Reschedule Gradients: Temporal Non-IID Resilient Federated Learning

Xianyao You, Ximeng Liu, *Senior Member, IEEE*, Nan Jiang, Jianping Cai, and Zuobin Ying, *Member, IEEE*

***Abstract*—Federated learning is a popular framework designed to perform the distributed machine learning while protecting client privacy. However, the heterogeneous data distribution in real-world environments makes it difficult to converge when performing model training. In this article, we propose the federated gradient scheduling (FedGS), an improved historical gradient sampling utilization method for optimizers that utilize historical gradients in the federated learning to alleviate the instability problem of historical gradient information due to non-IID. FedGS consists of two main steps to improve federated learning performance. First, clustering uses clients' label distributions, which relabel clients and their submitting gradients. Second, sampling gradient clusters to generate an IID gradient set, which feeds to optimizers to derive valid momentum information. Besides, we introduce differential privacy to collaborate with FedGS to enhance clients' privacy protection strength. Compared to previous non-IID federated learning solutions, our method can achieve greater resistance to temporal non-IID. Moreover, experiments show that FedGS can achieve faster convergence and performance gain of up to 10% over existing state-to-art methods in some scenarios. FedGS can combine with existing methods easily to achieve better performance. We further verify that our method has robust performance gains in different non-IID scenarios, demonstrating the adaptability of FedGS for different scenarios.**

***Index Terms*—Data privacy, deep learning, differential privacy, federated learning, non-IID, optimizer.**

I. Introduction

FEDERATED learning is a distributed machine learning framework originally designed by Google [1] to address the need for model training in real-world settings with limited data on a single subject and, most importantly, to address the privacy and security issues that arise when using participants' data in centralized learning. To achieve the goal of protecting clients' privacy, the federated learning framework is designed to restrict the training process only to the local part of the participant and share only the final training results with the server, ensuring that the data is not known to entities outside the participant.

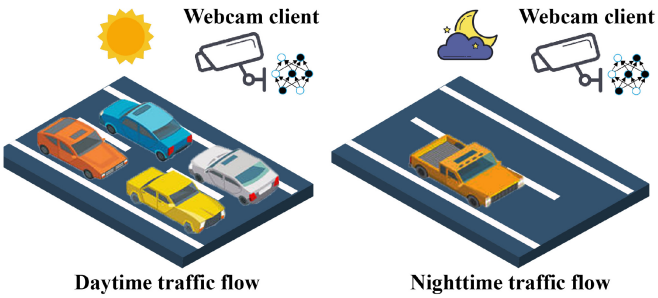


Fig. 1. Temporal non-IID examples.

To prevent possible malicious servers from obtaining private information about clients from training samples, federated learning cannot preprocess the data at the global level when training the model, but must face the heterogeneity of data between different clients [2]. The existence of this heterogeneity makes the application of traditional model optimization algorithms suffer from different degrees of convergence difficulties. For the convergence problem, [3] provides a concrete theoretical analysis of the convergence correctness of the traditional federated learning algorithm federated averaging (FedAvg) under data heterogeneity and [4] provides a specific experimental analysis of this practical problem under federated learning.

We notice that in response to the problem of data heterogeneity under federated learning, although many existing works have also proposed their respective solutions, few works are specified to the temporal non-IID problem under federated learning. For the time-series data heterogeneity scenario in federated learning, the training data available to the model at different periods are heterogeneous. The following figure shows examples of temporal data heterogeneity. In Fig. 1, the number of cars in the training data collected by the camera on different periods shows different distributions.

In the specific federated learning scenario, the data held by the participants at a certain time or even for a long period may only be a partial part of the potential global data distribution, e.g., reflecting data under a limited number of labels. In this case, as the federated learning model

Manuscript received 6 March 2022; revised 8 June 2022 and 28 July 2022; accepted 29 August 2022. Date of publication 31 August 2022; date of current version 22 December 2022. This work was supported in part by the National Natural Science Foundation of China under Grant 62072109 and Grant U1804263, and in part by the Natural Science Foundation of Fujian Province under Grant 2021J06013. *(Corresponding author: Ximeng Liu.)*

Xianyao You, Ximeng Liu, and Jianping Cai are with the College of Computer and Data Science and the Fujian Provincial Key Laboratory of Information Security of Network Systems, Fuzhou University, Fuzhou 350108, China (e-mail: snbnix@gmail.com).

Nan Jiang is with the Department of Internet of Things, East China Jiaotong University, Nanchang 330013, China (e-mail: jiangnan@ecjtu.edu.cn).

Zuobin Ying is with the Faculty of Data Science, City University of Macau, Macau, China (e-mail: zbying@cityu.mo).

Digital Object Identifier 10.1109/JIOT.2022.3203233

has to be trained on these biased data for a long time, the distribution fitted by the model gradually increases the fitting error about the global data distribution and overfits a local distribution of the global data. The presence of this scenario makes the model easily misled by the presently exposed data and difficult to converge. What's worse, when using optimizers based on historical gradient information, such as AMSGrad [5], AdamW [6], AdaBelief [7], etc., their historical gradients also tend to fail to yield valid guidance information, which means we cannot make use of start-to-art optimization methods to achieve better model performance in these scenarios.

Analyzing existing solutions specified to data heterogeneity problem, most of them have some kind of preference assumption, for example, with personalized learning preference, [8], [9], [10] and other methods try to use the data heterogeneity of federated learning to generate client-specific local models, but these methods do not provide methodology design to improve the overall performance of the federated model. There are also some general improvement methods, such as [11], [12], which use regularization to provide improvement. Despite the generality of these methods' design, the generality also means they cannot achieve generally excellent performance under all data heterogeneity scenarios. In addition, we note that in the existing methods, probably more scenarios are about the data heterogeneity between clients in the same round, while the scenarios of data heterogeneity between clients on the temporal order are not discussed in depth. For the data heterogeneity exhibited within and between rounds, the performance gain from the existing solutions may be discounted.

To overcome this challenge, we propose federated gradient scheduling (FedGS) federated learning algorithm for improving the optimization method on the server under federated learning. The algorithm is optimized for the gradient utilization process of the optimizer based on historical gradient information under federated learning, which makes it better adaptable for non-IID scenarios. For the construction of this algorithm, we are inspired by the above-mentioned realistic scenario. During the training process of federated learning, the client's label distribution is clustered by the existing K-means algorithm to relabel the gradients submitted by the historical client according to clusters. In the subsequent utilization of historical gradients by optimizers, the federated learning server tries to sample equal amounts of gradients from each cluster based on the historical gradients classified after clustering. In the case, that the federated learning task is correlated with the labels of its data, we expect the sampled historical gradients to be IID. Moreover, FedGS is not designed for a specific optimization process, which makes it possible to be utilized on all optimization methods based on utilization of historical gradients. In addition, due to the privacy protection limitations of federated learning, we combine the differential privacy [13] mechanism in the algorithm to protect information from the clients to obtain the label distribution of each client in the algorithm.

The main contributions of this article can be summarized as follows.

1) *Federated Gradient Clustering:* We design FedGS, an improved algorithm for federated learning in temporal non-IID scenarios, based on historical gradient clustering and sampling mechanism, which has better convergence performance under temporal non-IID scenarios.
2) *Gradient-Scheduling:* In response to features of temporal non-IID, we propose a gradient aggregation method based on existing optimization algorithms for history gradient utilization. Specifically, performed by reclustering and sample of history gradients, our gradient-scheduling aggregation process can improve the information extraction ability of optimizer for historical gradients under temporal non-IID scenarios.
3) *Extensible Framework:* FedGS provides a scalable framework that can be extended to any model optimization method with historical gradient utilization, including possible future optimization methods, correcting their performance shortcomings under temporal non-IID scenarios.

The remainder of this article is organized as follows. In Section II, we discuss existing work in related areas. Section III gives some background knowledge related to our topic. Section IV lists the problems and goals we are targeting. Section V presents the implements detail of FedGS. And some security analyses of our method are given in Section VI. Experiments supporting our method are evaluated and analyzed in Section VII. The last section concludes this article.

II. Related Work

Federated learning was first seen in a paper published by [1] as FedAvg schema. In federated learning, there exists a vertical federated learning architecture that is different from the traditional horizontal federated learning (e.g., FedAvg). In this vertical architecture, different participants have different feature spaces, and there may be a client with data labels acting as a server-like architecture. Existing related works try to improve different aspects of federated learning, such as communication efficiency [14], privacy security [15], and heterogeneity resistance [16] to enhance the usability and effectiveness based on the basic federated learning framework.

Aiming at the convergence problem of the training process caused by the heterogeneity of data under federated learning, the existing work proposes different mitigation methods from different angles [17], [18], [19]. We can compare our methods with some existing solutions.

1) *FedAvg:* The initial and most basic federated learning method. Clients transfer gradients to the server directly, and the server simply aggregates gradients from each client by averaging. Leaking resilient to non-IID.
2) *FedProx:* Adding regularization term to client-side loss function when applying gradient descent (GD) methods liking stochastic GD (SGD), to suppress model feature space drift. Limited performance gain in many scenarios especially in temporal non-IID.
3) *FedAws [16]:* Specifically, designed for local data sets only with one-class scenarios. Using regularization to

maximum the distance across embedding vectors for each client. Assuming clients' data non-IID exists inside each round but without considering the non-IID situation across rounds, which limits its performance under temporal non-IID scenarios.

In contrast to the above methods, the main difference in the problem background of our proposed FedGS is that FedGS highlights the problem of solving convergence in a temporal non-IID scenario. In contrast, existing methods may be designed to mainly consider the non-IID problem for data between clients participating in training at the same point in time.

In addition to these approaches, there exist some other approaches that are specific to different scenarios. For example, [20] focuses on security analysis in distributed learning under unsupervised tasks. Tong *et al.* [21], on the other hand, introduced the idea of personalized learning to generate client-specific models by using knowledge transfer. Different from these works, FedGS mainly considers how to enhance the convergence of global models under non-IID federated learning with labeled data.

The optimization algorithms commonly used in federated learning, besides, the first proposed GD, and SGD algorithms, existing model optimization algorithms based on them utilize, in one form or another, the gradients computed in the historical training process. These existing start-to-art optimization methods include AdamW [6], AMSGrad [5], and AdaBelief [7]. In many of these algorithms, the step lengths in some dimensions of the currently computed gradients are improved by calculating the primary or quadratic averages of the historical gradients. Many existing machine learning scenarios, such as CIFAR10 [22], are still used traditional optimization methods to achieve better model performance [23]. Tong *et al.* [24] tried to improve the parameters of the optimization method to have better performance and stability under non-IID. Similarly, in this article, we will improve the optimization method in terms of historical gradient utilization.

To ensure the performance of the method in this article, we incorporate differential privacy mechanisms in the method design to be able to enhance the privacy-preserving properties. Differential privacy is a method of secure computing whose main goal is to make it difficult for an attacker to infer information about a specific individual from query results even with background knowledge by noise-enhancing the overall distribution information while protecting the privacy of individual data. In machine learning scenarios, differential privacy mechanisms are designed to protect the trained models from attackers performing reconstruction attacks [25] or membership inference attacks [26], and other malicious methods resulting in individual privacy disclosure [27]. Geyer *et al.* [28] further details the scenario of differential privacy under federated learning, proposes a method about how to protect the fact that participation in federated learning by differential privacy at the client level, and proposed a theoretical approach to calculate privacy loss when applying differential privacy mechanisms in the federated learning process based on relevant theories from existing work. Wei *et al.* [29] applied differential privacy to client data samples to obtain a broader privacy-preserving coverage. In this article, we build on the theory of these works for the construction of differential privacy-related processes.

TABLE I
NOTATION DESCRIPTIONS

Notations	Descriptions
ϵ	Privacy budget for differential privacy
δ	Relaxation term for differential privacy
$\hat{M}$	Gradient clipping threshold
D	Client-side data sets
θ	Model parameters
∇	Model parameter gradient
Δ	Model parameter updates
m	Gradient first-order momentum
v	Gradient second order momentum
η	Learning rate
ρ	Historical gradient exponential decay weight
C	Client collection
L	Dataset label vector
G	Historical gradient collection
T	Federated learning iteration rounds
N	Client number
$\mathcal{L}$	Laplace distribution
$\mathcal{D}$	Dirichlet distribution

III. PRELIMINARIES

In this section, we provide a basic introduction to the principles involved, including the federated learning architecture and differential privacy mechanisms. Some of the notations that we will use are explained in Table I.

A. Federated Learning

The first proposed algorithm in federated learning is FedAvg [1]. The training process defined by the FedAvg algorithm contains several entities, including a central server, a set of clients C. In each training round the central server selects a subset of the client set $C_t \in C$ and distributes a federated learning model to each of them. After receiving the central model, the client performs iterative training based on the local data set D_i for several epochs locally and later passes its model updates back to the server, which performs the specific parameter aggregation process. The selection of federated learning training rounds T is usually dynamic, and the selection criterion can be whether the global model reaches a certain threshold performance or the privacy budget is exhausted in the differential privacy mechanism, etc.

B. Temporal Non-IID Federated Learning

In temporal non-IID scenarios, there are important features that clients participating in one epoch or adjoining epoch have the same local data distribution bias, or more specifically, labels distribution bias. Formally, for clients set C in federated learning, we assume there are two clients c_1 and c_2 who participate in federated learning at two close moments. For a client's local data D, we can describe its data labels as set $L = \{\text{label}(d), d \in D\}$. For temporal non-IID scenario under federated learning, we can assume L_1 and L_2 have considerable degree of overlap in membership, where label set L_1 come from c_1 and L_2 come from c_2. If the above assumption holds for any two clients at two close time points throughout

the training process of federated learning, we say the federated learning is temporal non-IID.

C. Optimization Methods for Historical Gradient Utilization

Many different existing optimization methods make use of historical gradients in some form when computing the gradient of the current model, so we call them historical gradient-based. For different optimization methods, the historical gradients are utilized in different ways to find scenario-specific performance improvements under different inductive preferences. For different optimization methods, we give their specific update equations. Where η is the learning rate, $\nabla^{(t)}$ and $\Delta^{(t)}$ are the gradient and update of the model at moment t, respectively.

1) For the Momentum [30] optimization method. In the following equation, the current update is generated by accumulating the historical gradient by an exponential decay, where ρ^n represents the exponential decay weight of the historical gradient at time distance n

$$\Delta^{(t)} = -\sum_{n=1}^{t} \rho^n \left(-\eta \nabla^{(t-n)}\right). \tag{1}$$

2) For the AdamW optimization method. $w^{(t)}$ represents the parameter decay rate and $\theta^{(t)}$ represents the parameter value at t moments. A first-order momentum mechanism is used similar to the Momentum method, but the second-order momentum information based on historical gradients is used to set the learning rate adaptively. By using weight decay to achieve better convergence with respect to Adam. β_1 and β_2 control the exponential decay weights of the momentum $m^{(t)}$ and the adaptive gradient $v^{(t)}$, respectively. ϵ is a small number whose role is to prevent the problem of dividing by zero in the adaptive learning rate mechanism

$$\begin{aligned} m^{(t)} &= \sum_{n=1}^{t} \beta_1^n \left((1-\beta_1)\nabla^{(t-n)}\right) \\ v^{(t)} &= \sum_{n=1}^{t} \beta_2^n \left((1-\beta_2)\left(\nabla^{(t-n)}\right)^2\right) \\ \Delta^{(t)} &= \eta \left(\frac{m^{(t)}/\left(1-\beta_1^t\right)}{\sqrt{v^{(t)}/\left(1-\beta_2^t\right)}+\epsilon} + w^{(t)}\theta^{(t-1)}\right). \end{aligned} \tag{2}$$

3) For the AMSGrad optimization method. Similar to Adam, but using continuous second-order gradient momentum accumulation to solve the learning rate oscillation problem in Adam. The meaning of the parameters is similar to that of AdamW

$$\begin{aligned} m^{(t)} &= \sum_{n=1}^{t} \beta_1^n \left((1-\beta_1)\nabla^{(t-n)}\right) \\ v^{(t)} &= \sum_{n=1}^{t} \beta_2^n \left((1-\beta_2)\left(\nabla^{(t-n)}\right)^2\right) \\ \Delta^{(t)} &= \frac{\eta \cdot m^{(t)}}{\sqrt{\max\left\{v^{(i)}, i \in \{1 \cdots t\}\right\}}+\epsilon}. \end{aligned} \tag{3}$$

4) For the AdaBelief optimization method. The same adaptive gradient method as AdaGrad and AdamW. By using the variance information of historical gradients to achieve generalizability, convergence, and stability at the same time. $s^{(t)}$ has a similar role to $v^{(t)}$ in AdamW, and in this article, we are not concerned with the differences in effects between them, but only with the computational form of the cumulative process

$$\begin{aligned} m^{(t)} &= \sum_{n=1}^{t} \beta_1^n \left((1-\beta_1)\nabla^{(t-n)}\right) \\ s^{(t)} &= \sum_{n=1}^{t} \beta_2^n \left((1-\beta_2)\left(\nabla^{(t-n)} - m^{(t-n)}\right)^2\right) \\ \Delta^{(t)} &= \frac{\eta \cdot m^{(t)}}{\sqrt{s^{(t)}}+\epsilon}. \end{aligned} \tag{4}$$

For specific scenarios, better convergence may be achieved by changing the parameters of the optimizer update formula. In the FedGS approach of this article, an effort is made to optimize for the temporal non-IID scenario by changing its default moment-by-moment accumulation process for the historical gradients.

D. Differential Privacy

Differential privacy usually assumes the existence of a query function $f(x) : x \to R$, x from a data distribution D. For two data distributions D, D' that differ only to the presence of a specific data item, then D, D' is called an adjacent data set. For adjacent data sets, (5) defines sensitivity between them in differential privacy

$$\Delta f = \max_{D,D'} \|f(D) - f\left(D'\right)\|_p \tag{5}$$

where the norm corresponding to p is determined by the specific differential privacy mechanism, e.g., L_1 norm is used in the case of the Laplace mechanism, and L_2 norm is used for the Gaussian mechanism. Regardless of the specific implementation, for the case, where the (6) is satisfied based on the definition of sensitivity, we can say that the mechanism satisfies ϵ differential privacy

$$P[M(D) \in S] \le e^{\epsilon} P\left[M\left(D'\right) \in S\right]. \tag{6}$$

For the Laplacian mechanism of differential privacy, its privacy protection is accomplished by adding a specific amount of Laplacian noise to the output, which is controlled by the parameters μ and σ of the Laplacian distribution $\mathcal{L}$. Given a query function $f(D)$ on specific data distribution, adding Laplace noise satisfying the distribution $\mathcal{L}(0, \Delta f/\epsilon)$ to the output can make it satisfy ϵ differential privacy. For the Gaussian mechanism elaborated by [28], adding Gaussian noise satisfying the distribution $\mathcal{N}(0, \sigma\hat{M})$ to the output, where $\hat{M} = \sum \text{norm}(f(d)), d \in D$, makes it satisfy the ϵ differential privacy, but with δ probability the protection fails, where $\delta \le (4/5)\exp(-(\sigma\epsilon)^2/2)$.

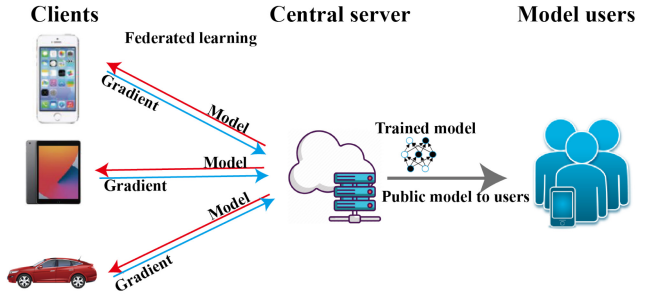


Fig. 2. Our system model.

IV. Problem Formulation

A. System Model

There are two main types of entities included in our scenario: 1) the central server and 2) the clients. The system model is illustrated in Fig. 2. Regarding the role of these two types of entities in the system model, we give a description.

1) *Central Server:* The central server is responsible for task definition, task distribution, and control regarding the task execution phases, such as the start and end of the training, during the federated learning process. In addition, the central server is responsible for aggregating the training data from the client during the training process and interacting with the client for the aggregated model.
2) *Clients:* Each client individually has a local data set and trains the local model in federated learning and exchanges the trained model with the server. Usually, the data distribution of each client is non-IID.
3) *Model Users:* The model user can access the model at the end of the federated learning training, but is not involved in the training itself and cannot see the various data during the training process.

In a typical horizontal federated learning scenario, clients join as participants in a distributed training process coordinated by a central server. Due to the privacy-preserving design of federated learning, the server is not able to make assumptions about the data distribution of the participants or to preprocess the data. In this case, passive acceptance and aggregation of gradient data from clients often do not lead to effective convergence. In the temporal non-IID scenario, applying optimization methods based on historical gradient information under federated learning does not show the same effective convergence improvement as under centralized training because the historical gradients come from heterogeneous data.

Formally describe the training process of federated learning, initially with a set C of training participants consisting of N clients, each holding a local data set $D_i, i \in [N]$ ($[N] = \{1 \cdots N\}$). Federated learning has $T \in \mathbb{N}$ rounds of training processes, each with a sampling rate of q for the clients. In the training round $t \in \{1 \cdots T\}$, the sampled subset of clients C^t undergoes a training process based on E epochs of local data D_i on the central model θ^t from the server and submits the trained model θ_i^t to the server, which aggregates it to produce a consistent update as the central model θ^{t+1} for the next round. In this process, the server can see in round t the gradients $G^t = \{g_i^t : i \in [N]\}$ submitted from the clients in the current round. For an optimizer based on historical gradient information, it uses the set of historical gradients $G = \{\cup G^t : t \in [N]\}$ to generate the currently available information and computes it with the current post-aggregated gradient ∇^t to generate the actual update. Generally, the process can be expressed as

$$m_\tau = \sum_{i=1}^{t} \sum_{j=1}^{N} w_i \cdot \tau\left(g_j^i\right)$$
$$\theta^{(t+1)} - \theta^{(t)} = -\eta \nabla^{t+1} = -\eta \cdot \upsilon\left(m_{\tau_1}, \ldots, m_{\tau_n}\right) \quad (7)$$

where τ denotes the transformation method for the historical gradient of a specific optimization method, and υ denotes how the optimization method uses the cumulative results of each historical gradient information.

B. Threat Model

We assume that the parties in the model are semi-honest. A semi-honest participant will run the agreed protocol honestly but will try to obtain as much additional knowledge as possible from the data they received. In our model, there are three semi-honest parties.

1) *Central Server:* A semi-honest central server can obtain the gradients sent from each client and try to recover client privacy information based on the global model and the set of these gradients.
2) *Clients:* A semi-honest client can access the information distributed from the server, such as the global model. But compared to the server, a client cannot see training process data, such as gradients, from other clients under the assumption that each client is semi-honest and not complicit.
3) *Model Users:* A model user usually exists after the entire training process is over. Compared to the client and server roles, they cannot access information during the training process and thus have relatively limited attack capabilities. However, they can still try to infer the training process or the client's private information from the final published model.

In our model, the attacks that occur on the server tend to be stronger because the client, as well as the model users, always get indirect data from other clients, such as the trained model, via the server. Therefore, FedGS mainly considers attacks from the server in terms of security.

C. Design Goals

In the FedGS approach of this article, we will improve the optimization method for the scenario of temporal heterogeneity by targeting the process of how the optimizer utilizes the set of historical gradients G. For the improvement of the non-IID federated learning method, we try to achieve the following two goals.

1) *Model Performance:* FedGS method can bring higher final model performance and faster convergence compared to existing methods. In addition, a certain degree

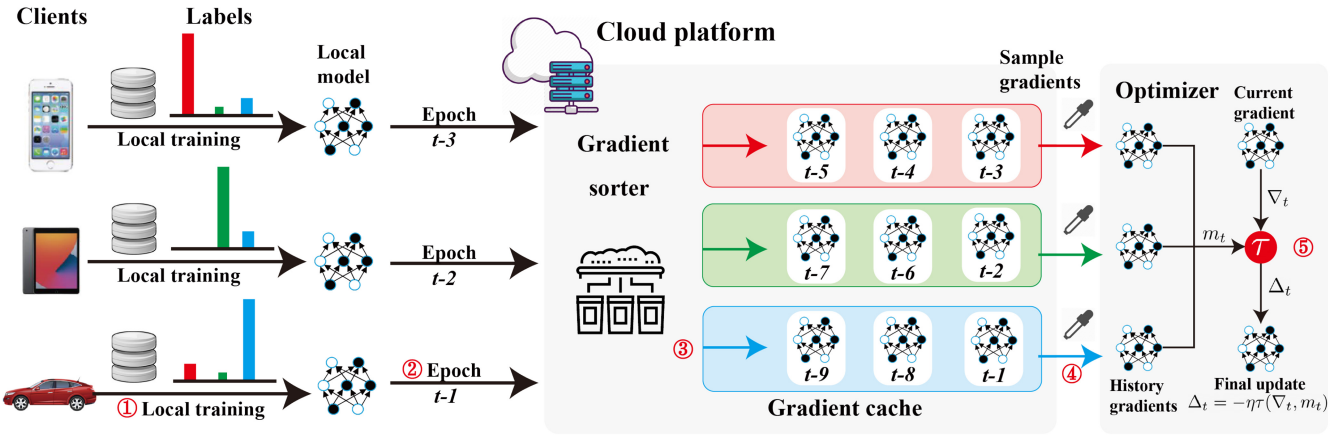


Fig. 3. Overview of FedGS. Take epoch $t-1$ moment as an example: (1) the client trains the model based on local data; (2) the client uploads the gradient to the server; (3) the server sorts the gradient based on the label distribution information of the client; (4) sampling the historical gradients regarding the label distribution IID; and (5) applying the cumulative information of the historical gradient to the current gradient.

of robustness of the method for different temporal non-IID scenarios needs to be ensured.

2) *Security:* In Federated Learning, FedGS needs to be able to guarantee the security of the training information exchanged during the training process so that this information cannot be accessed by malicious third parties, either directly or through an attack process.
3) *Easy Application:* The modifications to the existing federated learning framework by FedGS are as simple as possible to the client-side and still able to achieve performance and security design goals. Further, FedGS should be able to be applied simultaneously with existing federated learning improvement methods.

V. Methodology

In this section, we describe the architecture and basic process of FedGS, and illustrate the principles and feasibility of the design. Due to the complexity of realistic scenarios for federated learning, we construct the scenarios on existing data sets on which the experiments are based, and describe the methods and parameter design in the construction process.

A. Proposed Method

The general framework of FedGS is shown in Fig. 3. In this schematic there are three parts: 1) the clients; 2) the gradient sorter; and 3) the optimizer. The client trains the model on local data and submits the gradient after training. The gradient sorter sorts the gradients of the clients to the clusters corresponding to the clients according to the preclustered client divisions. The optimizer optimizes the global model based on sampling gradients from each cluster and the current gradient.

Our algorithm consists of two main parts: the first part preclusters the clients in federated learning according to their label distribution and the second part samples historical gradients with preclustered clusters as the input to the historical gradient-based optimizer. A description of the algorithm workflow is given in Fig. 4.

1) Client Preclustering: We assume that the data set is about a single-label classification task, so the label of each data

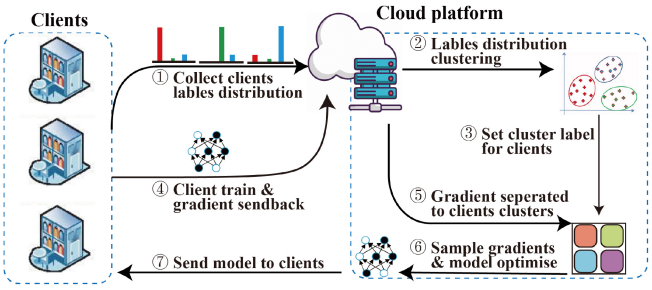


Fig. 4. Workflow of FedGS.

item can be represented as a one-hot vector. For the client's label distribution we take the following definition.

Definition 1: For the entire client, we define its label distribution vector L_i as the proportion of the total number of samples for each label, and for each data item $d \in D_i$ of a specific client, its corresponding label is label(d), then, the label distribution about this client can be expressed as

$$L_i = \left[L_i^{(y)}\right] = \left[\frac{\sum_{d \in D_i} \text{label}(d) = y}{|D_i|}\right] \tag{8}$$

where the superscript (y) denotes the sample proportion corresponding to label category y. The square brackets represent the vector composed of the sample proportions of all label categories.

Before the federated learning process, the server tries to obtain the label distribution of each client to use this information as the client's identity for the subsequent clustering process. Due to the privacy-preserving design of federated learning, the client's label distribution cannot be obtained directly from clients because it may contain private information. For this reason, we add differential privacy protection to the label distribution before clients submit it to the server. For a specific data set D, assume that its set of label vectors is L. Then, for a specific client c, we normalize the maximum value of its label distribution vector, i.e., $L_i^{(y)} \in [0, 1]$ and $\sum_y L_i^{(y)} = 1$, then, it is known that the sensitivity of the generating function $f(D_i)$ with respect to label

Algorithm 1 Label Distribution Generating

Input: Client C_i and client local data D_i; Privacy budget ϵ
Output: With-noise label distribution $\widetilde{L}_i$

procedure $generate\_\widetilde{L}(C_i, \epsilon)$
  $L_i^{(y)} = \frac{\sum_{d \in D_i} label(d)=y}{|D_i|}$ ▷ generate local label distribution L_i
  $Y \sim \mathcal{L}(0, \frac{1}{\epsilon})$ ▷ generate dp-noise
  $\widetilde{L}_i = L_i + Y$ ▷ add noise
  send $\widetilde{L}_i$ to server
end procedure

distribution is 1 for that client. Taking the Laplace mechanism as an example, given a specific privacy budget ϵ, we add Laplace noise satisfying $\mathcal{L}(0, 1/\epsilon)$ to it such that the client's label distribution satisfies ϵ differential privacy protection. The client's label distribution generating algorithm pseudocode is described in Algorithm 1.

After the server receives the noisy label distribution $\widetilde{L}$ from clients, we try to set a cluster S_i to which each label belongs, so that the cluster settings of all clients satisfy the optimization condition of (9) concerning their label distribution

$$\arg\min_{S} \sum_{i=1}^{k} \left(\sum_{u,v \in S_i} KL(\widetilde{L}_u, \widetilde{L}_v) + \sum_{u \in S_i, v \notin S_i} -KL(\widetilde{L}_u, \widetilde{L}_v) \right) \quad (9)$$

where KL represents the Kullback–Leibler scatter. The equation in parentheses consists of two parts. The first part calculates the distance of the label distribution within the same cluster. And the second part calculates the distance of the label distribution within different clusters. By satisfying this optimization equation, we can mark clients with similar local distributions as identical clusters to sample the IID gradient set from these clusters. We can implement this optimization process with the standard k-means clustering algorithm. For a given number of clusters K, the clustering result about the label distribution makes the clients to be partitioned into K clusters.

The significance of clustering clusters is strongly related to the global bias of the local data distribution corresponding to clients. The process is based on the assumption that clients with similar label biases tend to correspond to similar parts of the global data distribution among their local data. This allows us to obtain an approximate representation of the global data distribution by sampling clients from different clusters of label distributions and combining them.

2) Gradient Sampling: During each round of aggregation on the server, the optimizer will give the final model update result by combining historical gradient information with the current aggregated gradient. Instead of the default cumulative process of averaging the historical gradients in optimization algorithms, such as AdaBelief, AMSGrad, and AdamW, we try to sample an unbiased set of historical gradients about global data in each update round. Our strategy is that we try to sample the gradient closest to the current time point from each cluster from K subsets of clients divided by preclustering. Due to the similarity of the distribution of clients in one cluster, the gradients submitted by different clients within a cluster should be considered to have similar utilities. Also, considering that the utility of a gradient decreases as the distance from the current time point increases, we sample a finite number of gradients within a cluster that are closest to the current time when sampling historical gradients.

Algorithm 2 FedGS

Input: Clients set C; Total rounds T; Number of cluster K; Initial model θ^0
Output: Trained global model θ_g^T

$\theta_g^0 = \theta_0$ ▷ initialize the global model
$\widetilde{L} = \{generate\_\widetilde{L}(c), c \in C\}$
$S = kmeans(\widetilde{L}, K)$ ▷ client clusting
$G = \emptyset$ ▷ clustering set for gradients
for each round $t = 1 \cdots T$ **do**
  $C_t = sample\_client(C)$ ▷ server sample
  for each client $c \in C^t$ **do**
    $\theta_c^t = client\_train(\theta_g^t, D_i)$
  end for
  $\widetilde{\theta}^t = \sum \frac{1}{|C_t|} \theta_c^t$
  $\Delta\theta^t = \widetilde{\theta}^t - \theta_g^{t-1}$ ▷ vitual gradient
  $G_{S(c)} = G_{S(c)} \cup \theta_c^t, \forall c \in C^t$ ▷ cache gradient
  $\widetilde{G} = \cup \widetilde{G}_i, \widetilde{G}_i \in G_i$ ▷ sample gradient
  $\Delta^t = -\eta \cdot \upsilon\left(\sum_{g \in \widetilde{G}} w_i \cdot \tau(g)\right)$
  $\theta_g^t = \theta_g^{t-1} + \Delta^t$
end for

For the different optimizers, we manually adjust their historical gradient utilization process. This process is equivalent to trying to adjust the historical gradient utilization weights w in (7). In our method, we try to sample an IID set of historical gradients $\widetilde{G}$, and for $g \in \widetilde{G}$, we set the weights corresponding to g according to the weight generation method in the specific optimization method, while for the gradients that are not in the set, we set their weights to zero. Taking momentum optimizer as an example, suppose we sample a historical gradient set $\widetilde{G}$, and for its first-order momentum estimate about the gradient, we add the momentum from the sampled gradient set in the process of calculating the momentum in time order, as shown in (10), where $g_{(i)}$ represents the gradients in the sampled $\widetilde{G}$ in the temporal order. Here, ρ^i corresponds to the w_i item in (7)

$$m_t = \sum_{i=1}^{|\widetilde{G}|} \rho^i \cdot g_{(i)}, \ \ g_{(i)} \in \widetilde{G}. \quad (10)$$

For the whole process, we give a representation of its pseudocode form. It can be noted that relative to FedAvg, its lack of utilization of client's gradients on the current round is because with the sampling of gradients in the time-nearest order, it can be expected that the gradients in the current round will be sampled and utilized. The overall algorithm pseudocode is described in Algorithm 2.

3) Inert Samping: We note that in the method designed in this article, the optimizer needs to reconstruct the information about the historical gradient in each round. For existing optimizer methods, their historical gradient information is usually

cumulative on a round-by-round basis, which means that they need to handle information updates only once in each round. Analyzing from optimization process, take momentum optimization algorithm as example and we can describe its algorithm as the following form:

$$m_t = \sum_{n=1}^{t-1} \rho^n \left(\eta \nabla^{(t-n)}\right)$$
$$\Delta^{(t)} = -\left(m_t + \left(\eta \nabla^{(t)}\right)\right). \tag{11}$$

It could be seen that for some time point t, as momentum m_{t-1} comes from previous time point $t-1$, in current epoch we only need to summing the gradient of current moment $\nabla^{(t)}$ with m_{t-1}. And complexity of this calcultion is constant.

For our method, we cannot use the above cumulative transfer process. To analyze the reason, we assume that g_i^j denotes jth gradient member of ith gradient cluster $\widetilde{G}_i$. Assuming at some time point t there exists a specifical member of sampling gradient set $g_i^j \in \widetilde{G}$, and m_t can be derived from the following calculation process:

$$m_{t-r} = \sum_{n=1}^{t-r-1} \rho^n \left(\eta \nabla^{(t-n)}\right)$$
$$m_t = \sum_{n=t-r}^{t-1} \rho^n \left(\eta \nabla^{(t-n)}\right) + m_{t-r} \tag{12}$$

where r represents the distance of g_i^j relative to the current moment. According to our method process, we assume that the gradient of the current moment belongs to the same gradient cluster $\widetilde{G}_i$ as g_i^j. To keep gradient momentum m_{t+1} still idd concerning data labels, we need to replace g_i^j with the gradient of current moment $\nabla_{(t)}$ in the calculation process of momentum. But according to (12), replacing g_i^j will cause gradient momentum on moment $t-r$ to t to be recalculated. Since it needs to sample from the set of historical gradients and perform gradient aggregation on these sampled sets in a single round, it may become a performance hindrance when the number of model parameters is huge, even if the server computational power is generous compared to the client. For this reason, we propose inert gradient sampling updates to alleviate this possible performance problem.

In many federated learning scenarios, a round of the training process often involves only a small number of clients. It means that for most of the clients, it may not produce gradient updates for many rounds. Further, since this article is based on both the interclient labels and the temporal distribution of label tendencies non-IID, the clients within a cluster may not or only a small number of clients may participate in training for a long time. So it may not be necessary to repeat the sampling for these clusters frequently. For this reason, we reconstruct the historical gradient information once after several rounds of n in the improvement. Thus, in the actual iterative update round, the optimizer utilizes the inert updated historical gradient information and performs one update using the current gradient information to generate the actual model update. Taking momentum as an example, the actual update calculation process becomes

$$\Delta\theta = \beta m_i + \eta \sum_{c \in \widetilde{C}} \Delta\theta_c^t, \ \ i = \{n, 2n, \ldots, T\}. \tag{13}$$

It can be seen that for specific inert step n, the calculation overhead regarding historical gradient information will be $(1/n)$ of the original one.

B. Non-IID Data Set

To validate the effectiveness of our designed algorithm, we adapted the existing data set by basing it on the heterogeneity generated by the Dirichlet distribution. For the α parameter of the Dirichlet distribution, we set two different parameters to control the two different non-IID presentations. The first parameter α_l controls the heterogeneity of the label distribution within clients. If we assume that the probability of presence of each label in the overall data distribution is uniform, then, the random variable about the probability of label presence on the client satisfies $Y \sim \mathcal{D}(\alpha_l, \mathcal{U})$ ($\mathcal{D}$ is the Dirichlet distribution). The second parameter α_t controls the heterogeneity within the client set regarding the preferences of different label distributions, assuming that the client label set statistical distribution can be expressed as a d-dimensional vector $L \in \mathbb{R}^d$, then, the random variable will satisfy $L \sim \mathcal{D}(\alpha_t, \mathcal{U})$. By adjusting these two parameters, we can construct the set of clients under different non-IID scenarios corresponding to practice.

To more realistically simulate the preferences of the participating clients on the label distribution at different moments, we assume that the probability of presenting a moment-specific label preference at a moment is α_q, while the probability of presenting other label preferences is $1-\alpha_q$. Simplistically and without loss of generality, to describe whether two clients belong to the same label preference, we cluster the label distributions of different clients in constructing the experimental setting to determine the preference class to which the client belongs.

VI. Security Analysis

In our system model, the attack could come from a semi-honest center server. The server can get information generated by clients based on clients' local data. A malicious server can perform an attack as the existing work describes [31], [32]. By performing attack based on the information exchanged with server by clients, the server can recover data that is close to or exactly the same as the client training data. If that happens, clients' privacy information may be leaked to the attacker and may lead to greater losses for clients.

To protect clients from attack on server, we introduce the differential privacy mechanism to protect information send to the server during the federated learning training process. In FedGS, there are two types of information exchanged with the server, respectively, label distribution and gradient. By adding noise determined by differential privacy mechanism, existing and potential attacks on this data can be alleviated or even avoided. But due to noise reducing the validity of information, there is a tradeoff between our performance and

protection strength. So it is necessary to analyze the property of differential privacy mechanism under each part in FedGS.

A. Security of Client Preclustering

By combining existing privacy-preserving mechanisms, we can guarantee that the label distribution passed from the client preclustering process to the server is protected with the same strength as the gradient of the training process. To analyze the differential privacy properties of the preclustering process concerning the overall federated learning process, we introduce the following theorem.

Lemma 1 (Sequential Combination Theorem) [33]: Suppose a mechanism M consists of a sequence of submechanisms $\{M_1, \dots M_n\}$ serially combined, and each subprocess satisfies ϵ_i differential privacy by itself. For this case, it is noted that the sequence $M_i(X)$ should satisfy the differential privacy of $\sum_i^n \epsilon_i$, where X is the output of the previous item.

Taking combining differential privacy as an example, depending on the training process each client's training result as the output of function $f(\theta, D_i)$ and label distribution as the output of function $L(D_i)$, according to the combination theorem of differential privacy, for client c, if the differential privacy mechanism $\mathcal{M}(L_c(D_i))$ satisfies a given privacy budget ϵ_L under ϵ_L differential privacy, and the differential privacy mechanism $\mathcal{M}(f(\theta, D_i))$ satisfies ϵ_f differential privacy under a given privacy budget ϵ_f, then, according to Lemma 1, the differential privacy sequential combination theorem, the total privacy budget of the whole process $\mathcal{M}(\{L\} \cup \{f_t(\theta, D_i), t = 1 \cdots T\})$ in the worst case will be of the (14). This means, that for a given privacy budget ϵ, we can treat it as a whole with differential privacy protection. Therefore, with a suitable choice of privacy budget, the label distribution and gradient submitted by clients can be guaranteed to be privacy-safe over the training process

$$\epsilon = \epsilon_L + \sum_{t=1}^{T} \epsilon_f. \tag{14}$$

We can analyze the privacy budget for differential privacy in the preclustering process. For a specific client, concerning the classification task whose number of label types is c, then its normalized label distribution is $L_i \in [0, 1]^c$. Due to the design of the normalized distribution, we can consider L as a discrete random variable with a finite value domain, and the defined domain $D_L \in [1, c] \cap \mathbb{Z}$ and whose probability distribution function $F_L(x)$ satisfies the constraint

$$F_L(x) = \begin{cases} 0, & x < 1 \\ \sum_{x=1}^{c} f_L(x) \in [0, 1], & 1 \le x \le c \\ 1, & x > c \end{cases} \tag{15}$$

where the probability density function satisfies the constraint $f_L(x) \in [0, 1]$ in its definition domain. Under the satisfaction of this constraint, a family of label distribution functions S_L with different distributions can be constructed. Then, we can give the sensitivity about the label distribution based on the mathematical concept related to the distance between distributions

$$\Delta L = \arg \max_{L_i, L_j \in S_L} ||L_i - L_j||_p. \tag{16}$$

Given the privacy budget required for each client regarding this process in data sets where each client has the same label space, we can easily give the amount of noise added based on the form of noise added in the specific differential privacy and vice versa.

In particular, with the increased heterogeneity of label data, the privacy budget consumed by the process regarding the label distribution decreases as the sensitivity of $L(D_i)$ decreases due to the convergence of the individual sample labels. And since the label distribution is only queried to clients once during the whole federated learning training process, its total privacy budget consumption relative to the gradient during iterative training can be almost negligible.

B. Security of Gradient Sampling

In FedGS, the optimizer tries to give the actual amount of model updates on the current server at a given moment using historical gradients. In this process, a subset of gradients $\widetilde{G}$ is assumed to be sampled, from which the optimizer extracts either first-order momentum m or second-order momentum information v to correct the current gradient $\widetilde{\nabla}$.

To analyze the differential privacy property of this process, we introduce here the parallel combination theorem of differential privacy.

Lemma 2 (Parallel Combination Theorem) [33]: Suppose a mechanism M consists of a sequence of submechanisms $\{M_1, \dots M_n\}$ combined in parallel, and each subprocess satisfies ϵ differential privacy by itself. Suppose D_i is any data set that satisfies the input requirements of mechanism M_i. For this case, the sequence $M_i(X \cap D_i)$ should satisfy ϵ differential privacy, where X is the output of the previous item.

For each sampled historical gradient $g_i \in \widetilde{G}$, it is assumed that it was submitted to the server at some historical moment in time as satisfying the ϵ differential privacy mechanism. For the generalized optimizer process using historical gradients described by (7), it can be considered as consisting of a sequence of subprocesses $\nabla_{i,j} = w_i \cdot \tau(g_j^i)$. According to the parallel combination theorem for differential privacy of Lemma 2, we can conclude the following.

Corollary 1: In the gradient sampling process of federated learning, suppose that for any submechanism $\nabla_{i,j}$ satisfying ϵ of differential privacy, then, its parallel combinatorial process $\nabla = \sum_{i=1}^{t} \sum_{j=1}^{N} \nabla_{i,j}$ should also satisfy ϵ differential privacy.

Thus, relative to the differential privacy process proposed in [28], FedGS does not change the differential privacy property of the actual update Δ after the final aggregation for the model aggregation process.

C. Security of Different Clients Size

The strength of protection that the same privacy budget can provide varies from scenario to scenario. To analyze the attack resistance of FedGS concerning different scenarios, we introduce the following theorem.

Lemma 3 [34]: For a particular process M and data set D, during the T rounds of training with data sampled at a sampling rate q, if for differential privacy noise variance σ satisfies $\sigma < (1/[16\sigma])$, for any $\epsilon < c_1 q^2 T$ there exist two constants c_1

and c_2, under *delta* > 0 $\mathcal{M}$ satisfies ϵ differential privacy with probability δ of failure, where each variable needs to satisfy the following constraint:

$$\epsilon \geq c_2 \frac{q\sqrt{T\log(1/\delta)}}{\sigma}. \tag{17}$$

Assuming exist a federated learning scenario in which the server may be semi-honest, such as an Internet company collecting data from clients for training, and the clients are mutually distrustful with the company and the clients. In the semi-honest model, the server performs the federated learning protocol correctly but retains and tries to derive information about clients from the communication messages and may perform gradient-based attacks. To be able to reduce the possibility of such attacks more effectively, we consider a specific scenario, where the number of clients in federated learning is much larger than the number of training rounds and the number of clients involved in each training round is much smaller than the number of clients. This situation is equivalent to the case where the client sampling rate q in each training round is reduced to close to $(1/|C|)$. According to Lemma 3, when the sampling rate q is reduced, the privacy budget ϵ can be smaller with the same other parameters, corresponding to less noise and higher data validity, or the differential privacy failure probability δ can be reduced to obtain stronger protection validity. Thus, the limited number of queries to a client, in this case, allows the server to obtain less information about a specific client, and the client can obtain stronger privacy protections against possible attack processes from the server with the same amount of noise.

VII. Experiments

In this section, we verify the effectiveness of the algorithm designed in this article by performing it in simulation environments. To demonstrate the effectiveness, we compare the performance between the algorithm of this article and the existing algorithms. Further, we validate some of the ideas revealed in the experiments.

A. Experimental Setup

1) Federated Data Sets: In our experiments, we use existing data sets to construct the local data of clients that are available in federal learning. To keep the experiment simple without losing generality, we choose the Fashion-MNIST [35], SVHN [36], and CIFAR10 [22] data sets as the data sources in our experiments here. The properties of the datasets are shown in Table III. To construct a federated learning scenario with strong non-IID for local data distribution on the clients, we chunk-cut the data set under the control of the α_l parameter to construct local data sets on the clients. To simulate the temporal local invariance of label bias, we perform a temporal ordering on the final selected set of clients with respect to label bias based on the parameter α_l. In addition, to simulate the periodic fluctuations of data that exist in reality, we restrict the participation of clients in federated learning to be periodic with respect to the size of the client set, and the probability of participation at this specific moment is controlled by the parameter α_q.

2) Implementation: For the parameter design in federated learning, in order to produce stronger non-IID to demonstrate the effectiveness of our algorithm in extreme scenarios, for multiple iterations of federated learning, the server only takes one client in each iteration to participate in the training. For other parameters, we set the amount of client data $|D_i| = 20$ and the total number of clients $|C| = 10K$ throughout the experiments. Simulating a scenario of sparse participation in the temporal order, each client participates only once in the training process. For different specific optimizer applications, which have different optimal learning rates in the experiments, we set the same learning rate for the same optimizer in different environments to ensure comparability. With such parameter settings, we use a budget of $\epsilon = 10$ for differential privacy preservation of the client label distribution. For the differential privacy protection of the gradient during training, we refer to the implementation in [28] and set the parameters $\epsilon = 0.1, \sigma = 1.0$, and the cropping threshold $\hat{M} = 100$.

B. Performance Comparison

In this experiment, we compare existing federated learning methods as well as our proposed method regarding the test set accuracy as well as our defined area under the curve (AUC) metric on a temporal non-IID environment constructed from different data sets. The implementation of the existing optimization algorithms for combining momentum, etc., in federated learning can be found in [24]. To illustrate the effectiveness of our method in temporal non-IID scenarios, we set $\alpha_l = 0.001, \alpha_t = \infty$, and $\alpha_q = 0.3$ to build several different environments and compare the performance between the algorithms. In our experiments, we combine this article's method with momentum, AMSGrad, and AdamW optimizers to perform comparison experiments with existing non-IID methods, including FedAvg, FedProx, and FedAwS, respectively.

In order to be able to reflect a comprehensive evaluation of convergence speed and final accuracy metrics in federated learning, we introduce the evaluation metric of the AUC here, which evaluates the area enclosed by the training curve and epoch axis from the beginning of training to the final convergence. In our experiment setting, we perform federated learning from epoch $t_{\text{start}} = 0$ to $t_{\text{stop}} = 10\mathbf{k}$ and get the trained federal model. Formally, we define this metric as $\text{AUC}(\text{Acc}(t)) = \sum_{t=0}^{t_{\text{stop}}} \text{Acc}(t)$.

The experimental results are shown in Table II. It can be seen that for training in different environments, our method exhibits improved performance relative to existing optimizers. For this performance improvement, we suggest that it is due to the ability of our method to utilize the gradient information from clients with different label distributions in an IID manner and effectively improve the correctness of the momentum estimation on single-step updates. Under non-IID, due to the exponentially decaying accumulation mechanism of momentum, the direct application of existing optimization methods tends to make the information of timing distant label distributions in the history overly faded, resulting in incorrect momentum estimation.

TABLE II
COMPARISON OF DIFFERENT METHODS ON DIFFERENT TEMPORAL NON-IID ENVIRONMENTS

Dataset	Optimizer	FedAvg Acc(%)	FedAvg AUC	FedProx Acc(%)	FedProx AUC	FedAwS Acc(%)	FedAwS AUC	FedGS (ours) Acc(%)	FedGS (ours) AUC
Fashion-MNIST	AdaBelief	74.12 (1.8)	34.45 (0.7)	74.17 (1.8)	34.30 (0.7)	74.37 (1.9)	34.45 (0.7)	**79.98 (1.0)**	**35.58 (0.3)**
	AMSGrad	72.90 (2.1)	33.55 (1.2)	72.95 (1.8)	33.52 (1.2)	73.44 (2.1)	33.68 (1.2)	**79.98 (0.8)**	**35.45 (0.4)**
	AdamW	72.02 (1.7)	31.05 (1.6)	72.17 (1.9)	31.12 (1.7)	72.75 (2.4)	31.43 (1.6)	**74.90 (2.5)**	**33.00 (1.1)**
SVHN	AdaBelief	60.25 (2.9)	17.44 (2.4)	60.99 (2.2)	18.01 (2.3)	61.87 (4.9)	18.89 (2.2)	**73.14 (1.0)**	**25.62 (1.7)**
	AMSGrad	56.69 (4.6)	15.19 (1.6)	55.96 (4.8)	15.12 (1.7)	60.40 (3.6)	17.28 (1.7)	**74.56 (1.0)**	**29.20 (0.9)**
	AdamW	53.32 (4.5)	13.14 (1.1)	53.27 (4.5)	13.10 (1.1)	57.81 (3.4)	14.90 (1.2)	**68.55 (1.4)**	**20.20 (1.9)**
CIFAR10	AdaBelief	26.86 (2.4)	11.41 (0.9)	26.86 (2.1)	11.43 (1.0)	27.12 (3.1)	11.44 (1.3)	**36.65 (0.7)**	**13.78 (0.4)**
	AMSGrad	21.88 (1.5)	10.20 (0.5)	22.09 (1.5)	10.31 (0.5)	21.41 (1.6)	9.900 (0.4)	**38.96 (0.8)**	**15.91 (0.4)**
	AdamW	21.95 (2.0)	9.638 (0.6)	22.20 (2.0)	9.700 (0.6)	21.17 (1.8)	9.338 (0.6)	**30.42 (0.8)**	**11.50 (0.7)**

Notes. The std error is shown in parentheses. AUC and its error are shown at $\times 10^{-1}$.

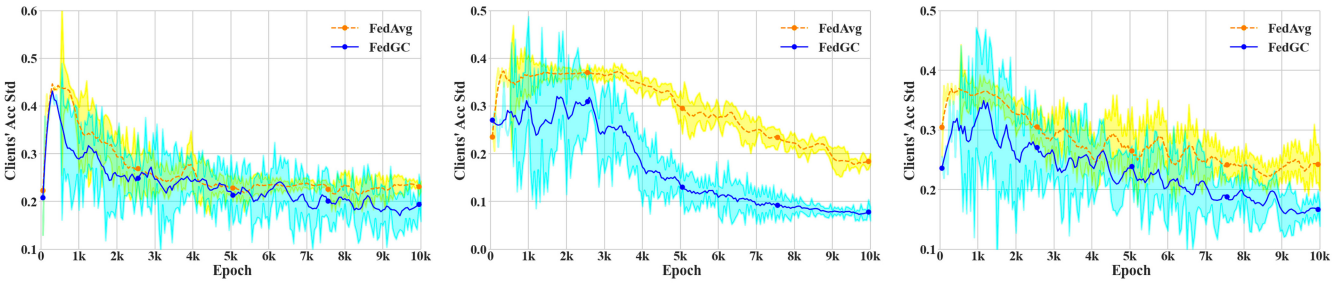


Fig. 5. Accuracies' standard deviation on clients' local data set under different data set environments. (a) Fashion-MNIST. (b) SVHN. (c) CIFAR10.

TABLE III
FEDERATED DATA SETS IN EXPERIMENT

Dataset	Samples	Classes	Images size
Fashion-MNIST	Train 60,000 / Test 10000	10	28×28 Gray
SVHN	Train 73257 / Test 26032	10	32×32 RGB
CIFAR10	Train 50000 / Test 10000	10	32×32 RGB

This performance improvement is not constant but is influenced by different environment settings. For most of the experimental settings, our method shows some improvement in the final model classification accuracy with respect to existing optimizers. In addition, it can be seen that the method significantly outperforms existing optimizer methods in terms of convergence speed for most of the temporal non-IID cases in the experiments. Combining these two experimental metrics, it can be concluded that our method is able to converge more rapidly to better accuracy performance than existing methods under temporal non-IID, which is meaningful in the context of communication-constrained federated learning scenarios.

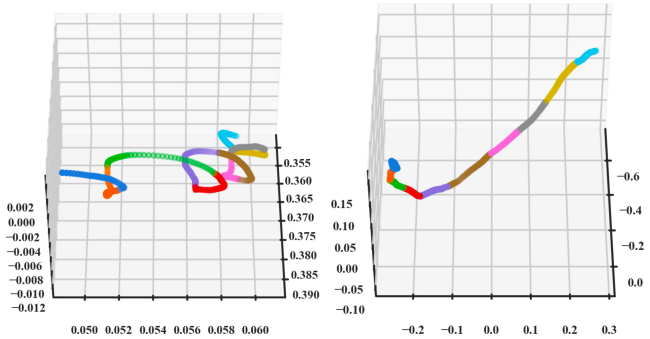
Although our proposed method can achieve better performance in many scenarios, we can combine our method with existing methods to better performance concerning the baseline of these methods. Table IV shows the experiments result that combining FedGS with FedProx, FedAws, or both of them. It can be seen that compared with the baseline performance concerning these existing methods, combined methods can archive better accuracy and convergence speed which is shown by AUC in many experiment scenarios. Based on this result, we can suggest that for the federated learning framework, the improvement points of FedGS for the optimizer do not conflict with the improvement points of FedProx and FedAws for the regular terms of the gradient and output layer parameters, respectively. Therefore, the results of the existing methods combined with FedGS each other will not affect the effectiveness of FedGS. Meanwhile, the result also shows the expandability of FedGS to combine with existing non-IID federated learning solutions. However, it can be seen that for different environments, different combinations of methods may need to be chosen to achieve the optimal performance.

We compare accuracy of clients' local data sets during the training process between FedGS and FedAvg. We construct this experiment on CIFAR10 data set and AMSGrad optimizer. Results are shown in Fig. 5. It can be seen that during the federated learning training process, our proposed method can achieve lower accuracy deviation among clients compared to FedAvg, which means that our method can make the federal model more appliable to each client at the same training

TABLE IV
COMPARISON OF DIFFERENT METHOD COMBINATIONS ON DIFFERENT TEMPORAL NON-IID ENVIRONMENTS

Dataset	Optimizer	FedGS Acc(%)	FedGS AUC	FedGS+Prox Acc(%)	FedGS+Prox AUC	FedGS+AwS Acc(%)	FedGS+AwS AUC	FedGS+Prox+AwS Acc(%)	FedGS+Prox+AwS AUC
Fashion-MNIST	AdaBelief	79.74 (0.9)	35.60 (0.2)	79.79 (1.1)	35.58 (0.1)	**79.98 (0.7)**	**35.65 (0.1)**	79.88 (0.9)	35.65 (0.2)
	AMSGrad	**80.18 (1.0)**	**35.48 (0.4)**	80.18 (1.1)	35.40 (0.4)	80.08 (0.8)	35.48 (0.4)	79.98 (0.9)	35.48 (0.4)
	AdamW	74.71 (2.5)	32.98 (1.0)	74.71 (2.5)	33.00 (1.0)	74.85 (2.9)	**33.02 (1.0)**	**75.00 (2.5)**	33.00 (1.0)
SVHN	AdaBelief	**73.34 (1.4)**	25.45 (1.1)	72.36 (1.6)	25.49 (1.6)	72.17 (1.3)	25.83 (1.0)	72.22 (1.2)	**26.03 (1.4)**
	AMSGrad	**74.80 (1.0)**	29.25 (1.0)	74.76 (0.9)	29.23 (0.9)	74.02 (0.8)	**29.30 (0.9)**	74.02 (1.0)	29.28 (0.9)
	AdamW	68.85 (1.3)	20.25 (1.9)	68.75 (1.5)	20.08 (1.9)	**68.90 (1.6)**	**20.40 (1.9)**	68.65 (1.3)	20.34 (1.8)
CIFAR10	AdaBelief	**36.87 (0.7)**	13.84 (0.4)	36.62 (0.7)	13.64 (0.6)	36.43 (0.8)	13.91 (0.4)	36.01 (0.5)	**13.94 (0.4)**
	AMSGrad	39.01 (0.6)	**15.94 (0.4)**	**39.16 (0.6)**	15.91 (0.4)	38.75 (1.0)	15.91 (0.4)	38.70 (0.9)	15.93 (0.3)
	AdamW	30.42 (0.9)	11.54 (0.7)	**30.71 (1.0)**	**11.64 (0.7)**	30.27 (0.7)	11.51 (0.7)	30.42 (0.8)	11.56 (0.7)

Notes. The std error is shown in parentheses. AUC and its error are shown at $\times 10^{-1}$.



(a) (b)

Fig. 6. PCA plot of model layer parameters. (a) FedAvg. (b) FedGS.

moment. In practical situations, clients want to get access to available early after they participated in the training process, rather than wait for the whole federated learning process to be over. And FedGS can achieve this better.

1) Analysis of the Effect on Convergence: We illustrate the effect of FedGS on the convergence path by visualization. We plot the visualization of the convergence paths during training by performing PCA dimensionality reduction on the last layer of parameters of the model and projecting them into the 3-D space. The results are presented in Fig. 6. We choose the parameters for the first 400 rounds of the training process to plot. The different colored segments in the figure represent a temporal non-IID cycle, implying that a subset of clients clustered with different label distributions was each sampled.

Fig. 6(a) shows the convergence curve under FedAvg. The effect of non-IID causes the convergence curve of the model to show a back-and-forth rotation path within and between cycles, which makes the convergence of the optimization process take longer. Fig. 6(b) shows the convergence curve of the model under FedGS, and we can find that the optimization algorithm can produce a relatively smooth convergence curve through the FedGS gradient scheduling mechanism.

TABLE V
WITH SVHN DATA SETS AND FEDGS. (a) WITH SVHN DATA SETS AND ADABELIEF OPTIMIZER. (b) WITH SVHN DATA SETS AND FEDGS

(a)

FedAvg	FedProx	FedAws	FedGS
11±0.2 ms	14±0.2 ms	11±0.2 ms	12±0.2 ms

(b)

Momentum	AMSGrad	AdamW	AdaBelief
11±0.2 ms	12±0.2 ms	12±0.4 ms	13±0.1 ms

2) Analysis of Communication and Time: In FedGS, since the improved utilization process regarding the history gradient is performed on the server, it makes the process transparent to the clients, which means that the clients do not need to perform additional communication to support the execution of FedGS. The limited amount of additional communication occurs when the client reports its noise-added label distribution to the server before the federated learning training starts. Assuming that the local data set for client c has $|L_c|$ distinct labels, only $4|L_c|$ bytes additional communication overhead is spent in the 32-bit floating-point representation. Thus, under the same conditions, our method has the almost same amount of communication as FedAvg. This also allows our method to work better in communication-restricted scenarios.

To show the impact of our method for federated learning elapsed time, we compare the difference of elapsed time between our method and existing methods under different conditions. In Table V(a), you can see that FedGS only slightly increases the time spent per round of federated learning compared to existing methods. In the scenario, where the federated learning clients and servers have asymmetric computational capabilities, FedGS can achieve a training time overhead close to that of FedAvg when the server is a high-performance data center, since the improvement point of the momentum computation under the non-IID historical gradients in FedGS occurs only on the server. Also, FedGS can achieve competitive training time overhead compared to FedProx and FedAws methods,

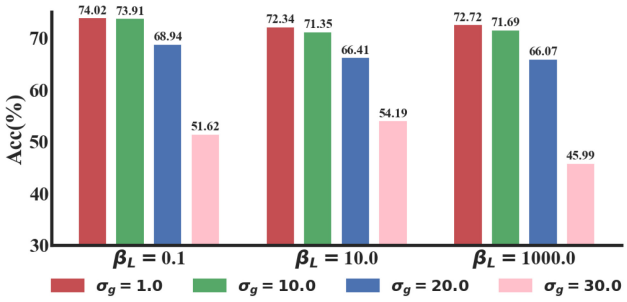


Fig. 7. Model performance with different β_L and σ_g.

which also improve only the server computation part. And since different optimization algorithms have different computational complexities, as shown in Table V(b), the FedGS time consumption increases when combined with more complex optimization methods.

In federated learning scenarios, it is common that the clients have limited computational performance while the servers have more computational power. Since the computation process of FedGS is performed only on the server, the increased computational complexity can be compensated by enhancing the server performance without significantly increasing the overall runtime of federated learning, which is more easily dealt with.

3) Analysis of Differential Privacy: In our proposed methods, there are two differential privacy steps that make up the overall federated learning process. The first step is differential privacy of clients' label distributions, which scale is controlled by parameters ϵ_L, where ϵ_L is the privacy budget of the Laplace mechanism. Due to we can simply convert ϵ_L and Laplace noise scale β_L to each other under only one step query, from a more understandable perspective, we use parameter β_L to denote privacy protection strength. The second step is differential privacy of clients' gradients during the federated model training, which scale is controlled by the parameter σ_g, where σ_g is the noise scale of Gaussian mechanism [28]. In this experiment, we construct different federated learning processes with different β_L and σ_g on SVHN data set and AdamW optimizer. The results are shown in the following Fig. 7.

It can be seen that model performances after convergence varies under differential privacy protection strength. With the increase of parameter σ_g, model accuracy decreases gradually. But we note that in a large scale of parameter β_L that ranges from 0.1 to 1000 compared with the smaller range of σ_g, model accuracy remains relatively constant under the same σ_g setting. This result shows that our preclustering process only needs a small privacy budget to ensure the validity of add-noise information about clients' label distribution.

4) Analysis of Client Data Set Size: The size of the client's local data set in federated learning is often an important factor affecting the convergence results. To illustrate that our method is robust to training results on different local data set sizes, we perform experiments comparing with FedAvg on environment design for different local data set sizes to illustrate the stability of performance improvement of our method. We construct this experiment on AMSGrad optimizer. The experimental results are shown in Fig. 8. It can be seen that on different data sets and client data set sizes settings, our method has different degrees of improvement in post-convergence accuracy relative to FedAvg, and the convergence speed of training is significantly faster on some data sets as SVHN.

5) Analysis of Inert Sampling: We conduct comparative experiments under different inert update steps for the improved Adam optimization process based on our method in the non-IID environment of $\alpha_l = 0.001, \alpha_t = \infty$, and $\alpha_q = 0.3$. We construct this experiment on AMSGrad optimizer. The experimental results are shown in Fig. 9. From the experimental results, it can be seen that there is no significant difference between the experimental setting with $n = 10$ and $n = 1$, which computes updates in every round. However, as n increases to 40 and 100, the model performance has a significant decrease. Thus, the choice of inert steps should be limited to a range, otherwise, it may produce a large degradation in the convergence ability. Besides, from curves of different data sets experiments, it can be seen that the model performance under the same inert steps can be different, which means the choice of inert steps needs to consider specific environmental factors.

6) Analysis of Non-IID Parameter α_q: We adjust α_q on the set of taken values $\{0., 0.3, 1.0\}$ and fix $\alpha_t = 0.001$ and $\alpha_l = 10^6$ to verify the behavior of FedGS at different non-IID strengths. We construct this experiment on AMSGrad optimizer. Fig. 10 gives the convergence curves about FedGS and FedAvg. When α_q gradually decreases from 1.0 to 0, the convergence rate of model under starts to gradually slow down, but models performance under FedGS are still better than FedAvg ones on different α_q. The results show that the FedGS algorithm can archive stable performance improvements on different temporal non-IID scenarios. Besides, it can be seen that even under $\alpha_q = 1.0$ setting, which means that there is no temporal non-IID, our proposed method still has significantly better performance than FedAvg baseline. This can be considered that our method also is adaptable to temporal IID but local label distribution non-IID scenario.

7) Analysis of Non-IID Parameter α_t and α_l: We adjust α_t on the set of taken values $\{10^{-3}, 10^{-2}, 10^{-1}, 10^{-0}, 10^{2}\}$ and α_l on the set of taken values $\{10^{-1}, 10^{0}, 10^{1}, 10^{3}, 10^{6}\}$ and fix $\alpha_q = 0.3$ to verify the behavior of FedGS at different non-IID strengths. We construct this experiment on AMSGrad optimizer. Fig. 11 gives the model performance comparison between different α_t and α_l. It can be seen that with fixed α_t, the model performance increases as α_l increases. Because as α_l increases, the number of samples of different labels in the global data distribution tends to be the same. On the other hand, with fixed α_l, the model performance increases, and then, decreases as α_t increases. This can be understood as with α_t increasing, data distributions among clients tend to be IID, but on the other hand, it's harder to cluster the gradients correctly according to clients' noise-added label distributions as the distances among distributions decrease. With a larger differential privacy budget setting, there can be smaller noise added to label distribution and FedGS can perform better on the same α_t. Thus, in practice, the choice of differential privacy budgets requires a tradeoff between model performance and security under FedGS.

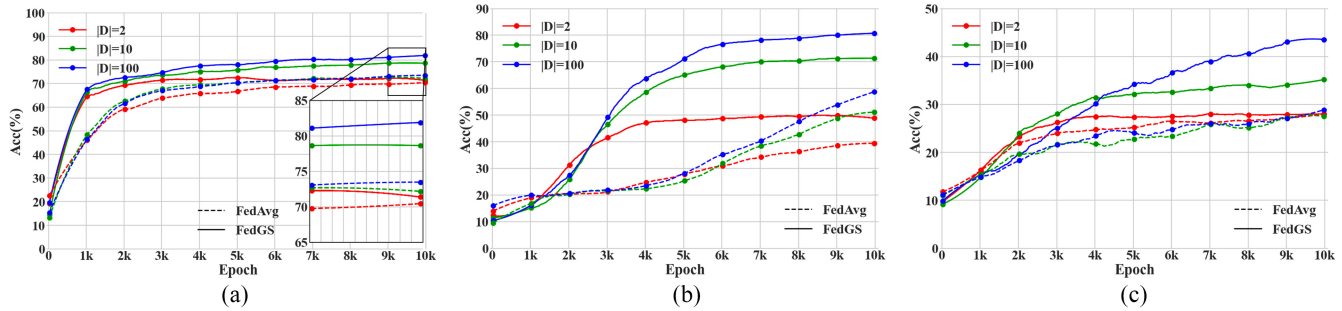


Fig. 8. Comparison of different local data set sizes on model convergence. (a) Fashion-MNIST. (b) SVHN. (c) CIFAR10.

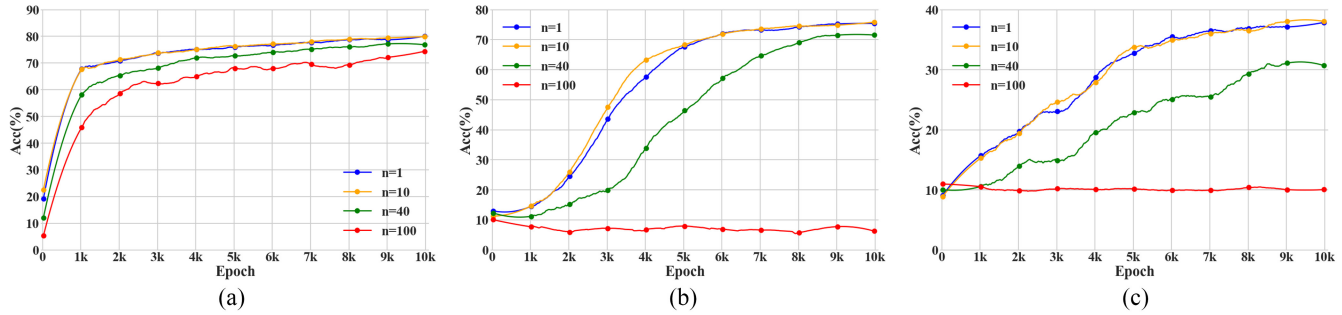


Fig. 9. Comparison of different inert update steps on model convergence. (a) Fashion-MNIST. (b) SVHN. (c) CIFAR10.

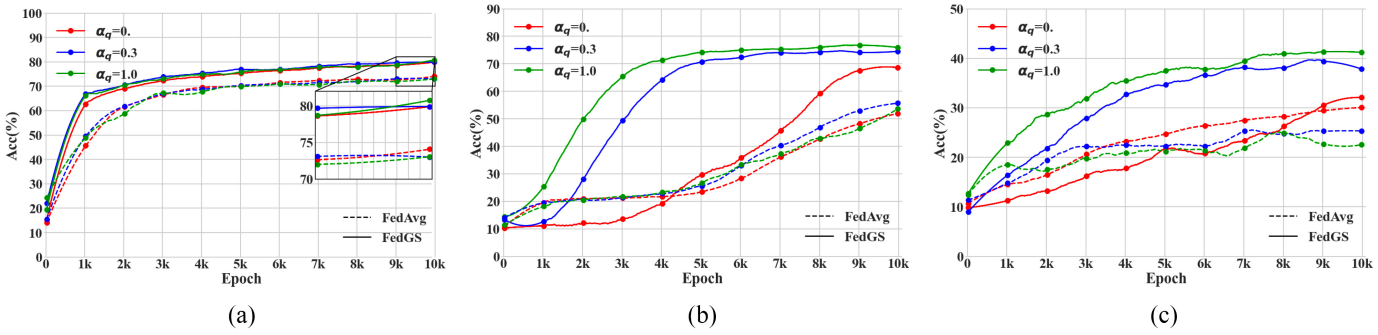


Fig. 10. Comparison of different degrees of α_q on model convergence. (a) Fashion-MNIST. (b) SVHN. (c) CIFAR10.

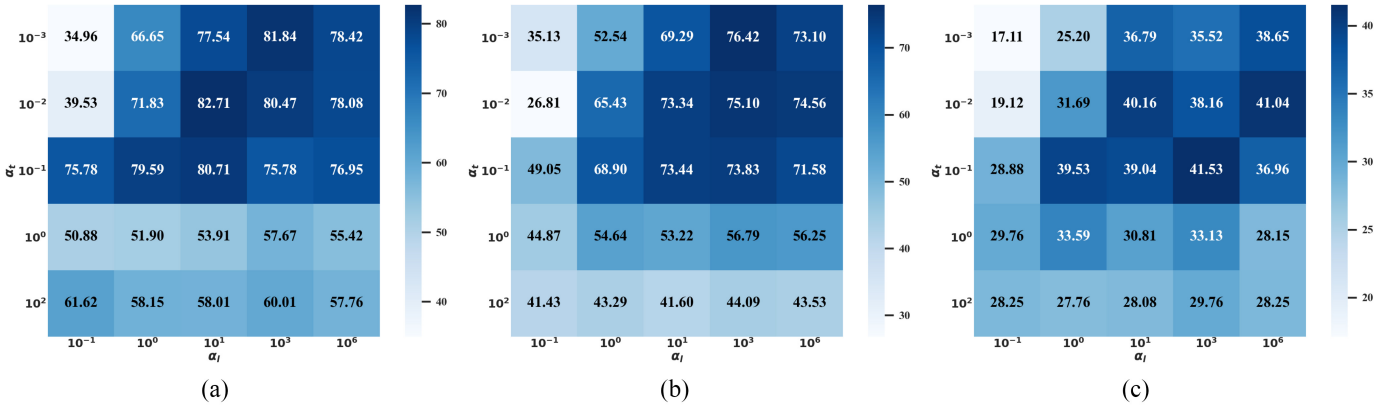


Fig. 11. Model accuracy(%) comparison with different α_t and α_l on FedGS. (a) Fashion-MNIST. (b) SVHN. (c) CIFAR10.

VIII. Conclusion

Due to the problem of data heterogeneity under federated learning, optimizers it incorporates may have difficulty extracting valid gradient information and perform poorly. In this article, we propose a method to provide an approximate IID historical gradient to optimizers by indirectly clustering the historical gradient under federated learning using clients' label distribution information. We validate our method in

environments with different settings set by the non-IID degree parameter of federated learning to show the adaptability of our method to different scenarios. Furthermore, we experimentally demonstrate that FedGS is compatible with other existing federated learning improvement methods, which makes FedGS more feasible to be applied in practice. It can be expected that our proposed FedGS method may have relatively high application value in scenarios with high temporal non-IID, such as those where a large number of clients are involved but a limited number of clients are active at the same time.

Due to the limitations of the experimental environment and the complexity of federated learning itself, we believe that there are still some issues that need to be addressed above this article.

1) *Applicability Under Complex Optimization Algorithms:* In the current experiments, we validate the effectiveness of FedGS for several current environments on typical optimizers. However, it can be predicted that as the complexity of the scenario leads to the presence of more complex optimizers, the effectiveness of the FedGS algorithm in these optimizer scenarios needs to be further verified.
2) *Analysis Under Differential Privacy Protection Technologies:* In this article, we typically analyze the security of FedGS using differential privacy as a privacy-preserving technique. For other privacy-preserving techniques, such as homomorphic encryption and multiparty secure computation, it is necessary to analyze the usability and security of FedGS under these approaches.
3) *Security on Other Role Perspectives:* With the increasing complexity of federated learning, an attacker may be able to obtain different information from additional perspectives to perform an attack. For example, an attacker may control multiple clients or even include the server to obtain private information against other clients. In addition, a scenario from a backdoor attack from the clients is described in [20]. Since the different perspectives may lead to the failure of security guarantees of existing methods, it is necessary to analyze the security of FedGS in these scenarios.

REFERENCES

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

[2] T. Li, A. K. Sahu, M. Zaheer, M. Sanjabi, A. Talwalkar, and V. Smith, "Federated optimization in heterogeneous networks," 2018, *arXiv:1812.06127*.

[3] X. Li, K. Huang, W. Yang, S. Wang, and Z. Zhang, "On the convergence of fedavg on non-iid data," 2019, *arXiv:1907.02189*.

[4] Q. Li, Y. Diao, Q. Chen, and B. He, "Federated learning on non-iid data silos: An experimental study," 2021, *arXiv:2102.02079*.

[5] S. J. Reddi, S. Kale, and S. Kumar, "On the convergence of adam and beyond," 2019, *arXiv:1904.09237*.

[6] I. Loshchilov and F. Hutter, "Fixing weight decay regularization in adam," in *Proc. ICLR*, 2018, pp. 1–14.

[7] J. Zhuang *et al.*, "Adabelief optimizer: Adapting stepsizes by the belief in observed gradients," 2020, *arXiv:2010.07468*.

[8] Q. Wu, K. He, and X. Chen, "Personalized federated learning for intelligent IoT applications: A cloud-edge based framework," *IEEE Open J. Comput. Soc.*, vol. 1, pp. 35–44, 2020.

[9] P. P. Liang *et al.*, "Think locally, act globally: Federated learning with local and global representations," 2020, *arXiv:2001.01523*.

[10] A. Li *et al.*, "LotteryFL: Personalized and communication-efficient federated learning with lottery ticket hypothesis on non-iid datasets," 2020, *arXiv:2008.03371*.

[11] C. T. Dinh, N. H. Tran, and T. D. Nguyen, "Personalized federated learning with moreau envelopes," 2020, *arXiv:2006.08848*.

[12] D. A. E. Acar, Y. Zhao, R. M. Navarro, M. Mattina, P. N. Whatmough, and V. Saligrama, "Federated learning based on dynamic regularization," 2021, *arXiv:2111.04263*.

[13] C. Dwork, "Differential privacy," in *Proc. Int. Colloq. Automata, Languages, Programming*, 2006, pp. 1–12.

[14] S. Liu, J. Yu, X. Deng, and S. Wan, "FedCPF: An efficient-communication federated learning approach for vehicular edge computing in 6G communication networks," *IEEE Trans. Intell. Transp. Syst.*, vol. 23, no. 2, pp. 1616–1629, Feb. 2022.

[15] C. Wang, X. Wu, G. Liu, T. Deng, K. Peng, and S. Wan, "Safeguarding cross-silo federated learning with local differential privacy," *Digit. Commun. Netw.*, vol. 8, no. 4, pp. 446–454, 2021.

[16] F. Yu, A. S. Rawat, A. Menon, and S. Kumar, "Federated learning with only positive labels," in *Proc. Int. Conf. Mach. Learn.*, 2020, pp. 10946–10956.

[17] 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. (ICPR)*, 2021, pp. 5020–5027.

[18] O. Marfoq, G. Neglia, A. Bellet, L. Kameni, and R. Vidal, "Federated multi-task learning under a mixture of distributions," in *Proc. Adv. Neural Inf. Process. Syst.*, vol. 34, 2021, pp. 1–14.

[19] N. Shoham *et al.*, "Overcoming forgetting in federated learning on non-iid data," 2019, *arXiv:1910.07796*.

[20] Z. Yan, J. Wu, G. Li, S. Li, and M. Guizani, "Deep neural backdoor in semi-supervised learning: Threats and countermeasures," *IEEE Trans. Inf. Forensics Security*, vol. 16, pp. 4827–4842, 2021.

[21] G. Tong, G. Li, J. Wu, and J. Li, "GradMFL: Gradient memory-based federated learning for hierarchical knowledge transferring over non-IID data," in *Proc. Int. Conf. Algorithms Archit. Parallel Process.*, 2021, pp. 612–626.

[22] A. Krizhevsky and G. Hinton, "Learning multiple layers of features from tiny images," Dept. Comput. Sci., Univ. Toronto, Toronto, ON, USA, Rep. TR-2009, 2009.

[23] X. Gastaldi, "Shake-shake regularization," 2017, *arXiv:1705.07485*.

[24] Q. Tong, G. Liang, and J. Bi, "Effective federated adaptive gradient methods with non-iid decentralized data," 2020, *arXiv:2009.06557*.

[25] W. Wei *et al.*, "A framework for evaluating client privacy leakages in federated learning," in *Proc. Eur. Symp. Res. Comput. Security*, 2020, pp. 545–566.

[26] H. Hu, Z. Salcic, G. Dobbie, and X. Zhang, "Membership inference attacks on machine learning: A survey," 2021, *arXiv:2103.07853*.

[27] M. Jegorova *et al.*, "Survey: Leakage and privacy at inference time," 2021, *arXiv:2107.01614*.

[28] R. C. Geyer, T. Klein, and M. Nabi, "Differentially private federated learning: A client level perspective," 2017, *arXiv:1712.07557*.

[29] W. Wei, L. Liu, Y. Wut, G. Su, and A. Iyengar, "Gradient-leakage resilient federated learning," in *Proc. IEEE 41st Int. Conf. Distrib. Comput. Syst. (ICDCS)*, 2021, pp. 797–807.

[30] N. Qian, "On the momentum term in gradient descent learning algorithms," *Neural Netw.*, vol. 12, no. 1, pp. 145–151, 1999.

[31] L. Zhu, Z. Liu, and S. Han, "Deep leakage from gradients," in *Advances in Neural Information Processing Systems*, vol. 32. Red Hook, NY, USA: Curran, 2019.

[32] B. Zhao, K. R. Mopuri, and H. Bilen, "iDLG: Improved deep leakage from gradients," 2020, *arXiv:2001.02610*.

[33] F. D. McSherry, "Privacy integrated queries: An extensible platform for privacy-preserving data analysis," in *Proc. ACM SIGMOD Int. Conf. Manage. Data*, 2009, pp. 19–30.

[34] M. Abadi *et al.*, "Deep learning with differential privacy," in *Proc. ACM SIGSAC Conf. Comput. Commun. Security*, 2016, pp. 308–318.

[35] H. Xiao, K. Rasul, and R. Vollgraf, "Fashion-MNIST: A novel image dataset for benchmarking machine learning algorithms," 2017, *arXiv:1708.07747*.

[36] Y. Netzer, T. Wang, A. Coates, A. Bissacco, B. Wu, and A. Y. Ng, "Reading digits in natural images with unsupervised feature learning," in *Proc. NIPS Workshop Deep Learn. Unsupervised Feature Learn.*, 2011, pp. 1–9.

Xianyao You received the B.Sc. degree from the College of Computer and Data Science, Fuzhou University, Fuzhou, China, in 2021, where he is currently pursuing the master's degree.

His current research interests include privacy and security in federated learning.

Ximeng Liu (Senior Member, IEEE) received the B.Sc. degree in electronic engineering and the Ph.D. degree in cryptography from Xidian University, Xi'an, China, in 2010 and 2015, respectively.

He is Currently a Full Professor with the College of Computer and Data Science, Fuzhou University, Fuzhou, China. He was a Research Fellow with the School of Information System, Singapore Management University, Singapore. He has published more than 250 papers on the topics of cloud security and big data security, including papers in IEEE TRANSACTIONS ON COMPUTERS, IEEE TRANSACTIONS ON INFORMATION FORENSICS AND SECURITY, IEEE TRANSACTIONS ON DEPENDABLE AND SECURE COMPUTING, IEEE TRANSACTIONS ON PARALLEL AND DISTRIBUTED SYSTEMS, IEEE TRANSACTIONS ON KNOWLEDGE AND DATA ENGINEERING, and IEEE INTERNET OF THINGS JOURNAL. His research interests include cloud security, applied cryptography, and big data security.

Prof. Liu's awards include the "Minjiang Scholars" Distinguished Professor, the "Qishan Scholars" in Fuzhou University, and the ACM SIGSAC China Rising Star Award in 2018.

Nan Jiang was born in 1981. He received the Ph.D. degree from Nanjing University of Aeronautics and Astronautics, Nanjing, China, in 2008.

From 2013 to 2014, he was a Research Scholar with the Complex Networks and Security Research Lab, Virginia Tech, Blacksburg, VA, USA. He is currently a Professor with the Department of Internet of Things and the Director of the Intelligent Sensor Networks Lab, East China Jiaotong University, Nanchang, China. He has published more than 60 papers in international journals and conferences. His current research interests lie in the Internet of Things, cyber physical systems, and social networks.

Prof. Jiang received several National Science Foundation of China Grants, the Career Award for Young Scientists of Jiangxi Province of China, and leaders of academic and technical of Jiangxi Province of China. He served as a Program Chair and committee for many international conferences, such as CSS 2019, IEEE EUC 2017, and ISICA 2015. He is a Professional Member of ACM and IEEE, and a Senior Member of the China Computer Federation.

Jianping Cai received the master's degree from Fuzhou University, Fuzhou, China, in 2016, where he is currently pursuing the Ph.D. degree with the College of Computer and Data Science.

His research interests include federated learning, Internet of Things technology, machine learning, differential privacy, and optimization theory.

Zuobin Ying (Member, IEEE) received the Ph.D. degree in computer architecture from Xidian University, Xi'an, China, in 2016.

From 2019 to 2021, he was a Research Fellow with Nanyang Technological University of Singapore, Singapore. He is currently an Assistant Professor with the Faculty of Data Science, City University of Macau, Macau, China. His research interests lie in cloud security, applied cryptography, and blockchain.