IEEE Access
Multidisciplinary | Rapid Review | Open Access Journal

Date of publication xxxx 00, 0000, date of current version xxxx 00, 0000.

Digital Object Identifier 10.1109/ACCESS.2024.0429000

Split Consensus Federated Learning: an Approach for Distributed Training and Inference

B. CAMAJORI TEDESCHINI[1] (Graduate Student Member, IEEE), M. BRAMBILLA[1] (Member, IEEE), M. NICOLI[2] (Senior Member, IEEE)

[1]Dipartimento di Elettronica, Informazione e Bioingegneria, Politecnico di Milano, Via Ponzio 34/5, 20133, Milano, Italy (e-mail: bernardo.camajori@polimi.it, mattia.brambilla@polimi.it)

[2]Department of Management, Economics and Industrial Engineering, Politecnico di Milano, Via Lambruschini, 4/B, 20156, Milano, Italy (email: monica.nicoli@polimi.it)

Corresponding author: Bernardo Camajori Tedeschini (e-mail: 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, without requiring a coordinating PS. Inspired by the iterative procedure of Message Passing Neural Networks (MPNN), the proposed SCFL framework allows distributed inference and training of the neural networks at the clients, keeping the privacy of locally stored data. We present three different strategies for SCFL implementation, each possessing distinctive features and performances, and we validate them in a cooperative positioning use case where clients, i.e., agents, use D-ML to localize themselves relying on ego and inter-agent measurements. Results show that the proposed SCFL is able to combine the computational power of all clients to train local models which closely approximate the C-ML solution at convergence.

INDEX TERMS Split consensus federated learning, split learning, federated learning, message passing neural network, consensus, cooperative positioning.

I. INTRODUCTION

A. CONTEXTUALIZATION AND BACKGROUND

In recent years, Machine Learning (ML) has expanded to 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 collection and processing of data from multiple clients is performed at a single aggregation point, 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 effectively address the increased complexity and security issues while retaining the benefits of conventional ML techniques.

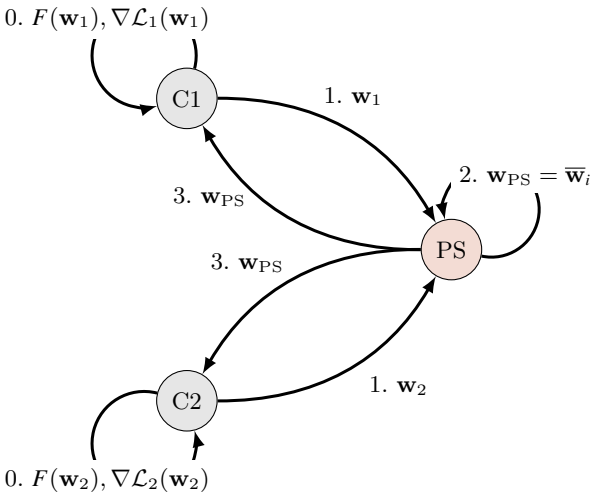
Distributed Machine Learning (D-ML) [7], [8] has emerged as a viable solution for distributing the processing and avoiding the aggregation of data at a single central entity. A popular D-ML mechanism is Federated Learning (FL) [9], [10], which allows spatially distributed clients to collaboratively train 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 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]–[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 designed to address the challenges

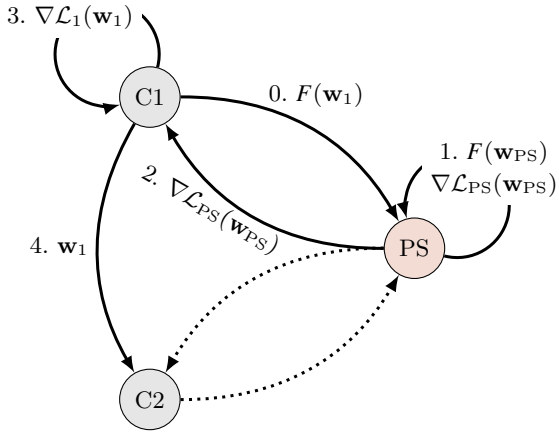
of resource-constrained setups, such as Internet of Things (IoT) networks where clients may have limited computing capabilities and energy resources. In SL, the model training and validation processes are divided between the clients and a PS, each having only a partial access/visibility to/of a specific 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 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 client. 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, there are still open problems to address, such as leakage reduction [20]–[24], non-Independent and Identical Distributed (IID) data distribution among clients [25]–[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.

B. RELATED WORKS

In this section, we discuss how the limitation of FL and SL have been addressed so far in the literature and our intuition to solve the remaining open problems.

Vanilla FL architectures present two main problems 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]–[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]–[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.

C. INTUITION TO SOLVE FL AND SL OPEN PROBLEMS

The remaining open problems of FL and SL constitute a literature gap on the unavailability of a fully-distributed methodology for both training and inference in resource-constrained networks of devices. Filling this literature gap is the main objective of this work.

For the design of a completely distributed architecture performing the training and inference tasks, we proceed as follows. 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. Indeed, the splitting of the model between clients and PS can be viewed as a smart strategy to lower the computational complexity of each node. Therefore, for distributed SL, we conceive to train a big 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. The description of how MPNN are originally used in a fully-distributed architecture without a PS is given in next section.

D. CONTRIBUTION

In this work, we propose to incorporate the message passing scheme inside the MPNN as a sequence of smashed data exchange in which clients (represented by the nodes of the graph) cannot complete a whole inference of the model but they can just perform one iteration of the message passing. 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 assures the respect of privacy concerns, just as in vanilla SL, since each client holds private labels or outputs. To accelerate the entire training process, analogous 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 of the parallelism between FL, SL and SFL, with focus on the iterative processing steps for training;
- The design of a fully-distributed SFL architecture, namely SCFL, which exploits the characteristics of centralized MPNN to perform distributed training and inference procedures between physically separated clients;
- The proposal and validation of three distributed 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 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 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.
- 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-distributed architecture, and the low complexity training, which are fundamental for deployment in privacy-sensitive, resource-constrained environments.

TABLE 1. Comparison of SL, FL, SFL and the proposed SCFL

	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

A summary of main differences between the proposed SCFL and SL, FL and SFL is provided in Table 1.

The proposed SCFL 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]–[48]. Examples of possible domains of application are within the fields of vehicular networks [49]–[52], IoT [53], [54], maritime surveillance [55], [56] and drones [57], [58].

E. 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, 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 $\mathcal{I} = \{1, \dots, I\}$. 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 $n = 1, \dots, N$ 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 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.

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 $n = 1, \dots, N$ **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

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 synthetic representation of the SFL workflow.

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

Algorithm 2 Split Learning

procedure SL(η)
  initialize $\mathbf{w}_{i,0} \quad \forall\, i \in \mathcal{I}$
  for each round $n = 1, \dots, N$ **do** ▷ Training loop
    for client $i = 1, \dots, I$ **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

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 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

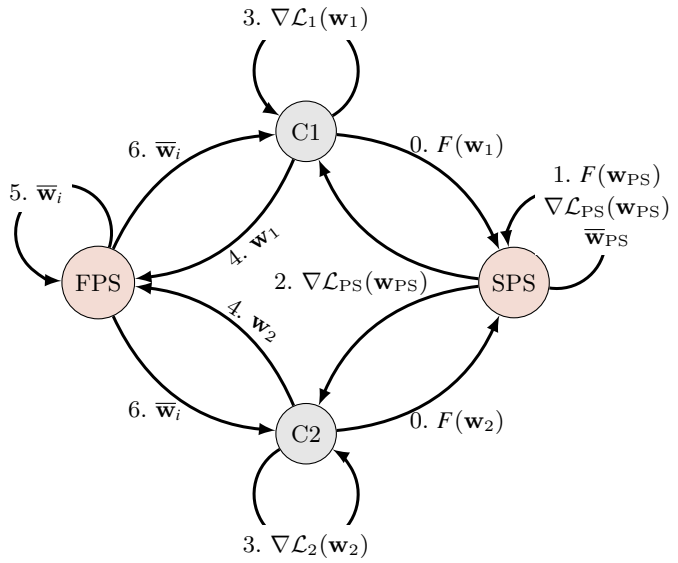


FIGURE 2. SFL framework with vanilla architecture composed of a split PS, i.e., SPS, and a federated PS, i.e., FPS. For ease of notation, step 2) is represented in the same way for both the clients.

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 pass 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 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

$$\mathbf{v}_{i,n}^{(t)} = g_v\left(\mathbf{v}_{i,n}^{(t-1)}, \Phi(\{\mathbf{e}_{j\to i,n}^{(t)}\}_{j\in\mathcal{N}_i})\right), \tag{6}$$

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 $\widehat{\mathbf{y}}_{i,n}^{(t)}$ 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.

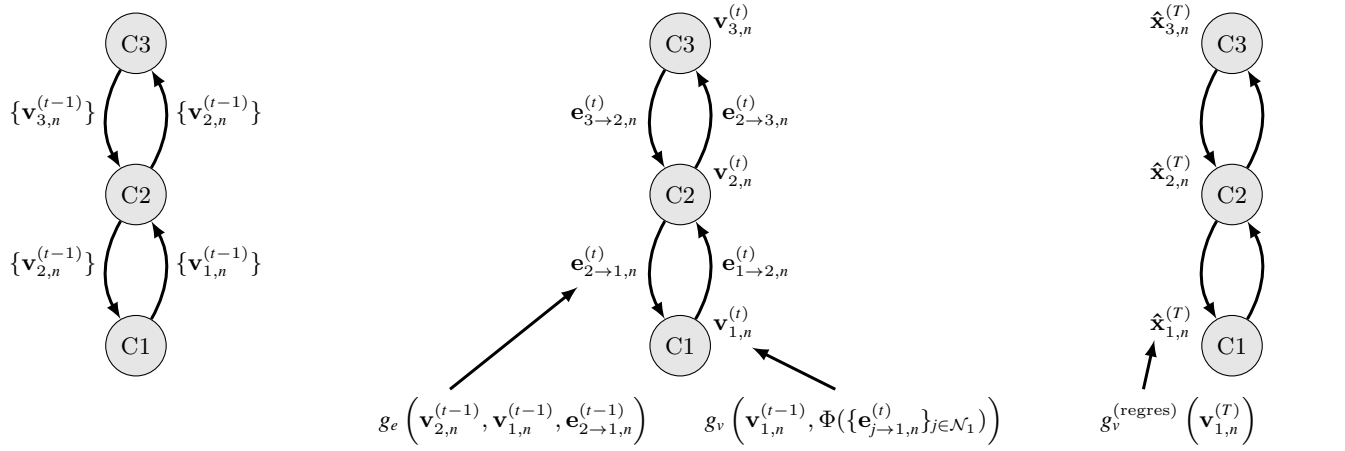


FIGURE 3. **Distribute inference procedure. a) node embedding exchange. b) edge and node embedding update. c) prediction.**

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:
$$\mathbf{e}_{j\to i,n}^{(t)} = g_e\left(\mathbf{v}_{j,n}^{(t-1)}, \mathbf{v}_{i,n}^{(t-1)}, \mathbf{e}_{j\to i,n}^{(t-1)}\right), \quad \forall j \in \mathcal{N}_i. \quad (9)$$
Subsequently, the node embeddings are updated as:
$$\mathbf{v}_{i,n}^{(t)} = g_v\left(\mathbf{v}_{i,n}^{(t-1)}, \Phi\left(\{\mathbf{e}_{j\to i,n}^{(t)}\}_{j\in\mathcal{N}_i}\right)\right). \quad (10)$$
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:
$$\widehat{\mathbf{y}}_{i,n} = \widehat{\mathbf{y}}_{i,n}^{(T)} = g_v^{(\text{regres})}\left(\mathbf{v}_{i,n}^{(T)}\right). \quad (11)$$

A visualization 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. 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

Algorithm 3 3-Steps Strategy

procedure 3SS($\mathcal{N}_i, \eta$) ▷ Run on client i
  initialize $\mathbf{w}_{i,0} \leftarrow$ client i
  for each round $n = 1, \dots, N$ **do** ▷ Training loop
    for each message passing iter $t = 1, \dots, T$ **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 $\nabla\mathcal{L}_{i,n}^{(t)}(\mathbf{w}_{i,n})$ ▷ Backward-pass
      broadcast $\nabla\mathcal{L}_{i,n}^{(t)}(\mathbf{w}_{i,n})$
      receive $\{\nabla\mathcal{L}_{j,n}^{(t)}(\mathbf{w}_{j,n})\}_{j\in\mathcal{N}_i}$
      $\nabla\overline{\mathcal{L}}_{i,n}^{(t)}(\mathbf{w}_{i,n}) \leftarrow \sum_{j'\in\mathcal{N}_{i*}} S_{j'}\nabla\mathcal{L}_{j',n}^{(t)} / \sum_{j\in\mathcal{N}_{i*}} S_j$
      $\mathbf{w}_{i,n} \leftarrow \mathbf{w}_{i,n} - \eta\nabla\overline{\mathcal{L}}_{i,n}^{(t)}(\mathbf{w}_{i,n})$ ▷ Local SGD
    end for
  end for
end procedure

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 visualization of the SCFL - distributed training 3SS strategy 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 visualization of the SCFL - distributed training 2SS strategy 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

Algorithm 4 2-Steps Strategy

procedure 2SS($\mathcal{N}_i, \eta$) ▷ Run on client i
  initialize $\mathbf{w}_{i,0} \leftarrow$ client i
  for each round $n = 1, \dots, N$ **do** ▷ Training loop
    for each message passing iter $t = 1, \dots, T$ **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 $\nabla\mathcal{L}_{i,n}^{(t)}(\mathbf{w}_{i,n})$ ▷ Backward-pass
      $\mathbf{w}_{i,n} \leftarrow \mathbf{w}_{i,n} - \eta\nabla\mathcal{L}_{i,n}^{(t)}(\mathbf{w}_{i,n})$ ▷ 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($\mathcal{N}_i, \eta$) ▷ Run on client i
  initialize $\mathbf{w}_{i,0} \leftarrow$ client i
  for each round $n = 1, \dots, N$ **do** ▷ Training loop
    for each message passing iter $t = 1, \dots, T$ **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

the centralized loss in (8). A visualization of the SCFL - distributed training D-MPNN strategy 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 fully-distributed FL. 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 the CP of 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).

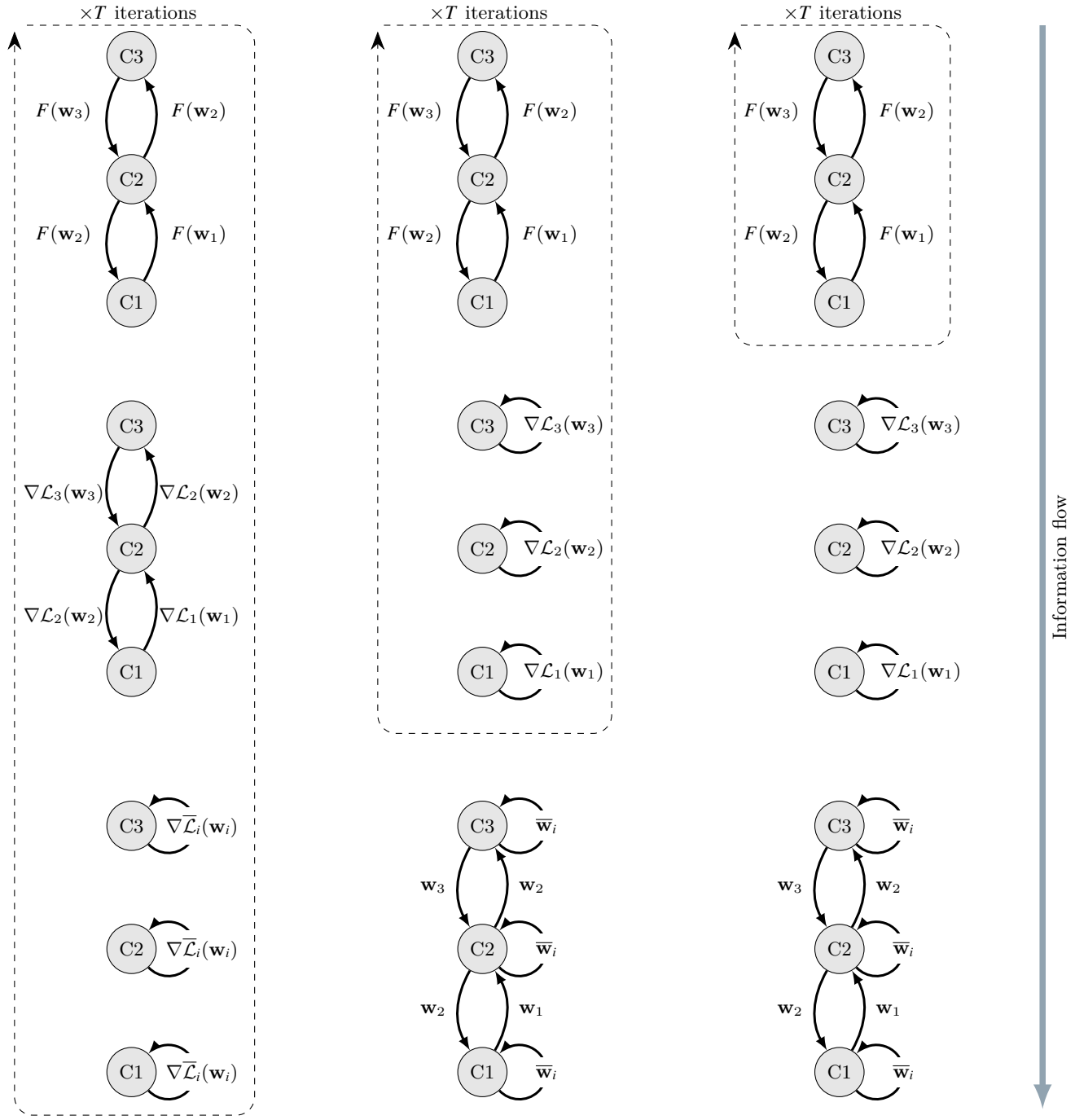
B. SIMULATION SCENARIO

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., $\mathbf{w}_{i,n}^{(\mathrm{A})} \sim \mathcal{N}(\mathbf{0}_4, \mathbf{C}_{\mathbf{w}^{(\mathrm{A})}})$, 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, 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 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



(a) SCFL - distributed training 3SS

(b) SCFL - distributed training 2SS

(c) SCFL - distributed training D-MPNN

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.**

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,

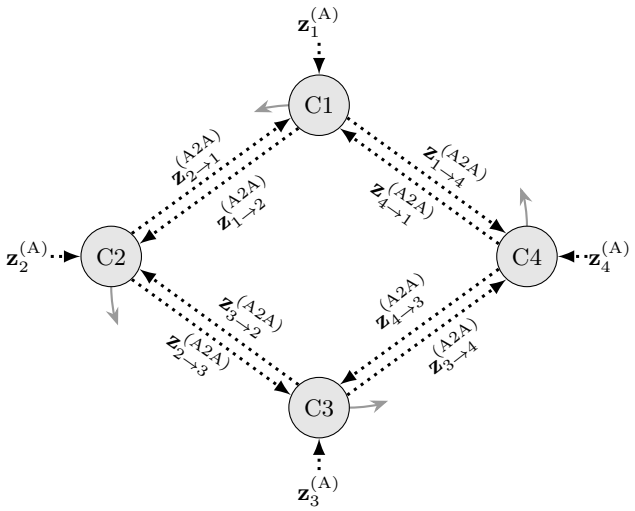


FIGURE 5. **CP task with four agents, represented by circles. The measurements are represented as black dotted arrows, while trajectory paths are indicated with colored solid arrows. For ease of notation, we omit the epoch index *n*.**

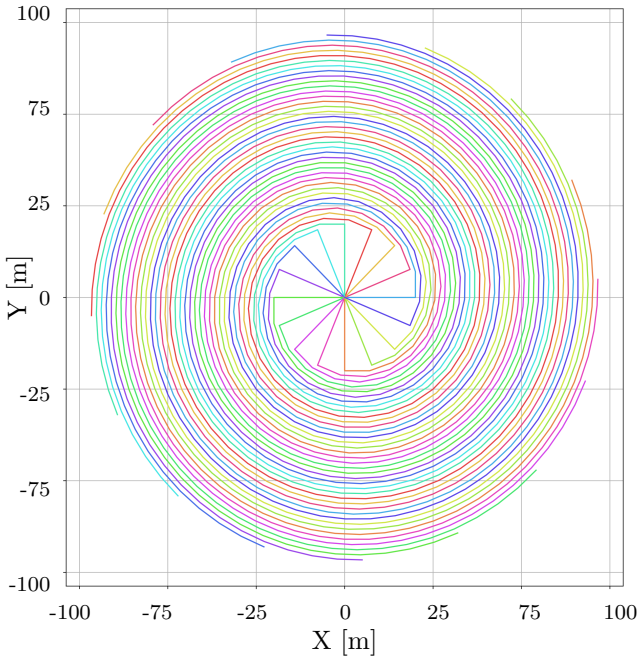


FIGURE 6. **Example of spiral scenario composed of 16 cooperating agents represented by different colors.**

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).

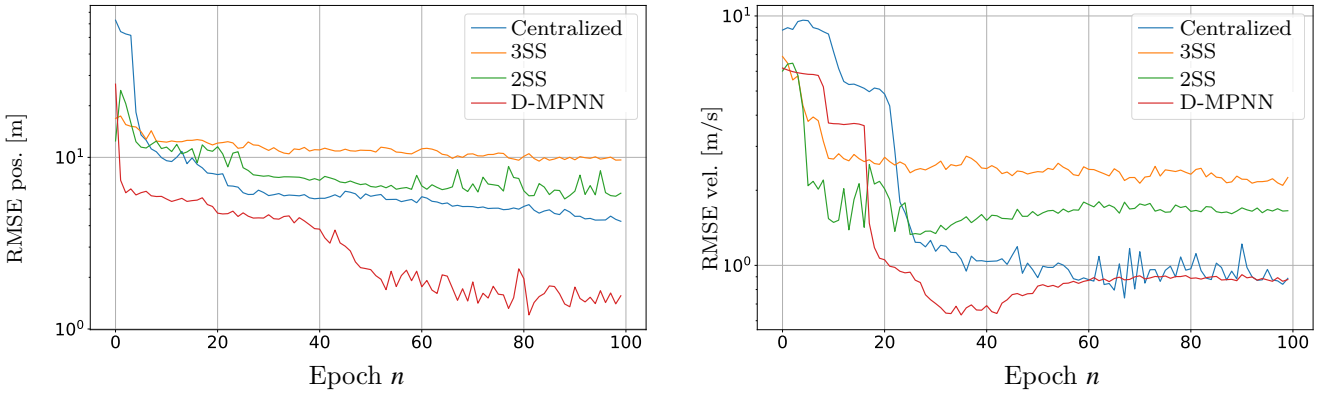
Observing the results, we note that at corresponding epoch, the best performances are reached by the distributed D-MPNN and the centralized solution, followed by 2SS and 3SS implementations. This result suggests that the cooperative forwarding pass of the training should be performed for all message 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 with D-MPNN outperforms even 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.

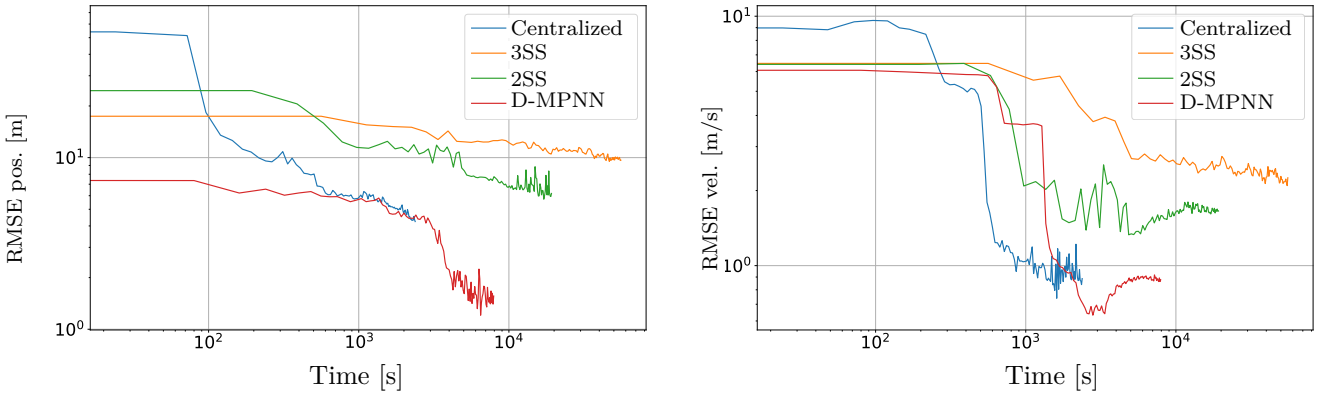
2) Convergence

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 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 local biased 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.



(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.

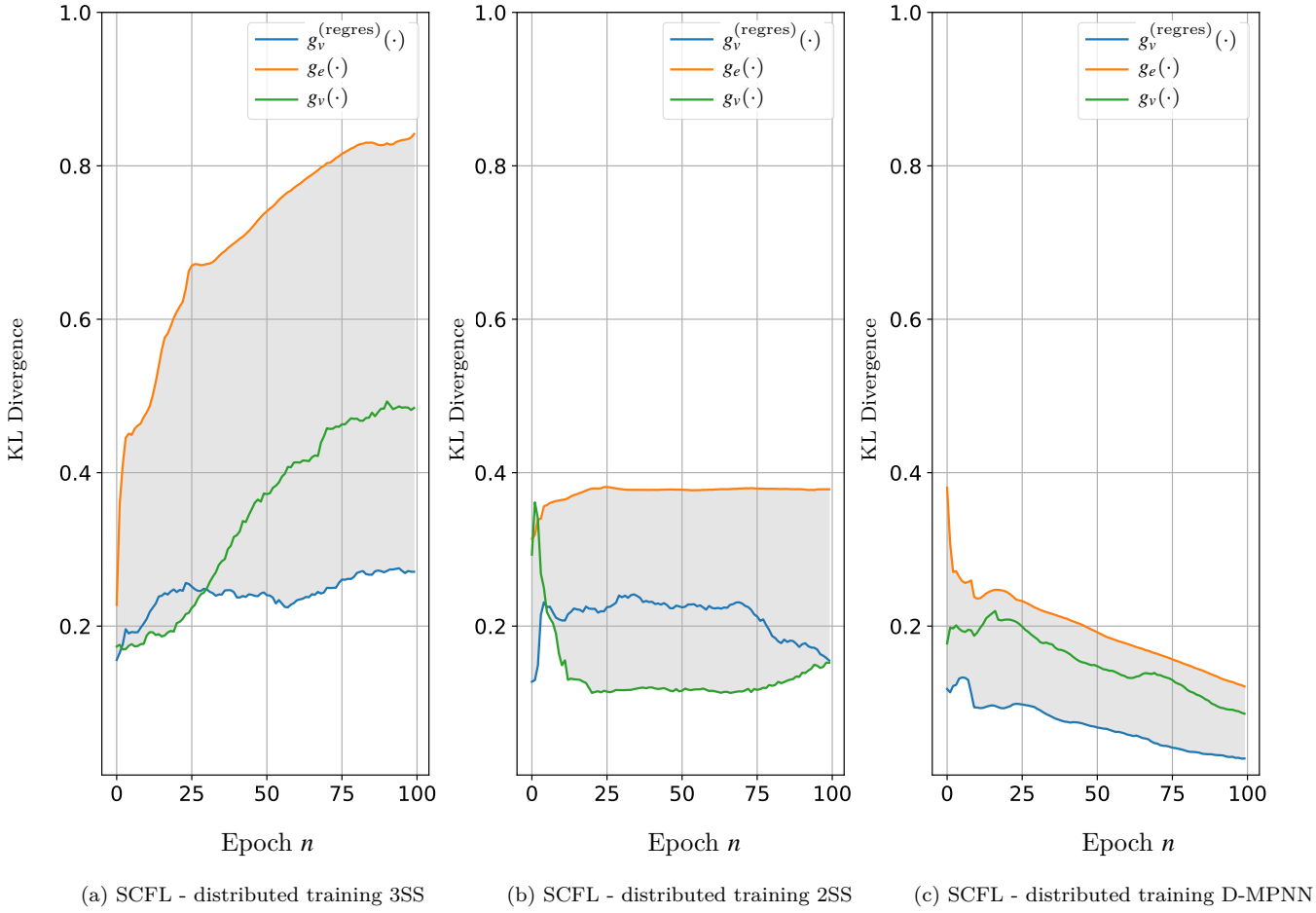
3) Baseline comparison

In this assessment, we compare our proposed D-MPNN 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.

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 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



(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.**

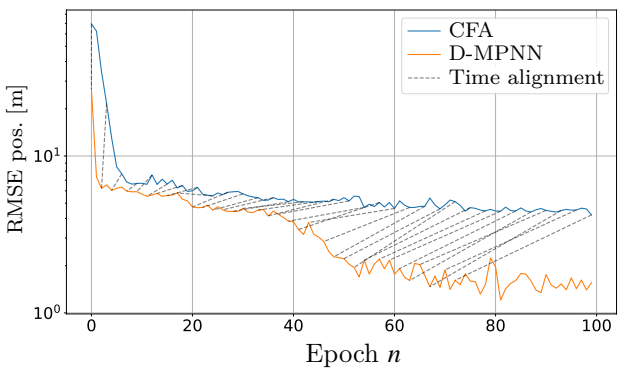


FIGURE 9. **Comparison of the proposed SCFL D-MPNN method with respect to the CFA algorithm in terms of RMSE. The points related to the same training time are connected by dashed lines.**

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.

REFERENCES

[1] A. Qayyum, J. Qadir, M. Bilal, and A. Al-Fuqaha, "Secure and robust machine learning for healthcare: A survey," *IEEE Reviews in Biomedical Engineering*, 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. 97 531–97 554, 2022.

[4] M. F. Dixon, I. Halperin, and P. Bilokon, *Machine Learning in Finance: From Theory to Practice*. Springer International Publishing, 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, 2019.

[6] L. Barbieri, S. Savazzi, M. Brambilla, and M. Nicoli, "Decentralized federated learning for extended sensing in 6G connected vehicles," *Vehicular Communications*, vol. 33, p. 100396, 2022.

[7] 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*, 2018.

[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*, M. H. U. Rehman and M. M. Gaber, Eds. Springer International Publishing, 2021, vol. 965, pp. 79–109.

[9] J. Konečný, B. McMahan, and D. Ramage, "Federated optimization: Distributed optimization beyond the datacenter," *arXiv preprint arXiv:1511.03575*, 2015.

[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 *International Conference on Artificial Intelligence and Statistics*, 2016, pp. 1273–1282.

[11] Q. Yang, Y. Liu, T. Chen, and Y. Tong, "Federated machine learning: Concept and applications," *ACM Transactions on Intelligent Systems and Technology*, vol. 10, no. 2, pp. 1–19, 2019.

[12] T. Li, A. K. Sahu, A. Talwalkar, and V. Smith, "Federated learning: Challenges, methods, and future directions," *IEEE Signal Processing Magazine*, vol. 37, no. 3, pp. 50–60, 2020.

[13] 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.

[14] B. Camajori Tedeschini, S. Savazzi, and M. Nicoli, "A traffic model based approach to parameter server design in federated learning processes," *IEEE Communications Letters*, vol. 27, no. 7, pp. 1774–1778, 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. 22 155–22 173, 2023.

[16] L. Barbieri, S. Savazzi, and M. Nicoli, "Communication-efficient distributed learning in V2X networks: Parameter selection and quantization," in *GLOBECOM 2022 - 2022 IEEE Global Communications Conference*, 2022, pp. 603–608.

[17] O. Gupta and R. Raskar, "Distributed learning of deep neural network over multiple agents," *Journal of Network and Computer Applications*, vol. 116, pp. 1–8, 2018.

[18] K. Palanisamy, V. Khimani, M. H. Moti, and D. Chatzopoulos, "Spliteasy: A practical approach for training ML models on mobile devices," in *Proceedings of the 22nd International Workshop on Mobile Computing Systems and Applications*, 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," *arXiv preprint arXiv:1909.09145*, 2019.

[20] T. Titcombe, A. J. Hall, P. Papadopoulos, and D. Romanini, "Practical defences against model inversion attacks for split neural networks," *arXiv preprint arXiv:2104.05743*, 2021.

[21] O. Li *et al.*, "Label leakage and protection in two-party split learning," *arXiv preprint arXiv:2102.08504*, 2022.

[22] E. Erdoğan, A. Küpçü, and A. E. Çiçek, "Unsplit: Data-oblivious model inversion, model stealing, and label inference attacks against split learning," in *Proceedings of the 21st Workshop on Privacy in the Electronic Society*, 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*, M. Tuba, S. Akashe, and A. Joshi, Eds. Springer Nature Singapore, 2022, vol. 321, pp. 103–112.

[24] D. Pasquini, G. Ateniese, and M. Bernaschi, "Unleashing the tiger: Inference attacks on split learning," in *Proceedings of the 2021 ACM SIGSAC Conference on Computer and Communications Security*, 2021, pp. 2113–2129.

[25] Y. Li and X. Lyu, "Convergence analysis of split learning on non-IID data," *arXiv preprint arXiv:1511.03575*, 2023.

[26] Y. Gao *et al.*, "Evaluation and optimization of distributed machine learning techniques for internet of things," *IEEE Transactions on Computers*, vol. 71, no. 10, pp. 2538–2552, 2022.

[27] Y. Gao *et al.*, "End-to-end evaluation of federated learning and split learning for internet of things," in *2020 International Symposium on Reliable Distributed Systems (SRDS)*, 2020, pp. 91–100.

[28] I. Hegedűs, G. Danner, and M. Jelasity, "Gossip learning as a decentralized alternative to federated learning," in *Distrib. Appl. Interoperable Syst.*, J. Pereira and L. Ricci, Eds. Cham: Springer International Publishing, 2019, vol. 11534, 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 of Things Journal*, vol. 7, no. 5, pp. 4641–4654, 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, 2020.

[31] H. Choi and I. V. Bajić, "Deep feature compression for collaborative object detection," in *2018 25th IEEE International Conference on Image Processing (ICIP)*, 2018, pp. 3743–3747, iSSN: 2381-8549.

[32] Y. Kang *et al.*, "Neurosurgeon: Collaborative intelligence between the cloud and mobile edge," *ACM SIGARCH Computer Architecture News*, vol. 45, no. 1, pp. 615–629, 2017.

[33] S. Teerapittayanon, B. McDanel, and H. Kung, "Distributed deep neural networks over the cloud, the edge and end devices," in *2017 IEEE 37th International Conference on Distributed Computing Systems (ICDCS)*, 2017, pp. 328–339, iSSN: 1063-6927.

[34] J. Jeon and J. Kim, "Privacy-sensitive parallel split learning," in *2020 International Conference on Information Networking (ICOIN)*, 2020, pp. 7–9.

[35] C. Thapa, P. C. Mahawaga Arachchige, S. Camtepe, and L. Sun, "SplitFed: When federated learning meets split learning," *Proceedings of the AAAI Conference on Artificial Intelligence*, vol. 36, no. 8, pp. 8485–8493, 2022.

[36] W. Wu *et al.*, "Split learning over wireless networks: Parallel design and resource management," *IEEE Journal on Selected Areas in Communications*, vol. 41, no. 4, pp. 1051–1066, 2023.

[37] S. Oh *et al.*, "Locfedmix-SL: Localize, federate, and mix for improved scalability, convergence, and latency in split learning," in *Proceedings of the ACM Web Conference 2022*, 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 and Applications*, vol. 83, no. 10, pp. 28 891–28 911, 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 *Proceedings of the Conference on Research in Adaptive and Convergent Systems*, 2022, pp. 131–138.

[40] F. Scarselli, M. Gori, Ah Chung Tsoi, M. Hagenbuchner, and G. Monfardini, "Computational capabilities of graph neural networks," *IEEE Transactions on Neural Networks*, vol. 20, no. 1, pp. 81–102, 2009.

[41] F. Scarselli, M. Gori, Ah Chung Tsoi, M. Hagenbuchner, and G. Monfardini, "The graph neural network model," *IEEE Transactions on Neural Networks*, vol. 20, no. 1, pp. 61–80, 2009.

[42] B. Camajori Tedeschini, M. Brambilla, L. Barbieri, and M. Nicoli, "Addressing data association by message passing over graph neural networks," in *2022 25th International Conference on Information Fusion (FUSION)*, 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 Transactions on Signal Processing*, 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 Transactions on Cognitive Communications and Networking*, vol. 9, no. 6, pp. 1666–1676, 2023.

[45] M. Brambilla *et al.*, "Cooperative localization and multitarget tracking in agent networks with the sum-product algorithm," *IEEE Open Journal of Signal Processing*, 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 Processing Magazine*, vol. 22, no. 4, pp. 54–69, 2005.

[47] M. Z. Win *et al.*, "Network localization and navigation via cooperation," *IEEE Communications Magazine*, vol. 49, no. 5, pp. 56–62, 2011.

[48] M. Z. Win, Y. Shen, and W. Dai, "A theoretical foundation of network localization and navigation," *Proceedings of the IEEE*, vol. 106, no. 7, pp. 1136–1165, 2018.
[49] G. Soatti, M. Nicoli, N. Garcia, B. Denis, R. Raulefs, and H. Wymeersch, "Implicit cooperative positioning in vehicular networks," *IEEE Transactions on Intelligent Transportation Systems*, vol. 19, no. 12, pp. 3964–3980, 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 Transactions on Intelligent Transportation Systems*, vol. 21, no. 4, pp. 1646–1663, 2020.
[51] L. Barbieri, B. C. Tedeschini, M. Brambilla, and M. Nicoli, "Deep learning-based cooperative LiDAR sensing for improved vehicle positioning," *IEEE Transactions on Signal Processing*, vol. 72, pp. 1666–1682, 2024.
[52] B. Camajori Tedeschini and M. Nicoli, "Cooperative deep-learning positioning in mmwave 5G-advanced networks," *IEEE Journal on Selected Areas in Communications*, 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 of Things Journal*, vol. 6, no. 4, pp. 6548–6555, 2019.
[54] A. Conti, S. Mazuelas, S. Bartoletti, W. C. Lindsey, and M. Z. Win, "Soft information for localization-of-things," *Proceedings of the IEEE*, vol. 107, no. 11, pp. 2240–2264, 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]," *Proceedings of the IEEE*, vol. 110, no. 10, pp. 1532–1537, 2022.
[56] Y. Li, Y. Wang, W. Yu, and X. Guan, "Multiple autonomous underwater vehicle cooperative localization in anchor-free environments," *IEEE Journal of Oceanic Engineering*, vol. 44, no. 4, pp. 895–911, 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 Vehicular Technology Magazine*, vol. 12, no. 3, pp. 73–82, 2017.
[58] A. Guerra, F. Guidi, D. Dardari, and P. M. Djurić, "Networks of UAVs of low complexity for time-critical localization," *IEEE Aerospace and Electronic Systems Magazine*, vol. 37, no. 10, pp. 22–38, 2022.
[59] D. P. Kingma and J. Ba, "Adam: A method for stochastic optimization," *arXiv preprint arXiv:1412.6980*, 2017.
[60] J. Gilmer, S. S. Schoenholz, P. F. Riley, O. Vinyals, and G. E. Dahl, "Neural message passing for quantum chemistry," in *Proceedings of the 34th International Conference on Machine Learning*, 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," *Information Fusion*, vol. 53, pp. 289–304, 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.