# Decentralized Federated Averaging

Local Training K step then aggregate

Tao Sun, Dongsheng Li, and Bao Wang

**Abstract**—Federated averaging (FedAvg) is a communication-efficient algorithm for distributed training with an enormous number of clients. In FedAvg, clients keep their data locally for privacy protection; a central parameter server is used to communicate between clients. This central server distributes the parameters to each client and collects the updated parameters from clients. FedAvg is mostly studied in centralized fashions, requiring massive communications between the central server and clients, which leads to possible channel blocking. Moreover, attacking the central server can break the whole system's privacy. Indeed, decentralization can significantly reduce the communication of the busiest node (the central one) because all nodes only communicate with their neighbors. To this end, in this paper, we study the decentralized FedAvg with momentum (DFedAvgM), implemented on clients that are connected by an undirected graph. In DFedAvgM, all clients perform stochastic gradient descent with momentum and communicate with their neighbors only. To further reduce the communication cost, we also consider the quantized DFedAvgM. The proposed algorithm involves the mixing matrix, momentum, client training with multiple local iterations, and quantization, introducing extra items in the Lyapunov analysis. Thus, the analysis of this paper is much more challenging than previous decentralized (momentum) SGD or FedAvg. We prove convergence of the (quantized) DFedAvgM under trivial assumptions; the convergence rate can be improved to sublinear when the loss function satisfies the PŁ property. Numerically, we find that the proposed algorithm outperforms FedAvg in both convergence speed and communication cost.

**Index Terms**—Decentralized optimization, federated averaging, momentum, stochastic gradient descent

## 1 Introduction

Federated learning (FL) is a privacy-preserving distributed machine learning (ML) paradigm [1]. In FL, a central server connects with enormous clients (e.g., mobile phones, pads, etc.); the clients keep their data without sharing it with the server. In each communication round, clients receive the current global model from the server, and a small portion of clients are selected to update the global model by running stochastic gradient descent (SGD) [2] for multiple iterations using local data. The central server then aggregates these updated parameters to obtain the updated global model. The above learning algorithm is known as the federated average (FedAvg) [1]. In particular, if the clients are homogeneous, FedAvg is equivalent to the local SGD [3]. FedAvg involves multiple local SGD updates and one aggregation by the server in each communication round, which significantly reduces the communication cost between sever and clients compared to the conventional distributed training with one local SGD update and one communication.

### 1.1 Motivations

In FL applications, large companies and government organizations usually play the role of the central server. On the one hand, since the number of clients in FL is massive, the communication cost between the server and clients–the busiest communication in the centralized system–can be a bottleneck because all clients are connected with the central server [4]. On the other hand, the updated models collected from clients encode the private information of the local data; adversaries can attack the central server to break the privacy of the whole system, which remains the privacy issue as a serious concern. To this end, decentralized federated learning has been proposed [5], [6], where all clients are connected with an undirected graph, a.k.a. overlay network. Decentralized FL (DFL) replaces the server-clients communication in FL with clients-clients communication, or peer-to-peer communication.

Compared with centralized federated learning, decentralized federated learning enjoys several advantages: 1) *DFL significantly reduces the communication costs of the busiest node in FL.* DFL reduces the communication cost of centralized FL's busiest node (central server), since all nodes are connected to the central server in the centralized FL. However, in the decentralized case, all nodes only communicate with their neighbors. One of the simplest decentralized FL cases is using a ring graph to connect all clients, in which each node just connects two topological neighbors. 2) *DFL is more robust to clients' failures than the centralized FL.* Centralized FL will stop if the central server is broken. While decentralized FL can still work even several clients are out of order. Thus, the

- *Tao Sun and Dongsheng Li are with the College of Computer, National University of Defense Technology, Changsha, Hunan 410073, China. E-mail: nudtsuntao@163.com, dsli@nudt.edu.cn.*
- *Bao Wang is with the Scientific Computing & Imaging Institute and Department of Mathematics, University of Utah, Salt Lake City, UT 84112 USA. E-mail: wangbaonj@gmail.com.*

*Manuscript received 17 November 2021; revised 7 July 2022; accepted 2 August 2022. Date of publication 4 August 2022; date of current version 6 March 2023.*
*This work was supported in part by the National Key R&D Program of China under Grant 2021YFB0301200, in part by the Hunan Provincial Natural Science Foundation of China under Grant 2022JJ10065, and in part by the National Science Foundation of China under Grants 62025208 and 61906200. (Corresponding authors: Tao Sun.)*
*Recommended for acceptance by T. Pock.*
*This article has supplementary downloadable material available at https://doi.org/10.1109/TPAMI.2022.3196503, provided by the authors.*
*Digital Object Identifier no. 10.1109/TPAMI.2022.3196503*

TABLE 1
Three Advantages of Decentralized FedAvg Over FedAvg

| | FedAvg | decentralized FedAvg |
|---|---|---|
| 1. Communication costs of the busiest node. | The communication between server and clients is massive. | Each client only communicates with a few neighbors. |
| 2. Robust to clients' failures. | The breaking of the central server will stop the whole FL system. | The system still works even some clients are out of order. |
| 3. Resilient to privacy attacks. | Attacking the central server can destroy the system's privacy. | No node contains the whole system's information. |

decentralized one is more robust to potential clients' failures. 3) *DFL is more resilient to potential privacy attacks than FL.* Privacy is another primary concern of federated learning since the central server is also exposed to adversarial attacks. Notice that the central server contains all clients' information in FedAvg. If someone successfully attacks the centralized server, all information may be divulged. While in the decentralized case, all clients only communicate with their neighbors. Only part of the information will be leaked if some clients are attacked. As confirmed numerically in our paper, DFL is more robust to potential privacy attacks, e.g., membership inference attacks.

In this paper, we consider two crucial issues about decentralized FL: 1) Although there is no expensive communication between server and clients in decentralized FL, the communication between local clients can be costly when the size of ML model is large. Therefore, it is crucial to ask *can we reduce the client-client communication cost in DFL systems?* 2) Momentum is a well-established acceleration technique for SGD [7]. It is natural to ask *can we use SGD with momentum to improve the training of ML models in decentralized FL with theoretical convergence guarantees?*

### 1.2 More Related Works and the Novelty

We briefly review three lines of work that are most related to this paper, i.e., federated learning, decentralized training, and decentralized federated learning.

*Federated Learning*. Many variants of FedAvg have been developed with theoretical guarantees. In [8], the authors use the momentum method for local clients training in FedAvg. The authors of [9] propose the adaptive FedAvg, whose central parameter server uses the adaptive learning rate to aggregate local models. Lazy and quantized gradients are used to reduce communications [10], [11]. In the paper [12], the authors propose a Newton-type scheme for federated learning. The federated learning method has been applied to Internet of Things (IoT) researches [13]. The convergence analysis of FedAvg on heterogeneous data is discussed in [14], [15], [16]. More details and applications of federated learning could be found in [17], [18]. Recent advances and some open problems in FL is available in survey papers [19], [20].

*Decentralized Training*. Decentralized algorithms were originally developed to calculate the mean of data that are stored over multiple sensors [21], [22], [23], [24]. Decentralized (sub) gradient descents (DGD), one of the simplest and efficient decentralized algorithms, have been studied in [25], [26], [27], [28], [29]. In DGD, the convexity assumption is unnecessary [30], which makes DGD useful for nonconvex optimization. A provably convergent decentralized SGD (DSGD) is proposed in [4], [31], [32]. The paper [31] provides the complexity result of a decentralized stochastic algorithm. In [32], the authors design a decentralized stochastic algorithm with dual information and provides the theoretical convergence guarantee. The authors of [4] prove that DSGD outperforms SGD in communication efficiency. Asynchronous DSGD is analyzed in [33]. DGD with momentum is proposed in [34], [35]. Quantized DSGD has been proposed in [36].

*Decentralized Federated Learning*. Decentralized FL is a learning paradigm of choice when the edge devices do not trust the central server in protecting their privacy [18]. The authors in [37] propose a novel FL framework without a central server for medical applications, and the new method offers a highly dynamic peer-to-peer environment. The papers [5], [6] consider training an ML model with a connected overlay network whose nodes take a Bayesian-like approach by introducing a prior of the parameter space.

Compared with existing works on FedAvg [1], [8], [9], [10], [11], this paper uses a decentralized framework to enhance the robustness of FL to node failures and privacy attacks. In contrast to decentralized training [5], [6], [37], all nodes perform multiple local iterations rather than only one and employ momentum in our algorithm. To further reduce the communication costs, we use the quantization technique. These new algorithms are much more complicated than FedAvg or DSGD, and their convergence analysis is significantly more challenging than analyzing FedAvg and DSGD. We present the detailed convergence results of the proposed algorithms in convex, nonconvex, and PŁ conditions. From a practical viewpoint, decentralized FL enjoys communication efficiency and fast convergence; we summarize the advantages of decentralized FL over FL in Table 1. Moreover, we present a sufficient condition that reveals when quantization enjoys communication efficiency and convergence tradeoffs.

### 1.3 Our Contributions

We propose decentralized FedAvg with momentum (DFedAvgM) to improve training machine learning models in a DFL fashion. To further reduce the communication cost between clients, we also integrate model quantization, i.e., quantize the local machine learning models before communication, with DFedAvgM. Our contributions in this paper are elaborated below in threefold.

- Algorithmically, we extend FedAvg to the decentralized setting, where all clients are connected by an undirected graph. We motivate DFedAvgM from the DSGD algorithm. In particular, we use SGD with momentum to train ML models on each client. To

reduce the communication cost between each client, we further introduce a quantized version of DFedAvgM, in which each client will send and receive a quantized model.
- Theoretically, we prove the convergence of (quantized) DFedAvgM. Our theoretical results show that the convergence rate of (quantized) DFedAvgM is not inferior to that of SGD or DSGD. More specifically, we show that the convergence rates of both DFedAvgM and quantized DFedAvgM depend on the local training and the graph that connects all clients. Besides the convergence results under nonconvex assumptions, we also establish their convergence guarantee under the Polyak-Łojasiewicz (PŁ) condition, which has been widely studied in nonconvex optimization. Under the PŁ condition, we establish a faster convergence rate for (quantized) DFedAvgM. Furthermore, we present a sufficient condition to guarantee reducing communication costs.
- Empirically, we perform extensive numerical experiments on training deep neural networks (DNNs) on various datasets in both **IID** and **Non-IID** settings. Our results show the effectiveness of (quantized) DFedAvgM for training ML models, saving communication costs, and protecting membership privacy of training data.

### 1.4 Organizations

We organize this paper as follows: in Section 2, we present a mathematical formulation of our problem and some necessary assumptions. In Section 3, we present the DFedAvgM and its quantized version. We present the convergence of the proposed algorithm in Section 4. We provide extensive numerical verification of DFedAvgM in Section 5. This paper ends up with concluding remarks.

### 1.5 Notation

We denote scalars and vectors by lower case and lower case boldface letters, respectively, and matrices by upper case boldface letters. For a vector $\mathbf{x} = (x_1, \ldots, x_d) \in \mathbb{R}^d$, we denote its $\ell_p$ norm ($p \geq 1$) by $\|\mathbf{x}\|_p = (\sum_{i=1}^d |x_i|^p)^{1/p}$, and denote the $\ell_\infty$ norm of $\mathbf{x}$ by $\|\mathbf{x}\|_\infty = \max_{i=1}^d |x_i|$ and denote its $\ell_2$ norm as $\|\mathbf{x}\|$. For a matrix $\mathbf{A}$, we denote its transpose as $\mathbf{A}^\top$. Given two sequences $\{a_n\}$ and $\{b_n\}$, we write $a_n = \mathcal{O}(b_n)$ if there exists a positive constant $0 < C < +\infty$ such that $a_n \leq Cb_n$, and we write $a_n = \mathbf{\Theta}(b_n)$ if there exist two positive constants $C_1$ and $C_2$ such that $a_n \leq C_1 b_n$ and $b_n \leq C_2 a_n$. $\tilde{\mathcal{O}}(a_n)$ hides the logarithmic factor of $a_n$. For a function $f(\mathbf{x}) : \mathbb{R}^d \to \mathbb{R}$, we denote its gradient as $\nabla f(\mathbf{x})$ and its Hessian as $\nabla^2 f(\mathbf{x})$, and denote its minimum as $\min f$. We use $\mathbb{E}[\cdot]$ to denote the expectation with respect to the underlying probability space.

## 2 Problem Formulation and Assumptions

We consider the following optimization task

$$\min_{\mathbf{x}\in\mathbb{R}^d} f(\mathbf{x}) := \frac{1}{m}\sum_{i=1}^m f_i(\mathbf{x}), \quad f_i(\mathbf{x}) = \mathbb{E}_{\xi\sim\mathcal{D}_i} F_i(\mathbf{x};\xi), \tag{1}$$

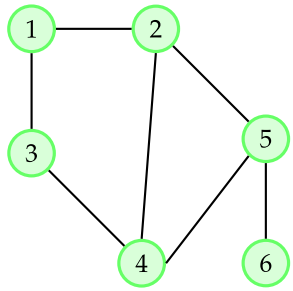



Fig. 1. An example of an undirected graph with six nodes. Each node owns its data, executes computations locally, and only communicates with its neighbors.

where $\mathcal{D}_i$ denotes the data distribution of the $i$th client, and $F_i(\mathbf{x};\xi)$ is the loss function associated with the training data $\xi$. Problem (1) models many machine learning applications, which is known as empirical risk minimization (ERM). This paper is devoted to studying decentralized federated learning, a particular class of distributed machine learning paradigms.

The structure of the decentralized system is that an undirected graph connects all clients in the system. We stress that there is no central server in the decentralized setting. Fig. 1 depicts an example of a decentralized system, where six nodes are connected by an undirected graph. This paper considers decentralized FL, which is fully distributed because data are stored in each client (node). In the iterations, all clients only communicate with their neighbors.

An important notion in decentralized optimization is the *mixing matrix*, which is usually associated with a connected graph $\mathcal{G} = (\mathcal{V}, \mathcal{E})$ with the vertex set $\mathcal{V} = \{1, \ldots, m\}$ and the edge set $\mathcal{E} \subseteq \mathcal{V} \times \mathcal{V}$. Any edge $(i, l) \in \mathcal{E}$ represents a communication channel between nodes $i$ and $l$. The set of neighbors of node $i$ is denoted by $\mathcal{N}(i) := \{j \in \mathcal{V} \,|\, (i, j) \in \mathcal{E}\}$. We recall the definition of the mixing matrix associated with the graph $\mathcal{G}$.

**Definition 1 (Mixing Matrix).** *The mixing matrix* $\mathbf{W} = [w_{i,j}] \in \mathbb{R}^{m\times m}$ *is assumed to have the following properties: 1. (Graph) If* $i \neq j$ *and* $(i, j) \notin \mathcal{E}$, *then* $w_{i,j} = 0$, *otherwise,* $w_{i,j} > 0$; *2. (Symmetry)* $\mathbf{W} = \mathbf{W}^\top$; *3. (Null space property)* $\text{null}\{\mathbf{I} - \mathbf{W}\} = \text{span}\,\mathbf{1}$; *4. (Spectral property)* $\mathbf{I} \succeq \mathbf{W} \succ -\mathbf{I}$.

For a graph, the corresponding mixing matrix is not unique; given the adjacency matrix of a graph, its maximum-degree matrix and metropolis-hastings matrix [38] are both mixing matrices. The symmetric property of $\mathbf{W}$ indicates that its eigenvalues are all real and can be sorted in the non-increasing order. Let $\lambda_i(\mathbf{W})$ denote the $i$th largest eigenvalue of $\mathbf{W}$, that is, $\lambda_1(\mathbf{W}) = 1 > \lambda_2(\mathbf{W}) \geq \cdots \geq \lambda_m(\mathbf{W}) > -1$.[1] The mixing matrix also serves as a probability transition matrix of a Markov chain. A quite important constant of $\mathbf{W}$ is $\lambda = \lambda(\mathbf{W}) := \max\{|\lambda_2(\mathbf{W})|, |\lambda_m(\mathbf{W})|\}$, which describes the speed of the Markov chain introduced by the mixing matrix converges to the stable state.

*Assumptions*: In the rest of this section, we list several assumptions for the subsequent analysis.

1. This is based on the spectral property of mixing matrix.

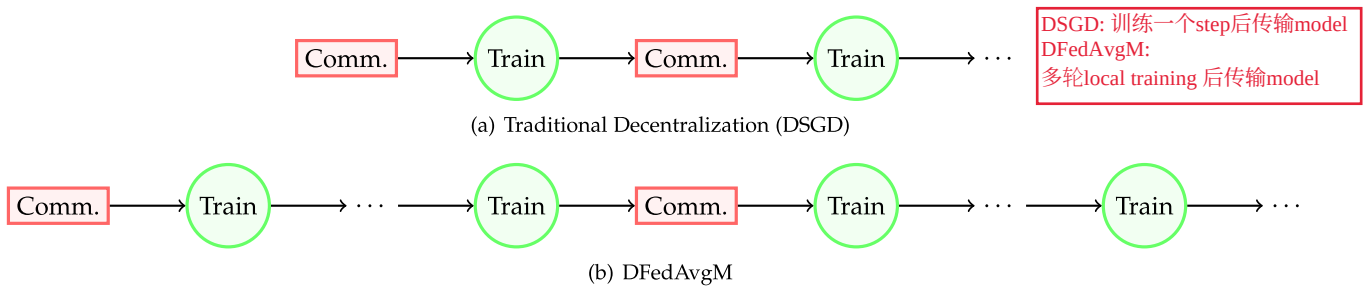



Fig. 2. Comparison of communication and training styles of traditional decentralized stochastic gradient descent (DSGD) and the proposed decentralized federated average with momentum (DFedAvgM). In DSGD, each client will communicate with its neighbors after one single training step. In DFedAvgM, however, each client will communicate with its neighbors after multiple training iterations.

- **Assumption 1.** *The function $f_i$ is differentiable and $\nabla f_i$ is $L$-Lipschitz continuous, $\forall i \in \{1, 2, \ldots, m\}$, i.e., $\|\nabla f_i(\mathbf{x}) - \nabla f_i(\mathbf{y})\| \leq L\|\mathbf{x} - \mathbf{y}\|$, for all $\mathbf{x}, \mathbf{y} \in \mathbb{R}^d$.*

Here, for simplicity, we suppose all functions have the same Lipschitz constant $L$. We can also assume that these functions have different Lipschitz constants, which does not affect the convergent analysis. The first-order Lipschitz assumption is commonly used in the ML community and can be satisfied by various ML tasks.

- **Assumption 2.** *The gradient of the function $f_i$ have $\sigma_l$-bounded variance, i.e., $\mathbb{E}[\|\nabla F_i(\mathbf{x}; \xi) - \nabla f_i(\mathbf{x})\|^2] \leq \sigma_l^2$ for all $\mathbf{x} \in \mathbb{R}^d$ and for $\forall i \in \{1, 2, \ldots, m\}$. We also assume $\frac{1}{m}\sum_{i=1}^m \|\nabla f_i(\mathbf{x}) - \nabla f(\mathbf{x})\|^2 \leq \sigma_g^2$ for all $\mathbf{x} \in \mathbb{R}^d$.*

The uniform local variance assumption is also used for the ease of presentation, which is straightforward to generalize to non-uniform cases. The global variance assumption has been used substantially, see, e.g., [9], [39]. The constant $\sigma_g$ reflects the heterogeneity of the data sets $(\mathcal{D}_i)_{1\leq i\leq m}$, and $\sigma_g = 0$ when $(\mathcal{D}_i)_{1\leq i\leq m}$ follow the same distribution. The bounded variance assumption is widely used in stochastic optimization and FL, which is quite mild.

- **Assumption 3.** *([30]) For any $i \in \{1, 2, \ldots, m\}$ and $\mathbf{x} \in \mathbb{R}^d$, we have $\|\nabla f_i(\mathbf{x})\| \leq B$ for some $B > 0$.*

The bounded gradient assumption is widely used for the nonconvex decentralized gradient optimization. This assumption holds for many neural network training tasks due to the cross entropy loss function. Consider the binary classification using logistic function, the loss is then $F(\mathbf{x}; \xi) := -\mathbf{b}_\xi \log\left(\frac{1}{1+e^{-\langle \xi, \mathbf{x}\rangle}}\right)$, where $\mathbf{b}_\xi \in \{-1, 1\}$ is the label of $\xi$. Then, we have $\nabla F(\mathbf{x}; \xi) = -\frac{e^{-\langle \xi, \mathbf{x}\rangle}}{1+e^{-\langle \xi, \mathbf{x}\rangle}} \mathbf{b}_\xi \xi$. Therefore, $\|\mathbb{E}\nabla f(\mathbf{x})\| \leq \mathbb{E}\|\nabla F(\mathbf{x}; \xi)\| \leq \mathbb{E}_{\xi\sim\mathcal{D}}\|\xi\|$.

- **Assumption 4.** *([40], [41], [42]) For a smooth function $f$, we say it satisfies PŁ-$\nu$ property provided*

$$\|\nabla f(\mathbf{x})\|^2 \geq 2\nu(f(\mathbf{x}) - \min f), \quad \forall \mathbf{x} \in \mathrm{dom}(f). \tag{2}$$

Recent nonconvex analysis [40], [41], [42] have analyzed the algorithmic performance under the PŁ property, which is named after Polyak and Łojasiewicz [43], [44]. Strong convexity implies PŁ condition, but not vice verse. Several examples satisfy Assumption 4 include the loss of training one-hidden-layer networks [45], the loss of training ResNets with linear activation [46], and objective function in matrix factorization [42]. However, the PŁ condition cannot be satisfied by the loss of training general networks, which means the convergence result under the PŁ condition does not apply to a broad of learning tasks.

## 3 Decentralized Federated Averaging

### 3.1 Decentralized FedAvg With Momentum

We first briefly review the previous work on decentralized training, which executes in the following steps:

1) client $i$ holds an approximate copy of the parameters $\mathbf{x}(i) \in \mathbb{R}^d$ and calculates an unbiased estimate of $\nabla f_i := \mathbf{g}(i)$ at $\mathbf{x}(i)$. $(\mathbf{x}(i))_{1\leq i\leq m}$ can be non-consensus;
2) (*communication*) client $i$ updates its local parameters $\mathbf{x}(i)$ as the weighted average of its neighbors: $\tilde{\mathbf{x}}(i) = \sum_{l\in\mathcal{N}(i)} w_{i,l}\mathbf{x}(l)$;
3) (*training*) client $i$ updates its parameters as $\mathbf{x}(i) \leftarrow \tilde{\mathbf{x}}(i) - \eta\mathbf{g}(i)$ with a learning rate $\eta > 0$.

**Algorithm 1.** DFedAvgM

**Parameters**: $\eta > 0, K \in \mathbb{Z}^+, 0 \leq \theta < 1$.
**Initialization**: $\mathbf{x}^0 = \mathbf{0}$
**for** $t = 1, 2, \ldots$ **do**
  **for** $i = \{1, 2, \ldots, m\}$ **do**
    node $i$ performs local training (5) $K$ times and sends $\mathbf{z}^t(i) = \mathbf{y}^{t,K}(i)$ to $\mathcal{N}(i)$
  node $i$ updates as (6)
  **end for**
**end for**

The traditional decentralized training algorithm can be described in Fig. 2a, in which one *communication* step is required after each local *training* iteration. This indicates that the above vanilla decentralized training algorithm is different from FedAvg, and note the later performs multiple local *training* steps before each *communication*. To this end, we have to slightly modify the scheme of the decentralized algorithm. For simplicity, we consider modifying DSGD to motivate our decentralized FedAvg algorithm. Note that when the original DSGD is applied to solve (1), we end up with the following iteration

$$\mathbf{x}^{t+1}(i) = \sum_{l\in\mathcal{N}(i)} w_{i,l}\mathbf{x}^t(l) - \gamma\mathbf{g}^t(i) = \sum_{l\in\mathcal{N}(i)} w_{i,l}[\mathbf{x}^t(l) - \gamma\mathbf{g}^t(i)], \tag{3}$$

where we used the fact that $\sum_{l\in\mathcal{N}(i)} w_{i,l} = 1$. In (3), if we replace $\mathbf{x}^t(l)$ by $\mathbf{x}^t(i)$, the algorithm then iterates as

$$\mathbf{x}^{t+1}(i) = \sum_{l\in\mathcal{N}(i)} w_{i,l}[\mathbf{x}^t(i) - \gamma\mathbf{g}^t(i)]. \tag{4}$$

In (4), clients communicate with their neighbors after one local training iteration, which is then possible to be generalized to the federated optimization setting. We replace the single SGD iteration in (4) with multiple iterations by using SGD with heavy-ball momentum [47]. Therefore, the DFedAvgM can be formulated as follows: In each $t \in \mathbb{Z}^+$, for each client $i \in \{1, 2, \ldots, m\}$, let $\mathbf{y}^{t,-1}(i) = \mathbf{y}^{t,0}(i) = \mathbf{x}^t(i)$. The inner iteration in each node then performs as

$$\mathbf{y}^{t,k+1}(i) = \mathbf{y}^{t,k}(i) - \eta\tilde{\mathbf{g}}^{t,k}(i) + \theta(\mathbf{y}^{t,k}(i) - \mathbf{y}^{t,k-1}(i)), \tag{5}$$

where $\mathbb{E}\tilde{\mathbf{g}}^{t,k}(i) = \nabla f_i(\mathbf{y}^{t,k}(i))$. After $K$ inner iterations in each local client, the resulting parameters $\mathbf{z}^t(i) \leftarrow \mathbf{y}^{t,K}(i)$ is sent to its neighbors $\mathcal{N}(i)$. Every client then updates its parameters by taking the local average as follows

$$\mathbf{x}^{t+1}(i) = \sum_{l\in\mathcal{N}(i)} w_{i,l}\mathbf{z}^t(l). \tag{6}$$

The procedure of DFedAvgM can be illustrated as in Fig. 2b. It is seen that DFedAvgM balances the tradeoff between local computing and communications. It is well-known that the communication costs are usually much more expensive than the computation costs [48], which indicates DFedAvgM can be more efficient than DSGD.

**Algorithm 2.** Quantized DFedAvgM

1: **parameters**: $\eta > 0$, $K \in \mathbb{Z}^+$, $0 \le \theta < 1$, $s$, $b$.
2: **initialization**: $\mathbf{x}^0 = \mathbf{0}$
3: **for** $t = 1, 2, \ldots$ **do**
4: **for** $i = \{1, 2, \ldots, m\}$ **do**
5: node $i$ performs local training (5) $K$ times and sends $\mathbf{q}^t(i) = Q[\mathbf{y}^{t,K}(i) - \mathbf{x}^t(i)]$ to $\mathcal{N}(i)$
6: node $i$ updates as (8)
7: **end for**
8: **end for**

### 3.2 Efficient Communication via Quantization

In DFedAvgM, client $i$ needs to send $\mathbf{x}^t(i)$ to its neighbors $\mathcal{N}(i)$. Thus, when the number of neighbours $\mathcal{N}(i)$ grows, client-client communications become the major bottleneck on the efficiency of the algorithm. We leverage the quantization trick [49], [50] to reduce the communication cost. In particular, we consider the following quantization procedure: Given a constant $s > 0$ and the limited bit number $b \in \mathbb{Z}^+$, the representable range is then $\{-2^{b-1}s, -(2^{b-1}-1)s, \ldots, 0, s, 2s, \ldots, (2^{b-1}-1)s\}$. For any $a \in \mathbb{R}$ with $-2^{b-1}s \le a < (2^{b-1}-1)s$, we can find an integer $k \in \mathbb{Z}$ such that $ks \le a < (k+1)s$ and we then use $ks$ to replace $a$. The above quantization scheme is deterministic, which can be written as $q(a) := \lfloor \frac{a}{s} \rfloor s$ for $a \in \mathbb{R}$; Besides the deterministic rule, the stochastic quantization uses the following scheme

$$q(a) := \begin{cases} ks, \text{ w.p. } 1 - \frac{a-ks}{s}, \\ (k+1)s, \text{ w.p. } \frac{a-ks}{s}. \end{cases}$$

It is easy to see that the stochastic quantization is unbiased, i.e., $\mathbb{E}[q(a)] = a$ for any $a \in \mathbb{R}$. When $s$ is small, deterministic and stochastic quantization schemes perform very similarly. For a vector $\mathbf{x} \in \mathbb{R}^d$ whose coordinates are all stored with 32 bits, we consider quantizing all coordinates of $\mathbf{x} = [x_1, x_2, \ldots, x_d] \in \mathbb{R}^d$. The multi-dimension quantization operator is then defined as

$$Q(\mathbf{x}) := [q(x_1), q(x_2), \ldots, q(x_d)]. \tag{7}$$

For both deterministic and stochastic quantization schemes, we have $\mathbb{E}\|Q(\mathbf{x}) - \mathbf{x}\|^2 \le \frac{d}{4}s^2$ if $x_i \in [-2^{b-1}s, (2^{b-1}-1)s]$ for $i \in \{1, 2, \ldots, d\}$. The two quantization schemes presented satisfy $\mathbb{E}\|Q(\mathbf{x}) - \mathbf{x}\|^2 \le \frac{d}{4}s^2$ with $s > 0$ for any $\mathbf{x} \in \mathbb{R}^d$.

Directly quantizing the parameters is feasible for sufficiently smooth loss functions, but maybe impossible for DNNs. To this end, we consider quantizing the difference of parameters. Quantized DFedAvgM can be summarized as follows: (1) After running (5) for $K$ times, client $i$ quantizes $\mathbf{q}^t(i) \leftarrow Q(\mathbf{y}^{t,K}(i) - \mathbf{x}^t(i))$ and send it to $\mathcal{N}(i)$. (2) After receiving $[\mathbf{q}^t(j)]_{j\in\mathcal{N}(i)}$, every client updates its local parameters as

$$\mathbf{x}^{t+1}(i) = \mathbf{x}^t(i) + \sum_{l\in\mathcal{N}(i)} w_{i,l}\mathbf{q}^t(l). \tag{8}$$

In each communication round, client $i$ just needs to send the pair $(s, \mathbf{q}^t(i))$ to $\mathcal{N}(i)$, whose representation requires $(32 + db)\deg(\mathcal{N}(i))$ bits rather than $32d\deg(\mathcal{N}(i))$ bits for sending the unquantized version. If $d$ is large and $b < 32$, the communications can be significantly reduced.

## 4 Convergence Analysis

In this section, we analyze the convergence of the proposed (quantized) DFedAvgM. The convergence analysis of DFedAvgM is much more complicated than SGD, SGD with momentum, and DSGD; the technical difficulty is because $\mathbf{z}^t(i) - \mathbf{x}^t(i)$ fails to be an unbiased estimate of the gradient $\nabla f_i(\mathbf{x}^t(i))$ after multiple iterations of SGD or SGD with momentum in each client. In the previous stochastic decentralized training, all nodes only perform one step of local training and the local gradients are all unbiased. The global scheme then can be formulated as the following iteration

$$\mathbf{X}^{t+1} = \mathbf{W}\mathbf{X}^t - \eta\mathbf{G}^t, \tag{9}$$

where $\mathbb{E}\mathbf{G}^t = [\nabla f_1(\mathbf{x}^t(1)), \ldots, \nabla f_m(\mathbf{x}^t(m))]^\top$ and $\mathbf{X}^t := [\mathbf{x}^t(1), \ldots, \mathbf{x}^t(m)]^\top$. The iteration (9) can be regarded as an iteration under a stochastic operator. However, in our algorithm,

$$\mathbf{X}^{t+1} = \mathbf{W}\mathbf{Z}^t, \tag{10}$$

where $\mathbf{Z}^t := [\mathbf{z}^t(1), \ldots, \mathbf{z}^t(m)]^\top$, and $\mathbf{z}^t(i)$ is the $K$th iteration generated by momentum SGD from $\mathbf{x}^t(i)$. The relation between $\mathbf{z}^t(i)$ and $\mathbf{x}^t(i)$ is implicit, and thus iteration (10) cannot be presented as an iteration under some stochastic

operator. Even in the momentum free and one iteration case ($K = 1$), our algorithm is $\mathbf{X}^{t+1} = \mathbf{W}\mathbf{X}^t - \eta\mathbf{W}\mathbf{G}^t$, which does not reduce to (9).

The aggregation in our algorithm indeed differs from the one in the centralized FL settings. We explain this two aspects: 1. In the centralized FL settings, only the central parameter sever needs to aggregate all information from all nodes. Thus, in each iteration, centralized FL only do once aggregation. However, in our algorithm, all nodes need to aggregate information from its neighbors. In each iteration of our algorithm, the nodes need to do N times aggregations (with only neighbors). 2. From another perspective, consider the multiple iterations reduce to once iteration, centralized FL then reduces to the mini-batch SGD, but our algorithm reduces to a decentralized-style SGD (not decentralized SGD). Even the decentralized SGD cannot follow the analysis of SGD, let alone with a new decentralized scheme.

Our main technical contribution is integrating the multiple local momentum training with decentralized optimization. In each node, we need to investigate the sequence after local multiple iterations, which is similar to the centralized FL. But how to merging the multiple local iterations into decentralized training is non-trivial (note that our algorithm is not exactly the decentralized SGD even with one iteration).

In the following, we consider the convergence of the average point, which is defined as $\overline{\mathbf{x}^t} := \sum_{i=1}^m \mathbf{x}^t(i)/m$.

In the following, we present the convergence of DFedAvgM under the PŁ condition. We first present the convergence of DFedAvgM for PŁ objective function in the following Theorem.

**Theorem 1 (PŁ Condition).** *Assume Assumption 1, 2, 3 and 4 hold, then the following convergence rate holds*

$$\mathbb{E}f(\overline{\mathbf{x}^T}) - \min f \leq [1 - \nu\gamma(K,\eta)]^T (f(\overline{\mathbf{x}^0}) - \min f) + \frac{\alpha(K,\eta)}{2\nu} + \frac{\beta(K,\eta,\lambda)}{2\nu}, \tag{11}$$

*where the constants are given as*

$$\gamma(K,\eta) := \frac{\eta(K - \theta\frac{1-\theta^K}{1-\theta})}{(1-\theta)} - \frac{64L^2K^4\eta^3}{K - \theta\frac{1-\theta^K}{1-\theta}} - 64LK^2\eta^2,$$

$$\alpha(K,\eta) := \frac{(\frac{L^2K^2\eta^3}{(K-\theta\frac{1-\theta^K}{1-\theta})} + L\eta^2)(8K\sigma_l^2 + 32K^2\sigma_g^2 + \frac{64K^2\theta^2(\sigma_l^2+B^2)}{(1-\theta)^2})}{\frac{\eta(K-\theta\frac{1-\theta^K}{1-\theta})}{(1-\theta)} - \frac{64L^2K^4\eta^3}{K-\theta\frac{1-\theta^K}{1-\theta}} - 64LK^2\eta^2}$$

*and*

$$\beta(K,\eta,\lambda) := (\frac{64L^4K^4\eta^5}{(K-\theta\frac{1-\theta^K}{1-\theta})} + 64L^3K^2\eta^4) \times \frac{(8K\sigma_l^2 + 32K^2\sigma_g^2 + 32K^2B^2 + \frac{64K^2\theta^2}{(1-\theta)^2}(\sigma_l^2+B^2))}{[(1-\lambda)^2(\frac{\eta(K-\theta\frac{1-\theta^K}{1-\theta})}{(1-\theta)} - \frac{64L^2K^4\eta^3}{K-\theta\frac{1-\theta^K}{1-\theta}} - 64LK^2\eta^2)]},$$

*and $T$ is the total number of communication rounds.*

If the learning rate enjoys the form as $\eta = \ln^{c_1} T/(LKT^{c_2})$ with $c_1, c_2 > 0$ [2], we can prove the following results on the optimal choices for $c_1, c_2$.

**Proposition 1.** *Let $\eta = c_3\ln^{c_1} T/(LKT^{c_2})$ with $c_1, c_2 > 0$, the optimal rate of DFedAvgM is $\tilde{\mathcal{O}}(1/T)$, in which case $c_1 = c_2 = 1$ and $c_3 = L/\nu$, that is, $\eta = \ln T/(\nu KT)$. As such, we have the following convergence rate for DFedAvgM*

$$\mathbb{E}(f(\overline{\mathbf{x}^T}) - \min f) = \tilde{\mathcal{O}}\Bigg(\frac{(1-\theta)(f(\overline{\mathbf{x}^1}) - \min f)}{T} + \frac{(1-\theta)\sigma_l^2 + (1-\theta)K\sigma_g^2 + \frac{\theta^2}{(1-\theta)}K(\sigma_l^2+B^2)}{KT} + \frac{(1-\theta)(\sigma_l^2 + K\sigma_g^2 + KB^2) + \frac{\theta^2}{(1-\theta)}K(\sigma_l^2+B^2)}{(1-\lambda)^2KT^3}\Bigg).$$

The convergence rate in Proposition 1 coincides with existing results of the optimal rate of SGD with respect to $T$ under the strong convexity assumption [51], [52]. To reach an $\epsilon > 0$ error, DFedAvgM needs $\widetilde{\mathcal{O}}(\max\{\frac{1}{\epsilon}, \frac{1}{\epsilon^{1/3}(1-\lambda)^{2/3}}\})$ communication rounds under the PŁ condition. There are also many proved convergence rates for the FedAvg and its variants in the strongly convex case. In [14], [15], the authors prove that FedAvg needs $\widetilde{\mathcal{O}}(\frac{1}{\epsilon})$ communication rounds to output the same level error. In [39], the authors generalize FedAvg and propose FedProx with the $\widetilde{\mathcal{O}}(\frac{1}{\epsilon})$ rate being proved. In [53], by using control variates in local updates, the authors propose SCAFFOLD and prove $\widetilde{\mathcal{O}}(\frac{1}{\epsilon})$ rate. If $\lambda \leq 1 - \epsilon$ (this setting is easily satisfied because $\epsilon$ is small), our results match the common rate of FedAvg and its variants with strongly convex assumption on the objectives.

The results in Proposition 1 also show that the variance affects the convergence, a larger variance leads to a worse rate, unsurprisingly. We can see that the graph structure also impacts the convergence rate. The ideal case is that $\lambda = 0$, which is yet usually unachievable because the structure of graph is fixed. Luckily, the mixing matrix is non-unique, and we can choose the one whose $\lambda$ is the smallest; it has been shown that searching such a mixing matrix can be relaxed as a constrained convex quadratic optimization problem [38]. As $K$ increases, the error bound goes down, which is reasonable. When $\theta = 0$, we consider an extreme case, in which $K$ is infinity; the rate then becomes $\mathcal{O}(\frac{1}{T} + \frac{\sigma_g^2}{T} + \frac{\sigma_g^2+B^2}{(1-\lambda)^2T^3})$. We can see the local variance then vanishes and contributes nothing. This phenomenon coincides with our intuition: in local client, the use of large $K$ can result in a local minimizer; then the local variance bound will hurt nothing. It is worth mentioning that the theoretical results in Proposition 1 (as $K$ is large enough) show that whether the momentum $\theta$ can accelerate the algorithm depends on the relation between $f(\overline{\mathbf{x}^1}) - \min f + \sigma_g^2$ and $\sigma_l^2 + B^2$. In particular, if $f(\overline{\mathbf{x}^1}) - \min f + \sigma_g^2 \gg \sigma_l^2 + B^2$, as $\theta \in [0,1)$ increases, and the rate improves; if $f(\overline{\mathbf{x}^1}) - \min f + \sigma_g^2 \ll \sigma_l^2 + B^2$, large $\theta$ may degrade the performance of DFedAvgM.

2. This learning rate is commonly used in the ML community.

In many cases, objective functions are simply assumed to be smooth assumption. In the following, we present the convergence under general nonconvex case.

**Theorem 2 (General Nonconvexity).** *Let the sequence $\{\mathbf{x}^t(i)\}_{t\geq 0}$ be generated by DFedAvgM for $i \in \{1, 2 \ldots, m\}$ and suppose Assumptions 1, 2 and 3 hold. Moreover, assume the stepsize $\eta$ for SGD with momentum that used for training client models satisfies*

$$0 < \eta \leq \frac{1}{8LK} \quad \text{and} \quad 64L^2K^2\eta^2 + 64LK\eta < 1,$$

*where $L$ is the Lipschitz constant of $\nabla f$ and $K$ is the number of client iterations before each communication. Then*

$$\min_{1\leq t\leq T} \mathbb{E}\|\nabla f(\overline{\mathbf{x}^t})\|^2 \leq \frac{2f(\overline{\mathbf{x}^1}) - 2\min f}{\gamma(K,\eta)T} + \alpha(K,\eta) + \beta(K,\eta,\lambda).$$

To get an explicit rate on $T$ from Theorem 2, we choose $\eta = \Theta(1/LK\sqrt{T})$. As $T$ is large enough and $64L^2K^2\eta^2 + 64LK\eta < 1$. Then

$$\gamma(K,\eta) = \Theta(1/((1-\theta)\sqrt{T})),$$

and

$$\alpha(K,\eta) = \Theta\left(\frac{(1-\theta)\sigma_l^2 + (1-\theta)K\sigma_g^2 + \frac{\theta^2}{(1-\theta)}K(\sigma_l^2+B^2)}{K\sqrt{T}}\right),$$

and

$$\beta(K,\eta,\lambda) = \Theta\left(\frac{(1-\theta)(\sigma_l^2 + K\sigma_g^2 + KB^2) + \frac{\theta^2}{(1-\theta)}K(\sigma_l^2+B^2)}{(1-\lambda)^2KT^{3/2}}\right).$$

Based on this choice of $\eta$ and the Theorem 2, we have the following convergence rate for DFedAvgM.

**Proposition 2.** *Let $\eta = \Theta(\frac{1}{LK\sqrt{T}})$, as the communication round number $T$ is large enough, it holds that*

$$\min_{1\leq t\leq T} \mathbb{E}\|\nabla f(\overline{\mathbf{x}^t})\|^2 = \mathcal{O}\Bigg(\frac{(1-\theta)(f(\overline{\mathbf{x}^1}) - \min f)}{\sqrt{T}} + \frac{(1-\theta)\sigma_l^2 + (1-\theta)K\sigma_g^2 + \frac{\theta^2}{(1-\theta)}K(\sigma_l^2+B^2)}{K\sqrt{T}} + \frac{(1-\theta)(\sigma_l^2 + K\sigma_g^2 + KB^2) + \frac{\theta^2}{(1-\theta)}K(\sigma_l^2+B^2)}{(1-\lambda)^2KT^{3/2}}\Bigg).$$

The general nonconvex DFedAvgM can use a much larger stepsize than the one used under the PŁ condition. Note that SGD also uses a smaller stepsize in the PŁ case than that used in the general nonconvex case [51], [52]. Recall the convergence of DFedAvgM in the PŁ case, we must choose a smaller $\eta$; otherwise, the term $[1 - \nu\gamma(K,\eta)]^T(f(\overline{\mathbf{x}^0}) - \min f)$ will dominate the right-hand side of (11) and decays very slowly, which then yields a deceleration. Similar to Proposition 1, from Proposition 2, we can see that the speed of DFedAvgM can be improved when the number of local iteration, $K$, increases. Also, when the momentum $\theta$ is 0 and $K$ is large enough, the bound will be dominated by $\mathcal{O}(\frac{1}{\sqrt{T}} + \frac{\sigma_g^2}{\sqrt{T}} + \frac{\sigma_g^2+B^2}{(1-\lambda)^2T^{3/2}})$, in which case the local variance bound vanishes. As $\lambda$ decreases, the bound can be improved; while when the variances increase, the rate will be hurt. The explanation for how $\theta$ helps or hurts DFedAvgM in the PŁ condition also works for the general nonconvex case. To reach any given $\epsilon > 0$ error, DFedAvgM requires $\mathcal{O}(\max\{\frac{1}{\epsilon^2}, \frac{1}{(1-\lambda)^{4/3}\epsilon^{2/3}}\})$ communication rounds. We recall the communication round needed for FedAvg and its variants to reach the same error in the general nonconvex case: In [54], the authors prove the $\mathcal{O}(\frac{1}{\epsilon^2})$ rate. The same rate results are proved for the variants of FedAvg, including FedProx and SCAFFOLD [39], [53]. Similar to the PŁ case, if $\lambda \leq 1 - \epsilon$, our results match the existing rates for classical federated learning algorithms.

Next, we provide the convergence guarantee for the quantized DFedAvgM, which is stated in the following Proposition.

**Proposition 3.** *Let the sequence $\{\mathbf{x}^t(i)\}_{t\geq 0}$ be generated by the quantized DFedAvgM for all $i \in \{1, 2 \ldots, m\}$, and all the assumptions in Theorem 2. Let $\eta = \Theta(\frac{1}{LK\sqrt{T}})$, as $T$ is sufficiently large, it holds that*

$$\min_{1\leq t\leq T} \mathbb{E}\|\nabla f(\overline{\mathbf{x}^t})\|^2 = \mathcal{O}\Bigg(\frac{(1-\theta)(f(\overline{\mathbf{x}^1}) - \min f)}{\sqrt{T}} + \frac{(1-\theta)(\sigma_l^2 + K\sigma_g^2) + \frac{\theta^2}{(1-\theta)}K(\sigma_l^2+B^2)}{K\sqrt{T}} + \frac{(1-\theta)(\sigma_l^2 + K\sigma_g^2 + KB^2) + \frac{\theta^2}{(1-\theta)}K(\sigma_l^2+B^2)}{(1-\lambda)^2KT^{3/2}} + \sqrt{T}s\Bigg).$$

*If the function $f$ further satisfies the PŁ condition and $\eta = \frac{1}{\nu TK\ln T}$, it follows that*

$$\mathbb{E}(f(\overline{\mathbf{x}^T}) - \min f) = \tilde{\mathcal{O}}\Bigg(\frac{(1-\theta)(f(\overline{\mathbf{x}^1}) - \min f)}{T} + \frac{(1-\theta)\sigma_l^2 + (1-\theta)K\sigma_g^2 + \frac{\theta^2}{(1-\theta)}K(\sigma_l^2+B^2)}{KT} + \frac{(1-\theta)(\sigma_l^2 + K\sigma_g^2 + KB^2) + \frac{\theta^2}{(1-\theta)}K(\sigma_l^2+B^2)}{(1-\lambda)^2KT^3} + Ts\Bigg).$$

Proposition 3 establishes the relations between the measured error and $T$ and $s$. At first glance, the results show that the convergence of the quantized DFedAvgM is degraded by $Ts$ in the PŁ case but $\sqrt{T}s$ in the general nonconvex case. Does that mean quantized DFedAvgM gets decelerated under a stronger assumption? The answer is No! This is because to get the same desired error, the communication round $T$ is different for different cases. For simplicity, assume that $\lambda$ is not too closed to 1. According to Proposition 3, to reach any given $\epsilon > 0$ error in general nonconvex case, we need to set $s = \mathcal{O}(\epsilon^2)$ and set the the number of communication round as $T = \Theta(\frac{1}{\epsilon^2})$. While with PŁ condition, we set $T = \Theta(\frac{1}{\epsilon})$ and $s = \mathcal{O}(\epsilon^2)$. We can see that for both PŁ and general nonconvex cases, the quantization level is set to be the same. In both cases, it follows that $\mathbb{E}(f(\overline{\mathbf{x}^T}) - \min f) = \widetilde{\mathcal{O}}(\epsilon)$. Therefore, under the PŁ condition, the number of communication round is still reduced.

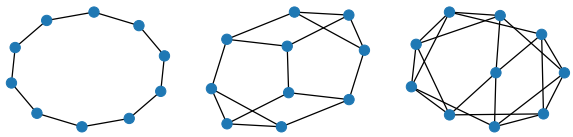


Fig. 3. An illustration of ring graph (left), 3-regular expander graph (middle), and 4-regular expander graph (right) with 10 graph nodes.

In the following, we provide a sufficient condition for communications-savings of the two quantization rules mentioned in section 3.2 that used for quantizing DFedAvgM.

**Proposition 4.** *Assume we use the stochastic or deterministic quantization rule with $b$ bits using stepsize $\eta = \frac{1}{LK\sqrt{T}}$. Assume that the parameters trained in all clients do not overflow, that is, all coordinates are contained in $[-2^{b-1}s, (2^{b-1}-1)s]$. Let Assumptions 1, 2 and 3 hold. If the desired error*

$$\epsilon > (1-\theta)\sqrt{3LBsd}\sqrt[4]{\times} \sqrt{2(f(\overline{\mathbf{x}^0}) - \min f) + \frac{8\sigma_l^2}{K} + 32\sigma_g^2 + \frac{64\theta^2(\sigma_l^2 + B^2)}{(1-\theta)^2}}$$

*and $b < \frac{128}{9} + \frac{32}{d}$, the quantized DFedAvgM can beat DFedAvgM with 32 bits in term of the required communications to reach $\epsilon$.*

We stress that $b$ is the quantization level. Even $d$ is large, $b \leq 14$, which is not small for quantization. Proposition 4 indicates that the superiority of the quantized DFedAvgM retains when the desired error $\epsilon$ is not smaller than $\mathcal{O}((1-\theta)\sqrt{s})$. We can also see that as $K$ increases, the guaranteed lower bound of $\epsilon$ decreases, which demonstrates the necessity of multiple local iterations. Moreover, a larger $\theta$ can also reduce the lower bound.

## 5 Numerical Results

We apply the proposed DFedAvgM with communication quantization to train DNNs for both image classification and language modeling, where we consider both ring and $d$-regular expander graphs [55] based overlay networks for communication, see Fig. 3 for an illustration. We aim to verify that DFedAvgM can train DNNs effectively, especially in communication efficiency. Moreover, we consider the membership privacy protection when DFedAvgM is used for training DNNs. We apply the membership inference attack (MIA) [56] to test the efficiency of (quantized) DFedAvgM in protecting the training data's membership privacy. In MIA, the attack model is a binary classifier [3], which is to decide if a data point is in the training set of the target model. For each of the following dataset, to perform MIA we first split its training set into $D_{\text{shadow}}$ and $D_{\text{target}}$ with the same size. Furthermore, we split $D_{\text{shadow}}$ into two halves with the same size and denote them as $D^{\text{train}}_{\text{shadow}}$ and $D^{\text{out}}_{\text{shadow}}$, and we split $D_{\text{target}}$ by half into $D^{\text{train}}_{\text{target}}$ and $D^{\text{out}}_{\text{target}}$. MIA proceeds as follows: 1) train the shadow model by using $D^{\text{train}}_{\text{shadow}}$; 2) apply the trained shadow model to predict all data points in $D_{\text{shadow}}$ and train the corresponding classification probabilities of belonging to each class. Then we take the top three classification probabilities (or two in the case of binary classification) to form the feature vector for each data point. A feature vector is tagged as1if the corresponding data point is in $D^{\text{train}}_{\text{shadow}}$, and 0 otherwise. Then we train the attack model by leveraging all the labeled feature vectors; 3) train the target model by using $D^{\text{train}}_{\text{target}}$ and obtain feature vector for each point in $D_{\text{target}}$. Finally, we leverage the attack model to decide if a data point is in $D^{\text{train}}_{\text{target}}$. Note the attack model we build is a binary classifier, which is to decide if a data point is in the training set of the target model. For any data $\xi \in D_{\text{target}}$, we apply the attack model to predict its probability ($p$) of belonging to the training set of the target model. Given any fixed threshold $t$ if $p \geq t$, we classify $\xi$ as a member of the training set (positive sample), and if $p < t$, we conclude that $\xi$ is not in the training set (negative sample); so we can obtain different attack results with different thresholds. We can plot the ROC curve for different threshold, and regard the area under the ROC curve (AUC) as an evaluation of the membership inference attack. The target model protects perfect membership privacy if the AUC is 0.5 (attack model performs random guess), and the higher AUC is, the less private the target model is.

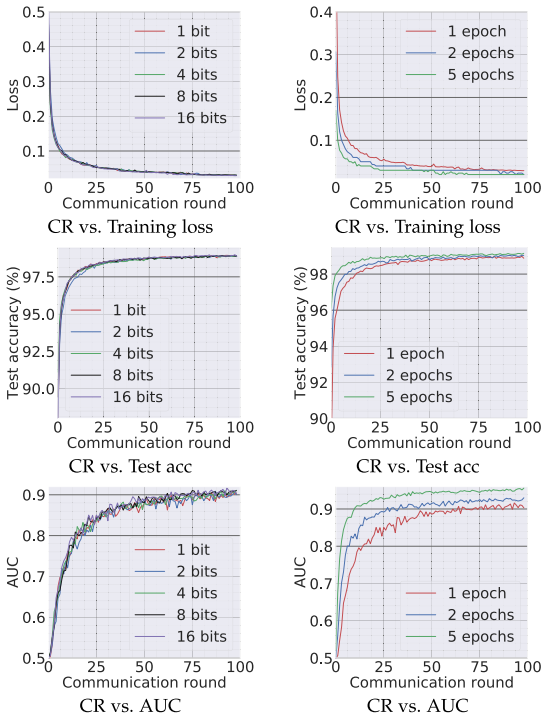



Fig. 4. Training CNN for IID MNIST classification with DFedAvgM using: different communication bits but fix local epoch to one (left column) and different local epochs but fix the communication bits to 16 (right column). Different quantized DFedAvgM performs almost similar, and more local epoch can accelerate training at the cost of faster privacy leakage. CR: communication round.

### 5.1 MNIST Classification

*The Efficiency of DFedAvgM.* We train two models for MNIST classification using 100 clients: 1) A multilayer-perceptron with 2-hidden layers with 200 units each using ReLU activation (199,210 total parameters), which we refer to as 2NN. 2) A CNN with two $5 \times 5$ convolution layers (the first with 32 channels, the second with 64, each followed with $2 \times 2$ max

3. We use a multilayer perceptron with a hidden layer of 64 nodes, followed by a softmax output function as the attack model, which is adapted from [56].

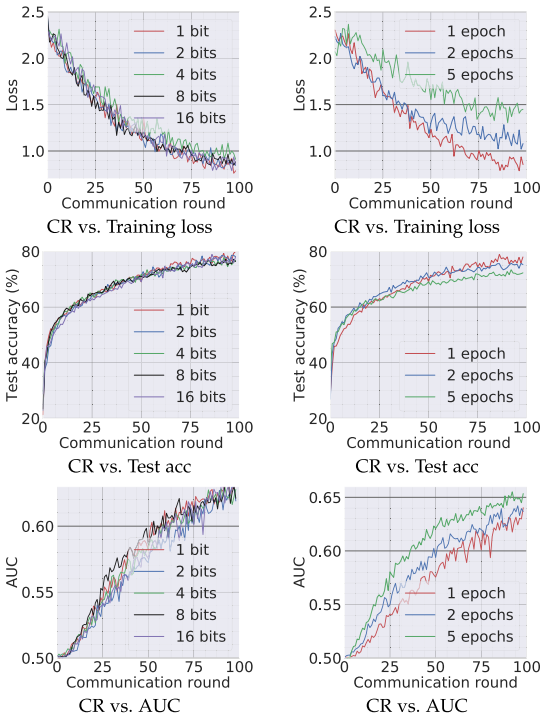



Fig. 5. Training CNN for Non-IID MNIST classification with DFedAvgM using: different communication bits but fix local epoch to one (left column) and different local epochs but fix the communication bits to 16 (right column). Different quantized DFedAvgM does not lead to much difference in performance. More local epoch does not help in accelerating training or protect data privacy.

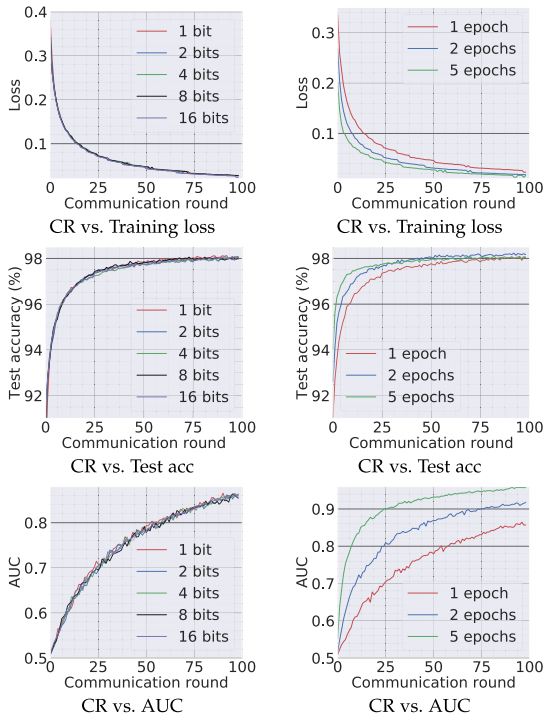



Fig. 6. Training 2NN for IID MNIST classification with DFedAvgM using: different communication bits but fix local epoch to one (first column) and different local epochs but fix the communication bits to 16 (second column). Different quantized DFedAvgM performs almost similar, and more local epoch can accelerate training at the cost of faster privacy leakage.

pooling), a fully connected layer with 512 units and ReLU activation, and the final output layer (1,663,370 total parameters). We study both IID and Non-IID partitioning of the MNIST data over clients, In IID setting, the data is shuffled, and then partitioned into 20 clients each receiving 3000 examples. In Non-IID, we first sort the data by digit label, divide it into 40 shards of size 1500, and assign each of 20 clients 2 shards. In training, we set the local batch size (batch size of the training data on clients) to be 50, learning rate 0.01, and momentum 0.9. Figs. 4 and 5 show the results of training CNN for MNIST classification (Fig. 4: IID and Fig. 5 Non-IID) by DFedAvgM using different communication bits and different local epochs. These results confirm the efficiency of DFedAvgM for training DNNs; in particular, when the clients' data are IID. For both IID and Non-IID settings, the communication bits do not affect the performance of DFedAvgM; as we see that the training loss, test accuracy, and AUC under the membership inference attack are almost identical. Increasing local training epochs can accelerate training for IID setting at the cost of faster privacy leakage. However, for Non-IID, increasing local training epochs does not help DFedAvgM in either training or privacy protection. Training 2NN by DFedAvgM behaves similarly, see Figs. 6 and 7.

*Comparison Between DFedAvgM, FedAvg, and DSGD.* Now, we compare DFedAvgM (DFedAvgM with ring graph), DFedAvgM with 3-expander graph (DFedAvgM-E3), DFedAvgM with 4-expander graph (DFedAvgM-E4), FedAvg,

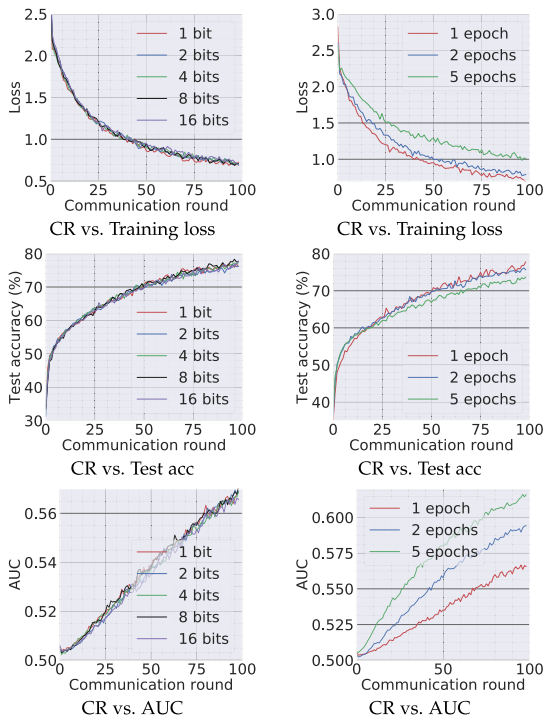



Fig. 7. Training 2NN for Non-IID MNIST classification with DFedAvgM using: different communication bits but fix local epoch to one (first column) and different local epochs but fix the communication bits to 16 (second column). Different quantized DFedAvgM does not lead to much difference in performance. More local epoch does not help in accelerating training or protect data privacy.

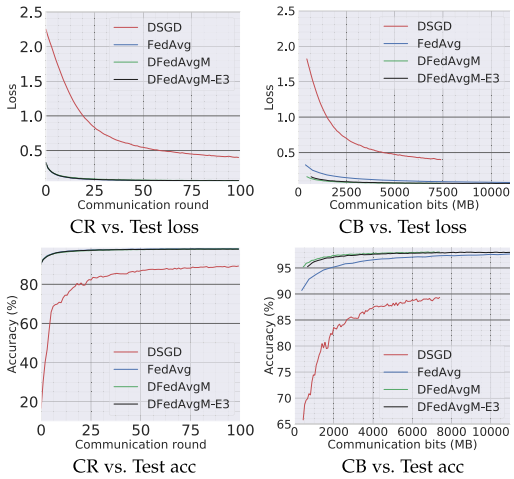



Fig. 8. The efficiency comparison between DSGD, FedAvg, DFedAvgM, and DFedAvgM-E3 in training 2NN for IID MNIST classification. (a) and (c): test loss and test accuracy versus communication round. (b) and (d): test loss and test accuracy versus communication bits. DFedAvgM and DFedAvgM-E3 perform on par with FedAvg in terms of communication rounds, but DFedAvgM and DFedAvgM-E3 are significantly more efficient than FedAvg from the communication cost viewpoint. CR: communication round; CB: communication bits.

and DSGD in training 2NNs for MNIST classification. We use the same local batch size 50 for both FedAvg and DSGD, and the learning rates are both set to $0.1^4$. For FedAvg, we select all clients to get involved in training and communication in each round. Figs. 8 and 9 compare the different algorithms in terms of test loss and test accuracy for IID and Non-IID MNIST, respectively, over communication round and communication cost. Tables 2 and 3 contrast the communication cost of different algorithms, at a given accuracy, for IID and Non-IID MBNIST classification,

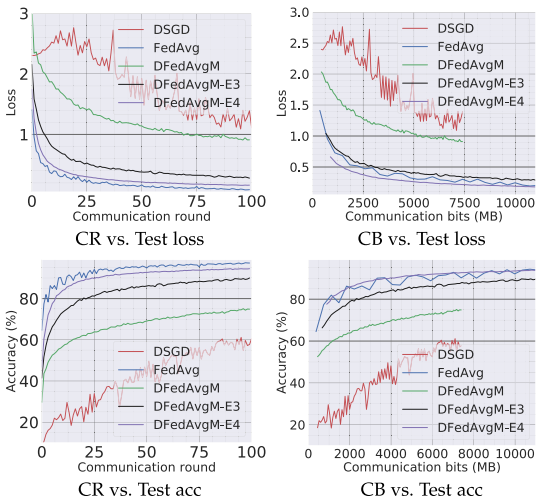



Fig. 9. The efficiency comparison between DSGD, FedAvg, DFedAvgM, DFedAvgM-E3, and DFedAvgM-E4 in training 2NN for Non-IID MNIST classification. (a) and (c): test loss and test accuracy versus communication round. (b) and (d): test loss and test accuracy versus communication bits. In terms of communication round, FedAvg outperforms DFedAvgM and DSGD, but the performance gap between FedAvg and decentralized federated learning can be closed by using a better communication graph, as shown by the result in DFedAvgM-E3 and DFedAvgM-E4. In terms of communication bits, DFedAvgM, DFedAvgM-E3, and DFedAvgM-E4 enjoy significant communication advantages over DSGD and FedAvg. CR: communication round; CB: communication bits.

TABLE 2
The Communication Cost, at a Given Test Accuracy, of FedAvg, DFedAvgM, and DFedAvgM-E3 in Training 2NN for IID MNIST Classification

| Algorithm | 92% | 95% | 98% |
|---|---|---|---|
| FedAvg | 639.3 | 1848.2 | 20331.5 |
| DFedAvgM | 100.9 | 425.4 | 6432.1 |
| DFedAvgM-E3 | 98.9 | 443.6 | 5095.6 |

*(Unit: MB).*

respectively. For the IID case, in terms of communication rounds, DFedAvgM converges as fast as FedAvg, and both are much faster than DSGD. DFedAvgM has a significant advantage over FedAvg and DSGD in communication costs. For Non-IID MNIST, training 2NN by FedAvg can achieve 96.81% test accuracy, but both DFedAvg and DSGD cannot bypass 85%. This disadvantage is because both DSGD and DFedAvgM only communicate with their neighbors, while the neighbors and itself may not contain enough training data to cover all possible classes. Fig. 9 shows that DFedAvgM-E3 and DFedAvgM-E4 significantly close the performance gap between FedAvg and DFedAvgM, indicating that the performance of decentralized federated learning can be significantly improved by using a better sparse network topology for communication, which can balance the performance and communication tradeoff. Indeed, though DFedAvgM-E4 performs slightly inferior to FedAvg in terms of communication round versus test loss and test accuracy. In terms of communication bits versus test loss and test accuracy, DFedAvgM-E4 can even outperform FedAvg when the accuracy is high. We leave the design of optimal sparse communication graph as future work.

### 5.2 LSTM for Language Modeling

We consider the SHAKESPEARE dataset and we follow the processing as that used in [1], resulting in a dataset distributed over 1146 clients in the Non-IID fashion. On this data, we use DFedAvgM to train a stacked character-level LSTM language model, which predicts the next character after reading each character in a line. The model takes a series of characters as input and embeds each of these into a learned 8-dimensional space. The embedded characters are then processed through 2 LSTM layers, each with 256 nodes. Finally, the output of the second LSTM layer is sent to a softmax output layer with one node per character. The full

TABLE 3
Given a Test Accuracy, the Communication Cost of FedAvg, DFedAvgM, and DFedAvgM-E3/E4 in Training 2NN for Non-IID MNIST Classification

| Algorithm | 60% | 70% | 80% |
|---|---|---|---|
| FedAvg | 63.9 | 225.9 | 732.7 |
| DFedAvgM | 982.9 | 4066.3 | NA |
| DFedAvgM-E3 | 333.5 | 898.1 | 2228.9 |
| DFedAvgM-E4 | 105.7 | 391.4 | 782.3 |

*DFed- AvgM cannot reach 80% accuracy in 100 communication rounds. (Unit: MB)*

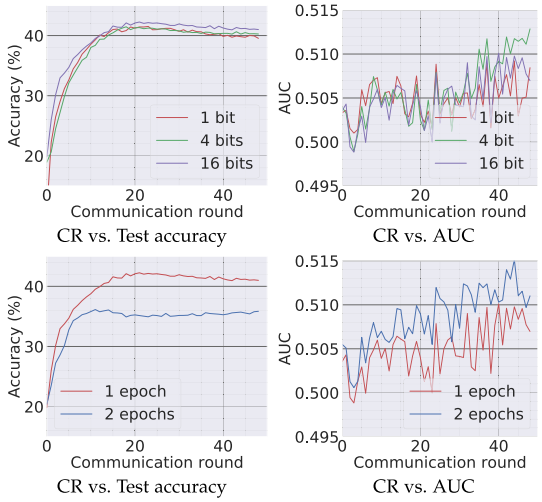



Fig. 10. Training LSTM for SHAKESPEARE classification with DFedAvgM using: different communication bits but fix local epoch to one (first row) and different local epochs but fix the communication bits to 16 (second row). More local epoch does not help in accelerating training or protect data privacy. Using higher precision communication can slightly improve the performance in accuracy; as shown in the top two panels, the accuracy and AUC versus communication round are very close when different quantization levels are used. Overall, using 16 bits has slight advantages over using 1 or 4 bits in terms of accuracy but does not show a remarkable difference in membership privacy protection.

model has 866,578 parameters, and we trained using an unroll length of 80 characters. We set the local batch size to 10, and we use a learning rate of 1.47, which is the same as [1]. The momentum is selected to be 0.9. Fig. 10 plots the communication round versus test accuracy and AUC under MIA for different quantization and different local epochs. These results show that 1) both the accuracy and MIA increase as training goes; 2) higher communication cost can lead to faster convergence; 3) increase local epochs deteriorate the performance of DFedAvgM.

### 5.3 CIFAR10 Classification

Finally, we use DFedAvgM to train ResNet20 for CIFAR10 classification, which consists of 10 classes of $32 \times 32$ images with three channels. There are 50,000 training and 10,000 testing examples, which we partitioned into 20 clients uniformly, and we only consider the IID setting following [1]. We use the same data augmentation and DNN as that used in [1]. The local batch size is set to 50, the learning rate is set to 0.01, and the momentum is set to 0.9. Fig. 11 shows the communication round versus test accuracy and AUC under MIA for different quantization and different local epochs. If the local epoch is set to 1, different communication bits does not lead to a significant difference in training loss, test accuracy, and AUC under MIA. However, for the fixed communication bits 16, increase the local epochs from 1 to 2 or to 5 will make training not even converge.

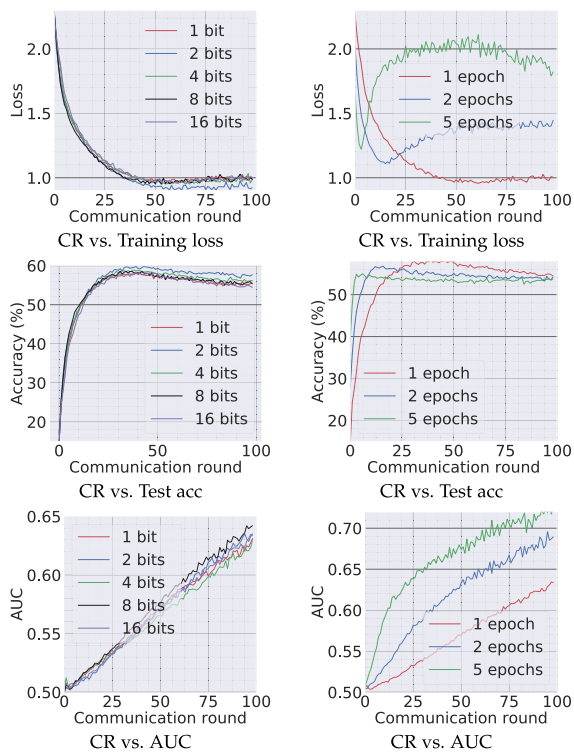



Fig. 11. Training CNN for IID CIFAR10 classification with DFedAvgM using: different communication bits but fix local epoch to one (first column) and different local epochs but fix the communication bits to 16 (second column). More local epochs can accelerate training at the beginning but does not perform very well as training continues. Different quantized DFedAvgMs perform almost similarly. As shown in the left three panels, at the beginning period, the training loss, test accuracy, and AUC are near identical when different communication bits are used; as training goes, using 2 bits communication gives a slightly more accurate prediction than the other cases.

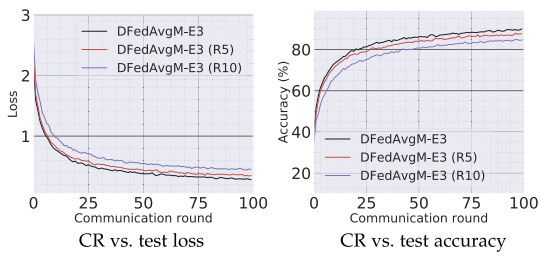



Fig. 12. Contrasting the performance of DFL with a 3-regular expander graph (DFedAvgM-E3) and its variants by randomly removing 5 (DFedAvgM-E3 (R5)) and 10 (DFedAvgM-E3 (R10)) edges.

### 5.4 Graphs With Various Numbers of Neighbors

In the previous studies, we considered both ring and $d$-regular graphs. All nodes in these graphs have the same number of neighbors. We further consider the efficacy of DFL when the underlying graph has various numbers of neighbors. In particular, we repeat the previous Non-IID MNIST experiments in section 5.1 using a 3-regular expander graph but randomly delete 5 or 10 edges from the original graph, and keep all the other experimental settings unchanged. Fig. 12 compares the performance of DFL with a 3-regular expander graph and its variant by randomly removing 5 and 10 edges. When randomly removing 5 edges from the 3-regular expander graphs, the curves of test loss and test

4. We note that DFedAvg requires smaller learning rates than FedAvg and DSGD for numerical convergence.

accuracy versus communication round are very close to that of the original 3-regular expander graphs. Indeed, in terms of training loss and test accuracy, the gaps are within 0.1 and 2%, respectively. Even after removing 10 edges, the convergence of test loss and test accuracy of DFedAvgM with an incomplete 3-regular expander graph is comparable to the complete ones. These results confirm that DFedAvgM learns effectively even when all clients are connected by a graph with various numbers of neighbors.

## 6 Concluding Remarks

In this paper, we proposed a DFedAvgM and its quantized version. Under general nonconvex assumptions, we established the theoretical convergence of DFedAvgM and its quantized version, and we showed that the worst-case convergence rate of (quantized) DFedAvgM is the same as that of DSGD. In particular, we proved a sublinear convergence rate of (quantized) DFedAvgM when the objective functions satisfy the PŁ condition. In the quantization case, we present a sufficient condition that guarantees quantized DFedAvgM outperforms DFedAvgM. We perform extensive numerical experiments to verify the efficiency of DFedAvgM and its quantized version in training ML models and protect membership privacy. The numerical results show that the (quantized) DFedAvgM outperforms FedAvg in both iteration efficiency and communication costs.

One possible future direction is to remove Assumption 3 used in our analysis. A particular idea is leveraging unbounded gradient convergence analysis [57], [58], and the core technique is using the inequality for functions with the Lipschitz gradient established in [59]. However, due to the complicated form of the DFedAvgM, we need to deal with nontrivial details to get the desired Lyapunov functions.

## References


[1] H. B. Mcmahan, E. Moore, D. Ramage, S. Hampson, and B. A. Y. Arcas, "Communication-efficient learning of deep networks from decentralized data," in *Proc. 20th Int. Conf. Artif. Intell. Statist.*, 2017, pp. 1273–1282.

[2] H. Robbins and S. Monro, "A stochastic approximation method," *Ann. Math. Statist.*, vol. 22, no. 3, pp. 400–407, 1951.

[3] M. Zinkevich, M. Weimer, L. Li, and A. J. Smola, "Parallelized stochastic gradient descent," in *Proc. Adv. Neural Inf. Process. Syst.*, 2010, pp. 2595–2603.

[4] X. Lian, C. Zhang, H. Zhang, C.-J. Hsieh, W. Zhang, and J. Liu, "Can decentralized algorithms outperform centralized algorithms? a case study for decentralized parallel stochastic gradient descent," in *Proc. Adv. Neural Inf. Process. Syst.*, 2017, pp. 5330–5340.

[5] A. Lalitha, S. Shekhar, T. Javidi, and F. Koushanfar, "Fully decentralized federated learning," in *Proc. 3rd Workshop Bayesian Deep Learn.*, 2018, pp. 1–9.

[6] A. Lalitha, O. C. Kilinc, T. Javidi, and F. Koushanfar, "Peer-to-peer federated learning on graphs," 2019, *arXiv:1901.11173*.

[7] I. Sutskever, J. Martens, G. Dahl, and G. Hinton, "On the importance of initialization and momentum in deep learning," in *Proc. Int. Conf. Mach. Learn.*, 2013, pp. 1139–1147.

[8] T.-M. H. Hsu, H. Qi, and M. Brown, "Measuring the effects of non-identical data distribution for federated visual classification," 2019, *arXiv:1909.06335*.

[9] S. J. Reddi et al., "Adaptive federated optimization," in *Proc. Int. Conf. Learn. Representations*, 2021, pp. 1–38.

[10] T. Chen, G. Giannakis, T. Sun, and W. Yin, "Lag: Lazily aggregated gradient for communication-efficient distributed learning," in *Proc. Adv. Neural Inf. Process. Syst.*, 2018, pp. 5050–5060.

[11] J. Sun, T. Chen, G. Giannakis, and Z. Yang, "Communication-efficient distributed learning via lazily aggregated quantized gradients," in *Proc. Adv. Neural Inf. Process. Syst.*, 2019, pp. 3365–3375.

[12] T. Li, A. K. Sahu, M. Zaheer, M. Sanjabi, A. Talwalkar, and V. Smithy, "FedDANE: A federated Newton-type method," in *Proc. 53rd Asilomar Conf. Signals, Syst., Comput.*, 2019, pp. 1227–1231.

[13] A. Imteaj, U. Thakker, S. Wang, J. Li, and M. H. Amini, "A survey on federated learning for resource-constrained IoT devices," *IEEE Internet Things J.*, vol. 9, no. 1, pp. 1–24, Jan. 2022.

[14] A. Khaled, K. Mishchenko, and P. Richtárik, "First analysis of local GD on heterogeneous data," 2019, *arXiv:1909.04715*.

[15] X. Li, K. Huang, W. Yang, S. Wang, and Z. Zhang, "On the convergence of FedAvg on non-IID data," in *Proc. Int. Conf. Learn. Representations*, 2020, pp. 1–26.

[16] A. K. Sahu, T. Li, M. Sanjabi, M. Zaheer, A. Talwalkar, and V. Smith, "On the convergence of federated optimization in heterogeneous networks," 2018, *arXiv:1812.06127*.

[17] C. Zhang, Y. Xie, H. Bai, B. Yu, W. Li, and Y. Gao, "A survey on federated learning," *Knowl.-Based Syst.*, vol. 216, 2021, Art. no. 106775.

[18] Q. Yang, Y. Liu, Y. Cheng, Y. Kang, T. Chen, and H. Yu, *Federated Learning*. San Rafael, CA, USA: Morgan & Claypool, 2019.

[19] H. B. McMahan et al., "Advances and open problems in federated learning," *Found. Trends Mach. Learn.*, vol. 14, no. 1/2, pp. 1–210, 2021.

[20] T. Li, A. K. Sahu, A. Talwalkar, and V. Smith, "Federated learning: Challenges, methods, and future directions," 2019, *arXiv:1908.07873*.

[21] S. Boyd, A. Ghosh, B. Prabhakar, and D. Shah, "Gossip algorithms: Design, analysis and applications," in *Proc. 24th Annu. Joint Conf. IEEE Comput. Commun. Soc.*, 2005, pp. 1653–1664.

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

[23] L. Schenato and G. Gamba, "A distributed consensus protocol for clock synchronization in wireless sensor network," in *Proc. 46th IEEE Conf. Decis. Control*, 2007, pp. 2289–2294.

[24] T. C. Aysal, M. E. Yildiz, A. D. Sarwate, and A. Scaglione, "Broadcast gossip algorithms for consensus," *IEEE Trans. Signal Process.*, vol. 57, no. 7, pp. 2748–2761, Jul. 2009.

[25] A. Nedic and A. Ozdaglar, "Distributed subgradient methods for multi-agent optimization," *IEEE Trans. Autom. Control*, vol. 54, no. 1, pp. 48–61, Jan. 2009.

[26] A. I. Chen and A. Ozdaglar, "A fast distributed proximal-gradient method," in *Proc. 50th Annu. Conf. Commun., Control, Comput.*, 2012, pp. 601–608.

[27] D. Jakovetić, J. Xavier, and J. M. F. Moura, "Fast distributed gradient methods," *IEEE Trans. Autom. Control*, vol. 59, no. 5, pp. 1131–1146, May 2014.

[28] I. Matei and J. S. Baras, "Performance evaluation of the consensus-based distributed subgradient method under random communication topologies," *IEEE J. Sel. Topics Signal Process.*, vol. 5, no. 4, pp. 754–771, Aug. 2011.

[29] K. Yuan, Q. Ling, and W. Yin, "On the convergence of decentralized gradient descent," *SIAM J. Optim.*, vol. 26, no. 3, pp. 1835–1854, 2016.

[30] J. Zeng and W. Yin, "On nonconvex decentralized gradient descent," *IEEE Trans. Signal Process.*, vol. 66, no. 11, pp. 2834–2848, Jun. 2018.

[31] B. Sirb and X. Ye, "Consensus optimization with delayed and stochastic gradients on decentralized networks," in *Proc. IEEE Int. Conf. Big Data*, 2016, pp. 76–85.

[32] G. Lan, S. Lee, and Y. Zhou, "Communication-efficient algorithms for decentralized and stochastic optimization," *Math. Program.*, vol. 180, no. 1, pp. 237–284, 2020.

[33] X. Lian, W. Zhang, C. Zhang, and J. Liu, "Asynchronous decentralized parallel stochastic gradient descent," in *Proc. 35th Int. Conf. Mach. Learn.*, 2018, pp. 3043–3052.

[34] T. Sun, P. Yin, D. Li, C. Huang, L. Guan, and H. Jiang, "Non-ergodic convergence analysis of heavy-ball algorithms," in *Proc. AAAI Conf. Artif. Intell.*, 2019, pp. 5033–5040.

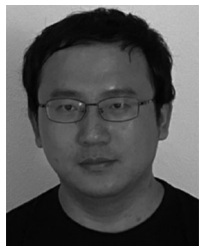
[35] R. Xin and U. A. Khan, "Distributed heavy-ball: A generalization and acceleration of first-order methods with gradient tracking," *IEEE Trans. Autom. Control*, vol. 65, no. 6, pp. 2627–2633, Jun. 2020.

[36] A. Reisizadeh, A. Mokhtari, H. Hassani, and R. Pedarsani, "Quantized decentralized consensus optimization," in *Proc. IEEE Conf. Decis. Control*, 2018, pp. 5838–5843.

[37] H. Xing, O. Simeone, and S. Bi, "Decentralized federated learning via SGD over wireless D2D networks," in *Proc. IEEE 21st Int. Workshop Signal Process. Adv. Wirel. Commun.*, 2020, pp. 1–5.
[38] S. Boyd, P. Diaconis, and L. Xiao, "Fastest mixing markov chain on a graph," *SIAM Rev.*, vol. 46, no. 4, pp. 667–689, 2004.
[39] T. Li, A. K. Sahu, M. Zaheer, M. Sanjabi, A. Talwalkar, and V. Smith, "Federated optimization in heterogeneous networks," *Proc. 1st Adaptive Multitask Learn. Workshop*, 2019.
[40] H. Karimi, J. Nutini, and M. Schmidt, "Linear convergence of gradient and proximal-gradient methods under the polyak-łojasiewicz condition," in *Proc. Joint Eur. Conf. Mach. Learn. Knowl. Discov. Databases*, 2016, pp. 795–811.
[41] S. J. Reddi, A. Hefny, S. Sra, B. Poczos, and A. Smola, "Stochastic variance reduction for nonconvex optimization," in *Proc. Int. Conf. Mach. Learn.*, 2016, pp. 314–323.
[42] D. J. Foster, A. Sekhari, and K. Sridharan, "Uniform convergence of gradients for non-convex learning and optimization," in *Proc. Adv. Neural Inf. Process. Syst.*, 2018, pp. 8759–8770.
[43] B. T. Polyak, "Gradient methods for minimizing functionals," *Zhurnal Vychislitel'noi Matematiki i Matematicheskoi Fiziki*, vol. 3, no. 4, pp. 643–653, 1963.
[44] S. Lojasiewicz, "A topological property of real analytic subsets," *Coll. du CNRS, Les Équations aux Dérivées Partielles*, vol. 117, pp. 87–89, 1963.
[45] K. Zhong, Z. Song, P. Jain, P. L. Bartlett, and I. S. Dhillon, "Recovery guarantees for one-hidden-layer neural networks," in *Proc. Int. Conf. Mach. Learn.*, 2017, pp. 4140–4149.
[46] C. Liu, L. Zhu, and M. Belkin, "Loss landscapes and optimization in over-parameterized non-linear systems and neural networks," *Appl. Comput. Harmon. Anal.*, vol. 59, pp. 85–116, 2022.
[47] B. T. Polyak, "Some methods of speeding up the convergence of iteration methods," *Ussr Comput. Math. Math. Phys.*, vol. 4, no. 5, pp. 1–17, 1964.
[48] M. Li, D. G. Andersen, A. J. Smola, and K. Yu, "Communication efficient distributed machine learning with the parameter server," in *Proc. Adv. Neural Inf. Process. Syst.*, 2014, pp. 19–27.
[49] D. Alistarh, D. Grubic, J. Li, R. Tomioka, and M. Vojnovic, "QSGD: Communication-efficient SGD via gradient quantization and encoding," in *Proc. Adv. Neural Inf. Process. Syst.*, 2017, pp. 1709–1720.
[50] S. Magnússon, H. Shokri-Ghadikolaei, and N. Li, "On maintaining linear convergence of distributed learning and optimization under limited communication," *IEEE Trans. Signal Process.*, vol. 68, pp. 6101–6116, 2020.
[51] S. Shalev-Shwartz, Y. Singer, N. Srebro, and A. Cotter, "Pegasos: Primal estimated sub-gradient solver for SVM," *Math. Program.*, vol. 127, no. 1, pp. 3–30, 2011.
[52] A. Nemirovski, A. Juditsky, G. Lan, and A. Shapiro, "Robust stochastic approximation approach to stochastic programming," *SIAM J. Optim.*, vol. 19, no. 4, pp. 1574–1609, 2009.
[53] S. P. Karimireddy, S. K. Kale, M. Mohri, S. J. Reddi, S. U. Stich, and A. T. Suresh, "Scaffold: Stochastic controlled averaging for federated learning," in *Proc. 37th Int. Conf. Mach. Learn.*, 2020, pp. 5132–5143.
[54] H. Yu, S. Yang, and S. Zhu, "Parallel restarted SGD for non-convex optimization with faster convergence and less communication," in *Proc. AAAI Conf. Artif. Intell.*, 2019, pp. 5693–5700.
[55] S. Hoory, N. Linial, and A. Wigderson, "Expander graphs and their applications," *Bull. Amer. Math. Soc.*, vol. 43, no. 4, pp. 439–561, 2006.
[56] A. Salem, Y. Zhang, M. Humbert, P. Berrang, M. Fritz, and M. Backes, "Ml-leaks: Model and data independent membership inference attacks and defenses on machine learning models," in *Proc. Annu. Netw. Distrib. Syst. Secur. Symp.*, 2019, pp. 1–15.
[57] Y. Lei, T. Hu, and K. Tang, "Stochastic gradient descent for nonconvex learning without bounded gradient assumptions," 2019, *arXiv:1902.00908*.
[58] L. M. Nguyen, P. H. Nguyen, M. van Dijk, P. Richtárik, K. Scheinberg, and M. Takáč, "SGD and hogwild! convergence without the bounded gradients assumption," 2018, *arXiv:1802.03801*.
[59] Y. Nesterov, *Lectures on Convex Optimization*, Berlin, Germany: Springer, vol. 137, 2018.

**Tao Sun** received the PhD degree from the National University of Defense Technology, in 2018. Currently, he is an assistant professor with National Laboratory for Parallel and Distributed Processing, National University of Defense Technology. His research interests include optimization for machine learning, reinforcement learning, distributed system and neutral networks.

**Dongsheng Li** received the PhD degree from the National University of Defense Technology, in 2005. He is currently a full professor with the National Laboratory for Parallel and Distributed Processing, National University of Defense Technology. His research interests include distributed computing, cloud computing, computer network, and large-scale data management. He was awarded the prize of the National Excellent Doctoral Dissertation of China by Ministry of Education of China, in 2008.



**Bao Wang** received the PhD degree from Michigan State University, in 2016. He is currently an assistant professor of mathematics and joint with the Scientific Computing and Imaging Institute, University of Utah. His research interests include deep learning and scientific computing.