Accelerating Decentralized Federated Learning with Probabilistic Communication in Heterogeneous Edge Computing

Jianchun Liu, *Member, IEEE,* Jiaming Yan, *Hongli Xu, *Member, IEEE,* Lun Wang, Zhiyuan Wang, Jinyang Huang, Chunming Qiao, *Fellow, IEEE*

Abstract—Decentralized federated learning (DFL) has gained popularity for training machine learning models on massive data in edge computing, as it avoids the potential bottleneck of conventional parameter server architectures. However, the existing DFL solutions typically use deterministic topologies that struggle with both system heterogeneity and non-IID local data, resulting in high bandwidth costs and slow convergence rates. In this paper, we propose a novel mechanism called Communication-efficient Decentralized Federated Learning (CEDFL) to accelerate model training. In CEDFL, each worker will communicate with each of its neighbors (*i.e.*, model exchange) according to a certain probability at each epoch, so as to reduce bandwidth consumption. To this end, we then propose an efficient algorithm to adaptively determine the optimal probability for each worker pair according to real-time system situations (*e.g.*, data distribution and bandwidth resource). Our proposed mechanism has been extensively tested on classical models and datasets, and the results demonstrate its high effectiveness. CEDFL has been shown to reduce completion time for model training by approximately 55% and improve test accuracy by 11% under the bandwidth constraint, compared to state-of-the-art solutions.

Index Terms—*Decentralized Federated Learning, Edge Computing, Non-IID data, Probabilistic Communication.*

1 Introduction

With the rise of the Internet of Things (IoT), a significant amount of data is being generated every day by intelligent devices such as mobile phones and wearable devices [1]. As the storage and computation capabilities of these devices continue to increase, the concept of edge computing has emerged, allowing for data to be stored locally while computation functions are pushed to the network edge [2]. This development has also led to the application of federated learning (FL), which facilitates distributed machine learning at the network edge [3], [4].

A typical FL system, such as Favor [5] and R2SP [6], requires a logically centralized parameter server (PS) to coordinate the large federation of the participating workers (*e.g.*, various devices or edge nodes), as shown in the left plot of Fig. 1. At the beginning of model training, the PS first sends the fresh global model to workers. Then, the workers train local models over their own datasets and send the model updates (*e.g.*, gradients or parameters) to the parameter server for synchronization. Subsequently, these model updates will be aggregated to derive the latest global model on the parameter server. However, with more and more workers participating in the federated training, the PS may become the system bottleneck due to the frequent communication between PS and a large number of workers, leading to the risk of single-point failure, network congestion and non-ideal convergence rate. For example, as the size of VGG16 for the ImageNet dataset reaches 512MB [7], FL requires to consume more than 50GB of bandwidth in one global round if there are hundreds of workers, bringing an enormous traffic workload to the PS. Moreover, the transmission of users' private models to the PS also gives rise to privacy concerns, particularly in edge computing scenarios such as mobile phones or security monitoring, as the PS may be vulnerable to manipulation by hackers.

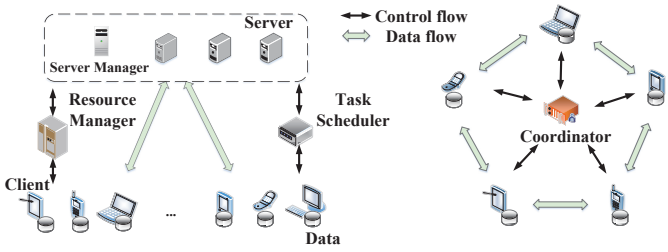


Fig. 1: Illustration of the two different frameworks for federated learning. Left: centralized FL; right: decentralized FL.

To address the aforementioned risks, several decentralized model training methods have been proposed, such as decentralized federated learning (DFL). These methods coordinate workers to establish a peer-to-peer (P2P) communication network, as illustrated in the right plot of Fig. 1. Each worker trains a local model on its own dataset and aggregates models from its neighbors using

- *J. Liu, J. Yan and H. Xu are with the School of Computer Science and Technology, University of Science and Technology of China, Hefei, Anhui, China, 230027, and also with Suzhou Institute for Advanced Research, University of Science and Technology of China, Suzhou, Jiangsu, China, 215123. E-mail: jcliu17@ustc.edu.cn, jmyan@mail.ustc.edu.cn, xuhongli@ustc.edu.cn.*
- *L. Wang and Z. Wang are with the Alibaba Group, Hangzhou, Zhejiang, China, 310000. E-mail: yuchen.wl@alibaba-inc.com, pushi.wzy@alibaba-inc.com.*
- *J. Huang is with the School of Computer and Information, Hefei University of Technology, Anhui, China, 230002. E-mail: hjy@hfut.edu.cn*
- *C. Qiao is with the Department of Computer Science and Engineering, University at Buffalo, the State University of New York, Buffalo, NY 14260, USA. E-mail: qiao@buffalo.edu.*
- **Corresponding author: Hongli Xu.*

TABLE 1: The advantages and disadvantages of the previous schemes and ours.

Schemes	Non-IID Data	Limited Bandwidth	System Heterogeneity
BAT [22]	✓	×	✓
Ring [23]	×	✓	×
NetMax [12]	×	✓	✓
Ours	✓	✓	✓

the P2P setting. By avoiding the need to transmit local models from workers to the PS, these methods eliminate the potential for single-point failure or network congestion on the PS, which in turn accelerates the training process. [8]. Without exposing models to the PS, the users' privacy can also be efficiently protected [9].

With the above advantages, decentralized federated learning (DFL) with P2P communication has been extensively studied in the pieces of literature [10], [11], [12]. However, three critical challenges make it difficult to perform efficient model training. 1) **Non-IID Local Data:** DFL relies on stochastic gradient descent (SGD) [13], which is widely used in training deep networks due to its good empirical performance. When training over independent and identically distributed (IID) data, the stochastic gradient provides an unbiased estimate of the full gradients [14]. However, in practice, it is unrealistic to assume that the local data on each edge device is always IID. For example, in a patient monitoring scenario [15], different patients' data often exhibit a high level of heterogeneity due to diverse human biological features, physical environments, and even sensor biases [16]. When training over non-IID data, where each worker trains only on a single class of data, the accuracy of federated learning can be reduced by approximately 55% [17]. 2) **Limited Communication Bandwidth:** Different from the datacenters with sufficient communication resources, the bandwidth between the workers is always limited in edge computing [18]. For example, the bandwidth available within typical WANs is usually limited to 5∼25Mb/s [19], which is significantly less than that available within datacenters (*e.g.*, over 10Gb/s [20]). As the size of VGG16 for the ImageNet dataset reaches 512MB [7], the network may still easily get congested because of the frequent model transmission. 3) **System Heterogeneity:** The link speeds or network resources (*e.g.*, bandwidth budgets) between the workers are significantly different, leading to the system heterogeneity [21]. For instance, the bandwidth budgets of the workers in the network always vary because of different charging rules made by the internet service providers (ISPs) and personalized choice of the users [12].

A natural solution of decentralized federated learning is to perform model training on a deterministic topology. Specifically, in order to achieve similar training performance with the centralized FL on non-IID data, each worker directly or indirectly exchanges models with all other workers in the logically fully-connected topology (*e.g.*, grid [24], referred to as the BAT scheme), resulting in high bandwidth cost [25]. To this end, some studies propose the communication-efficient schemes (*e.g.*, Ring-Allreduce [11], [26], referred to as the Ring scheme) which perform fewer model exchanges among the workers for decentralized federated learning. In this solution, each worker only communicates with its two neighbors in the network to form a logically ring topology. However, the training performance of these schemes is usually worse than that of BAT (*e.g.*, lower test accuracy or slower convergence rate) because of neglecting the system heterogeneity and less communication among the workers, especially under the non-IID setting.

Different from the above solutions, we adopt the probabilistic topology in which the communications among the workers can be dynamically adapted according to the network conditions (*e.g.*, bandwidth constraints and different link speeds). The most related work to our paper is a decentralized federated learning approach, namely NetMax [12], that enables workers to communicate preferably through high-speed links to significantly speed up the training process. Specifically, in each training epoch, the fastest link between any two workers will be allocated the largest probability. However, this solution only chooses one link between two workers while ignoring the features of non-IID data, leading to an enormous number of training epochs until model convergence under non-IID setting, which will be validated through experiments in Section 4. We summarize the advantages and disadvantages of the previous solutions in Table 1. We observe that none of the aforementioned works can fully address the three critical challenges in DFL.

To accommodate bandwidth constraints, system heterogeneity and non-IID data, we propose a communication-efficient decentralized federated learning (CEDFL) mechanism for edge computing. CEDFL adopts the probabilistic communication between the workers in the network, which will significantly reduce the bandwidth cost without training performance degradation. Specifically, in CEDFL, the coordinator will assign a proper probability to each link, also called communication policy, according to the difference of bandwidth budgets and data distributions among workers, so that two connected workers will exchange their local models with this probability at each training epoch. As a result, our proposed mechanism can effectively improve the training performance and accelerate the training process of DFL over non-IID data under bandwidth constraints and system heterogeneity. The main contributions of this paper are summarized as follows:

- We design a decentralized federated learning mechanism, named CEDFL, that is optimized for communication efficiency by utilizing probabilistic communication. During each epoch, two connected workers will exchange their local models with a suitable probability. Furthermore, we have conducted a formal proof of CEDFL's convergence.
- We then propose an efficient algorithm (termed CPG-DFL) to adaptively determine the optimal probability for each link in the network, thus reducing the completion time and optimizing the utilization of bandwidth resources during the model training.
- The proposed algorithm has demonstrated high effectiveness through extensive experiments on standard models and datasets. Notably, when compared to benchmark methods, CEDFL has been observed to decrease completion time by approximately 55% and enhance training accuracy by roughly 11% while operating under bandwidth constraints.

2 PRELIMINARIES AND PROBLEM FORMULATION

TABLE 2: Key Notations

Symbol	Semantics
T	the number of training rounds
ϵ	the number of local training epochs in a round
$\mathcal{M}$	a set of all workers
$\mathbf{A}$	a symmetric adjacency matrix
$\mathcal{N}_i$	the neighbors set of worker i
Γ_i	the local dataset on worker i
$\mathbf{P}^t$	the communication policy in round t
$\mathbf{W}^t$	the gossip matrix in round t
Θ^t	the data distribution difference in round t
$\mathcal{Z}^t$	the communication matrix in round t
$p_{i,j}$	the communication probability of workers i and j
p_i	the probability of worker i communicating with one of its neighbors at any epoch
p_{min}	the smallest value among $p_i, \forall i \in [m]$
ω_i	the local model of worker i in round t
$F(\omega;\xi)$	the loss value of model ω over mini-batch ξ_i
$f(\Omega)$	the global objective of DFL
B_i	the traffic budget for worker i
b	the traffic consumption for a model exchange
η	the learning rate
ρ	the hyper-parameter for model aggregation

2.1 Network Model

An edge computing system comprises a group of workers, such as IoT devices or small base stations, that collaborate to train machine learning models using their respective local datasets. Rather than sharing their original data, each worker communicates model parameters with its neighbors. Communication between two workers takes the form of a logistic P2P (or device-to-device) procession. Note that asynchronous DFL (ADFL) may be challenging to combine with complementary techniques like differential privacy [10]. Thus, we mainly perform DFL and train models in a synchronous manner. The training process of DFL under the synchronous scheme involves many global rounds (*i.e.*, T) until convergence, each of which includes local model training, model exchanging, and model aggregation. Note that there may be one or several local training epochs (*i.e.*, ϵ) in one global round.

The P2P network topology can be represented as an undirected graph $\mathcal{G} = (\mathcal{M}, \mathcal{E})$. Here, $\mathcal{M} = \{1, 2, ..., m\}$ represents the set of workers, and $\mathcal{E}$ indicates the links between workers in the network. The graph can be denoted as a symmetric adjacency matrix $\mathbf{A} = \{a_{i,j} \in \{0, 1\}, 1 \leq i, j \leq m\}$, where $a_{i,j}$ indicates if a link exists between worker i and worker j. The neighbors set of worker i is denoted as $\mathcal{N}_i$. The degree matrix $\mathbf{D}$ is defined as a diagonal matrix, where $\mathbf{D}_{i,i} = |\mathcal{N}_i| = \sum_{j \in \mathcal{N}_i} a_{i,j}$, and $\mathbf{D}_{i,j} = 0, \forall i \neq j$. By combining the adjacency matrix and the degree matrix, the Laplacian matrix can be obtained as $\mathbf{L} = \mathbf{D} - \mathbf{A}$. As per spectral graph theory [27], $\lambda_2(\mathbf{L}) > 0$ if and only if the topology is connected, where λ_m indicates the m-th smallest eigenvalue of matrix $\mathbf{L}$.

A coordinator (*e.g.*, a worker or a cloud server) collects global information about the model training (*e.g.*, test accuracy) and network conditions (*e.g.*, network traffic), and determines the communication policy (*i.e.*, the probability of each link) at each round. It is worth noting that the role of the coordinator in our system is distinct from that of a parameter server which is responsible for aggregating local gradients and updating the global model. The required network information (*e.g.*, traffic consumption) for probability decisions at the coordinator can be represented by only several bytes, which is smaller than the model size with hundreds of millions of bytes or more. Thus, it is reasonable to assume that the cost for information collection can be ignored [28]. For ease of expression, we list some important notations in Table 2.

2.2 Decentralized Federated Learning

Each worker $i \in \mathcal{M}$ possesses a local dataset Γ_i, where Γ_i and Γ_j, $i \neq j$, may have overlapping data points. The distribution of local data may not necessarily be independent and identically distributed. To enable collaborative training of a shared machine learning model through P2P communication, each worker i trains a local model w_i on its own dataset and shares the model parameters with its neighboring workers without disclosing their respective training samples to one another directly. Let $F(\omega;\xi)$ represent the loss function of model ω over the mini-batch ξ. The optimization objective of DFL can be formulated as follows [12]:

$$\min_{\Omega} f(\Omega) = \sum_{i=1}^{m} [\mathbb{E}_{\xi \in \Gamma_i}[F(\omega_i;\xi)] + \frac{\rho}{2} \sum_{j=1}^{m} a_{i,j} \|\omega_i - \omega_j\|_2^2] \quad (1)$$

where $\Omega = \{\omega_1, \omega_2, \cdots, \omega_m\}$, and ω_i is the local model of worker i. The weight ρ controls the relative importance of the model difference between neighboring workers in the objective function. The Euclidean distance between the local models of worker i and j is measured by $\|\omega_i - \omega_j\|_2^2$, where $\|\cdot\|_2$ represents the Euclidean or ℓ_2 norm of the vectors. Since DFL lacks global aggregation, the local models of workers are expected to be similar to each other, *i.e.*, with small consensus distance $\|\bar{\omega} - \omega_i\|$ for each worker i, where $\bar{\omega}$ is the averaged model of all workers [8].

We adopt the decentralized stochastic gradient descent (DSGD) method [13] to solve the optimization problem in Eq. (1). Each worker performs the model update by incorporating the received local model parameters from the neighbor(s) and its own local gradient information. Specifically, in each global round $t \in \{1, 2, \cdots, T\}$, worker i first performs local training epochs for ϵ times via the SGD algorithm:

$$\omega_i^{t,k+1} \leftarrow \omega_i^{t,k} - \eta \nabla F(\omega_i^{t,k}, \xi_i^{t,k}) \quad t \in \{1, 2, \cdots, \epsilon - 1\} \quad (2)$$

where $\xi_i^{t,k}$ is a mini-batch randomly sampled from the dataset Γ_i, and η is learning rate. $\omega_i^{t,k}$ represent the local model of worker i at k-th local epoch of global round t. Then, worker i pulls its neighbors' models and aggregates them to update the local model:

$$\omega_i^{t+1} \leftarrow \omega_i^{t+\frac{1}{2}} - \eta\rho \sum_{j \in \mathcal{Z}_i^t} \frac{1}{p_{i,j}^t} (\omega_i^{t+\frac{1}{2}} - \omega_j^{t+\frac{1}{2}}) \quad (3)$$

where $\omega_i^{t+\frac{1}{2}} = \omega_i^{t,\epsilon}$. We denote the communication policy (*i.e.*, probability matrix) in round t as $\mathbf{P}^t = \{p_{i,j}^t\}_{1 \leq i,j \leq m}$, where $p_{i,j}$

Algorithm 1 Procedure at the Coordinator

1: Initialize the data distribution difference $\Theta^0 \leftarrow [0]_{m\times m}$, and the traffic budget $\mathbf{B}^0 \leftarrow [B_i]_{1\times m}$
2: **for** each global round $t \in \{1, 2, \cdots, T\}$ **do**
3: Send the request to workers for information collection
4: Receive the requested information to update Θ^t and $\mathbf{B}^t$
5: Determine the communication policy $\mathbf{P}^t$ and the parameter ρ^t by the CPG-DFL algorithm (Section 3.2)
6: **while** True **do**
7: Generate a random matrix $\mathcal{Q}$ with a different seed
8: Construct communication matrix $\mathcal{Z}^t$ with $\mathbf{P}^t$ and $\mathcal{Q}$
9: Compute the Laplacian matrix of $\mathcal{Z}^t$ (*i.e.*, $\mathbf{L}_z$)
10: **if** $\lambda_2(\mathbf{L}_z) > 0$ **then**
11: Break
12: Send $\mathbf{P}_i^t$, $\mathcal{Z}_i^t$ and ρ^t to each worker i

Algorithm 2 Procedure at Worker i

1: Initialize local model ω_i^0, parameter ρ^0, and learning rate η
2: Initialize the communication probabilities $\mathbf{P}_i^0 = [\frac{1}{m}]_{1\times m}$, the neighbors set $\mathcal{Z}_i^0 = \{j | a_{i,j} = 1\}$, and the vector of data distribution difference $\Theta_i \leftarrow [0]_{1\times m}$
3: **for each** global round $t \in \{1, 2, \cdots, T\}$ **do**
4: **if** new policy is received **then**
5: Update $\mathbf{P}_i^t$, $\mathcal{Z}_i^t$ and parameter ρ^t
6: **for** $k \in \{1, 2, \cdots, \epsilon\}$ **do**
7: Sample a random mini-batch $\xi_i^{t,k}$ from Γ_i
8: $\omega_i^{t,k+1} \leftarrow \omega_i^{t,k} - \eta \nabla F(\omega_i^{t,k}, \xi_i^{t,k})$
9: Set $\omega_i^{t+\frac{1}{2}} \leftarrow \omega_i^{t,\epsilon}$
10: Send its model $\omega_i^{t+\frac{1}{2}}$ to the neighbors in $\mathcal{Z}_i^t$
11: Receive the model $\omega_j^{t+\frac{1}{2}}$ from every neighbor $j \in \mathcal{Z}_i^t$
12: $\omega_i^{t+1} \leftarrow \omega_i^{t+\frac{1}{2}} - \eta\rho^t \sum_{j\in\mathcal{Z}_i^t} \frac{1}{p_{i,j}^t}(\omega_i^{t+\frac{1}{2}} - \omega_j^{t+\frac{1}{2}})$
13: Update the vector Θ_i^{t+1} and the traffic budget B_i^{t+1}
14: Return the final model ω_i^T

represents the probability of worker i and j exchanging models in round t. $\mathcal{Z}_i^t (\subset \mathcal{N}_i)$ is the set of neighbors that engage in model exchanges with worker i in round t. Notably, the model update rule in Eq. (3) assigns higher weights to the pulled models from neighbors with lower communication probabilities, which enables worker to maintain enough information from these neighbors [12].

2.3 Communication-Efficient DFL

This section introduces the proposed communication-efficient DFL (CEDFL) mechanism for edge computing. In CEDFL, a worker communicates (or exchanges model parameters) with each of its neighbors with a probability, instead of communicating with all its neighbors as in [22], [24], [29]. Our proposed mechanism is mainly composed of two parts, including Alg. 1 at the coordinator and Alg. 2 at the worker.

The Coordinator Side: In order to estimate the network condition and determine the communication policy among workers, a coordinator periodically collects the differences of data distribution and traffic budgets of the workers, described in Alg. 1. At the beginning, the coordinator will initialize the parameters, such as the matrix of data distribution difference $\Theta^0 = [0]$, and the traffic budgets of the workers $\mathbf{B}$ (Line 1). In each global round t, the coordinator first sends an request to the workers for information collection (Line 3). Upon updating Θ and $\mathbf{B}$, the coordinator proceeds to calculate the policy $\mathbf{P}^t$ and the parameter ρ^t. This is achieved through the implementation of the CPG-DFL algorithm, as detailed in Section 3.2 (Lines 4-5). Through adaptive tuning, both the policy $\mathbf{P}^t$ and parameter ρ^t are optimized to facilitate swift convergence of the entire training procedure. Consequently, the coordinator generates a random matrix $\mathcal{Q} = \{q_{i,j}\}_{1\le i,j\le m}$ with the uniform distribution to facilitate the construction of communication matrix $\mathcal{Z}^t = \{z_{i,j}^t\}_{1\le i,j\le m}$, where $z_{i,j} \in \{0, 1\}$. Specifically, if $q_{i,j} \le p_{i,j}$, we set $z_{i,j} = 0$, which means that worker i and worker j will exchange their models in the upcoming aggregation epoch, otherwise $z_{i,j} = 0$. Notably, due to the bidirectional nature of model exchange, we control that matrices $\mathbf{P}$, $\mathcal{Q}$, and $\mathcal{Z}$ are all symmetric. Finally, the coordinator computes the Laplacian matrix of the communication matrix $\mathcal{Z}^t$, denoted as $\mathbf{L}_z$, to check its connectivity. The process of constructing matrix $\mathcal{Z}^t$ (Lines 7-9) will be repeated with different random seeds until the topology is connected (*i.e.*, $\lambda_2(\mathbf{L}_z) > 0$). Finally, the coordinator sends the neighbors set $\mathcal{Z}_i^t = \{j | z_{i,j} = 1\}$ and parameter ρ^t to each worker i for the following training process (Line 12).

The Worker Side: Each worker performs model training with the SGD algorithm on its local data and communicates with its neighbors according to the policy $\mathbf{P}$. The detailed algorithm is depicted in Alg. 2. First, the worker i initializes some parameters, such as local model ω_i^0, parameter ρ^0, and learning rate η (Lines 1-2). In each global round t, the communication probabilities $\mathbf{P}_i^t$, the neighbors set $\mathcal{Z}_i^t$, and the parameter ρ^t will be updated if the worker has received the latest policy from the coordinator (Lines 4-5). Next, the worker executes the SGD algorithm for ϵ times to update its model (Lines 6-8). After that, the worker will exchange models with its neighbors and aggregate the received models for model update (Lines 10-12). Then, the information of worker i will be updated, including Θ_i and B_i. We employ weight divergence to quantify the disparity in data distributions. Specifically, in round t, worker i updates its measure of data distribution divergence with its neighbor j as follows:

$$\theta_{i,j} = \left\|\omega_i^t - \omega_j^t\right\| / \left\|\omega_i^t\right\| \tag{4}$$

where $j \in \mathcal{Z}_i^t$. Moreover, let b denote the traffic consumption for model exchanging with a neighbor. The traffic budget of worker i is updated as $B_i^{t+1} \leftarrow B_i^t - b * |\mathcal{Z}_i^t|$. After T rounds, each worker i will obtain its final model ω_i^T (Line 14).

For a better explanation of CEDFL, we illustrate the model training based on a randomly constructed basic topology (BAT scheme), Ring scheme and CEDFL in Fig. 2. Assume that there are six workers in the network. We train the VGG9 model [30] over the CIFAR10 dataset [31]. The bandwidth of each link is the same, and the data distribution among the workers is non-IID.

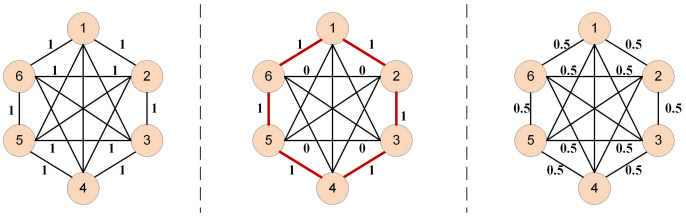


Fig. 2: Illustration of proposed CEDFL mechanism. There are six workers in the network. *Left plot:* BAT scheme; *middle plot:* Ring scheme; *right plot:* CEDFL.

We conduct two sets of experiments given a target accuracy and a certain number of model exchanges. Table 3 shows the resource cost and training performance by the three different schemes. In BAT, each worker communicates with all its neighbors at each round. That is, the probability of each link is 1.0. In order to make the expectation of traffic consumption equal to or even less than that of Ring, CEDFL assigns the probability of each link as 0.5 in the network, *i.e.*, each worker has a chance of 50% for communicating with each neighbor at every round. If a random probability is smaller than 0.5, the link will be selected, otherwise, it will be discarded, which will be explained in Section 3.2. As shown in Table 3, given the target accuracy of 60%, the total number of model exchanges of CEDFL is 958, while that of BAT and Ring is 1,640 and 4,200, respectively, in the first set of tests. In other words, CEDFL can reduce the number of model exchanges (or traffic consumption) by about 41.5% and 77.2% compared with BAT and Ring, respectively. In the second set of tests, CEDFL can improve the test accuracy by about 6.5% and 24.9% compared with BAT and Ring, after 2,000 model exchanges, respectively. The testing results show that CEDFL with a fixed probability of 0.5 can efficiently improve the test accuracy and reduce traffic consumption compared with the two benchmarks.

In practice, due to the limited network traffic, heterogeneous communication and different data distributions among workers, it is necessary to dynamically adjust the probability of each link, so as to achieve efficient training. The impact of probability assignment is striking on federated training, which will be introduced in Section 3.1. In CEDFL, the optimal probability assignment depends on the traffic budgets and differences of data distributions among the workers (Section 3.2). Thus, our proposed CEDFL mechanism can efficiently alleviate the training performance degradation caused by system heterogeneity and non-IID data under traffic constraints. Besides, CEDFL can be directly compatible with Differential Privacy (DP) techniques. To protect the privacy of each worker, DP ensures that the exchanged model updates or gradients contain carefully calibrated noise, making it statistically difficult to infer individual contributions while preserving the global learning utility.

2.4 Convergence Analysis

In this section, we will discuss the convergence analysis of our CEDFL algorithm and demonstrate that the models on the workers will eventually converge to the same optima. Prior to providing formal proof, we will begin by stating some assumptions, as outlined in [12].

TABLE 3: Resource cost and model accuracy of different schemes.

Schemes	Fixed Target Accuracy		Fixed Number of Exchanges	
	Model Exchanges	Test Accuracy	Model Exchanges	Test Accuracy
BAT	1640	60.3%	2030	64.3%
Ring	4200	60.4%	2040	45.9%
CEDFL	958	60.1%	2028	70.8%

Assumption 1. ***(Lipschitzian Gradient)*** *The loss function $F(\omega,\xi)$ and its local gradient $\nabla F(\omega,\xi)$ are L-Lipschitz: $\|\nabla F(x)-\nabla F(y)\|_2 \le L\|x-y\|_2, \forall x,y\in\mathcal{R}^d$.*

Assumption 2. ***(Unbiased Local Gradient Estimator)*** *Let ξ_i^t denote the local data sample on worker i at round t. Then, the local gradient estimator is unbiased as follows:*

$$\mathbb{E}\left[\nabla F\left(\omega_i^t,\xi_i^t\right)\right]=\nabla F\left(\omega_i^t\right). \tag{5}$$

Assumption 3. ***(Additive Noise)*** *Each worker i experiences additive noise π_i in its stochastic gradients, resulting from stochastic sampling of the dataset. This noise has a mean of zero, $\mathbb{E}[\pi_i]=0$, and its variance is bounded, such that $\mathbb{E}[\pi_i^{\mathbb{T}}\pi_i]\le\varphi^2$.*

We first prove that our proposed framework CEDFL can converge to a small domain if the following assumption holds.

Assumption 4. *If μ is a positive number, the loss function F_i satisfies μ-strong convexity if for any pair of points x and y, the following inequality holds:*

$$F(y)-F(x)\ge\langle\nabla F(x),y-x\rangle+\frac{\mu}{2}\|y-x\|^2. \tag{6}$$

Remark 1. *Assuming that Assumption 1 holds, and using the properties of convex functions, we can conclude that for any pair of points $x,y\in\mathbb{R}^d$, the following inequality holds:*

$$\begin{aligned}&(\nabla F(x)-\nabla F(y))^{\mathbb{T}}(x-y)\\ &\ge\frac{\mu L}{\mu+L}(x-y)^{\mathbb{T}}(x-y)\\ &\quad+\frac{1}{\mu+L}(\nabla F(x)-\nabla F(y))^{\mathbb{T}}(\nabla F(x)-\nabla F(y))\end{aligned} \tag{7}$$

According to Alg. 2, at each global round t, the model update of each worker can be viewed as two sequential steps. In the first step, worker i performs the local training epochs for ϵ times using the SGD algorithm (Lines 6-8 in Alg. 2):

$$\omega_i^{t+\frac{1}{2}}=\omega_i^t-\eta g_i^t \tag{8}$$

where $g_i^t=\sum_{k=1}^{\epsilon}\nabla F(\omega_i^{t,k};\xi_i^{t,k})$. For simplicity, let $\tau_{i,j}^t=\frac{a_{i,j}}{p_{i,j}^t}$. In the second step, worker i updates the local model using its neighbors' models (Lines 10-12 in Alg. 2):

$$\begin{aligned}\omega_i^{t+1}&=\omega_i^{t+\frac{1}{2}}-\eta\rho\sum_{j\in\mathcal{M}}\tau_{i,j}^t(\omega_i^{t+\frac{1}{2}}-\omega_j^{t+\frac{1}{2}}))\\ &=\omega_i^t-\eta g_i^t-\eta\rho\sum_{j\in\mathcal{M}}\tau_{i,j}^t(\omega_i^t-\eta g_i^t-\omega_j^t+\eta g_i^t)\\ &=(1-\eta\rho\sum_{j\in\mathcal{M}}\tau_{i,j}^t)(\omega_i^t-\eta g_i^t)-\eta\rho\sum_{j\in\mathcal{M}}\tau_{i,j}^t(\omega_j^t-\eta g_j^t)\end{aligned} \tag{9}$$

Let $\Omega^t=[\omega_1^t,\omega_2^t,\cdots,\omega_m^t]^{\mathbb{T}}$ and $\boldsymbol{g}^t=[g_1^t,g_2^t,\cdots,g_m^t]^{\mathbb{T}}$ denote the vectors containing all workers' models and gradients at round

t, respectively. Then, Eq. (9) can be rewritten as the following matrix form:

$$\Omega^{t+1} = \mathbf{W}^t(\Omega^t - \eta \boldsymbol{g}^t) \tag{10}$$

where $\mathbf{W}^t$ is an $m \times m$ matrix expressed as:

$$\mathbf{W}^t = \boldsymbol{I} + \eta\rho\mathbf{U}^t \tag{11}$$

where $\mathbf{U}^t = \{u_{i,j}^t, 1 \le i,j \le m\}$. Specifically, $u_{i,j}^t = \tau_{i,j}^t$ for any $i \neq j$, while $u_{i,i}^t = -\sum_{j\in\mathcal{M}} \tau_{i,j}^t$. Thus, the matrix $\mathbf{W}^t$ mainly depends on the communication policy $\mathbf{P}^t$. By applying Eq. (11), the expectation is derived *w.r.t.* the matrix $\mathbf{W}^t$ as follows:

$$\begin{aligned}\mathbf{Y}^t &= \mathbb{E}\left[(\mathbf{W}^t)^{\mathbb{T}}\mathbf{W}^t\right] \\ &= \mathbb{E}\left[\boldsymbol{I} + \eta\rho(\mathbf{U}^t)^{\mathbb{T}} + \eta\rho\mathbf{U}^t + \eta^2\rho^2(\mathbf{U}^t)^{\mathbb{T}}\mathbf{U}^t\right] \quad (12)\\ &= [y_{i,j}^t]_{m\times m} \quad (13)\end{aligned}$$

where

$$\begin{cases} y_{i,i}^t = & 1 - 2\eta\rho\sum_{j\in\mathcal{M}}\tau_{i,j}^t \\ & +\eta^2\rho^2\left[(\sum_{j\in\mathcal{M}}\tau_{i,j}^t)^2 + \sum_{j\in\mathcal{M}}(\tau_{i,j}^t)^2\right], \forall i \in \mathcal{M} \\ y_{i,j}^t = & 2\eta\rho\tau_{i,j}^t + \sum_{k\in\mathcal{M}-\{i,j\}}\tau_{i,k}^t\tau_{j,k}^t \\ & -\eta^2\rho^2\tau_{i,j}^t\sum_{k\in\mathcal{M}}(\tau_{i,k}^t + \tau_{j,k}^t), \qquad \forall i \neq j \end{cases} \tag{14}$$

We denote the largest and second-largest eigenvalues of the matrix $\mathbf{Y}^t$ as $\alpha_1(\mathbf{Y}^t)$ and $\alpha_2(\mathbf{Y}^t)$, respectively. If $\mathbf{Y}^t$ is a doubly stochastic symmetric matrix, we set $\alpha^t = \alpha_2(\mathbf{Y}^t)$, otherwise we set $\alpha^t = \alpha_1(\mathbf{Y}^t)$. It should be noted that there is a situation in which a worker does not communicate with any of its neighbors in a round, resulting in unconnected communication. However, the model training can achieve convergence if the undirected graph $\mathcal{G}$ is connected [12]. Then we can derive the following theorem.

Theorem 2. *Let ω^* denote a minimizer of the global loss function F. Combining the proposed assumptions and the remarks, when* CEDFL *is performed in the form of Eq.* (10) *with the learning rate $0 < \eta \le \frac{2}{\mu+L}$ and $\alpha < 1$, we have*

$$\mathbb{E}[\|\Omega^T - \boldsymbol{w}^*\|^2] \le \alpha^T\|\Omega^0 - \boldsymbol{w}^*\|^2 + \frac{\alpha}{1-\alpha}\eta^2\varphi^2 \tag{15}$$

and CEDFL *will converge to a small domain after T rounds, where $\boldsymbol{w}^* = \mathbf{1}(\omega^*)^{\mathbb{T}}$ and $\|\cdot\|$ denotes the spectral norm. The detail proof is presented in APPENDIX A.*

Then, we further conduct theoretical convergence analysis of CEDFL for non-convex loss functions.

Lemma 3. *Let $\eta \le \frac{1}{4L\epsilon}$. We have the following expression:*

$$\begin{aligned}\mathbb{E}f(\overline{\omega}^{t+1}) \le f(\overline{\omega}^t) &- \frac{\eta\epsilon}{4}\left\|\nabla f(\overline{\omega}^t)\right\|_2^2 \\ &+ \frac{\eta L^2\epsilon}{m}\sum_{i=1}^m\left\|\overline{\omega}^t - \omega_i^t\right\|_2^2 + \frac{\sigma^2\eta^2\epsilon^2 L}{m}\end{aligned} \tag{16}$$

where ϵ is the local update frequency of the worker. The proof is presented in APPENDIX B.

Remark 4. *Summing up for T rounds, and rearranging the terms in Eq.* (16)*, we can derive that:*

$$\begin{aligned}\frac{1}{T}\sum_{t=1}^T\left\|\nabla f(\overline{\omega}^t)\right\|_2^2 &\le \frac{4*(f(\overline{\omega}^1) - f(\overline{\omega}^*))}{\eta\epsilon T} \\ &+ \frac{4L^2}{mT}\sum_{t=1}^T\sum_{i=1}^m\left\|\overline{\omega}^t - \omega_i^t\right\|_2^2 + \frac{4L\eta\epsilon\sigma^2}{m}\end{aligned} \tag{17}$$

Lemma 5. *Let $\frac{27L\eta^2}{(1-\alpha)^2} < 1$. We have the following formulation:*

$$\begin{aligned}\sum_{t=1}^T\sum_{i=1}^m\mathbb{E}\left\|\overline{\omega}^t - \omega_i^t\right\|_2^2 &\le \frac{2m\eta^2(\sigma^2 + 3\zeta^2)T}{(1-\alpha)^2 - 3\eta^2L^2} \\ &+ \frac{6m\eta^2}{(1-\alpha)^2 - 3\eta^2L^2}\sum_{t=1}^T\mathbb{E}\left\|\nabla f(\overline{\omega}^t)\right\|_2^2\end{aligned} \tag{18}$$

The proof is presented in APPENDIX C.

Remark 6. *Inserting Eq.* (18) *into Eq.* (17)*, we obtain the following convergence bound:*

$$\begin{aligned}\frac{1}{T}\sum_{t=1}^T\left\|\nabla f(\overline{\omega}^t)\right\|_2^2 &\le \frac{4(f(\overline{\omega}^1) - f(\overline{\omega}^*))((1-\alpha)^2 - 3\eta^2L^2)}{\eta\epsilon T((1-\alpha)^2 - 27\eta^2L^2)} \\ &+ \frac{8L^2\eta^2(\sigma^2 + 3\zeta^2)}{(1-\alpha)^2 - 27\eta^2L^2} + \frac{(1-\alpha)^2 - 3\eta^2L^2}{(1-\alpha)^2 - 27\eta^2L^2}\frac{4L\eta\epsilon\sigma^2}{m}\end{aligned} \tag{19}$$

Accordingly, the communication topology weight matrix $\mathbf{W}$ (reflected by α) has significant impact on the convergence bound with Eq. (19). With the increasing of topology sparsity (*i.e.*, large α), the above convergence bound will increase. Thus, the convergence bound is closely related to the communication probability both under convex and non-convex cases, which will be well determined by our proposed algorithm in Section 3.

Discussion: NetMax [12] is the method most closely related to ours, and is also founded on the probabilistic communication technique. However, the convergence analysis of NetMax is predicated on the assumption that the loss function is strongly convex. As a result, it is inapplicable to the DFL training of non-convex models, such as deep neural networks (DNNs). Moreover, NetMax fails to take into account the heterogeneous data distribution among workers, which does not align with the actual circumstances of edge networks. Different from NetMax, our convergence analysis does not rely on the strong convexity assumption.

2.5 Problem Formulation

In this section, we give the definition of resource-efficient decentralized federated learning with probabilistic communication (DFL-PC) problem. Given a fixed topology, the traffic consumption of transmitted models will be further reduced through probabilistic communication between the workers. However, an identical probability assignment for all links cannot fully utilize the available bandwidth because of the different bandwidth and heterogeneous data among workers. Therefore, we formalize DFL-PC as an optimization problem to determine link probabilities that enhance DFL training performance under bandwidth constraints.

$$\min_{\mathbf{P}^t, \rho^t(t\in[T])} f(\Omega^T)$$

$$s.t. \begin{cases} \alpha^T \le \varepsilon, & \forall \varepsilon > 0 \\ \sum_{t=1}^T\sum_{j=1}^m p_{i,j}^t b \le B_i, & \forall i \in \mathcal{M} \\ 1 - p_{i,j}^t \le \beta\theta_{i,j}^t\rho, & \forall i,j \in \mathcal{M}, \forall t \in [T] \\ p_{i,j}^t \in [0,1], & \forall i,j \in \mathcal{M}, \forall t \in [T] \end{cases} \tag{20}$$

According to Theorem 2, the first inequality represents that the models of the workers will converge to a minimal domain near the optimal model if α^T is less than a small positive value ε. The second set of inequalities ensures that each worker adheres

to the traffic constraints. The third set of inequalities ensure that neighbors with greater data distribution discrepancies are assigned higher communication probabilities, thereby enhancing training performance with the non-IID data, where β is a pre-defined hyper-parameter. The fourth set of inequalities specifies the range of values for the communication probabilities. Our goal is to minimize the final loss function $f(\Omega^T)$.

3 Algorithm Design

3.1 Motivation for Algorithm Design

The federated training performance under the IID setting is always better than that under non-IID setting, such as classification accuracy [17]. The DFL algorithm [9] favors exchanging models or gradients between workers that have distinct data distributions to expedite model training convergence and enhance training performance. To this end, different probabilities will be assigned for links in the basic topology. Specifically, the communication probability $p_{i,j}^t$ will be larger if the data distribution between workers i and j is significantly different at round t, *i.e.*, the higher model exchange probability.

In order to motivate our algorithm design, we give an example in Fig. 3. We divide all the workers into three groups, in which the workers in each group have the same data labels, *i.e.*, IID data, and the workers in different groups have different data labels, *i.e.*, non-IID data. We adopt the classical neural model (*e.g.*, VGG9 [30]) and dataset (*e.g.*, CIFAR10 [31]) in our test, and perform 200 training rounds by default. We observe the test accuracy performance and the number of model exchanges under three different communication schemes, including Ring, BAT and CedFL. Let the pair $\mathbf{P}_I$-$\mathbf{P}_O$ to denote the probability of intra (inter) link between the workers in the same (different) group(s) in CedFL. For example, CedFL ($\mathbf{P}_I$:0.1-$\mathbf{P}_O$:1.0) denotes that the probabilities for intra and inter links are 0.1 and 1.0, respectively. The results in Figs. 4-5 show that CedFL can achieve better training performance with less resource consumption compared with the other two benchmarks. For instance, given 200 training rounds, the test accuracy of CedFL (0.5-0.5) is very close to that of BAT, *i.e.*, 69.6% and 70.3%, respectively. However, the number of model exchanges of BAT is 4,000, while that of CedFL (0.5-0.5) is only 2,086. In other words, compared with BAT, CedFL can reduce the resource consumption by about 47.9%.

Besides, we test the impact of different probability assignments on the performance of model training, *e.g.*, CedFL (0.1-1.0) and CedFL (0.5-0.5). In Figs. 4-5, the number of model exchanges by CedFL (0.5-0.5) and (0.1-1.0) are almost the same, 2086 and 2158, respectively. However, the test accuracy of CedFL (0.5-0.5) is about 69.6%, while that of CedFL (0.1-1.0) is increased to 73.4%. The frequent model exchange has little significance for performance improvement if the data distribution of the two workers is consistent. Thus, the algorithm will allocate a smaller probability to reduce traffic consumption. It is challenging to determine the optimal communication policy to accelerate the convergence of the model training with less resource cost.

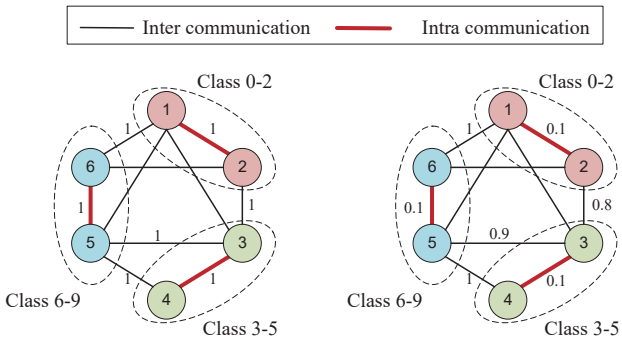


Fig. 3: Illustration of the CedFL mechanism. The **thick links** indicate the intra communication betwen the workers with the same data distribution, while the other links indicate the inter communication between the workers with different data distributions.

3.2 Algorithm Description

In this section, we propose an efficient algorithm for probability assignment under resource constraints, considering the data distributions among workers. Since the search space (*e.g.*, the numerous optional probabilities for the communication links) of DFL-PC is very large, it is time-consuming to search for the optimal solution by the stand method. Intuitively, the worker may expect to communicate with its neighbors as much as possible to speed up the training procedure, *i.e.*, $\max \sum_{j\in\mathcal{N}_i} p_{i,j}^t, \forall i,t$, considering the differences of their data distributions. If there are significant differences of data distribution between two workers, we will assign a larger communication probability to this link, *i.e.*, the higher model exchange frequency between two workers, vice versa. Instead of seeking the optimal solution, we suggest an efficient approach to discover a practical solution with sub-optimal performance. Our observation is that given the parameters (*e.g.*, ρ), we believe that a feasible communication policy $\mathbf{P}$ can minimize the objective loss value in DFL-PC. Therefore, our primary strategy is to explore a policy $\mathbf{P}$ with the smallest loss value while taking the differences of data distributions and resource constraints into considerations. In order to reduce the search space, we explore an alternative problem with some shared constraints of DFL-PC, which is formulated as:

$$\max \sum_{t=1}^{T}\sum_{i=1}^{m}\sum_{j\in\mathcal{N}_i} p_{i,j}^t$$

$$s.t.\begin{cases} \sum_{t=1}^{T}\sum_{j=1}^{m} p_{i,j}^t \cdot b_i \le B_i & \forall i \in \mathcal{M} \\ 1 - p_{i,j}^t \le \beta\theta_{i,j}^t\rho, & \forall i,j \in \mathcal{M}, \forall t \in [T] \\ p_{i,j}^t \in [0,1], & \forall i,j \in \mathcal{M}, \forall t \in [T] \end{cases} \quad (21)$$

In order to quickly find the feasible policy, the objective of Eq. (21) can be rewritten as $\min \sum_{t=1}^{T}\sum_{i\in\mathcal{M}} p_{i,i}^t$, which aims to minimize the probabilities of workers selecting themselves for communications, *i.e.*, local model training.

Alg. 3 depicts the process of generating the communication policy for the DFL algorithm, known as CPG-DFL. This policy is composed of two nested loops, where the first one is responsible for determining the number of search rounds, denoted by R, while the second loop sets the number of evaluation rounds, represented by $\widehat{T}$, which are initialized at the beginning of the algorithm, and will be adjusted according to the network conditions. Different

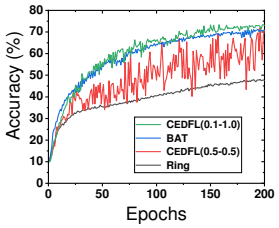


Fig. 4: Test accuracy under four different communication schemes.

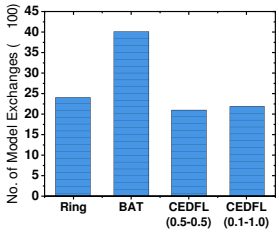


Fig. 5: Number of model exchanges under four different communication schemes.

from T, which may reach thousands or even more, $\widehat{T}$ (one or several epochs) is much less than T. The traffic budgets and the differences in data distribution among the workers, which are collected at the coordinator, serve as the inputs of the algorithm (Line 1). In each search round $r \in \{1, 2, ..., R\}$, we set ρ as $\frac{1}{2\eta r}$ and call the subfunction SEARCHPOLICY to determine the policy $\mathbf{P}$ (Line 3-5). In the subfunction, the policy matrix $\mathbf{P}$ will be obtained by solving the Nonlinear Programming (NLP) problem (Line 9). Then, we adjust the obtained communication policy by setting the probability of link $a_{i,j}$ (*i.e.*, $p_{i,j}^t$) to 0 when the traffic budget of any worker i or j is exhausted or there is no data distribution difference between worker i and j. To evaluate the effects of the current policy $\mathbf{P}$ and parameter ρ, the DFL will be performed for $\widehat{T}$ rounds according to policy $\mathbf{P}$ and models Ω^t using Algs. 1-2 (Line 12-13). Since the training rounds $\widehat{T}$ is much smaller than the total rounds T, the resource costs for policy evaluation can be ignored. After $\widehat{T}$ evaluation rounds, the subfunction returns the matrix $\mathbf{P}$ and the loss value $f(\Omega^{t+\widehat{T}})$ (Line 16). Finally, the policy matrix $\mathbf{P}$ with minimum loss value $f(\Omega^{t+\widehat{T}})$ will be greedily selected (Line 6-7). A feasible policy $\mathbf{P}$ which can achieve the smallest objective loss value has been found through CPG-DFL.

It should be noted that some links may not be selected for several consecutive rounds due to their low probabilities. To ensure the performance of model training, we set a hard threshold (*e.g.*, 5). When a link has not been selected for model exchanging during a threshold number of rounds, the link's probability will be multiplied by a coefficient (*e.g.*, 1.5) at each following round. Once this link has been selected, its probability will be reset to the original value.

Algorithm 3 Communication Policy Generation (CPG) for DFL

1: Initialize learning rate η, search rounds R, evaluation rounds $\widehat{T}$, remaining traffic budgets $\mathbf{B}^t = \{B_1^t, ..., B_m^t\}$, differences of data distribution Θ^t
2: **function** POLICYGENERATION($\eta, R, \widehat{T}, \mathbf{B}^t, \Theta^t$)
3: **for** $r \in \{1, 2, ..., R\}$ **do**
4: $\rho \leftarrow \frac{1}{2\eta r}$
5: $(\mathbf{P}, f(\Omega^{t+\widehat{T}})) \leftarrow$ SEARCHPOLICY($\eta, \rho, \widehat{T}, \mathbf{B}^t, \Theta^t$)
6: Find the item $(\mathbf{P}, \rho)$ with the minimum loss $f(\Omega^{t+\widehat{T}})$
7: Return the probabilities matrix $\mathbf{P}$ and parameter ρ
8: **function** SEARCHPOLICY($\eta, \rho, \widehat{T}, \mathbf{B}^t, \Theta^t$)
9: Solve NLP problem in (21) to obtain the policy $\mathbf{P}$
10: **if** $\min\{B_i^t, B_j^t\} \leq 0$ or $\theta_{i,j}^t = 0 (\forall i, j \in \mathcal{M})$ **then**
11: $p_{i,j}^t = 0$
12: **for** $k \in \{1, 2, ..., \widehat{T}\}$ **do**
13: Perform DFL training with Ω^t by Algs. 1-2
14: Compute the loss value $f(\Omega^{t+\widehat{T}})$ in Eq. (1)
15: Update the traffic budget $\mathbf{B}^t$
16: Return the matrix $\mathbf{P}$ and the loss value $f(\omega)$

4 PERFORMANCE EVALUATION

4.1 Performance Metrics and Benchmarks

We use the following metrics to evaluate the performance of CEDFL and the baselines:

- *Training loss* measures whether an FL algorithm can effectively achieve convergence.
- At the end of each round, we evaluate the global model on the test dataset and record the *test accuracy*.
- *Bandwidth consumption* quantifies the communication cost of training, and it is calculated as the total size of the model transmitted through the network during model aggregation.
- *Completion time* measures the time taken for training to terminate, which reflects the training speed.

We adopt three typical FL schemes, *i.e.*, Ring [9], BAT [29] and NetMax [12], as benchmarks for performance comparison. Since the ring topology is very sparse and communication-efficient, it has been widely adopted for decentralized model training [9]. In the BAT scheme [29], each worker exchanges models with all its neighbors, which can improve training performance after model aggregation but may consume more resources. In NetMax [12], the high-speed link will be selected with a higher probability for communications among the workers. We implement the benchmarks and our proposed algorithm on a fixed P2P network topology [12].

4.2 Models and Datasets

To perform the image classification tasks, we select two well-known deep learning models, VGG9 [30] and ResNet9 [32], which have different structures and parameters. For optimization, we utilized the SGD optimizer with a momentum of 0.9. The learning rates for VGG9 and ResNet9 were initialized to 0.05 and 0.1, respectively, and their learning rates decayed correspondingly at rates of 0.98 and 0.99 [33]. We conduct all experiments using a batch size of 64 and by default set the number of local updates in each round to 1 for both models.

To evaluate our proposed algorithm, we adopt two classical datasets, namely CIFAR10 (referred to as C10) and CIFAR100 (referred to as C100) [31], during the training process. C10 contains 60,000 32×32 color images labeled in 10 classes, with 50,000 samples for training and 10,000 for testing. On the other hand, C100 has the same total number of image samples as C10 but is more challenging to train models for classification as it consists of 100 classes. Specifically, we trained VGG9 on C10 and ResNet9 on C100.

TABLE 4: Test accuracy of different models trained with BAT, Ring, NetMax and CEDFL under different data settings.

Schemes	VGG9 over CIFAR10		ResNet9 over CIFAR100	
	IID	non-IID	IID	non-IID
BAT	78.3%	72.6%	52.7%	46.1%
Ring	72.4%	67.2%	48.3%	41.5%
NetMax	77.8%	68.4%	50.6%	42.4%
CEDFL	83.6%	77.4%	59.6%	54.8%

4.3 Simulation Evaluation

4.3.1 Evaluation Settings

We perform the simulations using an AMAX deep learning workstation, which is equipped with an Intel(R) Core(TM) i9-10900X CPU, 4 NVIDIA GeForce RTX 2080Ti GPUs, and 256 GB RAM. To simulate a decentralized federated learning edge computing system, we utilize 30 workers, with each implemented as a process in the system, along with a coordinator responsible for recording training performance and adjusting the probability of each link in the network. The PyTorch[1] framework is used for model training on each worker, and the communication between workers is established using the socket library in Python.

System Configuration. To simulate communication in our decentralized federated learning system, we consider a scenario where each worker communicates with its neighbors through either LANs or WANs. To reflect the heterogeneity and dynamics of P2P networks in our simulations, we allow the inbound bandwidth of each worker to fluctuate between 1Mb/s and 20Mb/s. Considering that the outbound bandwidth in typical WANs is usually smaller than the inbound bandwidth [17], we configure it to fluctuate between 0.5Mb/s and 10Mb/s.

In terms of computation, we conduct extensive tests on several commercial devices (*e.g.*, laptops, TX2[2]) with background applications. We measure the running time of a single local update (with a batch size of 32) for VGG9 on C10 to vary between 0.05s and 0.15s, while that for ResNet9 on C100 varies between 0.15s and 0.25s [33]. Based on these measurements, we dynamically and randomly assign running time within the ranges for different workers while conducting local updates on different learning tasks.

4.3.2 Simulation Results

We conduct three sets of simulations to validate the effectiveness of our proposed approach, and the results of the simulations are presented below:

IID vs. Non-IID: Table 4 presents the test accuracy performance of two models trained separately on IID and non-IID local data using BAT, Ring, NetMax, and CEDFL, with a fixed number of model exchanges (*e.g.*, 3,000). Our focus is on comparing the four schemes, rather than achieving state-of-the-art performances. From Table 4, we observe that models trained with all schemes perform better under IID settings than under non-IID settings. Additionally, CEDFL outperforms the other three benchmarks by training for more rounds, which enhances the performance of federated training. For instance, when training ResNet9 on C100 under non-IID settings, the accuracy of CEDFL is approximately 54.8%, compared to 41.5%, 46.1%, and 42.4% achieved by Ring, BAT, and NetMax, respectively. In other words, CEDFL can improve accuracy by about 13.3%, 8.7%, and 12.4% over the three benchmarks, respectively.

[1]https://pytorch.org/

[2]https://www.nvidia.com/en-us/autonomous-machines/embedded-systems/

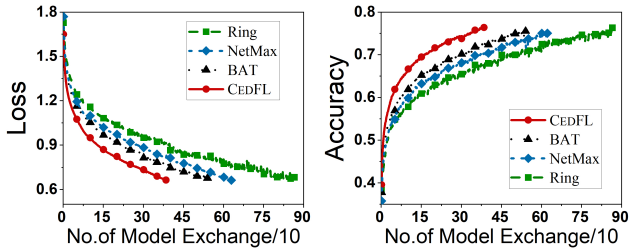


Fig. 6: Training performance of loss and accuracy with VGG9 trained over CIFAR10.

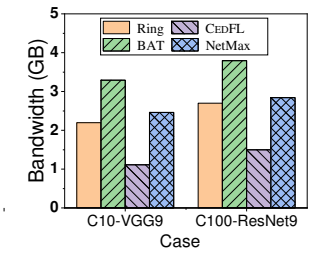


Fig. 7: Bandwidth consumption of two models trained with four different schemes.

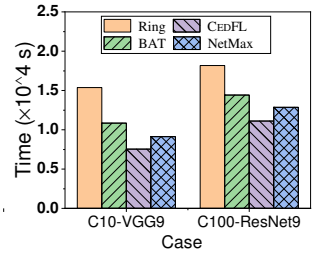


Fig. 8: Completion time of two models trained with four different schemes.

Convergence Performance: In the second set of simulations, we evaluate the training performance of four schemes under the non-IID setting, with a fixed accuracy requirement (*e.g.*, 75%). CEDFL effectively reduces the impact of non-IID issues without performing unnecessary model exchanges among workers with similar data distribution, thereby accelerating the model training process. As depicted in Fig. 6, the required number of model exchanges for CEDFL is lower than that of the other three benchmarks. For a training accuracy of 75%, CEDFL requires 372 model exchanges, while BAT, Ring, and NetMax require 547, 865, and 622, respectively. In other words, CEDFL can reduce the required number of model exchanges by approximately 31.9%, 56.9%, and 44.2% compared to BAT, Ring, and NetMax, respectively.

Resource Consumption: We conduct tests on the resource consumption of four different schemes, all with a fixed accuracy requirement of 75%. Specifically, during the model training process on simulated network topologies, the worker in CEDFL performs local updates without any performance degradation instead of exchanging models with its neighbors. Our findings, presented in Figs. 7-8, demonstrate that the bandwidth consumption and completion time of model training with CEDFL are significantly lower than those of the other three benchmarks. For example, the bandwidth consumption of C100 trained with CEDFL is approximately 1.5GB, which is 42.3%, 60.5%, and 48.2% lower

Fig. 9: The test-bed platform.

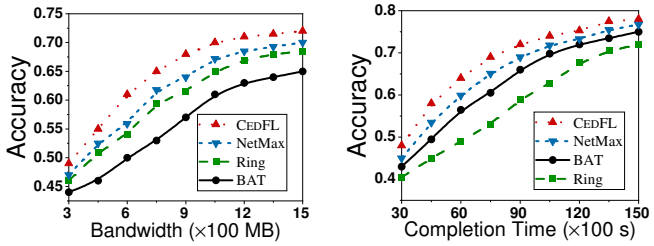


Fig. 10: Test accuracy with bandwidth and completion time constraints (ResNet9 trained over CIFAR100) in Test-bed.

than Ring, BAT, and NetMax, respectively, whose bandwidth consumption is around 2.6GB, 3.8GB, and 2.9GB, respectively. Furthermore, the Ring scheme requires more rounds to achieve the same test accuracy, resulting in a longer completion time. For example, VGG9 trained over C10 requires approximately 30.4%, 50.9%, and 17.3% less completion time with CEDFL compared to BAT, Ring, and NetMax, respectively. Therefore, our proposed CEDFL mechanism effectively reduces the resource cost of federated training.

4.4 Test-bed Evaluation

4.4.1 Implementation on the Platform

We carry out experiments on a real test-bed environment consisting of two main components: a deep learning workstation with four NVIDIA Titan RTX GPUs and 30 devices, comprising 15 NVIDIA Jetson TX2 and 15 NVIDIA Jetson Xavier NX. The workstation acts as the coordinator, responsible for making probability assignments and estimating the network condition. To simulate a real-world edge computing environment, we place the devices at different locations at least 2,000 meters apart and allowed them to communicate through a link with a bandwidth of approximately 50Mbps. Additionally, we logically connect these devices through the torch.distributed package. As the transmission delay between the devices and servers is significantly shorter in practical scenarios, we deploy the devices at various locations in a laboratory of approximately 400m^2 and enable them to communicate through high-speed 5GHz WiFi.

Data Distribution on Workers The performance of model training is significantly influenced by the different categories of data distribution, namely IID and non-IID. In our experiments,

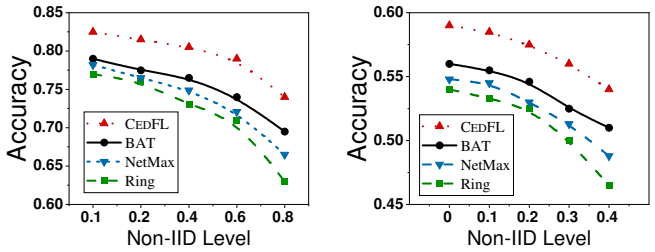


Fig. 11: Test accuracy under different Non-IID levels in Test-bed. *Left*: VGG9 over CIFAR10; *right*: ResNet9 over CIFAR100.

we examine the impact of data distributions on model training using five different cases, including IID data and four levels of non-IID data. For C10, we assign each worker a unique class in 10 classes, with $p\%$ ($p = 10, 20, 40, 60$ and 80) of the samples belonging to that class, while the remaining samples of each class were partitioned uniformly to other workers. It is worth noting that the case of $p = 10$ corresponds to IID data distribution. We refer to these five different cases of data distributions as 0.1, 0.2, 0.4, 0.6, and 0.8. For C100, we remove p classes of data samples from each worker, where p can take on values of 0, 10, 20, 30, and 40. The samples of one class were distributed uniformly to only $(10 - p/10)$ workers. Specifically, when $p = 0$, the data distribution is uniform. We label the non-IID levels of C100 as 0, 0.1, 0.2, 0.3, and 0.4.

4.4.2 Testing Results

We perform two groups of experiments to evaluate the efficiency of our proposed algorithm.

Effect of Resource Constraints: The first set of experiments investigates the impact of resource constraints, such as network bandwidth and completion time, on the training performance of ResNet9 over C100. By utilizing model exchanges with proper probability assignment, CEDFL can effectively reduce both bandwidth consumption and completion time. As depicted in Fig. 10, the training performance, measured by test accuracy, of all schemes exhibits improvement with increasing resource budgets. Notably, CEDFL achieves better training performance than the other three benchmarks, with a significant margin. For example, when the bandwidth budget is 1GB in the left plot, the test accuracy of CEDFL is around 68.8%, while that of Ring, BAT, and NetMax is approximately 62.5%, 59.4%, and 63.6%, respectively. Hence, CEDFL can improve test accuracy by about 6.3%, 9.4%, and 5.2% compared to Ring, BAT, and NetMax, respectively.

In the right plot of Fig. 10, it can be observed that longer completion times for model training can significantly enhance the test accuracy of all schemes. Nonetheless, CEDFL outperforms the three benchmarks by achieving higher accuracy within the same completion time. For example, given a completion time of 9,000s, the accuracy of CEDFL is approximately 72.8%, whereas that of Ring, BAT, and NetMax is around 59.2%, 66.4%, and 68.6%, respectively. Consequently, CEDFL can boost the accuracy by about 13.6%, 6.4%, and 4.2% compared to the three benchmarks, respectively.

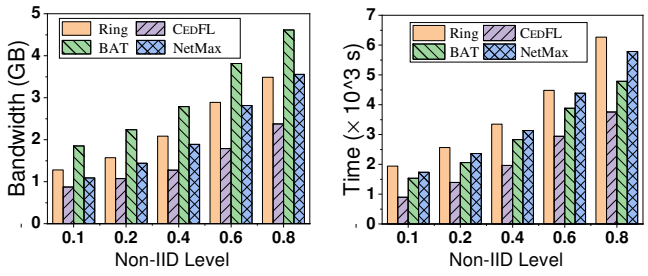


Fig. 12: Bandwidth consumption and completion time of VGG9 trained over CIFAR10 under different Non-IID levels in Test-bed.

Different Levels of Non-IID: The second set of experiments investigates the impact of different non-IID levels on the training performance of the four schemes with a fixed number (*e.g.*, 3,000) of model exchanges. Fig. 11 demonstrates that the test accuracy decreases with the increase of non-IID levels. However, conducting more model exchanges among workers with different data distributions can mitigate the non-IID issue. Hence, CEDFL achieves better performance of federated training than the other three benchmarks, particularly under a large non-IID level. For example, when training ResNet9 over C100 with a non-IID level of 0.4, the test accuracy of CEDFL is about 54.7%, while that of BAT, Ring, and NetMax is about 51.4%, 46.8%, and 48.5%, respectively. This indicates that CEDFL can improve the test accuracy by about 3.3%, 7.9%, and 6.2% compared to BAT, Ring, and NetMax, respectively.

We also evaluate the bandwidth consumption and completion time of training VGG9 on C10 dataset with fixed accuracy requirements at different levels of non-IID. As depicted in Fig. 12, both the bandwidth consumption and completion time of training increase with the non-IID level. Nevertheless, the increase in these metrics is comparatively slower in CEDFL than the other three benchmarks. In fact, CEDFL outperforms the other three benchmarks in terms of bandwidth consumption and completion time. For instance, at a non-IID level of 0.6, CEDFL consumes only 1.76GB of bandwidth, whereas Ring, BAT, and NetMax require 2.87GB, 3.91GB, and 2.74GB, respectively. Therefore, CEDFL reduces the bandwidth consumption of model training by approximately 38.6%, 54.9%, and 35.7% when compared to BAT, Ring, and NetMax, respectively.

To summarize, CEDFL substantially outperforms the three benchmarks. Firstly, our CEDFL can significantly alleviate non-IID issue under resource constraints and system heterogeneity. Therefore, CEDFL achieves faster convergence speed than three benchmarks under the non-IID setting. Secondly, CEDFL is superior in training performance (*e.g.*, accuracy) with less resource cost, *i.e.*, network bandwidth and completion time.

5 RELATED WORKS

The roots of decentralized optimization can be traced back to the seminal work by [34]. The fundamental principles of decentralized Machine Learning (ML) and Federated Learning (FL) are also based on decentralized optimization, which involves the practice of distributed optimization without any central entity to assist. In order to grasp the essential features of CEDFL in comparison to previous algorithms, it is necessary to provide a brief overview of the frameworks of decentralized optimization and communication-efficient distributed optimization.

To overcome the potential bottlenecks associated with parameter servers, researchers have proposed decentralized ML methods such as those described by [9], [35]. In these methods, model parameters are exchanged between any two edge nodes or workers in the network, rather than being aggregated and broadcast by a central server. Such methods are known as consensus-based decentralized optimization, which is derived from distributed averaging algorithms aimed at computing the mean of all data distributed across multiple workers. In general, a dense topology can lead to faster convergence in terms of training epochs [36]. However, this can result in network congestion and long communication times. Thus, designing a topology or decentralized optimization framework that achieves fast convergence becomes a critical challenge. For instance, Wang *et al.* [37] proposed decomposing the original topology into disjoint communication subgraphs and allocating different probabilities for these subgraphs, with each subgraph being selected during model training according to a certain probability. This approach can strike a balance between error and runtime in decentralized Stochastic Gradient Descent (SGD) by determining the appropriate communication probabilities. However, it mainly relies on link speeds for worker communication, which can result in poor training performance under non-IID data settings.

In large scale networks, the cost of communication often surpasses the computation time, leading to the need for communication-efficient distributed optimization techniques. To decrease the bandwidth usage per channel use, reducing the size of communication payloads is a popular solution that can be achieved through methods such as gradient quantization [38], model parameter quantization [39], and model output exchange via knowledge distillation for large-sized models [40]. To minimize the number of channel uses per communication round, model updates can be restricted only to workers whose computation delays are less than a target threshold [41] or to those workers whose updates differ sufficiently from the previous updates in terms of gradients [42] or model parameters [43]. Quantization reduces communication cost by using fewer bits to represent each element of the original parameters (*e.g.*, from float32 to float16). Sparsification is another common compression method, which transmits a sparse vector that includes only a subset of the original model parameters. Sebastian *et al.* [44] demonstrate that sparsification, when combined with error compensation, can achieve the same convergence rate as vanilla SGD. Some methods select only those elements whose absolute values are above a fixed [45] or adaptive [46] threshold. Moreover, some works [47], [48] combine both quantization and sparsification to further decrease the volume of transferred data. Nevertheless, all of the aforementioned distributed algorithms necessitate a parameter server to be connected to each worker, which can result in expensive communication links or may be infeasible, particularly for workers located beyond the server's coverage. In contrast, our primary objective is to create a decentralized optimization framework that guarantees rapid convergence

without the need for a central entity.

In recent years, several decentralized optimization algorithms have been proposed, such as decentralized gradient descent and stochastic gradient descent [9], [49], which aim to achieve fast convergence without a central entity. These methods often rely on reducing communication overheads by quantizing model updates [50] or sparsifying them through top-k or random-k approaches [44], [51]. Another approach is analog transmissions that allow each worker to utilize the entire bandwidth and transmit model updates using analog signals that are superpositioned over-the-air channels, thus constructing a globally averaged model update [52]. However, these methods assume sufficient network connectivity, characterized by symmetric and doubly stochastic, column stochastic connectivity matrix or an extended star network topology. Unfortunately, the aforementioned decentralized methods rely on decentralized gradient descent, which has a slow convergence rate. Recently, Cao *et al.* [53] proposed a decentralized asynchronous training framework on heterogeneous workers. It adopts a probabilistic partial model aggregation scheme to alleviate the impacts of straggler workers on model convergence. The proposed framework accelerates model training and eliminates the communication pressure of the central server without increasing the overall communication volume. However, the framework mainly focuses on the asynchronous model training, which may lead to performance degradation compared to the synchronous scheme under the same number of training rounds [54].

6 Conclusion

This paper introduces the CEDFL mechanism, which aims to tackle the challenges of decentralized federated learning (DFL) in edge computing, such as non-IID data, system heterogeneity, and limited bandwidth resources. The proposed approach adopts communication probabilities to ensure communication efficiency. To evaluate the performance of CEDFL, both a simulated and a real DFL environment were built, and extensive experiments were conducted. The experimental results confirm that CEDFL effectively improves model accuracy while reducing resource consumption, including network bandwidth and completion time, for edge computing.

Acknowledgment

This article is supported in part by the National Science Foundation of China (NSFC) under Grants 62132019; in part by the Jiangsu Province Science Foundation for Youths (Grant No. BK20230275); in part by the Anhui Province Science Foundation for Youths (Grant No. 2408085QF185); in part by the Fundamental Research Funds for the Central Universities (Grants No. WK2150110033).

References

[1] G. Zhu, D. Liu, Y. Du, C. You, J. Zhang, and K. Huang, "Towards an intelligent edge: Wireless communication meets machine learning," *arXiv preprint arXiv:1809.00343*, 2018.

[2] M. Satyanarayanan, "The emergence of edge computing," *Computer*, vol. 50, no. 1, pp. 30–39, 2017.

[3] H. B. McMahan and D. Ramage., "http://www.googblogs.com/federated-learning-collaborative-machine-learning-without-centralized-training-data/," Google, 2017.

[4] T. Nishio and R. Yonetani, "Client selection for federated learning with heterogeneous resources in mobile edge," in *ICC 2019-2019 IEEE International Conference on Communications (ICC)*. IEEE, 2019, pp. 1–7.

[5] H. Wang, Z. Kaplan, D. Niu, and B. Li, "Optimizing federated learning on non-iid data with reinforcement learning," in *IEEE INFOCOM 2020-IEEE Conference on Computer Communications*. IEEE, 2020, pp. 1698–1707.

[6] C. Chen, W. Wang, and B. Li, "Round-robin synchronization: Mitigating communication bottlenecks in parameter servers," in *IEEE INFOCOM 2019-IEEE Conference on Computer Communications*. IEEE, 2019, pp. 532–540.

[7] A. Krizhevsky, I. Sutskever, and G. E. Hinton, "Imagenet classification with deep convolutional neural networks," in *Advances in neural information processing systems*, 2012, pp. 1097–1105.

[8] S. Savazzi, M. Nicoli, and V. Rampa, "Federated learning with cooperating devices: A consensus approach for massive iot networks," *IEEE Internet of Things Journal*, vol. 7, no. 5, pp. 4641–4654, 2020.

[9] 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 *Advances in Neural Information Processing Systems*, 2017, pp. 5330–5340.

[10] P. Kairouz, H. B. McMahan, B. Avent, A. Bellet, M. Bennis, A. N. Bhagoji, K. Bonawitz, Z. Charles, G. Cormode, R. Cummings *et al.*, "Advances and open problems in federated learning," *arXiv preprint arXiv:1912.04977*, 2019.

[11] A. Elgabli, J. Park, A. S. Bedi, M. Bennis, and V. Aggarwal, "Gadmm: Fast and communication efficient framework for distributed machine learning." *Journal of Machine Learning Research*, vol. 21, no. 76, pp. 1–39, 2020.

[12] P. Zhou, Q. Lin, D. Loghin, B. C. Ooi, Y. Wu, and H. Yu, "Communication-efficient decentralized machine learning over heterogeneous networks," in *2021 IEEE 37th International Conference on Data Engineering (ICDE)*. IEEE, 2021, pp. 384–395.

[13] L. Bottou, "Stochastic gradient descent tricks," in *Neural networks: Tricks of the trade*. Springer, 2012, pp. 421–436.

[14] A. Rakhlin, O. Shamir, and K. Sridharan, "Making gradient descent optimal for strongly convex stochastic optimization," in *Proceedings of the 29th International Coference on International Conference on Machine Learning*, 2012, pp. 1571–1578.

[15] J. Li, Y. Rong, H. Meng, Z. Lu, T. Kwok, and H. Cheng, "Tatc: predicting alzheimer's disease with actigraphy data," in *Proceedings of the 24th ACM SIGKDD International Conference on Knowledge Discovery & Data Mining*, 2018, pp. 509–518.

[16] X. Ouyang, Z. Xie, J. Zhou, J. Huang, and G. Xing, "Clusterfl: a similarity-aware federated learning system for human activity recognition," in *Proceedings of the 19th Annual International Conference on Mobile Systems, Applications, and Services*, 2021, pp. 54–66.

[17] B. McMahan, E. Moore, D. Ramage, S. Hampson, and B. A. y Arcas, "Communication-efficient learning of deep networks from decentralized data," in *Artificial Intelligence and Statistics*, 2017, pp. 1273–1282.

[18] N. Wang, B. Varghese, M. Matthaiou, and D. S. Nikolopoulos, "Enorm: A framework for edge node resource management," *IEEE transactions on services computing*, 2017.

[19] M. Satyanarayanan, T. Eiszler, J. Harkes, H. Turki, and Z. Feng, "Edge computing for legacy applications," *IEEE Pervasive Computing*, vol. 19, no. 4, pp. 19–28, 2020.

[20] P. Goyal, P. Dollár, R. Girshick, P. Noordhuis, L. Wesolowski, A. Kyrola, A. Tulloch, Y. Jia, and K. He, "Accurate, large minibatch sgd: Training imagenet in 1 hour," *arXiv preprint arXiv:1706.02677*, 2017.

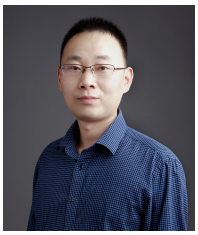
[21] K. Hsieh, A. Harlap, N. Vijaykumar, D. Konomis, G. R. Ganger, P. B. Gibbons, and O. Mutlu, "Gaia: Geo-distributed machine learning approaching {LAN} speeds," in *14th {USENIX} Symposium on Networked Systems Design and Implementation ({NSDI} 17)*, 2017, pp. 629–647.

[22] A. Bellet, A.-M. Kermarrec, and E. Lavoie, "D-cliques: Compensating noniidness in decentralized federated learning with topology," *arXiv preprint arXiv:2104.07365*, 2021.

[23] X. Lian, W. Zhang, C. Zhang, and J. Liu, "Asynchronous decentralized parallel stochastic gradient descent," in *International Conference on Machine Learning*. PMLR, 2018, pp. 3043–3052.
[24] Y. Esfandiari, S. Y. Tan, Z. Jiang, A. Balu, E. Herron, C. Hegde, and S. Sarkar, "Cross-gradient aggregation for decentralized learning from non-iid data," *arXiv preprint arXiv:2103.02051*, 2021.
[25] D. Adjodah, D. Calacci, A. Dubey, A. Goyal, P. Krafft, E. Moro, and A. Pentland, "Leveraging communication topologies between learning agents in deep reinforcement learning," in *Proceedings of the 19th International Conference on Autonomous Agents and MultiAgent Systems*, 2020, pp. 1738–1740.
[26] Z. Wang, Y. Hu, J. Xiao, and C. Wu, "Efficient ring-topology decentralized federated learning with deep generative models for industrial artificial intelligent," *arXiv preprint arXiv:2104.08100*, 2021.
[27] F. R. Chung and F. C. Graham, *Spectral graph theory*. American Mathematical Soc., 1997, no. 92.
[28] X. Lyu, C. Ren, W. Ni, H. Tian, R. P. Liu, and E. Dutkiewicz, "Optimal online data partitioning for geo-distributed machine learning in edge of wireless networks," *IEEE Journal on Selected Areas in Communications*, vol. 37, no. 10, pp. 2393–2406, 2019.
[29] K. Hsieh, A. Phanishayee, O. Mutlu, and P. Gibbons, "The non-iid data quagmire of decentralized machine learning," in *International Conference on Machine Learning*. PMLR, 2020, pp. 4387–4398.
[30] K. Simonyan and A. Zisserman, "Very deep convolutional networks for large-scale image recognition," *arXiv preprint arXiv:1409.1556*, 2014.
[31] A. Krizhevsky, G. Hinton *et al.*, "Learning multiple layers of features from tiny images," Citeseer, Tech. Rep., 2009.
[32] S. Targ, D. Almeida, and K. Lyman, "Resnet in resnet: Generalizing residual architectures," *arXiv preprint arXiv:1603.08029*, 2016.
[33] S. Zheng, Q. Meng, T. Wang, W. Chen, N. Yu, Z.-M. Ma, and T.-Y. Liu, "Asynchronous stochastic gradient descent with delay compensation," in *International Conference on Machine Learning*, 2017, pp. 4120–4129.
[34] J. Tsitsiklis, D. Bertsekas, and M. Athans, "Distributed asynchronous deterministic and stochastic gradient optimization algorithms," *IEEE transactions on automatic control*, vol. 31, no. 9, pp. 803–812, 1986.
[35] B. Sirb and X. Ye, "Decentralized consensus algorithm with delayed and stochastic gradients," *SIAM Journal on Optimization*, vol. 28, no. 2, pp. 1232–1254, 2018.
[36] A. Nedić, A. Olshevsky, and M. G. Rabbat, "Network topology and communication-computation tradeoffs in decentralized optimization," *Proceedings of the IEEE*, vol. 106, no. 5, pp. 953–976, 2018.
[37] J. Wang, A. K. Sahu, Z. Yang, G. Joshi, and S. Kar, "Matcha: Speeding up decentralized sgd via matching decomposition sampling," in *2019 Sixth Indian Control Conference (ICC)*. IEEE, 2019, pp. 299–300.
[38] A. T. Suresh, X. Y. Felix, S. Kumar, and H. B. McMahan, "Distributed mean estimation with limited communication," in *International Conference on Machine Learning*. PMLR, 2017, pp. 3329–3337.
[39] N. Sriranga, C. R. Murthy, and V. Aggarwal, "A method to improve consensus averaging using quantized admm," in *2019 IEEE International Symposium on Information Theory (ISIT)*. IEEE, 2019, pp. 507–511.
[40] E. Jeong, S. Oh, H. Kim, J. Park, M. Bennis, and S.-L. Kim, "Communication-efficient on-device machine learning: Federated distillation and augmentation under non-iid private data," *arXiv preprint arXiv:1811.11479*, 2018.
[41] S. Wang, T. Tuor, T. Salonidis, K. K. Leung, C. Makaya, T. He, and K. Chan, "Adaptive federated learning in resource constrained edge computing systems," *IEEE Journal on Selected Areas in Communications*, vol. 37, no. 6, pp. 1205–1221, 2019.
[42] T. Chen, G. B. Giannakis, T. Sun, and W. Yin, "Lag: Lazily aggregated gradient for communication-efficient distributed learning," *arXiv preprint arXiv:1805.09965*, 2018.
[43] Y. Liu, W. Xu, G. Wu, Z. Tian, and Q. Ling, "Communication-censored admm for decentralized consensus optimization," *IEEE Transactions on Signal Processing*, vol. 67, no. 10, pp. 2565–2579, 2019.
[44] S. U. Stich, J.-B. Cordonnier, and M. Jaggi, "Sparsified sgd with memory," *arXiv preprint arXiv:1809.07599*, 2018.
[45] N. Strom, "Scalable distributed dnn training using commodity gpu cloud computing," in *Sixteenth Annual Conference of the International Speech Communication Association*, 2015.
[46] A. F. Aji and K. Heafield, "Sparse communication for distributed gradient descent," *arXiv preprint arXiv:1704.05021*, 2017.
[47] F. Sattler, S. Wiedemann, K.-R. Müller, and W. Samek, "Robust and communication-efficient federated learning from non-iid data," *IEEE transactions on neural networks and learning systems*, vol. 31, no. 9, pp. 3400–3413, 2019.
[48] ——, "Sparse binary compression: Towards distributed deep learning with minimal communication," in *2019 International Joint Conference on Neural Networks (IJCNN)*. IEEE, 2019, pp. 1–8.
[49] A. Nedic and A. Ozdaglar, "Distributed subgradient methods for multi-agent optimization," *IEEE Transactions on Automatic Control*, vol. 54, no. 1, pp. 48–61, 2009.
[50] A. Koloskova, T. Lin, S. U. Stich, and M. Jaggi, "Decentralized deep learning with arbitrary communication compression," *arXiv preprint arXiv:1907.09356*, 2019.
[51] Y. Chen, A. Hashemi, and H. Vikalo, "Communication-efficient algorithms for decentralized optimization over directed graphs," *arXiv preprint arXiv:2005.13189*, 2020.
[52] H. Xing, O. Simeone, and S. Bi, "Decentralized federated learning via sgd over wireless d2d networks," in *2020 IEEE 21st International Workshop on Signal Processing Advances in Wireless Communications (SPAWC)*. IEEE, 2020, pp. 1–5.
[53] J. Cao, Z. Lian, W. Liu, Z. Zhu, and C. Ji, "Hadfl: Heterogeneity-aware decentralized federated learning framework," in *2021 58th ACM/IEEE Design Automation Conference (DAC)*. IEEE, 2021, pp. 1–6.
[54] A. Wongpanich, Y. You, and J. Demmel, "Rethinking the value of asynchronous solvers for distributed deep learning," in *Proceedings of the International Conference on High Performance Computing in Asia-Pacific Region*, 2020, pp. 52–60.
[55] H. Tang, S. Gan, C. Zhang, T. Zhang, and J. Liu, "Communication compression for decentralized training," *Advances in Neural Information Processing Systems*, vol. 31, pp. 7652–7662, 2018.

Jianchun Liu (Member, IEEE) received the Ph.D. degree in School of Data Science from the University of Science and Technology of China in 2022. He is currently an associate researcher in the School of Computer Science and Technology at University of Science and Technology of China. His main research interests are software defined networks, network function virtualization, edge computing and federated learning.

Jiaming Yan received the B.S. degree in 2021 from Hefei University of Technology. He is currently studying for a doctor's degree in the School of Computer Science, University of Science and Technology of China (USTC). His main research interests are edge computing, deep learning and federated learning.

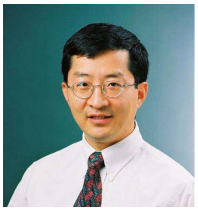


Hongli Xu (Member, IEEE) received the B.S. degree in computer science from the University of Science and Technology of China, China, in 2002, and the Ph. D degree in computer software and theory from the University of Science and Technology of China, China, in 2007. He is a professor with the School of Computer Science and Technology, University of Science and Technology of China (USTC), China. He was awarded the Outstanding Youth Science Foundation of NSFC, in 2018. He has won the best paper award or the best paper candidate in several famous conferences. He has published more than 100 papers in famous journals and conferences, including the IEEE/ACM Transactions on Networking, IEEE Transactions on Mobile Computing, IEEE Transactions on Parallel and Distributed Systems, Infocom and ICNP, etc. He has also held more than 30 patents. His main research interest is software defined networks, edge computing and Internet of Thing.

Lun Wang received the B.S. degree in software engineering from the University of Electronic Science and Technology of China in 2019 and the Ph.D. degree in computer software and theory from the University of Science and Technology of China in 2024. He is currently a senior engineer at Alibaba Group. His research interests include cloud-native system and distributed machine learning.

Zhiyuan Wang received the B.S. degree from the Jilin University in 2019, and the Ph.D. drgree in the School of Computer Science, University of Science and Technology of China (USTC) in 2024. He currently works in Alibaba. His main research interests are edge computing, federated learning and distributed machine learning.

Jinyang Huang received the Ph.D. degree in School of Cyberspace Security from the University of Science and Technology of China in 2022. He is currently a lecturer in the School of Computer and Information at Hefei University of Technology. His research interests include Wireless Security and Wireless Sensing. He is a TPC Member of ACM MM, IEEE ICME, and Globecom. He is now an editorial board member of Applied Sciences.



Chunming Qiao directs the Lab for Advanced Network Design, Analysis, and Research (LANDER) at SUNY Buffalo. He is a recipient of many awards including the SUNY Chancellors Award for Excellence in Scholarship and Creativity. He has published many highly cited papers with an h-index of over 50 (according to Google Scholar). He has seven US patents. He pioneered research on Optical Internet in 1997, and one of his papers has been cited by more than 2,000 times. He also pioneered the work on integrated cellular and ad hoc relaying systems (iCAR) in 1999, which is recognized as the harbinger for todayfs push towards the convergence between heterogeneous wireless technologies, and has been featured in Business Week and Wireless Europe, as well as at the websites of New Scientists and CBC. These works have inspired a lot of follow-up works around the world and attracted funding from a dozen of major IT and telecommunications companies including Alcatel, Cisco, Google, NEC labs, Nokia, and Telcordia. Dr. Qiao has given a dozen of keynotes, and numerous invited talks on the above research topics. He has chaired and co-chaired a dozen of international conferences and workshops. He was an editor of several leading IEEE journals, and chaired a number of technical committees. He was elected to IEEE Fellow for his contributions to optical and wireless network architectures and protocols.

APPENDIX A
PROOF OF THEOREM 2

Proof. Combined with Eq. (10), we can bound the expected sum of squares deviation as follows:

$$\begin{aligned}
&\mathbb{E}[\|\Omega^{T+1} - \boldsymbol{w}^*\|^2 \mid \Omega^T] \\
&= \mathbb{E}[\|\mathbf{W}^T(\Omega^T - \eta \boldsymbol{g}^T) - \boldsymbol{w}^*\|^2 \mid \Omega^T] \\
&= \mathbb{E}[(\Omega^T - \boldsymbol{w}^* - \eta \boldsymbol{g}^T)^{\mathbb{T}}(\mathbf{W}^T)^{\mathbb{T}}\mathbf{W}^T(\Omega^T - \boldsymbol{w}^* - \eta \boldsymbol{g}^T) \mid \Omega^T] \\
&\leq \alpha \mathbb{E}[(\Omega^T - \boldsymbol{w}^* - \eta \boldsymbol{g}^T)^{\mathbb{T}}(\Omega^T - \boldsymbol{w}^* - \eta \boldsymbol{g}^T) \mid \Omega^T] \qquad (22)
\end{aligned}$$

where α is the maximum among historical eigenvalues α^t ($t \in [T]$). Let $h_{i,j}$ denote the duration time of one local epoch for worker i communicating with worker j. Then, the average duration time for worker i is

$$\bar{h}_i = \sum_{j=1}^{m} h_{i,j} \cdot p_{i,j} \cdot d_{i,j} \qquad (23)$$

We use p_i to denote the probability of worker i communicating with one of its neighbors at any epoch, which can be derived as

$$p_i = \frac{1/\bar{h}_i}{\sum_{i=1}^{m}(1/\bar{h}_i)} \qquad (24)$$

where $1/\bar{h}_i$ is the iterative frequency of worker i. The larger iterative frequency a worker node has, the higher probability (w.r.t. p_i) it can communicate at a global iteration step. Combining Eqs. (22) and (24), we have

$$\begin{aligned}
&\mathbb{E}[(\Omega^T - \boldsymbol{w}^* - \eta \boldsymbol{g}^T)^{\mathbb{T}}(\Omega^T - \boldsymbol{w}^* - \eta \boldsymbol{g}^T)] \\
&= \mathbb{E}[\|\Omega^T - \boldsymbol{w}^*\|^2 - 2\eta(\boldsymbol{g}^T)^{\mathbb{T}}(\Omega^T - \boldsymbol{w}^*) + \eta^2(\boldsymbol{g}^T)^{\mathbb{T}}\boldsymbol{g}^T] \\
&= \|\Omega^T - \boldsymbol{w}^*\|^2 - 2\eta \sum_{i=1}^{m} p_i \nabla F_i(\omega_i^T)^{\mathbb{T}}(\omega_i^T - \omega^*) \\
&+ \eta^2 \sum_{i=1}^{m} p_i \nabla F_i(\omega_i^T)^{\mathbb{T}} \nabla F_i(\omega_i^T) + \eta^2 \sum_{i=1}^{m} p_i \mathbb{E}[(\pi_i^T)^{\mathbb{T}}\pi_i^T] \qquad (25)
\end{aligned}$$

Since π_i^T has zero mean assumption, we drop the terms that are linear in π_i^T.

Let $x = \omega_i^T$ and $y = \omega^*$. Combining with Remark 1, we have that

$$\begin{aligned}
&-\nabla F(\omega_i^T)^{\mathbb{T}}(\omega_i^T - \omega^*) \\
&= -(\nabla F(\omega_i^T) - 0)^{\mathbb{T}}(\omega_i^T - \omega^*) \\
&= -(\nabla F(\omega_i^T) - \nabla F(\omega^*))^{\mathbb{T}}(\omega_i^T - \omega^*) \\
&\leq -\frac{\mu L}{\mu + L}(\omega_i^T - \omega^*)^{\mathbb{T}}(\omega_i^T - \omega^*) \\
&\quad - \frac{1}{\mu + L}(\nabla F(\omega_i^T) - \nabla F(\omega^*))^{\mathbb{T}}(\nabla F(\omega_i^T) - \nabla F(\omega^*)) \\
&\leq -\frac{\mu L}{\mu + L}(\omega_i^T - \omega^*)^{\mathbb{T}}(\omega_i^T - \omega^*) - \frac{1}{\mu + L}\nabla F(\omega_i^T)^{\mathbb{T}}\nabla F(\omega_i^T)
\end{aligned} \qquad (26)$$

According to Eq. (26), we can derive that:

$$\begin{aligned}
&-2\eta \sum_{i=1}^{m} p_i \nabla F(\omega_i^T)^{\mathbb{T}}(\omega_i^T - \omega^*) \\
&\leq -\frac{2\eta\mu L}{\mu + L} \sum_{i=1}^{m} p_i (\omega_i^T - \omega^*)^{\mathbb{T}}(\omega_i^T - \omega^*) \\
&\quad - \frac{2\eta}{\mu + L} \sum_{i=1}^{m} p_i \nabla F(\omega_i^T)^{\mathbb{T}} \nabla F(\omega_i^T) \\
&\leq -\frac{2\eta\mu L}{\mu + L} p_{\min} \sum_{n=1}^{M} (\omega_i^T - \omega^*)^{\mathbb{T}}(\omega_i^T - \omega^*) \\
&\quad - \frac{2\eta}{\mu + L} \sum_{i=1}^{m} p_i \nabla F(\omega_i^T)^{\mathbb{T}} \nabla F(\omega_i^T) \qquad (27)
\end{aligned}$$

where $p_{\min}$ is the smallest value among $p_i, \forall i \in [m]$. By utilizing Eq. (25) and Eq. (27), we can derive that

$$\begin{aligned}
&\mathbb{E}[(\Omega^T - \boldsymbol{w}^* - \eta \boldsymbol{g}^T)^{\mathbb{T}}(\Omega^T - \boldsymbol{w}^* - \eta \boldsymbol{g}^T) \mid \Omega^T] \\
&\leq (1 - \frac{2\eta\mu L}{\mu + L} p_{\min}) \|\Omega^T - \boldsymbol{w}^*\|^2 + \eta^2 \varphi^2 \\
&\quad + (\eta^2 - \frac{2\eta}{\mu + L}) \sum_{n=1}^{M} p_n \nabla F(\omega_i^T)^{\mathbb{T}} \nabla F(\omega_i^T) \qquad (28)
\end{aligned}$$

The last term in Eq. (28) can be droped if $0 < \eta < \frac{2}{\mu + L}$. Then, we have

$$\begin{aligned}
&\mathbb{E}[\|\Omega^{T+1} - \boldsymbol{w}^*\|^2 \mid \Omega^T] \\
&\leq \alpha \mathbb{E}[(\Omega^T - \boldsymbol{w}^* - \eta \boldsymbol{g}^T)^{\mathbb{T}}(\Omega^T - \boldsymbol{w}^* - \eta \boldsymbol{g}^T) \mid \Omega^T] \\
&\leq \alpha(1 - \frac{2\eta\mu L}{\mu + L} p_{\min}) \|\Omega^T - \boldsymbol{w}^*\|^2 + \alpha\eta^2\varphi^2 \\
&\leq \alpha \|\Omega^T - \boldsymbol{w}^*\|^2 + \alpha\eta^2\varphi^2 \qquad (29)
\end{aligned}$$

By expanding the recursion, we can express the statement as

$$\begin{aligned}
\mathbb{E}[\|\Omega^T - \boldsymbol{w}^*\|^2] &\leq \alpha^T \|\Omega^0 - \boldsymbol{w}^*\|^2 + \frac{1 - \alpha^T}{1 - \alpha} \alpha\eta^2\varphi^2 \\
&\leq \alpha^T \|\Omega^0 - \boldsymbol{w}^*\|^2 + \frac{\alpha}{1 - \alpha} \eta^2\varphi^2 \qquad (30)
\end{aligned}$$

□

APPENDIX B
PROOF OF LEMMA 3

Proof: For convenience, we first introduce the following matrix notations:

$$\begin{cases}
\Omega^t := [\omega_1^t, \ldots, \omega_m^t], \\
\overline{\Omega}^t := [\overline{\omega}^t, \ldots, \overline{\omega}^t], \\
\nabla F(\Omega^t) := [\nabla F(\omega_1^t), \ldots, \nabla F(\omega_m^t)],
\end{cases} \qquad (31)$$

where $\overline{\omega}^t = \frac{1}{m}\sum_{i=1}^{m} \omega_i^t$ and $\overline{\nabla F}(\Omega^t) = \frac{1}{m}\sum_{i=1}^{m} \nabla F(\omega_i^t)$.

According to the Lipschitz smoothness property in Assumption 1, we obtain:

$$\begin{aligned}
\mathbb{E} f(\overline{\omega}^{(t+1)}) &= \mathbb{E} f\left(\overline{\omega}^t - \frac{\eta\epsilon}{m} \sum_{i=1}^{m} \nabla F(\omega_i^t)\right) \\
&\leq f(\overline{\omega}^t) - \epsilon \mathbb{E}\left\langle \nabla f(\overline{\omega}^t), \frac{\eta}{m} \sum_{i=1}^{m} \nabla F(\omega_i^t) \right\rangle \\
&\quad + \frac{L\eta^2\epsilon^2}{2} \mathbb{E}\left\| \frac{1}{m} \sum_{i=1}^{m} \nabla F(\omega_i^t) \right\|_2^2. \qquad (32)
\end{aligned}$$

Then we bound the second term:

$$-\mathbb{E}\langle \nabla f(\overline{\omega}^t), \frac{\eta}{m} \sum_{i=1}^{m} \nabla F(\omega_i^t) \rangle$$

$$
\begin{aligned}
&= \mathbb{E}\langle \nabla f(\overline{\omega}^t), \eta \nabla f(\overline{\omega}^t) - \frac{\eta}{m}\sum_{i=1}^{m} \nabla F(\omega_i^t) - \eta \nabla f(\overline{\omega}^t)\rangle \\
&= \mathbb{E}\langle \nabla f(\overline{\omega}^t), \frac{\eta}{m}\sum_{i=1}^{m} \nabla f_i(\overline{\omega}^t) - \frac{\eta}{m}\sum_{i=1}^{m} \nabla F(\omega_i^t) - \eta \nabla f(\overline{\omega}^t)\rangle \\
&= \langle \nabla f(\overline{\omega}^t), \frac{\eta}{m}\sum_{i=1}^{m} \left(\nabla f_i(\overline{\omega}^t) - \nabla f_i(\omega_i^t)\right)\rangle - \eta \left\|\nabla f(\overline{\omega}^t)\right\|_2^2 \\
&= \frac{\eta}{m}\sum_{i=1}^{m} \langle \nabla f(\overline{\omega}^t), \nabla f_i(\overline{\omega}^t) - \nabla f_i(\omega_i^t)\rangle - \eta \left\|\nabla f(\overline{\omega}^t)\right\|_2^2 \\
&\leq \frac{\eta}{2m}\sum_{i=1}^{m} \left\|\nabla f_i(\overline{\omega}^t) - \nabla f_i(\omega_i^t)\right\|_2^2 - \frac{\eta}{2}\left\|\nabla f(\overline{\omega}^t)\right\|_2^2, \qquad (33)
\end{aligned}
$$

where the last step comes from the inequality:

$$2\langle \mathbf{a}, \mathbf{b}\rangle \leq \|\mathbf{a}\|_2^2 + \|\mathbf{b}\|_2^2, \forall \mathbf{a}, \mathbf{b} \in \mathbb{R}^d$$

For the third term, we add and subtract $\nabla f(\overline{\omega}^t)$ and the sum of $\nabla f_i(\omega_i^t)$:

$$
\begin{aligned}
&\mathbb{E}\left\|\frac{1}{m}\sum_{i=1}^{m} \nabla F(\omega_i^t)\right\|_2^2 \\
\leq &\mathbb{E}\left\|\frac{1}{m}\sum_{i=1}^{m} \left(\nabla F(\omega_i^t) - \nabla f_i(\omega_i^t)\right)\right\|_2^2 \\
&+ \left\|\frac{1}{m}\sum_{i=1}^{m} \left(\nabla f_i(\omega_i^t) - \nabla f_i(\overline{\omega}^t) + \nabla f_i(\overline{\omega}^t)\right)\right\|_2^2 \\
\leq &\frac{2}{m}\sum_{i=1}^{m} \left\|\nabla f_i(\omega_i^t) - \nabla f_i(\overline{\omega}^t)\right\|_2^2 + 2\left\|\nabla f(\overline{\omega}^t)\right\|_2^2 + \frac{\sigma^2}{m}, \qquad (34)
\end{aligned}
$$

where the first step comes from the Assumption 2 and the following inequality with $\kappa = 1$:

$$\|\mathbf{a} + \mathbf{b}\|_2^2 \leq (1+\kappa)\|\mathbf{a}\|_2^2 + (1+\kappa^{-1})\|\mathbf{b}\|_2^2, \kappa > 0,$$

The last step comes from Assumption 3, and we have bounded gradient variance: $\mathbb{E}\left\|\nabla F(\omega_i, \xi_i) - \nabla F(\omega_i)\right\|_2^2 \leq \sigma^2, \forall \omega \in \mathbb{R}^d, \forall i \in \mathcal{M}$. Combining Eq. (32), Eq. (33) and Eq. (34) as well as Assumption 1, we can obtain:

$$
\begin{aligned}
\mathbb{E} f(\overline{\omega}^{(t+1)}) \leq f(\overline{\omega}^t) &+ \frac{\eta L^2 \epsilon}{m}\left(\frac{1}{2} + \eta L \epsilon\right)\sum_{i=1}^{m} \left\|\overline{\omega}^t - \omega_i^t\right\|_2^2 \\
&+ \eta\epsilon\left(L\eta\epsilon - \frac{1}{2}\right)\left\|\nabla f(\overline{\omega}^t)\right\|_2^2 + \frac{\sigma^2\eta^2\omega^2 L}{m}. \qquad (35)
\end{aligned}
$$

Applying $\eta \leq \frac{1}{4L\epsilon}$ in the second and the third terms, we complete the proof.

□

APPENDIX C
PROOF OF LEMMA 5

Proof: Based on the updating rule, we have:

$$
\begin{aligned}
\Omega^t &= \sum_{s=1}^{t-1} \Omega^s \mathbf{W}^{t-s} + \sum_{s=1}^{t-1} \eta \nabla F(\Omega^s)\mathbf{W}^{t-s-1}, \\
\overline{\Omega}^t &= \sum_{s=1}^{t-1} \Omega^s \mathbf{W}^{t-s}\frac{\mathbf{1}}{m} + \sum_{s=1}^{t-1} \eta \nabla F(\Omega^s)\mathbf{W}^{t-s-1}\frac{\mathbf{1}}{m} \\
&= \sum_{s=1}^{t-1} \overline{\omega}^s + \sum_{s=1}^{t-1} \eta \overline{\nabla F}(\Omega^s). \qquad (36)
\end{aligned}
$$

Thus, we can obtain the following result:

$$
\begin{aligned}
&\sum_{i=1}^{m} \mathbb{E}\left\|\overline{\omega}^t - \omega_i^t\right\|_2^2 \\
=&\sum_{i=1}^{m} \mathbb{E}\left\|\sum_{s=1}^{t-1}\left(\Omega^s \mathbf{W}^{t-s}\mathbf{e}^i - \overline{\omega}^s\right)\right. \\
&\left. - \sum_{s=1}^{t-1} \eta\left(\nabla F(\Omega^s)\mathbf{W}^{t-s-1}\mathbf{e}^i - \overline{\nabla F}(\Omega^s)\right)\right\|_F^2 \\
\leq &2\sum_{i=1}^{m}\sum_{s=1}^{t-1} \mathbb{E}\left\|\Omega^s \mathbf{W}^{t-s}\mathbf{e}^i - \overline{\omega}^s\right\|_F^2 \\
&+ 2\sum_{i=1}^{m} \mathbb{E}\left\|\sum_{s=1}^{t-1} \eta\left(\nabla F(\Omega^s)\mathbf{W}^{t-s-1}\mathbf{e}^i - \overline{\nabla F}(\Omega^s)\right)\right\|_F^2 \\
\leq &2\mathbb{E}\sum_{s=1}^{t-1} \left\|\alpha^{t-s}\Omega^s\right\|_F^2 + 2\mathbb{E}\left(\sum_{s=1}^{t-1} \eta\alpha^{t-s-1}\left\|\nabla F(\Omega^s)\right\|_F\right)^2. \qquad (37)
\end{aligned}
$$

We assume that $\frac{1}{m}\sum_{i=1}^{m}\left\|\nabla f_i(x_i) - \nabla f(x)\right\|_2^2 \leq \zeta^2, \forall x \in \mathbb{R}^d, \forall i \in [m]$. To bound the term $\left\|\nabla F(\Omega^s)\right\|_F^2$ in Eq. (37), we first bound $\left\|\nabla F(\omega_i^t)\right\|_2^2$ as follows:

$$
\begin{aligned}
&\mathbb{E}\left\|\nabla F(\omega_i^t)\right\|_2^2 \\
=&\mathbb{E}\left\|\nabla F(\omega_i^t) - \nabla f_i(\omega_i^t) + \nabla f_i(\omega_i^t)\right\|_2^2 \\
=&\mathbb{E}\left\|\nabla F(\omega_i^t) - \nabla f_i(\omega_i^t)\right\|_2^2 + \mathbb{E}\left\|\nabla f_i(\omega_i^t)\right\|_2^2 \\
&+ 2\mathbb{E}\left\langle \mathbb{E}\nabla F(\omega_i^t) - \nabla f_i(\omega_i^t), \nabla f_i(\omega_i^t)\right\rangle \\
=&\mathbb{E}\left\|\nabla F(\omega_i^t) - \nabla f_i(\omega_i^t)\right\|_2^2 + \mathbb{E}\left\|\nabla f_i(\omega_i^t)\right\|_2^2 \\
\leq &\sigma^2 + \mathbb{E}\left\|\nabla(f_i(\omega_i^t) - \nabla f_i(\overline{\omega}^t))\right. \\
&\left. + (\nabla f_i(\overline{\omega}^t) - \nabla f(\overline{\omega}^t)) + \nabla f(\overline{\omega}^t)\right\|_2^2 \\
\leq &\sigma^2 + 3\mathbb{E}\left\|\nabla f_i(\omega_i^t) - \nabla f_i(\overline{\omega}^t)\right\|_2^2 \\
&+ 3\mathbb{E}\left\|\nabla f_i(\overline{\omega}^t) - \nabla f(\overline{\omega}^t)\right\|_2^2 + 3\mathbb{E}\left\|\nabla f(\overline{\omega}^t)\right\|_2^2 \\
\leq &\sigma^2 + 3L^2\mathbb{E}\left\|\overline{\omega}^t - \omega_i^t\right\|_2^2 + 3\zeta^2 + 3\mathbb{E}\left\|\nabla f(\overline{\omega}^t)\right\|_2^2, \qquad (38)
\end{aligned}
$$

which means

$$
\begin{aligned}
\mathbb{E}\left\|\nabla F(\Omega^t)\right\|_F^2 &\leq \sum_{i=1}^{m}\left\|\nabla F(\omega_i^t)\right\|_2^2 \\
&\leq m\sigma^2 + 3L^2\sum_{i=1}^{m}\mathbb{E}\left\|\overline{\omega}^t - \omega_i^t\right\|_2^2 \\
&\quad + 3m\zeta^2 + 3m\mathbb{E}\left\|\nabla f(\overline{\omega}^t)\right\|_2^2. \qquad (39)
\end{aligned}
$$

Inserting Eq. (38) into Eq. (37), applying Lemmas 5 and 6 in [55], and setting $\frac{3L^2\eta^2}{(1-\alpha)^2} < 1$, we complete the proof:

$$
\begin{aligned}
\sum_{t=1}^{T}\sum_{i=1}^{m}\mathbb{E}\left\|\overline{\omega}^t - \omega_i^t\right\|_2^2 &\leq \frac{2m\eta^2(\sigma^2 + 3\zeta^2)T}{(1-\alpha)^2 - 3\eta^2L^2} \\
&+ \frac{6m\eta^2}{(1-\alpha)^2 - 3\eta^2L^2}\sum_{t=1}^{T}\mathbb{E}\left\|\nabla f(\overline{\omega}^t)\right\|_2^2 \qquad (40)
\end{aligned}
$$

□