Machine Intelligence Research
www.mi-research.net

19(4), August 2022, 319-330
DOI: 10.1007/s11633-022-1338-z

A Weighted Average Consensus Approach for Decentralized Federated Learning

Alessandro Giuseppi[1] Sabato Manfredi[2] Antonio Pietrabissa[1]

[1]Department of Computer, Control, and Management Engineering, University of Rome La Sapienza, Rome 00185, Italy

[2]Department of Electrical Engineering and Information Technology, University of Naples Federico II, Naples 80125, Italy

Abstract: Federated learning (FedL) is a machine learning (ML) technique utilized to train deep neural networks (DeepNNs) in a distributed way without the need to share data among the federated training clients. FedL was proposed for edge computing and Internet of things (IoT) tasks in which a centralized server was responsible for coordinating and governing the training process. To remove the design limitation implied by the centralized entity, this work proposes two different solutions to decentralize existing FedL algorithms, enabling the application of FedL on networks with arbitrary communication topologies, and thus extending the domain of application of FedL to more complex scenarios and new tasks. Of the two proposed algorithms, one, called FedLCon, is developed based on results from discrete-time weighted average consensus theory and is able to reconstruct the performances of the standard centralized FedL solutions, as also shown by the reported validation tests.

Keywords: Federated learning (FedL), deep learning, federated averaging (FedAvg), machine learning (ML), artificial intelligence, discrete-time consensus, distributed systems.

Citation: A. Giuseppi, S. Manfredi, A. Pietrabissa. A weighted average consensus approach for decentralized federated learning. *Machine Intelligence Research*, vol.19, no.4, pp.319–330, 2022. http://doi.org/10.1007/s11633-022-1338-z

1 Introduction

Federated learning (FedL) is a specialized branch of machine learning (ML) developed to solve problems in which, for privacy or security reasons, it is not possible to gather the available data into a single knowledge base. In such a setting, the goal of a FedL training process is to let the clients of the federation cooperate by sharing their knowledge, while avoiding any data exchange.

To actuate this knowledge-sharing procedure, the information exchanged among the clients is typically limited to the parameters of their ML predictor (e.g., deep neural networks (DeepNNs)), trained on their locally available data. A model averaging procedure[1] is then typically deployed so that the knowledge of the clients contained in the shared parameters can be propagated into the rest of the federation.

Most of the FedL solutions rely on the availability of a centralized server that collects all the data and coordinates the entire learning process, typically consisting of a model averaging procedure, i.e., averaging the trainable parameters of the clients′ ML predictors. This architecture allows for communication-efficient FedL solutions, as the information exchanges occur only between clients and the server, but at the same time imposes the significant design constraint of having a centralized entity that has to be trusted by the entire federation. The degree of trust that the federation is required to have towards this central entity may be too demanding for real-world applications, such as in healthcare facilities: legal regulations and commercial agreements often even prevent all kinds of data exchanges between two entities.

When dealing with a federation of critical institutions, it is worth studying FedL algorithms that are able to cope with decentralized federations characterized by point-to-point client agreements (i.e., with sparse communication topologies), as depicted in Fig. 1. This work considers such a scenario and thus will focus on the performance and convergence aspects of the algorithm that will be assured by the results of the consensus theory. It is expected for scenarios of this kind to become the utmost criticality in the future, since a reserved and secure information exchange is an enabling technology for fast response to emergency situations, and significantly boosts research cooperation opportunities among parties that are interested/obliged not to share their data directly.

Besides the fact that the centralized server that oversees the FedL process must be trusted by the entire federation, it also represents a single point of failure, as the federated entities have no means of verifying the beha-

Research Article
Manuscript received February 23, 2022; accepted May 5, 2022
Recommended by Associate Editor Li-Wei Wang
Colored figures are available in the online version at https://link.springer.com/journal/11633

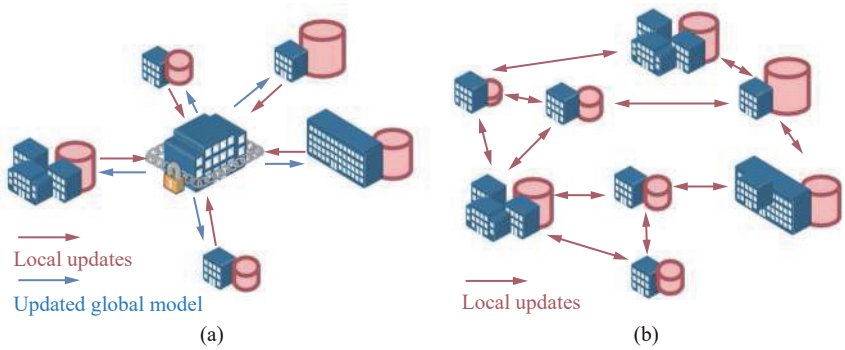

Fig. 1 Architectures of federated learning systems: (a) Standard federated learning system with a secure server (left); (b) Example of a decentralized FedL setting with bi-directional point-to-point communications (right).

viour of the server and do not communicate with one another.

With these motivations, this work aims at developing fully-decentralized solutions that are able to extend the FedL paradigm to a group of federated entities characterized by sparse and arbitrary communication agreements. The highlights and main contributions of the work are as follows:

1) The extension of standard FedL algorithms, and in particular the original federated averaging (FedAvg)[1], to a fully decentralized setting that does not require a coordinating server.

2) The design of two decentralized FedL algorithms: Decentralized federated averaging (DecFedAvg), a direct decentralization of FedAvg, and consensus-based federated learning (FedLCon), a more complex solution derived from discrete-time averaging consensus theory.

3) FedLCon relies on classic results from consensus theory, originally obtained for the control of dynamical systems, that enable its deployment on federations with arbitrary communication topologies – The topology, however, impacts the communication overhead with respect to DecFedAvg.

4) The FedLCon approach will be shown to be compatible with any FedAvg-like algorithm, making it suitable to seamlessly decentralise most of the latest results from the literature, such as federated proximal (FedProx)[2]; furthermore, FedLCon will be proven to be able to reconstruct with arbitrary precision the performances reached by any standard federated approach.

5) Validating examples are discussed, demonstrating the applicability and performance properties of the proposed algorithms in various standard settings.

The operative requirements that led to the design of the proposed algorithms were identified in the scope of an e-health project, FedMedAI (federated medical AI), conducted in collaboration with the Italian National Institute of Health.

The remainder of the paper is organized as follows. Section 2 discusses the relevant works in the literature. Section 3 provides the reader with some background on FedL and discrete-time weighted consensus theory. Section 4 presents the two proposed algorithms. Section 5 validates the algorithms on two test scenarios. Section 6 draws the conclusions and presents possible future works.

2 Related works

Contrary to standard distributed learning solutions, FedL does not envisage any coordinated split or redistribution of the data and is specifically designed to analyze data partitioned as it is, i.e., close to its sources. FedL was originally proposed as FedAvg in [1, 3] to address a problem in which a group of smartphones cooperates by sharing their knowledge without disclosing any data from their users.

Among the first and most impactful applications of FedL for the Internet of things (IoT)/edge computing setting[4, 5], we mention mobile keyboard predictions[6–8] and distributed image analytics/vision[9, 10]. For its privacy-preserving properties, FedL has then been applied also in scenarios in which organisations and institutions cooperate to analyze complex and highly confidential data, as typical in the healthcare domain[11, 12], and has found new applications in several fields[13], such as 3D connectivity-based heterogeneous networks enabled by aerial drones[14] and Industry 4.0[15]. More general architectures, designed to cope with federations in which the various clients do not have a common data structure and/or feature space, were proposed under the name of vertical FedL[16], whereas the standard FedL setting, that is the one considered in this work, has also been referred to as horizontal FedL.

Over the years, several improvements have been proposed from the original formulation[17, 18], mainly covering aspects related to privacy and security enhancements[19–22] to prevent direct or indirect data leakage[23–25], and to reduce the communication cost associated with distributed training[26–29]. Nevertheless, the common characteristic that all the algorithms presented in the literature share with the original formulation is the ability to deal with data distributed over clients in a non-independent and identically distributed (IID) and imbalanced way. The focus on such a setting is due to the distributed nature of the data sources that characterize the

considered tasks, as they are often geographically dislocated. The spatial distribution of the sources is paired with different local data distributions and a high variance in the number of the available samples.

The core of FedAvg and, in general, of FedL is a model averaging procedure, in which the DeepNNs trained by the federated clients on their locally available data are collected and averaged by a centralized server. After the averaging, the server propagates the resulting DeepNN to the federation clients. This procedure proved to be a successful strategy for massive scenarios, such as IoT/edge computing systems, but may be limiting in other settings. For instance, if the federation is constituted by a limited number of entities with no computation or communication constraints other than the requirement of not sharing any data, the communication-efficiency related features of the algorithm may be removed for better performance and complete decentralization. We mention that the typical target use cases of FedL involve a private service provider responsible for both data harvesting and analysis, whereas a decentralized setting is a more suitable choice for situations in which different public or privately owned entities cooperate.

The removal of the centralized server has been already investigated in the literature, as in [30], where a peer-to-peer gossip algorithm was proposed to exchange portions of the DeepNN weights among computing nodes with limited bandwidth and improve the overall communication efficiency of the federation. A similar goal is pursued Jiang and Hu[31], who proposed an algorithm based on a partial aggregation of the gradients produced by the training nodes. Works such as [32, 33] investigated how blockchain technology may be used in the FedL setting[32]; proposed a specialized privacy-preserving blockchain (called LearningChain) to sustain a decentralized stochastic gradient descent training algorithm[33]; developed a FedL algorithm in which the averaged models are collected on a blockchain after a decentralized committee of computing nodes reaches a consensus (i.e., an agreement) regarding the validity of the updates received from rest of the federation.

Contrary to the cited approaches, this paper focuses on developing a solution to attain the exact decentralization of existing FedL algorithms, enabling the seamless deployment of such solutions in new scenarios and applications. The algorithms of this work are hence not directly related to the design of new distributed training procedures, but instead provide a framework for the complete decentralisation of FedL algorithms.

For the sake of presentation clarity, the two algorithms presented in this work, particularly FedLCon, were specialized for decentralizing the FedAvg algorithm. The reason behind this choice is twofold: Many of the newer algorithms available in the literature, such as FedProx[2], are derived from it; most of the latest results related to communication efficiency and privacy in FedL, like those discussed in [17, 27], are typically compliant with any algorithm that shares the same structure of FedAvg. Nevertheless, we mention that the FedLCon algorithm may be specialized to decentralise other FedL algorithms, provided that they require the presence of a centralised model averaging procedure that is replaced by a so-called consensus round, as it will be clarified in Section 4.2 and in Remark 2.

Consensus is a fundamental paradigm of multi-agent systems, in which a set of communicating systems exchange information (e.g., their internal state and sensor readings) over a communication topology to estimate/control variables of interest (see, e.g., [34, 35] and the references therein). As many fundamental problems over networks can be reduced to an underlying consensus framework for multi-agent systems (including opinion dynamics[36], formation control[37] and synchronization of the electric power grid[38]), FedLCon can be seen as a consensus problem, as the clients in the federation aim at obtaining a common DeepNN, able to solve the given task by combining their local information.

3 Background

3.1 Background on federated learning

Let I be the set of N clients. Considering client $i \in I$, let w_i be the vector of trainable parameters/weights of its DeepNN and let $D_i = \{(\alpha_n, \beta_n), \quad n \in \{1, \cdots, | D_i |\}\}$ be the dataset containing its available input-output pairs. We denote the total data available as $\mathbb{D} = \bigcup_i D_i$, and we assume that the clients share the same DeepNN architecture, implying that the cardinality of the weight vectors is the same, i.e., $| w_i | = | w_j |$ for all $i, j \in I$.

In the federation, the client i is trained to minimize the so-called loss function $l_i((\alpha_n, \beta_n) \mid w_i)$ over the entire dataset D_i. Loss functions are used to quantify the quality of the output produced by DeepNNs when given as input the generic $\alpha_n \in D_i$ against the corresponding ground-truth value β_n (e.g., typical choices for loss functions are mean squared error for regression tasks or categorical cross-entropy for classification ones).

By setting

$$L_i(w_i) = L_i(D_i \mid w_i) = \frac{1}{| D_i |} \sum_{(\alpha_n, \beta_n) \in D_i} l_i((\alpha_n, \beta_n) \mid w_i) \qquad (1)$$

as the loss function of the client i over its entire dataset D_i, the goal of the federation is then to find the optimal vector of parameters w^* that, when shared by all clients, solves the minimization problem with the joint cost function defined as[39]

$$\min_{w} \mathcal{L}(w) := \sum_{i \in I} p_i L_i(w) \tag{2}$$

with $p_i = \mid D_i \mid / \mid \mathbb{D} \mid$.

In the standard ML setting, a centralized system directly deals with the optimization (2): The availability of the whole $\mathbb{D}$ enables the computation of the gradient $\nabla\mathcal{L}(w)$. Instead, in the distributed setting, the gradient $\nabla\mathcal{L}(w)$ has to be estimated starting on the gradients of the clients $\nabla L_i(w_i)$.

In standard (non-federated) distributed learning, data can be distributed arbitrarily by a centralized entity over the clients. The typical assumption for this distribution is that the datasets D_i are IID with respect to $\mathbb{D}$, implying $\mathbb{E}[L_i(w)] = \mathcal{L}(w)$. In practice, under this assumption, $L_i(w)$ provides a good approximation of $\mathcal{L}(w)$[1] and the locally computed gradients $\nabla L_i(w_i)$ can be averaged to reconstruct $\nabla L(w)$.

On the contrary, in the federated setting, such IID hypothesis cannot be assumed, as the training data is processed without any redistribution, and $L_i(w)$ could provide an arbitrarily bad approximation of $\mathcal{L}(w)$. For this reason, in the original FedL algorithm, FedAvg[1, 3], McMahan et al. proposed a round-based iterative procedure for model averaging.

FedAvg is divided into two main phases, which are repeated iteratively. In the first phase (local training), the server selects a subset of clients that update the weights of their DeepNNs by training on their local dataset D_i with a gradient descent update rule:

$$\tilde{w}_i(t) = w_i(t-1) - \eta \nabla L_i(w_i(t-1)) \tag{3}$$

where $0 < \eta < 1$ is the learning rate, and $\tilde{w}_i(t)$ is the locally updated[1] weight vector of the DeepNN of agent i at time-step t. We mention that, in the FedL setting, it is typically assumed that all clients share a common initial weight vector, i.e., $w_i(0) = w_0,\ \forall i \in I$[1].

In the second phase (centralized averaging), the server collects the $\tilde{w}_i$s, computes the weight vector $w(t)$ as the weighted average:

$$w(t) = \sum_{i \in I} p_i \tilde{w}_i(t) \tag{4}$$

and propagates the weight vector $w(t)$ to all the clients:

$$w_i(t) = w(t),\ \forall\ i \in I. \tag{5}$$

[1] Actually, in practice, the local weight update is performed iteratively over E training epochs using a variation of gradient descent (mini-batch gradient descent) that splits D_i into a set of mini-batches. Equation (3) exemplifies the update rule with $E = 1$ and over the complete dataset, whereas the pseudo-codes report the mini-batch multi-epoch version of the algorithms.

We report the pseudo-code for FedAvg (see Algorithm 1), showing an implementation where the clients perform E local training epochs using mini-batch gradient descent with a batch size of B. In the code, it is assumed for simplicity that all clients participate in the averaging procedure.

Even if several variants of FedL algorithms have been developed (e.g., [2, 17]), most of the solutions available in the literature share with FedAvg both the centralized setting and the two-phases approach.

3.2 Background discrete-time weighted consensus

Consensus algorithms denote protocols distributively implemented among communicating dynamical systems (agents) that allow each agent's state estimation to evolve to a common value that takes the name of consensus value.

Representing the consensus network as a graph in which the nodes are the N clients connected by a set of edges, it is possible to define the following matrices: the adjacency matrix $A = (a_{ij}) \in \mathbf{R}^{N \times N}$, with $a_{ij} = 1$ if an edge connects clients i and j and 0 otherwise; the out-degree diagonal matrix $O = (o_{ij}) \in \mathbf{R}^{N \times N}$, with $o_{ii} = \sum_j a_{ij}$ computed as the clients' out-degrees; the Laplacian matrix $L = O - A$ and the diagonal matrix $P = \mathrm{diag}(p_i) \in \mathbf{R}^{N \times N}$, with p_i representing the weight given to the client i.

Algorithm 1. FedAvg[1]

SERVER UPDATE
for each communication round $t = 1, \cdots, T$ **do**
  select a subset of clients for the averaging procedure
  for all selected client i **do**
    CLIENT UPDATE
    receive $\tilde{w}_i$ from client i
  end for
  set $w(t) = \sum_i p_i \tilde{w}_i(t)$
  propagate w in the federation $(w_i(t) = w(t), \forall i)$
end for
CLIENT UPDATE
for each local epoch e from 1 to E **do**
  for each mini-batch b from D_i **do**
    $w_i(t-1) \leftarrow w_i(t-1) - \eta \nabla L_i(b \mid w(t-1))$
  end for
end for
set $\tilde{w}_i(t) = w_i(t-1)$
return $\tilde{w}_i(t)$ to the server

Let $x_i(t)$ be the state of agent i at time-step t, and let N_i be its set of neighbors. Under the hypothesis of a strongly connected undirected consensus graph and under the following discrete-time update rule:

$$x_i(t+1) = x_i(t) + \frac{\epsilon}{p_i} \sum_{j \in N_i} a_{ij}(x_j(t) - x_i(t)) \tag{6}$$

by assuming that the sampling time ϵ is such that $\epsilon < \min_{i \in I}(p_i/o_{ii})$[40, 41], the clients reach a consensus value in their states x_i that coincides with the weighted average of their initial conditions:

$$\bar{x} = \frac{\sum_{i \in I} p_i x_i(0)}{\sum_{i \in I} p_i}. \tag{7}$$

The convergence of the agents follows the dynamics of the discrete-time system (6) that can be equivalently written in matrix form[41] as

$$x_i(t+1) = H_p x(t) \tag{8}$$

with $H_p = I - \epsilon P^{-1} L$.

From (8), starting from the well-known definition of dominant time constant for a discrete-time linear time-invariant system and its settling time[42], it follows that the agents will reach convergence, with precision of 99%, after a number of steps n_ϵ:

$$n_\epsilon = 5 \max_{i \in I} \left\lceil \frac{-1}{\ln(\mid \lambda_i(H_p) \mid)} \right\rceil \tag{9}$$

where $\lambda_i(H_p)$ is the i-th eigenvalue different from 1 of the matrix H_p and $\lceil \cdot \rceil$ denotes the ceiling function of its argument, with a resulting 1%-settling time $t_a \approx n_\epsilon \times \epsilon$.

4 Decentralized federated learning

This section presents the two algorithms developed for decentralized federated learning: A decentralized version of FedAvg in Section 4.1 and an algorithm based on discrete-time weighted average consensus in Section 4.2.

4.1 Decentralized federated averaging

Algorithm 2. DecFedAvg
1) **DECENTRALIZED FEDERATED TRAINING**
2) **for** all communication rounds $t = 1, \cdots, T$ **do**
3) **for** all clients $i \in I$ **do**
4) **for** each local epoch e from 1 to E **do**
5) **for** each mini-batch b from D_i **do**
6) $w_i(t-1) \leftarrow w_i(t-1) - \eta \nabla L_i(b \mid w(t-1))$
7) **end for**
8) **end for**
9) set $\tilde{w}_i(t) = w_i(t-1)$
10) update $w_i(t)$ according to (10)
11) **end for**
12) **end for**

As discussed, FedAvg was tailored to minimize communication costs (e.g., by limiting the number of clients exchanging their models at each communication round). It was designed for IoT/edge computing settings, as most of its evolutions (see, e.g., [2, 4, 5, 18]).

In scenarios where communication costs are negligible (e.g., in healthcare facilities cooperation), the main communication constraints are related to the presence of point-to-point communications among the clients and to the unavailability of a centralized server. A natural choice is then to develop a decentralized version of FedAvg, i.e., of (3)–(5). A direct decentralization of FedAvg would consist of using (3) and the following equation:

$$w_i(t) = \frac{1}{\mid \mathbb{D}_i \mid} \Big(\mid D_i \mid \tilde{w}_i(t) + \sum_{j \in N_i} \mid D_j \mid \tilde{w}_j(t) \Big), \ \forall \ i \in I \tag{10}$$

with $\mid \mathbb{D}_i \mid = \mid D_i \mid + \sum_{j \in N_i} \mid D_j \mid$. If the graph is complete (i.e., all clients are neighbours of each other), substituting (4) into (5) yields that (3) and (10) are equivalent to (3)–(5).

Equation (10) states that, at every communication round t, the clients exchange the weights $\tilde{w}_i(t)$ of their locally trained DeepNNs with their neighbours. Each client then updates the vector $w_i(t)$ by computing a weighted average of the collected vectors $\tilde{w}_j(t)$, with $j \in N_i \cup \{i\}$ (including its own vector), with weights set as the cardinality $\mid D_j \mid$ of the clients' data. In general, arbitrary weights can be attributed to the clients, e.g., depending on the in/out-degree of the nodes in the federation graph or reflecting the trust level that client i has in client j.

The pseudo-code for the decentralized FedAvg algorithm (DecFedAvg) is reported (see Algorithm 2) to improve the clarity of the presentation.

Remark 1. The proposed decFedAvg algorithm does not utilize any information on the communication network topology. Also, we note that the proposed distribution of (3)–(5), for its simplicity, can in principle be applied to any FedL algorithm with the same structure as FedAvg. However, the convergence results of such algorithms typically rely on the propagation of a common averaged DeepNN into the federation. Hence, the convergence of the decentralized versions is not guaranteed, as, in general, at each communication round t, the various $w_i(t)$ are different. In the FedAvg case, (3) and (10) yield that, at each t, the various $w_i(t)$, and consequently the performance of the clients, may differ significantly.

4.2 Consensus-based decentralized federated learning

Algorithm 3. FedLCon
1) **DECENTRALIZED FEDERATED TRAINING**
2) **for** all communication rounds $t = 1, \cdots, T$ **do**
3) **for** all clients $i \in I$ **do**
4) **for** each local epoch e from 1 to E **do**
5) **for** each mini-batch b from D_i **do**
6) $w_i(t-1) \leftarrow w_i(t-1) - \eta \nabla L_i(b \mid w(t-1))$
7) **end for**
8) **end for**

9) set $\tilde{w}_i(t) = w_i(t-1)$
10) **end for**
11) update $w_i(t)$ via a CONSENSUS ROUND
12) **end for**
13) **CONSENSUS ROUND**
14) Compute n_ϵ according to (9) depending on the topology
15) Set $x_i(0) = \tilde{w}_i(t)$ for all clients $i \in I$
16) **for** $k = 0, \cdots, n_\epsilon - 1$ **do**
17) **for** all clients $i \in I$ **do**
18) update x_i according to (11)
19) **end for**
20) **end for**
21) set $w_i(t) = x_i(n_\epsilon)$ for all clients $i \in I$

To overcome the main limitation of the DecFedAvg algorithm, i.e., the lack of convergence guarantees, we propose a novel consensus-based algorithm. In fact, even if the clients considered in the FedL setting are not dynamical systems, there are similarities between the framework for FedL and the one for discrete-time weighted average consensus. By interpreting the weights $w_i(t)$ of the federated clients as the states $x_i(t)$ of a set of agents seeking consensus (see Section 3.2), we propose to combine (3)–(5) and (6) as described below.

At each communication round t, the $\tilde{w}_i$s are computed by the same (3) used in standard FedL solutions. Differently from the DecFedAvg algorithm, the update of the weight vectors $w_i(t)$ is not performed by (10) but involves a consensus round. Let k be the consensus round index and recall that n_ϵ is the number of iterations required to reach consensus within the round. Then, to reach consensus the federated clients exchange information n_ϵ times, starting from the initial values $x_i(0) = \tilde{w}_i(t)$, for all $i \in I$. The following iteration rule is executed for $k = 0, \cdots, n_\epsilon - 1$:

$$x_i(k+1) = x(k) + \frac{\epsilon}{|\, D_i \,|} \sum_{j \in N_i} a_{ij} \big(x_j(k) - x_i(k) \big) \tag{11}$$

with ϵ chosen as in Section 3.2.

Due to the structure of the update rule (11), which is the same as (6), for the consensus-based convergence properties presented in Section 3.2 and discussed in [40, 41], one observes that the states of the federation clients converge towards the value

$$x_i(n_\epsilon) \approx \frac{\sum_i |\, D_i \,|\, \tilde{w}_i(t)}{|\, \mathbb{D} \,|}, \ \ \forall\ i \in I \tag{12}$$

i.e., at the end of the communications (when consensus is reached among the federated clients), the proxy variables x_i approximate the weights $w(t)$ computed by the centralized FedL case with (4). Setting

$$w_i(t) = x_i(n_\epsilon) \tag{13}$$

the procedure can be repeated starting from the training of (3) for all the communication rounds t. Note that each communication round t now yields n_ϵ information exchanges since it involves a consensus round.

The resulting consensus-based distributed federated learning algorithm (FedLCon) is reported as a pseudocode (see Algorithm 3) in the same form as the two previous cases.

Remark 2. Contrary to DecFedAvg, FedLCon approximates the exact decentralization of the original FedAvg algorithm, as at the end of each communication round (i.e., after the n_ϵ steps of the consensus round), the weights of all the clients converge to the same values. This consideration has two consequences:

1) The proposed consensus-based solution can be directly applied to any FedAvg-like algorithm available in the literature, as it is transparent from the model-averaging point of view and its implementation details (e.g., privacy-preserving features and complex weighting criteria for the various clients);

2) As the consensus round is transparent to the FedAvg algorithm, different consensus algorithms can be used to exploit the communication and/or topology properties of the application scenarios (e.g., multi-hop discrete-time consensus).

Remark 3. The introduction of the consensus round (and its n_ϵ information exchanges) constitutes a communication overhead and the main limitation of the proposed algorithm. In fact, from (9), one may note how n_ϵ is influenced by the eigenvalues of the matrix H_p, which in turn depends on the communication network Laplacian matrix L. We mention that, in general, such eigenvalues do not depend directly on the number of clients in the federation and instead capture the topology connectivity level, meaning that the scalability of FedLCon is mostly affected by the number of links available in the communication topology and their positioning. In order to reduce the impact of the communication overhead, one may apply to the information exchanges over the consensus network some of the latest solutions for communication efficiency that have been designed for the standard FedL setting (e.g., gradient/model compression approaches in [27, 28, 43, 44]; the approach in [45], where clients evaluate the contribution of their training before joining the data exchange; or transfer learning solutions, as they envisage the exchange of only a small portion of the overall model). Nevertheless, in scenarios in which the time for data transmission is limited and/or non-negligible with respect to the training time, one may consider employing DecFedAvg at the cost of a performance degradation.

Remark 4. From the point of view of the federation client, both DecFedAvg and FedLCon involve comparable computations, that are negligible when compared to the training process. In fact, the amount of operations (unrelated to the training) conducted in a communica-

tion round by a client in a DecFedAvg-based federation increases linearly with its number of neighbours, whereas FedLCon repeats the same amount of computations n_ϵ times due to the consensus round.

5 Simulations

This section reports on the test simulations performed to validate the proposed approaches and to assess their differences in two different settings. We utilized the MNIST (modified National Institute of Standards and Technology database) dataset[46] (one of the most used benchmark solutions in the ML literature, also used in the original FedL paper[1]), distributed among the clients in two different ways, as will be described in the following subsections. The MNIST dataset consists of a set of 60 K+10 K labeled images of handwritten digits (from 0 to 9), where the last 10 K images are used as a test dataset to evaluate the performance of DeepNNs trained over the first 60 K training data samples.

We tested our algorithms on four different topologies with $N = 6$ clients: A complete graph, two sparse topologies – a circle and a star one – and a fairly connected random topology with 9 links. Additionally, we included in all our tests a centralized benchmark implementing the standard FedAvg algorithm that involves the presence of a coordinating server. For our consensus-based algorithm, we set for each federation $\epsilon = 0.99 \min_{i \in I}(|\ D_i\ |/o_{ii})$, leading to $n_\epsilon = 5$ for the complete and circle topologies, $n_\epsilon = 10$ for the random one and $n_\epsilon = 25$ for the star topology (for the evaluation of (9), we mention that the Laplacian matrix L may be trivially obtained from the graphs in Fig. 2). Note that o_{ii} and, consequently, ϵ vary depending on the topology. The topologies were selected so that the proposed algorithms will have to deal with information flows of different natures and characteristics to better assess their limits and adaptability properties. The amount of communication overhead that FedLCon has compared to DecFedAvg depends on the different values of n_ϵ.

In this work, we used for all clients a convolutional neural network constituted by two convolutional layers with 32 and 64 filters of size 3×3, respectively, followed by a 2×2 maxpooling layer with a dropout of 0.25 and a dense layer of 128 neurons with a dropout of 0.5. All the activation functions were set as rectified linear units (ReLUs), save for the output layer that was constituted by a dense layer of 10 neurons with a softmax activation function, as common in multi-label classification tasks. The resulting DeepNN has about 1.2 M trainable parameters (weights). The optimizer employed for the training was Adam, with an initial learning rate of 0.01. All the federation clients trained locally for 2 epochs ($E = 2$ in the algorithm pseudo-codes), i.e., at each training step the clients train twice over their entire dataset, with a mini-batch size of 32. All the other parameters and the initialization are the standard ones from Keras 2.4.0.

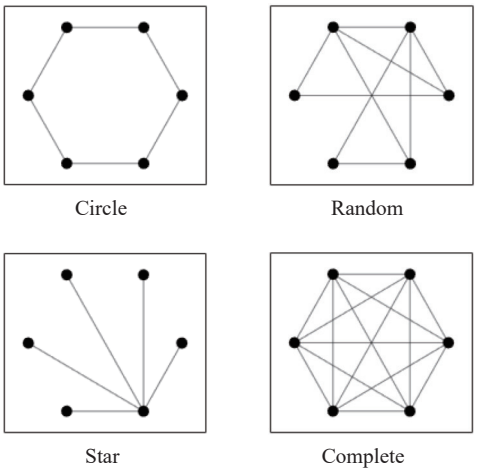

Fig. 2 Considered federation topologies

For the sake of presentation, in Figs.3–8 of this section, for the decentralized algorithms, only the accuracy curves for the first client of the federation (i.e., $i = 1$) averaged over 10 runs, as the behaviour shown by the others is equivalent. Table 1 reports the main characteristics of the three simulations performed, briefly reporting their main results.

5.1 Test 1.A (MNIST): One missing class per client

In this test, the MNIST dataset was split into 6 parts and distributed to the clients so that client i, provided with about 10 K data samples, had no samples of the class i (i.e., the $(i-1)$-th digit). The purpose of this test is to demonstrate how the algorithms perform on a fairly even data distribution compared to the standard, centralized, FedAvg solution involving a server.

Fig. 3 reports the accuracy evolution versus the communication rounds for the DecFedAvg algorithm of Section 4.1. The picture shows how the lines for the centralized (FedAvg) and the complete topologies perfectly overlap, as expected from (3) and (10). Minor performance drops (less than 1%) are observed for the sparser topologies, i.e., the circle and star ones.

Fig. 4 reports the behaviour of a federation governed by the FedLCon algorithm. In this case, the performances achieved with all the topologies – including the sparser ones – converge towards the performance reached by the centralized solution within errors of less than 0.2%. The drawback of FedLCon is that it requires a significantly higher communication overhead (n_ϵ information exchanges per communication round versus one exchange for DecFedAvg).

In this first simulation, both algorithms performed well due to the balanced data distribution that makes the local updates of DecFedAvg comparable, from a knowledge discovery point of view, to the global ones recon-

Table 1 Summary of the simulations

Test	Objective	Data set	Nodes	Results summary
1.A	Validate the proposed algorithms against FedAvg	MNIST, balanced	6	DecFedAvg and FedLCon perform well and similarly to the centralised case (FedAvg).
1.B	Test the scalability of the algorithms on a bigger network	CIFAR-10, IID	20	FedLCon manages to reconstruct the performance of FedAvg, while we observe that the performance of DecFedAvg degrades slightly.
2	Test the algorithms on non-IID data distributions	MNIST, non-IID	6	DecFedAvg fails to converge on sparse topologies, FedLCon is unaffected and successfully obtains the same performances of the centralized case.

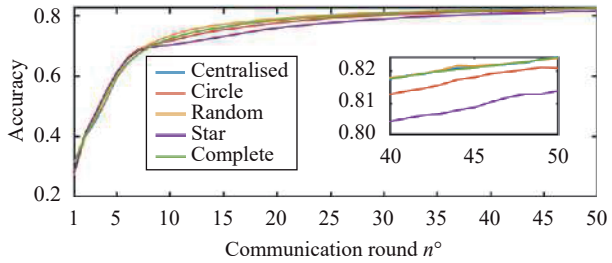

Fig. 3 Test 1.A: Accuracy evolution over communication rounds for the DecFedAvg algorithm

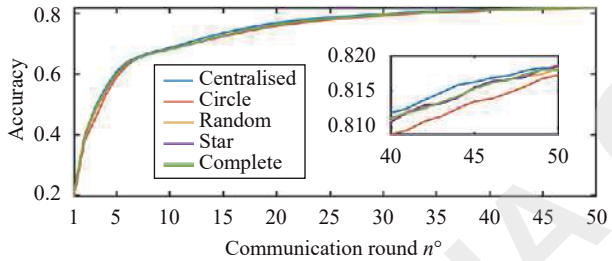

Fig. 4 Test 1.A: Accuracy evolution over communication rounds for the FedLCon algorithm

structed by FedLCon. The fact that the topology did not affect the performance of any algorithm significantly suggests that the data distribution allows all agents to solve a good portion of the task independently of their neighbours′ contribution. In the following test, we aim to discover whether the two algorithms continue to perform similarly on harder tasks.

5.2 Test 1.B (CIFAR-10): Transfer learning with IID-data

This test was designed to assert the scalability of the algorithms and to highlight their seamless integration capabilities with more complex tasks such as transfer learning. In this test, we analyze the well known CIFAR-10 (Canadian Institute For Advanced Research – 10)[47] by applying transfer learning to the VGG19 (Visual Geometry Group – 19)[48] neural network trained for ImageNet[49]. As customary with transfer learning tasks, our DeepNN was constituted by the pre-trained VGG19 combined with a stack of fully connected layers of 1 024, 512 and 256 neurons. For this simulation, we considered a federation of 20 clients, depicted in Fig. 5 ($n_\epsilon = 20$). The entire dataset was distributed uniformly to the clients.

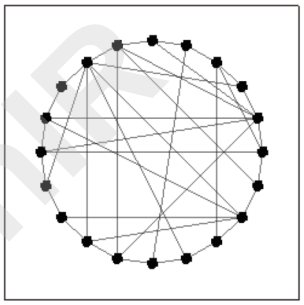

Fig. 5 Random federation topology of 20 clients

As in the previous case, thanks to the balanced data distribution, from Fig. 6 we can note how both algorithms perform similarly, with a slight advantage of FedLCon that is able to reconstruct from the very start the performance of FedAvg whereas DecFedAvg slightly lags behind. This slower increase rate for the accuracy of DecFedAvg is most likely due to the fact that a client obtains information (knowledge) only from its neighbours: This may be too limiting in complex tasks, as more communication rounds are required to gather knowledge from the entire federation. FedLCon, on the other hand, at the end of each consensus round, ensures that all clients are provided with the same information, in the form of having the same DeepNN approximated by the entire federation.

In the following test, we will demonstrate how DecFedAvg is also substantially more sensitive to unbalanced data distributions among clients, whereas FedLCon, thanks to its consensus-based update protocol, shows a significantly more robust behaviour.

5.3 Test 2 (MNIST): Four random classes per client

In this test, the MNIST dataset is distributed so that each client has access only to four digits, with each client having between 7.5 K and 8.5 K data samples. In particular, client 1 had digits $1, 2, 3, 4$, client 2 had digits

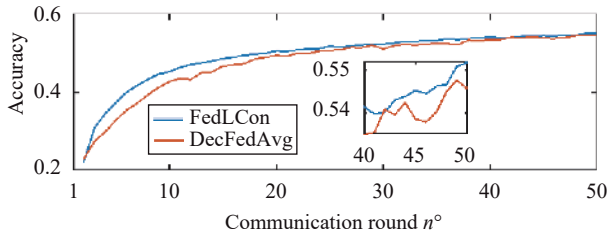

Fig. 6 Test 1.B: Accuracy evolution over the communication rounds for the FedLCon and DecFedAvg algorithms

$0, 2, 8, 9$, client 3 had digits $3, 4, 5, 6$, client 4 had digits $0, 7, 8, 9$, client 5 had digits $1, 2, 7, 9$ and client 6 had digits $1, 3, 4, 6$. The goal is to evaluate the algorithm performance with data distributions that are significantly different among the clients.

With the DecFedAvg algorithm, Fig. 7 highlights, as expected, that also, in this test, the plots of the centralized and the complete topologies overlap. However, Fig. 7 shows that the algorithm struggles with the unfavourable data distribution, as it achieves significantly lower performances with the sparser star and circle topologies. It is worth noting that, from the very first communication round, the various topologies have different accuracy levels: This is due to the local nature of the algorithm – each client exchanges information only with its neighbours – with no convergence guarantee, meaning that each client has a different DeepNN.

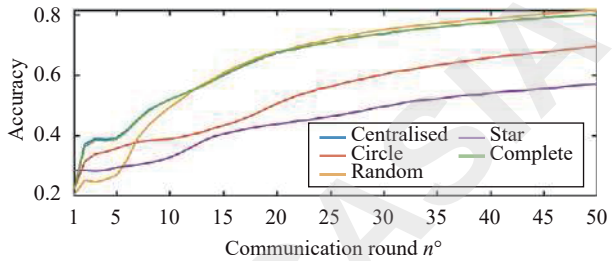

Fig. 7 Test 2: Accuracy evolution over communication rounds for the DecFedAvg algorithm.

We also observe that in this case the accuracy obtained with the random topology is surprisingly close to the one obtained with the centralized one. In general, the DeepNNs of a federation controlled by (3) and (10) do not converge to the DeepNN of the centralized case. However, depending on the data distribution, it may happen (as depicted in Fig. 7) that sparser connectivity graphs lead to a well-performing DeepNN.

Fig. 8 reports the accuracy curves for a federation controlled by FedLCon. The accuracy evolution is practically equivalent with all topologies, with a performance drop of less than 2% compared to the centralized case. This result is in line with the nature of the consensus-based communication protocol, as at the beginning of each training phase, all the various clients have at their disposal the same common values for their w_is (save for slight deviations due to the usage of the 1% settling time in (9)), independently from the communication topology.

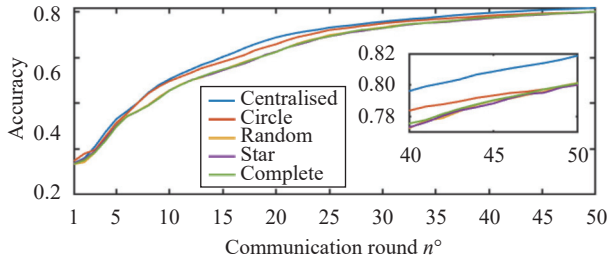

Fig. 8 Test 2: Accuracy evolution over communication rounds for the FedLCon algorithm.

We observe that, in this scenario, the communication overhead of FedLCon allowed the federation to reach satisfactory performances on arbitrary topologies, whereas the simpler DecFedAvg algorithm failed the task on sparser communication graphs where the local nature of the updates prevents the information from being exchanged correctly over the federation, causing the clients not to generalize correctly.

6 Conclusions and future works

This paper presented two decentralized federated learning (FedL) algorithms, the former (DecFedAvg), obtained as a direct decentralization of the original FedL algorithm, FedAvg, and the latter (FedLCon), obtained on the ground of results from discrete-time average consensus theory. As shown by simulations, the proposed consensus-based algorithm is an enabler to deploy FedL solutions on arbitrary topologies at the expense of higher communication overhead. On the contrary, DecFedAvg proved to be a reasonable approximation of FedLCon in scenarios in which data is distributed more uniformly over the clients and the communication topology is not sparse.

Decentralized FedL solutions are of crucial interest for small-scale federations of collaborating companies, such as healthcare facilities, that may be prevented from joining a centralized federation, e.g., due to privacy regulations. The proposed consensus-based solution is then relevant also considering that, in such scenarios, the communication overhead is not a key performance indicator as, e.g., in IoT scenarios. In this direction, FedLCon represents a solution that is able to be seamlessly applied to almost all FedAvg-like FedL algorithms.

On-going and future work is aimed at using the proposed consensus-based approach to tailor solutions to specific use cases, in the first place considering healthcare federations, in line with the objectives of the FedMedAI project.

Acknowledgements

This work was Supported by the Lazio region, in the

scope of the project FedMedAI, Regional Operative Programme (POR) of the European fund for regional development (FESR) Lazio 2014 – 2020 (Azione 1.2.1) (No. A0375-2020-36491-23/10/2020).

References

[1] H. B. McMahan, E. Moore, D. Ramage, S. Hampson. Communication-efficient learning of deep networks from decentralized data. In *Proceedings of the 20th International Conference on Artificial Intelligence and Statistics*, Fort Lauderdale, USA, pp. 1273–1282, 2016.

[2] T. Li, A. K. Sahu, M. Zaheer, M. Sanjabi, A. Talwalkar, V. Smith. Federated optimization in heterogeneous networks. In *Proceedings of the 3rd Conference on Machine learning and System*, Austin, USA, pp. 429–450, 2018.

[3] H. B. McMahan, E. Moore, D. Ramage, B. A. Y. Arcas. Federated learning of deep networks using model averaging. [Online], Available: https://arxiv.org/abs/1602.05629, 2016.

[4] Y. F. Ye, S. Li, F. Liu, Y. H. Tang, W. T. Hu. EdgeFed: Optimized federated learning based on edge computing. *IEEE Access*, vol. 8, pp. 209191–209198, 2020. DOI: 10.1109/access.2020.3038287.

[5] L. U. Khan, M. Alsenwi, I. Yaqoob, M. Imran, Z. Han, C. S. Hong. Resource optimized federated learning-enabled cognitive internet of things for smart industries. *IEEE Access*, vol. 8, pp. 168854–168864, 2020. DOI: 10.1109/access.2020.3023940.

[6] A. Hard, K. Rao, R. Mathews, S. Ramaswamy, F. Beaufays, S. Augenstein, H. Eichner, C. Kiddon, D. Ramage. Federated learning for mobile keyboard prediction. [Online], Available: https://arxiv.org/abs/1811.03604, 2018.

[7] T. Yang, G. Andrew, H. Eichner, H. C. Sun, W. Li, N. Kong, D. Ramage, F. Beaufays. Applied federated learning: Improving Google keyboard query suggestions. [Online], Available: https://arxiv.org/abs/1812.02903, 2018.

[8] S. Ramaswamy, R. Mathews, K. Rao, F. Beaufays. Federated learning for Emoji prediction in a mobile keyboard. [Online], Available: https://arxiv.org/abs/1906.04329, 2019.

[9] J. H. Luo, X. Y. Wu, Y. Luo, A. B. Huang, Y. F. Huang, Y. Liu, Q. Yang. Real-world image datasets for federated learning. [Online], Available: https:/arxiv.org/abs/1910.11089, 2019.

[10] L. Ahmed, K. Ahmad, N. Said, B. Qolomany, J. Qadir, A. Al-Fuqaha. Active learning based federated learning for waste and natural disaster image classification. *IEEE Access*, vol. 8, pp. 208518–208531, 2020. DOI: 10.1109/access.2020.3038676.

[11] T. S. Brisimi, R. D. Chen, T. Mela, A. Olshevsky, I. C. Paschalidis, W. Shi. Federated learning of predictive models from federated electronic health records. *International Journal of Medical Informatics*, vol. 112, pp. 59–67, 2018. DOI: 10.1016/j.ijmedinf.2018.01.007.

[12] M. J. Sheller, G. A. Reina, B. Edwards, J. Martin, S. Bakas. Multi-institutional deep learning modeling without sharing patient data: A feasibility study on brain tumor segmentation. In *Proceedings of the 4th International Workshop on Brainlesion: Glioma, Multiple Sclerosis, Stroke and Traumatic Brain Injuries*, Springer, Granada, Spain, pp. 92–104, 2019. DOI: 10.1007/978-3-030-11723-8_9.

[13] M. Aledhari, R. Razzak, R. M. Parizi, F. Saeed. Federated learning: A survey on enabling technologies, protocols, and applications. *IEEE Access*, vol. 8, pp. 140699–140725, 2020. DOI: 10.1109/access.2020.3013541.

[14] H. L. Yang, J. Zhao, Z. H. Xiong, K. Y. Lam, S. M. Sun, L. Xiao. Privacy-preserving federated learning for UAV-enabled networks: Learning-based joint scheduling and resource management. *IEEE Journal on Selected Areas in Communications*, vol. 39, no. 10, pp. 3144–3159, 2021. DOI: 10.1109/JSAC.2021.3088655.

[15] M. Hao, H. W. Li, X. Z. Luo, G. W. Xu, H. M. Yang, S. Liu. Efficient and privacy-enhanced federated learning for industrial artificial intelligence. *IEEE Transactions on Industrial Informatics*, vol. 16, no. 10, pp. 6532–6542, 2020. DOI: 10.1109/TII.2019.2945367.

[16] K. Wei, J. Li, C. Ma, M. Ding, S. Wei, F. Wu, G. H. Chen, T. Ranbaduge. Vertical federated learning: Challenges, methodologies and experiments. [Online], Available: https://arxiv.org/abs/2202.04309, 2022.

[17] Q. B. Li, Z. Y. Wen, Z. M. Wu, S. X. Hu, N. B. Wang, Y. Li, X. Liu, B. S. He. A survey on federated learning systems: Vision, hype and reality for data privacy and protection. *IEEE Transactions on Knowledge and Data Engineering*, to be published. DOI: 10.1109/TKDE.2021.3124599.

[18] W. Y. B. Lim, N. C. Luong, D. T. Hoang, Y. T. Jiao, Y. C. Liang, Q. Yang, D. Niyato, C. Y. Miao. Federated learning in mobile edge networks: A comprehensive survey. *IEEE Communications Surveys & Tutorials*, vol. 22, no. 3, pp. 2031–2063, 2020. DOI: 10.1109/COMST.2020.2986024.

[19] K. Bonawitz, V. Ivanov, B. Kreuter, A. Marcedone, H. B. McMahan, S. Patel, D. Ramage, A. Segal, K. Seth. Practical secure aggregation for privacy-preserving machine learning. In *Proceedings of ACM SIGSAC Conference on Computer and Communications Security*, ACM, Dallas, USA, pp. 1175–1191, 2017. DOI: 10.1145/3133956.3133982.

[20] R. C. Geyer, T. Klein, M. Nabi. Differentially private federated learning: A client level perspective. [Online], Available: https://arxiv.org/abs/1712.07557, 2017.

[21] S. Truex, N. Baracaldo, A. Anwar, T. Steinke, H. Ludwig, R. Zhang, Y. Zhou. A hybrid approach to privacy-preserving federated learning. In *Proceedings of the 12th ACM Workshop on Artificial Intelligence and Security*, ACM, London, USA, pp. 1–11, 2019. DOI: 10.1145/3338501.3357370.

[22] Q. S. Zhang, B. Gu, C. Deng, H. Huang. Secure bilevel asynchronous vertical federated learning with backward updating. In *Proceedings of AAAI Conference on Artificial Intelligence*, vol. 35, no. 12, pp. 10896–10904, 2021.

[23] Z. B. Wang, M. K. Song, Z. F. Zhang, Y. Song, Q. Wang, H. R. Qi. Beyond inferring class representatives: User-level privacy leakage from federated learning. In *Proceedings of IEEE Conference on Computer Communications*, IEEE,

Paris, France, pp. 2512–2520, 2019. DOI: 10.1109/infocom.2019.8737416.

[24] S. Kim. Incentive design and differential privacy based federated learning: A mechanism design perspective. *IEEE Access*, vol. 8, pp. 187317–187325, 2020. DOI: 10.1109/access.2020.3030888.

[25] B. Gu, A. Xu, Z. Y. Huo, C. Deng, H. Huang. Privacy-preserving asynchronous federated learning algorithms for multi-party vertically collaborative learning. [Online], Available: https://arxiv.org/abs/2008.06233, 2020.

[26] J. Konečný, H. B. McMahan, F. X. Yu, P. Richtárik, A. T. Suresh, D. Bacon. Federated learning: Strategies for improving communication efficiency. [Online], Available: https:/arxiv.org/abs/1610.05492, 2016.

[27] F. Sattler, S. Wiedemann, K. R. Müller, W. Samek. Robust and communication-efficient federated learning from Non-i.i.d. data. *IEEE Transactions on Neural Networks and Learning Systems*, vol. 31, no. 9, pp. 3400–3413, 2020. DOI: 10.1109/tnnls.2019.2944481.

[28] Q. S. Zhang, B. Gu, C. Deng, S. X. Gu, L. F. Bo, J. Pei, H. Huang. AsySQN: Faster vertical federated learning algorithms with better computation resource utilization. In *Proceedings of the 27th ACM SIGKDD Conference on Knowledge Discovery & Data Mining*, ACM, Washington DC, USA, pp. 3917–3927, 2021. DOI: 10.1145/3447548.3467169.

[29] Q. S. Zhang, B. Gu, Z. Y. Dang, C. Deng, H. Huang. Desirable companion for vertical federated learning: New Zeroth-order gradient based algorithm. In *Proceedings of the 30th ACM International Conference on Information & Knowledge Management*, ACM, Atlanta, USA, pp. 2598–2607, 2021. DOI: 10.1145/3459637.3482249.

[30] C. H. Hu, J. Y. Jiang, Z. Wang. Decentralized federated learning: A segmented gossip approach. [Online], Available: https://arxiv.org/abs/1908.07782, 2019.

[31] J. Y. Jiang, L. Hu. Decentralised federated learning with adaptive partial gradient aggregation. *CAAI Transactions on Intelligence Technology*, vol. 5, no. 3, pp. 230–236, 2020. DOI: 10.1049/trit.2020.0082.

[32] X. H. Chen, J. L. Ji, C. Q. Luo, W. X. Liao, P. Li. When machine learning meets blockchain: A decentralized, privacy-preserving and secure design. In *Proceedings of IEEE International Conference on Big Data*, IEEE, Seattle, USA, pp. 1178–1187, 2018. DOI: 10.1109/BigData.2018.8622598.

[33] Y. Z. Li, C. Chen, N. Liu, H. W. Huang, Z. B. Zheng, Q. Yan. A blockchain-based decentralized federated learning framework with committee consensus. *IEEE Network*, vol. 35, no. 1, pp. 234–241, 2021. DOI: 10.1109/MNET.011.2000263.

[34] A. Nedic. Distributed gradient methods for convex machine learning problems in networks: Distributed optimization. *IEEE Signal Processing Magazine*, vol. 37, no. 3, pp. 92–101, 2020. DOI: 10.1109/MSP.2020.2975210.

[35] S. Manfredi, D. Angeli. Robust distributed estimation of the maximum of a field. *IEEE Transactions on Control of Network Systems*, vol. 7, no. 1, pp. 372–383, 2020. DOI: 10.1109/TCNS.2019.2906865.

[36] B. D. O. Anderson, M. B. Ye. Recent advances in the modelling and analysis of opinion dynamics on influence networks. *International Journal of Automation and Computing*, vol. 16, no. 2, pp. 129–149, 2019. DOI: 10.1007/s11633-019-1169-8.

[37] W. Ren. Consensus based formation control strategies for multi-vehicle systems. In *Proceedings of American Control Conference*, IEEE, Minneapolis, USA, pp. 4237–4242, 2006. DOI: 10.1109/ACC.2006.1657384.

[38] Z. A. Zhang, M. Y. Chow. Convergence analysis of the incremental cost consensus algorithm under different communication network topologies in a smart grid. *IEEE Transactions on Power Systems*, vol. 27, no. 4, pp. 1761–1768, 2012. DOI: 10.1109/TPWRS.2012.2188912.

[39] C. T. Dinh, N. H. Tran, M. N. H. Nguyen, C. S. Hong, W. Bao, A. Y. Zomaya, V. Gramoli. Federated learning over wireless networks: Convergence analysis and resource allocation. *IEEE/ACM Transactions on Networking*, vol. 29, no. 1, pp. 398–409, 2021. DOI: 10.1109/tnet.2020.3035770.

[40] R. Olfati-Saber, R. M. Murray. Consensus problems in networks of agents with switching topology and time-delays. *IEEE Transactions on Automatic Control*, vol. 49, no. 9, pp. 1520–1533, 2004. DOI: 10.1109/TAC.2004.834113.

[41] F. Pedroche, M. Rebollo, C. Carrascosa, A. Palomares. Convergence of weighted-average consensus for undirected graphs. *International Journal of Complex Systems in Science*, vol. 4, no. 1, pp. 13–16, 2014.

[42] K. Ogata. *Discrete-time Control Systems*, 2nd ed., New York, USA: Prentice-Hall, Inc., 1995.

[43] F. Haddadpour, M. M. Kamani, A. Mokhtari, M. Mahdavi. Federated learning with compression: Unified analysis and sharp guarantees. In *Proceedings of the 24th International Conference on Artificial Intelligence and Statistics*, Beijing, China, pp. 2350–2358, 2021.

[44] A. Albasyoni, M. Safaryan, L. Condat, P. Richtárik. Optimal gradient compression for distributed and federated learning. [Online], Available: https://arxiv.org/abs/2010.03246, 2020.

[45] L. P. Wang, W. Wang, B. Li. CMFL: Mitigating communication overhead for federated learning. In *Proceedings of the 39th IEEE International Conference on Distributed Computing Systems*, IEEE, Dallas, USA, pp. 954–964, 2019. DOI: 10.1109/icdcs.2019.00099.

[46] Y. LeCun, C. Cortes, C. J. C. Burges. MNIST handwritten digit database. ATT Labs. [Online], Available: http://yann.lecun.com/exdb/mnist, 2010.

[47] A. Krizhevsky. Learning Multiple Layers of Features from Tiny Images, Technical Report TR-2009, University of Toronto, Toronto, Canada, 2009.

[48] M. Shaha, M. Pawar. Transfer learning for image classification. In *Proceedings of the 2nd International Conference on Electronics, Communication and Aerospace Technology*, IEEE, Coimbatore, India, pp. 656–660, 2018. DOI: 10.1109/ICECA.2018.8474802.

[49] M. Huh, P. Agrawal, A. A. Efros. What makes ImageNet good for transfer learning? [Online], Available: https://arxiv.org/abs/1608.08614, 2016.

Alessandro Giuseppi received the M. Sc. degree in control engineering and the Ph. D. degree in automatica from University of Rome La Sapienza, Italy in 2016 and 2019, respectively. Since 2016, he has participated in 6 other EU and National research projects, mainly in the fields of control systems and artificial intelligence. He is an assistant professor in automatica at Department of Computer, Control, and Management Engineering Antonio Ruberti (DIAG), University of Rome La Sapienza, Italy. Currently, he is the scientific coordinator of the ESA-funded research project ARIES, related to wildfire emergency management and work package leader in the EU-Korea H2020 project 5G-ALLSTAR. Since 2020, he is serving as associate editor for the *International Journal of Control, Automation and Systems* (Springer). His main research activities are in the fields of network control and intelligent systems, where he published about 50 papers in international journals and conferences.

His research interests include intelligent systems and automatic control, with emphasis on smart networks.

E-mail: giuseppi@diag.uniroma1.it (Corresponding author)
ORCID iD: 0000-0001-5503-8506

Sabato Manfredi received the M. Sc. degree in electronics engineering and the Ph. D. degree in computer science and automatica from University of Naples Federico II, Italy in 2001 and 2004, respectively. He is currently an associate professor of automatic control with Department of Electrical Engineering and Information Technology, University of Naples Federico II, Italy. He has been a visiting academic with the Control and Power Group, Electrical and Electronic Engineering Department, Imperial College London, UK since 2012. He has been a visiting professor with School of Mathematical Sciences, Queen Mary, UK, during 2017–2018. He has authored/coauthored more than 90 scientific publications including 18 single-author papers and the monograph: Multilayer Control of Networked Cyber-Physical Systems. Application to Monitoring, Autonomous and Robot Systems (Advances in Industrial Control Series, Springer, 2017). He collaborates with many international universities and companies, holds European patent, is a proponent member of an academic spin-off, and is involved in a range of academic, industrial, and consulting projects.

His research interests include automatic control with a special emphasis on nonlinear and complex networks, distributed control and optimization, sensor/drone networks, and new technologies/algorithms for smart city and cyber–physical systems.

E-mail: sabato.manfredi@unina.it

Antonio Pietrabissa received the M. Sc. degree in electronics engineering and the Ph. D. degree in systems engineering from University of Rome "La Sapienza", Italy in 2000 and 2004, respectively, and where he teaches automatic control and process automation. Since 2000, he has participated in about 25 EU and National research projects. He is associate professor at Department of Computer, Control, and Management Engineering "Antonio Ruberti" (DIAG), University of Rome "La Sapienza", Italy. Currently, he is the coordinator of the project ARIES on fire emergency prevention, funded by ESA, and the scientific responsible of the research projects 5G-ALLSTARS on 5G communications, funded within the H2020 Europe-South Korea cooperation, and FedMedAI on medical applications of federated learning. He serves as associate editor for *Control Engineering Practice* (Elsevier) and for *IEEE Transactions on Automation Science and Engineering*. He is author of more than 50 journal papers and 80 conference papers.

His research interest is the application of systems and control theory to the analysis and control of networks.

E-mail: pietrabissa@diag.uniroma1.it