sensors

MDPI

Article

DACFL: Dynamic Average Consensus-Based Federated Learning in Decentralized Sensors Network

Zhikun Chen [1,†], Daofeng Li [1,†], Jinkang Zhu [1,2] and Sihai Zhang [1,3,4,*]

[1] Department of Electronic Engineering and Information Science, School of Information Science and Technology, University of Science and Technology of China, No. 96 Jinzhai Road, Hefei 230026, China; zhikunch@mail.ustc.edu.cn (Z.C.); df007@mail.ustc.edu.cn (D.L.); jkzhu@ustc.edu.cn (J.Z.)
[2] PCNSS Laboratory, School of Information Science and Technology, University of Science and Technology of China, Hefei 230026, China
[3] CAS Key Laboratory of Wireless-Optical Communications, School of Information Science and Technology, University of Science and Technology of China, Hefei 230026, China
[4] School of Microelectronics, University of Science and Technology of China, Hefei 230026, China
* Correspondence: shzhang@ustc.edu.cn
† These authors contributed equally to this work.

Abstract: Federated Learning (FL) is a privacy-preserving way to utilize the sensitive data generated by smart sensors of user devices, where a central parameter server (PS) coordinates multiple user devices to train a global model. However, relying on centralized topology poses challenges when applying FL in a sensors network, including imbalanced communication congestion and possible single point of failure, especially on the PS. To alleviate these problems, we devise a Dynamic Average Consensus-based Federated Learning (DACFL) for implementing FL in a decentralized sensors network. Different from existing studies that replace the model aggregation roughly with neighbors' average, we first transform the FL model aggregation, which is the most intractable in a decentralized topology, into the dynamic average consensus problem by treating a local training procedure as a discrete-time series.We then employ the first-order dynamic average consensus (FODAC) to estimate the average model, which not only solves the model aggregation for DACFL but also ensures model consistency as much as possible. To improve the performance with non-i.i.d data, each user also takes the neighbors' average model as its next-round initialization, which prevents the possible local over-fitting. Besides, we also provide a basic theoretical analysis of DACFL on the premise of i.i.d data. The result validates the feasibility of DACFL in both time-invariant and time-varying topologies and declares that DACFL outperforms existing studies, including CDSGD and D-PSGD, in most cases. Take the result on Fashion-MNIST as a numerical example, with i.i.d data, our DACFL achieves 19$\sim$34% and 3$\sim$10% increases in average accuracy; with non-i.i.d data, our DACFL achieves 30$\sim$50% and 0$\sim$10% increases in average accuracy, compared to CDSGD and D-PSGD.

Keywords: decentralized sensors network; dynamic average consensus; federated learning

check for updates

Citation: Chen, Z.; Li, D.; Zhu, J.; Zhang, S. DACFL: Dynamic Average Consensus-Based Federated Learning in Decentralized Sensors Network. *Sensors* **2022**, *22*, 3317. https://doi.org/10.3390/s22093317

Academic Editor: DaeEun Kim

Received: 2 April 2022
Accepted: 25 April 2022
Published: 26 April 2022

1. Introduction

The unprecedentedly growing intelligent devices with smart sensors are providing a vast amount of privacy-sensitive data, which are usually related to the device owners. According to the General Data Protection Regulation (GDPR) [1], how to utilize these data in a privacy-preserving way has become a critical issue in the smart sensors network. To this end, federated learning (FL) [2,3] advocates training the machine learning model and storing the data locally, uploading only the parameters to a central parameter server (PS) for model fusion. However, there are defects in FL because of relying on a centralized topology (Figure 1a). For example, the PS iteratively synchronizes multiple local models from user devices and sends back the result to them, which leads to an extremely imbalanced communication burden of the sensors network. In detail, multiple devices communicate with the PS concurrently, so the communication traffic jam is likely to happen to the PS, especially in a sensors network where the bandwidth may be usually low. What is worse, if the PS suffered a single point of failure, the FL would be paralyzed.

To alleviate the bottlenecks aroused by centralized topology, an intuitive idea is to facilitate FL in a decentralized topology without a PS (Figure 1b). Hopefully, existing studies on device-to-device (D2D) communication have conferred the communication ability in a decentralized sensors network [4–6]. Therefore, we argue that it is not only important but also applicable for a sensors network to apply FL in a decentralized way. Actually, there have already been some studies investigating FL in a decentralized topology (refer to Section 2.2). Apart from them, references [7,8] are the two most similar studies to our work. In [7], a consensus-based distributed SGD (CDSGD) for collaborative deep learning over a fixed (time-invariant) topology is proposed, which enables data parallelization and decentralized computation. However, there are two main limitations with CDSGD. (i) It requires a uniform interaction matrix of which the elements are identical; (ii) It assumes independent and identically distributed (i.i.d) data over all devices. Therefore, the CDSGD becomes infeasible for decentralized sensors network where a time-varying topology and non-i.i.d data are usually common occurrences. In [8], a decentralized parallel SGD (D-PSGD) is studied on a fixed decentralized ring topology. The D-PSGD is also inapplicable for a decentralized sensors network because there is no physical PS to perform an additional network-wide model average which is explicitly required in D-PSGD. Imagine, if all devices are required to perform a network-wide average, it would inevitably result in unacceptable communication congestion. Besides, a fixed ring topology in D-PSGD also encounters the same limitation (i) as CDSGD.

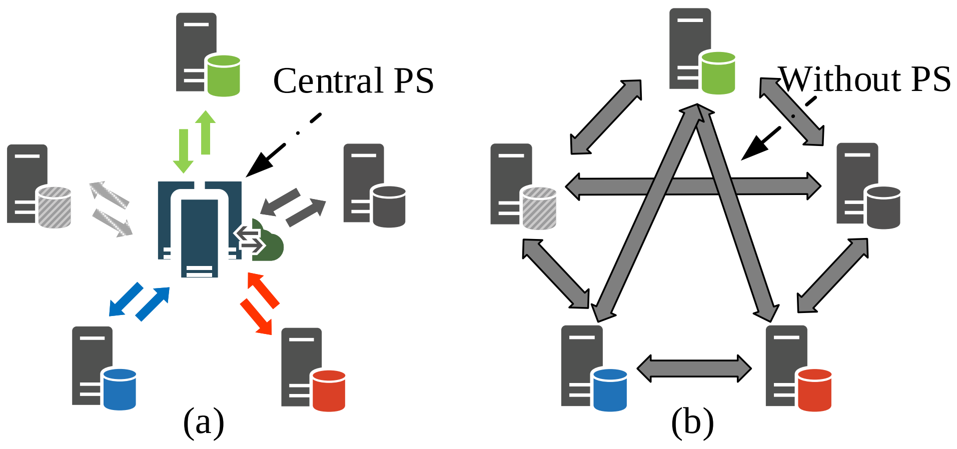


Figure 1. An overview of (**a**) centralized topology and (**b**) decentralized topology.

In this paper, we aim to facilitate FL in a more generic decentralized sensors network, involving densely and sparsely connected, as well as time-invariant and time-varying topology, with i.i.d and non-i.i.d data over user devices, while ensuring the consistency across users' models as much as possible (Although there are studies about personalized FL which obtains personalized final models [9–11], they are beyond the scope of this paper). To this end, we propose a Dynamic Average Consensus-based Federated Learning (DACFL). Our insights are with three folds. First, we transform the model aggregation of FL into a dynamic average consensus problem. Specifically, user devices are connected through an undirected graph denoted by a doubly stochastic and symmetric matrix (also called a mixing matrix). The model parameters of each device in the training procedure are regarded as a discrete-time series. In this way, the FL model aggregation, whose objective is to generate a global average model, fits well the dynamic average consensus, whose goal is to estimate the global average of all reference inputs. Second, we apply the first-order dynamic average consensus (FODAC) [12] to approximate the average model, which solves the model aggregation in a decentralized way while ensuring consistency across different models. Third, to improve the performance on non-i.i.d data, each device uses its neighborhood weighted average model as its next-round model initialization, which prevents the possible local over-fitting problem during the training procedure. For a better understanding, we summarize the difference between DACFL, CDSGD, and D-PSGD in Table 1. In detail, instead of replacing roughly the model aggregation with neighbors' model average, our DACFL applies FODAC to estimate the average model of all users, which ensures the model consistency; when compared to CDSGD, our DACFL is superior in time-varying topology, sparse topology, and non-i.i.d data; when comparing to D-PSGD, our DACFL requires no additional network-wide model average which avoids communication congestion.

Table 1. Comparison with CDSGD, D-PSGD. ✓ means enabled and ✗ means disabled in our result.

Solution	Model Aggregation	Network-Wide Average	Time-Invariant	Time-Varying	Dense	Sparse	i.i.d	Non-i.i.d
CDSGD [7]	replace by neighbors' average	not required	✓	✗	✓	✗	✓	✗
D-PSGD [8]	replace by neighbors' average	required	✓	✓	✓	✓	✓	✓
DACFL (ours)	by FODAC	not required	✓	✓	✓	✓	✓	✓

The contributions of this paper are summarized as follows:

- This paper devises a new decentralized FL implementation coined as DACFL, which applies to a more generic decentralized sensors network topology while ensuring consistency across different users. Unlike CDSGD and D-PSGD roughly replacing the model aggregation with neighbors' average, the DACFL treats each device's local training as a discrete-time process and applies FODAC to estimate the average model, through which the devices can obtain a near-average model in the absence of PS during the training procedure.
- We provide a basic theoretical convergence analysis of DACFL with some assumptions. The numeric result offers a convergence guarantee of DACFL and reveals a positive correlation of the convergence speed to the learning rate and a negative correlation to the topology size. Specific experimental results also support our analysis.
- A line of experiments on public datasets show that our DACFL outperforms CDSGD and D-PSGD w.r.t Average of Acc and Var of Acc in most cases.

2. Related Works

In this section, we first provide a brief introduction to FL and then summarize existing studies about decentralized FL implementations and about dynamic average consensus.

2.1. Federated Learning

As per [3], FL can be categorized into Horizontal Federated Learning (HFL) [2,13], Vertical Federated Learning [14–16] and Federated Transfer Learning [17–19] based on the distribution characteristics of the data. Throughout this paper, we focus only on the HFL. In what follows, we present the basic concept of HFL.

In HFL, a distributed training model is executed by a number of devices that share local model updates with a central PS who aggregates these updates to build a global model. Generally, an FL scenario consists of two main phases, local update, and global aggregation. In the local update phase, devices compute the gradients to minimize the underlying loss function using their local data. While in the global aggregation phase, the PS collects model updates from devices, aggregates them to form a global model, and sends back the global model to devices for their next training.

Formally, suppose there is a subset of devices $C \subseteq N$ selected by the PS at training epoch $t \leq T$. Each device $c \in C$ keeps a local dataset $D_c = \{X_c, Y_c\}$, where $X_c \in \mathbb{R}^{|D_c| \times d}$

represents the feature space of device c's training data and $Y_c \in \mathbb{R}^{|D_c| \times m}$ is the associated label space. Let $\ell(\omega; x_i, y_i)$ denote the loss function of data sample x_i, where ω is the model parameters, then the local loss of device c over training dataset D_c can be expressed as

$$f_c(\omega) = \frac{1}{|D_c|} \sum_{i \in D_c} \ell(\omega; x_i, y_i). \tag{1}$$

Then the global loss across all devices can be given as

$$f(\omega) = \sum_{c=1}^{|C|} \frac{|D_c|}{|D|} f_c(\omega), \tag{2}$$

where $D = \bigcup_c D_c$ represents for the whole training dataset over devices subset C and $|D| = \sum_{c=1}^{|C|} |D_c|$ denotes the total number of the data samples.

To solve the above-distributed optimization problem, an incomplete list of studies have offered their solutions [2,20–26]. In [2], the FederatedAveraging (FedAvg) is first advocated to combine local stochastic gradient descent (SGD) on each device with a server that performs model averaging. To address the communication bottleneck and the scalability of FL, a Federated Learning method with Periodic Averaging and Quantization (FedPAQ) is proposed [20], which consists of server periodic model averaging, partial device participation, and quantized message passing. In [21], a secure aggregation framework Turbo-Aggregate reduces the model aggregation overhead from O(N^2) to O(NlogN) by employing a multi-group circular strategy with additive secret sharing and novel coding techniques. To reduce the up-link communication overhead and improve the performance on non-i.i.d data, a Semi-Federated Learning (Semi-FL) framework divides users into multiple clusters and uploads only the cluster heads' models to the server [22]. By carefully designing an in-cluster sequential training manner, Semi-FL improves the performance on non-i.i.d data. Similar to Semi-FL, ref. [23] also divides users into clusters and devises a client-edge-cloud Hierarchical Federated Learning (Hier-FL), which reduces the costly communication with the cloud. In [24], a layers-wise Federated Matched Averaging (FedMA) is proposed for convolutional neural networks (CNNs) and long-short-term memory (LSTM) to address the data heterogeneity. In [25], the authors propose Federated Learning Based on a SPADE MAS (FLaMAS), which designs a multi-agent system to enable flexibility and dynamism in FL. In [26], a Federated Learning-Based Graph Convolutional Network (FedGCN) is proposed to process non-Euclidean data. All the above studies address the FL from a theoretical or practical point of view. However, they all rely on a centralized topology where a central PS is required to execute the global aggregation, which poses challenges including imbalanced communication burden and the single point of failure when applied to a sensors network. For a more comprehensive study of FL, please refer to [3,27–31].

2.2. Decentralized Implementation of Federated Learning

There have already been some studies enabling FL into a decentralized topology. A fully decentralized FL framework is proposed in [32], where users take a Bayesian-like approach to iterate and aggregate the beliefs of their one-hop neighbors and collaboratively estimate the global optimal parameter. In [33], a peer-to-peer approach, BrainTorrent is proposed targeting towards medical applications in which all clients are pair-wisely connected and update models by checking the local model version with the latest model version over the network. However, references [32,33] do not afford sufficient flexibility for users to manipulate compute-graph or node-level data sharing preferences. To this end, a universal framework, Scatterbrained, is therefore proposed [34]. In [35], a gossip communication protocol based on SGD, GossipGraD, is designed for scaling deep learning on large-scale systems without a PS, which reduces overall communication complexity and also enables asynchronous communication. Similarly, references [36–38] also design decentralized FL based on gossip protocol. In [36], the Combo is designed based on the gossip protocol and a model segmentation level synchronization mechanism, which is then extended to a bandwidth aware solution by greedily choosing the bandwidth-sufficient worker to reduce the transmission delay, called BACombo [37]. Furthermore, an experimental study compares gossip learning with FL and finds gossip learning is comparable to FL [38]. In addition to the gossip protocol, blockchain also assists the decentralized implementation of FL [39–45]. In [39], a blockchain-enabled FL (FL-Block) scheme enables the autonomous machine learning without any centralized authority to maintain the global model and coordinates by using a Proof-of-Work consensus mechanism of the blockchain, which improves the privacy issue and insufficient performance of fog computing. A crowdsourcing framework, CrowdSFL, where users can implement crowdsourcing with less overhead and higher security is proposed by combining FL with blockchain [40]. In [41], the BFLC framework uses blockchain to store the global model and exchange the local model update. In [45], a blockchain-assisted decentralized FL (BLADE-FL) is developed with an upper bound on the global loss function, based on which the authors optimize the computing resource allocation and explore the impact of lazy clients. In [42], a decentralized paradigm for big data-driven cognitive computing is developed by using a blockchain-enabled FL to introduce an incentive mechanism to solve the data island problem with privacy protection. An overview of the fundamentals of FL and blockchain is presented in [43]. The authors also propose the FLchain in mobile-edge computing networks by integrating FL with blockchain. In [44], the authors propose an innovative FL with asynchronous convergence (FedAC) considering a staleness coefficient by using a blockchain network to aggregate the global model, which avoids real-world issues such as interruption by abnormal local device training failure, dedicated attacks, etc.

Based on the above illustration, existing decentralized FL (DFL) implementations can be summarized into two main categories, DFL based on gossip protocol and DFL based on blockchain. Our solution differentiates these two categories from the following perspectives. (i) The methods are different. Unlike existing studies using gossip protocol or blockchain to accomplish the model aggregation in a decentralized topology, we first transform the model aggregation problem into a dynamic average consensus problem and then employ the FODAC to approximate the average model, thus tackling the model aggregation in a decentralized topology. (ii) The results achieved are different. Compared to DFL based on gossip protocol which usually requires a pair-wisely connected topology, our DACFL is robust to a more generic topology, not necessarily pair-wisely connected or even sparsely connected. The only prerequisite is a symmetric and doubly stochastic mixing matrix. As a result, our solution reduces the overall communication burden. While when compared to DFL based on blockchain, which is usually with high computational complexity, our solution is more less computationally complex. Therefore, the DACFL is more practically feasible when applied to a decentralized sensors network where the devices are usually with sparing capabilities of communication and computation.

2.3. Dynamic Average Consensus

The dynamic average consensus problem is referred to as the problem in which a set of autonomous agents aims to track the average of individually measured time-varying signals by local communication with neighbors. Existing papers have studied the dynamic average consensus problem regarding the continuous-time reference inputs. In [46], the authors use standard frequency-domain techniques and show that their algorithm can track the average of ramp reference inputs with zero steady-state error. In the context of input-to-state stability, the authors [47] show that a proportional dynamic average consensus algorithm can track with bounded steady-state error the average of bounded reference inputs with bounded derivatives. In [48], the authors propose a dynamic consensus algorithm and apply it to design consensus filters. Their algorithm can track with some bounded steady-state error the average of a common reference input with a bounded derivative. In [49], the authors further assume that agents know the nonlinear model, which generates the time-varying reference function. In [50], the l_1-regularized H_∞ filtering is introduced to solve the estimation problem. In [51], a tracking algorithm for sparse and dynamic underwater sensor networks based on particle filter (TASD) is proposed to improve the slow convergence rate and low filtering accuracy of the traditional particle filter. While for the dynamic average consensus problem regarding the discrete-time reference inputs, reference [12] proposes a class of discrete-time dynamic average consensus algorithms and analyzes their convergence properties. The algorithms can track a class of time-varying reference inputs, including polynomials, logarithmic-type functions, periodic functions, and other functions whose n-th-order differences are bounded, for $n \geq 1$, with zero or sufficiently small steady-state error.

For our decentralized FL implementation in this paper, we employ the first-order dynamic average consensus (FODAC) algorithm [12] (see Algorithm 1) for each device to track the average model during the training procedure in the absence of central PS.

Algorithm 1: First-order dynamic average consensus [12].

```
  Input: Reference inputs of N nodes: $r_i(t), (i = 1, 2, \cdots, N)$;
  Output: Consensus states of N nodes: $x_i(T), (i = 1, 2, \cdots, N)$;
1 Initialize: mixing matrix $\mathbf{W} = \left[\mathbf{w}_{ij}\right] \in \mathbb{R}^{N \times N}$; initialized consensus state:
    $x_i(0) = r_i(0)$; number of iterations: $T$;
2 for $t = 0, 1, 2, \cdots, T$ do
3     $x_i(t+1) = x_i(t) + \sum_{j \neq i} \mathbf{w}_{ij}\left(x_j(t) - x_i(t)\right) + \Delta r_i(t)$;
4     $\Delta r_i(t) = r_i(t) - r_i(t-1)$;
5 end
```

3. System Model and Problem Formulation

This section first provides the node model and communication model, then formally constitutes the decentralized FL implementation as a minimization optimization problem.

3.1. Node Model

A node model refers to a device (In the rest of the paper, we no longer distinguish user, device, and node), which contains a local dataset and a local model. Suppose there are N nodes in a decentralized topology, labeled by $i \in V = \{1, 2, \cdots, N\}$. The local dataset on the i-th node is D_i, and the whole dataset is $D = D_1 \cup \cdots \cup D_N$, where $D_i \cap D_j = \varnothing$ if $i \neq j$. The local model of node i at round t is represented by ω_i^t. Generally, the models on all nodes are required to be structural identical and initialized by the same parameters, i.e., $\omega_1^0 = \omega_2^0 = \cdots = \omega_N^0 = \omega^0$. In the local update phase, each node trains its ω_i^t based on D_i and generates a discrete-time series $\boldsymbol{\omega}_i = \{\omega_i^0, \omega_i^1, \cdots, \omega_i^T\}$. We denote $\bar{\omega}^t = \frac{1}{N}\sum_{i=1}^{N} \omega_i^t$ the average model over N nodes at round t. Therefore, the global aggregation phase aims to complete the $\bar{\omega}^t$. In the following part, we provide an estimation method to approximate the $\bar{\omega}^t$ in the absence of a PS.

3.2. Communication Model

A communication model refers to two rules that govern the information exchange between all nodes. (i) A connectivity rule ensuring that the information of each node influences the information of any other nodes; (ii) A rule on connection weights that a node uses when combing its own information with the information received from its neighbors. In practice, the connection weights may be affected by the distance or the channel interference, e.g., a longer distance or a more severe interference corresponds to smaller connection weights and vice versa.

The decentralized topology at round t is represented as an undirected graph $\mathcal{G}(t) = (V, E(t))$, where V is a node set and $E(t) \subset V \times V$ is an edge set. We call node i and j neighbors to each other if $(i, j) \in E(t)$, which indicates node i and j are enabled one-hop communication. A connected graph is required such that the information of node i can influence the information of any other nodes directly or indirectly. For simplicity, we use a mixing matrix $\mathbf{W}(t) = \left[\mathbf{w}_{ij}(t)\right] \in \mathbb{R}^{N\times N}$ to denote the graph with $\mathbf{w}_{ij}(t)$ representing for the connection weights demonstrated in the above rule (ii) between node i and j, where

$$\begin{cases} 0 < \mathbf{w}_{ij}(t) < 1, \text{if } (i,j) \in E(t) \\ \mathbf{w}_{ij}(t) = 0, \text{if } (i,j) \notin E(t) \end{cases}. \tag{3}$$

Here $\mathbf{W}(t)$ is required to be symmetric and doubly stochastic, i.e., $\mathbf{W}(t)\mathbf{1} = \mathbf{1}$, $\mathbf{1}^{\mathrm{T}}\mathbf{W}(t) = \mathbf{1}^{\mathrm{T}}$.

3.3. Problem Formulation

Based on the above models, an important remaining problem is to execute the global aggregation phase without PS. In the following, we transform the global aggregation in a decentralized topology into a dynamic average consensus problem. Specifically, we treat the local update phase as a discrete-time process and take the intermediate model parameters $\boldsymbol{\omega}_i = \{\omega_i^0, \omega_i^1, \cdots, \omega_i^T\}$ as the reference inputs on node i. Then, the global aggregation whose goal is to obtain the average model can be transformed into a dynamic average consensus problem, of which the objective is to track the average of the reference inputs over all nodes,

$$\min_{\boldsymbol{x}^t} \left\| \boldsymbol{x}^t - \bar{\omega}^{t-1}\mathbf{1} \right\|_2^2. \tag{4}$$

Here $\boldsymbol{x}^t = \left[x_1^t; x_2^t; \cdots; x_N^t\right]$ is a vector denoting all the estimations on N nodes at round t. In this paper, we employ the FODAC (Algorithm 1) to solve the problem in Equation (4). Besides, as is shown in (2), the objective of FL is to minimize the global loss. So, we formally combine Equations (2) and (4) and summarize the objective of the decentralized FL as

$$\begin{cases} \min\limits_{\omega} f(\omega) := \sum_{i=1}^{N} \frac{|D_i|}{|D|} f_i(\omega) \\ \min\limits_{\boldsymbol{x}^t} \left\| \boldsymbol{x}^t - \bar{\omega}^{t-1}\mathbf{1} \right\|_2^2 \end{cases}. \tag{5}$$

Especially, if each node holds the same number of training data, Equation (5) can be further expressed as

$$\begin{cases} \min\limits_{\omega} f(\omega) := \frac{1}{N} \sum_{i=1}^{N} f_i(\omega) \\ \min\limits_{\boldsymbol{x}^t} \left\| \boldsymbol{x}^t - \bar{\omega}^{t-1}\mathbf{1} \right\|_2^2 \end{cases}. \tag{6}$$

In Section 4, we design a DACFL algorithm to solve this problem.

4. Methods

In this section, we first construct the symmetric doubly stochastic matrix, which is further used to connect multiple user devices. Then, we briefly introduce the first-order dynamic average consensus and apply it to design the DACFL for implementing FL in a decentralized sensors network. Figure 2 shows an overview of our solution, which is explained in Section 4.3.

4.1. Construct a Symmetric Doubly Stochastic Matrix

As illustrated in Section 3, the mixing matrix is essential for our decentralized communication model. Therefore, it is important to construct the mixing matrix beforehand. Actually, a very simple doubly stochastic and symmetric matrix could be designed as $\mathbf{W} := \left[\frac{1}{n}\right]^{n\times n}$. However, to get a more generic mixing matrix, we use the Sinkhorn–Knopp algorithm [52] to design the doubly stochastic and symmetric matrix in this paper.

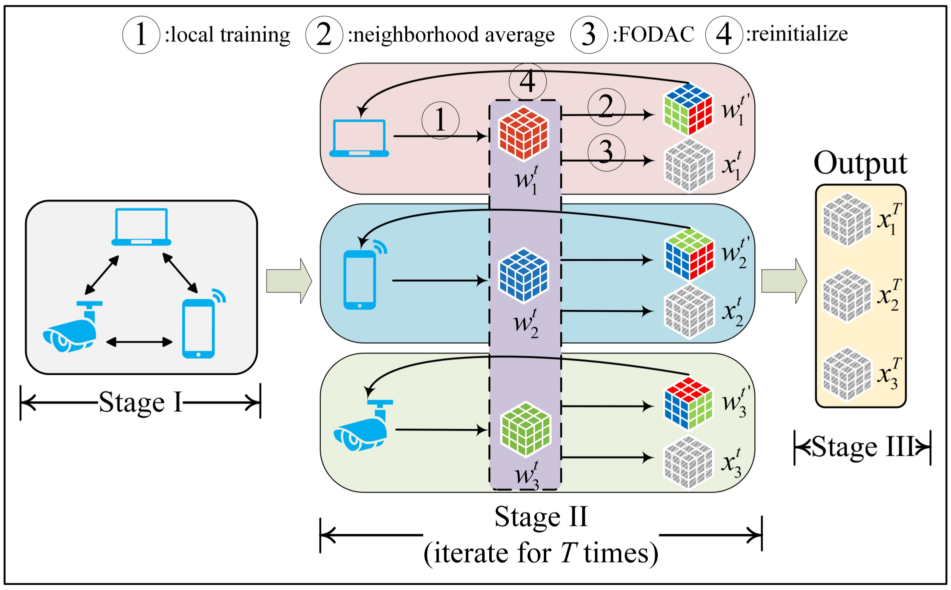


Figure 2. An overview of DACFL. Here a simple decentralized topology with 3 devices being pair-wisely connected is used as an example. Actually, the "neighborhood average" and "FODAC" in Stage II are carried out based on the models from a device's neighbors, which contains not necessarily all models, but rely on a topology (not necessarily pair-wisely connected) instead.

4.2. First-Order Dynamic Average Consensus

As in Section 3.3, by treating the intermediate models during the local update phase as the discrete-time reference inputs, the average model can be approximated by a dynamic average consensus algorithm. In this paper, we use the FODAC to approximate the average model in our DACFL, which has been proved to track the average state with either a zero steady-state error or an upper-bounded steady-state error [12]. Algorithm 1 briefly shows thepseudo-codee of FODAC.

4.3. Dynamic Average Consensus-Based Federated Learning

In Equation (5), the objective of decentralized FL implementation is to simultaneously minimize a global loss and solve a dynamic average consensus problem. For the former sub-problem, a distributed stochastic gradient descent can be used here-in, while for the latter sub-problem, we employ the FODAC to solve it. By combining the distributed gradient descent and the FODAC, we devise an algorithm, Dynamic Average Consensus-based Federated Learning (DACFL). Figure 2 shows an overview of the DACFL, which is constituted of three main stages. In stage I, multiple user devices are connected through the mixing matrix, which can be constructed by the methods introduced in Section 4.1. In stage II, each device parallel performs the training procedure. In stage III, the estimated average models are output as the final result. Four main steps constitute the stage II of DACFL: (i) each device trains its own model using its local data; (ii) each device computes a neighborhood weighted average model $\omega_i^{t'}$ by exchanging its intermediate model with its neighbors; (iii) each device performs the FODAC to track the average model. (iv) each device takes the neighborhood weighted average model as its next-round initialization. More specifically, step (i) can also be referred to as the local update phase in FL. In step (ii), the neighborhood weighted average model is further used as the device's next-round initialization in step (iv) (line 6, Algorithm 2), which is empirically demonstrated more robust to sparse topology and non-i.i.d data as it to some extent prevents the local over-fitting. In step (iii), we employ the FODAC to estimate the average model of all users in a decentralized way, which helps to handle the global aggregation phase without a central PS. The pseudo-code of the DACFL training procedure is summarized in the Algorithm 2.

Algorithm 2: Dynamic Average Consensus-based Federated Learning.

```
   Input: mixing matrix: $\mathbf{W}(t) = [\mathbf{w}_{ij}(t)] \in \mathbb{R}^{N\times N}$;
   initialized models of $N$ nodes: $\omega_1^0 = \omega_2^0 = \cdots = \omega_N^0$;
   initialized consensus states of $N$ nodes: $x_1^0 = x_2^0 = \cdots = x_N^0$;
   number of communication rounds $T$;
   number of nodes $N$; learning rate $\lambda$;
   Output: consensus states of $N$ nodes: $x_1^T, x_2^T, \cdots, x_N^T$;
1  Initialize: $\boldsymbol{x}^0 := [x_1^0, x_2^0, \cdots, x_N^0]$; $\boldsymbol{\omega}^0 := [\omega_1^0, \omega_2^0, \cdots, \omega_N^0]$; $\boldsymbol{x}^0 = \boldsymbol{\omega}^0$;
2  for each node $i \in 1, 2, \cdots, N$ parallel do
3      for $t = 0, 1, 2, \cdots, T-1$ do
4          Compute the neighborhood weighted average model: $\omega_i^{t'} = \sum_{j=1}^N \mathbf{w}_{ij}(t)\omega_j^t$;
5          Randomly sample a batch training examples $\zeta_i^t$ from $D_i$;
6          Use neighborhood weighted average model to reinitialize and update the
            local model: $\omega_i^{t+1} \leftarrow \omega_i^{t'} - \lambda\nabla f_i\left(\omega_i^{t'}; \zeta_i^t\right)$;
           /* $f_i(\cdot)$ indicates the loss function on node $i$; $\nabla$ indicates the
              gradient operator;                                                    */
7          Perform the average consensus by FODAC: $\Delta\omega_i^t = \omega_i^t - \omega_i^{t-1}$;
              /* $\omega_i^{-1} = \omega_i^0$                                          */
8          $x_i^{t+1} = \sum_{j=1}^N \mathbf{w}_{ij}(t)x_j^t + \Delta\omega_i^t$;
9      end
10 end
```

5. Convergence Analysis

In this section, we provide a basic theoretical proof of our solution with i.i.d data. Without loss of generality, each user is assumed to hold the same number of training data such that the loss function $f(x)$ can be denoted as

$$f(\omega) := \frac{1}{N}\sum_{i=1}^{N} f_i(\omega). \tag{7}$$

To complete the analysis, we make the following assumptions.

Assumption 1 (L-Smooth)**.** *Each loss function $f_i(\omega)$ is L-smooth such that*

$$f_i(v) \leq f_i(w) + (v-w)^{\mathrm{T}}\nabla f_i(w) + \frac{L}{2}\|v-w\|_2^2. \tag{8}$$

In decentralized FL, each user holds its local dataset D_i and gradient $\nabla f_i(\omega)$. To declare the user-wise gradients $\nabla f_i(\omega)$, we have Assumptions 2 and 3 on the premise of i.i.d data.

Assumption 2 (Bounded Gradients)**.** *For each user i and a randomly sampled batch data ζ_i, there exists an upper bound $G > 0$ such that*

$$\mathbb{E}_{\zeta_i\sim D_i}\|\nabla f_i(\omega;\zeta_i)\|^2 \leq G^2. \tag{9}$$

Assumption 3 (Uniform Gradient First-order Difference)**.** *At any round t, define*

$$\Delta g_i^t = \nabla f_i(\omega^{t'};\zeta_i) - \nabla f_i(\omega^{t-1'};\zeta_i) = g_i^t - g_i^{t-1}$$

as the first-order difference of gradient, this paper assumes an i.i.d data distribution across all users for the convergence analysis, such that

$$\mathbb{E}\left[\Delta g_i^t\right] = \mathbb{E}\left[\Delta g_j^t\right] = \Delta g^t, \forall i, j, 1 \leq i, j \leq N. \tag{10}$$

Assumption 4 (Bounded First-order Differences of Model Parameter)**.** *At any round t, there is a constant $\theta > 0$ ensuring an upper bound of each user's model parameters such that*

$$\left\|\omega_i^t - \omega_i^{t-1}\right\|^2 \leq \theta^2. \tag{11}$$

In practice, a sufficiently small learning rate λ guarantees this assumption. Following the Assumption 4, we have

$$\Delta\omega_{\max}^t - \Delta\omega_{\min}^t \leq \kappa, \tag{12}$$

where κ is an upper bound related to θ. We also define

$$\begin{aligned} \Delta\omega_i^t &:= \omega_i^t - \omega_i^{t-1}, \\ \Delta\omega_{\max}^t &:= \max_{i=1,2,\dots,N} \Delta\omega_i^t, \\ \Delta\omega_{\min}^t &:= \min_{i=1,2,\dots,N} \Delta\omega_i^t. \end{aligned}$$

The above (12) ensures the FODAC tracks the average of the time-varying model parameters ω_i^t with a sufficiently small steady-state error. Detailed proof can be found in [12]. Following the above assumptions, we present the convergence rate of DACFL in Theorem 1.

Theorem 1. *Following the aforementioned assumptions, we have the average expected squared gradient norm following*

$$\begin{aligned} \frac{1}{T}\sum_{t=0}^{T-1}\left\|\nabla f(\bar{\omega}^t)\right\|^2 &\leq \frac{2}{\lambda T}\mathbb{E}\left(f(\bar{\omega}^0) - f(\bar{\omega}^T)\right) + G^2 + \frac{\theta^2}{\lambda^2} + \frac{L\theta^2}{\lambda} \\ &\leq \underbrace{\frac{2}{\lambda T}\left(f(\bar{\omega}^0) - f^*\right)}_{:\mathrm{C}_0} + \underbrace{G^2 + \frac{\theta^2}{\lambda^2} + \frac{L\theta^2}{\lambda}}_{:\mathrm{C}_1}, \end{aligned} \tag{13}$$

where f^* denotes the minimum loss value of $f(x)$.

In (13), the average squared gradient norm is bounded by $\mathrm{C}_0 + \mathrm{C}_1$, where the C_0 gradually tends to 0 when training iteration T increases. In other words, the average squared gradient norm is bounded by a learning rate related term C_1 when $T \to +\infty$. So, if $\bar{\omega}^t$ is regarded as the solution of f, the convergence is guaranteed. Besides, the final output of DACFL $\boldsymbol{x}^T = [x_1^T, x_2^T, \dots, x_N^T]$ tracks the $\bar{\omega}^T$ with a sufficiently small error, hence providing a convergence guarantee for DACFL. For detailed proof of Theorem 1, please refer to the Appendix A.

6. Experiments and Performance Evaluation

In this section, we first declare the experimental setup and then evaluate the performance of DACFL with different topology and data allocations.

6.1. Experimental Setup

6.1.1. Datasets, Allocation, Topology, and Neural Network Structure

(1) Datasets: Three public datasets including MNIST, Fashion-MNIST (FMNIST) and CIFAR-10 are used for our performance validation in this paper [53–55]. (i) MNIST: a dataset includes 70,000 images of hand-written digits, total of 10 classes, with a training set consisting of 60,000 examples and a test set consisting of 10,000 examples, respectively. (ii) Fashion-MNIST: a similar dataset comprises 28 × 28 grayscale images of 70,000 fashion products from 10 categories. (iii) CIFAR-10: a dataset consists totally of 60,000 color images

with three RGB channels, which can be classified into 10 classes. The training set has 50,000 examples and the test set has 10,000 examples.

(2) Allocation: We design two ways for data allocations. (i) i.i.d: each user is assigned the same number of training samples with a uniformly random distribution over 10 classes. (ii) non-i.i.d: the training set is sorted by labels first and then divided into multiple shards with the same training data; each user samples 2 shards without replacement.

(3) Topology: Several topology from different perspectives are designed. (i) Time-varying and time-invariant: for the time-invariant topology, we initialize the mixing matrix and keep it unchanged during the training process; for the time-varying topology, we reconstruct the mixing matrix every 10 training rounds. (ii) sparse and dense connectivity: for the sparse topology, half elements of the mixing matrix are 0 (ψ = 0.5); while for the dense topology, all elements in the mixing matrix are non-zeros (ψ = 1.0).

(4) Neural Network Structure: We use the same CNN structure for MNIST and FMNIST which contains two 5×5 convolutional layers (each layer is followed with a batch normalization and 2×2 max pooling), a fully connected layer with ReLu activation and a final softmax output layer. For CIFAR-10, we use a CNN consisting of two convolutional layers (each layer is followed with batch normalization, ReLu activation, and 2×2 max-pooling), two fully connected layers with ReLu activation, and a final softmax output layer.

6.1.2. Baselines and Performance Metrics

We compare DACFL with a centralized FL method, FedAvg [2], and another two decentralized methods, CDSGD [7] and D-PSGD [8]. For FedAvg, all users participate in each training round. For CDSGD and D-PSGD, the mixing matrix and other hyper-parameters are consistent with DACFL. Two metrics, including Average of Acc and Var of Acc are defined to indicate the performances. In detail, each user's trained model (or estimated model in DACFL) is separately test, then an averaged result of all users' test accuracy is used as Average of Acc and the variance over all users' test accuracy is used as Var of Acc. For FedAvg and D-PSGD, the final output is a only global model; therefore, the Average of Acc is actually the same as the test accuracy and the Var of Acc is always 0.

We carried out the experiments on a Ubuntu 18.04 computer with 4 RTX 2080Ti GPU cards. All the baselines and our proposed solution are implemented by Python 3.8 and Pytorch 1.8.2 with CUDA 10.2, and we use MATLAB to visualize the result. Unless otherwise specified, some important parameters are set as Table 2.

Table 2. Experimental parameters setting.

Parameter	Numeric Value
Number of nodes:	10
Training rounds:	100
Local batch size:	20
Local epoch:	1
Decaying for learning rate:	0.995
Loss function:	Cross Entropy
Learning rate:	MNIST/FMNIST: 0.001, CIFAR: 0.005

6.2. Why Choose FODAC? A Numerical Perspective

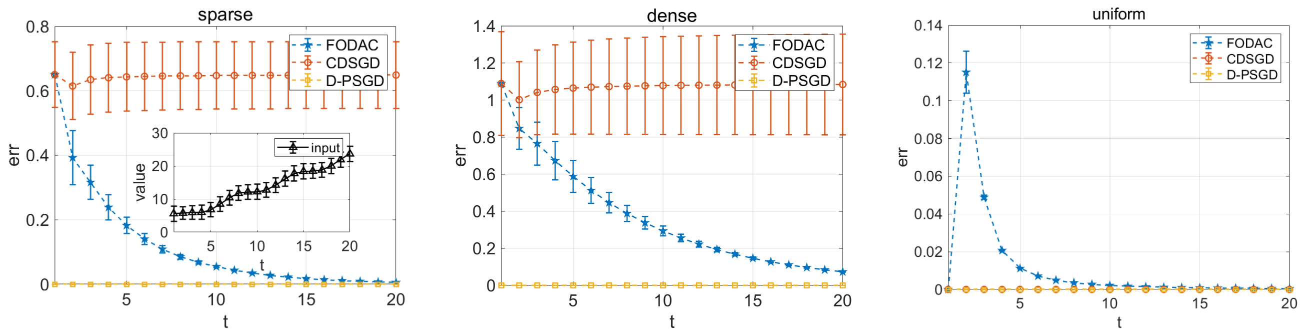
A specific numerical experiment is designed in this section to clarify the benefit of FODAC. Specifically, we separately apply FODAC, CDSGD and D-PSGD to track the average of two types of discrete-time inputs under three different mixing matrices. (i) $R_i(t) = \sin(t) + (\frac{1}{t})^i + t + i$, inputs with relatively large variance between each user; (ii) $R_i(t) = \sin(t) + (\frac{1}{t})^i + t$, inputs with relatively small variance. Here $R_i(t)$ denotes the input of i-th user at time t, where $i \in \{1, 2, \cdots, 10\}$, $t \in \{1, 2, \cdots, 20\}$.

Three 10×10 mixing matrices are defined in this experiment. (i) sparse: ψ = 0.5, i.e., half elements are 0; (ii) dense: ψ = 1.0, i.e., all elements are non-zeros; (iii) uniform:

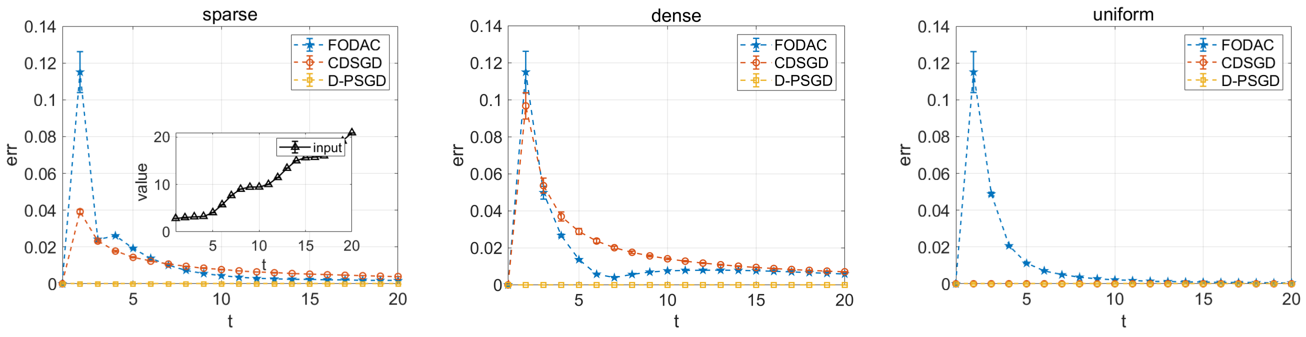
all elements are 0.1. For CDSGD, we take roughly the neighborhood weighted average as the estimated value; for D-PSGD, the estimated value is the network-wide average on the estimation by CDSGD. While for FODAC, we take the consensus state (see line 3 in Algorithm 1) as the estimated result. Then, the absolute error can be computed by

$$\text{err} = \left|\bar{R}_i(t) - \hat{R}_i(t)\right|, \tag{14}$$

here $\hat{R}_i(t)$ is the estimation and $\bar{R}_i(t)$ is the average of inputs. Figure 3 shows a comparison of these three methods.



(a)



(b)

Figure 3. The result of approximating the average by different methods. (**a**) inputs with large variance; (**b**) inputs with small variance.

With large-variance inputs, because CDSGD roughly takes the neighbors' average as the mean value, which becomes distorted with either large variance inputs or an ununiform mixing matrix. Therefore, a distinct error and variance in both sparse and dense mixing matrices in Figure 3 is observed. Comparatively, the only feasible result of CDSGD with small variance inputs or uniform mixing matrix also supports our suspect. While with small-variance inputs in Figure 3b, the FODAC still outperforms CDSGD from convergence speed in both sparse and dense mixing matrix. The D-PSGD is observed smaller error and variance than FODAC and CDSGD because it additionally carries out a network-wide average and takes this network-wide average as the mean value, which is exactly equal to the actual average value. However, such a network-wide average is practically infeasible when it comes to a decentralized sensors network where no physical PS is responsible for it. To sum up, the FODAC is more feasible than CDSGD and D-PSGD in approximating the average in a more generic decentralized topology; this actually motivates us to apply it into DACFL.

6.3. Performance on i.i.d Data

Figures 4 and 5, respectively, show the performances with i.i.d data allocation in time-invariant and time-varying topology. The parameters are set following Table 2.

6.3.1. Time-Invariant Topology

Figure 4 shows the result with i.i.d data in time-invariant topology. The following conclusions can be drawn from this result.

First, the DACFL outperforms D-PSGD and CDSGD in terms of Average Acc, albeit slightly inferior to FedAvg. From Figure 4a–c, the DACFL achieves 97%, 86%, 70% accuracy and 96%, 85%, 67% accuracy in a densely and sparsely connected topology, on three datasets, which is superior to the result of CDSGD with 93%, 64%, 18% accuracy and 68%, 51%, 20% accuracy, and the result of D-PSGD with 97%, 83%, 55% accuracy and 95%, 75%, 45% accuracy. The D-PSGD has higher accuracy than CDSGD because it additionally performs a model averaged over all users, which, however, leads to unacceptable communication congestion or even becomes practically infeasible when there is no physical PS in a decentralized sensors network. Nonetheless, due to the ununiform mixing matrix, its final accuracy is still slightly worse than FedAvg. As a deviation across different local models always exists in an ununiform mixing matrix, where the FODAC approximates the average model more effectively, which further leads to a superior accuracy for DACFL.

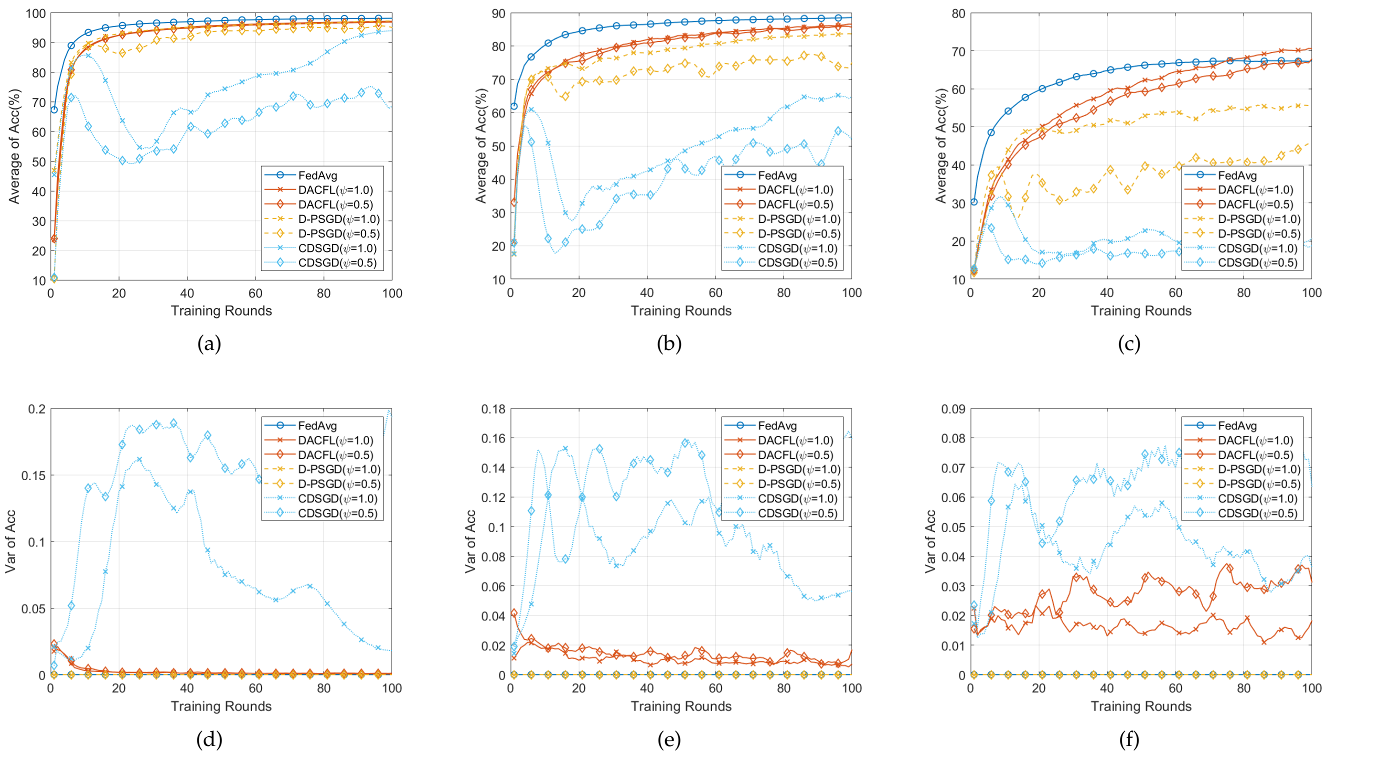


Figure 4. Performance comparison with i.i.d data and time-invariant topology. (**a**,**d**) on MNIST; (**b**,**e**) on FMNIST; (**c**,**f**) on CIFAR-10.

Second, the DACFL is more robust to the sparse topology. Specifically, the DACFL has 1%, 1%, 3% accuracy degradation on three datasets. While the D-PSGD degrades 2%, 8%, 10% accuracy, and the CDSGD degrades more than 10% accuracy. Because the FODAC is always effective as long as the mixing matrix satisfies the symmetric and doubly stochastic property, the DACFL is also feasible in sparse topology. In contrast, replacing roughly the average model with the neighbors' average is very sensitive to the sparsity of the mixing matrix. Therefore, the CDSGD and D-PSGD show more serious accuracy degradation in sparse topology.

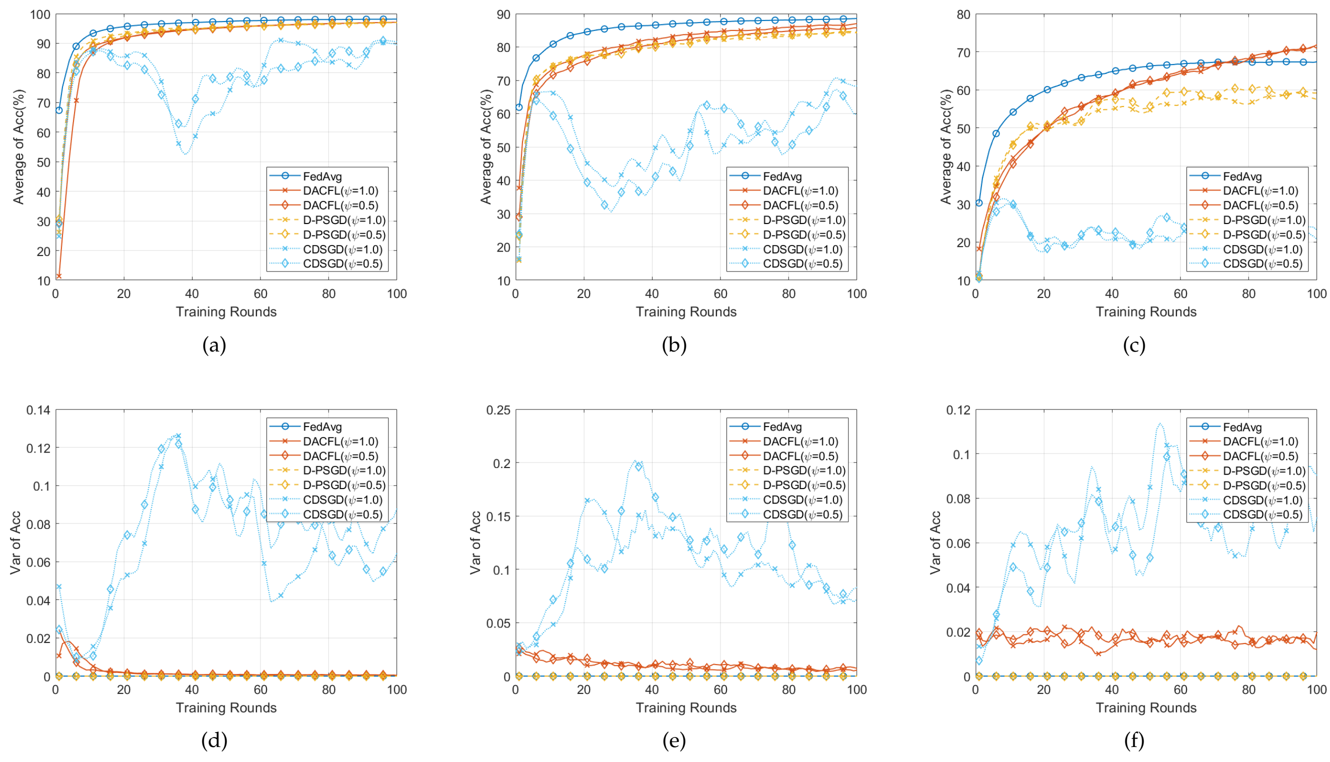


Figure 5. Performance comparison with i.i.d data and time-varying topology. (**a**,**d**) on MNIST; (**b**,**e**) on FMNIST; (**c**,**f**) on CIFAR-10.

Second, the DACFL is more robust to the sparse topology. Specifically, the DACFL has 1%, 1%, 3% accuracy degradation on three datasets. While the D-PSGD degrades 2%, 8%, 10% accuracy, and the CDSGD degrades more than 10% accuracy. Because the FODAC is always effective as long as the mixing matrix satisfies the symmetric and doubly stochastic property, the DACFL is also feasible in sparse topology. In contrast, replacing roughly the average model with the neighbors' average is very sensitive to the sparsity of the mixing matrix. Therefore, the CDSGD and D-PSGD show more serious accuracy degradation in sparse topology.

Third, the result of each user in DACFL is more consistent than that in CDSGD. Figure 4d–f show the variance of accuracy across different users. The variance in DACFL is observed to be smaller and more stable compared to CDSGD, which gradually tends to around 0 as the training progresses. This is because the FODAC is more effective in estimating the average model while the CDSGD replacing roughly the average model with the neighbors' average may generate more diversified local models.

6.3.2. Time-Varying Topology

Figure 5 presents the result on i.i.d data in time-varying topology. Because the time-varying mixing matrix is also ensured symmetric and doubly stochastic, the DACFL still outperforms D-PSGD and CDSGD. For example, in Figure 5b, the DACFL achieves an 87% accuracy better than the 84% accuracy of D-PSGD and 68% of CDSGD. Changing the topology does not affect the FedAvg, which performs better than other decentralized methods. Besides, the accuracy degradation caused by the topology sparsity becomes smaller for all decentralized methods. Especially, for D-PSGD on FMNIST (Figure 5b) and CIFAR-10 (Figure 5c), the average accuracy under a sparse topology is even greater than that under a dense topology. We suspect the randomness introduced by time-varying topology reduces the possibility of early over-fitting in a sparse topology. Finally, for the variance shown in Figure 5d–f, a similar result to that of a time-invariant topology is observed. The DACFL has smaller and more stable variance of accuracy than CDSGD in both densely and sparsely connected topology.

In summary, with i.i.d data, the DACFL is superior to CDSGD and D-PSGD in both time-varying and time-invariant topology.

6.4. Performance on Non-i.i.d Data

In this section, we test the performance of DACFL on non-i.i.d data and show the result in Figures 6 and 7. The parameters are set following Table 2.

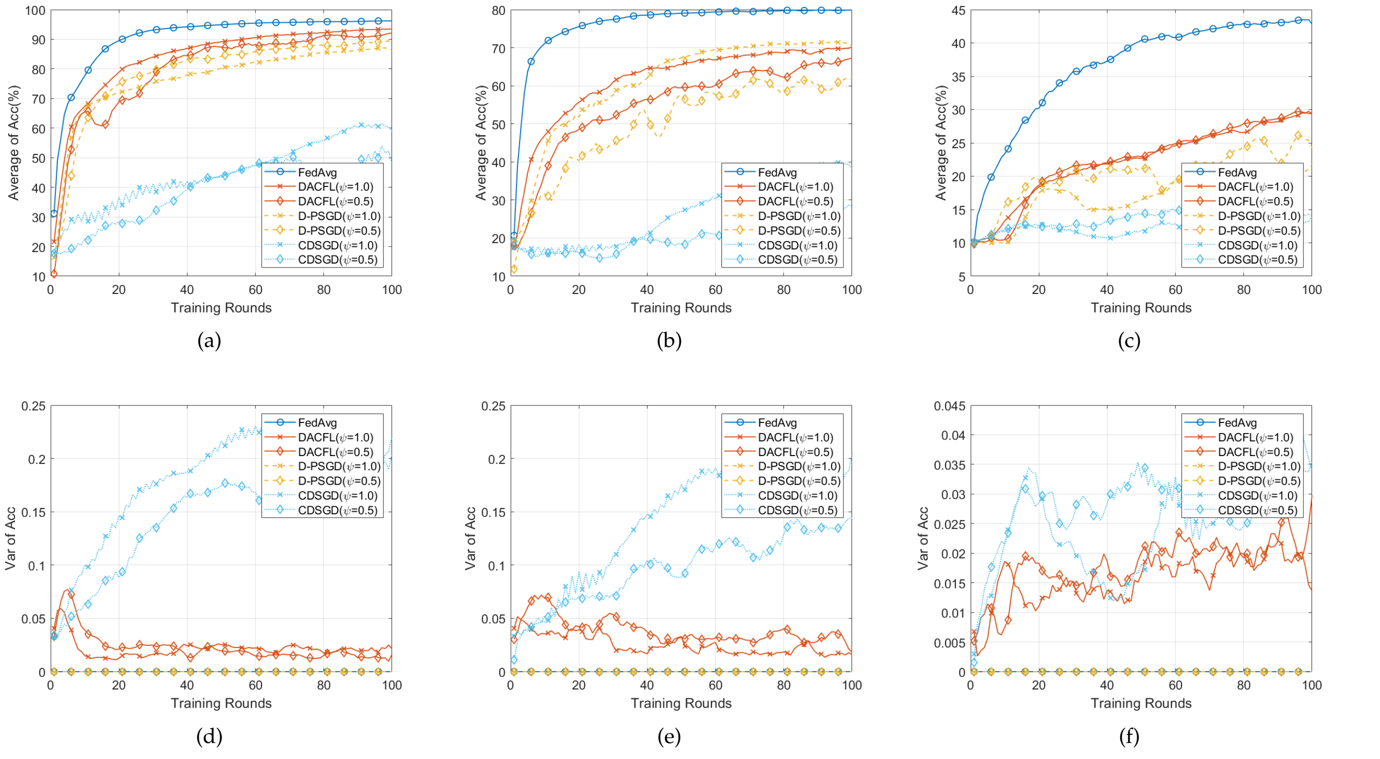


Figure 6. Performance comparison with non-i.i.d data and time-invariant topology. (**a**,**d**) on MNIST; (**b**,**e**) on FMNIST; (**c**,**f**) on CIFAR-10.

6.4.1. Time-Invariant Topology

Figure 6 presents the result on non-i.i.d data in time-invariant topology. Comparing with results on i.i.d data, all methods have accuracy degradation on three datasets. This is because the non-i.i.d property leads to users' local model divergence and early over-fitting. Since D-PSGD additionally performs a network-wide model average, it has higher accuracy than DACFL and CDSGD in MNIST and FMNIST (The result on CIFAR-10 is not counted here because all decentralized methods do not converge after 100 rounds). However, a network-wide model average usually results in an excessive communication overhead, especially when users are very dispersedly distributed in a large topology. In case of no network-wide model average, our DACFL outperforms CDSGD. Take the result in dense topology as an example; DACFL gets average accuracy of 86%, 70%, while CDSGD only achieves 58%, 40% on MNIST and FMNIST, respectively. It is the re-initialization by neighbors' weighted average model which prevents the possible local over-fitting during the training procedure, thus improving the performance of DACFL in non-i.i.d data. Besides, the variance results in Figure 6d–f also shows the superiority of DACFL compared to CDSGD. Actually, the variance is larger than that of i.i.d data due to the non-i.i.d property. This is because that non-i.i.d would inherently lead to heterogeneous local models; therefore, CDSGD replacing the average model with the neighbors' average would result in more diversified models across different users. However, the variance of

DACFL still decreases and stabilizes as the training progresses due to the effectiveness of FODAC estimating the average model.

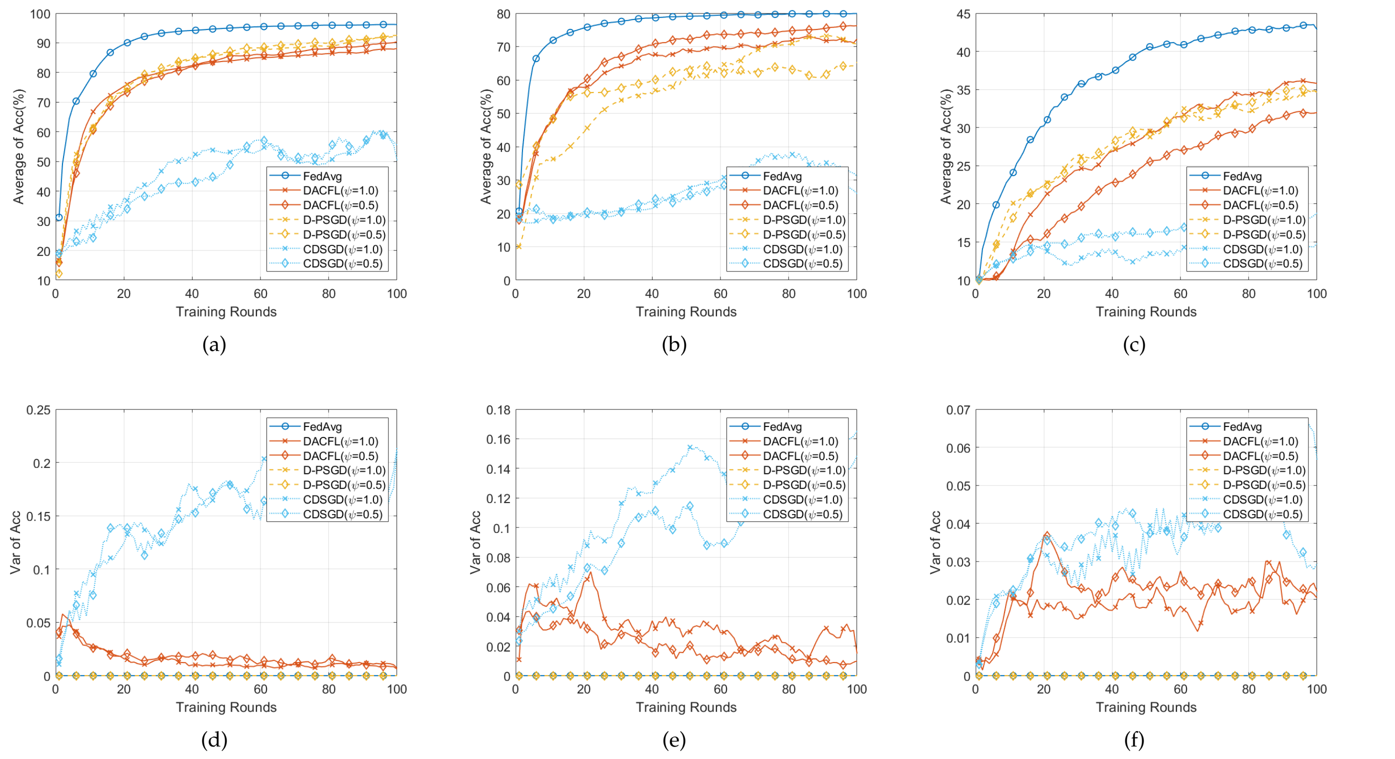


Figure 7. Performance comparison with non-i.i.d data and time-varying topology. (**a**,**d**) on MNIST; (**b**,**e**) on FMNIST; (**c**,**f**) on CIFAR-10.

6.4.2. Time-Varying Topology

Figure 7 presents the result on non-i.i.d data in time-varying topology. For the average accuracy in Figure 7a–c, similar accuracy degradation due to non-i.i.d data still exists when compared with the result on i.i.d data. Because the randomness introduced by time-varying topology possibly alleviates the local model over-fitting, our DACFL has a better performance than that of a time-invariant topology shown in Figure 6, especially on FMNIST and CIFAR-10. For the variance, similar result to Figure 6 happens, i.e., the variance of DACFL gradually decreases and tends to 0, which confirms the viability of DACFL in tracking the average model.

To sum up, the DACFL is also viable on non-i.i.d data (MNIST and FMNIST) under time-varying topology.

6.5. Convergence vs. Learning Rate and Topology Size

To find out how the learning rate and topology size affect our solution, the average test accuracy and average training loss on i.i.d MNIST with different learning rates and topology sizes are shown in Figure 8. Note only densely connected topology is considered in this part. Except for learning rate and topology size, other parameters in this experiment also follow Table 2.

6.5.1. Performance vs. Learning Rate

Figure 8a,b show the test accuracy and training loss vs. different learning rates. From Figure 8a,b, a larger λ within the range $0.001 \le \lambda \le 0.01$ leads to faster convergence. This is because a larger learning rate makes the loss function decreases with a larger step size, which leads to a faster convergence. However, this situation is the opposite when $0.05 \le \lambda \le 0.1$, i.e., when λ increases from 0.05 to 0.1, the convergence speed and convergence result become worse, with a smaller average test accuracy and larger average training loss. It is because an excessive learning rate λ would lead to a larger upper bound of first-order difference of model parameter θ and thus cause a relatively larger upper bound of first-order difference κ in (12). Consequently, a larger steady-state error while using FODAC arises. So, $\lambda = 0.01$ should be the best choice in this experiment, which gets a higher average test accuracy and lower variance while ensuring fast convergence.

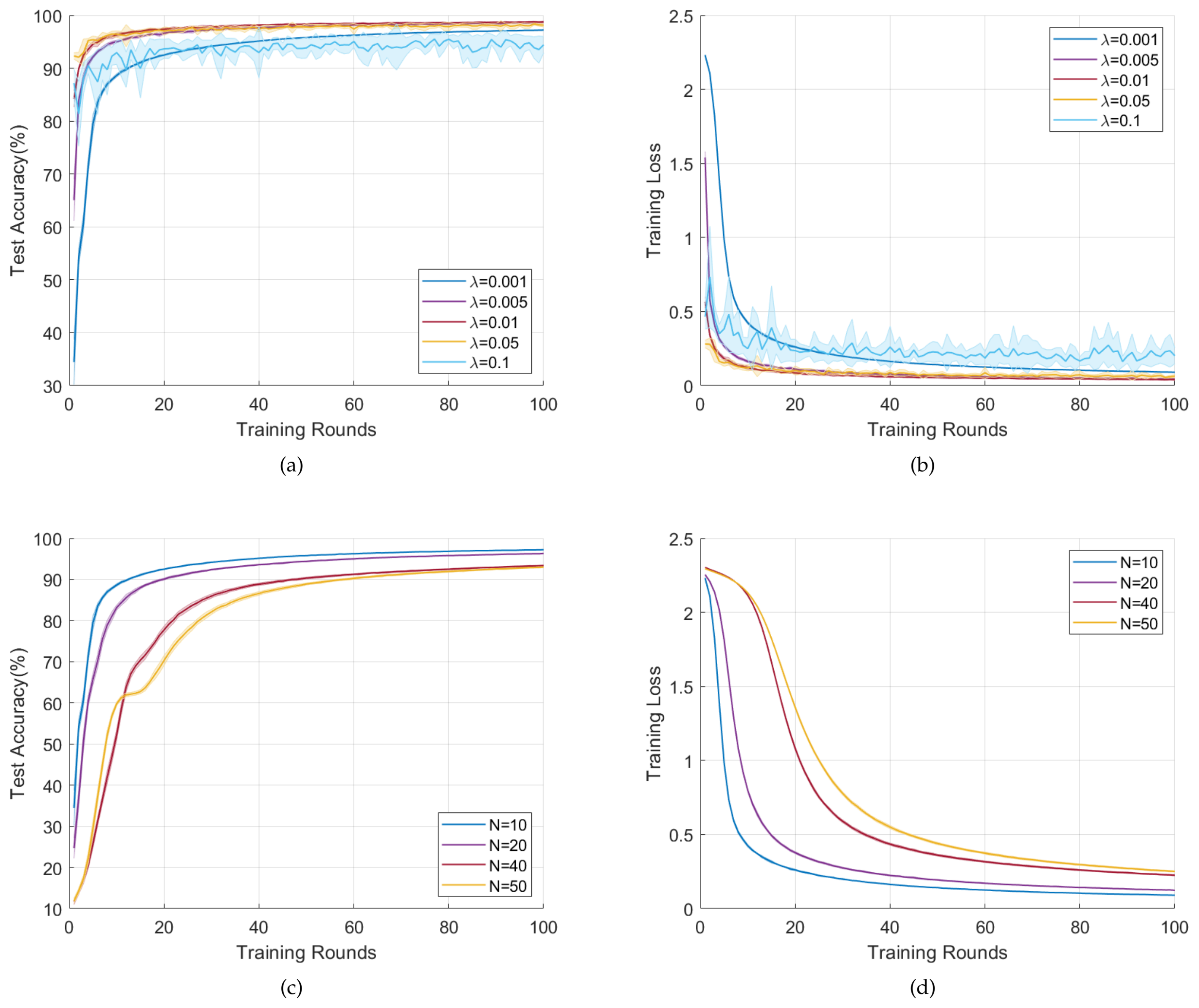


Figure 8. Performance vs learning rate and topology size. (**a**) Acc vs. λ; (**b**) Loss vs. λ; (**c**) Acc vs. N; (**d**) Loss vs. N.

6.5.2. Performance vs. Topology Size

Figure 8c,d present the result with different topology size. In Figure 8c,d, as the size N grows, the convergence speed slows down and the final average accuracy declines. We suspect a larger topology size would lead to a larger deviation across users, which further poses a challenge when using FODAC to track the average model.

In summary, a proper learning rate can accelerate the convergence of DACFL. Although DACFL is robust to different topologies, a smaller size helps attain a better performance within limited training rounds.

7. Conclusions and Future Work

Over-reliance on centralized topology poses challenges, including imbalanced communication burden and possible single point of failure, for the application of FL in smart sensors network. For this reason, existing studies have proposed different methods like

CDSGD and D-PSGD for implementing FL in a decentralized topology, which, however, have some deficiencies. Specifically, there exists distinct variance across users' models in CDSGD, while the D-PSGD additionally requires a network-wide model average which is infeasible in practical decentralized sensors network due to no physical PS or unacceptable communication congestion. Therefore, this paper devises a new decentralized FL scheme, DACFL, aiming to implement FL in a decentralized sensors network while ensuring consistency across user devices. To this end, we first transform the most critical FL model aggregation in decentralized topology into a dynamic average consensus problem by treating the local training procedure as a discrete-time series. We then employ FODAC to track the average model across all users. We also provide a basic theoretical analysis of i.i.d data, which offers a convergence guarantee of our solution. Specific experiments on different public datasets verify the feasibility of DACFL in a more generic topology and declare the superiority of DACFL over CDSGD and D-PSGD.

Some issues need further investigation. First, a more communication-efficient method for DACFL is deserved because of each device in DACFL exchanges both estimation states and local models during the training process. This is especially important in a practical sensors network, usually with the insufficient capability of communication and computation. Two high-level ways are suggested to improve communication efficiency. One is to reduce the amount of data transmitted per communication round using a line of compression techniques; the other is to reduce the number of total communication rounds by speeding up the convergence. To reduce the computational burden, methods like network pruning and binary connection would help design a lightweight NN structure to reduce the computational burden. Second, an asynchronous DACFL deserves investigation for practical application. In this paper, we simply assume a synchronization across devices which would be, however, time-consuming with heterogeneous communication and computation capabilities of devices, because a faster device should wait for a slower device until it finishes its task before exchanging model information. Therefore, a staleness-aware asynchronous training mechanism needs to be carefully designed for DACFL in the future. Third, designing a halfway-drop and halfway-join-aware DACFL is worthy of future investigation. In a practical smart sensors network, some devices may be disconnected and reconnected during the training procedure, which would destroy the mixing matrix. Therefore, how to seamlessly reconstruct a symmetric and doubly stochastic matrix is very important.

Author Contributions: Conceptualization, S.Z.; methodology, D.L.; software, Z.C.; validation, Z.C. and D.L.; formal analysis, Z.C.; investigation, Z.C. and D.L.; resources, Z.C.; data curation, Z.C.; writing—original draft preparation, Z.C.; writing—review and editing, S.Z.; visualization, Z.C.; supervision, S.Z.; project administration, J.Z.; funding acquisition, J.Z. All authors have read and agreed to the published version of the manuscript.

Funding: This research was funded by Huawei Technology Innovative Research.

Institutional Review Board Statement: Not applicable.

Informed Consent Statement: Not applicable.

Data Availability Statement: Not applicable.

Acknowledgments: This research is supported by Huawei Technology Innovative Research.

Conflicts of Interest: The authors declare no conflict of interest.

Appendix A

Appendix A.1. Preliminary

Before the detailed proof, here are some notations avoiding ambiguity. We denote the mixing matrix $\mathbf{W} = \left[\mathbf{w}_{ij}(t)\right] \in \mathbb{R}^{N\times N}$ a decentralized topology with N users, where x_i^t, ω_i^t represents for the estimation and local model of i-th user at round t, respectively. $\bar{\omega}^t = \frac{1}{N}\sum_{i=1}^{N}\omega_i^t$ is defined as the average model of all users at round t and

$\omega_i^{t'} = \sum_{j=1}^{t} \mathbf{w}_{ij}(t)\omega_j^t$ is the neighborhood weighted average. We use $g_i^t = \nabla f_i\left(\omega_i^{t'}, \zeta_i^t\right)$ to denote the stochastic gradient of user i at round t, where the $\zeta_i^t \subseteq D_i$ is uniformly sampled from D_i. The following notations are defined as

$$\boldsymbol{x}^t = \left[x_1^t, x_2^t, \cdots, x_N^t\right]^{\mathrm{T}},$$

$$\boldsymbol{\omega}^t = \left[\omega_1^t, \omega_2^t, \cdots, \omega_N^t\right]^{\mathrm{T}}.$$

In the following proof, a time-invariant topology $\mathbf{W}(t) = \mathbf{W}$ with $\mathbf{w}_{ij}(t) = \mathbf{w}_{ij}$ is considered.

Appendix A.2. Proof of the Theorem 1

Proof. According to Algorithm 2 (line 4 to line 6), $\omega_i^{t'} = \sum_{j=1}^{N} \mathbf{w}_{ij}(t)\omega_j^t, \quad \omega_i^{t+1} = \omega_i^{t'} - \lambda g_i^t$. Therefore,

$$\begin{aligned} \mathbb{E}\left[\left\|\bar{\omega}^{t+1} - \bar{\omega}^t\right\|^2\right] &= \mathbb{E}\left[\frac{1}{N^2}\left\|\sum_{i=1}^{N}\left(\omega_i^{t+1} - \omega_i^t\right)\right\|^2\right] \\ &\leq \mathbb{E}\left[\frac{1}{N}\sum_{i=1}^{N}\left\|\omega_i^{t+1} - \omega_i^t\right\|^2\right] \\ &\leq \theta^2, \end{aligned} \tag{A1}$$

where (A1) follows Assumption 4.

Take the following recursive equations as examples,

$$\begin{aligned} \omega_i^{t+1} - \omega_i^t &= \sum_{j=1}^{N} \mathbf{w}_{ij}\omega_j^t - \lambda g_i^t - \left(\sum_{j=1}^{N} \mathbf{w}_{ij}\omega_j^{t-1} - \lambda g_i^{t-1}\right) \\ &= \sum_{j=1}^{N} \mathbf{w}_{ij}\left(\omega_j^t - \omega_j^{t-1}\right) - \lambda\left(g_i^t - g_i^{t-1}\right), \end{aligned}$$

$$\begin{aligned} &\omega_j^t - \omega_j^{t-1} = \sum_{i=1}^{N} \mathbf{w}_{ij}\left(\omega_i^{t-1} - \omega_i^{t-2}\right) - \lambda\left(g_j^{t-1} - g_j^{t-2}\right), \\ &\omega_i^{t-1} - \omega_i^{t-2} = \sum_{j=1}^{N} \mathbf{w}_{ij}\left(\omega_j^{t-2} - \omega_j^{t-3}\right) - \lambda\left(g_i^{t-2} - g_i^{t-3}\right), \\ &\omega_j^{t-2} - \omega_j^{t-3} = \sum_{i=1}^{N} \mathbf{w}_{ij}\left(\omega_i^{t-3} - \omega_i^{t-4}\right) - \lambda\left(g_j^{t-3} - g_j^{t-4}\right), \\ &\vdots \\ &\omega_i^1 - \omega_i^0 = \sum_{j=1}^{N} \mathbf{w}_{ij}\left(\omega_j^0 - \omega_j^{-1}\right) - \lambda\left(g_i^0 - g_i^{-1}\right). \end{aligned}$$

Then we have

$$\begin{aligned} \omega_i^{t+1} - \omega_i^t &= \underbrace{\sum_{j=1}^{N}\mathbf{w}_{ij}\sum_{i=1}^{N}\mathbf{w}_{ij}\cdots\sum_{j=1}^{N}\mathbf{w}_{ij}}_{\Sigma_i \text{ or } \Sigma_j \text{ total } t \text{ times}}\left(\omega_j^0 - \omega_j^{-1}\right) - \underbrace{\sum_{i=1}^{N}\mathbf{w}_{ij}\sum_{j=1}^{N}\mathbf{w}_{ij}\cdots\sum_{j=1}^{N}\mathbf{w}_{ij}}_{\Sigma_i \text{ or } \Sigma_j \text{ total } t\text{-1 times}}\lambda\left(g_i^0 - g_i^{-1}\right) - \cdots \\ &\quad - \sum_{j=1}^{N}\mathbf{w}_{ij}\lambda\left(g_j^{t-1} - g_j^{t-2}\right) - \lambda\left(g_i^t - g_i^{t-1}\right) \\ &\overset{(a)}{=} -\underbrace{\sum_{i=1}^{N}\mathbf{w}_{ij}\sum_{j=1}^{N}\mathbf{w}_{ij}\cdots\sum_{j=1}^{N}\mathbf{w}_{ij}}_{\Sigma_i \text{ or } \Sigma_j \text{ total } t\text{-1 times}}\lambda\left(g_i^0 - g_i^{-1}\right) - \underbrace{\sum_{j=1}^{N}\mathbf{w}_{ij}\sum_{i=1}^{N}\mathbf{w}_{ij}\cdots\sum_{j=1}^{N}\mathbf{w}_{ij}}_{\Sigma_i \text{ or } \Sigma_j \text{ total } t\text{-2 times}}\lambda\left(g_i^1 - g_i^0\right) - \cdots \\ &\quad - \sum_{j=1}^{N}\mathbf{w}_{ij}\lambda\left(g_j^{t-1} - g_j^{t-2}\right) - \lambda\left(g_i^t - g_i^{t-1}\right) \\ &\overset{(b)}{=} -\lambda\left(\Delta g_i^t + \Delta g_j^{t-1} + \cdots + \Delta g_i^0\right) \\ &\overset{(c)}{=} -\lambda\sum_{t=0}^{t}\Delta g^t, \end{aligned}$$

where (a) follows from the initialization $\boldsymbol{\omega}^0 = \boldsymbol{\omega}^{-1}$, (b) follows from the Assumption 4, and (c) follows from Assumption 3.

Substituting the above equation into (A1), we have

$$\mathbb{E}\left[\left\|\bar{\omega}^{t+1}-\bar{\omega}^{t}\right\|^{2}\right]=\frac{\lambda^{2}}{N^{2}}\left\|\sum_{i=1}^{N}\sum_{t=0}^{t}\Delta g^{t}\right\|^{2}\leq\frac{\lambda^{2}}{N}\sum_{i=1}^{N}\underbrace{\left\|\sum_{t=0}^{t}\Delta g^{t}\right\|^{2}}_{:\mathrm{T}_0}\leq\mathbb{E}\left[\frac{1}{N}\sum_{i=1}^{N}\left\|\omega_i^{t+1}-\omega_i^{t}\right\|^{2}\right]\leq\theta^{2}. \tag{A2}$$

So T_0 can be bounded following (A2),

$$\mathrm{T}_0\leq\frac{\theta^2}{\lambda^2}. \tag{A3}$$

Given the L-smooth assumption, the following inequality holds

$$\mathbb{E}\left[f(\bar{\omega}^{t+1})\right]\leq\mathbb{E}[f(\bar{\omega}^{t})]+\underbrace{\mathbb{E}\left[\left\langle\nabla f(\bar{\omega}^{t}),\bar{\omega}^{t+1}-\bar{\omega}^{t}\right\rangle\right]}_{:\mathrm{T}_1}+\frac{L}{2}\mathbb{E}\left[\left\|\bar{\omega}^{t+1}-\bar{\omega}^{t}\right\|^{2}\right]. \tag{A4}$$

where

$$\begin{aligned}\mathrm{T}_1&=\langle\nabla f(\bar{\omega}^{t}),\frac{1}{N}\sum_{i=1}^{N}\left(\omega_i^{t+1}-\omega_i^{t}\right)\rangle=\langle\nabla f(\bar{\omega}^{t}),\frac{1}{N}\sum_{i=1}^{N}\left(-\lambda\sum_{t=0}^{t}\Delta g^{t}\right)\rangle=-\lambda\langle\nabla f(\bar{\omega}^{t}),\frac{1}{N}\sum_{i=1}^{N}\left(\sum_{t=0}^{t}\Delta g^{t}\right)\rangle\\&\overset{(d)}{=}-\frac{\lambda}{2}\left[\|\nabla f(\bar{\omega}^{t})\|^{2}+\left\|\frac{1}{N}\sum_{i=1}^{N}\left(\sum_{t=0}^{t}\Delta g^{t}\right)\right\|^{2}\right]+\frac{\lambda}{2}\left[\left\|\nabla f(\bar{\omega}^{t})-\frac{1}{N}\sum_{i=1}^{N}\left(\sum_{t=0}^{t}\Delta g^{t}\right)\right\|^{2}\right],\end{aligned} \tag{A5}$$

where (d) follows from the fundamental equation $\langle\mathbf{A},\mathbf{B}\rangle=\frac{1}{2}\left[\|\mathbf{A}\|^{2}+\|\mathbf{B}\|^{2}-\|\mathbf{A}-\mathbf{B}\|^{2}\right]$ for any vector $\mathbf{A},\mathbf{B}$.

Substituting (A2) and (A5) into (A4), we have

$$\mathbb{E}\left[f(\bar{\omega}^{t+1})\right]\leq\mathbb{E}[f(\bar{\omega}^{t})]-\frac{\lambda}{2}\|\nabla f(\bar{\omega}^{t})\|^{2}-\frac{\lambda}{2}\left\|\frac{1}{N}\sum_{i=1}^{N}\left(\sum_{t=0}^{t}\Delta g^{t}\right)\right\|^{2}+\frac{\lambda}{2}\underbrace{\left[\left\|\nabla f(\bar{\omega}^{t})-\frac{1}{N}\sum_{i=1}^{N}\left(\sum_{t=0}^{t}\Delta g^{t}\right)\right\|^{2}\right]}_{:\mathrm{T}_2}+\frac{L\lambda^{2}}{2N^{2}}\left\|\sum_{i=1}^{N}\sum_{t=0}^{t}\Delta g^{t}\right\|^{2}. \tag{A6}$$

Now let's bound the T_2,

$$\begin{aligned}\mathrm{T}_2&=\left\|\frac{1}{N}\sum_{i=1}^{N}\nabla f_i(\bar{\omega}^{t})-\frac{1}{N}\sum_{i=1}^{N}\sum_{t=0}^{t}\Delta g^{t}\right\|^{2}=\frac{1}{N^{2}}\left\|\sum_{i=1}^{N}\left(\nabla f_i(\bar{\omega}^{t})-\sum_{t=0}^{t}\Delta g^{t}\right)\right\|^{2}\\&\overset{(e)}{\leq}\frac{1}{N}\sum_{i=1}^{N}\left\|\nabla f_i(\bar{\omega}^{t})-\sum_{t=0}^{t}\Delta g^{t}\right\|^{2}\\&\overset{(f)}{\leq}\frac{1}{N}\sum_{i=1}^{N}\left(\|\nabla f_i(\bar{\omega}^{t})\|^{2}+\left\|\sum_{t=0}^{t}\Delta g^{t}\right\|^{2}\right)\\&\overset{(g)}{\leq}\frac{1}{N}\sum_{i=1}^{N}\left(G^{2}+\frac{\theta^{2}}{\lambda^{2}}\right)\\&=G^{2}+\frac{\theta^{2}}{\lambda^{2}},\end{aligned} \tag{A7}$$

where (e) follows by the inequality $\left\|\sum_{i=1}^{N}\mathbf{z}_i\right\|^{2}\leq N\sum_{i=1}^{N}\|\mathbf{z}_i\|^{2}$, (f) follows from the inequality $\|\mathbf{A}+\mathbf{B}\|^{2}\leq\|\mathbf{A}\|^{2}+\|\mathbf{B}\|^{2}$ for any vector $\mathbf{A},\mathbf{B}$, and (g) follows from Assumption 2 and (A3).

Substituting (A7) into (A6), we have

$$\begin{aligned}
\mathbb{E}\Big[f(\bar{\omega}^{t+1})\Big] &\leq \mathbb{E}\big[f(\bar{\omega}^{t})\big] - \frac{\lambda}{2}\big\|\nabla f(\bar{\omega}^{t})\big\|^2 - \underbrace{\frac{\lambda}{2}\left\|\frac{1}{N}\sum_{i=1}^{N}\left(\sum_{t=0}^{t}\Delta g^t\right)\right\|^2}_{:\mathrm{T}_3} + \frac{\lambda}{2}\left(G^2 + \frac{\theta^2}{\lambda^2}\right) + \frac{L\lambda^2}{2N^2}\left\|\sum_{i=1}^{N}\sum_{t=0}^{t}\Delta g^t\right\|^2 \\
&\overset{(h)}{\leq} \mathbb{E}\big[f(\bar{\omega}^{t})\big] - \frac{\lambda}{2}\big\|\nabla f(\bar{\omega}^{t})\big\|^2 + \frac{\lambda}{2}\left(G^2 + \frac{\theta^2}{\lambda^2}\right) + \frac{L\lambda^2}{2N^2}\left\|\sum_{i=1}^{N}\sum_{t=0}^{t}\Delta g^t\right\|^2 \\
&\overset{(i)}{\leq} \mathbb{E}\big[f(\bar{\omega}^{t})\big] - \frac{\lambda}{2}\big\|\nabla f(\bar{\omega}^{t})\big\|^2 + \frac{\lambda}{2}\left(G^2 + \frac{\theta^2}{\lambda^2}\right) + \frac{L\lambda^2}{2N}\sum_{i=1}^{N}\left\|\sum_{t=0}^{t}\Delta g^t\right\|^2 \\
&\overset{(j)}{\leq} \mathbb{E}\big[f(\bar{\omega}^{t})\big] - \frac{\lambda}{2}\big\|\nabla f(\bar{\omega}^{t})\big\|^2 + \frac{\lambda}{2}\left(G^2 + \frac{\theta^2}{\lambda^2}\right) + \frac{L\theta^2}{2} \\
&= \mathbb{E}\big[f(\bar{\omega}^{t})\big] - \frac{\lambda}{2}\big\|\nabla f(\bar{\omega}^{t})\big\|^2 + \frac{\lambda}{2}\left(G^2 + \frac{\theta^2}{\lambda^2} + \frac{L\theta^2}{\lambda}\right).
\end{aligned} \tag{A8}$$

where (h) follows because $\mathrm{T}_3 \geq 0$, (i) and (j) follow from the inequality $\left\|\sum_{i=1}^{N}\mathbf{z}_i\right\|^2 \leq N\sum_{i=1}^{N}\|\mathbf{z}_i\|^2$ aforementioned and (A2), respectively.

Rearrange the (A8), we have

$$\big\|\nabla f(\bar{\omega}^{t})\big\|^2 \leq \frac{2}{\lambda}\mathbb{E}\Big[f(\bar{\omega}^{t}) - f(\bar{\omega}^{t+1})\Big] + G^2 + \frac{\theta^2}{\lambda^2} + \frac{L\theta^2}{\lambda}. \tag{A9}$$

For (A9), we sum it over $t \in \{0, 1, 2, \cdots, T-1\}$ first and then divide both sides by T,

$$\begin{aligned}
\frac{1}{T}\sum_{t=0}^{T-1}\big\|\nabla f(\bar{\omega}^{t})\big\|^2 &\leq \frac{2}{\lambda T}\mathbb{E}\Big(f(\bar{\omega}^{0}) - f(\bar{\omega}^{T})\Big) + G^2 + \frac{\theta^2}{\lambda^2} + \frac{L\theta^2}{\lambda} \\
&\leq \frac{2}{\lambda T}\Big(f(\bar{\omega}^{0}) - f^*\Big) + G^2 + \frac{\theta^2}{\lambda^2} + \frac{L\theta^2}{\lambda},
\end{aligned} \tag{A10}$$

where the f^* is the optimum of loss function f, and this completes the proof. □

References

1. Goddard, M. The EU General Data Protection Regulation (GDPR): European Regulation that has a Global Impact. *Int. J. Mark. Res.* **2017**, *59*, 703–705. [CrossRef]
2. McMahan, B.; Moore, E.; Ramage, D.; Hampson, S.; Arcas, B.A.Y. Communication-Efficient Learning of Deep Networks from Decentralized Data. *Proc. Mach. Learn. Res.* **2017**, *54*, 1273–1282.
3. Yang, Q.; Liu, Y.; Chen, T.; Tong, Y. Federated Machine Learning: Concept and Applications. *ACM Trans. Intell. Syst. Technol.* **2019**, *10*, 1–19. [CrossRef]
4. Gandotra, P.; Kumar Jha, R.; Jain, S. A survey on device-to-device (D2D) communication: Architecture and security issues. *J. Netw. Comput. Appl.* **2017**, *78*, 9–29. [CrossRef]
5. González, E.; Casanova-Chafer, J.; Romero, A.; Vilanova, X.; Mitrovics, J.; Llobet, E. LoRa Sensor Network Development for Air Quality Monitoring or Detecting Gas Leakage Events. *Sensors* **2020**, *20*, 6225. [CrossRef]
6. Nikodem, M.; Slabicki, M.; Bawiec, M. Efficient Communication Scheme for Bluetooth Low Energy in Large Scale Applications. *Sensors* **2020**, *20*, 6371. [CrossRef]
7. Jiang, Z.; Balu, A.; Hegde, C.; Sarkar, S. Collaborative Deep Learning in Fixed Topology Networks. In *Advances in Neural Information Processing Systems*; Guyon, I., Luxburg, U.V., Bengio, S., Wallach, H., Fergus, R., Vishwanathan, S., Garnett, R., Eds.; Curran Associates, Inc.: Nice, France, 2017; Volume 30.
8. Lian, X.; Zhang, C.; Zhang, H.; Hsieh, C.J.; Zhang, W.; Liu, J. Can Decentralized Algorithms Outperform Centralized Algorithms? A Case Study for Decentralized Parallel Stochastic Gradient Descent. In *Advances in Neural Information Processing Systems*; Guyon, I., Luxburg, U.V., Bengio, S., Wallach, H., Fergus, R., Vishwanathan, S., Garnett, R., Eds.; Curran Associates, Inc.: Nice, France, 2017; Volume 30.
9. Tan, A.Z.; Yu, H.; Cui, L.; Yang, Q. Towards Personalized Federated Learning. *IEEE Trans. Neural Netw. Learn. Syst.* **2022**, 1–17. [CrossRef]
10. Fallah, A.; Mokhtari, A.; Ozdaglar, A. Personalized Federated Learning with Theoretical Guarantees: A Model-Agnostic Meta-Learning Approach. In *Advances in Neural Information Processing Systems*; Larochelle, H., Ranzato, M., Hadsell, R., Balcan, M., Lin, H., Eds.; Curran Associates, Inc.: Nice, France, 2020; Volume 33, pp. 3557–3568.

11. Kelli, V.; Argyriou, V.; Lagkas, T.; Fragulis, G.; Grigoriou, E.; Sarigiannidis, P. IDS for Industrial Applications: A Federated Learning Approach with Active Personalization. *Sensors* **2021**, *21*, 6743. [CrossRef]
12. Zhu, M.; Martínez, S. Discrete-time dynamic average consensus. *Automatica* **2010**, *46*, 322–329. [CrossRef]
13. Bonawitz, K.; Ivanov, V.; Kreuter, B.; Marcedone, A.; McMahan, H.B.; Patel, S.; Ramage, D.; Segal, A.; Seth, K. Practical Secure Aggregation for Privacy-Preserving Machine Learning. In Proceedings of the 2017 ACM SIGSAC Conference on Computer and Communications Security, Dallas, TX, USA, 30 October–3 November 2017; Association for Computing Machinery: New York, NY, USA, 2017; pp. 1175–1191. [CrossRef]
14. Hardy, S.; Henecka, W.; Ivey-Law, H.; Nock, R.; Patrini, G.; Smith, G.; Thorne, B. Private federated learning on vertically partitioned data via entity resolution and additively homomorphic encryption. *arXiv* **2017**, arXiv:1711.10677.
15. Zhang, Q.; Gu, B.; Deng, C.; Gu, S.; Bo, L.; Pei, J.; Huang, H. AsySQN: Faster Vertical Federated Learning Algorithms with Better Computation Resource Utilization. In Proceedings of the 27th ACM SIGKDD Conference on Knowledge Discovery & Data Mining, Virtual, 14–18 August 2021; Association for Computing Machinery: New York, NY, USA, 2021; pp. 3917–3927. [CrossRef]
16. Cha, D.; Sung, M.; Park, Y.R. Implementing Vertical Federated Learning Using Autoencoders: Practical Application, Generalizability, and Utility Study. *JMIR Med. Inform.* **2021**, *9*, e26598. [CrossRef] [PubMed]
17. Saha, S.; Ahmad, T. Federated transfer learning: Concept and applications. *Intell. Artif.* **2021**, *15*, 35–44. [CrossRef]
18. Maurya, S.; Joseph, S.; Asokan, A.; Algethami, A.A.; Hamdi, M.; Rauf, H.T. Federated Transfer Learning for Authentication and Privacy Preservation Using Novel Supportive Twin Delayed DDPG (S-TD3) Algorithm for IIoT. *Sensors* **2021**, *21*, 7793. [CrossRef]
19. Bowler, A.L.; Pound, M.P.; Watson, N.J. Domain Adaptation and Federated Learning for Ultrasonic Monitoring of Beer Fermentation. *Fermentation* **2021**, *7*, 253. [CrossRef]
20. Reisizadeh, A.; Mokhtari, A.; Hassani, H.; Jadbabaie, A.; Pedarsani, R. Fedpaq: A communication-efficient federated learning method with periodic averaging and quantization. In Proceedings of the International Conference on Artificial Intelligence and Statistics, Virtual, 26–28 August 2020; pp. 2021–2031.
21. So, J.; Güler, B.; Avestimehr, A.S. Turbo-Aggregate: Breaking the Quadratic Aggregation Barrier in Secure Federated Learning. *IEEE J. Sel. Areas Inf. Theory* **2021**, *2*, 479–489. [CrossRef]
22. Chen, Z.; Li, D.; Zhao, M.; Zhang, S.; Zhu, J. Semi-Federated Learning. In Proceedings of the 2020 IEEE Wireless Communications and Networking Conference (WCNC), Seoul, Korea, 25–28 May 2020; pp. 1–6. [CrossRef]
23. Liu, L.; Zhang, J.; Song, S.; Letaief, K.B. Client-Edge-Cloud Hierarchical Federated Learning. In Proceedings of the ICC 2020—2020 IEEE International Conference on Communications (ICC), Dublin, Ireland, 7–11 June 2020; pp. 1–6. [CrossRef]
24. Wang, H.; Yurochkin, M.; Sun, Y.; Papailiopoulos, D.; Khazaeni, Y. Federated Learning with Matched Averaging. *arXiv* **2020**, arXiv:2002.06440.
25. Rincon, J.; Julian, V.; Carrascosa, C. FLaMAS: Federated Learning Based on a SPADE MAS. *Appl. Sci.* **2022**, *12*, 3701. [CrossRef]
26. Hu, K.; Wu, J.; Li, Y.; Lu, M.; Weng, L.; Xia, M. FedGCN: Federated Learning-Based Graph Convolutional Networks for Non-Euclidean Spatial Data. *Mathematics* **2022**, *10*, 1000. [CrossRef]
27. Wahab, O.A.; Mourad, A.; Otrok, H.; Taleb, T. Federated Machine Learning: Survey, Multi-Level Classification, Desirable Criteria and Future Directions in Communication and Networking Systems. *IEEE Commun. Surv. Tutor.* **2021**, *23*, 1342–1397. [CrossRef]
28. Nguyen, D.C.; Pathirana, P.N.; Ding, M.; Seneviratne, A. Blockchain for 5G and beyond networks: A state of the art survey. *J. Netw. Comput. Appl.* **2020**, *166*, 102693. [CrossRef]
29. AbdulRahman, S.; Tout, H.; Ould-Slimane, H.; Mourad, A.; Talhi, C.; Guizani, M. A Survey on Federated Learning: The Journey From Centralized to Distributed On-Site Learning and Beyond. *IEEE Internet Things J.* **2021**, *8*, 5476–5497. [CrossRef]
30. Ma, X.; Liao, L.; Li, Z.; Lai, R.X.; Zhang, M. Applying Federated Learning in Software-Defined Networks: A Survey. *Symmetry* **2022**, *14*, 195. [CrossRef]
31. Abreha, H.G.; Hayajneh, M.; Serhani, M.A. Federated Learning in Edge Computing: A Systematic Survey. *Sensors* **2022**, *22*, 450. [CrossRef] [PubMed]
32. Lalitha, A.; Shekhar, S.; Javidi, T.; Koushanfar, F. Fully decentralized federated learning. In Proceedings of the Third Workshop on Bayesian Deep Learning (NeurIPS), Montreal, QC, Canada, 7 December 2018.
33. Roy, A.G.; Siddiqui, S.; Pölsterl, S.; Navab, N.; Wachinger, C. BrainTorrent: A Peer-to-Peer Environment for Decentralized Federated Learning. *arXiv* **2019**, arXiv:1905.06731.
34. Wilt, M.; Matelsky, J.K.; Gearhart, A.S. Scatterbrained: A flexible and expandable pattern for decentralized machine learning. *arXiv* **2021**, arXiv:2112.07718.
35. Daily, J.; Vishnu, A.; Siegel, C.; Warfel, T.; Amatya, V. GossipGraD: Scalable Deep Learning using Gossip Communication based Asynchronous Gradient Descent. *arXiv* **2018**, arXiv:1803.05880.
36. Hu, C.; Jiang, J.; Wang, Z. Decentralized Federated Learning: A Segmented Gossip Approach. *arXiv* **2019**, arXiv:1908.07782.
37. Jiang, J.; Hu, L.; Hu, C.; Liu, J.; Wang, Z. BACombo—Bandwidth-Aware Decentralized Federated Learning. *Electronics* **2020**, *9*, 440. [CrossRef]
38. Hegedűs, I.; Danner, G.; Jelasity, M. Gossip Learning as a Decentralized Alternative to Federated Learning. In *Distributed Applications and Interoperable Systems*; Pereira, J., Ricci, L., Eds.; Springer International Publishing: Cham, Switzerland, 2019; pp. 74–90.
39. Qu, Y.; Gao, L.; Luan, T.H.; Xiang, Y.; Yu, S.; Li, B.; Zheng, G. Decentralized Privacy Using Blockchain-Enabled Federated Learning in Fog Computing. *IEEE Internet Things J.* **2020**, *7*, 5171–5183. [CrossRef]

40. Li, Z.; Liu, J.; Hao, J.; Wang, H.; Xian, M. CrowdSFL: A Secure Crowd Computing Framework Based on Blockchain and Federated Learning. *Electronics* **2020**, *9*, 773. [CrossRef]
41. Li, Y.; Chen, C.; Liu, N.; Huang, H.; Zheng, Z.; Yan, Q. A Blockchain-Based Decentralized Federated Learning Framework with Committee Consensus. *IEEE Netw.* **2021**, *35*, 234–241. [CrossRef]
42. Qu, Y.; Pokhrel, S.R.; Garg, S.; Gao, L.; Xiang, Y. A Blockchained Federated Learning Framework for Cognitive Computing in Industry 4.0 Networks. *IEEE Trans. Ind. Inform.* **2021**, *17*, 2964–2973. [CrossRef]
43. Nguyen, D.C.; Ding, M.; Pham, Q.V.; Pathirana, P.N.; Le, L.B.; Seneviratne, A.; Li, J.; Niyato, D.; Poor, H.V. Federated Learning Meets Blockchain in Edge Computing: Opportunities and Challenges. *IEEE Internet Things J.* **2021**, *8*, 12806–12825. [CrossRef]
44. Liu, Y.; Qu, Y.; Xu, C.; Hao, Z.; Gu, B. Blockchain-Enabled Asynchronous Federated Learning in Edge Computing. *Sensors* **2021**, *21*, 3335. [CrossRef]
45. Li, J.; Shao, Y.; Wei, K.; Ding, M.; Ma, C.; Shi, L.; Han, Z.; Poor, H.V. Blockchain Assisted Decentralized Federated Learning (BLADE-FL): Performance Analysis and Resource Allocation. *IEEE Trans. Parallel Distrib. Syst.* **2022**, *33*, 2401–2415. [CrossRef]
46. Spanos, D.P.; Olfati-Saber, R.; Murray, R.M. Dynamic consensus on mobile networks. In Proceedings of the IFAC World Congress, Prague, Czech Republic, 3–8 July 2005; pp. 1–6.
47. Freeman, R.A.; Yang, P.; Lynch, K.M. Stability and Convergence Properties of Dynamic Average Consensus Estimators. In Proceedings of the 45th IEEE Conference on Decision and Control, San Diego, CA, USA, 13–15 December 2006; pp. 338–343. [CrossRef]
48. Olfati-Saber, R.; Shamma, J. Consensus Filters for Sensor Networks and Distributed Sensor Fusion. In Proceedings of the 44th IEEE Conference on Decision and Control, Seville, Spain, 15 December 2005; pp. 6698–6703. [CrossRef]
49. Ren, W. Consensus Seeking in Multi-vehicle Systems with a Time-varying Reference State. In Proceedings of the 2007 American Control Conference, New York, NY, USA, 9–13 July 2007; pp. 717–722. [CrossRef]
50. Yu, H.; Zhang, R.; Wu, J.; Li, X. Distributed Field Estimation Using Sensor Networks Based on $H\infty$ Consensus Filtering. *Sensors* **2018**, *18*, 3557. [CrossRef]
51. Liu, H.; Xu, B.; Liu, B. A Tracking Algorithm for Sparse and Dynamic Underwater Sensor Networks. *J. Mar. Sci. Eng.* **2022**, *10*, 337. [CrossRef]
52. Knight, P.A. The Sinkhorn–Knopp Algorithm: Convergence and Applications. *SIAM J. Matrix Anal. Appl.* **2008**, *30*, 261–275. [CrossRef]
53. Lecun, Y.; Bottou, L.; Bengio, Y.; Haffner, P. Gradient-based learning applied to document recognition. *Proc. IEEE* **1998**, *86*, 2278–2324. [CrossRef]
54. Xiao, H.; Rasul, K.; Vollgraf, R. Fashion-MNIST: A Novel Image Dataset for Benchmarking Machine Learning Algorithms. *arXiv* **2017**, arXiv:1708.07747.
55. Krizhevsky, A.; Hinton, G. *Learning Multiple Layers of Features from Tiny Images*; Technical Report; Citeseer: University Park, PA, USA, 2009.