FedVeca: Federated Vectorized Averaging on Non-IID Data With Adaptive Bi-Directional Global Objective

Ping Luo, *Student Member, IEEE*, Jieren Cheng, *Member, IEEE*, N. Xiong, *Senior Member, IEEE*, Zhenhao Liu, *Student Member, IEEE*, and Jie Wu, *Fellow, IEEE*

***Abstract*—Federated Learning (FL) is a distributed machine learning framework in parallel and distributed systems. However, the systems' Non-Independent and Identically Distributed (Non-IID) data negatively affect the communication efficiency, since clients with different datasets may cause significant gaps to the local gradients in each communication round. In this article, we propose a Federated Vectorized Averaging (FedVeca) method to optimize the FL communication system on Non-IID data. Specifically, we set a novel objective for the global model which is related to the local gradients. The local gradient is defined as a bi-directional vector with step size and direction, where the step size is the number of local updates and the direction is divided into positive and negative according to our definition. In FedVeca, the direction is influenced by the step size, thus we average the bi-directional vectors to reduce the effect of different step sizes. Then, we theoretically analyze the relationship between the step sizes and the global objective, and obtain upper bounds on the step sizes per communication round. Based on the upper bounds, we design an algorithm for the server and the client to adaptively adjusts the step sizes that make the objective close to the optimum. Finally, we conduct experiments on different datasets, models and scenarios by building a prototype system, and the experimental results demonstrate the effectiveness and efficiency of the FedVeca method.**

***Index Terms*—Federated learning, machine learning, non-IID data, optimization.**

I. Introduction

PARALLEL and distributed systems have transformed computing by allowing tasks to be executed simultaneously across multiple nodes, enhancing performance and scalability in high-performance computing environments, data centers, and cloud services [1]. With the rise of Big Data and artificial intelligence, these systems have become crucial for distributed machine learning frameworks that handle vast datasets across different locations while maintaining efficiency [2]. For real-world applications of distributed machine learning, considerations must be made for data silos resulting from privacy concerns, leading to the emergence of Federated Learning (FL) as an extension of parallel and distributed systems [3].

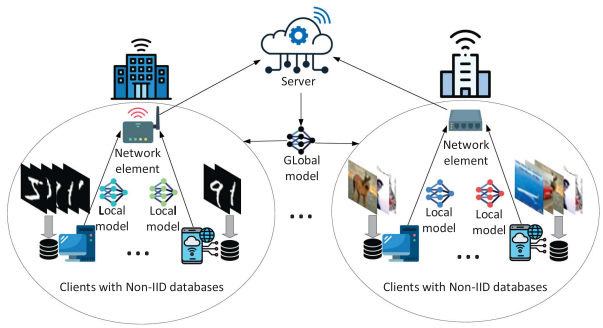

Fig. 1. A typical FL communication networks system.

FL allows decentralized clients to train a global model using their own local data, thus alleviating the problems of data silos and user privacy [4], [5]. FL local training generally adopts the optimization method of Stochastic Gradient Descent (SGD) [6], which requires independent and identically distributed (IID) data to ensure unbiased estimation of global gradient in the training process [7], [8], [9]. Since the clients' local data are obtained from the local environment and usage habits, the generated datasets are usually Non-IID due to differences in size and distribution [10], [11], [12]. The Non-IID datasets and multiple local SGD iterations will bring a drift to the global model in each communication round, which significantly reduces the FL performance and stability in the training process, thus requiring more communication rounds to converge or even fail to converge [13], [14], [15]. Therefore, it becomes a key challenge to optimize the FL objective by reducing the negative impact of Non-IID data.

As illustrated in Fig. 1, we consider a typical Federated Averaging Algorithm (FedAvg) [3] communication networks system, where clients are composed of communication-connected devices with local Non-IID training databases. Intuitively, the

Received 4 September 2023; revised 29 August 2024; accepted 31 August 2024. Date of publication 4 September 2024; date of current version 19 September 2024. This work was supported in part by the Key Research and Development Program of Hainan Province under Grant ZDYF2024GXJS014 and Grant ZDYF2023GXJS163, in part by the National Natural Science Foundation of China (NSFC) under Grant 62162022 and Grant 62162024, in part by the Collaborative Innovation Project of Hainan University under Grant XTCX2022XXB02. Recommended for acceptance by D.-N. Yang. *(Corresponding author: Jieren Cheng.)*

Ping Luo, Jieren Cheng, N. Xiong, and Zhenhao Liu are with the Hainan Blockchain Technology Engineering Research Center, School of Computer Science and Technology, Hainan University, Haikou 570228, China (e-mail: luoping@hainanu.edu.cn; cjr22@163.com; nnxiong10@yahoo.com; lzhcs11@163.com).

Jie Wu is with the Department of Computer and Information Sciences, Temple University, Philadelphia, PA 19122 USA (e-mail: jiewu@temple.edu).

This article has supplementary downloadable material available at https://doi.org/10.1109/TPDS.2024.3454203, provided by the authors.

Digital Object Identifier 10.1109/TPDS.2024.3454203

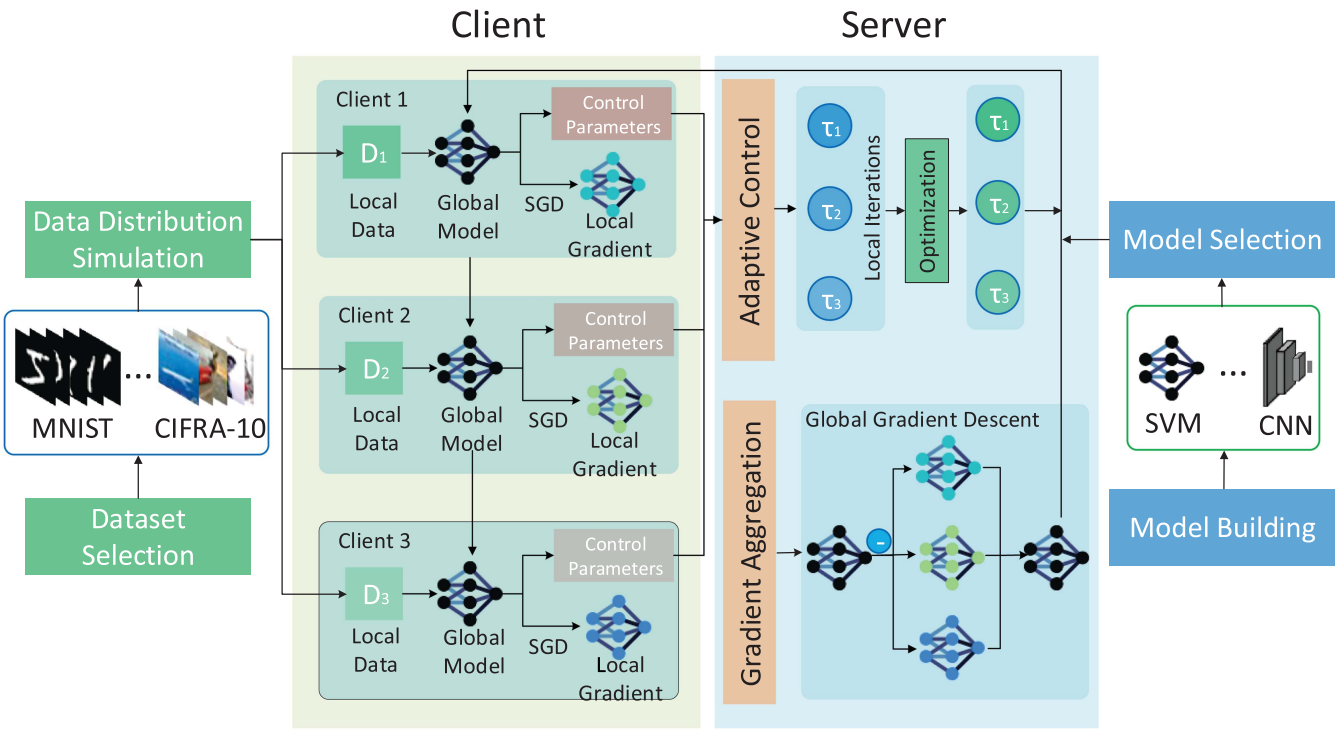

Fig. 2. The overall architecture of proposed Federated Vectorized Averaging (FedVeca) method on Non-IID data.

negative impact is due to the fact that the client has Non-IID data, and the common strategy is to adjust the data distribution from an optimization perspective of the data, thus effectively improving the generalization performance of the global model [16]. Data distribution is usually a fundamental property of FL that is not easily changed, and a more easily tuned parameter is the number of local SGD iterations as a way of counteracting global objective drift [17]. Furthermore, our previous research has revealed that the number of local SGD iterations correlates with the Non-IID nature of datasets on individual clients [18]. Hence, we proceeded to investigate its viability within a more sophisticated FL framework, namely, Normalized Averaging Federated learning (FedNova) [19].

In this paper, we propose a Federated Vectorized Averaging (FedVeca) method to make the global model close to the optimal global model on Non-IID data. Specifically, we define the local gradients as a bi-directional vectors with positive and negative (two categories of directions). Positive vectors represent local gradients that exhibit less deviation from the global gradient vector, while negative vectors indicate a greater disparity. The visual representation of the positive and negative directions of the bi-directional vector is provided in Fig. 4, with the analysis detailed in Section III-C. The standard quantization and classification of these vectors are detailed in Theorem 2 of Section IV. In addition to the direction, we define the step size for the bi-directional vector as the number of local SGD iterations. Then, we adaptively adjust the step size of the averaged bi-directional vector at each client to optimize global model. As shown in Fig. 2, the FedVeca method starts with the selection of the dataset and the construction of the model. Next, the global model is distributed from the server to the individual clients and then trained on the local dataset $D_{1\sim3}$ to obtain the local gradients and control parameters. Local gradients are weighted and aggregated into the global gradient for the global model update, and the control parameters act on the averaged bi-directional vector to optimize the local step size $\tau_{1\sim3}$ of the clients. Finally, the updated global model and the local step size are sent to the individual clients for the next round of updates.

The main contributions of this paper are as follows:

- We set a novel objective for the global model of FedVeca. In a qualitative analysis of the relationship between the global objective and the bi-directional vector's step size, we obtained an upper bound on the step size of each client in each communication round.
- Using the above theoretical upper bound, we propose an algorithm for FedVeca that adaptively balances the step size at each client to accelerate the speed of the global model convergence.
- We evaluate the general performance of the proposed algorithm via extensive experiments on general public datasets, which confirms that the FedVeca algorithm provides efficient performance in the Non-IID datasets.

This paper is organized as follows. Related work is introduced in Section II. The basics and definitions of FL are summarized in Section III. The convergence analysis and control algorithm are presented in Section IV. Experimentation results are shown in Section V and the conclusion is presented in Section VI.

II. Related Work

Based on the strategy of adjusting the distribution of dataset on FL clients, there is a mean-augmented FL strategy [20]

which is inspired by data augmentation methods [21] and allows each client to average the data after exchanging updated model parameters. And the averaged data are sent to each client and used to reduce the degree of local data distribution imbalance. Moreover, other related studies treat each client as a domain, and data augmentation is applied to each client's data by selecting transformations related to the overall distribution to generate similar data distributions [22], [23]. Rather than modifying data directly, some schemes indirectly improve data distribution through client-side selection. For example, the FL process can be stabilized and sped up by describing the data distribution on the client through the uploaded model parameters, thus the best subsets of clients are intelligently selected in each communication round [24]. Similarly, there is a scheme to evaluate the class distribution without knowing the original data, which uses a client selection strategy oriented towards minimizing class imbalance [25]. And the method proposed in [26] creates a globally shared subset of data to obtain better learning performance in the case of Non-IID data distributions. Besides, there are some schemes that share part of local data to the central server for mitigating the negative impact of Non-IID data [27], [28].

In addition to the optimization of FL strategies by improving data distribution, the broader focus aims at adapting the model to local tasks. For adapting the model, some methods create a better initial model by local fine-tuning. The scheme proposed in [29] makes the local model have a better initial global model by using Model-Agnostic Meta-Learning (MAML) [30]. As an extension, [31] proposes a federated meta-learning formulation using Moreau envelopes. Besides, Multi-tasking is proposed to solve statistical challenges in FL environments by finding relationships between clients' data such that similar clients learn similar models [32], [33], [34]. With the idea of extracting information from large models to small models, knowledge distillation has also been generalized to optimization strategies for FL [35]. For example, a distillation framework for model fusion with robustness is proposed in [36], using unlabeled data output from client models to train a central classifier that flexibly aggregates heterogeneous models of clients. [37] proposed a domain-adaptive federated optimization method that aligns the learned representations of different clients with the data distribution of the target nodes and uses feature decomposition to enhance knowledge transfer.

The above algorithms are based on the same number of local SGD iterations per client in the training process, and we abbreviate this setup as synchronous FL in the paper. In synchronous FL optimization, the training time always depends on the slowest training node [19]. Besides, some clients are unable to complete the training tasks within the specified time, and these clients will be discarded under normal circumstances, thus affecting the performance and accuracy of the trained model [3]. Since the synchronous FL does not conform to the update principles of clients in real-world environments, it is necessary for each client to have a different number of local SGD iterations. But on Non-IID data, clients with different numbers of local SGD iterations lead to a problem where the global model is biased in the direction of the local model with a higher number of local SGD iterations. To solve this problem, Proximal Federated Learning (FedProx) [38] introduces a regularization term in the local loss function to adjust the distance from the global model, so that there is no need to manually adjust the number of local SGD iterations. Moreover, [39] proposed a local SGD with reduced variance and can further reduce the communication complexity by removing the dependence of different clients on gradient variance, which has a significant performance on Non-IID datasets. Further, Stochastic Controlled Federated Learning (SCAFFOLD) [40] corrects the direction of global model training by using a control variable (reduction variance), that solves the client-drift problem caused by local SGD iterations on Non-IID datasets. In the FedNova scheme [19], a regularization is introduced based on the contribution of the local gradients, limiting the bias problem for the global model caused by excessive local SGD iterations.

Building on the FedNova framework, our proposed FedVeca method introduces a strategy to adaptively regulate the local step size, which is contingent on the positive degree of the local gradient at each node. This strategy aims to modulate both the direction and magnitude of the local gradient vector. In the subsequent section, we formulate the training procedure and objectives of FedVeca, providing a proof of convergence along with the corresponding algorithm.

III. Preliminaries and Definitions

In this section, we introduce some related preliminaries and our definitions. For convenience, we assume that all vectors are column vectors in this paper and use $\boldsymbol{w}^T$ to denote the transpose of $\boldsymbol{w}$. We use "$\triangleq$" to denote "is defined to be equal to" and use $\|.\|$ to denote the L norm.

A. Generalized Update Rules of FL

Assume that we have N clients with local train datasets $\mathcal{D}_1, \mathcal{D}_2, \ldots, \mathcal{D}_i, \ldots, \mathcal{D}_N$, where i denotes the client index. FL performs the training process on these local train datasets under generalized update rules [19]. In the generalized update rules, the training process has K ($K \geq 1$) communication rounds and the server has global model parameters $\boldsymbol{w}_k$, where $k = 0, 1, 2, \ldots, K$ denotes the round index. In k round, each client i has $\tau_{(k,i)}$ ($\tau_{(k,i)} \geq 1$) local SGD iterations and its local model parameters $\boldsymbol{w}^{\lambda}_{(k,i)}$, where $\lambda = 0, 1, 2, \ldots, \tau_{(k,i)}$ denotes the local SGD iteration index. When FL begins ($k = 0$), the server initializes the global model parameters $\boldsymbol{w}_0$ and sends it to all clients. At $\lambda = 0$, the local model parameters for all clients are received from the server that $\boldsymbol{w}^0_{(k,i)} = \boldsymbol{w}_k$. For $0 \leq \lambda \leq \tau_{(k,i)} - 1$, the gradients $\nabla F_i(\boldsymbol{w}^{\lambda}_{(k,i)})$ are computed according to the local loss function $F_i(\boldsymbol{w})$ and the local model parameters $\boldsymbol{w}^{\lambda}_{(k,i)}$, and $\boldsymbol{w}^{\lambda}_{(k,i)}$ are updated by local SGD update rule which is

$$\boldsymbol{w}^{\lambda+1}_{(k,i)} - \boldsymbol{w}^{\lambda}_{(k,i)} = -\eta \nabla F_i \left(\boldsymbol{w}^{\lambda}_{(k,i)} \right), \tag{1}$$

where $\eta > 0$ denotes the learning rate which is used in SGD. In this paper, η is fixed with a pre-specified value and is equal across all clients.

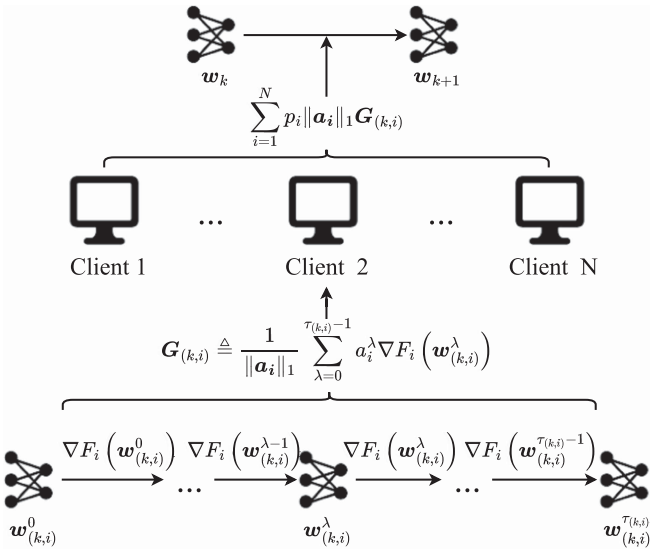

Fig. 3. Generalized update rules in the kth round.

For each node i, the total gradient on the collection of data samples at this node is

$$\boldsymbol{G}_{(k,i)} \triangleq \frac{1}{\|\boldsymbol{a_i}\|_1} \sum_{\lambda=0}^{\tau_{(k,i)}-1} a_i^\lambda \nabla F_i\left(\boldsymbol{w}_{(k,i)}^\lambda\right), \tag{2}$$

where $\boldsymbol{a}_i \in \mathbb{R}^{\tau_{(k,i)}}$ is a non-negative vector and defines how stochastic gradients are locally accumulated, and a_i^λ is the element with index λ in the vector $\boldsymbol{a}_i$. Under this rule, $\boldsymbol{G}_{(k,i)}$ denotes a locally normalized gradient at node i.

After one or multiple local SGD iterations, a global step is performed through the parameter server to update the global model parameters, which is

$$\boldsymbol{w}_{k+1} - \boldsymbol{w}_k = -\eta \sum_{i=1}^{N} p_i \|\boldsymbol{a_i}\|_1 \boldsymbol{G}_{(k,i)}. \tag{3}$$

We define $D_i \triangleq |\mathcal{D}_i|$ and $D \triangleq \sum_{i=1}^{N} D_i$, where $|.|$ denotes the size of the set, and the weight p_i is equal to D_i/D.

At the end of each round $k = 0, 1, 2, \ldots, K$, the aggregator estimates the global loss function that we define

$$F\left(\boldsymbol{w}_{k+1}\right) \triangleq \sum_{i=1}^{N} p_i F_i\left(\boldsymbol{w}_k^{i,\lambda=\tau_i}\right), \tag{4}$$

where $F_i(\boldsymbol{w}_k^{i,\lambda=\tau_i})$ is local loss function which is computed from $\boldsymbol{w}_k^{i,\lambda=\tau_i}$ and local training dataset at work node i. It is important to note that the loss function values estimated during the training phase serve merely as references. Accurate assessment of the loss function should be performed directly on the test dataset.

We define that each round k includes one global update step and $\tau_{(k,i)}$ local SGD iterations at each node i. In the kth round, the generalized update rules are shown in Fig. 3. Later in this section, we will present two algorithms that are based on these general update rules.

B. FedAvg and FedNova Algorithms

In the FedAvg and FedNova algorithms, each client performs E epochs (traversals of their local dataset) of local SGD with a mini-batch size B. Thus, if a client has D_i local data samples, the number of local SGD iterations is $\tau_{(k,i)} = \lfloor E(D_i/B) \rfloor$, which can vary widely across clients.

1) The FedAvg Algorithm: The FedAvg algorithm provided the basic principles for FL in the Non-IID datasets. As mentioned in [17], SGD can be seen as an approximation to Deterministic Gradient Descent (DGD). Therefore, for convergence analysis of federated optimization, it is generally assumed that the number of local updates is the same across all clients (that is, $\tau_{(k,i)} = \tau$ for all clients i).

According to the definition in Section III-A, FedAvg has $\boldsymbol{a}_i = [1, 1, \ldots, 1]$ and $\|\boldsymbol{a}_i\|_1 = \tau$, the update rule specializes (2) and (3) and is as follows:

$$\begin{cases} \boldsymbol{G}_{(k,i)} \triangleq \sum_{\lambda=0}^{\tau-1} \nabla F_i\left(\boldsymbol{w}_{(k,i)}^\lambda\right), \\ \boldsymbol{w}_{k+1} - \boldsymbol{w}_k = -\eta \sum_{i=1}^{N} p_i \boldsymbol{G}_{(k,i)}. \end{cases} \tag{5}$$

Therefore, FedAvg is a simplified special case of the generalized update rules. And next, we will introduce an algorithm that improves on the generalized update rules.

2) The FedNova Algorithm: The FedNova algorithm has the same definition of $\boldsymbol{a}_i = [1, 1, \ldots, 1]$ as the FedAvg algorithm, but FedNova does not impose a special constraint on the number of local SGD iterations, thus $\|\boldsymbol{a_i}\|_1 = \tau_{(k,i)} = \lfloor E(D_i/B) \rfloor$. FedNova still uses the local SGD update rule of (1), the locally normalized gradient of (2) and the global aggregation rule is as follows:

$$\begin{cases} \boldsymbol{G}_{(k,i)} \triangleq \frac{1}{\tau_{(k,i)}} \sum_{\lambda=0}^{\tau_{(k,i)}-1} \nabla F_i\left(\boldsymbol{w}_{(k,i)}^\lambda\right), \\ \boldsymbol{w}_{k+1} - \boldsymbol{w}_k = -\eta \tau_k \boldsymbol{d}_k, \end{cases} \tag{6}$$

where $\tau_k \triangleq \sum_{i=1}^{N} p_i \tau_{(k,i)}$ is the aggregated value of the number of SGD iterations on each client and $\boldsymbol{d}_k \triangleq \sum_{i=1}^{N} p_i \boldsymbol{G}_{(k,i)}$ is the normalized averaging gradient.

Equation (6) can be seen as a decomposition of (3), where $-\boldsymbol{d}_k$ denotes the direction of global gradient descent and $\eta\tau_k$ denotes the step size of global gradient descent.

C. Learning Objective and FedVeca Method

In FL, for each global model $\boldsymbol{w}_k$ there is its corresponding loss function $F(\boldsymbol{w}_k)$. The objective of learning is to find optimal global model $\boldsymbol{w}_K^*$ to minimize $F(\boldsymbol{w}_K^*)$

$$\boldsymbol{w}_K^* \triangleq \arg\min F(\boldsymbol{w}_K), \tag{7}$$

thus the value of $F(\boldsymbol{w}_k)$ should decrease as k increases. In the kth round, we simplify this objective to find an optimal global model $\boldsymbol{w}_{k+1}^*$ that is

$$\boldsymbol{w}_{k+1}^* \triangleq \arg\min \left\{F(\boldsymbol{w}_{k+1}^*) - F(\boldsymbol{w}_k)\right\}. \tag{8}$$

In FedVeca, we use the update rule (6) of FedNova and define the local gradient $\boldsymbol{G}_{(k,i)}$ as an averaged bi-directional vector whose step size is the value of $\tau_{(k,i)}$. In addition to that, we allow a heterogeneous value of step size at each client and analyze the relationship between our objective $\boldsymbol{w}_{k+1}^*$ and FedNova's global

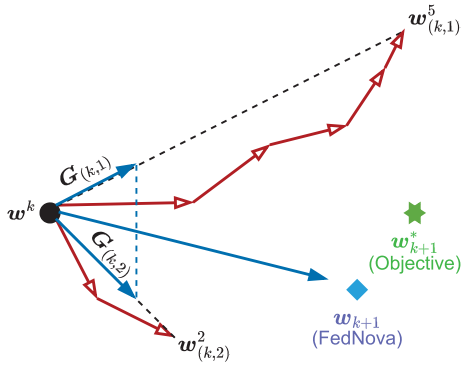

Fig. 4. Illustrative diagram of the learning problem in the kth round.

model $\boldsymbol{w}_{k+1}$. As shown in Fig. 4, we assume that there are two clients, the Client 1 with 5 local SGD iterations (step size of 5) and the Client 2 with 2 local SGD iterations (steo size of 2). On these two clients, the local gradient $\boldsymbol{G}_{(k,1)}$ on Client 1 is more closely aligned with the direction of $\boldsymbol{w}_k - \boldsymbol{w}^*_{k+1}$, making it a positive vector. In contrast, the local gradient on Client 2 deviates more significantly from the direction of $\boldsymbol{w}_k - \boldsymbol{w}^*_{k+1}$, making it a negative vector. The mathematical quantification underlying the classification of these two types of vectors will be presented in detail in Theorem 2 of Section IV-A. The direction of the global gradient $\boldsymbol{w}_k - \boldsymbol{w}_{k+1}$ is then determined by the sum of these two vectors.

In order to make the global model $\boldsymbol{w}_{k+1}$ close to our objective $\boldsymbol{w}^*_{k+1}$, two factors need to be controlled: direction and step size of the global gradient. According to the definition of d_k and τ_k in Section III-B2, we know that the averaged bi-directional vector $\boldsymbol{G}_{(k,i)}$ and it's step size $\tau_{(k,i)}$ control the direction and step size of the global gradient, respectively. Moreover, we can see in (6) that there is a certain connection between $\boldsymbol{G}_{(k,i)}$ and $\tau_{(k,i)}$, thus a certain connection between the global model $\boldsymbol{w}_{k+1}$ and the bi-directional vector's step size $\tau_{(k,i)}$.

We set that $\tau_{(k,i)}$ satisfy $\tau_{(k,i)} \neq \lfloor E(D_i/B) \rfloor$ and can be different in each round k and at each client i. Therefore, a natural question is how to determine the optimal values of $\tau_{(k,i)}$ at each node i, so that to get the global objective $\boldsymbol{w}^*_{k+1}$. Based on this, if we want to get the optimal $\boldsymbol{w}^*_K$, we need adaptive control of $\tau_{(k,i)}$ for each round k. And next, we will describe how our FedVeca algorithm achieves this objective.

IV. Convergence Analysis and FedVeca Algorithm

In this section, we first analyze the convergence of FedVeca and get an upper bound of $\tau_{(k,i)}$. Then, we use this upper bound to design our algorithm to adaptively control the value of $\tau_{(k,i)}$ in each round k and at each word node i.

A. Convergence Analysis

We analyze the convergence of FedVeca in this subsection and find an upper bound of $F(\boldsymbol{w}_{k+1}) - F(\boldsymbol{w}_k)$. To facilitate the analysis, we first introduce the definition of the global gradient $\nabla F(\boldsymbol{w}_k)$ which is the optimal gradient of $\boldsymbol{w}^*_{k+1} - \boldsymbol{w}_k$ on all training datasets, and we have $\|\boldsymbol{d}_k\| \leq \|\nabla F(\boldsymbol{w}_k)\|$ by comparing (6) and (8). However, the global gradient $\nabla F(\boldsymbol{w}_k)$ can not be calculated directly from all training datasets in the kth round of FL. Thus, at the end of each round $k = 1, 2, \ldots, K$, the server estimates the last global gradient that we define

$$\nabla F(\boldsymbol{w}_{k-1}) \triangleq \sum_{i=1}^{N} p_i \nabla F_i\left(\boldsymbol{w}_{k-1}\right), \tag{9}$$

where $\nabla F_i(\boldsymbol{w}_{k-1})$ is the local gradient of $\boldsymbol{w}_{k-1}$ at client i (can be calculated directly from the local training dataset). We assume the global gradient $\nabla F(\boldsymbol{w}_{k-1})$ is convex and all the assumptions in this paper are as follows:

Assumption 1 (Global Lipschitz smooth): Each global gradient is convex and Lipschitz smooth for $k \in [0, k-1]$, that is,

$$\|\nabla F(\boldsymbol{w}_{k+1}) - \nabla F(\boldsymbol{w}_k)\| \leq L\|\boldsymbol{w}_{k+1} - \boldsymbol{w}_k\|. \tag{10}$$

Assumption 2: For $k \in [0, k-1]$, we have

$$\|\boldsymbol{d}_k\| \leq \|\nabla F\left(\boldsymbol{w}_k\right)\|. \tag{11}$$

Assumption 3 (Local Lipschitz smooth): For any local gradient $\nabla F_i(\boldsymbol{w}^{\lambda}_{(k,i)})$ and $\lambda \in [0, \tau_{(k,i)} - 1]$, there exist constants $\beta_{(k,i)} \geq 0$ such that

$$\left\|\nabla F_i\left(\boldsymbol{w}_k\right) - \nabla F_i\left(\boldsymbol{w}^{\lambda}_{(k,i)}\right)\right\| \leq \beta_{(k,i)}\left\|\boldsymbol{w}_k - \boldsymbol{w}^{\lambda}_{(k,i)}\right\|. \tag{12}$$

Assumption 4 (Bounded gradient error): For all local gradients, $s \in [0, \lambda]$ and $\lambda \in [1, \tau_{(k,i)} - 1]$, there exist constants $\delta_{(k,i)} \geq 0$ such that

$$\left\|\sum_{s=0}^{\lambda-1} \nabla F_i\left(\boldsymbol{w}^{\lambda}_{(k,i)}\right)\right\|^2 \leq \delta_{(k,i)} \sum_{s=0}^{\lambda-1} \left\|\nabla F(\boldsymbol{w}_k)\right\|^2. \tag{13}$$

Under Assumptions 1 to 4 and the update rule of FedVeca, we get the upper bound of $F(\boldsymbol{w}_{k+1}) - F(\boldsymbol{w}_k)$.

Theorem 1: In the kth round for $k \in [0, k-1]$, when $\eta\tau_k L - 1 \geq 0$, we have

$$\frac{F\left(\boldsymbol{w}_{k+1}\right) - F\left(\boldsymbol{w}_k\right)}{\left\|\nabla F\left(\boldsymbol{w}_k\right)\right\|^2} \leq \eta\tau_k \left(\frac{1}{4}\eta \sum_{i=1}^{N} p_i \left(\tau_{(k,i)}\left(2L + A_{(k,i)}\right) - A_{(k,i)}\right) - 1\right), \tag{14}$$

where $A_{(k,i)} \triangleq \eta\beta^2_{(k,i)}\delta_{(k,i)}$ is a variable that varies with $\delta_{(k,i)}$ and $\beta^2_{(k,i)}$.

Proof: We first perform an inequality transformation on $F(\boldsymbol{w}_{k+1}) - F(\boldsymbol{w}_k)$ and then convert the equation to contain only the similar items of $\|\nabla F(\boldsymbol{w}_k)\|^2$. For details, see Appendix A, available online. □

According to Theorem 1, if the model can converge under our method, then it should be consistent with $F(\boldsymbol{w}_{k+1}) - F(\boldsymbol{w}_k) < 0$. It is equivalent to $\frac{1}{4}\eta\sum_{i=1}^{N} p_i(\tau_{(k,i)}(2L + A_{(k,i)}) - A_{(k,i)}) < 1$, and this inequality is related to $\tau_{(k,i)}$. We can see from (14) that the parameters in $A_{(k,i)}$ (the effect of Non-IID) do not play a role when $\tau_{(k,i)} = 1$. Therefore, we

set the lower bound of $\tau_{(k,i)} > 1$, and have another theorem to determine the upper bound on $\tau_{(k,i)}$.

Theorem 2: In the kth round for $k \in [0, k-1]$, the model converges when

$$\tau_{(k,i)} \leq \frac{A_{(k,i)}}{A_{(k,i)} - \alpha_k \min_i A_{(k,i)}}. \tag{15}$$

for each client i, where α_k is a real number and has $\alpha_k \in (0, \frac{2L}{\min_i A_{(k,i)}})$ (when $\frac{2L}{\min_i A_{(k,i)}} < 1$) and $\alpha_k \in (0, 1)$ (when $\frac{2L}{\min_i A_{(k,i)}} > 1$).

Proof: For details, see Appendix B, available online. □

According to (15), we can define the positive and negative direction of the bid-irectional vector (in Section III-C) based on the gap between the value of $A_{(k,i)}$ and the value of $\alpha_k \min_i A_{(k,i)}$. Further, we can obtain the relationship between the bounds on the step size $\tau_{(k,i)}$ and the corresponding direction of the bi-directional vector. We next describe how our FedVeca algorithm adaptively controls $\tau_{(k,i)}$ to achieve our objective in Section III-C.

B. FedVeca Algorithm

FL is required to perform more local computations in each communication round [3]. Thus, according to the convex assumption and Theorem 2, we predict that

$$\tau_{(k+1,i)} = \left\lfloor \frac{A_{(k,i)}}{A_{(k,i)} - \alpha_k \min_i A_{(k,i)}} \right\rfloor. \tag{16}$$

That is, we use the results of the previous round to predict the number of local SGD iterations on each node in the next round. When $\tau_{(k+1,i)}$ is calculated as $\tau_{(k+1,i)} = 1$, to keep $\tau_{(k+1,i)} > 1$, we reset $\tau_{(k+1,i)} = 2$ in our algorithm. Moreover, when $\min_i A_{(k,i)}$ is determined in the kth round, we can choose a suitable value of α_k to make the global model for the next round close to our global objective (which is mentioned in Section III-C).

As mentioned earlier, the local update runs on the clients and the global aggregation is performed with the assistance of the parameter server. The complete process of the parameter server and each client is presented in Algorithms 1 and 2, respectively, where Lines 5-8 of Algorithm 2 are local updates and the rest are considered as part of initialization, global aggregation and computing the value of $\tau_{(k+1,i)}$. We assume that the server initiates the learning process, then initializes $\tau_{(0,i)}$, $\boldsymbol{w}_0$ and $\nabla F(\boldsymbol{w}_0)$ and sends it to all clients. The input consists of a given K, an α_k that transforms with round k, and a η that is fixed for all rounds, and finally FedVeca algorithm gives the global model $\boldsymbol{w}_K$ obtained for the final round K.

1) Handling of SGD: When using SGD with FedVeca algorithm at all clients, all their gradients are computed on mini-batches. Each SGD step corresponds to a model update step where the gradient is computed on a mini-batch of local training data in Line 6 of Algorithm 2. The mini-batch changes for every step of the local iteration, i.e., for each new local iteration, a new mini-batch of a given size is randomly selected from the local training data. After $\tau_{(k,i)}$ local update steps, we average the corresponding model gradient vectors on these training data and get obtain the direction vector $G_{(k,i)}$ of SGD at node i (Line 11 of Algorithm 2).

Algorithm 1: Procedure at the Server.

Input: Total round K, parameter α_k, learning rate η and minimum loss value F_m
Output: $\boldsymbol{w}_K$
1: Initialize $k = 0$, $F_m = \infty$, $\tau_{(k,i)}$ and $\boldsymbol{w}_k$;
2: **repeat**
3: Send $\tau_{(k,i)}$ and $\boldsymbol{w}_k$ to all clients;
4: Receive $\nabla F_i(\boldsymbol{w}_k)$ and $F_i(\boldsymbol{w}_{(k,i)}^{\lambda=\tau_i})$ from each node i;
5: Compute $\nabla F(\boldsymbol{w}_k)$ and $F(\boldsymbol{w}_{k+1})$ according to (9) and (4), respectively;
6: Receive $\boldsymbol{G}_{(k,i)}$ from each node i;
7: Compute $\boldsymbol{w}_{k+1}$ according to (6);
8: **if** $F(\boldsymbol{w}_{k+1}) \leq F_m$ **then**
9: Set $F_m \leftarrow F(\boldsymbol{w}_{k+1})$
10: **else**
11: Set $\boldsymbol{w}_{k+1} \leftarrow \boldsymbol{w}_k$
12: **end if**
13: **if** $k \geq 1$ **then**
14: Send $\nabla F(\boldsymbol{w}_{k-1})$ to all clients;
15: Receive $\beta_{(k,i)}$ and $\delta_{(k,i)}$;
16: **if** $k = 1$ **then**
17: Estimate $L_{k-1} \leftarrow \|\nabla F(\boldsymbol{w}_{k-1})\| / \|\boldsymbol{w}_{k-1}\|$;
18: **else**
19: Estimate $L_{k-1} \leftarrow \|\nabla F(\boldsymbol{w}_{k-1}) - \nabla F(\boldsymbol{w}_{k-2})\| / \|\boldsymbol{w}_{k-1} - \boldsymbol{w}_{k-2}\|$;
20: **end if**
21: Estimate $L \leftarrow \max_k L_{k-1}$;
22: Compute the value of $\tau_{(k+1,i)}$ according to (16);
23: Compute τ_{k+1} according to (6);
24: **if** $\tau_{(k+1,i)} \leq 1$ **then**
25: Set $\tau_{(k+1,i)} = 2$;
26: **end if**
27: **end if**
28: $k \leftarrow k + 1$;
29: **if** $k = 1$ **then**
30: Set $\tau_{(k,i)} = \tau_{(k-1,i)}$;
31: **end if**
32: **if** $k = K$ **then**
33: Set $STOP$ flag, and send it to all clients;
34: **end if**
35: **until** $STOP$ flag is set

When the parameter server receives $G_{(k,i)}$ from each client, it updates $\boldsymbol{w}_{k+1}$ (Line 7 of Algorithm 1) according to the second term in (6) where τ_k is calculated based on $\tau_{(k,i)}$ in Line 23 of Algorithm 1. With the value of $F_i(\boldsymbol{w}_{(k,i)}^{\lambda=\tau_i})$ received from the clients (Line 4 of Algorithm 1) and (4), we can estimate the value of the loss function of $\boldsymbol{w}_{k+1}$ (Line 5 of Algorithm 1). Compare it with the set minimum loss function value F_m, if it is less than, then accept this global update and assign the value to F_m, and if not, then do not accept this global update (Line 8-12 of Algorithm 1). When the program proceeds to the set number of rounds k, the final model parameter $\boldsymbol{w}_K$ is obtained at the server

Algorithm 2: Procedure at Each Client i.

1: Initialize $\lambda \leftarrow 0$, $k = 0$ and $\boldsymbol{w}^{\lambda}_{(k,i)}$;
2: **repeat**
3: Receive $\tau_{(k,i)}$ and $\boldsymbol{w}_k$ from the server;
4: Set $\boldsymbol{w}^{\lambda}_{(k,i)} \leftarrow \boldsymbol{w}_k$;
5: **for** $\lambda = 0, 1, 2, \ldots, \tau_{(k,i)} - 1$ **do**
6: Compute $\nabla F_i(\boldsymbol{w}^{\lambda}_{(k,i)})$;
7: Perform local update of $\boldsymbol{w}^{\lambda}_{(k,i)}$ according to (1);
8: **end for**
9: Compute $\nabla F_i(\boldsymbol{w}_k)$ and $F_i(\boldsymbol{w}^{\lambda=\tau_i}_{(k,i)})$;
10: Send $\nabla F_i(\boldsymbol{w}_k)$ and $F_i(\boldsymbol{w}^{\lambda=\tau_i}_{(k,i)})$ to parameter server;
11: Compute $\boldsymbol{G}_{(k,i)}$ according to (6);
12: Send $\boldsymbol{G}_{(k,i)}$ to parameter server;
13: **if** $k \geq 1$ **then**
14: Receive $\nabla F(\boldsymbol{w}_{k-1})$ from the server;
15: Estimate $\beta^{\lambda}_{(k,i)} \leftarrow \left\|\nabla F_i(\boldsymbol{w}_k) - \nabla F_i(\boldsymbol{w}^{\lambda}_{(k,i)})\right\| / \left\|\boldsymbol{w}_k - \boldsymbol{w}^{\lambda}_{(k,i)}\right\|$ for $\lambda \in [0, \tau_{(k,i)} - 1]$;
16: Estimate $\beta_{(k,i)} \leftarrow \max_{\lambda} \beta^{\lambda}_{(k,i)}$;
17: Estimate $\delta^{\lambda}_{(k,i)} \leftarrow \left\|\sum_{s=0}^{\lambda} \nabla F_i(\boldsymbol{w}^{s}_{(k,i)})\right\|^2 / \sum_{s=0}^{\lambda} \|\nabla F(\boldsymbol{w}_{k-1})\|^2$ for $\lambda \in [1, \tau_{(k,i)} - 1]$;
18: Estimate $\delta_{(k,i)} \leftarrow \max_{\lambda} \delta^{\lambda}_{(k,i)}$;
19: Send $\beta_{(k,i)}$ and $\delta_{(k,i)}$ to parameter server;
20: **end if**
21: $k \leftarrow k + 1$;
22: **until** $STOP$ flag is received

in Lines 32-34 of Algorithm 1. Then we set the $STOP$ flag to stop the server-side program and send the flag to all clients to stop the local program.

2) Estimation of Parameters: According to Assumptions 1, 3 and 4, we have three parameters L, $\beta_{(k,i)}$ and $\delta_{(k,i)}$ that need to be estimated in real-time during the learning process. L is the parameter describing the smooth of global gradients and should be ensured to satisfy Assumption 1, so we perform the estimation of its maximum value (in all rounds for $k > 0$) on the server side in Lines 21 of Algorithm 1. Estimating L requires the global gradient of the previous round, which does not exist when $k = 0$, so we give this program a delay of one round in Lines 19 of Algorithm 1. We know that L is not involved in the calculation of the new $\tau_{(k,i)}$ in (16) and used to satisfy that premise $\eta\tau_k L - 1 \geq 0$ of Theorem 1, so a delay of one round in its estimation L_{k-1} has little effect on the final result.

Therefore FedVeca algorithm should estimate the value of $A_{(k,i)}$. The expression of $A_{(k,i)}$ includes parameters $\beta_{(k,i)}$ and $\delta_{(k,i)}$ which need to be estimated in practice on the client side. Before starting the estimation, we need to calculate $F_i(\boldsymbol{w}_k)$ on the local training dataset in Line 9 of Algorithm 2. Then, the estimation of $\beta_{(k,i)}$ is computed from $\lambda = 0$ to $\lambda = \tau_{(k,i)} - 1$ and the estimation of $\delta_{(k,i)}$ is computed from $\lambda = 1$ to $\lambda = \tau_{(k,i)} - 1$ in Lines 15 and 17 of Algorithm 2. Finally, the largest estimations of $\beta_{(k,i)}$ and $\delta_{(k,i)}$ from these iterations will be selected (Lines 16 and 18 of Algorithm 2) and sent to the parameter server when $k > 0$ (Lines 19 of Algorithm 2).

3) Computing $\tau_{(k+1,i)}$: When $k = 0$, the server does not receive the relevant parameters ($\beta_{(k,i)}$ and $\delta_{(k,i)}$) from the clients to calculate the value of $\tau_{(k+1,i)}$. This is because the value of L cannot be estimated at this point, and we have no way to choose the value of α_k to input. When $k \geq 1$, the value of $\tau_{(k+1,i)}$ can be computed according to (16) in Line 22 of Algorithm 1. Then, to satisfy $\tau_{(k+1,i)} > 1$ in Section IV-A, we set $\tau_{(k+1,i)} = 2$ for $\tau_{(k+1,i)} \leq 1$ in Lines 24-26 of Algorithm 1.

After computing the value of $\tau_{(k+1,i)}$, we can compute the global step size τ_{k+1} for the next round ($k \leftarrow k + 1$) of aggregation in Line 23 of Algorithm 1. Combining our obtained estimation of L, we can verify the premise $\eta\tau_k L - 1 \geq 0$ of Theorem 1. The verification results and the test results of FedVeca algorithm will be shown in Section V.

V. Performance Analysis

To evaluate FedVeca algorithm, the experiments focused on the qualitative and quantitative analysis of general properties under the setup of our prototype system and the simulation of IID and Non-IID datasets.

A. Setup

We first evaluate the general performance of FedVeca algorithm, thus we conducted experiments on a networked prototype system with five clients. The prototype system consists of five Raspberry Pi (version 4B) devices and one laptop computer, which are all interconnected via Wi-Fi in an office building. The laptop computer has an aggregator and implements FedVeca algorithm of parameter server, and the Raspberry Pi device implements FedVeca algorithm of client. All of these five clients have model training with local datasets and different simulated dataset distributions of IID and Non-IID.

1) Baselines: We compare FedVeca method with the following baseline approaches:

- Centralized SGD where the entire training dataset is stored on a single device and the model is trained directly on that device using a standard (centralized) SGD procedure.
- Standard FL approach which uses FedAvg algorithm and has the fixed (non-adaptive) value of $\tau_{(k,i)}$ at all clients in all rounds.
- Novel FL approach which uses FedNova algorithm and has the same value of $\tau_{(k,i)}$ as FedAvg algorithm at all clients in all rounds.
- Novel FL approach which uses FedProx algorithm and has the same value of $\tau_{(k,i)}$ as FedAvg algorithm at all clients in all rounds.
- Novel FL approach which uses SCAFFOLD algorithm and has the same value of $\tau_{(k,i)}$ as FedAvg algorithm at all clients in all rounds.

For a fair comparison, we first run FedVeca algorithm and then record the total number of local iterations τ_{all} run by all nodes in all K rounds. Then for the centralized SGD, we train τ_{all} iterations and each iteration randomly picks a batch of the same size B (fixed) as FedVeca algorithm, then we use this model for evaluating the convergence of other trained models. When evaluating the FedAvg algorithm and FedNova algorithm,

we compute the average epoch $E_{avg} = \tau_{all}/K \times B/D$ of all rounds, and assign a fixed $\tau_{(k,i)} = \lfloor E_{avg}(D_i/B) \rfloor$ to the clients for each round.

2) Model and Datasets: We evaluate the training of two different models on two different datasets, which represent both small and large models and datasets. The models include squared-Support Vector Machine (SVM)[1] (we refer to as SVM in short in the following) and deep Convolutional Neural Network (CNN).[2] Among them, the loss functions for SVM satisfy Assumption 1, whereas the loss functions for CNN are non-convex and thus do not satisfy Assumption 1.

SVM is trained on the original MNIST (which we refer to as MNIST in short in the following) dataset [41], which contains the gray-scale images of 7×10^4 handwritten digits (6×10^4 for training and 10^4 for testing).

CNN is trained using SGD on two different datasets: the MNIST dataset and the CIFAR-10 dataset [42], and the CIFAR-10 dataset includes 6×10^4 color images (5×10^4 for training and 10^4 for testing) associated with a label from 10 classes. A separate CNN model is trained on each dataset, to perform multi-class classification among the 10 different labels in the dataset.

3) Simulation of Dataset Distribution: To simulate the dataset distribution, we set up two different Non-IID cases and a standard IID case.

- *Case 1 (IID):* Each data sample is randomly assigned to a client, thus each client has a uniform (but not full) information.
- *Case 2 (Non-IID):* All the data samples in each client have the same label, which means that the dataset on each node has its unique features.
- *Case 3 (Non-IID):* Data samples with the first half of the labels are distributed to the first half of the clients as in Case 1, the other samples are distributed to the second half of the clients as in Case 2.

4) Training and Control Parameters: In all our experiments, we set the maximum $\tau_{(k,i)}$ value $\max \tau_{(k,i)} = 50$ to reduce the impact of errors in the experiment. Unless otherwise specified, we set the control parameter α_k to a fixed value $\alpha_k = 0.95$ for all rounds. And we manually select the total round $K = 100$ and the learning rate fixed at $\eta = 0.01$, which is acceptable for our learning process. Except for the instantaneous results, the others are the average results of 10 independent experiment runs.

B. Results

1) Loss and Accuracy Values: In our first set of experiments, the SVM and CNN models were trained on the prototype system with Case 3, and the results $\boldsymbol{w}_k$ of each round will be calculated on the test dataset with the loss function values and prediction accuracy values which we refer to as loss and accuracy in short in the following.

[1]The squared-SVM has a fully connected neural network, and outputs a binary label that corresponds to whether the digit is even or odd.

[2]The CNN has two $5 \times 5 \times 32$ convolution layers, two 2×2 MaxPoll layers, a 1568×256 fully connected layer, a 256×10 fully connected layer, and a softmax output layer with 10 units.

TABLE I
MODEL LOSS VALUES AND ACCURACIES FOR EACH BASELINE ALGORITHM OBTAINED IN **CASE 3** AT THE END OF THE SET TRAINING ROUNDS ($K = 100$)

Model	Algorithm	Loss($\times 10^{-2}$)	Acc (%)	Centralized
SVM + MNIST	FedAvg	23.13	85.75	Loss & Acc
	FedNova	23.44	85.32	
	FedProx	23.14	85.76	
	SCAFFOLD	**22.07**	**86.92**	22.06 & 86.87
	FedVeca	22.22	86.63	
CNN + MNIST	FedAvg	32.32	90.67	Loss & Acc
	FedNova	33.05	90.37	
	FedProx	32.76	90.47	
	SCAFFOLD	**20.80**	**93.86**	22.67 & 93.43
	FedVeca	24.25	93.04	
CNN + CIFAR	FedAvg	170.50	38.62	Loss & Acc
	FedNova	171.63	38.10	
	FedProx	170.77	38.17	
	SCAFFOLD	163.28	41.23	160.88 & 42.38
	FedVeca	**160.32**	**42.41**	

We record the loss and accuracy on the SVM and CNN classifiers with FedVeca algorithm (with adaptive $\tau_{(k,i)}$), and compare them to baseline approaches, where the centralized case only has one optimal value as the training result and we show a flat line across different rounds for the ease of comparison, the results are shown in Fig. 5.

It can be observed from Fig. 5 that the curves of loss and accuracy values of FedVeca algorithm fluctuate less on the SVM model, while they have larger fluctuations on the CNN model. This is because the loss function of the SVM model satisfies Assumption 1 of a convex function and the loss function of the CNN model is non-convex, as we mentioned in Section V-A2. Since the loss function of the CNN model is non-convex, the variation (orientation and size) of the local model parameters are unstable when making updates, so it is unreliable for us to describe the overall variation of global model parameters with the estimated values of L, $\beta_{(k,i)}$ and $\delta_{(k,i)}$. Further, it is less useful to predict the number of local SGD iterations for the next round based on the new value of $\tau_{(k,i)}$ calculated from these estimated values. However, within the specified number of rounds ($K = 100$), FedVeca algorithm is the first to reach the loss and accuracy of the centralized SGD approach compared to FedAvg, FedNova and FedProx on all models and datasets (nearly simultaneous with SCAFFOLD), demonstrating the convergence and faster convergence speed of FedVeca algorithm with Non-IID datasets.

As demonstrated in Table I, on the CIFAR dataset, the FedVeca algorithm outperforms the FedAvg, FedNova, FedProx and SCAFFOLD algorithm in terms of loss and accuracy improvement in the final trained model, thereby underscoring the effectiveness of the algorithm. As for the MNIST dataset, We elaborate that although FedVeca exhibits marginally lower performance compared to the state-of-the-art algorithm SCAFFOLD, it nonetheless demonstrates a substantial performance advantage over other well-established algorithms, including FedAvg, FedNova, and FedProx. This underscores FedVeca's reliability and its capacity to deliver competitive results across various federated learning scenarios, even when compared with advanced methods.

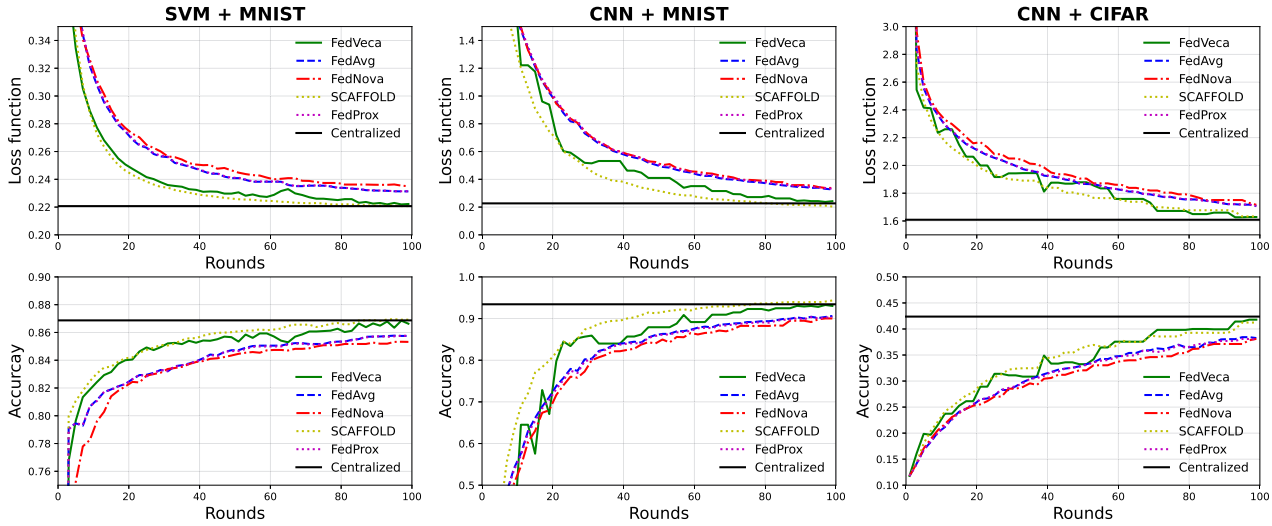

Fig. 5. Loss function values and classification accuracy values in Case 3. The curves show the results from FedVeca algorithm and the baselines with the different models and datasets [41], [42].

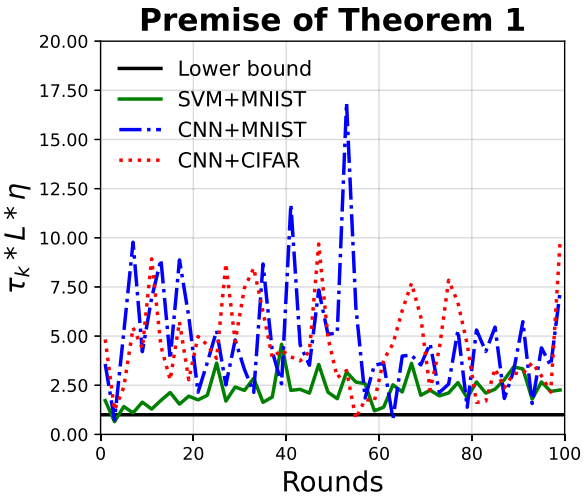

Fig. 6. The value of $\eta\tau_k L$ in Case 3. The curves show the results from FedVeca algorithm with the different models and datasets [41], [42].

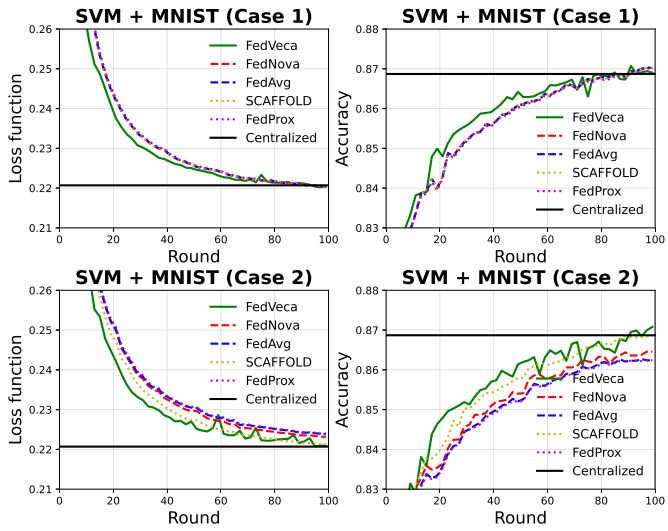

Fig. 7. Loss function and classification accuracy on the SVM model and the MNIST dataset [41] with dataset distriburion of Case 1 and Case 2.

2) Premise of Theorem 1: As described in Section IV-A, the premise for Theorem 1 to be valid is that $\eta\tau_k L - 1 \geq 0$. Therefore, we record the value of $\eta\tau_k L$ for each round within the specified number of rounds ($K = 100$) on all specified models and datasets in Case 3, the results are shown in Fig. 6.

In Fig. 6, we set the value '1' as a lower bound on the value of $\eta\tau_k L$ and use a straight line to represent. Moreover, according to the discussion in Section IV-B2, the estimated value of L_k in Algorithm 1 is delayed by one round, so we make a blank for $k = 0$ in Fig. 6. As we can see, the specified models and datasets satisfy the premise of Theorem 1 for all K rounds, except for the SVM model whose the values of $\eta\tau_k L$ are slightly smaller than the lower bound for the first few rounds on the MNIST dataset (which is within the estimation margin of error). And the SVM model has the most stable $\eta\tau_k L$ values on the MNIST dataset compared to the other specified models and datasets, which is consistent with our conclusion that non-convex loss functions cannot be described accurately as discussed in Section V-B1.

Due to the high complexity of evaluating CNN models, we focus on the SVM model in the following and provide further insights on the prototype system.

3) Dataset Distribution: We test the MNIST dataset distribution on the SVM model for the other two simulations which are Case 1 and Case 2, and the results of FedVeca algorithm and the baselines are shown in Fig. 7.

In Case 1, the curves of loss and accuracy values of FedVeca algorithm in each round overlap with the baseline methods, and both converge within $K = 100$ rounds (compared to Centralized SGD), proving that FedVeca algorithm is applicable on the IID dataset. In Case 2, FedVeca algorithm has a smaller difference in loss values and a larger difference in accuracy values in each

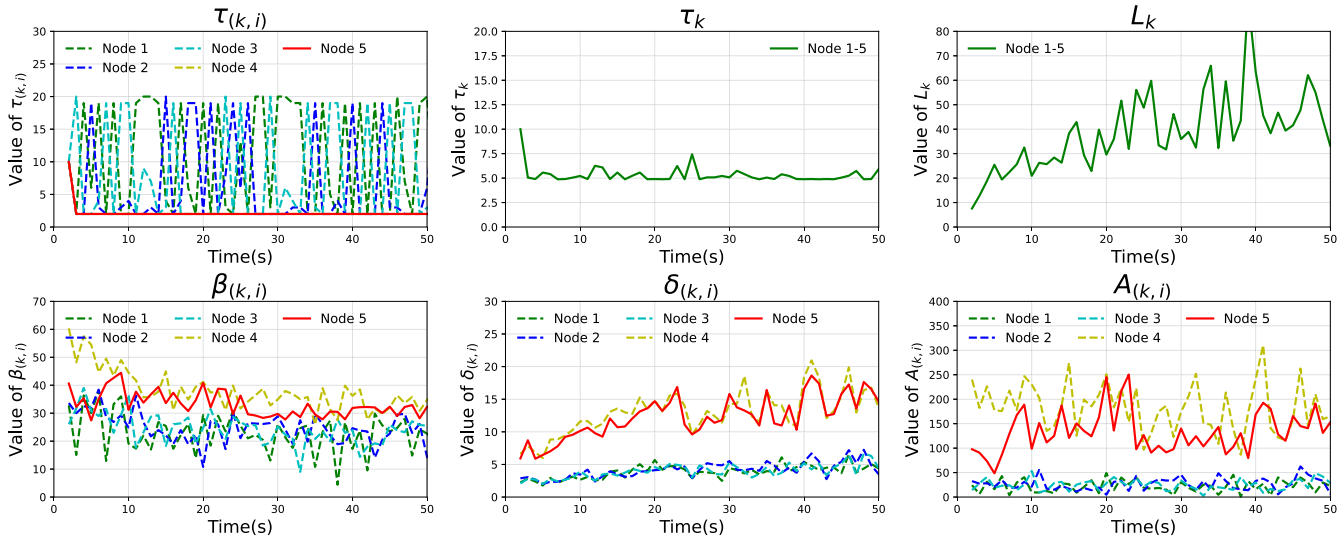

Fig. 8. Instantaneous results of a single run (on the SVM model and the MNIST dataset [41]) with FedVeca algorithm in Case 3.

round compared to the baseline methods, and both loss and accuracy values are the first to reach the convergence point. Together with the results of Case 3 we obtained in Section V-B1, we show that FedVeca algorithm is also applicable to the Non-IID dataset.

By comparing Figs. 7 and 5, we note that on the SVM model and the MNIST dataset (convex loss function), the convergence rate of the model obtained by FedVeca algorithm is essentially the same in Case 1, Case 2 and Case 3. Meanwhile, the models obtained by FedNova and FedAvg perform better in Case1 and Case2 and worse in Case3. This proves that FedVeca algorithm has good performance and stability on both IID and Non-IID datasets. Therefore, we focus on Case 3in the following and provide further insights on the prototype system.

4) Instantaneous Behavior: We study the instantaneous behavior of FedVeca algorithm for a single run on the prototype system. Results for SVM (MNIST) are shown in Fig. 8, where we record the amount of variation $\tau_{(k,i)}$, τ_k, L_k, $\beta_{(k,i)}$, $\delta_{(k,i)}$ and $A_{(k,i)}$ on the five clients, and represent only one curve for τ_k and L_k. To make the image clearer, we only captured the first 50 rounds of the show.

We can see that the value of $\tau_{(k,i)}$ fluctuates very much with round k, and the maximum number of SGD iterations in each round is uniformly distributed across clients, indicating that FedVeca algorithm can adaptively choose the optimal $\tau_{(k,i)}$ value according to the working conditions during the training process. τ_k represents the step size of global gradient descent in Section III-B2, and there is no large fluctuation in the curve of τ_k, indicating that the model converges smoothly globally despite the large difference in $\tau_{(k,i)}$ values at each client under adaptive control. For the value of L_k and Assumption 1, we know that as the number of rounds increases, the variation between the model parameters of adjacent rounds becomes smaller, which represents the convergence of the model.

According to Theorem 2, $A_{(k,i)}$ is the key variable for adaptive control of $\tau_{(k,i)}$ values and $A_{(k,i)}$ is composed of two variables, $\beta_{(k,i)}$ and $\delta_{(k,i)}$. In Fig. 8, we can see that the values of $A_{(k,i)}$ on Node 4 and Node 5 are widely spaced compared to those on the remaining three clients. This is related to the Case 3 we mentioned in Section V-A3. The data distribution on Node 4 and Node 5 are similar but differ significantly from the remaining three clients, causing differences between the $\beta_{(k,i)}$ and $\delta_{(k,i)}$ values. Therefore, FedVeca algorithm accurately quantifies these differences and adaptively optimizes the FL training process by controlling the $\tau_{(k,i)}$ values.

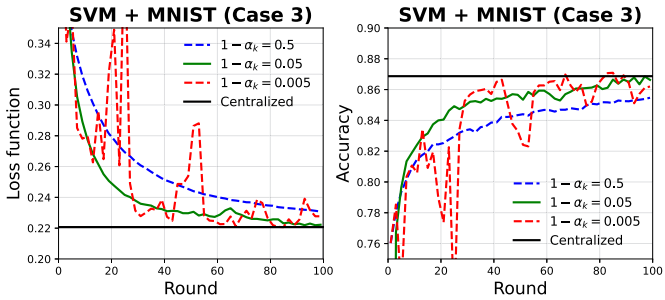

Fig. 9. Loss function and classification accuracy (on the SVM model and the MNIST dataset [41]) with the different values of α_k.

5) Sensitivity of α_k: In the setup of Section V-A4, the value of α_k is fixed for each round and we know that the maximum value of $\tau_{(k,i)}$ on all nodes in a round is related to $1-\alpha_k$ according to (15). Therefore, we choose three numbers 0.5, 0.05 and 0.005 for the values of $1-\alpha_k$ and record the variation of the model loss and accuracy values in these three scenarios, the results are shown in Fig. 9. We can see that when $1-\alpha_k = 0.5$, the loss and accuracy curves of the model are smooth, but their convergence rate is slow. When $1-\alpha_k = 0.005$, the loss and accuracy values of the model reach the convergence point first, but their curves are not smooth. Therefore, in our previous experiments, we chose $1-\alpha_k = 0.05$ which means $\alpha_k = 0.95$.

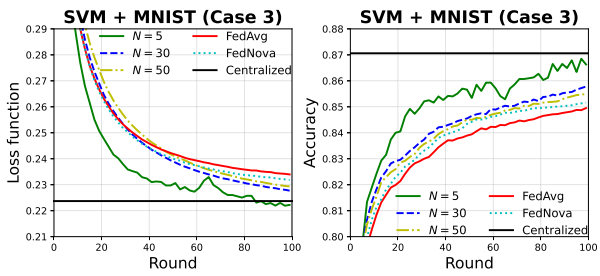

Fig. 10. Loss function and classification accuracy (on the SVM model and the MNIST dataset [41]) with the different number of clients.

In this setting, the loss and accuracy values of the model have both a fast convergence rate and a smooth curve.

6) Varying Number of Clients: To simulate the case of multiple nodes, we let each Raspberry Pi 4B in the prototype system run multiple programs of Algorithm 2 in parallel and set the number of programs running on each Raspberry Pi 4B to be equal from 1 to 10, thus simulating the number of nodes from 5 to 50. We record the loss and accuracy values in Case 3, where the number of clients is chosen to be 5, 30 and 50 for FedVeca algorithm, FedNova and FedAvg selected the number of clients as 50, and the results are shown in Fig. 10.

We can observe that the loss and accuracy values of the model do not converge faster as the number of clients increases, which is consistent with the phenomenon of diminishing returns which is described in [3]. Also, since the size of our total training dataset is fixed, when the number of nodes increases, the number of samples on each node decreases and the data becomes more dispersed, thus reducing the speed of model convergence. However, even at 50 nodes, the model of FedVeca algorithm converges faster than FedAvg and FedNova approach, proving that FedVeca algorithm is applicable at multiple nodes.

Considering the overall experiments, we can conclude that FedVeca algorithm provides an efficient performance for FL, and provides a novel innovation with theory for FL optimization.

VI. Conclusion and The Future Work

In order to improve the training efficiency of Federated Learning on the Non-IID dataset in communication networks, this paper proposed an method that is based on FedNova algorithm and controls the number of local SGD iterations on clients to obtain the optimal global training models in each communication round. We analyze the mathematical relationships between the number of local SGD iterations and the global objective in a round, and design an adaptive control algorithm from this relationship to predict the number of local SGD iterations on each client for the next round. Our experimental results confirmed the effectiveness of FedVeca method. In the future, we will explore how FedVeca method performs on the models with no-convex loss functions, and assign more effective weights for updating the global model based on the number and feature of samples per client. Furthermore, we intend to expand the application of the FedVeca algorithm to the rapidly growing field of Computer Vision (CV), specifically targeting more complex datasets such as CIFAR100 and ImageNet.

References

[1] R. Gu et al., "Towards efficient large-scale interprocedural program static analysis on distributed data-parallel computation," *IEEE Trans. Parallel Distrib. Syst.*, vol. 32, no. 4, pp. 867–883, Apr. 2021.

[2] J. Jiang, Z. Wen, Z. Wang, B. He, and J. Chen, "Parallel and distributed structured SVM training," *IEEE Trans. Parallel Distrib. Syst.*, vol. 33, no. 5, pp. 1084–1096, May 2022.

[3] B. McMahan, E. Moore, D. Ramage, S. Hampson, and B. A. y Arcas, "Communication-efficient learning of deep networks from decentralized data," in *Proc. Int. Conf. Artif. Intell. Statist.*, PMLR, 2017, pp. 1273–1282.

[4] J. Konečny, H. B. McMahan, F. X. Yu, P. Richtárik, A. T. Suresh, and D. Bacon, "Federated learning: Strategies for improving communication efficiency," 2016, *arXiv:1610.05492*.

[5] J. Liu et al., "From distributed machine learning to federated learning: A survey," *Knowl. Inf. Syst.*, vol. 64, pp. 885–917, 2022.

[6] J. Wang and G. Joshi, "Cooperative SGD: A unified framework for the design and analysis of local-update SGD algorithms," *J. Mach. Learn. Res.*, vol. 22, 2021, Art. no. 213.

[7] P. Dvurechensky, A. Gasnikov, and A. Kroshnin, "Computational optimal transport: Complexity by accelerated gradient descent is better than by sinkhorn's algorithm," in *Proc. Int. Conf. Mach. Learn.*, PMLR, 2018, pp. 1367–1376.

[8] Y. Lei, T. Hu, G. Li, and K. Tang, "Stochastic gradient descent for nonconvex learning without bounded gradient assumptions," *IEEE Trans. Neural Netw. Learn. Syst.*, vol. 31, no. 10, pp. 4394–4400, Oct. 2020.

[9] N. J. Harvey, C. Liaw, Y. Plan, and S. Randhawa, "Tight analyses for non-smooth stochastic gradient descent," in *Proc. Conf. Learn. Theory*, PMLR, 2019, pp. 1579–1613.

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

[11] L. Zhang, Y. Luo, Y. Bai, B. Du, and L.-Y. Duan, "Federated learning for non-iid data via unified feature learning and optimization objective alignment," in *Proc. IEEE/CVF Int. Conf. Comput. Vis.*, 2021, pp. 4420–4428.

[12] L. Gao, H. Fu, L. Li, Y. Chen, M. Xu, and C.-Z. Xu, "FedDC: Federated learning with non-iid data via local drift decoupling and correction," in *Proc. IEEE/CVF Conf. Comput. Vis. Pattern Recognit.*, 2022, pp. 10112–10121.

[13] F. Sattler, S. Wiedemann, K.-R. Müller, and W. Samek, "Robust and communication-efficient federated learning from non-iid data," *IEEE Trans. Neural Netw. Learn. Syst.*, vol. 31, no. 9, pp. 3400–3413, Sep. 2020.

[14] M. Luo, F. Chen, D. Hu, Y. Zhang, J. Liang, and J. Feng, "No fear of heterogeneity: Classifier calibration for federated learning with non-iid data," in *Proc. Adv. Neural Inf. Process. Syst.*, 2021, pp. 5972–5984.

[15] P. Zhang, H. Sun, J. Situ, C. Jiang, and D. Xie, "Federated transfer learning for IIoT devices with low computing power based on blockchain and edge computing," *IEEE Access*, vol. 9, pp. 98630–98638, 2021.

[16] H. Zhu, J. Xu, S. Liu, and Y. Jin, "Federated learning on non-iid data: A survey," *Neurocomputing*, vol. 465, pp. 371–390, 2021.

[17] S. Wang et al., "Adaptive federated learning in resource constrained edge computing systems," *IEEE J. Sel. Areas Commun.*, vol. 37, no. 6, pp. 1205–1221, Jun. 2019.

[18] J. Cheng, P. Luo, N. Xiong, and J. Wu, "AAFL: Asynchronous-adaptive federated learning in edge-based wireless communication systems for countering communicable infectious diseasess," *IEEE J. Sel. Areas Commun.*, vol. 40, no. 11, pp. 3172–3190, Nov. 2022.

[19] J. Wang, Q. Liu, H. Liang, G. Joshi, and H. V. Poor, "Tackling the objective inconsistency problem in heterogeneous federated optimization," in *Proc. Adv. Neural Inf. Process. Syst.*, 2020, pp. 7611–7623.

[20] T. Yoon, S. Shin, S. J. Hwang, and E. Yang, "FedMix: Approximation of mixup under mean augmented federated learning," in *Proc. Int. Conf. Learn. Representations*, 2020.

[21] H. Zhang, M. Cisse, Y. N. Dauphin, and D. Lopez-Paz, "mixup: Beyond empirical risk minimization," in *Proc. Int. Conf. Learn. Representations*, 2018.

[22] K. Zhou, Z. Liu, Y. Qiao, T. Xiang, and C. C. Loy, "Domain generalization: A survey," *IEEE Trans. Pattern Anal. Mach. Intell.*, vol. 45, no. 4, pp. 4396–4415, Apr. 2023.

[23] A. Back de Luca, G. Zhang, X. Chen, and Y. Yu, "Mitigating data heterogeneity in federated learning with data augmentation," 2022, *arXiv:2206.09979*.
[24] H. Wang, Z. Kaplan, D. Niu, and B. Li, "Optimizing federated learning on non-iid data with reinforcement learning," in *Proc. IEEE Conf. Comput. Commun.*, 2020, pp. 1698–1707.
[25] M. Yang, X. Wang, H. Zhu, H. Wang, and H. Qian, "Federated learning with class imbalance reduction," in *Proc. 29th Eur. Signal Process. Conf.*, 2021, pp. 2174–2178.
[26] Y. Zhao, M. Li, L. Lai, N. Suda, D. Civin, and V. Chandra, "Federated learning with non-iid data," 2018, *arXiv: 1806.00582*.
[27] T. Tuor, S. Wang, B. J. Ko, C. Liu, and K. K. Leung, "Overcoming noisy and irrelevant data in federated learning," in *Proc. 25th Int. Conf. Pattern Recognit.*, 2021, pp. 5020–5027.
[28] N. Yoshida, T. Nishio, M. Morikura, K. Yamamoto, and R. Yonetani, "Hybrid-FL: Cooperative learning mechanism using non-iid data in wireless networks," 2019, *arXiv: 1905.07210*.
[29] A. Fallah, A. Mokhtari, and A. Ozdaglar, "Personalized federated learning with theoretical guarantees: A model-agnostic meta-learning approach," in *Proc. Adv. Neural Inf. Process. Syst.*, 2020, pp. 3557–3568.
[30] C. Finn, P. Abbeel, and S. Levine, "Model-agnostic meta-learning for fast adaptation of deep networks," in *Proc. Int. Conf. Mach. Learn.*, PMLR, 2017, pp. 1126–1135.
[31] C. T. Dinh, N. Tran, and J. Nguyen, "Personalized federated learning with moreau envelopes," in *Proc. Adv. Neural Inf. Process. Syst.*, 2020, pp. 21394–21405.
[32] V. Smith, C.-K. Chiang, M. Sanjabi, and A. S. Talwalkar, "Federated multi-task learning," in *Proc. Adv. Neural Inf. Process. Syst.*, 2017, pp. 4427–4437.
[33] L. Corinzia, A. Beuret, and J. M. Buhmann, "Variational federated multi-task learning," 2019, *arXiv: 1906.06268*.
[34] F. Chen, M. Luo, Z. Dong, Z. Li, and X. He, "Federated meta-learning with fast convergence and efficient communication," 2018, *arXiv: 1802.07876*.
[35] Z. Zhu, J. Hong, and J. Zhou, "Data-free knowledge distillation for heterogeneous federated learning," in *Proc. Int. Conf. Mach. Learn.*, PMLR, 2021, pp. 12878–12889.
[36] T. Lin, L. Kong, S. U. Stich, and M. Jaggi, "Ensemble distillation for robust model fusion in federated learning," in *Proc. Adv. Neural Inf. Process. Syst.*, 2020, pp. 2351–2363.
[37] X. Peng, Z. Huang, Y. Zhu, and K. Saenko, "Federated adversarial domain adaptation," 2019, *arXiv: 1911.02054*.
[38] T. Li, A. K. Sahu, M. Zaheer, M. Sanjabi, A. Talwalkar, and V. Smith, "Federated optimization in heterogeneous networks," *Proc. Mach. Learn. Syst.*, vol. 2, pp. 429–450, 2020.
[39] X. Liang, S. Shen, J. Liu, Z. Pan, E. Chen, and Y. Cheng, "Variance reduced local SGD with lower communication complexity," 2019, *arXiv: 1912.12844*.
[40] S. P. Karimireddy, S. Kale, M. Mohri, S. Reddi, S. Stich, and A. T. Suresh, "SCAFFOLD: Stochastic controlled averaging for federated learning," in *Proc. Int. Conf. Mach. Learn.*, PMLR, 2020, pp. 5132–5143.
[41] Y. LeCun, L. Bottou, Y. Bengio, and P. Haffner, "Gradient-based learning applied to document recognition," *Proc. IEEE*, vol. 86, no. 11, pp. 2278–2324, Nov. 1998.
[42] A. Krizhevsky et al., "Learning multiple layers of features from tiny images," 2009.
[43] S. Bubeck et al., "Convex optimization: Algorithms and complexity," *Found. Trends Mach. Learn.*, vol. 8, no. 3/4, pp. 231–357, 2015.

Ping Luo (Student Member, IEEE) received the MAEng degree in computer science and technology from Hainan University, Haikou, China. He is currently working toward the PhD degree in computer science and technology from the National University of Defense Technology (NUDT), Changsha, China. His research interests include convex optimization, the industrial Internet of Things and artificial intelligence.

Jieren Cheng (Member, IEEE) received the PhD degree in computer science and technology from the National University of Defense Technology (NUDT), in 2010. He is now a professor and the associate dean of the School of Computer Science & Technology, Hainan University, China. He is awarded as "Famous South China Sea Scholar". He serves as the director of the Hainan Provincial Blockchain Technology Engineering Research Center. His research interests include blockchain, Big Data, cloud computing, cyberecurity, artificial intelligence and intelligent transportation.

N. Xiong (Senior Member, IEEE) received the PhD degree from the School of Information Science, Japan Advanced Institute of Science and Technology (JAIST), in 2008. He is current a distinguished professor with National Engineering Research Center for E-Learning, Central China Normal University (CCNU), Wuhan, Hu Bei Province, 430079, China. He is also with the Department of Computer Science, Georgia State University, Atlanta, Georgia. His research interests include deep learning, reliable networks, software engineering, and Big Data analytics. He works in CCNU for many years, and obtained many research funding and many industrial projects. He published more than 600 journal paper with more than 200 IEEE journal papers. He also creates a company about design and analysis for complex reliable software systems, and obtains more than 10 patents.

Zhenhao Liu (Student Member, IEEE) received the MAEng degree in electronic information from Hainan University, Haikou, China. He is currently working toward the PhD degree in agricultural information engineering with Huazhong Agricultural University, Wuhan, China. His research interests include federated learning, privacy-preserving, and artificial intelligence.

Jie Wu (Fellow, IEEE) is the director of the Center for Networked Computing and Laura H. Carnell professor with Temple University. He also serves as the director of International Affairs with the College of Science and Technology. He served as chair of the Department of Computer and Information Sciences from the summer of 2009 to the summer of 2016 and associate vice provost for International Affairs from the fall of 2015 to the summer of 2017. Prior to joining Temple University, he was a program director with the National Science Foundation and was a distinguished professor with Florida Atlantic University. His current research interests include mobile computing and wireless networks, routing protocols, network trust and security, distributed algorithms, applied machine learning, and cloud computing. He regularly publishes in scholarly journals, conference proceedings, and books. He serves on several editorial boards, including *IEEE Transactions on Mobile Computing*, *IEEE Transactions on Service Computing*, *Journal of Parallel and Distributed Computing*, and *Journal of Computer Science and Technology*. He is/was general chair/co-chair for IEEE IPDPS'08, IEEE DCOSS'09, IEEE ICDCS'13, ACM MobiHoc'14, ICPP'16, IEEE CNS'16, WiOpt'21, and ICDCN'22 as well as program chair/cochair for IEEE MASS'04, IEEE INFOCOM'11, CCF CNCC'13, and ICCCN'20. He was an IEEE Computer Society Distinguished Visitor, ACM Distinguished Speaker, and chair for the IEEE Technical Committee on Distributed Processing (TCDP). He is a fellow of the AAAS. He is the recipient of the 2011 China Computer Federation (CCF) Overseas Outstanding Achievement Award.