{v}} \times N_{\mathrm{c}}}. \tag{2}$$

In the next subsection, we show how to extract the ADCPM fingerprint from (2) obtained at the BS.

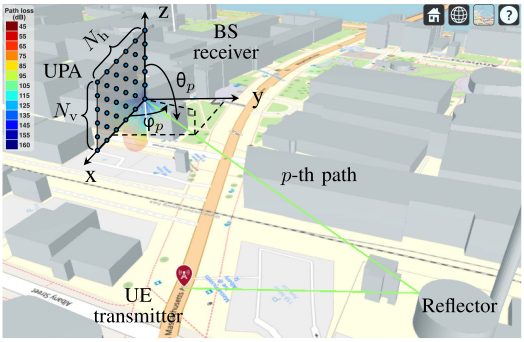


Fig. 1. An uplink scenario with a UE transmitting to a BS where the direction of arrival (DOA) of the p-th path is highlighted with zenith θ_p and azimuth φ_p angles.

B. Location-Dependent Fingerprint

For location estimation, it is advantageous to map the channel response into the angle-delay domain. This transformation simplifies the identification of macro-paths, i.e., clusters for both LOS and NLOS components, which vary with the environmental context, serving as location-specific features or fingerprints. To obtain angle-delay domain features, we convert the SFCRM as defined in (2) by employing phase-shifted discrete Fourier transform (DFT) matrices $\boldsymbol{V}_{N_{\mathrm{h}}} \in \mathbb{C}^{N_{\mathrm{h}} \times N_{\mathrm{h}}}$ and $\boldsymbol{V}_{N_{\mathrm{v}}} \in \mathbb{C}^{N_{\mathrm{v}} \times N_{\mathrm{v}}}$, where $[\boldsymbol{V}_{N_{\mathrm{h}}}]_{\bar{i},\bar{j}} = \frac{1}{\sqrt{N_{\mathrm{h}}}} e^{-j2\pi \frac{\bar{i}\left(\bar{j} - \frac{N_{\mathrm{h}}}{2}\right)}{N_{\mathrm{h}}}}$ and $[\boldsymbol{V}_{N_{\mathrm{v}}}]_{\bar{i},\bar{j}} = \frac{1}{\sqrt{N_{\mathrm{v}}}} e^{-j2\pi \frac{\bar{i}\left(\bar{j} - \frac{N_{\mathrm{v}}}{2}\right)}{N_{\mathrm{v}}}}$. Moreover, we denote with $\boldsymbol{F} \in \mathbb{C}^{N_{\mathrm{c}} \times N_{\mathrm{g}}}$ the matrix formed by the first N_{g} columns of N_{c} dimensional unitary DFT matrix where $[\boldsymbol{F}]_{\bar{i},\bar{j}} = \frac{1}{\sqrt{N_{\mathrm{c}}}} e^{-j2\pi \frac{\bar{i}\bar{j}}{N_{\mathrm{c}}}}$. The angle-delay channel response matrix (ADCRM) can be then computed as [53]

$$\mathbf{G} = \frac{1}{\sqrt{N_{\mathrm{h}} N_{\mathrm{v}} N_{\mathrm{c}}}} (\boldsymbol{V}_{N_{\mathrm{h}}}^{\mathrm{H}} \otimes \boldsymbol{V}_{N_{\mathrm{v}}}^{\mathrm{H}})\, \mathbf{H}\, \boldsymbol{F}^* \in \mathbb{C}^{N_{\mathrm{h}} N_{\mathrm{v}} \times N_{\mathrm{g}}} \tag{3}$$

where $\boldsymbol{V}_{N_{\mathrm{h}}}^{\mathrm{H}} \otimes \boldsymbol{V}_{N_{\mathrm{v}}}^{\mathrm{H}}$ and $\boldsymbol{F}^*$ project the SFCRM into the angle and delay domains, respectively.

From (3), we can obtain the ADCPM as

$$\boldsymbol{P} = \mathbb{E}\{\mathbf{G} \odot \mathbf{G}^*\} \in \mathbb{R}^{N_{\mathrm{h}} N_{\mathrm{v}} \times N_{\mathrm{g}}} \tag{4}$$

where $[\boldsymbol{P}]_{\bar{i},\bar{j}} = \mathbb{E}\{\|[\mathbf{G}]_{\bar{i},\bar{j}}\|^2\}$. An important property of the ADCPM is that for N_{v}, N_{h} and $N_{\mathrm{g}} \to \infty$, the ADCPM becomes a sparse matrix whose elements $[\boldsymbol{P}]_{\bar{i},\bar{j}}$ match the average channel power of the $\bar{i}$-th AOA and the $\bar{j}$-th TOF as [53]

$$\lim_{N_{\mathrm{h}}, N_{\mathrm{v}}, N_{\mathrm{g}} \to \infty} [\boldsymbol{P}]_{\bar{i},\bar{j}} = \sum_{p=1}^{N_{\mathrm{p}}} \sigma_p^2\, \delta(\bar{i} - m_p N_{\mathrm{v}} - n_p) \delta(\bar{j} - r_p) \tag{5}$$

where $r_p = \lfloor \frac{\tau_p}{T_s} \rfloor$ is the resolvable delay corresponding to the p-th path, $n_p = \frac{N_v}{2} + \frac{N_v d_v}{\lambda_c} \cos\theta_p$ and $m_p = \frac{N_h}{2} + \frac{N_h d_h}{\lambda_c} \sin\theta_p \cos\phi_p$. Hence, the statistical properties of the ADCPM facilitate the DL model's ability to capture location-specific attributes, providing consistent and reliable fingerprints for location estimation.

C. DL Model Input

We propose using the ADCPM in (4) as the measurement basis for estimating the UE location. This sparse matrix effectively serves as a visual snapshot of the multipath environment in the power-angle-delay domain, from which a DL model like a CNN can glean key location-centric features. In addition, the ADCPM encapsulates all essential information, i.e., TOF, AOA, and received signal strength (RSS) for each path, while maintaining low storage and computational requirements due to channel sparsity. To illustrate this aspect, in Fig. 2(b) we show an example ADCPM denoted by $\boldsymbol{P}$, comprising $N_g = 352$ delay samples and $N_h N_v = 64$ angular directions. Notably, even without an extensive array of antennas or high sample resolution, the sparsity of the matrix is evident. Therefore, in the experiment, we employ the ADCPM $\boldsymbol{P}$ as the DL model's input $\mathbf{x}$ for performing tracking.

Despite the ability of the ADCPM to provide consistent and reliable fingerprints for location estimation, we need to face the practical challenges of noisy measurements, i.e., the non-perfect matching between the input ADCPM and the related position due to multipath fading and channel estimate errors, and the limited spatial density of training points. In particular, these two challenges create two types of uncertainties, namely, aleatoric and epistemic uncertainties, respectively, which we propose to assess through the usage of BNN. Therefore, in Sec. IV, we detail how to train a BNN to obtain both the position and its related uncertainties.

IV. Real-Time BNN Methodology

A. Problem Formulation

We consider a supervised regression setting where the UE's position is defined by the target variable t, which is considered as a scalar here to simplify the derivation and modeled as

$$\mathsf{t} = f(\mathbf{x}) + \boldsymbol{\varepsilon}(\mathbf{x}) \tag{6}$$

where $f(\mathbf{x})$ is a non-linear function which takes as input $\mathbf{x}$ (e.g., a channel measurement as the ADCPM) and $\boldsymbol{\varepsilon}(\mathbf{x}) \sim \mathcal{N}\big(0, \sigma_{\boldsymbol{\varepsilon}}(\mathbf{x})^2\big)$ is a random noise. Extension to a vector target variable (i.e., 3D position) is in Sec. IV-E. The objective is to approximate the function $f(\mathbf{x})$ to $y(\mathbf{x}, \boldsymbol{\theta})$ using a NN with parameters $\boldsymbol{\theta}$. The vector $\boldsymbol{\theta}$ is learned using N training points composing a training dataset $\mathcal{D} = \{(t_n, \boldsymbol{x}_n) \mid t_n \in \mathcal{D}_t, \boldsymbol{x}_n \in \mathcal{D}_{\boldsymbol{x}}\}_{n=1}^{N}$, where $\mathcal{D}_t$ and $\mathcal{D}_{\boldsymbol{x}}$ contain the training targets and inputs, respectively. From the training dataset, we define the likelihood function as:

$$p_{\mathsf{D_t}|\mathsf{D_x},\boldsymbol{\theta}}(\mathcal{D}_t|\mathcal{D}_{\boldsymbol{x}}, \boldsymbol{\theta}) = \prod_{n=1}^{N} \mathcal{N}\big(t_n; y(\boldsymbol{x}_n, \boldsymbol{\theta}), \sigma_{\boldsymbol{\varepsilon}}(\boldsymbol{x}_n)^2\big) \tag{7}$$

assuming independence between target variables.

In non-Bayesian ML settings, a discriminative probabilistic approach is adopted [74], where $\boldsymbol{\theta}$ are obtained via maximum likelihood estimation (MLE) and by directly defining the posterior conditional probability as $p_{\mathsf{t}|\mathbf{x},\mathsf{D}}(t|\boldsymbol{x}, \mathcal{D}) = p_{\mathsf{t}|\mathbf{x},\boldsymbol{\theta}}(t|\boldsymbol{x}, \boldsymbol{\theta}_{\mathrm{MLE}})$, where $\boldsymbol{\theta}_{\mathrm{MLE}} = \mathrm{argmin}_{\boldsymbol{\theta}}\{-\log p(\mathcal{D}_t|\mathcal{D}_{\boldsymbol{x}}, \boldsymbol{\theta})\}$ are obtained with gradient descent optimization methods. Usually, the negative log-likelihood $-\log p(\mathcal{D}_t|\mathcal{D}_{\boldsymbol{x}}, \boldsymbol{\theta})$ is called loss or error function. On the contrary, in Bayesian settings, the network is stochastic and is described by random parameters with prior distribution $p_{\boldsymbol{\theta}}(\boldsymbol{\theta})$ accounting for the uncertainty of the model due to the finite size of the training dataset. In Bayesian NNs, a generative approach is employed by computing the so-called *posterior predictive distribution* [43]:

$$p_{\mathsf{t}|\mathbf{x},\mathsf{D}}(t|\boldsymbol{x}, \mathcal{D}) = \int p_{\mathsf{t}|\mathbf{x},\boldsymbol{\theta}}(t|\boldsymbol{x}, \boldsymbol{\theta})\, p_{\boldsymbol{\theta}|\mathsf{D}}(\boldsymbol{\theta}|\mathcal{D}) d\boldsymbol{\theta} \tag{8}$$

where $p_{\boldsymbol{\theta}|\mathsf{D}}(\boldsymbol{\theta}|\mathcal{D})$ is the posterior distribution. It is called *predictive* as it is used to make predictions on new, unseen data, and to differentiate it with respect to the posterior distribution over the model parameters. However, in practice, the posterior $p_{\boldsymbol{\theta}|\mathsf{D}}(\boldsymbol{\theta}|\mathcal{D})$ is highly dimensional and non-convex, leading to computational intractability. Thus, the majority of BNN methods approximate $p_{\boldsymbol{\theta}|\mathsf{D}}(\boldsymbol{\theta}|\mathcal{D})$ with a sampling procedure and estimate the (8) with MC sampling as

$$p_{\mathsf{t}|\mathbf{x},\mathsf{D}}(t|\boldsymbol{x}, \mathcal{D}) \backsimeq \frac{1}{L} \sum_{\ell=1}^{L} p(t|\boldsymbol{x}, \boldsymbol{\theta}_\ell) \tag{9}$$

where L is the number of samples $\boldsymbol{\theta}_\ell$ drawn from $p_{\boldsymbol{\theta}|\mathsf{D}}(\boldsymbol{\theta}|\mathcal{D})$.

B. Predictive Mean and Variance Estimation

Given the posterior predictive distribution in (9), we can obtain the predictive mean of t as

$$\mathbb{E}\{\mathsf{t}|\mathbf{x}, \mathsf{D}\} \backsimeq \frac{1}{L} \sum_{\ell=1}^{L} \int t\, p(t|\boldsymbol{x}, \boldsymbol{\theta}_\ell) dt \backsimeq \frac{1}{L} \sum_{\ell=1}^{L} y(\boldsymbol{x}, \boldsymbol{\theta}_\ell)\,. \tag{10}$$

For the estimation of the variance, i.e., uncertainty of the prediction of $\boldsymbol{x}$, we need to distinguish between two types of uncertainties, the aleatoric and the epistemic uncertainties. The former derives from the generation of data in (6). Since all the training points in $\mathcal{D}$ contain a realization of the noise $\boldsymbol{\varepsilon}(\mathbf{x})$, this uncertainty is intrinsic within the data and it cannot be reduced by providing more training samples. Still, since it is data-dependent, it can be learned by the NN through a specific loss function with the model $\sigma_{\boldsymbol{\varepsilon}}(\mathbf{x})^2 = y_{\mathrm{al}}(\mathbf{x}, \boldsymbol{\theta}) + \xi_{\mathrm{al}}$ [59], where $y_{\mathrm{al}}(\boldsymbol{x}, \boldsymbol{\theta})$ is an additional NN output which predicts the aleatoric uncertainty of $\boldsymbol{x}$, and $\xi_{\mathrm{al}} \sim \mathcal{N}(0, {\sigma_{\xi_{\mathrm{al}}}}^2)$.

On the contrary, the epistemic uncertainty derives from the uncertainty over the NN parameters, i.e., random variables $\boldsymbol{\theta}$, which contributes to the uncertainty measure in output. In conventional NN, given their point estimate, this uncertainty is zero, while in BNN it can be explained away by providing more training data [42]. In our setting, the total variance predicted by the BNN can

be written as

$$\begin{aligned}\mathbb{V}\{\mathsf{t}|\mathbf{x},\mathsf{D}\} &\backsimeq \frac{1}{L}\sum_{\ell=1}^{L}\int \big(t-\mathbb{E}\{\mathsf{t}|\mathbf{x},\mathsf{D}\}\big)^2 p(t|\boldsymbol{x},\boldsymbol{\theta}_\ell)dt \\ &\backsimeq \frac{1}{L}\sum_{\ell=1}^{L} y(\boldsymbol{x},\boldsymbol{\theta}_\ell)^2 - \left(\frac{1}{L}\sum_{\ell=1}^{L} y(\boldsymbol{x},\boldsymbol{\theta}_\ell)\right)^2 \\ &\quad + \frac{1}{L}\sum_{\ell=1}^{L} y_{\mathrm{al}}(\boldsymbol{x},\boldsymbol{\theta}_\ell) \end{aligned} \tag{11}$$

where the first two terms of (11) represent the epistemic uncertainty prediction, while the last term is the aleatoric uncertainty prediction.

C. Real-Time BNN

While BNNs offer the significant advantage of quantifying uncertainty in NN predictions, which is fundamental in applications for critical scenarios, the requirement to perform multiple inferences for total variance estimation (i.e., sample averaging in (11)) remains a substantial drawback. This can be resolved using non-Bayesian NNs trained to learn the aleatoric uncertainty as described in [59]. However, a second issue, which arises from the usage of NNs for uncertainty estimation, is the inability of predicting the epistemic uncertainty, that is, the incapacity of distinguishing between epistemic and aleatoric uncertainty. This calls for a BNN method that performs real-time inference and, simultaneously it is able to distinguish between epistemic and aleatoric uncertainty.

The first issue is present in all conventional BNN approaches such as VI, e.g., MC-Dropout [67] and Bayes by backpropagation (BBP) [66], which sample the parameters from an approximation of the posterior $p_{\boldsymbol{\theta}|\mathsf{D}}(\boldsymbol{\theta}|\mathcal{D})$, and MCMC methods, e.g., stochastic gradient Langevin dynamics (SGLD) [62], which directly sample from the real posterior. Moreover, the MCMC methods require storing all the sample parameters $\boldsymbol{\theta}_\ell$, which may not be feasible during deployment. A class of BNN which is able to perform real-time inference are the so-called teacher-student methods, e.g., BDK [68], where a conventional NN, i.e., student NN, is trained to approximate the behaviour, i.e., the output, of a teacher BNN. The term *dark knowledge* in BDK was introduced to denote the hidden information within the teacher network that can subsequently be transferred to the student. Considering the BDK teacher (T)-student (S) method, the model between input and output now becomes

$$\begin{aligned}&(\mathrm{T}): \quad \mathsf{t} = y^{(\mathrm{T})}(\mathbf{x},\boldsymbol{\theta}) + \boldsymbol{\varepsilon}^{(\mathrm{T})} \\ &(\mathrm{S}): \quad \begin{cases} \mathsf{t} & = y^{(\mathrm{S})}(\mathbf{x},\boldsymbol{w}) + \boldsymbol{\varepsilon}^{(\mathrm{S})}(\mathbf{x}) \\ \boldsymbol{\sigma}_{\boldsymbol{\varepsilon}^{(\mathrm{S})}}(\mathbf{x})^2 & = y_{\mathrm{al}}^{(\mathrm{S})}(\mathbf{x},\boldsymbol{w}) + \boldsymbol{\xi}_{\mathrm{al}}^{(\mathrm{S})} \end{cases} \end{aligned} \tag{12}$$

where $\boldsymbol{w}$ are the deterministic parameters of the student, $\boldsymbol{\varepsilon}^{(\mathrm{T})} \sim \mathcal{N}(0,{\sigma_{\boldsymbol{\varepsilon}^{(\mathrm{T})}}}^2)$, $\boldsymbol{\varepsilon}^{(\mathrm{S})}(\mathbf{x}) \sim \mathcal{N}(0,\sigma_{\boldsymbol{\varepsilon}^{(\mathrm{S})}}(\mathbf{x})^2)$ and $\boldsymbol{\xi}_{\mathrm{al}}^{(\mathrm{S})} \sim \mathcal{N}(0,{\sigma_{\boldsymbol{\xi}_{\mathrm{al}}^{(\mathrm{S})}}}^2)$. Note that the parameters of the teacher $\hat{\boldsymbol{\theta}}$ are stochastic, whereas the parameters of the student $\boldsymbol{w}$ are deterministic. During training, the teacher is trained as a regular BNN, while the student learns the output of the teacher using a Kullback-Leibler (KL) divergence loss function as [68]

$$\begin{aligned} J(\boldsymbol{w}|\boldsymbol{x}) &= \mathrm{KL}\big(p(t|\boldsymbol{x},\mathcal{D}) \,\|\, p(t|\boldsymbol{x},\boldsymbol{w})\big) \\ &= \int p(t|\boldsymbol{x},\mathcal{D}) \log \frac{p(t|\boldsymbol{x},\mathcal{D})}{p(t|\boldsymbol{x},\boldsymbol{w})} dt \\ &\backsimeq -\frac{1}{L}\sum_{\ell=1}^{L} \mathbb{E}_{p(t|\boldsymbol{x},\boldsymbol{\theta}_\ell)}\big\{\log p(t|\boldsymbol{x},\boldsymbol{w})\big\}. \end{aligned} \tag{13}$$

On the contrary, during inference, we discard the teacher model and we just keep $y^{(\mathrm{S})}(\boldsymbol{x},\boldsymbol{w})$ as the predictive mean and $y_{\mathrm{al}}^{(\mathrm{S})}(\boldsymbol{x},\boldsymbol{w})$ as the predictive variance. Since the efficacy of this method relies on the teacher training, usually a MCMC method, such as SGLD, is employed [68].

D. Proposed Bayesian Bright Knowledge Method

One drawback of BDK is that the student does not have any knowledge of the epistemic uncertainty of the teacher, since it only outputs the aleatoric uncertainty through $y_{\mathrm{al}}^{(\mathrm{S})}(\boldsymbol{x},\boldsymbol{w})$. Distinguishing between aleatoric and epistemic uncertainties is crucial for several reasons. Firstly, it sheds light on the reasons behind a DL model's uncertainty regarding a particular test sample, which might be due to either insufficient training data or the inherent noise present in the data. Secondly, pinpointing the origin of uncertainty facilitates effective data acquisition by guiding the process towards where collecting additional training samples would be most beneficial. Lastly, considering both kinds of uncertainties allows for a thorough assessment of the total uncertainty associated with a prediction. Another drawback of BDK is that the loss function for regression derived from (13) is such that the student is trained only with one teacher-parameter sample $\boldsymbol{\theta}_\ell$ at the time [68]. In other words, in BDK, the teacher and student are trained sequentially using a stochastic version of (13) where a single parameter sample $\boldsymbol{\theta}_\ell$ is used at each step. On the contrary, we aim at exploiting the full KL loss in (13) obtained by averaging the teacher's output over the L samples.

In order to solve these issues, we propose to employ a student NN that predicts not only the aleatoric uncertainty, but also the epistemic uncertainty. While on the one hand a stand-alone NN is not able to output the uncertainty of the prediction since the weights are deterministic, on the other hand, by approximating the epistemic uncertainty of an existing teacher BNN, we are able to fully capture and distinguish between uncertainties on the data and on the parameters. Since all the information is transferred from the teacher to the student, we call this strategy Bayesian bright knowledge (BBK). The proposed model is

$$\begin{aligned}&(\mathrm{T}): \quad \mathsf{t} = y^{(\mathrm{T})}(\mathbf{x},\boldsymbol{\theta}) + \boldsymbol{\varepsilon}^{(\mathrm{T})} \\ &(\mathrm{S}): \quad \begin{cases} \mathsf{t} & = y^{(\mathrm{S})}(\mathbf{x},\boldsymbol{w}) + \boldsymbol{\varepsilon}^{(\mathrm{S})}(\mathbf{x}) \\ \boldsymbol{\sigma}_{\boldsymbol{\varepsilon}^{(\mathrm{S})}}(\mathbf{x})^2 & = y_{\mathrm{al}}^{(\mathrm{S})}(\mathbf{x},\boldsymbol{w}) + \boldsymbol{\xi}_{\mathrm{al}}^{(\mathrm{S})} \\ \mathbb{V}\{\mathsf{t}|\mathbf{x},\mathsf{D},\boldsymbol{\varepsilon}^{(\mathrm{T})}\} & = y_{\mathrm{ep}}^{(\mathrm{S})}(\mathbf{x},\boldsymbol{w}) + \boldsymbol{\xi}_{\mathrm{ep}}^{(\mathrm{S})} \end{cases} \end{aligned} \tag{14}$$

where $\boldsymbol{\xi}_{\mathrm{ep}}^{(\mathrm{S})} \sim \mathcal{N}(0,{\sigma_{\boldsymbol{\xi}_{\mathrm{ep}}^{(\mathrm{S})}}}^2)$. Note that the student has three outputs, $y^{(\mathrm{S})}(\mathbf{x},\boldsymbol{w})$ for target location prediction, $y_{\mathrm{al}}^{(\mathrm{S})}(\mathbf{x},\boldsymbol{w})$ for aleatoric uncertainty prediction and $y_{\mathrm{ep}}^{(\mathrm{S})}(\mathbf{x},\boldsymbol{w})$ for epistemic

uncertainty prediction. The total predictive variance is therefore $y_{\text{al}}^{(\text{S})}(\boldsymbol{x},\boldsymbol{w}) + y_{\text{ep}}^{(\text{S})}(\boldsymbol{x},\boldsymbol{w})$.

Defining with M the number of samples per mini-batch $\mathcal{M}^{(\text{T})}$ and N_{epochs} the number of training epochs, at each step $i = \{1, 2, \ldots, N_{\text{iter}} = N_{\text{epochs}} \cdot \lfloor N/M \rfloor\}$, the teacher is trained with a SGLD step as

$$\Delta\boldsymbol{\theta}_{i+1} = \frac{\eta_i^{(\text{T})}}{2}\Big(\nabla_{\boldsymbol{\theta}} \log p(\boldsymbol{\theta}_i) + \frac{N}{M}\sum_{m\in\mathcal{M}^{(\text{T})}} \nabla_{\boldsymbol{\theta}} \log p(t_m|\boldsymbol{x}_m,\boldsymbol{\theta}_i)\Big) + \boldsymbol{z}_i \tag{15}$$

where $\Delta\boldsymbol{\theta}_{i+1} = \boldsymbol{\theta}_{i+1} - \boldsymbol{\theta}_i$, $\eta_i^{(\text{T})}$ is the teacher learning rate at step i, and $\boldsymbol{z}_i$ is a noise sample from $\mathcal{N}(\boldsymbol{0}, \eta_i^{(\text{T})}\boldsymbol{I}_{|\boldsymbol{\theta}|})$. Following standard SGLD initialization, the prior $p(\boldsymbol{\theta}_i)$ is chosen to be spherical Gaussian as $p(\boldsymbol{\theta}_i) = \mathcal{N}(\boldsymbol{\theta}_i; \boldsymbol{0}, \lambda^{(\text{T})}\boldsymbol{I}_{|\boldsymbol{\theta}|})$, where $\lambda^{(\text{T})}$ is the L2 regularizer. For the student, we first need to define the loss function and then a practical training procedure. Starting with the loss function, for an input sample $\boldsymbol{x}$, we propose a two blocks function $J(\boldsymbol{w}|\boldsymbol{x}) = A(\boldsymbol{w}|\boldsymbol{x}) + B(\boldsymbol{w}|\boldsymbol{x})$. The first term $A(\boldsymbol{w}|\boldsymbol{x})$ is the same original loss function (13) in BDK and induces the student to learn the target variable t and the aleatoric uncertainty $\sigma_{\boldsymbol{\epsilon}^{(\text{S})}}(\boldsymbol{x})^2$. It can be shown that for regression, the block $A(\boldsymbol{w}|\boldsymbol{x})$ can be approximated by (see Appendix A)

$$A(\boldsymbol{w}|\boldsymbol{x}) \backsimeq \frac{1}{L}\sum_{\ell=1}^{L}\Bigg[\log\Big(y_{\text{al}}^{(\text{S})}(\boldsymbol{x},\boldsymbol{w})\Big) + y_{\text{al}}^{(\text{S})}(\boldsymbol{x},\boldsymbol{w})^{-1} \times \Big({\sigma_{\epsilon^{(\text{T})}}}^2 + \Big\|y^{(\text{T})}(\boldsymbol{x},\boldsymbol{\theta}_\ell) - y^{(\text{S})}(\boldsymbol{x},\boldsymbol{w})\Big\|_2^2\Big)\Bigg]. \tag{16}$$

On the contrary, the second term $B(\boldsymbol{w}|\boldsymbol{x})$ forces the student to learn the epistemic uncertainty coming from the teacher and it is obtained via standard MLE as:

$$B(\boldsymbol{w}|\boldsymbol{x}) \backsimeq \frac{1}{2{\sigma_{\boldsymbol{\xi}_{\text{ep}}^{(\text{S})}}}^2}\Bigg\|\frac{1}{L}\sum_{\ell=1}^{L} y^{(\text{T})}(\boldsymbol{x},\boldsymbol{\theta}_\ell)^2 - \Big(\frac{1}{L}\sum_{\ell=1}^{L} y^{(\text{T})}(\boldsymbol{x},\boldsymbol{\theta}_\ell)\Big)^2 - y_{\text{ep}}^{(\text{S})}(\boldsymbol{x},\boldsymbol{w})\Bigg\|_2^2. \tag{17}$$

We refer to Appendix B for a derivation of the block $B(\boldsymbol{w}|\boldsymbol{x})$. It is important to note that calculating $J(\boldsymbol{w}|\boldsymbol{x})$ requires L predictions from the teacher. However, we aim to bypass the storage of L samples of $\boldsymbol{\theta}_\ell$ during training. To address this, we propose a training method that retains the samples $\boldsymbol{x}_m$ and target t_m across L forward-backward steps, subsequently updating the student's parameters as

$$\Delta\boldsymbol{w}_{i+1} = -\frac{\eta_i^{(\text{S})}}{2}\Big(\frac{1}{M}\sum_{m\in\mathcal{M}_{i-L}^{(\text{S})}} \nabla_{\boldsymbol{w}} J(\boldsymbol{w}_i|\boldsymbol{x}_m) + \lambda^{(\text{S})}\boldsymbol{w}_i\Big) \tag{18}$$

where $\Delta\boldsymbol{w}_{i+1} = \boldsymbol{w}_{i+1} - \boldsymbol{w}_i$, $\eta_i^{(\text{T})}$, $\eta_i^{(\text{S})}$ is the student learning rate at step i, $\mathcal{M}_i^{(\text{S})}$ is the mini-batch of the student at step i and $\lambda^{(\text{S})}$ is an L2 regularizer hyper-parameter of the student. Algorithm 1 describes the full training procedure, which takes as input the training teacher and student datasets $\mathcal{D}^{(\text{T})}$ and $\mathcal{D}_{\boldsymbol{x}}^{(\text{S})}$, respectively, and outputs the student parameters $\boldsymbol{w}$. We point out that the student dataset does not contain any target values since the teacher output is adopted as a target. This gives the flexibility to exploit unsupervised, i.e., unlabelled, datasets for training the student in input locations where we are interested in having reliable uncertainty metrics.

Algorithm 1 Training Procedure

Input: Training datasets $\mathcal{D}^{(\text{T})}$ and $\mathcal{D}_{\boldsymbol{x}}^{(\text{S})}$
Output: Student parameters $\boldsymbol{w}$

Initialize Teacher and Student parameters $\boldsymbol{\theta}_1$ and $\boldsymbol{w}_1$
Initialize $\mathcal{D}_{\boldsymbol{x},\text{old}}^{(\text{S})} \leftarrow \emptyset$ and $\mathcal{D}_{t,\text{old}}^{(\text{S})} \leftarrow \emptyset$
Set $N_{\text{iter}} \leftarrow N_{\text{epochs}} \cdot \lfloor N/M \rfloor$
for $i = \{1, \ldots, N_{\text{iter}}\}$ **do** ▷ Batch-wise iteration
  Sample minibatch $\mathcal{M}_i^{(\text{T})}$ of size M from $\mathcal{D}^{(\text{T})}$
  Sample $\boldsymbol{z}_i \sim \mathcal{N}(0, \eta_i^{(\text{T})}\boldsymbol{I}_{|\boldsymbol{\theta}|})$
  Update Teacher using (15)
  Sample minibatch $\mathcal{M}_i^{(\text{S})}$ of size M from $\mathcal{D}_{\boldsymbol{x}}^{(\text{S})}$
  $\mathcal{D}_{\boldsymbol{x},\text{old}}^{(\text{S})} \leftarrow \mathcal{D}_{\boldsymbol{x},\text{old}}^{(\text{S})} \cup \{\boldsymbol{x}_m\}_{m\in\mathcal{M}_i^{(\text{S})}}$
  $\mathcal{D}_{t,\text{old}}^{(\text{S})} \leftarrow \mathcal{D}_{t,\text{old}}^{(\text{S})} \cup \{y^{(\text{T})}(\boldsymbol{x},\boldsymbol{\theta}_{i+1})\}_{\boldsymbol{x}\in\mathcal{D}_{\boldsymbol{x},\text{old}}^{(\text{S})}}$
  if $i > L$ **then**
    $\mathcal{D}_{\boldsymbol{x},\text{old}}^{(\text{S})} \leftarrow \mathcal{D}_{\boldsymbol{x},\text{old}}^{(\text{S})} \backslash \{\boldsymbol{x}_m\}_{m\in\mathcal{M}_{i-L}^{(\text{S})}}$
    $\mathcal{D}_{t,\text{old}}^{(\text{S})} \leftarrow \mathcal{D}_{t,\text{old}}^{(\text{S})} \backslash \{y^{(\text{T})}(\boldsymbol{x}_m,\boldsymbol{\theta}_j)\}_{\substack{m\in\mathcal{M}_{i-L}^{(\text{S})},\\ j=\{i-L,\ldots,i\}}}$
    Update Student using (18)
  end if
end for

E. Multi-Dimensional Target Variable

In case the target variable is multi-dimensional, such as for 3D location, we have that $\mathbb{E}\{\mathbf{t}|\mathbf{x},\mathsf{D}\} \in \mathbb{R}^{|\mathbf{t}|\times 1}$ and $\mathbb{V}\{\mathbf{t}|\mathbf{x},\mathsf{D}\} \in \mathbb{R}^{|\mathbf{t}|\times|\mathbf{t}|}$. Therefore, the new teacher-student model becomes

$$\begin{aligned}
&(\text{T}): \quad \mathbf{t} = \boldsymbol{y}^{(\text{T})}(\mathbf{x},\boldsymbol{\theta}) + \boldsymbol{\epsilon}^{(\text{T})} \\
&(\text{S}): \begin{cases} \mathbf{t} & = \boldsymbol{y}^{(\text{S})}(\mathbf{x},\boldsymbol{w}) + \boldsymbol{\epsilon}^{(\text{S})}(\mathbf{x}) \\ \boldsymbol{\Sigma}_{\boldsymbol{\epsilon}^{(\text{S})}}(\mathbf{x}) & = R_{\substack{|\mathbf{t}|^2\times 1\\ |\mathbf{t}|\times|\mathbf{t}|}}\big(\boldsymbol{y}_{\text{al}}^{(\text{S})}(\mathbf{x},\boldsymbol{w})\big) + \boldsymbol{\xi}_{\text{al}}^{(\text{S})} \\ \mathbb{V}\Big\{\mathbf{t}|\mathbf{x},\mathsf{D},\boldsymbol{\epsilon}^{(\text{T})}\Big\} & = R_{\substack{|\mathbf{t}|^2\times 1\\ |\mathbf{t}|\times|\mathbf{t}|}}\big(\boldsymbol{y}_{\text{ep}}^{(\text{S})}(\mathbf{x},\boldsymbol{w})\big) + \boldsymbol{\xi}_{\text{ep}}^{(\text{S})} \end{cases}
\end{aligned} \tag{19}$$

where $R_{\substack{p\times q\\ n\times m}} : \mathbb{R}^{p\times q} \to \mathbb{R}^{n\times m}$ indicates the reshape operation, $\boldsymbol{\epsilon}^{(\text{T})} \sim \mathcal{N}(\boldsymbol{0}, \boldsymbol{\Sigma}_{\boldsymbol{\epsilon}^{(\text{T})}})$, $\boldsymbol{\epsilon}^{(\text{S})}(\mathbf{x}) \sim \mathcal{N}(\boldsymbol{0}, \boldsymbol{\Sigma}_{\boldsymbol{\epsilon}^{(\text{S})}}(\mathbf{x}))$, $\boldsymbol{\xi}_{\text{al}}^{(\text{S})} \sim \mathcal{N}(\boldsymbol{0}, \boldsymbol{\Sigma}_{\boldsymbol{\xi}_{\text{al}}^{(\text{S})}})$ and $\boldsymbol{\xi}_{\text{ep}}^{(\text{S})} \sim \mathcal{N}(\boldsymbol{0}, \boldsymbol{\Sigma}_{\boldsymbol{\xi}_{\text{ep}}^{(\text{S})}})$. Consequently, the $A(\boldsymbol{w}|\boldsymbol{x})$ term in (16) in matrix form becomes

$$\begin{aligned}
A(\boldsymbol{w}|\boldsymbol{x}) \backsimeq \frac{1}{L}\sum_{\ell=1}^{L}\Bigg[&\frac{1}{2}\log\Big(\det R_{\substack{|\mathbf{t}|^2\times 1\\ |\mathbf{t}|\times|\mathbf{t}|}}\big(\boldsymbol{y}_{\text{al}}^{(\text{S})}(\boldsymbol{x},\boldsymbol{w})\big)\Big) \\
&+\frac{1}{2}\big(\boldsymbol{y}^{(\text{T})}(\boldsymbol{x},\boldsymbol{\theta}_\ell) - \boldsymbol{y}^{(\text{S})}(\boldsymbol{x},\boldsymbol{w})\big)^\top \\
&\times R_{\substack{|\mathbf{t}|^2\times 1\\ |\mathbf{t}|\times|\mathbf{t}|}}\big(\boldsymbol{y}_{\text{al}}^{(\text{S})}(\boldsymbol{x},\boldsymbol{w})\big)^{-1}\big(\boldsymbol{y}^{(\text{T})}(\boldsymbol{x},\boldsymbol{\theta}_\ell) - \boldsymbol{y}^{(\text{S})}(\boldsymbol{x},\boldsymbol{w})\big) \\
&+\text{Tr}\Big(\boldsymbol{\Sigma}_{\boldsymbol{\epsilon}^{(\text{T})}} R_{\substack{|\mathbf{t}|^2\times 1\\ |\mathbf{t}|\times|\mathbf{t}|}}\big(\boldsymbol{y}_{\text{al}}^{(\text{S})}(\boldsymbol{x},\boldsymbol{w})\big)^{-1}\Big)\Bigg].
\end{aligned} \tag{20}$$

The $B(\boldsymbol{w}|\boldsymbol{x})$ term in (17) becomes

$$B(\boldsymbol{w}|\boldsymbol{x}) \backsimeq \frac{1}{2}\Big(\boldsymbol{y}_{\text{ep}}^{(\text{T})}(\boldsymbol{x}) - R_{\substack{|\mathbf{t}|^2\times 1\\ |\mathbf{t}|\times|\mathbf{t}|}}\big(\boldsymbol{y}_{\text{ep}}^{(\text{S})}(\boldsymbol{x},\boldsymbol{w})\big)\Big)^{\top} \times {\boldsymbol{\Sigma}_{\boldsymbol{\xi}_{\text{ep}}^{(\text{S})}}}^{-1}\Big(\boldsymbol{y}_{\text{ep}}^{(\text{T})}(\boldsymbol{x}) - R_{\substack{|\mathbf{t}|^2\times 1\\ |\mathbf{t}|\times|\mathbf{t}|}}\big(\boldsymbol{y}_{\text{ep}}^{(\text{S})}(\boldsymbol{x},\boldsymbol{w})\big)\Big) \tag{21}$$

where the predictive epistemic uncertainty of the teacher is

$$\boldsymbol{y}_{\text{ep}}^{(\text{T})}(\boldsymbol{x}) = \frac{1}{L}\sum_{\ell=1}^{L}\Big(\boldsymbol{y}^{(\text{T})}(\boldsymbol{x},\boldsymbol{\theta}_\ell)^{\top}\boldsymbol{y}^{(\text{T})}(\boldsymbol{x},\boldsymbol{\theta}_\ell)\Big) - \Big(\frac{1}{L}\sum_{\ell=1}^{L}\boldsymbol{y}^{(\text{T})}(\boldsymbol{x},\boldsymbol{\theta}_\ell)\Big)^{\top}\Big(\frac{1}{L}\sum_{\ell=1}^{L}\boldsymbol{y}^{(\text{T})}(\boldsymbol{x},\boldsymbol{\theta}_\ell)\Big). \tag{22}$$

F. AE-Based DL Model

For the cellular positioning application of the developed BNN method, we propose using an AE stored inside each BS, as depicted in Fig. 2(b), to derive key location features from the sparse ADCPM samples $\boldsymbol{x}$. The encoder function $E(\boldsymbol{x})$ generates the latent features $\boldsymbol{z}$, which encompass location-centric data inherent in the channel. Conversely, the decoder function $D(\boldsymbol{z})$ attempts to recreate the input samples, resulting in $\hat{\boldsymbol{x}}$. The AE aims to minimize the reconstruction error metric $\|\boldsymbol{x}-\hat{\boldsymbol{x}}\|_2^2$ [75], enabling the model to reproduce the input $\boldsymbol{x}$ using the condensed data in $\boldsymbol{z}$. This ensures that $\boldsymbol{z}$ includes all crucial information needed for the single-BS positioning objective. The DL model also contains a multilayer perceptron (MLP) positioning module which takes as input the latent features and predicts the 3D target position. Note that it is not necessary to detect LOS and NLOS conditions since the MLP is combined with other BSs outputs through the posterior predictive distribution. In other words, the positioning module's output is a *soft information* [2], [12], [27], i.e., a distribution, which is coherently combined for tracking applications.

In order to apply the BBK method, we created two identical DL models, one for the teacher and one for the student. In the teacher network, we treated the AE structure as a normal NN, whereas the positioning module is trained as a full BNN with SGLD optimizer. This is mainly done for faster convergence reasons and because unsupervised samples, i.e., CIR samples without position correspondence, can be easily gathered in every position, thereby solving the OOD problem. On the contrary, the student is trained as a conventional NN with Adam optimizer [76] and the loss function

$$J(\boldsymbol{w}|\boldsymbol{x}) = \lambda_{\text{pos}}\big(A(\boldsymbol{w}|\boldsymbol{x}) + B(\boldsymbol{w}|\boldsymbol{x})\big) + \lambda_{\text{rec}}\|\boldsymbol{x}-\hat{\boldsymbol{x}}\|_2^2 \tag{23}$$

where λ_{rec} regulates the sample reconstruction and λ_{pos} controls the position estimation relevance. Generally, the relation between these two hyper-parameters is $\lambda_{\text{rec}} < \lambda_{\text{pos}}$. This is because firstly, the AE model exhibits greater complexity compared to the MLPs used for positioning, resulting in a rapid decrease in reconstruction error. Secondly, the feature count in $\boldsymbol{x}$ surpasses the dimension of $\boldsymbol{t}$. This inherently amplifies the significance of the reconstruction error relative to the positioning error.

V. Location Tracking With BNN Methodology

In this section, we first introduce the general problem of sequential Bayesian tracking, and then we propose a solution that integrates the BNN method.

A. Tracking Problem

We consider the non-linear Bayesian tracking problem of a target whose state evolves according to the motion model [77]

$$\mathbf{t}_n = f_n^{(\text{t})}(\mathbf{t}_{n-1}) + \boldsymbol{\epsilon}_{n-1}^{(\text{t})} \tag{24}$$

where $f_n^{(\text{t})}(\mathbf{t}_{n-1})$ is a non-linear function of the state $\mathbf{t}_{n-1}$ at time $n-1$ and $\boldsymbol{\epsilon}_{n-1}^{(\text{t})}$ is a non-independent and identical distributed (non-IID) noise sequence. From (24), we define the corresponding transition PDF $p(\boldsymbol{t}_n|\boldsymbol{t}_{n-1})$. The network positioning system has measurements of the target modeled as

$$\mathbf{x}_n = f_n^{(\text{x})}(\mathbf{t}_n) + \boldsymbol{\epsilon}_n^{(\text{x})} \tag{25}$$

where $f_n^{(\text{x})}(\mathbf{t}_n)$ is a non-linear function which relates the state and the measurement, and $\boldsymbol{\epsilon}_n^{(\text{x})}$ is a non-IID noise sequence. Similarly to before, from (25), we can define a likelihood function $p(\boldsymbol{x}_n|\boldsymbol{t}_n)$. Moreover, we define with $\mathbf{x}_{1:n} = \{\mathbf{x}_i, i=1,\dots,n\}$ the set of all available measurements up to time n. Since (24) and (25) are 1st order hidden Markov model (HMM), we can write that $p(\boldsymbol{t}_n|\boldsymbol{t}_{n-1},\boldsymbol{x}_{1:n-1}) = p(\boldsymbol{t}_n|\boldsymbol{t}_{n-1})$ and express the usual prediction phase of the state through the Chapman-Kolmogorov equation

$$p(\boldsymbol{t}_n|\boldsymbol{x}_{1:n-1}) = \int p(\boldsymbol{t}_n|\boldsymbol{t}_{n-1})\, p(\boldsymbol{t}_{n-1}|\boldsymbol{x}_{1:n-1})\, d\boldsymbol{t}_{n-1} \tag{26}$$

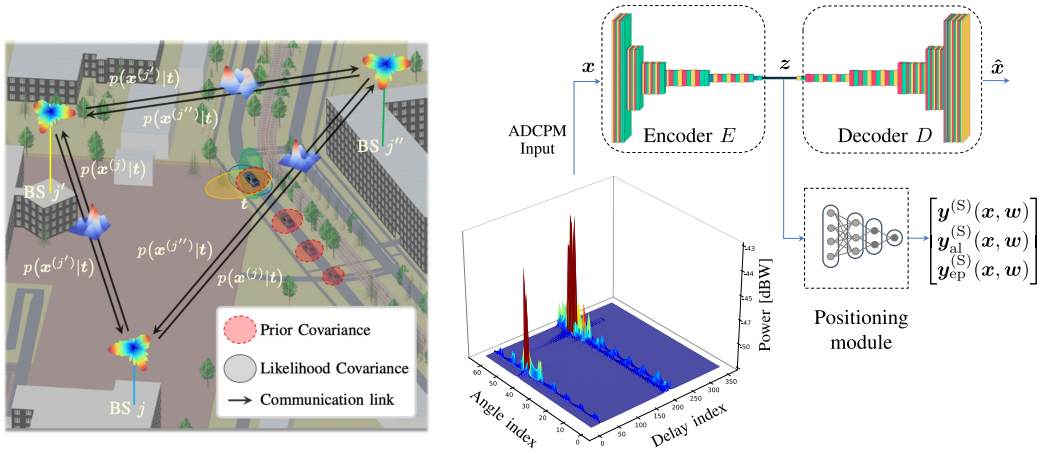
where $p(\boldsymbol{t}_{n-1}|\boldsymbol{x}_{1:n-1})$ is the posterior PDF at time $n-1$ and $p(\boldsymbol{t}_n|\boldsymbol{x}_{1:n-1})$ is the prior PDF at time n. Subsequently, the measurements are taken into account in the update phase which recovers the posterior at time n:

$$p(\boldsymbol{t}_n|\boldsymbol{x}_{1:n}) \propto p(\boldsymbol{x}_n|\boldsymbol{t}_n)\, p(\boldsymbol{t}_n|\boldsymbol{x}_{1:n-1})\,. \tag{27}$$

The EKF, or more complex conventional filters [28], implement the steps (26) and (27) by assuming explicit (and known) parametric models for functions $f_n^{(\text{t})}(\mathbf{t}_{n-1})$ and $f_n^{(\text{x})}(\mathbf{t}_n)$. This is not viable in mixed LOS/NLOS conditions and with complex measurement as ADCPM, which relate to the location through an unknown non-linear and site-dependent function $f_n^{(\text{x})}(\mathbf{t}_n)$. We thus propose to integrate a BNN into the Bayesian filter and learn such function from data, as detailed in the following section.

B. Proposed BNN for Tracking

For the integration of BNN in the tracking system, we consider a set of BSs $\mathcal{S}_{\text{BS}}$. During the offline training phase, each BS j trains its own BNN using a local training dataset $\mathcal{D}^{(j)} = \big\{\boldsymbol{t}_n^{(j)}, \boldsymbol{x}_n^{(j)}\big\}_{n=1}^{N^{(j)}}$. We point out that in this phase, any BNN algorithm and layer structure could be employed and assessed during an ad-hoc testing static positioning. Subsequently, in the online tracking phase, a set of BSs $\mathcal{S}_{\text{BS},n} \subseteq \mathcal{S}_{\text{BS}}$ detects a target at timestep n and obtains a set of samples $\boldsymbol{X}_n = \big\{\boldsymbol{x}_n^{(j)}\big\}_{j\in\mathcal{S}_{\text{BS},n}}$. The main idea is to leave unaltered the



(a) (b)

Fig. 2. (a) Cooperative tracking system composed of three BSs. The timestep index n has been dropped for simplicity of notation. (b) DL model composed of an AE structure and a positioning module. The input is the sparse ADCPM fingerprint, whereas the outputs are the reconstructed $\hat{\boldsymbol{x}}$, the target estimation $\boldsymbol{y}^{(\mathrm{S})}(\boldsymbol{x},\boldsymbol{w})$, the aleatoric uncertainty $\boldsymbol{y}_{\mathrm{al}}^{(\mathrm{S})}(\boldsymbol{x},\boldsymbol{w})$ and the epistemic uncertainty $\boldsymbol{y}_{\mathrm{ep}}^{(\mathrm{S})}(\boldsymbol{x},\boldsymbol{w})$.

prediction phase (where the dynamics function $f_n^{(\mathrm{t})}(\mathbf{t}_{n-1})$ is easier to model) and focus on the tight integration of BNN in the update phase. This is done to facilitate the integration with already existing algorithms since we can just replace or even augment the update part with the BNN as described in the following sections. Moreover, while training of a NN with static measurements is both affordable and precise, replacing the prediction phase with dynamic models (e.g., LSTMs) introduces complexities. One significant challenge is the data collection for target trajectories, which necessitates the gathering of ground truth data, i.e., a reliable benchmark to precisely measure the target's exact trajectory.

After performing the prediction phase and obtaining the prior distribution $p(\boldsymbol{t}_n|\boldsymbol{x}_{1:n-1})$, each BS $j \in \mathcal{S}_{\mathrm{BS},n}$ outputs the posterior predictive distribution $p\big(\boldsymbol{t}_n|\boldsymbol{x}_n^{(j)},\mathcal{D}^{(j)}\big)$, which we indicate with $p\big(\boldsymbol{t}_n|\boldsymbol{x}_n^{(j)}\big)$. Now, we note that the posterior in (27) resembles the posterior predictive distribution in (8), but they are fundamentally different. Indeed, while the former is the result of prior knowledge coming from the tracking, the latter does not have any knowledge on the sequentiality of the target state since the BNN network has just been trained with input-output samples. Therefore, at each timestep n, the BNN does not have any prior knowledge on where the target was at previous time $n-1$, formally, $\mathbf{t}_n \sim \mathcal{U}\big(\boldsymbol{t}_{\min}^{(j)},\boldsymbol{t}_{\max}^{(j)}\big)$, where $\boldsymbol{t}_{\min}^{(j)}$ and $\boldsymbol{t}_{\max}^{(j)}$ are the limits of the coverage area of the j-th BS. This is similar to what happens during the computation of the likelihood function $p(\boldsymbol{x}_n|\boldsymbol{t}_n)$ where the measurement model in (25) is considered without previous time-dependence. By analogy and considering the uniform distribution of $\mathbf{t}_n$, we can write $p\big(\boldsymbol{x}_n^{(j)}|\boldsymbol{t}_n\big) \propto p\big(\boldsymbol{t}_n|\boldsymbol{x}_n^{(j)},\mathcal{D}\big)$. In this way, we combine the predictions of multiple BSs with the prior PDF on the target state and obtain the updated posterior. The full tracking procedure can be found in Algorithm 2.

Algorithm 2 Tracking Procedure

Input: Posterior $p(\boldsymbol{t}_{n-1}|\boldsymbol{x}_{1:n-1})$ at time $n-1$
Output: Posterior $p(\boldsymbol{t}_n|\boldsymbol{x}_{1:n})$ at time n
1: Compute prediction phase in (26)
2: Measure sample $\boldsymbol{x}_n^{(j)}$
3: Compute $p\big(\boldsymbol{x}_n^{(j)}|\boldsymbol{t}_n\big) \propto p\big(\boldsymbol{t}_n|\boldsymbol{x}_n^{(j)},\mathcal{D}\big)$
4: **for** $j' \in \mathcal{S}_{\mathrm{BS},n} \backslash \{j\}$ **do**
5: Send $p\big(\boldsymbol{x}_n^{(j)}|\boldsymbol{t}_n\big)$ to j'
6: Receive $p\big(\boldsymbol{x}_n^{(j')}|\boldsymbol{t}_n\big)$ from j'
7: **end for**
8: Update $p(\boldsymbol{t}_n|\boldsymbol{x}_{1:n}) \propto \prod_{j\in\mathcal{S}_{\mathrm{BS},n}} p\big(\boldsymbol{x}_n^{(j)}|\boldsymbol{t}_n\big)\, p(\boldsymbol{t}_n|\boldsymbol{x}_{1:n-1})$

An example of cooperative tracking performed by three BSs is shown in Fig. 2(a), where the exchange of the likelihood functions permits the reduction of the target position uncertainty (represented by the intersections of covariance areas in the figure).

For the scenario described in Sec. VI, the posterior predictive distribution is described by two parameters, i.e., the predictive mean (10) and the predictive variance (11). Therefore, we propose to approximate the likelihood function obtained by each BS with a multivariate normal distribution as

$$\begin{aligned} p\big(\boldsymbol{x}_n^{(j)}|\boldsymbol{t}_n\big) &\backsimeq \mathcal{N}\Big(\boldsymbol{x}_n^{(j)};\mathbb{E}\{\mathbf{t}_n|\mathbf{x}_n^{(j)},\mathsf{D}\},\mathbb{V}\{\mathbf{t}_n|\mathbf{x}_n^{(j)},\mathsf{D}\}\Big) \\ &= \mathcal{N}\Big(\boldsymbol{x}_n^{(j)};\boldsymbol{\mu}_n^{(j)},\boldsymbol{\Sigma}_n^{(j)}\Big). \end{aligned} \tag{28}$$

Note that for the real-time BBK, the predictive mean and variance are respectively

$$\boldsymbol{\mu}_n^{(j)} = \boldsymbol{y}^{(\mathrm{S})}\big(\boldsymbol{x}_n^{(j)},\boldsymbol{w}^{(j)}\big) \tag{29}$$

$$\boldsymbol{\Sigma}_n^{(j)} = R_{\substack{|\mathbf{t}|^2\times 1\\ |\mathbf{t}|\times|\mathbf{t}|}}\Big(\boldsymbol{y}_{\text{al}}^{(\text{S})}\big(\boldsymbol{x}_n^{(j)},\boldsymbol{w}^{(j)}\big)+\boldsymbol{y}_{\text{ep}}^{(\text{S})}\big(\boldsymbol{x}_n^{(j)},\boldsymbol{w}^{(j)}\big)\Big). \quad (30)$$

This approximation makes it very easy and effective to combine the likelihood functions of the BSs to be used in (27) as [78]

$$p(\boldsymbol{x}_n|\boldsymbol{t}_n) = \prod_{j\in\mathcal{S}_{\text{BS},n}} p\big(\boldsymbol{x}_n^{(j)}|\boldsymbol{t}_n\big) \propto \mathcal{N}\big(\boldsymbol{x}_n;\boldsymbol{\mu}_n,\boldsymbol{\Sigma}_n\big),$$

where

$$\boldsymbol{\mu}_n = \boldsymbol{\Sigma}_n\Big(\sum_{j\in\mathcal{S}_{\text{BS},n}} {\boldsymbol{\Sigma}_n^{(j)}}^{-1}\boldsymbol{\mu}_n^{(j)}\Big)$$

$$\boldsymbol{\Sigma}_n = \Big(\sum_{j\in\mathcal{S}_{\text{BS},n}} {\boldsymbol{\Sigma}_n^{(j)}}^{-1}\Big)^{-1}.$$

VI. Simulation Experiments

A. Simulation Results

To assess the performance of the proposed BNN-based tracking system, due to the unavailability of real-world experimental activities, we simulate a 5G positioning network based on the 5G new radio (NR) MATLAB clustered delay line (CDL) channel model, which can be defined over a bandwidth of 2 GHz in the frequency range from 0.5 GHz to 100 GHz [79]. The radio wave propagation is simulated using a ray-tracing method [80], [81], [82] from the Wireless InSite 3D prediction tool [83], which plots the propagation paths from the UE to the BSs based on the surface geometry from a 3D map file. The ray-based solver can manage up to fifty reflections and three diffractions, ensuring a realistic simulation of the effect of buildings and terrains on the radio signal propagation. In mmWave scenarios, the propagation model integrates atmospheric absorption and allows the inclusion of vegetation within the propagation setting, assessing the impact of diffuse scattering on the channel response and ensuring spatial consistency. The channel is then obtained by using the rays as mean clusters and by including: the Doppler shift, according to the UE mobility, a main LOS cluster (if the UE is in visibility) with K-factor equal to 13.3 dB, a number of sub-cluster per cluster equal to 2, and moving scatterers in the channel. The cluster-wise root mean square (RMS) angle spreads and delay spreads have been set to 3 degrees and 3.90625 ns, respectively.

The 3D map is obtained through Google Maps, RenderDoc, and Blender software with MapsModelImporter plugin. An example of the extracted 3D map can be found in Fig. 3, which represents both the 3D patches with known textures. For our experimental setup, we emulate a 3GPP urban micro (UMi) environment [69] spanning a 1000 $\times$ 1000 m square near the Massachusetts Institute of Technology (MIT) campus, Cambridge, MA 02139, USA. The setting encompasses 19 sites with an inter-site distance (ISD) of 200 m, arranged in a hexagonal pattern. Every site is composed of 3 cells, each at a height of 25 m and spaced 120 degrees apart in azimuth.

Each cellular antenna is equipped with an UPA setup with $N_{\text{h}} = N_{\text{v}} = 8$ antenna elements and a mechanical downtilt of 15 degrees. The antenna element details were derived from [84], ensuring a front-to-back ratio of approximately 30 dB and a peak gain reaching 8 dBi. The UE trajectories are generated by the SUMO software, which simulates realistic vehicular traffic throughout a given road grid, according to the interactions between vehicles, geometry of the map and speed limits. Over 600 s of simulation, we created up to 100 vehicle trajectories and we gathered data points every second. The absolute velocities of the vehicles span in $[0, 34]$ km/h, with a mean and standard deviation of 9.4 and 6.3 km/h, respectively. In total, we obtained 2593 and 702 training and testing positions, respectively, and about $9.3\cdot10^4$ and $2.5\cdot10^4$ training and testing ADCPM samples, respectively. The simulated testing trajectories with their absolute velocities, along with the BSs composing the UMi scenario, are shown in Fig. 4. On the contrary, the different traffic densities can be found in Fig. 8-top. In each position, every UE broadcasts 5G sounding reference signal (SRS) to all neighbor BSs using a carrier frequency $f_{\text{c}} = 28$ GHz and a transmission bandwidth $B = 400$ MHz. Then, each BS performs OFDM demodulation and obtains the SFCRM in (2) through pilot signals and least squares (LS) channel estimation. Finally, the ADCPM is computed using (3) and (4).

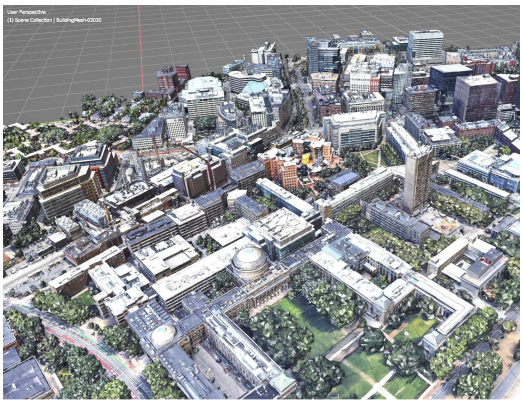


Fig. 3. Digital twin 3D map representation of the urban scenario around MIT campus.

B. Discussion on Practical Implementation

The tracking system is designed to separate the prediction phase in (26) and update phase in (27). Consequently, data acquisition can be performed statically at each position to train the BNN, whereas the motion model can be adjusted according to the dynamics of the UE. However, in case we want to speed up the process, we can adopt a vehicle or platoon of vehicles [85], [86] to record the timestamp and related noisy position with GNSS, which will then be mapped with the channel recording at the BSs. The uncertainty about the ground truth position is automatically learned by the aleatoric uncertainty prediction, whereas the density of the training points is taken into account by the epistemic uncertainty.

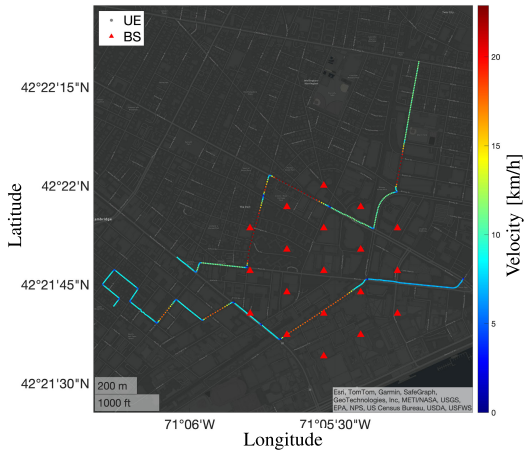


Fig. 4. Test trajectories with two vehicles in the area of Cambridge, MA, USA. The red triangles indicate the BS positions.

For the simulations, we did not consider multi-user interference (MUI) as the focus of the paper is to assess the best-case performances of the proposed BBK method and tracking system. However, when dealing with more than one user, the BSs can perform robust channel estimation methods, e.g., linear minimum mean-square error (LMMSE) or non-linear pre-coding schemes [87], [88], to reduce the impact of MUI. Other possible solutions include the usage of data association (DA) schemes to track and remove the multipath components of the interference [89].

C. DL Model Implementation

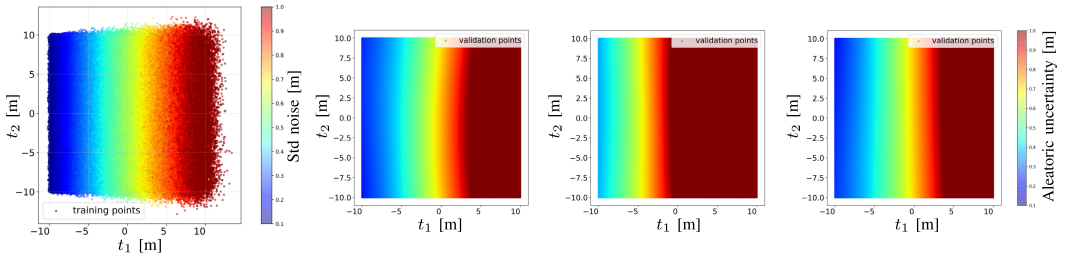
For the specific design of the DL positioning model, we design the AE part with the Segnet architecture [90]. This permits to manage the sparsity of the ADCPM input and perform robust feature extraction, which is pivotal for accurate positioning. Indeed, the upsampling layers utilize encoder pool indices for custom sparse feature mapping. During testing, the decoder segment is discarded, as input reconstruction is only needed for learning latent feature representations during training. Concerning the positioning module, given the natural regularization induced by the BNN teacher model, we inserted Gaussian error linear unit (GELU) activations functions after each linear layer. We performed an architecture search by doubling the number of neurons in each layer, until reaching both the latency requirements of 5 ms for fully autonomous driving vehicles [44] and a low bias in the performances. For the latter, we performed a similar procedure in [34] by testing the position module to perform localization with a synthetic dataset where the input latent features are substituted with geometric measurements (i.e., AOAs and TOFs). After multiple neural architecture searches, we set the number of neurons in each layer to: $[16, 32, 64, 128, 256, 512, 256, 128, 64, 32, 16, 9]$. For the prediction of the uncertainties, we placed softplus activation functions at the outputs of the positioning module which relates to the diagonal elements of $R_{\substack{|\mathbf{t}|^2\times 1\\ |\mathbf{t}|\times|\mathbf{t}|}}\left(\boldsymbol{y}_{\mathrm{al}}^{(\mathrm{S})}(\boldsymbol{x},\boldsymbol{w})\right)$ and $R_{\substack{|\mathbf{t}|^2\times 1\\ |\mathbf{t}|\times|\mathbf{t}|}}\left(\boldsymbol{y}_{\mathrm{ep}}^{(\mathrm{S})}(\boldsymbol{x},\boldsymbol{w})\right)$. This prevents the variances on the diagonal from being negative. Finally, in order to make the aleatoric and epistemic predictions valid covariances, we add a regularization term to the diagonal of each matrix prediction to ensure they are non-singular and enforce symmetry in each matrix by averaging them with their transpose.

To measure the time required by the system to perform inference, we report that the total number of floating point operations (FLOPs) required by the DL model are $27.4 \cdot 10^9$. Simulations were conducted on a workstation boasting an Intel(R) Xeon(R) Silver 4210R CPU @ 2.40 GHz, with 96 GB RAM and a Quadro RTX 6000 24 GB GPU. Since the GPU has a single-precision performance of $16.3 \cdot 10^{12}$ floating point operations per second (FLOPS), the inference time per sample can be estimated as $27.4 \cdot 10^9 / 16.3 \cdot 10^{12}\,\mathrm{s} = 1.6\,\mathrm{ms}$. Considering the delay required to exchange a packet comprising $\boldsymbol{\mu}_n^{(j)}$ and $\boldsymbol{\Sigma}_n^{(j)}$ between the two farthest BSs, i.e., less than 1 ms with fiber's length of 1 km and up to 80% of traffic load [91], we assume a latency of about 2 ms. In case of multiple targets, we can exploit the tensor operations of the GPU with $130.5 \cdot 10^{12}$ FLOPS.

All the experiments were performed using Pytorch [92] and, unless stated otherwise, models underwent training for 600 epochs with a batch size of $M = 256$ and the number of NN parameter samples $L = 40$. The learning rates of the teacher and student were set to $\eta_0^{(\mathrm{T})} = 10^{-5} < \eta_0^{(\mathrm{S})} = 10^{-4}$, in order to have a better convergence of the teacher (with lower learning rate) and, conversely, a faster convergence of the student. Regarding the hyper-parameters, λ_{rec} and λ_{pos} were empirically set using a grid method in the range $[0.1, 1]$ with a 0.1 step size and following the intuition described in Sec. IV-F, resulting in $\lambda_{\mathrm{rec}} = 0.1$ and $\lambda_{\mathrm{pos}} = 0.9$. For the L2 prior regularizers in (15) and (18), we recommend setting $\lambda^{(\mathrm{T})} \ll \lambda^{(\mathrm{S})}$ since the student is exposed to more data than the teacher, i.e., while the teacher repeatedly processes the same training data, the student encounters new, randomly generated data at every stage. Therefore, we set $\lambda^{(\mathrm{T})} = 0.1$ and $\lambda^{(\mathrm{S})} = 1$.

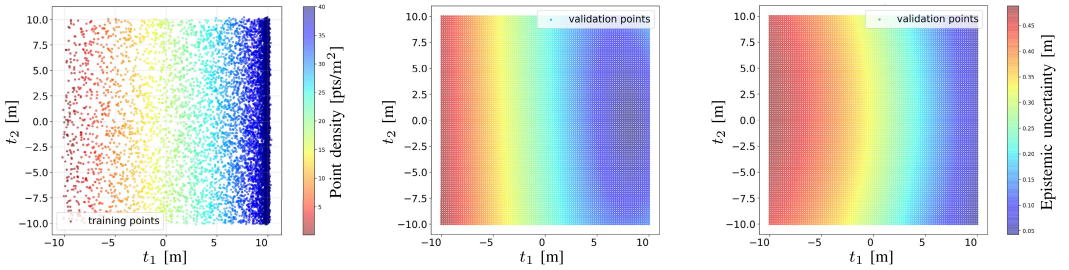
D. Numerical Results

1) Aleatoric and Epistemic Uncertainties: In this first experiment, we assess the capabilities of the proposed BBK method to learn both the aleatoric and epistemic uncertainty of the SGLD-based teacher. To this aim, we created a 2D artificial dataset where the input features are a noisy version of the UE 2D position (i.e., t_1 and t_2). To test the aleatoric uncertainty only, we employ a training dataset with densely sampled positions, so to induce the epistemic uncertainty to zero, and we add a Gaussian noise whose standard deviation varies linearly in $[0.1, 1]$ m along with the t_1 axes. Therefore, the objective is to predict this variation of aleatoric uncertainty which still remains in the dataset. The training dataset is shown in Fig. 5a. Then, we trained the SGLD-based teacher, the BDK-based student, and the BBK-based student, and we plotted the predicted variance measuring the aleatoric



(a) Training dataset (b) SGLD (c) BDK (d) BBK

Fig. 5. (a) Training points of the positioning dataset for aleatoric uncertainty assessment. (b), (c) and (d), predicted aleatoric uncertainty of the whole 2D space for SGLD, BDK and BBK, respectively.



(a) Training dataset (b) SGLD (c) BBK

Fig. 6. (a) Training points of the positioning dataset for epistemic uncertainty assessment. (b) and (c), predicted epistemic uncertainty of the whole 2D space for SGLD, and BBK, respectively.

uncertainty of the three methods on a 2D testing grid for $t_1, t_2 \in [-10, 10]$ m. From the results in Fig. 5, we observe that both the Bayesian teacher SGLD and the BBK are able to accurately reproduce the real additive noise in Fig. 5a. Indeed, the BBK-based student is trained to predict the SGLD-based teacher output, whose performances are boosted by the L consecutive predictions per input sample. On the contrary, the BDK achieves lower fidelity since this method is trained to approximate the point estimate of the teacher at each step, whereas the BBK is trained to predict the average of L predictions, i.e., the sum in (20).

To test the epistemic uncertainty, we fix the standard deviation of the additive Gaussian noise to 0.1 m, while we linearly change the density of the training points in the range $[4, 40]$ pts/m^2. The training dataset and the epistemic uncertainty predictions for SGLD and BBK are reported in Fig. 6. Note that, in this scenario, the BDK cannot be evaluated as it lacks the capability to predict epistemic uncertainty. For comparison of the epistemic uncertainty with the ground truth, we need to confront the predicted epistemic uncertainty with the squared root of the inverse density of the training points. This is because, in a well-calibrated BNN model, whenever a prediction has $P\,\%$ confidence, then the model's forecast aligns with the true occurrence approximately $P\,\%$ of the time. From the results, we can notice that the proposed BBK reconstructs almost completely the epistemic uncertainty, while being at the same time L times faster than the teacher model. Note that, whenever the training points are very dense,

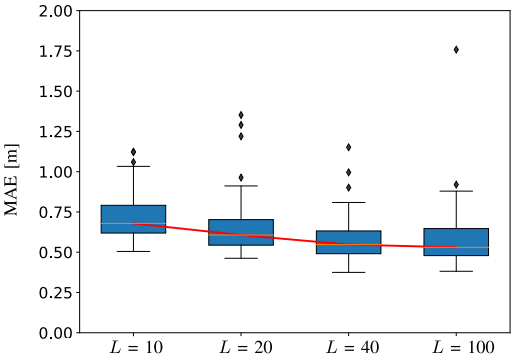


Fig. 7. Boxplot of the MAE per batch for different numbers of samples L realized from the posterior distribution $p_{\boldsymbol{\theta}|\mathrm{D}}(\boldsymbol{\theta}|\mathcal{D})$.

i.e., 40 pts/m^2, the epistemic uncertainty almost drops to zero. On the contrary, when we have a low density, such as the extreme case of 4 pts/m^2 (i.e., 1 pt/0.25 m^2), the predicted epistemic uncertainty is very similar to the squared root of the inverse density of the training points.

2) Hyper-Parameter Tuning for MC Sampling: This assessment is for tuning the number of samples L adopted in the teacher SGLD, as well as for verifying the maximum static positioning accuracy achieved by the proposed single-BS DL model in the ray-tracing dataset. To this aim, Fig. 7 shows

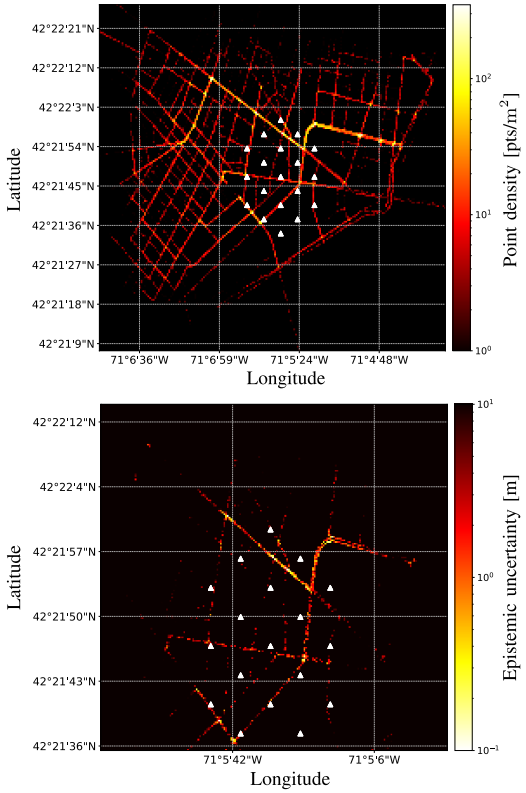


Fig. 8. Density of training ADCPM samples [pts/m^2] (top). Predicted testing epistemic uncertainty [m] (bottom).

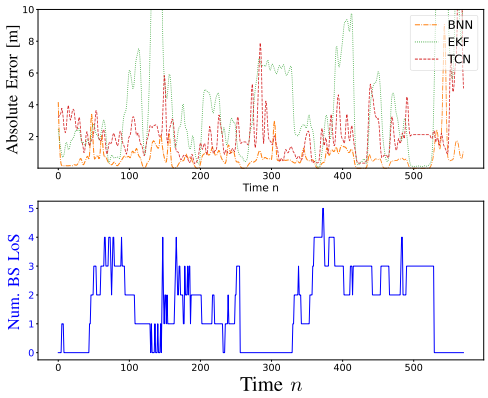


Fig. 9. Tracking performances in terms of absolute error (top). Number of BSs in LOS per timestep (bottom).

the boxplot of the testing MAE over the mini-batches varying $L \in \{10, 20, 40, 100\}$. First, we observe that even with a low L, the achieved median MAE is below 80 cm, confirming the capabilities of the model to infer the position from the full CIR. Second, we notice that generally increasing the number samples, i.e., the number of ensembles employed to estimate the posterior predictive distribution, decreases the positioning error. This is true up to a plateau of $L = 40$ where we can fully represent the real output distribution and achieve about 60 cm of error. In SGLD, a typical choice of L for traversing the posterior is given by N/M [62], which is equal to the number of steps to process the whole dataset. However, in practice, this number could be much smaller and thus, for the rest of the paper, we set $L = 40$. We point out that, considering a 1.6 ms of inference time per sample, using only the teacher with $L = 40$ for position prediction would be unfeasible for real-time applications.

3) Out of Distribution Uncertainty Estimation: This experiment has the goal of assessing the epistemic uncertainty of the DL model and BBK method in positions where no or few training samples are given. To verify this behaviour, in Fig. 8, we show the density of the training points (Fig. 8-top) and the predicted epistemic uncertainty of the student over random testing positions around the BS area (Fig. 8-bottom). From the figure, we can clearly notice that the model is much more confident where many training points are provided (around 10 cm for a density 80-90 pts/m^2), confirming the results of previous analysis. On the contrary, in the presence of around 10 pts/m^2, the student correctly predicts an uncertainty of about 50 cm. In the extreme cases of only 1 point and 2 points, the uncertainty greater than 2 m clearly indicates that the model is highly uncertain in those areas. Intuitively, this spatial uncertainty might be linked to the spatial decorrelation distance of the 5G system, but we leave this problem for future research. We point out that, by correctly predicting the epistemic uncertainty, the system automatically generalizes on unseen input samples as the predicted uncertainty is then adopted by the tracking system to weight the importance of the sample by means of the likelihood function.

4) Mobile Positioning in Urban Environment: This final experiment has the objective of comparing the performances of the integrated BNN tracking method with respect to two baselines: an EKF and a state-of-the-art TCN model described in [56]. For a fair comparison, both the BNN-based approach and the EKF adopt the same motion model (i.e., a random walk with 2 m standard deviation on the position), but they differ in the update step. For the EKF, we employ, as in conventional geometric localization, LOS time difference of flight (TDOF) measurements, estimated from the cross-correlation with the SRS according to 3GPP standard, and LOS AOA measurements, obtained through the multiple signal classification (MUSIC) algorithm [93]. Given the high blockage level of 5G signals due to the buildings, the UEs are also equipped with a GNSS receiver from which noisy measurements of the state are gathered. The standard deviation of the Gaussian noise on the GNSS measurements is set to 2 m and it serves as an upper-bound on the accuracy. On the contrary, for BNN-based tracking, we just employ the output of the real-time

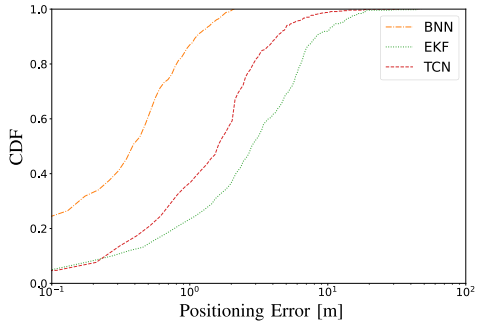


Fig. 10. Positioning performances in terms of cumulative density function (CDF) of the distance error for the proposed BNN-based tracking, EKF and TCN.

student with BBK method obtained from both LOS and NLOS ADCPM measurements. Regarding the TCN model, as suggested in [56], we consider the 1D CIR of the nearest BS by transforming the 2D ADCPM into a single vector.

The results of the tracking, i.e., testing trajectories in Fig. 4, are shown in Fig. 9 where we report the absolute location error per timestep, together with the number of BSs in LOS. Moreover, in Fig. 10 we also report the CDF of the positioning error for all the methods. From Fig. 9 we can clearly notice that the EKF struggles to achieve a location accuracy of 2 m when the number of LOS BSs is less than 3. However, even in a dense UMi scenario with 19 sites (57 BSs), the average number of LOS BSs is 1.6, leading to frequent inaccuracies in positioning. On the contrary, the TCN model achieves slightly higher performances by tracking the UE position even with just one BS measurement, thanks to the fingerprinting approach. However, among all methods, the BNN-based tracking consistently achieves sub-meter accuracy even in the absence of any LOS BS. This is mainly due to its ability to exploit BSs cooperation by fusing multiple NLOS position estimations and to the usage of 2D ADCPM with an AE structure which captures spatial relations in the input. Finally, observing the CDF of the absolute error, we can notice that the BNN-based tracking outperforms both the TCN and EKF by reaching a median error of 46 cm and under 1 m in 87% of the cases.

VII. Conclusion

In this paper, we addressed the problem of real-time 6G tracking in dense urban environments under heavy signal blockage, by presenting a first step toward the development of reliable and trustworthy DL models for precise positioning. We propose a novel teacher-student BNN method, namely BBK, which permits the prediction of real-time location estimates, together with an evaluation of both aleatoric and epistemic uncertainties. Estimation of both terms is of utmost importance for providing reliability indicators in critical applications and for optimizing the positioning process (e.g., augmenting training in uncertain areas). This enables the integration of the BNN method into a proposed tracking system, which seamlessly combines with existing tracking algorithms by substituting or enhancing the measurement-update step. The BNN method is applied to a proposed AE-based DL model, as foreseen by 3GPP standard, which takes as input the whole CIR by means of a 2D ADCPM. This permits to exploit position-linked attributes like TOF, AOA, and RSS of each propagation path, as a channel fingerprint.

The real-time BBK method and tracking integration are tested in a realistic C-ITS setting within a 3GPP-specification compliant UMi scenario, created by means of 3D maps and advanced ray-tracing simulations. The results show that the proposed BBK methodology is able to estimate both the aleatoric uncertainty, outperforming the state-of-the-art real-time BDK method, and the epistemic uncertainty in OOD scenarios from the reference teacher. Regarding mobile positioning performances, the proposed cooperative tracking methodology outperforms geometric-based tracking filters and state-of-the-art TCN models by localizing a moving target with a median absolute error of 46 cm.

Future works include extending our approach to both indoor and outdoor scenarios with next-generation cellular networks. In indoor scenarios, the main challenges include severe multipath and frequent changes of the channel characteristics due to moving objects in the environment. In outdoor scenarios, implementation and assessment in real-world C-ITS is an important research direction, where training procedure can be performed by digital twin simulation.

Appendix A
Proof of (16)

To prove (16), we start by writing the function $A(\boldsymbol{w}|\boldsymbol{x})$ as

$$\begin{aligned} A(\boldsymbol{w}|\boldsymbol{x}) &= \mathrm{KL}\big(p(t|\boldsymbol{x},\mathcal{D})\|p(t|\boldsymbol{x},\boldsymbol{w})\big) \\ &= \int p(t|\boldsymbol{x},\mathcal{D}) \log \frac{p(t|\boldsymbol{x},\mathcal{D})}{p(t|\boldsymbol{x},\boldsymbol{w})} dt \\ &\backsimeq -\mathbb{E}_{p(t|\boldsymbol{x},\mathcal{D})}\{\log p(t|\boldsymbol{x},\boldsymbol{w})\} \end{aligned} \tag{31}$$

where the last approximation comes from the removal of constant terms in $\boldsymbol{w}$. Therefore, we can write

$$\begin{aligned} A(\boldsymbol{w}|\boldsymbol{x}) &\backsimeq -\int \left(\int p(t|\boldsymbol{x},\boldsymbol{\theta}) p(\boldsymbol{\theta}|\mathcal{D}) d\boldsymbol{\theta} \right) \log p(t|\boldsymbol{x},\boldsymbol{w}) dt \\ &= -\int p(\boldsymbol{\theta}|\mathcal{D}) \left(\int p(t|\boldsymbol{x},\boldsymbol{\theta}) \log p(t|\boldsymbol{x},\boldsymbol{w}) dt \right) d\boldsymbol{\theta} \\ &= -\int p(\boldsymbol{\theta}|\mathcal{D}) \left(\mathbb{E}_{p(t|\boldsymbol{x},\boldsymbol{\theta})}\{\log p(t|\boldsymbol{x},\boldsymbol{w})\} \right) d\boldsymbol{\theta} \\ &\backsimeq -\frac{1}{L}\sum_{\ell=1}^{L} \mathbb{E}\Big\{ \log p(t|\boldsymbol{x},\boldsymbol{w}) \Big\} = \frac{1}{L}\sum_{\ell=1}^{L} A(\boldsymbol{w}|\boldsymbol{x},\boldsymbol{\theta}_\ell) \end{aligned} \tag{32}$$

where we adopted the MC approximation for calculating the integral using the samples $\boldsymbol{\theta}_\ell$. For regression tasks, $p(t|\boldsymbol{x},\boldsymbol{\theta}_\ell) = \mathcal{N}\left(t; y^{(\mathrm{T})}(\boldsymbol{x},\boldsymbol{\theta}_\ell), {\sigma_{\boldsymbol{\epsilon}^{(\mathrm{T})}}}^2\right)$, whereas $p(t|\boldsymbol{x},\boldsymbol{w}) = \mathcal{N}(t; y^{(\mathrm{S})}(\boldsymbol{x},\boldsymbol{w}), \sigma_{\boldsymbol{\epsilon}^{(\mathrm{S})}}(\boldsymbol{x})^2)$. Thus, we can write

$A(\boldsymbol{w}|\boldsymbol{x},\boldsymbol{\theta}_\ell)$ without considering the constant values in $\boldsymbol{w}$ as

$$\begin{aligned} A(\boldsymbol{w}|\boldsymbol{x},\boldsymbol{\theta}_\ell) &= -\int p(t|\boldsymbol{x},\boldsymbol{\theta}_\ell)\log p(t|\boldsymbol{x},\boldsymbol{w})dt \\ &\backsimeq \int p(t|\boldsymbol{x},\boldsymbol{\theta}_\ell)\Big(-\frac{\log\left(\sigma_{\boldsymbol{\epsilon}^{(\mathrm{S})}}(\boldsymbol{x})^2\right)}{2}+\frac{\|y^{(\mathrm{S})}(\boldsymbol{x},\boldsymbol{w})-t\|_2^2}{2\sigma_{\boldsymbol{\epsilon}^{(\mathrm{S})}}(\boldsymbol{x})^2}\Big)dt. \end{aligned} \tag{33}$$

Finally, by splitting the integral for each of the terms inside the L2 norm and by adopting completion of squares, we obtain

$$\begin{aligned} A(\boldsymbol{w}|\boldsymbol{x},\boldsymbol{\theta}_\ell) &\backsimeq \frac{1}{2}\log\left(\sigma_{\boldsymbol{\epsilon}^{(\mathrm{S})}}(\boldsymbol{x})^2\right)+\frac{1}{2}\sigma_{\boldsymbol{\epsilon}^{(\mathrm{S})}}(\boldsymbol{x})^{-2} \\ &\quad\times\left(\sigma_{\epsilon^{(\mathrm{T})}}{}^2+\left\|y^{(\mathrm{T})}(\boldsymbol{x},\boldsymbol{\theta}_\ell)-y^{(\mathrm{S})}(\boldsymbol{x},\boldsymbol{w})\right\|_2^2\right) \\ &\backsimeq \frac{1}{2}\log\left(y_{\mathrm{al}}^{(\mathrm{S})}(\boldsymbol{x},\boldsymbol{w})\right)+\frac{1}{2}y_{\mathrm{al}}^{(\mathrm{S})}(\boldsymbol{x},\boldsymbol{w})^{-1} \\ &\quad\times\left(\sigma_{\epsilon^{(\mathrm{T})}}{}^2+\left\|y^{(\mathrm{T})}(\boldsymbol{x},\boldsymbol{\theta}_\ell)-y^{(\mathrm{S})}(\boldsymbol{x},\boldsymbol{w})\right\|_2^2\right) \end{aligned} \tag{34}$$

where the final approximation arises from the model (14).

Appendix B
Proof of (17)

We derive the loss function block $B(\boldsymbol{w}|\boldsymbol{x})$ that permits the student network to learn the epistemic uncertainty of the teacher. By recalling the model in (14), we start by writing the negative log-likelihood according to the MLE approach:

$$\begin{aligned} B(\boldsymbol{w}|\boldsymbol{x}) &= -\log\left(p\left(\mathbb{V}\{\mathsf{t}|\mathsf{x},\mathsf{D},\boldsymbol{\epsilon}^{(\mathrm{T})}\}\Big|\boldsymbol{x},\boldsymbol{w}\right)\right) \\ &= -\log\left(\mathcal{N}\left(\mathbb{V}\{\mathsf{t}|\mathsf{x},\mathsf{D},\boldsymbol{\epsilon}^{(\mathrm{T})}\};\, y_{\mathrm{ep}}^{(\mathrm{S})}(\boldsymbol{x},\boldsymbol{w}),\sigma_{\boldsymbol{\xi}_{\mathrm{ep}}^{(\mathrm{S})}}{}^2\right)\right). \end{aligned} \tag{35}$$

Then, we remove constant values in $\boldsymbol{w}$ and approximate the epistemic uncertainty with the predictive epistemic uncertainty of the teacher as

$$\begin{aligned} B(\boldsymbol{w}|\boldsymbol{x}) &\backsimeq \frac{1}{2\sigma_{\boldsymbol{\xi}_{\mathrm{ep}}^{(\mathrm{S})}}{}^2}\left\|\mathbb{V}\{\mathsf{t}|\mathsf{x},\mathsf{D},\boldsymbol{\epsilon}^{(\mathrm{T})}\}-y_{\mathrm{ep}}^{(\mathrm{S})}(\boldsymbol{x},\boldsymbol{w})\right\|_2^2 \\ &\backsimeq \frac{1}{2\sigma_{\boldsymbol{\xi}_{\mathrm{ep}}^{(\mathrm{S})}}{}^2}\Bigg\|\frac{1}{L}\sum_{\ell=1}^{L}y^{(\mathrm{T})}(\boldsymbol{x},\boldsymbol{\theta}_\ell)^2 \\ &\quad-\left(\frac{1}{L}\sum_{\ell=1}^{L}y^{(\mathrm{T})}(\boldsymbol{x},\boldsymbol{\theta}_\ell)\right)^2-y_{\mathrm{ep}}^{(\mathrm{S})}(\boldsymbol{x},\boldsymbol{w})\Bigg\|_2^2 \end{aligned} \tag{36}$$

concluding the derivation.

References

[1] M. Z. Win et al., "Network localization and navigation via cooperation," *IEEE Commun. Mag.*, vol. 49, no. 5, pp. 56–62, May 2011.

[2] A. Conti et al., "Location awareness in beyond 5G networks," *IEEE Commun. Mag.*, vol. 59, no. 11, pp. 22–27, Nov. 2021.

[3] J. Talvitie, M. Säily, and M. Valkama, "Orientation and location tracking of XR devices: 5G carrier phase-based methods," *IEEE J. Sel. Topics Signal Process.*, vol. 17, no. 5, pp. 919–934, Sep. 2023.

[4] A. Ghosh, A. Maeder, M. Baker, and D. Chandramouli, "5G evolution: A view on 5G cellular technology beyond 3GPP release 15," *IEEE Access*, vol. 7, pp. 127639–127651, 2019.

[5] C. Yang, S. Mao, and X. Wang, "An overview of 3GPP positioning standards," *GetMobile, Mobile Comput. Commun.*, vol. 26, no. 1, pp. 9–13, May 2022.

[6] T. Nakamura, "5G evolution and 6G," in *Proc. IEEE Symp. VLSI Technol.*, Sep. 2020, pp. 1–5.

[7] J. Levy, *RAN Workshop on 5G: Chairman Summary*, document RWS-150073, InterDigital Communication Inc., New York, NY, USA, Sep. 2015.

[8] J. A. del Peral-Rosado, R. Raulefs, J. A. López-Salcedo, and G. Seco-Granados, "Survey of cellular mobile radio localization methods: From 1G to 5G," *IEEE Commun. Surveys Tuts.*, vol. 20, no. 2, pp. 1124–1148, 2nd Quart., 2018.

[9] S. Parkvall et al., "5G NR release 16: Start of the 5G evolution," *IEEE Commun. Standards Mag.*, vol. 4, no. 4, pp. 56–63, Dec. 2020.

[10] I. Rahman et al., "5G evolution toward 5G advanced: An overview of 3GPP releases 17 and 18," *Ericsson Technol. Rev.*, vol. 2021, no. 14, pp. 2–12, Oct. 2021.

[11] *Study on Artificial Intelligence (AI)/Machine Learning (ML) for NR Air Interface*, document TR 38.843, Version 18.0.0, 3rd Gener. Partnership Project (3GPP), Sophia Antipolis, France, Jan. 2022.

[12] F. Morselli, S. M. Razavi, M. Z. Win, and A. Conti, "Soft information based localization for 5G networks and beyond," *IEEE Trans. Wireless Commun.*, vol. 22, no. 12, pp. 9923–9938, Dec. 2023.

[13] H. van der Veen, *Summary of RAN Rel-18 Workshop*, document RWS-210659, NEC Laboratories Eur., Heidelberg, Germany, Jul. 2021.

[14] G. Torsoli, M. Z. Win, and A. Conti, "Blockage intelligence in complex environments for beyond 5G localization," *IEEE J. Sel. Areas Commun.*, vol. 41, no. 6, pp. 1688–1701, Jun. 2023.

[15] J. Zheng, J. Zhang, J. Cheng, V. C. M. Leung, D. W. K. Ng, and B. Ai, "Asynchronous cell-free massive MIMO with rate-splitting," *IEEE J. Sel. Areas Commun.*, vol. 41, no. 5, pp. 1366–1382, May 2023.

[16] W. Saad, M. Bennis, and M. Chen, "A vision of 6G wireless systems: Applications, trends, technologies, and open research problems," *IEEE Netw.*, vol. 34, no. 3, pp. 134–142, May 2020.

[17] N. Zhu, J. Marais, D. Bétaille, and M. Berbineau, "GNSS position integrity in urban environments: A review of literature," *IEEE Trans. Intell. Transp. Syst.*, vol. 19, no. 9, pp. 2762–2778, Sep. 2018.

[18] M. Z. Win, Y. Shen, and W. Dai, "A theoretical foundation of network localization and navigation," *Proc. IEEE*, vol. 106, no. 7, pp. 1136–1165, Jul. 2018.

[19] A. De Angelis and C. Fischione, "Mobile node localization via Pareto optimization: Algorithm and fundamental performance limitations," *IEEE J. Sel. Areas Commun.*, vol. 33, no. 7, pp. 1288–1303, Jul. 2015.

[20] Z. Liu, A. Conti, S. K. Mitter, and M. Z. Win, "Communication-efficient distributed learning over networks—Part I: Sufficient conditions for accuracy," *IEEE J. Sel. Areas Commun.*, vol. 41, no. 4, pp. 1081–1101, Apr. 2023.

[21] Z. Liu, A. Conti, S. K. Mitter, and M. Z. Win, "Communication-efficient distributed learning over networks—Part II: Necessary conditions for accuracy," *IEEE J. Sel. Areas Commun.*, vol. 41, no. 4, pp. 1102–1119, Apr. 2023.

[22] M. Z. Win, W. Dai, Y. Shen, G. Chrisikos, and H. V. Poor, "Network operation strategies for efficient localization and navigation," *Proc. IEEE*, vol. 106, no. 7, pp. 1224–1254, Jul. 2018.

[23] A. Maddumabandara, H. Leung, and M. Liu, "Experimental evaluation of indoor localization using wireless sensor networks," *IEEE Sensors J.*, vol. 15, no. 9, pp. 5228–5237, Sep. 2015.

[24] D. Fortin-Simard, J.-S. Bilodeau, K. Bouchard, S. Gaboury, B. Bouchard, and A. Bouzouane, "Exploiting passive RFID technology for activity recognition in smart homes," *IEEE Intell. Syst.*, vol. 30, no. 4, pp. 7–15, Feb. 2015.

[25] A. Conti, M. Guerra, D. Dardari, N. Decarli, and M. Z. Win, "Network experimentation for cooperative localization," *IEEE J. Sel. Areas Commun.*, vol. 30, no. 2, pp. 467–475, Feb. 2012.

[26] U. A. Khan, S. Kar, and J. M. F. Moura, "Distributed sensor localization in random environments using minimal number of anchor nodes," *IEEE Trans. Signal Process.*, vol. 57, no. 5, pp. 2000–2016, May 2009.

[27] A. Conti, S. Mazuelas, S. Bartoletti, W. C. Lindsey, and M. Z. Win, "Soft information for Localization-of-Things," *Proc. IEEE*, vol. 107, no. 11, pp. 2240–2264, Sep. 2019.

[28] F. Meyer et al., "Message passing algorithms for scalable multitarget tracking," *Proc. IEEE*, vol. 106, no. 2, pp. 221–259, Feb. 2018.

[29] S. Bartoletti, A. Giorgetti, M. Z. Win, and A. Conti, "Blind selection of representative observations for sensor radar networks," *IEEE Trans. Veh. Technol.*, vol. 64, no. 4, pp. 1388–1400, Apr. 2015.

[30] X. Lin, "An overview of 5G advanced evolution in 3GPP release 18," *IEEE Wireless Commun. Stand. Mag.*, vol. 6, no. 3, pp. 77–83, Sep. 2022.

[31] K. Gao, H. Wang, H. Lv, and W. Liu, "Toward 5G NR high-precision indoor positioning via channel frequency response: A new paradigm and dataset generation method," *IEEE J. Sel. Areas Commun.*, vol. 40, no. 7, pp. 2233–2247, Jul. 2022.

[32] A. Conti, G. Torsoli, C. A. Gómez-Vega, A. Vaccari, G. Mazzini, and M. Z. Win, "3GPP-compliant datasets for xG location-aware networks," *IEEE Open J. Veh. Technol.*, vol. 5, pp. 473–484, 2024.

[33] A. Conti, G. Torsoli, C. A. Gómez-Vega, A. Vaccari, and M. Z. Win, 2023, "xG-Loc: 3GPP-compliant datasets for xG location-aware networks," *IEEE Dataport*, Dec. 2023, doi: 10.21227/rper-vc03.

[34] B. C. Tedeschini and M. Nicoli, "Cooperative deep-learning positioning in mmWave 5G-advanced networks," *IEEE J. Sel. Areas Commun.*, vol. 41, no. 12, pp. 3799–3815, Dec. 2023.

[35] B. C. Tedeschini, M. Nicoli, and M. Z. Win, "On the latent space of mmWave MIMO channels for NLOS identification in 5G-advanced systems," *IEEE J. Sel. Areas Commun.*, vol. 41, no. 6, pp. 1655–1669, May 2023.

[36] F. Liu et al., "Integrated sensing and communications: Toward dual-functional wireless networks for 6G and beyond," *IEEE J. Sel. Areas Commun.*, vol. 40, no. 6, pp. 1728–1767, Jun. 2022.

[37] G. Kwon, Z. Liu, A. Conti, H. Park, and M. Z. Win, "Integrated localization and communication for efficient millimeter wave networks," *IEEE J. Sel. Areas Commun.*, vol. 41, no. 12, pp. 3925–3941, Dec. 2023.

[38] R. Liu et al., "Integrated sensing and communication based outdoor multi-target detection, tracking, and localization in practical 5G networks," *Intell. Converged Netw.*, vol. 4, no. 3, pp. 261–272, Sep. 2023.

[39] N. Decarli, A. Guerra, C. Giovannetti, F. Guidi, and B. M. Masini, "V2X sidelink localization of connected automated vehicles," *IEEE J. Select. Areas Commun.*, vol. 42, no. 1, pp. 120–133, Jan. 2024.

[40] L. Barbieri, B. C. Tedeschini, M. Brambilla, and M. Nicoli, "Implicit vehicle positioning with cooperative LiDAR sensing," in *Proc. IEEE Int. Conf. Acoust., Speech Signal Process. (ICASSP)*, Jun. 2023, pp. 1–5.

[41] N. Piperigkos, A. S. Lalos, and K. Berberidis, "Graph Laplacian diffusion localization of connected and automated vehicles," *IEEE Trans. Intell. Transp. Syst.*, vol. 23, no. 8, pp. 12176–12190, Aug. 2022.

[42] L. V. Jospin, H. Laga, F. Boussaid, W. Buntine, and M. Bennamoun, "Hands-on Bayesian neural networks—A tutorial for deep learning users," *IEEE Comput. Intell. Mag.*, vol. 17, no. 2, pp. 29–48, May 2022.

[43] M. Abdar et al., "A review of uncertainty quantification in deep learning: Techniques, applications and challenges," *Inf. Fusion*, vol. 76, pp. 243–297, Dec. 2021.

[44] *Study Enhancement 3GPP Support for 5G V2X Services*, document TR 22.886, Version 16.2.0, 3rd Gener. Partnership Project (3GPP), Sophia Antipolis, France, Dec. 2018.

[45] A. Fouda, R. Keating, and H.-S. Cha, "Toward cm-level accuracy: Carrier phase positioning for IIoT in 5G-advanced NR networks," in *Proc. IEEE 33rd Annu. Int. Symp. Pers., Indoor Mobile Radio Commun. (PIMRC)*, Sep. 2022, pp. 782–787.

[46] *Study Expanded Improved NR Positioning*, document TR 38.859, Version 18.0.0, 3rd Gener. Partnership Project (3GPP), Sophia Antipolis, France, Dec. 2023.

[47] X. Wang, L. Gao, S. Mao, and S. Pandey, "CSI-based fingerprinting for indoor localization: A deep learning approach," *IEEE Trans. Veh. Technol.*, vol. 66, no. 1, pp. 763–776, Jan. 2017.

[48] Y. Chapre, A. Ignjatovic, A. Seneviratne, and S. Jha, "CSI-MIMO: An efficient Wi-Fi fingerprinting using channel state information with MIMO," *Pervasive Mobile Comput.*, vol. 23, pp. 89–103, Oct. 2015.

[49] H. Chen, Y. Zhang, W. Li, X. Tao, and P. Zhang, "ConFi: Convolutional neural networks based indoor Wi-Fi localization using channel state information," *IEEE Access*, vol. 5, pp. 18066–18074, 2017.

[50] X. Wang, X. Wang, and S. Mao, "ResLoc: Deep residual sharing learning for indoor localization with CSI tensors," in *Proc. IEEE 28th Annu. Int. Symp. Pers., Indoor, Mobile Radio Commun. (PIMRC)*, Oct. 2017, pp. 1–6.

[51] X. Wang, X. Wang, and S. Mao, "CiFi: Deep convolutional neural networks for indoor localization with 5 GHz Wi-Fi," in *Proc. IEEE Int. Conf. Commun. (ICC)*, May 2017, pp. 1–6.

[52] A. Foliadis, M. H. C. Garcia, R. A. Stirling-Gallacher, and R. S. Thomä, "CSI-based localization with CNNs exploiting phase information," in *Proc. IEEE Wireless Commun. Netw. Conf. (WCNC)*, Mar. 2021, pp. 1–6.

[53] C. Wu et al., "Learning to localize: A 3D CNN approach to user positioning in massive MIMO-OFDM systems," *IEEE Trans. Wireless Commun.*, vol. 20, no. 7, pp. 4556–4570, Jul. 2021.

[54] A. Shahmansoori, B. Uguen, G. Destino, G. Seco-Granados, and H. Wymeersch, "Tracking position and orientation through millimeter wave lens MIMO in 5G systems," *IEEE Signal Process. Lett.*, vol. 26, no. 8, pp. 1222–1226, Aug. 2019.

[55] H. Kim, H. Wymeersch, N. Garcia, G. Seco-Granados, and S. Kim, "5G mmWave vehicular tracking," in *Proc. 52nd Asilomar Conf. Signals, Syst., Comput.*, Oct. 2018, pp. 541–547.

[56] J. Gante, G. Falc ao, and L. Sousa, "Deep learning architectures for accurate millimeter wave positioning in 5G," *Neural Process. Lett.*, vol. 51, no. 1, pp. 487–514, Feb. 2020.

[57] Y. Ruan, L. Chen, X. Zhou, G. Guo, and R. Chen, "Hi-Loc: Hybrid indoor localization via enhanced 5G NR CSI," *IEEE Trans. Instrum. Meas.*, vol. 71, pp. 1–15, 2023.

[58] E. Hullermeier and W. Waegeman, "Aleatoric and epistemic uncertainty in machine learning: An introduction to concepts and methods," *Mach. Learn.*, vol. 110, no. 3, pp. 457–506, 2021.

[59] A. Kendall and Y. Gal, "What uncertainties do we need in Bayesian deep learning for computer vision?" 2017, *arXiv:1703.04977*.

[60] J. Postels et al., "On the practicality of deterministic epistemic uncertainty," 2021, *arXiv:2107.00649*.

[61] P. Izmailov, S. Vikram, M. D. Hoffman, and A. Gordon Wilson, "What are Bayesian neural network posteriors really like?" 2021, *arXiv:2104.14421*.

[62] M. Welling and Y. W. Teh, "Bayesian learning via stochastic gradient Langevin dynamics," in *Proc.28th Int. Conf. Mach. Learn.*, Jun. 5555, pp. 681–688.

[63] E. Daxberger, A. Kristiadi, A. Immer, R. Eschenhagen, M. Bauer, and P. Hennig, "Laplace redux—Effortless Bayesian deep learning," 2021, *arXiv:2106.14806*.

[64] M. Teye, H. Azizpour, and K. Smith, "Bayesian uncertainty estimation for batch normalized deep networks," 2018, *arXiv:1802.06455*.

[65] M. E. Khan, D. Nielsen, V. Tangkaratt, W. Lin, Y. Gal, and A. Srivastava, "Fast and scalable Bayesian deep learning by weight-perturbation in Adam," 2018, *arXiv:1806.04854*.

[66] C. Blundell, J. Cornebise, K. Kavukcuoglu, and D. Wierstra, "Weight uncertainty in neural networks," in *Proc. 32nd Int. Conf. Mach. Learn.*, Jul. 2015, pp. 1613–1622.

[67] Y. Gal and Z. Ghahramani, "Dropout as a Bayesian approximation: Representing model uncertainty in deep learning," 2015, *arXiv:1506.02142*.

[68] A. Korattikara, V. Rathod, K. Murphy, and M. Welling, "Bayesian dark knowledge," 2015, *arXiv:1506.04416*.

[69] *Study NR Positioning Support*, document TR 38.855, Version 16.0.0, 3rd Gener. Partnership Project (3GPP), Sophia Antipolis, France, Sep. 2019.

[70] P. A. Lopez et al., "Microscopic traffic simulation using SUMO," in *Proc. 21st Int. Conf. Intell. Transp. Syst. (ITSC)*, Nov. 2018, pp. 2575–2582.

[71] H. L. Van Trees, *Optimum Array Processing: Part IV of Detection, Estimation, and Modulation Theory*, 1st ed. Hoboken, NJ, USA: Wiley, Mar. 2002.

[72] D. Tse and P. Viswanath, *Fundamentals of Wireless Communication*. Cambridge, U.K.: Cambridge Univ. Press, Dec. 2005.

[73] X. Sun, X. Gao, G. Y. Li, and W. Han, "Single-site localization based on a new type of fingerprint for massive MIMO-OFDM systems," *IEEE Trans. Veh. Technol.*, vol. 67, no. 7, pp. 6134–6145, Jul. 2018.

[74] C. M. Bishop, *Pattern Recognition and Machine Learning*, vol. 4. New York, NY, USA: Springer, Aug. 2006.

[75] J. Schmidhuber, "Deep learning in neural networks: An overview," *Neural Netw.*, vol. 61, pp. 85–117, Jan. 2015.

[76] D. P. Kingma and J. Ba, "Adam: A method for stochastic optimization," 2014, *arXiv:1412.6980*.

[77] M. S. Arulampalam, S. Maskell, N. Gordon, and T. Clapp, "A tutorial on particle filters for online nonlinear/non-Gaussian Bayesian tracking," *IEEE Trans. Signal Process.*, vol. 50, no. 2, pp. 174–188, Feb. 2002.

[78] K. B. Petersen and M. S. Pedersen, *The Matrix Cookbook*, document Version 20121115, Nov. 2012. [Online]. Available: http://www2.compute.dtu.dk/pubdb/pubs/3274-full.html

[79] *Study Channel Model for Frequencies From 0.5 to 100 GHz (Rel-16)*, document TR 38.901, Version 16.1.0, 3rd Gener. Partnership Project (3GPP), Sophia Antipolis, France, Nov. 2020.

[80] C.-F. Yang and C.-J. Ko, "A ray tracing method for modeling indoor wave propagation and penetration," in *IEEE Antennas Propag. Soc. Int. Symp. Dig.*, vol. 1, Jul. 1996, pp. 441–444.

Chapter 9. Efficient Uncertainty Quantification for Mobile Positioning

[81] H.-J. Li, C.-C. Chen, T.-Y. Liu, and H.-C. Lin, "Applicability of ray-tracing technique for the prediction of outdoor channel characteristics," *IEEE Trans. Veh. Technol.*, vol. 49, no. 6, pp. 2336–2349, Nov. 2000.

[82] A. Hsiao, C. Yang, T. Wang, I. Lin, and W. Liao, "Ray tracing simulations for millimeter wave propagation in 5G wireless communications," in *Proc. IEEE Int. Symp. Antennas Propag., USNC/URSI Nat. Radio Sci. Meeting*, Oct. 2017, pp. 1901–1902.

[83] (2023). *Wireless InSite 3D Wireless Prediction Software*. Accessed: Nov. 2023. [Online]. Available: https://www.remcom.com/wireless-insite-em-propagation-software

[84] *Guidelines for Evaluation of Radio Interface Technologies for IMT-2020*, document ITU-R M.2412-0, Int. Telecommun. Union, Geneva, Switzerland, Oct. 2017.

[85] P. Wang, B. Di, H. Zhang, K. Bian, and L. Song, "Platoon cooperation in cellular V2X networks for 5G and beyond," *IEEE Trans. Wireless Commun.*, vol. 18, no. 8, pp. 3919–3932, Aug. 2019.

[86] S. Roger, M. Brambilla, B. C. Tedeschini, C. Botella-Mascarell, M. Cobos, and M. Nicoli, "Deep-learning-based radio map reconstruction for V2X communications," *IEEE Trans. Veh. Technol.*, vol. 73, no. 3, pp. 3863–3871, Mar. 2024.

[87] L. Gopal, Y. Rong, and Z. Zang, "Tomlinson–Harashima precoding based transceiver design for MIMO relay systems with channel covariance information," *IEEE Trans. Wireless Commun.*, vol. 14, no. 10, pp. 5513–5525, Oct. 2015.

[88] A. Hindy and A. Nosratinia, "Ergodic fading MIMO dirty paper and broadcast channels: Capacity bounds and lattice strategies," *IEEE Trans. Wireless Commun.*, vol. 16, no. 8, pp. 5525–5536, Aug. 2017.

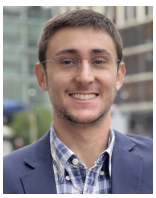
[89] R. Karásek and C. Gentner, "Stochastic data association for multipath assisted positioning using a single transmitter," *IEEE Access*, vol. 8, pp. 46735–46752, 2020.

[90] V. Badrinarayanan, A. Kendall, and R. Cipolla, "SegNet: A deep convolutional encoder-decoder architecture for image segmentation," *IEEE Trans. Pattern Anal. Mach. Intell.*, vol. 39, no. 12, pp. 2481–2495, Dec. 2017.

[91] G. Kalfas et al., "Next generation fiber-wireless Fronthaul for 5G mmWave networks," *IEEE Commun. Mag.*, vol. 57, no. 3, pp. 138–144, Mar. 2019.

[92] A. Paszke et al., "Automatic differentiation in PyTorch," in *Proc. 31st Int. Conf. Neural Inf. Process. Syst.*, Oct. 2017, pp. 1–4.

[93] R. Schmidt, "Multiple emitter location and signal parameter estimation," *IEEE Trans. Antennas Propag.*, vol. AP-34, no. 3, pp. 276–280, Mar. 1986.



Bernardo Camajori Tedeschini (Graduate Student Member, IEEE) is pursuing the Ph.D. degree in Information Technology at the Dipartimento di Elettronica, Informazione e Bioingegneria (DEIB), Politecnico di Milano, Milan, Italy, since November 2021. He received his M.Sc. (Hons.) degree in Telecommunications Engineering and B.Sc. (Hons.) degree in Computer Science from the Politecnico di Milano, Milan, Italy, in 2021 and 2019, respectively.

He is currently a visiting student with the Wireless Information and Network Sciences Laboratory at the Massachusetts Institute of Technology (MIT), Cambridge, MA, USA. In 2021, he has served as a Visiting Research Scientist at CERN, Geneva, Switzerland, where he worked on the CAFEIN project, focusing on the development and deployment of a Federated network platform. His research interests encompass federated learning, machine learning for signal processing and sensing over networks, and localization methods.

Mr. Camajori Tedeschini is a recipient of a Ph.D. grant from Italy's Ministero dell'Istruzione, dell'Università e della Ricerca (MIUR) and the Roberto Rocca Doctoral Fellowship, which was jointly awarded by MIT and Politecnico di Milano. He earned both his Bachelor's and Master's degrees with highest honors and he was honored with the best freshmen prize from Politecnico di Milano in 2017.

Girim Kwon (Member, IEEE) received the B.S. degree (with the highest honor) in electrical engineering from the University of Seoul, Seoul, South Korea, in 2013, and the M.S. and Ph.D. degrees in electrical engineering from the Korea Advanced Institute of Science and Technology (KAIST), Daejeon, South Korea, in 2014 and 2020, respectively.

He is currently a Postdoctoral Fellow with the Wireless Information and Network Sciences Laboratory at the Massachusetts Institute of Technology, Cambridge, MA, USA. His main areas of research are in statistical inference, information theory, optimization methods, and machine learning with applications to real-world problems, including wireless communications, network localization and navigation, and non-terrestrial networks.

Dr. Kwon received the S-Oil Best Dissertation Award in 2021, the Best Ph.D. Dissertation Award from KAIST in 2020, the ICT Paper Award from the Electronic Times in 2018, and the Global Ph.D. Fellowship from the Korean Government in 2015.

Monica Nicoli (Senior Member, IEEE) received the M.Sc. (Hons.) and Ph.D. degrees in communication engineering from Politecnico di Milano, Milan, Italy, in 1998 and 2002, respectively. She was a Visiting Researcher with ENI Agip, from 1998 to 1999, and Uppsala University, in 2001. In 2002, she joined Politecnico di Milano as a Faculty Member. She is currently an Associate Professor in telecommunications with the Department of Management, Economics and Industrial Engineering.

Her research interests include signal processing, machine learning, and wireless communications, with emphasis on smart mobility and Internet of Things (IoT). She was a recipient of the Marisa Bellisario Award, in 1999, and a co-recipient of the best paper awards of the EuMA Mediterranean Microwave Symposium, in 2022, the IEEE Symposium on Joint Communications and Sensing, in 2021, the IEEE Statistical Signal Processing Workshop, in 2018, and the *IET Intelligent Transport Systems journal*, in 2014. She is an Associate Editor of the IEEE TRANSACTIONS ON INTELLIGENT TRANSPORTATION SYSTEMS. She has also served as an Associate Editor for the *EURASIP Journal on Wireless Communications and Networking*, from 2010 to 2017, and a Lead Guest Editor for the Special Issue on Localization in Mobile Wireless and Sensor Networks, in 2011.

Moe Z. Win (Fellow, IEEE) is the Robert R. Taylor Professor at the Massachusetts Institute of Technology (MIT) and the founding director of the Wireless Information and Network Sciences Laboratory. Prior to joining MIT, he was with AT&T Research Laboratories and with the NASA Jet Propulsion Laboratory.

His research encompasses fundamental theories, algorithm design, and network experimentation for a broad range of real-world problems. His current research topics include ultra-wideband systems, network localization and navigation, network interference exploitation, and quantum information science. He has served the IEEE Communications Society as an elected Member-at-Large on the Board of Governors, as elected Chair of the Radio Communications Committee, and as an IEEE Distinguished Lecturer. Over the last two decades, he held various editorial positions for IEEE journals and organized numerous international conferences. He has served on the SIAM Diversity Advisory Committee.

Dr. Win is an elected Fellow of the AAAS, the EURASIP, the IEEE, and the IET. He was honored with two IEEE Technical Field Awards: the IEEE Kiyo Tomiyasu Award (2011) and the IEEE Eric E. Sumner Award (2006, jointly with R. A. Scholtz). His publications, co-authored with students and colleagues, have received several awards. Other recognitions include the MIT Frank E. Perkins Award (2024), the MIT Everett Moore Baker Award (2022), the IEEE Vehicular Technology Society James Evans Avant Garde Award (2022), the IEEE Communications Society Edwin H. Armstrong Achievement Award (2016), the Cristoforo Colombo International Prize for Communications (2013), the Copernicus Fellowship (2011) and the *Laurea Honoris Causa* (2008) from the Università degli Studi di Ferrara, and the U.S. Presidential Early Career Award for Scientists and Engineers (2004).

CHAPTER 10

Conclusions and Future Developments

In this thesis, we developed key solutions for open problems in MASs, mainly dividing between approaches for cooperative learning and cooperative inference. The firsts have been developed in the context of vehicular, medical, and IoT networks, whereas the seconds have been focused on next-generation cellular networks. In **Part II**, devoted to cooperative learning, we addressed the main challenges related to graph-aware, privacy-preserving, and non-stationary learning. Graph-aware centralized learning has been investigated for the tasks of DA and CP in complex networks where MPAs do not perform optimally. Privacy-preserving decentralized learning has been studied to evolve the concept of centralized learning into decentralized or fully decentralized networks of agents. We proposed a real FL platform, capable of performing asynchronous learning operations, which was tested in real medical and IoT networks. We subsequently explored solutions for the optimization of asynchronous FL processes, and we proposed innovative WAC algorithms for decentralized heterogeneous networks. Under resource-constrained devices, we extended the SL approach to a consensus architecture with the SCFL framework, enabling fully decentralized and low-complexity parallel training and testing procedures in MASs. The cooperative learning part concluded with the extension to algorithms for non-stationary learning, i.e., MARL algorithms in Dec-POMDP. In particular, we developed a data-driven extension of the ICP task where agents optimize both state learning and communication efficiency at the same time.

Gathered the knowledge of cooperative learning, in **Part III**, we explored the cooperative inference for sensing tasks in next-generation cellular networks, i.e., NLoS identification, CP and tracking. In particular, the objective was to exploit C-ML for creating models that were efficient, reliable, and accurate in the task at hand. For the NLoS identification problem, we proposed an anomaly detection framework where the latent feature distribution of high-dimensional channels is learned by means of a DAKDM model. This enabled an efficient representation and evaluation of distributions without

the need for large storage requirements, as well as multiple stages of training. For the CP task, we introduced a new loss function for simultaneous position regression and NLoS condition classification. Then, we proposed a cooperative inference algorithm that included the exchange of latent features among BSs to effectively perform localization in both NLoS and LoS conditions. Finally, we extended the static positioning to tracking scenarios where real-time inference is of paramount importance. To this aim, we first proposed a new BNN technique, namely BBK, that is able to evaluate the reliability, i.e., uncertainty of position estimates, without requiring sampling procedures and being able to distinguish the cause of the uncertainty. Then, we introduced an integration of general BNN methods into Bayesian tracking filters by combining the BNN predictions and related uncertainties into the update step.

Given the exponential increase of interest in cooperative techniques for MASs, many research paths can be undertaken to further improve the proposed algorithms and techniques in this thesis. Starting with graph-aware cooperative learning, future works could improve GNN-based DA by considering false and miss detections by means of intra-temporal measurement association or memory-based GNNs. Moreover, exactly as in MPA and neural-enhanced-BP, the GNN-based-CP could be extended to the more general task of MOT, so to have an entire data-driven graph-aware system capable of performing DA, CP and MOT. In D-ML and FD-ML, the asynchronous FL process optimization could be extended to consensus-based schemes taking into account not only agents' characteristics but also network delays and communications efficiency. Moreover, the study on decentralized FL algorithms for highly non-IID distributions could be extended to combine WAC with parameter optimization techniques to penalize divergence from the global model or to take into account the local distribution of data. Regarding non-stationary learning for ICP tasks, additional works could include the DA of target's measurements, methods for both decentralized training and execution procedures among agents, or the exploitation of general latent features of object detectors as targets for measurement fusion. In the context of cooperative inference in next-generation cellular networks, future works could investigate MOT by performing segmentation of the ADCPM to distinguish the components of the different UEs. Moreover, sim-to-real transfer learning could also be implemented where training is performed on simulated digital-twin maps, and then direct deployment and inference are performed in real systems. Finally, regarding BNN-based tracking, subsequent studies could expand the Gaussian likelihood approximation and also combine likelihoods of both geometric and fingerprinting measurements whenever LoS conditions are detected.

Given the advanced and cutting-edge nature of the proposed algorithms, significant challenges may arise when companies attempt to implement and deploy them in real-world scenarios. For instance, deploying the MPNN models for cooperative data association and the MARL algorithm ICP-MAPPO for cooperative positioning in vehicular networks may face obstacles such as the need for real-time, high-quality sensor data from multiple vehicles, reliable inter-vehicle communication links, and the computational limitations of onboard vehicle hardware. Vehicles may not have sufficient processing power to run complex algorithms in real-time, and ensuring low-latency communication between numerous fast-moving vehicles presents a significant technical challenge. For example, maintaining consistent performance during sudden changes in traffic flow or in areas with poor connectivity requires advanced communication

infrastructure. In the case of the custom FL system built on the MQTT protocol for medical networks, real-world implementation might encounter stringent data privacy regulations like GDPR. Hospitals and medical institutions must ensure that patient data remains confidential during transmission and processing. Additionally, the heterogeneity of medical devices in terms of computational capabilities and network connectivity can complicate the synchronization and efficiency of the FL process. The lack of standardized protocols across different medical devices can lead to compatibility issues, impeding seamless integration into the FL framework. Implementing interoperability standards such as Fast Healthcare Interoperability Resources (FHIR) and ensuring compliance with healthcare communication protocols like Health Level Seven International (HL7) can mitigate some of these challenges. For the SFL algorithms extended to the decentralized setting (i.e., SCFL), challenges include managing communication overhead and ensuring synchronization among resource-constrained IoT devices without a central coordinating server. Devices must efficiently exchange smashed data while preserving privacy and operating within their limited computational and energy budgets. In the context of next-generation cellular networks, implementing anomaly detection schemes (e.g., DAKDM) for NLoS identification or DL-based localization with latent feature exchange among BSs may be hindered by the need to process high-dimensional channel data (e.g., ADCPM) in real-time. This requires significant computational power and efficient data processing pipelines. Network infrastructure must be capable of handling the increased computational load without introducing latency that could degrade the performance of time-sensitive applications. In practice, network conditions may not always support the required bandwidth or latency, potentially affecting the accuracy and reliability of cooperative positioning. Lastly, integrating the BNN method BBK for real-time uncertainty quantification in mobile positioning into existing tracking systems presents challenges due to the need for frequent re-training to adapt to changing environments. BNNs may require extensive re-training when deployed in new areas or under varying conditions like infrastructure changes, weather, or signal propagation differences, which is time-consuming and computationally intensive. Collecting sufficient data for re-training can be impractical or costly. Ensuring accuracy and reliability without constant re-training is difficult. Techniques like transfer learning or domain adaptation may help but add complexity. Balancing updated models with computational and real-time processing constraints is crucial for deployment in dynamic, real-world scenarios.

The thesis successfully demonstrates the broad applications and advantages of cooperative machine learning within MASs, highlighting its pivotal role in improving the effectiveness and efficiency of complex task execution across various fields. Key takeaways include the demonstrated ability of MAS to handle tasks beyond the capability of individual agents, the critical importance of privacy and data integrity in distributed learning, and the flexibility of MAS in adapting to diverse and dynamic environments.

Bibliography

[1] A. Dorri, S. S. Kanhere, and R. Jurdak, "Multi-agent systems: A survey," *IEEE Access*, vol. 6, pp. 28 573–28 593, Apr. 2018.

[2] Y. Cao, W. Yu, W. Ren, and G. Chen, "An overview of recent progress in the study of distributed multi-agent coordination," *IEEE Trans. Ind. Inform.*, vol. 9, no. 1, pp. 427–438, Feb. 2013.

[3] N. Rieke, J. Hancox, W. Li, F. Milletarì, H. R. Roth, S. Albarqouni, S. Bakas, M. N. Galtier, B. A. Landman, K. Maier-Hein *et al.*, "The future of digital health with federated learning," *npj Digital Med.*, vol. 3, no. 1, p. 119, Sep. 2020.

[4] G. Fortino, W. Russo, C. Savaglio, W. Shen, and M. Zhou, "Agent-oriented cooperative smart objects: From IoT system design to implementation," *IEEE Trans. Syst., Man, Cybern.: Syst.*, vol. 48, no. 11, pp. 1939–1956, Nov. 2018.

[5] Q. Yang, S. Fu, H. Wang, and H. Fang, "Machine-learning-enabled cooperative perception for connected autonomous vehicles: Challenges and opportunities," *IEEE Netw.*, vol. 35, no. 3, pp. 96–101, May 2021.

[6] M. G. Kibria, K. Nguyen, G. P. Villardi, O. Zhao, K. Ishizu, and F. Kojima, "Big data analytics, machine learning, and artificial intelligence in next-generation wireless networks," *IEEE Access*, vol. 6, pp. 32 328–32 338, May 2018.

[7] Z. Zhang, P. Cui, and W. Zhu, "Deep learning on graphs: A survey," *IEEE Trans. Knowl. Data Eng.*, vol. 34, no. 1, pp. 249–270, Jan. 2022.

[8] E. Ahmed and H. Gharavi, "Cooperative vehicular networking: A survey," *IEEE Trans. Intell. Transp. Syst.*, vol. 19, no. 3, pp. 996–1014, Mar. 2018.

[9] M. Brambilla, M. Nicoli, G. Soatti, and F. Deflorio, "Augmenting vehicle localization by cooperative sensing of the driving environment: Insight on data association in urban traffic scenarios," *IEEE Trans. Intell. Transp. Syst.*, vol. 21, no. 4, pp. 1646–1663, Apr. 2020.

[10] G. Soatti, M. Nicoli, N. Garcia, B. Denis, R. Raulefs, and H. Wymeersch, "Implicit cooperative positioning in vehicular networks," *IEEE Trans. Intell. Transp. Syst.*, vol. 19, no. 12, pp. 3964–3980, Dec. 2018.

[11] J. Konecný, H. B. McMahan, D. Ramage, and P. Richtárik, "Federated optimization: Distributed machine learning for on-device intelligence," *ArXiv*, Oct. 2016.

[12] K. Hjerppe, J. Ruohonen, and V. Leppanen, "The general data protection regulation: Requirements, architectures, and constraints," in *2019 IEEE 27th Int. Requirements Eng. Conf. (RE)*. IEEE, Sep. 2019, pp. 265–275.

[13] D. C. Nguyen, M. Ding, P. N. Pathirana, A. Seneviratne, J. Li, and H. Vincent Poor, "Federated learning for internet of things: A comprehensive survey," *IEEE Commun. Surveys & Tuts.*, vol. 23, no. 3, pp. 1622–1658, Apr. 2021.

[14] B. Luo, X. Li, S. Wang, J. Huang, and L. Tassiulas, "Cost-effective federated learning in mobile edge networks," *IEEE J. Sel. Areas Commun.*, vol. 39, no. 12, pp. 3606–3621, Oct. 2021.

[15] H. T. Nguyen, V. Sehwag, S. Hosseinalipour, C. G. Brinton, M. Chiang, and H. Vincent Poor, "Fast-convergent federated learning," *IEEE J. Sel. Areas Commun.*, vol. 39, no. 1, p. 201–218, Jan. 2021.

[16] A. Bhowmick, J. Duchi, J. Freudiger, G. Kapoor, and R. Rogers, "Protection against reconstruction and its applications in private federated learning," Dec. 2018, arXiv:1812.00984.

[17] Z. Wang, Z. Zhang, Y. Tian, Q. Yang, H. Shan, W. Wang, and T. Q. S. Quek, "Asynchronous federated learning over wireless communication networks," *IEEE Trans. Wireless Commun.*, vol. 21, no. 9, pp. 6961–6978, Mar. 2022.

[18] S. Savazzi, M. Nicoli, and V. Rampa, "Federated learning with cooperating devices: A consensus approach for massive IoT networks," *IEEE Internet Things J.*, vol. 7, no. 5, pp. 4641–4654, Jan. 2020.

[19] P. Han, S. Wang, and K. K. Leung, "Adaptive gradient sparsification for efficient federated learning: An online learning approach," in *2020 IEEE 40th Int. Conf. Distrib. Comput. Syst. (ICDCS)*, Nov. 2020, pp. 300–310, iSSN: 2575-8411.

[20] P. Hernandez-Leal, M. Kaisers, T. Baarslag, and E. M. de Cote, "A survey of learning in multiagent environments: Dealing with non-stationarity," *ArXiv*, Jul. 2017.

[21] K. B. Letaief, Y. Shi, J. Lu, and J. Lu, "Edge artificial intelligence for 6G: Vision, enabling technologies, and applications," *IEEE J. Sel. Areas Commun.*, vol. 40, no. 1, pp. 5–36, Jan. 2022.

[22] A. Behravan, V. Yajnanarayana, M. F. Keskin, H. Chen, D. Shrestha, T. E. Abrudan, T. Svensson, K. Schindhelm, A. Wolfgang, S. Lindberg *et al.*, "Positioning and sensing in 6G: Gaps, challenges, and opportunities," *IEEE Veh. Technol. Mag.*, vol. 18, no. 1, pp. 40–48, Mar. 2023.

[23] S. Bartoletti, H. Wymeersch, T. Mach, O. Brunnegard, D. Giustiniano, P. Hammarberg, M. F. Keskin, J. O. Lacruz, S. M. Razavi, J. Ronnblom *et al.*, "Positioning and sensing for vehicular safety applications in 5G and beyond," *IEEE Commun. Mag.*, vol. 59, no. 11, pp. 15–21, Nov. 2021.

[24] A. Bourdoux, A. N. Barreto, B. van Liempd, C. de Lima, D. Dardari, D. Belot, E.-S. Lohan, G. Seco-Granados, H. Sarieddeen, H. Wymeersch *et al.*, "6G white paper on localization and sensing," *ArXiv*, Jun. 2020.

[25] 5GAA, "C-V2X use cases and service level requirements - volume II," 5GAA Automotive Association, Technical Report (TR), 2021.

[26] 5GAA, "C-V2X use cases and service level requirements - volume III," 5GAA Automotive Association, Technical Report (TR), 2023.

[27] 3GPP, "Release 15 description; summary of Rel-15 work items," 3rd Generation Partnership Project (3GPP), Technical Report (TR) 21.915, 2019, version 15.0.0 Release 15.

[28] *Study on enhancement of 3GPP Support for 5G V2X Services*, TR 22.886 Version 16.2.0, 3rd Generation Partnership Project (3GPP), Sophia Antipolis, France, Dec. 2018.

[29] W. Saad, M. Bennis, and M. Chen, "A vision of 6G wireless systems: Applications, trends, technologies, and open research problems," *IEEE Netw.*, vol. 34, no. 3, pp. 134–142, May 2020.

[30] D. Dardari, N. Decarli, A. Guerra, and F. Guidi, "LOS/NLOS near-field localization with a large reconfigurable intelligent surface," *IEEE Trans. Wireless Commun.*, vol. 21, no. 6, pp. 4282–4294, Jun. 2022.

[31] A. Xhafa, J. A. Del Peral-Rosado, J. A. López-Salcedo, and G. Seco-Granados, "Evaluation of 5G positioning performance based on UTDoA, AoA and base-station selective exclusion," *Sens.*, vol. 22, no. 1, p. 101, Dec. 2021.

[32] Y. Ge, H. Khosravi, F. Jiang, H. Chen, S. Lindberg, P. Hammarberg, H. Kim, O. Brunnegård, O. Eriksson, B.-E. Olsson *et al.*, "Experimental validation of single BS 5G mmWave positioning and mapping for intelligent transport," *ArXiv*, Mar. 2023.

[33] X. Lin, "An overview of 5G advanced evolution in 3GPP release 18," *IEEE Commun. Stand. Mag.*, vol. 6, no. 3, pp. 77–83, Sep. 2022.

[34] E. Hüllermeier and W. Waegeman, "Aleatoric and epistemic uncertainty in machine learning: an introduction to concepts and methods," *Mach. Learn.*, vol. 110, no. 3, pp. 457–506, Mar. 2021.

[35] C. Ruah, O. Simeone, and B. Al-Hashimi, "A Bayesian framework for digital twin-based control, monitoring, and data collection in wireless systems," *IEEE J. Sel. Areas Commun.*, vol. 41, no. 10, pp. 3146–3160, Aug. 2023.

[36] F. Kschischang, B. Frey, and H.-A. Loeliger, "Factor graphs and the sum-product algorithm," *IEEE Trans. Inf. Theory*, vol. 47, no. 2, pp. 498–519, Feb. 2001.

[37] H. Wymeersch, J. Lien, and M. Z. Win, "Cooperative localization in wireless networks," *Proc. IEEE*, vol. 97, no. 2, pp. 427–450, Feb. 2009.

[38] A. Conti, M. Guerra, D. Dardari, N. Decarli, and M. Z. Win, "Network experimentation for cooperative localization," *IEEE J. Sel. Areas Commun.*, vol. 30, no. 2, pp. 467–475, Feb. 2012.

[39] S. Zhang, E. Staudinger, T. Jost, W. Wang, C. Gentner, A. Dammann, H. Wymeersch, and P. A. Hoeher, "Distributed direct localization suitable for dense networks," *IEEE Trans. Aerosp. Electron. Syst.*, vol. 56, no. 2, pp. 1209–1227, Apr. 2020.

[40] A. Conti, S. Mazuelas, S. Bartoletti, W. C. Lindsey, and M. Z. Win, "Soft information for localization-of-things," *Proc. of the IEEE*, vol. 107, no. 11, pp. 2240–2264, Nov. 2019.

[41] D. Hall and J. Llinas, "An introduction to multisensor data fusion," *Proc. IEEE*, vol. 85, no. 1, pp. 6–23, Jan. 1997.

[42] J. Williams and R. Lau, "Approximate evaluation of marginal association probabilities with belief propagation," *IEEE Trans. Aerosp. Electron. Syst.*, vol. 50, no. 4, pp. 2942–2959, Oct. 2014.

[43] F. Meyer, T. Kropfreiter, J. L. Williams, R. Lau, F. Hlawatsch, P. Braca, and M. Z. Win, "Message passing algorithms for scalable multitarget tracking," *Proc. IEEE*, vol. 106, no. 2, pp. 221–259, Feb. 2018.

[44] F. Scarselli, M. Gori, Ah Chung Tsoi, M. Hagenbuchner, and G. Monfardini, "The graph neural network model," *IEEE Trans. Neural Netw.*, vol. 20, no. 1, pp. 61–80, Jan. 2008.

[45] V. Garcia Satorras and M. Welling, "Neural enhanced belief propagation on factor graphs," in *Proceedings of The 24th International Conference on Artificial Intelligence and Statistics*, ser. Proceedings of Machine Learning Research, A. Banerjee and K. Fukumizu, Eds., vol. 130. PMLR, Mar. 2020, pp. 685–693.

[46] M. Liang and F. Meyer, "Neural enhanced belief propagation for cooperative localization," in *2021 IEEE Statistical Signal Process. Workshop (SSP)*, Jul. 2021, pp. 326–330.

[47] J. Zhou, G. Cui, S. Hu, Z. Zhang, C. Yang, Z. Liu, L. Wang, C. Li, and M. Sun, "Graph neural networks: A review of methods and applications," *AI Open*, vol. 1, pp. 57–81, Apr. 2021.

[48] K. Yoon, R. Liao, Y. Xiong, L. Zhang, E. Fetaya, R. Urtasun, R. Zemel, and X. Pitkow, "Inference in probabilistic graphical models by graph neural networks," in *2019 53rd Asilomar Conf. Signals, Syst., Comput.* IEEE, Nov. 2019, pp. 868–875.

[49] H. B. McMahan, E. Moore, D. Ramage, S. Hampson, and B. A. y. Arcas, "Communication-efficient learning of deep networks from decentralized data," in *Proc. 20th Int. Conf. Artif. Intell. Statistics*, ser. Proceedings of Machine Learning Research, A. Singh and J. Zhu, Eds., vol. 54. PMLR, Apr. 2017, pp. 1273–1282.

[50] Y. Chen, Y. Ning, M. Slawski, and H. Rangwala, "Asynchronous online federated learning for edge devices with non-IID data," in *2020 IEEE Int. Conf. Big Data (Big Data)*, Dec. 2020, pp. 15–24.

[51] T. Li, A. K. Sahu, M. Zaheer, M. Sanjabi, A. Talwalkar, and V. Smith, "Federated optimization in heterogeneous networks," Dec. 2018, arXiv:1812.06127.

[52] H. Wu and P. Wang, "Fast-convergent federated learning with adaptive weighting," *IEEE Trans. Cogn. Commun. Netw.*, vol. 7, no. 4, pp. 1078–1088, May 2021.

[53] F. P.-C. Lin, S. Hosseinalipour, S. S. Azam, C. G. Brinton, and N. Michelusi, "Semi-decentralized federated learning with cooperative D2D local model aggregations," *IEEE J. Sel. Areas Commun.*, vol. 39, no. 12, pp. 3851–3869, Oct. 2021.

[54] G. Soatti, S. Savazzi, M. Nicoli, M. A. Alvarez, S. Kianoush, V. Rampa, and U. Spagnolini, "Distributed signal processing for dense 5G IoT platforms: Networking, synchronization, interference detection and radio sensing," *Ad Hoc Netw.*, vol. 89, pp. 9–21, Feb. 2019.

[55] R. Olfati-Saber, J. A. Fax, and R. M. Murray, "Consensus and cooperation in networked multi-agent systems," *Proc. IEEE*, vol. 95, no. 1, pp. 215–233, Mar. 2007.

[56] G. Zhu, Y. Wang, and K. Huang, "Broadband analog aggregation for low-latency federated edge learning," *IEEE Trans. Wireless Commun.*, vol. 19, no. 1, pp. 491–506, Oct. 2019.

[57] Y. Lu, X. Huang, Y. Dai, S. Maharjan, and Y. Zhang, "Differentially private asynchronous federated learning for mobile edge computing in urban informatics," *IEEE Trans. Ind. Inform.*, vol. 16, no. 3, pp. 2134–2143, Sep. 2019.

[58] O. Gupta and R. Raskar, "Distributed learning of deep neural network over multiple agents," *J. Netw. Comput. Appl.*, vol. 116, pp. 1–8, Aug. 2018.

[59] K. Palanisamy, V. Khimani, M. H. Moti, and D. Chatzopoulos, "Spliteasy: A practical approach for training ML models on mobile devices," in *Proc. 22nd Int. Workshop Mobile Comput. Syst. Appl.*, Feb. 2021, pp. 37–43.

[60] A. Singh, P. Vepakomma, O. Gupta, and R. Raskar, "Detailed comparison of communication efficiency of split learning and federated learning," *ArXiv*, Sep. 2019.

[61] J. Jeon and J. Kim, "Privacy-sensitive parallel split learning," in *2020 Int. Conf. Inf. Netw. (ICOIN)*, Mar. 2020, pp. 7–9.

[62] C. Thapa, P. C. Mahawaga Arachchige, S. Camtepe, and L. Sun, "Splitfed: When federated learning meets split learning," *Proc. AAAI Conf. Artif. Intell.*, vol. 36, no. 8, pp. 8485–8493, Jul. 2022.

[63] W. Wu, M. Li, K. Qu, C. Zhou, X. Shen, W. Zhuang, X. Li, and W. Shi, "Split learning over wireless networks: Parallel design and resource management," *IEEE J. Sel. Areas Commun.*, vol. 41, no. 4, pp. 1051–1066, Apr. 2023.

[64] R. S. Sutton and A. G. Barto, *Reinforcement Learning: An Introduction*. A Bradford Book, Oct. 2018.

[65] V. Mnih, K. Kavukcuoglu, D. Silver, A. A. Rusu, J. Veness, M. G. Bellemare, A. Graves, M. Riedmiller, A. K. Fidjeland, G. Ostrovski *et al.*, "Human-level

control through deep reinforcement learning," *Nature*, vol. 518, no. 7540, pp. 529–533, Feb. 2015.

[66] D. P. Bertsekas, *Lessons from AlphaZero for Optimal, Model Predictive, and Adaptive Control*, 1st ed. Athena Scientific, Aug. 2021.

[67] T. P. Lillicrap, J. J. Hunt, A. Pritzel, N. Heess, T. Erez, Y. Tassa, D. Silver, and D. Wierstra, "Continuous control with deep reinforcement learning," in *Proc. 4th Int. Conf. Learn. Representations*, Sep. 2016, pp. 1–14.

[68] N. R. Ke, A. Singh, A. Touati, A. Goyal, Y. Bengio, D. Parikh, and D. Batra, "Learning dynamics model in reinforcement learning by incorporating the long term future," in *Proc. 7th Int. Conf. Learn. Representations*, Mar. 2019, pp. 1–14.

[69] D. P. Bertsekas, *Dynamic Programming and Optimal Control*, 4th ed. Athena Scientific, Oct. 2000.

[70] F. A. Oliehoek and C. Amato, *A Concise Introduction to Decentralized POMDPs*, ser. SpringerBriefs in Intelligent Systems. Springer International Publishing, Jun. 2016.

[71] L. Kraemer and B. Banerjee, "Multi-agent reinforcement learning as a rehearsal for decentralized planning," *Neurocomputing*, vol. 190, no. C, pp. 82–94, May 2016.

[72] S. Omidshafiei, J. Pazis, C. Amato, J. P. How, and J. Vian, "Deep decentralized multi-task multi-agent reinforcement learning under partial observability," in *Proc. 34th Int. Conf. Mach. Learn.*, Aug. 2017, pp. 2681–2690.

[73] S. Bhattacharya, S. Kailas, S. Badyal, S. Gil, and D. Bertsekas, "Multiagent reinforcement learning: Rollout and policy iteration for POMDP with application to multirobot problems," *IEEE Trans. Robot.*, vol. 40, pp. 2003–2023, Dec. 2024.

[74] T. T. Nguyen, N. D. Nguyen, and S. Nahavandi, "Deep reinforcement learning for multiagent systems: A review of challenges, solutions, and applications," *IEEE Trans. Cybern.*, vol. 50, no. 9, pp. 3826–3839, Mar. 2020.

[75] S. Gronauer and K. Diepold, "Multi-agent deep reinforcement learning: a survey," *Artif. Intell. Rev.*, vol. 55, no. 2, pp. 895–943, Apr. 2021.

[76] Z. Xia, J. Du, J. Wang, C. Jiang, Y. Ren, G. Li, and Z. Han, "Multi-agent reinforcement learning aided intelligent UAV swarm for target tracking," *IEEE Trans. Veh. Technol.*, vol. 71, no. 1, pp. 931–945, Nov. 2021.

[77] B. Peng, G. Seco-Granados, E. Steinmetz, M. Frohle, and H. W. Wymeersch, "Decentralized scheduling for cooperative localization with deep reinforcement learning," *IEEE Trans. Veh. Technol.*, vol. 68, no. 5, pp. 4295–4305, May 2019.

[78] M. Rangwala and R. Williams, "Learning multi-agent communication through structured attentive reasoning," in *Proc. 34th Int. Conf. Neural Inf. Process. Syst.*, vol. 33, Dec. 2020, pp. 10 088–10 098.

[79] E. Pesce and G. Montana, "Learning multi-agent coordination through connectivity-driven communication," *Machine Learning*, vol. 112, no. 2, p. 483–514, Dec. 2022.

[80] S. Marano, W. Gifford, H. Wymeersch, and M. Win, "NLOS identification and mitigation for localization based on UWB experimental data," *IEEE Journal on Selected Areas in Communications*, vol. 28, no. 7, pp. 1026–1035, Sep. 2010.

[81] C. Huang, R. He, B. Ai, A. F. Molisch, B. K. Lau, K. Haneda, B. Liu, C.-X. Wang, M. Yang, C. Oestges *et al.*, "Artificial intelligence enabled radio propagation for communications—part II: Scenario identification and channel modeling," *IEEE Trans. on Antennas and Propag.*, vol. 70, no. 6, pp. 3955–3969, Jun. 2022.

[82] T. Van Nguyen, Y. Jeong, H. Shin, and M. Z. Win, "Machine learning for wideband localization," *IEEE J. Sel. Areas Commun.*, vol. 33, no. 7, pp. 1357–1380, Jul. 2015.

[83] H. Wymeersch, S. Marano, W. M. Gifford, and M. Z. Win, "A machine learning approach to ranging error mitigation for UWB localization," *IEEE Transactions on Communications*, vol. 60, no. 6, pp. 1719–1728, Jun. 2012.

[84] X. Yang, "NLOS mitigation for UWB localization based on sparse pseudo-input Gaussian process," *IEEE Sens. J.*, vol. 18, no. 10, pp. 4311–4316, May 2018.

[85] H. Chen, Y. Zhang, W. Li, X. Tao, and P. Zhang, "Confi: Convolutional neural networks based indoor wi-fi localization using channel state information," *IEEE Access*, vol. 5, pp. 18 066–18 074, Sep. 2017.

[86] Y. Jing, J. Hao, and P. Li, "Learning spatiotemporal features of CSI for indoor localization with dual-stream 3D convolutional neural networks," *IEEE Access*, vol. 7, pp. 147 571–147 585, Oct. 2019.

[87] J. E. Van Engelen and H. H. Hoos, "A survey on semi-supervised learning," *Machine Learning*, vol. 109, no. 2, pp. 373–440, Nov. 2020.

[88] M. Stahlke, S. Kram, F. Ott, T. Feigl, and C. Mutschler, "Estimating TOA reliability with variational autoencoders," *IEEE Sens. J.*, vol. 22, no. 6, pp. 5133–5140, Mar. 2022.

[89] C. Zhou and R. C. Paffenroth, "Anomaly detection with robust deep autoencoders," in *Proc. of the 23rd ACM SIGKDD Int. Conf. on Knowledge Discovery and Data Mining*, ser. KDD '17. Association for Computing Machinery, Aug. 2017, pp. 665–674.

[90] V. L. Cao, M. Nicolau, and J. McDermott, "A hybrid autoencoder and density estimation model for anomaly detection," in *Parallel Problem Solving from Nature – PPSN XIV*, J. Handl, E. Hart, P. R. Lewis, M. López-Ibáñez, G. Ochoa, and B. Paechter, Eds. Springer International Publishing, Aug. 2016, pp. 717–726.

[91] B. Zong, Q. Song, M. R. Min, W. Cheng, C. Lumezanu, D. Cho, and H. Chen, "Deep autoencoding Gaussian mixture model for unsupervised anomaly detection," in *Int. Conf. on Learning Representations*, Feb. 2018.

[92] P. Bahl and V. Padmanabhan, "RADAR: an in-building RF-based user location and tracking system," in *Proceedings IEEE INFOCOM 2000. Conference on Computer Communications. Nineteenth Annual Joint Conference of the IEEE Computer and Communications Societies (Cat. No.00CH37064)*, vol. 2. IEEE, Mar. 2000, pp. 775–784.

[93] M. Youssef and A. Agrawala, "The horus WLAN location determination system," in *Proceedings of the 3rd international conference on Mobile systems, applications, and services*. ACM, Jun. 2005, pp. 205–218.

[94] X. Wang, L. Gao, S. Mao, and S. Pandey, "Deepfi: Deep learning for indoor fingerprinting using channel state information," in *2015 IEEE Wireless Communications and Networking Conference (WCNC)*, Mar. 2015, pp. 1666–1671, iSSN: 1558-2612.

[95] Y. Chapre, A. Ignjatovic, A. Seneviratne, and S. Jha, "CSI-MIMO: An efficient wi-fi fingerprinting using channel state information with MIMO," *Pervasive Mobile Comput.*, vol. 23, pp. 89–103, Oct. 2015.

[96] X. Wang, L. Gao, and S. Mao, "Phasefi: Phase fingerprinting for indoor localization with a deep learning approach," in *2015 IEEE Global Commun. Conf. (GLOBECOM)*. IEEE, Dec. 2015, pp. 1–6.

[97] S. Savazzi, S. Sigg, M. Nicoli, V. Rampa, S. Kianoush, and U. Spagnolini, "Device-free radio vision for assisted living: Leveraging wireless channel quality information for human sensing," *IEEE Signal Processing Magazine*, vol. 33, no. 2, pp. 45–58, Mar. 2016.

[98] X. Wang, X. Wang, and S. Mao, "Resloc: Deep residual sharing learning for indoor localization with CSI tensors," in *2017 IEEE 28th Annu. Int. Symp. Pers., Indoor, Mobile Radio Commun. (PIMRC)*. IEEE, Oct. 2017, pp. 1–6.

[99] X. Wang, X. Wang, and S. Mao, "Cifi: Deep convolutional neural networks for indoor localization with 5 ghz wi-fi," in *2017 IEEE Int. Conf. Commun. (ICC)*. IEEE, May 2017, pp. 1–6.

[100] N. Lv, F. Wen, Y. Chen, and Z. Wang, "A deep learning-based end-to-end algorithm for 5G positioning," *IEEE Sens. Lett.*, vol. 6, no. 4, pp. 1–4, Apr. 2022.

[101] K. Gao, H. Wang, H. Lv, and W. Liu, "Toward 5G NR high-precision indoor positioning via channel frequency response: A new paradigm and dataset generation method," *IEEE J. Sel. Areas Commun.*, vol. 40, no. 7, pp. 2233–2247, Jul. 2022.

[102] C. Wu, X. Yi, W. Wang, L. You, Q. Huang, X. Gao, and Q. Liu, "Learning to localize: A 3D CNN approach to user positioning in massive MIMO-OFDM systems," *IEEE Trans. Wireless Commun.*, vol. 20, no. 7, pp. 4556–4570, Jul. 2021.

[103] A. Shahmansoori, B. Uguen, G. Destino, G. Seco-Granados, and H. Wymeersch, "Tracking position and orientation through millimeter wave lens MIMO in 5G systems," *IEEE Signal Process. Lett.*, vol. 26, no. 8, pp. 1222–1226, Aug. 2019.

[104] H. Kim, H. Wymeersch, N. Garcia, G. Seco-Granados, and S. Kim, "5G mmWave vehicular tracking," in *2018 52nd Asilomar Conf. Signals, Syst., Comput.* IEEE, Oct. 2018, pp. 541–547.

[105] J. Gante, G. Falcão, and L. Sousa, "Deep learning architectures for accurate millimeter wave positioning in 5G," *Neural Process. Lett.*, vol. 51, no. 1, pp. 487–514, Feb. 2020.

[106] Y. Ruan, L. Chen, X. Zhou, G. Guo, and R. Chen, "Hi-loc: Hybrid indoor localization via enhanced 5G NR CSI," *IEEE Trans. Instrum. Meas.*, vol. 71, pp. 1–15, Aug. 2022.

[107] A. Kendall and Y. Gal, "What uncertainties do we need in Bayesian deep learning for computer vision?" in *Proc. 31th Int. Conf. Neural Inf. Process. Syst.*, Dec. 2017, pp. 1–11.

[108] X. Zhang and S. Mahadevan, "Bayesian neural networks for flight trajectory prediction and safety assessment," *Decision Support Syst.*, vol. 131, no. C, Apr. 2020.

[109] Y. Pang, X. Zhao, H. Yan, and Y. Liu, "Data-driven trajectory prediction with weather uncertainties: A Bayesian deep learning approach," *Transp. Res. Part C: Emerg. Technol.*, vol. 130, p. 103326, Aug. 2021.

[110] M. A. M. Sadr, J. Gante, B. Champagne, G. Falcao, and L. Sousa, "Uncertainty estimation via Monte Carlo dropout in CNN-based mmWave MIMO localization," *IEEE Signal Process. Lett.*, vol. 29, pp. 269–273, Nov. 2022.

[111] A. Korattikara Balan, V. Rathod, K. Murphy, and M. Welling, "Bayesian dark knowledge," in *Proc. 28th Int. Conf. Neural Inf. Process. Syst.*, Jun. 2015, pp. 1–11.

[112] **B. Camajori Tedeschini**, M. Brambilla, L. Barbieri, and M. Nicoli, "Addressing data association by message passing over graph neural networks," in *2022 25th Int. Conf. Inf. Fusion (FUSION)*, Jul. 2022, pp. 01–07.

[113] **B. Camajori Tedeschini**, M. Brambilla, L. Barbieri, G. Balducci, and M. Nicoli, "Cooperative lidar sensing for pedestrian detection: Data association based on message passing neural networks," *IEEE Trans. Signal Process.*, vol. 71, pp. 3028–3042, Aug. 2023.

[114] **B. Camajori Tedeschini**, M. Brambilla, and M. Nicoli, "Message passing neural network versus message passing algorithm for cooperative positioning," *IEEE Trans. Cogn. Commun. Netw.*, vol. 9, no. 6, pp. 1666–1676, Aug. 2023.

[115] **B. Camajori Tedeschini**, S. Savazzi, R. Stoklasa, L. Barbieri, I. Stathopoulos, M. Nicoli, and L. Serio, "Decentralized federated learning for healthcare networks: A case study on tumor segmentation," *IEEE Access*, vol. 10, pp. 8693–8708, Jan. 2022.

[116] **B. Camajori Tedeschini**, S. Savazzi, and M. Nicoli, "A traffic model based approach to parameter server design in federated learning processes," *IEEE Commun. Lett.*, vol. 27, no. 7, pp. 1774–1778, May 2023.

[117] **B. Camajori Tedeschini**, S. Savazzi, and M. Nicoli, "Weighted consensus algorithms in distributed and federated learning," *submitted to IEEE Trans. Netw. Sci. and Eng.*, pp. 1–13, 2024.

[118] **B. Camajori Tedeschini**, M. Brambilla, and M. Nicoli, "Split consensus federated learning: an approach for distributed training and inference," *IEEE Access*, vol. 12, pp. 119 535–119 549, Aug. 2024.

[119] **B. Camajori Tedeschini**, M. Brambilla, M. Nicoli, and M. Z. Win, "Cooperative positioning with multi-agent reinforcement learning," in *2024 27th Int. Conf. Inf. Fusion (FUSION)*, 2024, pp. 1–7.

[120] **B. Camajori Tedeschini**, M. Brambilla, M. Nicoli, and M. Z. Win, "Multi-agent reinforcement learning for distributed cooperative positioning," *IEEE Trans. Intell. Veh.*, pp. 1–16, Oct. 2024.

[121] **B. Camajori Tedeschini**, M. Nicoli, and M. Z. Win, "On the latent space of mmWave MIMO channels for NLOS identification in 5G-advanced systems," *IEEE J. Sel. Areas Commun.*, vol. 41, no. 6, pp. 1655–1669, May 2023.

[122] **B. Camajori Tedeschini** and M. Nicoli, "Cooperative deep-learning positioning in mmWave 5G-advanced networks," *IEEE J. Sel. Areas Commun.*, vol. 41, no. 12, pp. 3799–3815, Dec. 2023.

[123] **B. Camajori Tedeschini**, G. Kwon, M. Nicoli, and M. Z. Win, "Empowering 6G positioning and tracking with Bayesian neural networks," in *ICC 2024 - 2024 IEEE Int. Conf. Commun. (ICC)*. IEEE, Jun. 2024, pp. 1–6.

[124] **B. Camajori Tedeschini**, G. Kwon, M. Nicoli, and M. Z. Win, "Real-time Bayesian neural networks for 6G cooperative positioning and tracking," *IEEE J. Sel. Areas Commun.*, vol. 42, no. 9, pp. 2322–2338, Aug. 2024.

[125] **B. Camajori Tedeschini**, M. Brambilla, L. Italiano, S. Reggiani, D. Vaccarono, M. Alghisi, L. Benvenuto, A. Goia, E. Realini, F. Grec *et al.*, "A feasibility study of 5G positioning with current cellular network deployment," *Sci. Reports*, vol. 13, no. 1, Sep. 2023.

[126] L. Barbieri, **B. Camajori Tedeschini**, M. Brambilla, and M. Nicoli, "Implicit vehicle positioning with cooperative lidar sensing," in *ICASSP 2023 - 2023 IEEE Int. Conf. Acoust., Speech Signal Process. (ICASSP)*, Jun. 2023, pp. 1–5, iSSN: 2379-190X.

[127] S. Roger, M. Brambilla, **B. Camajori Tedeschini**, C. Botella-Mascarell, M. Cobos, and M. Nicoli, "Deep-learning-based radio map reconstruction for V2X communications," *IEEE Trans. on Veh. Technol.*, pp. 1–9, Oct. 2023.

[128] L. Barbieri, **B. Camajori Tedeschini**, M. Brambilla, and M. Nicoli, "Deep learning-based cooperative LiDAR sensing for improved vehicle positioning," *IEEE Trans. Signal Process.*, vol. 72, pp. 1666–1682, Mar. 2024.

[129] U. Milasheuski, L. Barbieri, **B. Camajori Tedeschini**, M. Nicoli, and S. Savazzi, "On the impact of data heterogeneity in federated learning environments with

application to healthcare networks," in *IEEE Conf. Artif. Intell.* IEEE, Jun. 2024, pp. 1017–1023.

[130] L. Italiano, **B. Camajori Tedeschini**, M. Brambilla, and M. Nicoli, "Pedestrian positioning in urban environments with 5G technology," in *IEEE Mediterranean Commun. and Comput. Netw. Conf.* IEEE, Jun. 2024, pp. 1–6.

[131] L. Italiano, **B. Camajori Tedeschini**, M. Brambilla, H. Huang, M. Nicoli, and H. Wymeersch, "A tutorial on 5G positioning," *IEEE Commun. Surveys & Tuts*, pp. 1–48, Aug. 2024.

[132] M. Brambilla, M. Alghisi, **B. Camajori Tedeschini**, A. Fumagalli, F. Grec, L. Italiano, C. Pileggi, L. Biagi, S. Bianchi, A. Gatti *et al.*, "Integration of 5G and GNSS technologies for enhanced positioning: an experimental study," *IEEE Open J. of the Commun. Soc.*, pp. 1–20, Nov. 2024.

[133] N. Schatz, S. Kim, G. Kwon, **B. Camajori Tedeschini**, M. Ricard, T. Klein, V. Weerackody, A. Conti, and M. Z. Win, "Location verification in next-generation non-terrestrial networks," in *IEEE Military Commun. Conf.* IEEE, Nov. 2024, pp. 1–6.

[134] J. C. Morrison, N. Schatz, S. Kim, G. Kwon, **B. Camajori Tedeschini**, V. Weerackody, A. Conti, and M. Z. Win, "Sidelink-enabled cooperative localization for xG non-terrestrial networks," in *IEEE Military Commun. Conf.* IEEE, Nov. 2024, pp. 1–6.

[135] F. Meyer and M. Z. Win, "Scalable data association for extended object tracking," *IEEE Trans. Signal Inf. Process. over Netw.*, vol. 6, pp. 491–507, May 2020.

[136] R. W. Sittler, "An optimal data association problem in surveillance theory," *IEEE Trans. Mil. Electron.*, vol. 8, no. 2, pp. 125–139, Apr. 1964.

[137] S. Levine, A. Kumar, G. Tucker, and J. Fu, "Offline reinforcement learning: Tutorial, review, and perspectives on open problems," *ArXiv*, May 2020.

[138] L. V. Jospin, H. Laga, F. Boussaid, W. Buntine, and M. Bennamoun, "Hands-on Bayesian neural networks—a tutorial for deep learning users," *IEEE Comput. Intell. Mag.*, vol. 17, no. 2, pp. 29–48, May 2022.

[139] M. Welling and Y. W. Teh, "Bayesian learning via stochastic gradient Langevin dynamics," in *Proc. 28th Int. Conf. Mach. Learn.*, Jun. 2011, p. 681–688.

[140] Y. Gal and Z. Ghahramani, "Dropout as a Bayesian approximation: representing model uncertainty in deep learning," in *Proc. 33th Int. Conf. Mach. Learn.*, Jun. 2016, p. 1050–1059.

[141] C. Blundell, J. Cornebise, K. Kavukcuoglu, and D. Wierstra, "Weight uncertainty in neural networks," in *Proc. 32th Int. Conf. Mach. Learn.*, Jul. 2015, p. 1613–1622.

[142] F. Scarselli, M. Gori, Ah Chung Tsoi, M. Hagenbuchner, and G. Monfardini, "Computational capabilities of graph neural networks," *IEEE Trans. Neural Netw.*, vol. 20, no. 1, pp. 81–102, Jan. 2008.

[143] R. E. Schapire, *The Boosting Approach to Machine Learning: An Overview*. New York, NY: Springer New York, 2003, pp. 149–171.

[144] A. Sanchez-Gonzalez, N. Heess, J. T. Springenberg, J. Merel, M. A. Riedmiller, R. Hadsell, and P. W. Battaglia, "Graph networks as learnable physics engines for inference and control," in *Proc. 35th Int. Conf. Mach. Learn.*, Jul. 2018, pp. 4470–4479.

[145] J. Gilmer, S. S. Schoenholz, P. F. Riley, O. Vinyals, and G. E. Dahl, "Neural message passing for quantum chemistry," in *Proc. 34th Int. Conf. Mach. Learn. - Volume 70*. PMLR, Apr. 2017, p. 1263–1272.

[146] X. Wang, R. Girshick, A. Gupta, and K. He, "Non-local neural networks," in *2018 IEEE/CVF Conf. Comput. Vis. Pattern Recog. (CVPR)*, Dec. 2018, pp. 7794–7803.

[147] A. Santoro, D. Raposo, D. G. T. Barrett, M. Malinowski, R. Pascanu, P. Battaglia, and T. Lillicrap, "A simple neural network module for relational reasoning," in *Advances Neural Inform. Process. Syst.*, I. Guyon, U. V. Luxburg, S. Bengio, H. Wallach, R. Fergus, S. Vishwanathan, and R. Garnett, Eds., vol. 30. Curran Associates, Inc., Dec. 2017, pp. 1–16.

[148] M. Zaheer, S. Kottur, S. Ravanbakhsh, B. Poczos, R. Salakhutdinov, and A. Smola, "Deep sets," in *Advances Neural Inform. Process. Syst.*, I. Guyon, U. V. Luxburg, S. Bengio, H. Wallach, R. Fergus, S. Vishwanathan, and R. Garnett, Eds., vol. 30. Curran Associates, Inc., Dec. 2017, p. 3394–3404.

[149] A. H. Lang, S. Vora, H. Caesar, L. Zhou, J. Yang, and O. Beijbom, "Pointpillars: Fast encoders for object detection from point clouds," in *2019 IEEE/CVF Conf. Comput. Vis. Pattern Recog. (CVPR)*, Jun. 2019, pp. 12 689–12 697.

[150] C. Thapa, M. A. P. Chamikara, and S. A. Camtepe, "Advancements of federated learning towards privacy preservation: From federated learning to split learning," in *Federated Learn. Syst.*, M. H. U. Rehman and M. M. Gaber, Eds. Springer International Publishing, Jun. 2021, vol. 965, pp. 79–109.

[151] P. Vepakomma, T. Swedish, R. Raskar, O. Gupta, and A. Dubey, "No peek: A survey of private distributed deep learning," *arXiv preprint arXiv:1812.03288*, Dec. 2018.

[152] D. P. Kingma and J. Ba, "Adam: A method for stochastic optimization," in *Proc. 3rd Int. Conf. Learn. Representations*, Dec. 2015, pp. 1–15.

[153] S. Oh, J. Park, P. Vepakomma, S. Baek, R. Raskar, M. Bennis, and S.-L. Kim, "Locfedmix-SL: Localize, federate, and mix for improved scalability, convergence, and latency in split learning," in *Proc. ACM Web Conf. 2022*, Apr. 2022, pp. 3347–3357.

[154] V. Fox, J. Hightower, L. Liao, D. Schulz, and G. Borriello, "Bayesian filtering for location estimation," *IEEE Pervasive Comput.*, vol. 2, no. 3, pp. 24–33, Sep. 2003.

[155] M. Arulampalam, S. Maskell, N. Gordon, and T. Clapp, "A tutorial on particle filters for online nonlinear/non-Gaussian Bayesian tracking," *IEEE Trans. Signal Process.*, vol. 50, no. 2, pp. 174–188, Feb. 2002.

[156] M. Brambilla, D. Gaglione, G. Soldi, R. Mendrzik, G. Ferri, K. D. LePage, M. Nicoli, P. Willett, P. Braca, and M. Z. Win, "Cooperative localization and multitarget tracking in agent networks with the sum-product algorithm," *IEEE Open J. Signal Process.*, vol. 3, pp. 169–195, Mar. 2022.

[157] *Study on NR positioning support*, TR 38.855 Version 16.0.0, 3rd Generation Partnership Project (3GPP), Sophia Antipolis, France, 2019, Sep. 2019.

[158] H. L. Van Trees, *Optimum Array Processing: Part IV of Detection, Estimation, and Modulation Theory*, 1st ed. Wiley, Mar. 2002.

[159] D. Tse and P. Viswanath, *Fundamentals of wireless communication*. Cambridge University Press, 2005, oCLC: ocm57751753.

[160] X. Sun, X. Gao, G. Y. Li, and W. Han, "Single-site localization based on a new type of fingerprint for massive MIMO-OFDM systems," *IEEE Trans. Veh. Technol.*, vol. 67, no. 7, pp. 6134–6145, Jul. 2018.

[161] M. D. Zeiler and R. Fergus, "Visualizing and understanding convolutional networks," in *Comput. Vis. - ECCV 2014*, D. Fleet, T. Pajdla, B. Schiele, and T. Tuytelaars, Eds. Springer International Publishing, 2014, pp. 818–833.

[162] D. P. Kingma and M. Welling, "Auto-encoding variational Bayes," *ArXiv*, Dec. 2013.

[163] E. Parzen, "On estimation of a probability density function and mode," *The Annals of Mathematical Statistics*, vol. 33, no. 3, pp. 1065–1076, Dec. 2007.

[164] C. M. Bishop, *Pattern Recognition and Machine Learning*. Springer, Aug. 2006, vol. 4, no. 4.

[165] B. Silverman, *Density Estimation for Statistics and Data Analysis*, 1st ed. Routledge, Mar. 2018.

[166] P. Lv, Y. Yu, Y. Fan, X. Tang, and X. Tong, "Layer-constrained variational autoencoding kernel density estimation model for anomaly detection," *Knowledge-Based Systems*, vol. 196, p. 105753, Mar. 2020.