Information Sciences 589 (2022) 170–185

Contents lists available at ScienceDirect

Information Sciences

journal homepage: www.elsevier.com/locate/ins

Fairness and accuracy in horizontal federated learning

Wei Huang [a,b,c], Tianrui Li [a,b,c,*], Dexian Wang [a,b,c], Shengdong Du [a,b,c], Junbo Zhang [b,d,e], Tianqiang Huang [f,*]

[a] *School of Computing and Artificial Intelligence, Southwest Jiaotong University, Chengdu 611756, China*
[b] *Institute of Artificial Intelligence, Southwest Jiaotong University, Chengdu 611756, China*
[c] *National Engineering Laboratory of Integrated Transportation Big Data Application Technology, Southwest Jiaotong University, Chengdu, China*
[d] *JD Intelligent Cities Business Unit, JD Digits, Beijing, China*
[e] *JD Intelligent Cities Research, Beijing, China*
[f] *College of Mathematics and Informatics, Fujian Normal University, Fuzhou 350117, China*

A R T I C L E I N F O

Article history:
Received 27 May 2021
Received in revised form 16 December 2021
Accepted 27 December 2021
Available online 31 December 2021

Keywords:
Federated learning
Weighting strategies
Momentum gradient
Information quantity

A B S T R A C T

In the horizontal federated learning setting, multiple clients jointly train a model under the coordination of the central server, while the training data is kept on the client to ensure privacy. Normally, inconsistent distribution of data across different devices in a horizontal federated network and limited communication bandwidth between end devices impose both statistical heterogeneity and expensive communication as major challenges for federated learning. This paper proposes an algorithm to achieve more fairness and accuracy in horizontal federated learning (FedFa). It introduces an optimization scheme that employs a double momentum gradient, thereby accelerating the convergence rate of the model. An appropriate weight selection algorithm that combines the information quantity of training accuracy and training frequency to measure the weights is proposed. This process assists clients in aggregating at the server with a more fair weighting. Our results show that the proposed FedFa algorithm outperforms the baseline algorithm in terms of accuracy and fairness.

1. Introduction

The centralized machine learning approach leads to serious practical problems, such as high communication costs, large consumption of device batteries, and the risk of violating the privacy and security of user data. At the same time, there is a worldwide trend toward tighter management of user data privacy and security, which is reflected in the recent introduction of the General Data Protection Regulation (GDPR) in the European Union [15].

The establishment of data security regulations will clearly contribute to a more civilized society, but will also pose new challenges to the data handling procedures commonly used in AI. In this new legislative environment, it is becoming increasingly difficult to collect and share data between different organizations. In addition, the sensitive nature of certain data (e.g., financial transactions and medical records) prohibits the collection, fusion, and use of data, and forces data to exist in isolated data silos maintained by data owners [35]. As a result, a way has been sought to train machine learning models without having to centralize all data into a central storage point.

* Corresponding author.
E-mail addresses: huangweifujian@126.com (W. Huang), trli@swjtu.edu.cn (T. Li), wangdexian@my.swjtu.edu.cn (D. Wang), sddu@swjtu.edu.cn (S. Du), msjunbozhang@outlook.com (Junbo Zhang), fjhtq@fjnu.edu.cn (T. Huang).

https://doi.org/10.1016/j.ins.2021.12.102
0020-0255/© 2021 Elsevier Inc. All rights reserved.

The concept of federated learning was first introduced by McMahan et al. [24] in 2016. Federated learning has received a lot of attention for its ability to train large-scale models in a decentralized manner without direct access to user data. It helps protect users' private data from centralized collection. In contrast to distributed machine learning, federated learning aims to process Non-IID and unbalanced data from a variety of real-world applications (e.g., apps in smartphones [24], travel mode identification from Non-IID GPS trajectories [42]).

According to the characteristics of federated data, federated learning can be classified into three categories [36]: Horizontal Federated Learning (HFL), Vertical Federated Learning (VFL), and Federated Transfer Learning (FTL). The essence of HFL is the federation of samples, applying to the scenario when there are many overlapping features and few overlapping users. The nature of VFL is the union of features, which applies to the scenario when there are many overlapping users and few overlapping features. FTL can be considered when there is little feature and sample overlap among participants and is mainly applicable to scenarios where a deep neural network is the base model. Horizontal federated networks consist of a large number of devices, so communication speeds in the network are many orders of magnitude slower than local computing speeds. Devices often generate and collect data in a way that is not identically distributed across the network, and the amount of data across devices can vary greatly. This data generation violates the assumption of independent identical distributions commonly used in distributed optimization and increases the complexity of modeling, analysis, and evaluation.

Compared with other machine learning paradiagms, HFL is subject to the following challenges [24]:

(1) ***Non-Independent Identical Distribution (Non-IID):*** The distribution of data in each device is not representative of the global data distribution, i.e., the categories in each device are incomplete. For example, there are 100 categories of images in the full set, and in one device are landscape images, while in another device are people or plant images. The former is one distribution and the latter is another. That is the Non-IID in federated learning. Conversely, if there are 100 categories of pictures in a certain device, and the other devices also have these 100 categories of pictures, then they are identically distributed. In real daily life, data samples distributed across different platforms are not independently and homogeneously distributed. When dealing with Non-IID data, the training complexity of the federated learning model may be greatly increased.

(2) ***Unbalanced:*** The data amount on the clients may be different, because some clients produce a lot of data for model learning and some clients produce less. For example, a device has one hundred pictures in the landscape category and ten pictures in the animal category, while a device has ten thousand pictures in the landscape category and one thousand pictures in the animal category. The difference in the number of samples per device or in the number of samples per category is generally described as unbalanced. Unbalanced data also creates the problem of heterogeneity, which leads to an increase in the training complexity of the federated learning model.

(3) ***Massively distributed:*** The size of the clients participating in federated learning is much larger than the average number of examples per client. The clients for distributed learning are the compute nodes in a single cluster or data center, which are typically $1\sim1000$ clients. Cross-silo federated learning is commonly $2\sim100$ clients. While cross-device federated learning uses massive parallelism and can reach 10^{10} clients.

(4) ***Limited communication:*** Clients that participate in model learning are frequently offline or on slow or expensive connections. Participating in federated learning may be mobile devices such as cell phones and tablets, which may be disconnected from the central server at any time due to the habits of users and the unstable network environment where they are located. Moreover, the devices in federated learning may be distributed in different geographical locations, and they are generally connected to the central server remotely, and the communication cost is higher due to the network bandwidth of different devices. Therefore, with a large number of clients participating in federated learning, communication bandwidth is increasingly becoming a bottleneck.

Generally the entire federated learning network contains a large number of devices and the raw data is stored locally on the remote devices, the devices must constantly interact with the central server to complete the construction of the global model, and the network communication speed is many orders of magnitude slower than centralized computing, which results in high communication cost. Therefore, this challenge of massively distributed is mainly addressed in the challenge of limited communication. In addition, devices usually generate and collect data in different distributions over the network, and the amount of data, features, etc. may vary greatly across devices, so that the data in the federated learning network is Non-IID. Currently, many algorithms are built based on the premise of IID data. And the characteristics of Non-IID data bring great challenges to modeling, analysis and evaluation. Therefore, we focus on the study of communication overhead and statistical heterogeneity.

In this work, we propose FedFa, a horizontal federated optimization algorithm with a double momentum gradient method, and an appropriate weighting strategy based on the information of the frequency of participation or learning accuracy. In particular, the contributions of our work are listed as follows:

- We introduce momentum gradient descent in both client and server to help the gradient converge faster. It can reduce the number of communication rounds for federated learning, which is beneficial for the limited communication of federated learning.

- We use the information quantity about the accuracy and frequency of participation in the training to assign appropriate weight to the client during server aggregation.
- We conduct the experiments on one synthetic and four real datasets to evaluate the performance of the proposed FedFa. As a result, our proposed method has faster convergence and stability in comparison to traditional federated learning.

The remainder of the paper is organized as follows. In Section 2, we provide a related work on federated learning. The detail of the proposed algorithm, FedFa, especially the double momentum gradient method, and appropriate weighting strategies are described in Section 3. In Section 4, we provide a thorough empirical evaluation of FedFa on a suite of synthetic and real-world federated datasets. Finally, conclusions are drawn in Section 5.

2. Related Work

Federated learning faces some challenges including the joint optimization of heterogeneity and expensive communication. The Federated Average Method (FedAvg), as the fundamental framework under the HFL setup, has been improved by numerous researchers [3,17,30].

Heterogeneity is a fundamental problem for federated learning. If heterogeneity is not addressed, the accuracy of the model cannot be improved, making it impossible to put federated learning into practical use. Heterogeneity in federated learning encompasses both systematic and statistical heterogeneity [25,10,6,26]. Zhao et al. [41] argued that the loss of accuracy of federated learning on Non-IID data can be explained by weight divergence, which improves the training of Non-IID data by introducing EMD (earth move distance) distances and sharing a small portion of global data between clients. While this approach does allow for the creation of more accurate models, it has several drawbacks, the most critical of which is that we typically cannot assume the availability of such a public dataset. Li et al. [21] proposed a broader framework based on FedAvg, FedProx, which is capable of handling federated data with statistical heterogeneity, while maintaining similar privacy and computational advantages. The framework does not depend on the particular solver used on each device and can improve the robustness and stability of convergence in heterogeneous federated networks. However, the proximal term constraint is isotropic and is not personalized for solving client heterogeneity problems in federated learning. Li et al. [18] provided a benchmark, NIID-Bench, for the division strategy of Non-IID data, and also conducted a systematic comparison of several algorithms. Huang et al. [11] put forward the federated learning method FedAMP that promotes cooperation of clients with similar data for the Non-IID setting and proposed a heuristic method HeurFedAMP to further improve the performance on deep models. Li et al. [22] introduced a personalization method based on federated multitask learning, the Ditto algorithm, which can simultaneously improve fairness and robustness in federated learning. Li et al. [20] presented the q-Fair method, which enables the reweighting of the loss for different devices by introducing q parameterized weights, thus reducing the variance of the accuracy distribution and achieving a more fair distribution of accuracy.

Communication between the server and edge devices is usually slow and expensive, becoming a bottleneck for federated learning [38,4,37,39]. Therefore, there are many scholars working on communication-efficient federated learning methods. Gao et al. [8] proposed a novel efficient communication distributed stochastic gradient algorithm to address the challenge of large communication overhead in the effective training of large-scale models for federated learning. Li et al. [19] came up with a practical one-shot federated learning algorithm FedKT, which can outperform other federated learning algorithms by a single round of communication. Yapp et al. [40] presented a working prototype of a federated edge learning framework for blockchain enablement and communication efficiency, which enhances the security and scalability of large-scale implementations of FEL. Wang et al. [31] proposed the federated matched average algorithm, FedMA, which utilizes probabilistic matching and model size adaption for modern CNNs and LSTMs architectures and can improve communication efficiency. Karimireddy et al. [14] developed an algorithm called SCAFFOLD that adds an additional parameter control variate to correct the client-drift that occurs in FedAvg, thus speeding up the convergence and reducing the number of communications. Wang et al. [32] introduced a normalized averaging method, FedNova, which normalizes local updates before averaging and can eliminate objective inconsistencies while maintaining fast error convergence. Liu et al. [23] proposed to perform momentum federated learning with momentum gradient descent in the local update step to accelerate the convergence of FL. Xu et al. [34] proposed federated averaging with client-level momentum, FedCM, which uses momentum terms that aggregate global gradient information to modify local gradient updates. Huo et al. [12] investigated the model averaging step of the FedAvg algorithm, and proposed a novel accelerated federated momentum algorithm, FedMom, to improve the convergence speed. These momentum-based approaches mainly consider the momentum update rules from the client-side level. Our approach considers the momentum update rules from both client-side and server-side levels.

There are also many other researches on federated learning, e.g., federated multi-task learning, federated meta-learning, personalized federated learning and federated multi-center learning, etc. Shoham et al. [28] proposed the FedCur framework, which combines the idea of multi-task learning to address how to learn a task in federated learning without interfering with different tasks learned on the same model. Smith et al. [29] introduced the MOCHA framework that connects each task with different parameters and learns the independence of each device by adding loss terms, while modeling task correlations through multi-task learning using shared representations. MOCHA uses a primal pairwise formulation to solve optimization problems, which is limited to convex objectives, and has limited ability to extend large-scale networks and is not suitable for deep networks. Jiang et al. [13] argued that the FedAvg method proposed by McMahan [24] has many similarities with

MAML [7] and can be interpreted in terms of meta-learning algorithms. Xie et al. [33] developed a novel multi-center aggregation method to address the Non-IID challenges of federated learning, and presented the federated stochastic expectation maximization (FeSEM) to solve the optimization problem of the proposed objective function. However, the influence of prior values of multiple centers is significant, and the initialized values are uncertain for the optimization process.

The statistical heterogeneity of HFL leads to the possibility of preferences for certain clients in the training process, i.e., there may be unfairness. We leverage the accuracy and frequency of our clients' involvement in the training process to select the appropriate weights for each round of server aggregation. One of the best ways to reduce network overload is to quickly and stably converge the target accuracy of the learning model in horizontal federated learning. Therefore, we introduce the momentum gradient descent method for the client and server, respectively.

3. Fairness and accuracy in federated learning (FedFa)

In this section, we will introduce fairness and accuracy in horizontal federated learning (FedFa). It is presented in two parts: the double momentum gradient approach and the appropriate weighting strategies. We begin with an introduction to the basic concepts of federated learning. Fig. 1 illustrates a framework for HFL.

In FedAvg, the framework is composed of a central server *PS* and *K* clients. As shown in Fig. 1, the process of the entire algorithm has the following steps for a *T*-round training loop: first, the server randomly selects a subset $K \ll N$ of the total devices. Then, the initialized parameter w_t is sent to these devices, and each device updates the parameter using the local dataset and sends it back to the server. Finally, the server aggregates and averages the parameters sent by these devices.

The pseudo code of FedAvg is provided in Algorithm 1.

Algorithm 1: Federated Averaging [24] (FedAvg)

```
Input parameters $K, T, \eta, E, w^0, N, p_k, k = 1, \dots, N$
Output well-trained $w$
for $t = 0, \dots, T-1$ do
    Server selects a subset $S_t$ of $K$ devices at random (each device $k$ is chosen with probability $p_k$)
    Server sends $w^t$ to all chosen devices
    Each device $k \in S_t$ updates $w^t$ for $E$ epochs of SGD on $F_k$ with step-size $\eta$ to obtain $w_k^{t+1}$
    Each device $k \in S_t$ sends $w_k^{t+1}$ back to the server
    Server aggregates the $w$'s as $w^{t+1} = \Sigma_{k \in S_t} \frac{n_k}{n} w_k^{t+1}$
end for
```

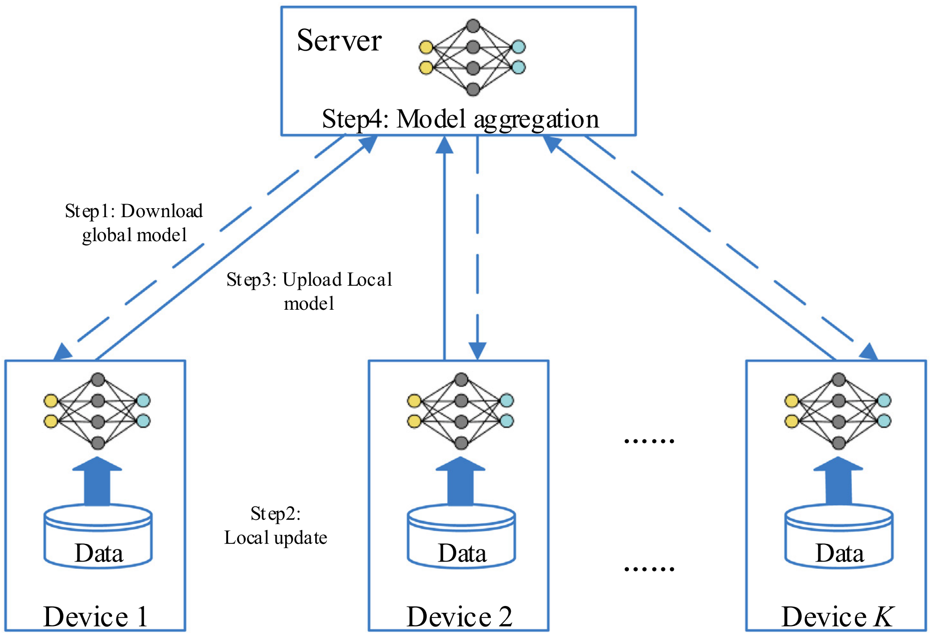


Fig. 1. Framework of a federated learning.

3.1. Double momentum gradient method

The ordinary gradient descent method is sufficient to solve conventional problems, such as linear regression. However, when the problem becomes complex, the ordinary gradient descent method faces many limitations. Specifically, for the ordinary gradient descent formula $w = w - \eta\Delta w$, where η denotes the learning rate and is the step size of the gradient adjustment on each time step. The gradient will be smaller when close to the optimal value. Since the learning rate is fixed, the ordinary gradient descent method will converge slower and sometimes even fall into a local optimum. If the historical gradient is taken into account, it will lead the parameters to converge faster towards the optimal value. The basic idea behind the momentum gradient descent method is to calculate the exponentially weighted average of the gradient and use that gradient to update the weights. If the current gradient descends in the same direction as the last update, the last update can act as a positive acceleration to the current search. Conversely, the last update can act as a deceleration to the current search. Thus, the formula for updating the client's parameters is as follows:

$$m_c = \gamma m_c + \eta\Delta w_k^{t+1}, \tag{1}$$

$$w_k^{t+1} = w_k^t - m_c, \tag{2}$$

where m_c is the client momentum term. γ denotes the influence of the historical gradient, and the larger γ is, the greater the influence of the historical gradient on the present. t represents the round of updates.

It is normal to use momentum gradient descent for the client's model to accelerate convergence. When the server aggregates the gradients of clients in each round, the historical gradients from the previous rounds of aggregation can also be considered. Based on the idea of momentum gradient, it will lead to faster convergence of the server aggregated parameters towards the optimal value and reduce communication. However, there is ordinarily no data on the server side in horizontal federated learning. Therefore, this paper takes the following formula of the server's parameter updates during model training:

$$w^{t+1} = \Sigma_{k\in S_t} weight_k \times w_k^{t+1}, \tag{3}$$

$$\Delta w = w^{t+1} - w^t, \tag{4}$$

$$m_s = \gamma m_s + \eta\Delta w, \tag{5}$$

$$w^{t+1} = w^t - m_s, \tag{6}$$

where m_s is the server momentum term, $weight_k$ is the weight of each client explained in subSection 3.2 and Δw is the server-side global pseudo-gradient. We approximate this gradient on the server side by computing the difference between the latest converged global model ($t+1$ round) and the previous global model (t round). Generally, in federated learning, the data between clients is Non-IID and the selection of clients to participate in the training is random each round. Therefore, we can wait for b rounds before using an approximate method of updating the momentum gradient on the server side (b can take 3, 5, etc.). The purpose of this is to allow the server-side to better capture the impact of the historical gradient. Then, the formula (6) will be changed in the following form:

$$w^{t+1} = w^t - m_s \quad (t+1 = bn, n \in N^*). \tag{7}$$

The value of $t+1$ is an integer multiple of b. It means that the server uses the formula (7) to update every b round. A proper value of b will make the global model consider the historical gradient more comprehensively, reduce oscillations, and ensure efficiency and correct convergence.

3.2. Appropriate weighting strategies

Generally, the number of devices in a federated network is large, which ranges from hundreds to millions. While the general goal of federated learning is to fit the model through some kind of empirical risk minimization objective to the data that is generated by the network of devices. Simply minimizing the average loss in such a large network may overly favor model performance on some devices. Although the average accuracy may be high, the accuracy of each device in the network may not always be very well. The situation is made worse by the reality that datasets are usually heterogeneous between devices in terms of size and distribution. Therefore, in this work, we design a weighting strategy to encourage a more fair distribution of model performance across devices in the federated network.

In FedAvg, it commonly implements a weighted aggregation strategy for each local model based on the size of the training data on the client. It is defined by

$$w^{t+1} = \Sigma_{k\in S_t} \frac{n_k}{n} w_k^{t+1}, \tag{8}$$

where n_k is the sample size of k-th client and n is the total number of training samples. Obviously, the larger the n_k, the greater the contribution of the local model on client k to the central model. Temporally weighted aggregation algorithm was proposed by [4] to make federated learning more efficient in communication. It achieves faster convergence of learning accuracy compared to traditional federated learning. But it only takes into account the latest round of learning for each user.

Aiming to improve learning speed and stability, we propose an appropriate weighting strategy in federated learning. When dealing with Non-IID data, the quality of the client's data may be different and the corresponding training accuracy may be also varied. In this case, the gradient of the local model should be processed differently in the aggregation step. And since the selection of clients to participate in the training is random in each round, the number of times each client participates in the training is also diverse. Therefore, we use the training accuracy Acc_i and the participation frequency f_i as weighting factors. At each training round, the client trains the model locally and sends the training accuracy and the number of training participation along with the gradient to the server. The server then performs a weighted aggregation based on the training accuracy and number of training participation of the clients. Since the amount of data for each client is constant, but the adaptability of the model to the client's data varies from round to round, and the number of times the client participates in the training keeps changing. Therefore, we use the change factor as a measure of the weighted strategy to build a more fair and accurate model in federated learning.

After the client sends information to the server in each round, the server normalizes the client's Acc_i and f_i as follows:

$$Acc_i = \frac{Acc_i}{\sum_{i=1}^{k} Acc_i}, \tag{9}$$

$$f_i = \frac{f_i}{\sum_{i=1}^{k} f_i}. \tag{10}$$

A weighting approach that simply utilizes only the client's data size does not ensure that federated learning is fair in the training process. The amount of information available to the client will vary as a result of the training accuracy and the frequency of participation.

We utilize the information quantity to measure weight. Due to the complexity of the local dataset, there can be significant differences in training accuracy between clients. In general, the lower the training accuracy, the more information the client has to learn. This is also intended to address the preference for certain clients in the federated learning process and to achieve a more fair result.

$$Acc_i\_inf = \left\{ \begin{cases} -log_2 Acc_i, & Acc_i \neq 0, \\ -log_2(Acc_i + c), & Acc_i = 0, \end{cases} \right. \tag{11}$$

where c is a very small constant close to 0. It is to avoid the antilogarithm of logarithms being 0.

The more rounds a client participates in training, the more information it has. On the other hand, the less information it has.

$$f_i\_inf = \left\{ \begin{cases} -log_2(1 - f_i), & 1 - f_i \neq 0, \\ -log_2(1 - f_i + c), & 1 - f_i = 0, \end{cases} \right. \tag{12}$$

where c has the same function as in Formula 11. Then, the information quantity for Acc_i_inf and f_i_inf can be normalized as follows:

$$Acc_i\_inf = \frac{Acc_i\_inf}{\sum_{i=1}^{k} Acc_i\_inf}, (13)$$

$$f_i\_inf = \frac{f_i\_inf}{\sum_{i=1}^{k} f_i\_inf}. (14)$$

We apply weights based on the frequency and training accuracy of learning to participate in the model aggregation process.

$$weight_i = \alpha Acc_i\_inf + \beta f_i\_inf, \tag{15}$$

where, $\alpha+\beta=1$. The formula for model aggregation is proposed based on the client's model accuracy and frequency of participation in learning. α and β are the proportional effects of each method on the aggregated model. The pseudo code of FedFa is provided in Algorithm 2.

Algorithm 2: Fairness and Accuracy Federated Learning (FedFa)

```
Input: parameters $K$, $T$, $\eta$, $E$, $w^0$, $N$, $p_k$, $m_s = 0$, $k = 1, \ldots, N$
Output: well-trained $w$
for $t = 0, \ldots, T - 1$ do:
```

(continued on next page)

a (*continued*)

Algorithm 2: Fairness and Accuracy Federated Learning (FedFa)

```
Server selects a subset $S_t$ of $K$ devices at random (each device $k$ is chosen with probability $p_k$)
Server sends $w^t$ to all chosen devices
Client:
Each device $k \in S_t$ updates $w^t$ for $E$ epochs of gradient descent with Momentum on $F_k$ with step-size $\eta$ to obtain $w_k^{t+1}$
        Each device $k \in S_t$ sends $w_k^{t+1}, Acc_k^{t+1}, f_k^{t+1}$ back to the server
  Server:
        Calculate the amount of information for $Acc_k\_inf$ and $f_k\_inf$
        Update the $weight$ as: $weight_k = \alpha Acc_k\_inf + \beta f_k\_inf$
        Aggregate the $w$'s as $w^{t+1} = \Sigma_{k \in S_t} weight_k \times w_k^{t+1}$
        Calculate the difference between two rounds of $w$ as a pseudo-gradient: $\Delta w = w^{t+1} - w^t$
        Calculate the momentum term: $m_s = \gamma m_s + \eta \Delta w$
        Update the $w$ as: $w^{t+1} = w^t - m_s$
end for
```

4. Evaluation

In this section, we present the empirical results of the proposed FedFa algorithm. In Section 4.1, we describe the experimental setup including the datasets used. Then, we demonstrate the effectiveness of FedFa for statistical heterogeneity in Section 4.2. And the improved fairness of FedFa is illustrated in Section 4.3. Finally, in Section 4.4, we discuss the effect of parameters.

4.1. Experimental Design

Federated Datasets. Our model is compared to baselines on the synthetic and four real datasets to validate our effectiveness and fairness. These datasets are drawn from previous federated learning work [2] [21] [24]. Table 1 summarizes the statistical information for the four real datasets. Details of all datasets are also presented below.

- *Synthetic:*These are three Non-IID datasets which set $(\rho, \varphi) = (0,0), (0.5, 0.5)$ and $(1,1)$, respectively. The increase in ρ and φ indicates that the degree of heterogeneity of the dataset increases. And one IID data which set the same $W, b \sim \mathcal{N}(0,1)$ on every device. In addition, x_k follows the same distribution $\mathcal{N}(v, \sum)$, where each element in the mean vector is zero and $\sum$ is diagonal to $\sum_{j,j} = j^{-1.2}$. There are 30 devices in total for all synthetic datasets, and the number of samples on each device follows the power law. Drawing on prior work [30], we study the effect of data heterogeneity by varying the concentration parameter δ of Dirichlet distribution on the synthetic dataset with δ from $\{0.1, 0.5, 3, 5\}$. For a smaller δ, the partition will be more unbalanced.
- *MNIST:*The MNIST [16] dataset is an image classification of handwritten digits 0–9. The dataset input to the model of multinomial logistic regression is a 784-dimensional (28×28) flattened image and the output is a class label between 0 and 9. Previous studies have distributed the data across 1000 devices. In order to simulate statistical heterogeneity, each device has only 2 digits per sample and the number of samples follows the power law.
- *FEMNIST:*FEMNIST is a federated dataset based on EMNIST with a total of 200 devices. The EMNIST dataset [5] is an image classification problem with 62 classes. In order to form a heterogeneous distribution of devices, the dataset takes 10 lowercase letters ('a'-'j') as subsamples from EMNIST and assigns only 5 classes to each device. A flattened 784-dimensional (28×28) image was used as input to the model of multinomial logistic regression, and the output was a class label between 0 and 9.

Table 1
Statistics of four real federated datasets.

hline Dataset	Devices	Samples	Samples/device	
			mean	stdev
MNIST	1,000	69,035	69	106
FEMNIST	200	18,345	92	159
Sent140	772	40,783	53	32
Shakespeare	143	517,106	3,616	6,808

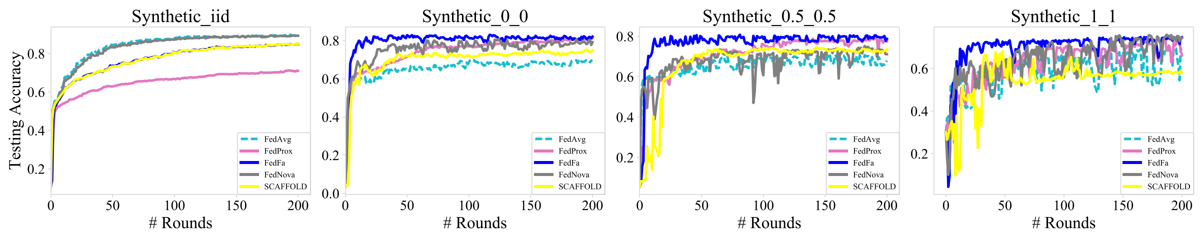
- *Sent140:*This dataset is a collection of Sentiment 140 [9] (Sent140) tweets for the text sentiment analysis task. Each Twitter account corresponds to one device. We model it as a binary classification problem. The model takes a sequence of 25 characters as input, embeds each word in a 300-dimensional space using pre-trained Glove [27], and outputs a binary tag after two LSTM layers and a dense join layer.
- *Shakespeare:*This is a dataset built from *The Complete Works of William Shakespeare* [24]. The model treats a sequence of 80 characters as input, embeds each character in the 8-dimensional space of learning, and outputs a character after two LSTM layers and a tightly connected layer. Each speaking character in a play represents a different device.

Implementation. We implement all the code in TensorFlow [1] Version 1.10.0 to simulate a federated network with one server and K devices. We randomly divide the data on each local device into 80% of the training set and 20% of the test set. For all experiments on all datasets, we fix the number of selected devices per round to 10. For all synthetic data experiments in FedFa, the learning rate is 0.0001. The server and client momentum factors are (0.5, 0.9) or (0.5, 0.5), respectively. For a fair comparison with all synthetic data experiments in FedAvg, FedProx, q-Fair, SCAFFOLD and FedNova, the learning rate is 0.01, 0.001 and 0.0001, respectively. For MNIST, FEMNIST, Sent140 and Shakespeare in FedFa, the learning rate is 0.0003, 0.0003, 0.0001, 0.08, respectively. For MNIST, FEMNIST, Sent140 and Shakespeare in FedAvg [24], FedProx [21], q-Fair [20], SCAFFOLD [14] and FedNova [32], we use the learning rates of 0.03, 0.003, 0.3, and 0.8, respectively. The control variables of SCAFFOLD are updated using the II option, which is based on the difference between the global and local models. The parameter E of FedNova is chosen to be 5. And the parameter q of q-Fair is set to 1.

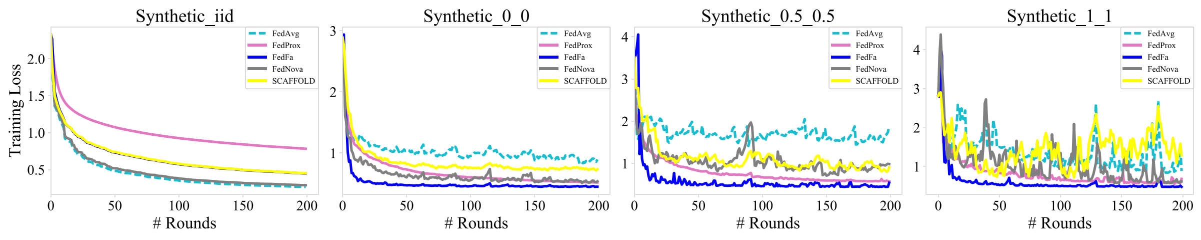
4.2. Effects of double momentum gradient

We use four synthetic datasets to study how statistical heterogeneity affects convergence (fixing E to be 20), and to demonstrate the effectiveness of our double dynamic mechanism. In Fig. 2, we compare the FedFa algorithm (which only has the momentum mechanism without applying the weighting strategy) with FedAvg, FedProx, SCAFFOLD and FedNova with a learning rate of 0.01.

Both the test accuracy and training loss in Fig. 2 show that the convergence effect of FedAvg becomes gradually unstable and fluctuates as the data heterogeneity increases. However, FedProx can cope with the problem of statistical heterogeneity. It can be seen from the experiments that FedProx converges better than FedAvg as the dataset with increasing degree of Non-IID. SCAFFOLD and FedNova converge better than FedAvg on the Synthetic_0_0 dataset, but are more oscillating on the Synthetic_0.5_0.5 and on the Synthetic_1_1 dataset. Considering the possible effect of learning rate, we also analyze the experimental results at different learning rates. Obviously, FedAvg and FedNova perform well in the case of IID data. However, in practice, it is difficult to have perfect IID data. By applying our proposed double-momentum mechanism, FedFa can not only improve the convergence speed in the case of IID data, but also speed up the convergence speed for Non-IID data. Since FedFa



(a) Testing Accuracy



(b) Training Loss

Fig. 2. The Training Loss and Test Accuracy of Experiments on Synthetic Datasets with Four Different Data Distributions (From left to right, the dataset has an increasing degree of Non-IID). The learning rate for FedAvg, FedProx, SCAFFOLD, and FedNova is 0.01. For the two synthetic data sets on the left, the momentum factors for the client and server side of FedFa are 0.9 and 0.5, respectively. For the two synthetic data sets on the right, the momentum factors for the client and server side of FedFa are 0.5 and 0.5, respectively.

takes into account the historical gradient information at both server and client sides, this helps the algorithm to converge better.

We also study the effects of different momentum factors on the client and server side. The synthetic dataset with the same distribution (Synthetic_iid) and the synthetic dataset (Synthetic_0_0) use the momentum factors of the client and the server of 0.9 and 0.5. Since for the dataset of weaker heterogeneity distribution, speeding up the convergence of the client does not lead to global gradient deviations. For the synthetic dataset with increasing degree of Non-IID (Synthetic_0.5_0.5 and Synthetic_1_1), we set the momentum factors for the client and server sides to 0.5 and 0.5. We argue that this will reduce the oscillations on the server side when aggregating the gradients during the Non-IID data training process.

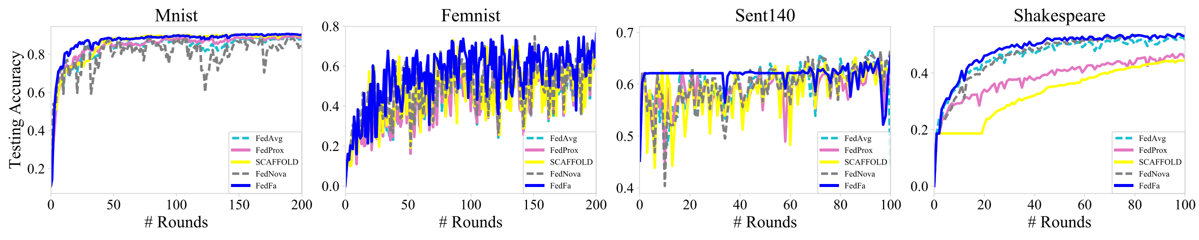
As shown in Fig. 3, in order to validate the double momentum mechanism, we also conduct experiments on four real datasets. The figure shows the test accuracy and training loss of FedAvg, FedProx, SCAFFOLD, FedNova and FedFa on these real datasets. It can be seen that the double momentum mechanism can also be worked in real datasets.Fig. 4,Fig. 5.

For a fair comparison, we explore FedAvg, FedProx, SCAFFOLD and FedNova on synthetic datasets at learning rates of 0.001 and 0.0001. As shown in Fig. 2, the learning rate of edAvg, FedProx SCAFFOLD and FedNova is 0.001. We can see that despite reducing the learning rate of FedAvg and Fedprox, the experimental test accuracy and loss can reduce oscillations during training, but will impair convergence. While SCAFFOLD and FedNova perform better at this learning rate and reduce the oscillations. But our method still works better than their results. Furthermore, our method performs better than these baselines on the synthetic dataset of IID when these algorithms reduce the learning rate.

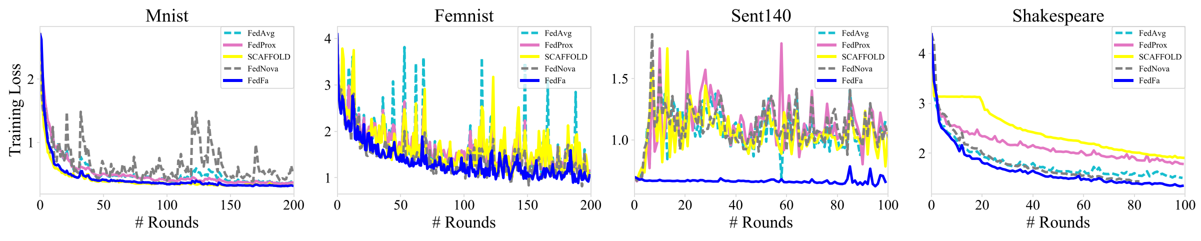
As shown in Fig. 2, it is a comparison of test accuracy and loss of FedFa with FedAvg, FedProx, SCAFFOLD and FedNova at the same learning rate. Obviously, our method is at the advantage of faster convergence. We demonstrate the effectiveness of our proposed double momentum gradient in federated learning through these experiments. The core of the mechanism is to approximate the server-side gradient using the difference between the two rounds of the model on the server-side, which allows the application of the momentum gradient method to the aggregate gradient on the server-side. The client employs the traditional momentum gradient descent method as a means of achieving a double momentum gradient.

Fig. 6 shows the experimental results of FedFa compared to FedAvg, FedProx, SCAFFOLD and FedNova under four datasets that were set using the Dirichlet function to set different levels of Non-IID. FedAvg, FedProx, FedAvg, FedProx, SCAFFOLD and FedNova are the better performers at a learning rate of 0.01. It can be seen from Fig. 6 that our method converges better as the level of Non-IID increases.

Our experiments also show that the momentum gradient method on the server side performs better in heterogeneity problems. The reason is that although the client-side momentum gradient method speeds up its own convergence, it is not friendly to global convergence. Therefore, when the heterogeneity of the data distribution between clients increases, one can consider appropriately decreasing the momentum factor for clients or increasing the round parameter *b* for the aggregation on the server side.

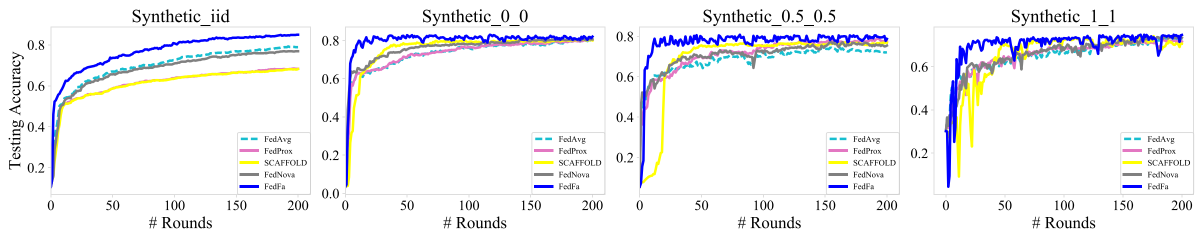


(a) Testing Accuracy

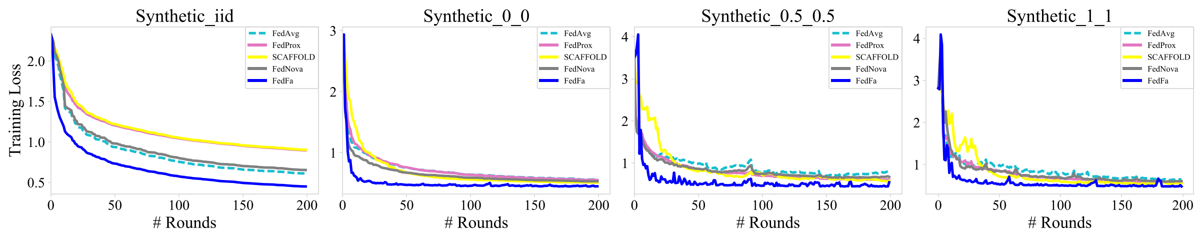


(b) Training Loss

Fig. 3. The Training Loss and Test Accuracy of Experiments on four real Datasets. For the two datasets (Mnist, Shakespeare), the momentum factors for the client and server side of FedFa are 0.5 and 0.5, respectively. For the dataset of Sent140, the momentum factors for the client and server side of FedFa are 0.1 and 0.5, respectively. For the dataset of Femnist, the momentum factors for the client and server side of FedFa are 0.2 and 0.5, respectively.

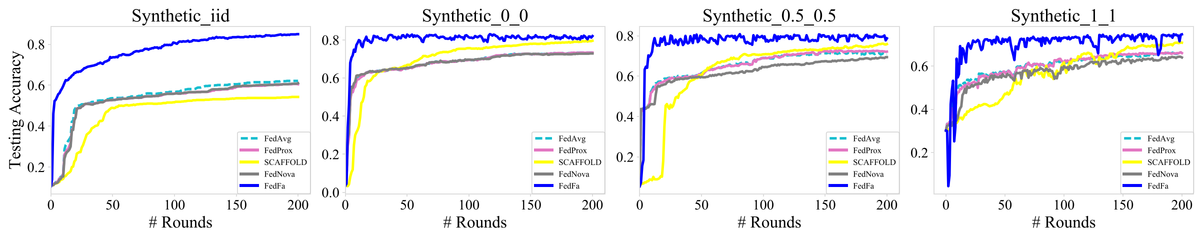


(a) Testing Accuracy

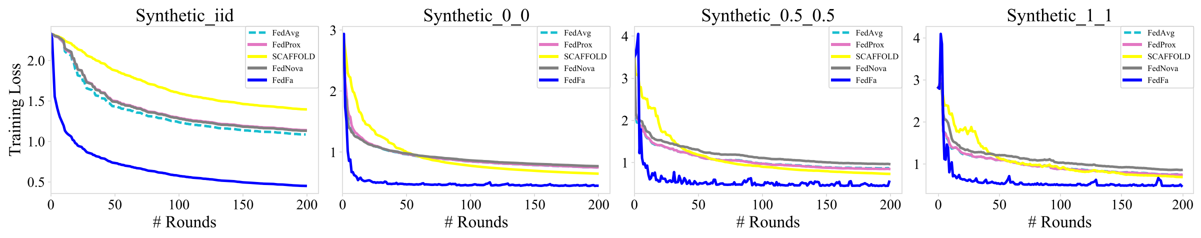


(b) Training Loss

Fig. 4. The Training Loss and Test Accuracy of Experiments on Synthetic Datasets with Four Different Data Distributions (From left to right, the dataset has an increasing degree of Non-IID). The learning rate for FedAvg, FedProx, SCAFFOLD, and FedNova is 0.001. All settings of FedFa are kept fixed.



(a) Testing Accuracy



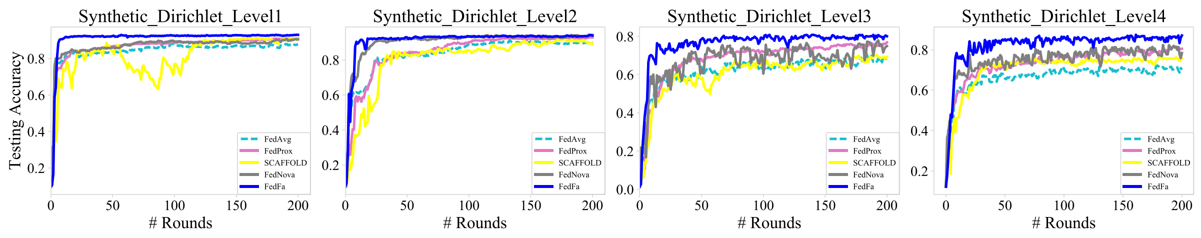
(b) Training Loss

Fig. 5. The Training Loss and Test Accuracy of Experiments on Synthetic Datasets with Four Different Data Distributions (From left to right, the dataset has an increasing degree of Non-IID). The learning rate for FedAvg, FedProx, SCAFFOLD, and FedNova is 0.0001. All settings of FedFa are kept fixed.

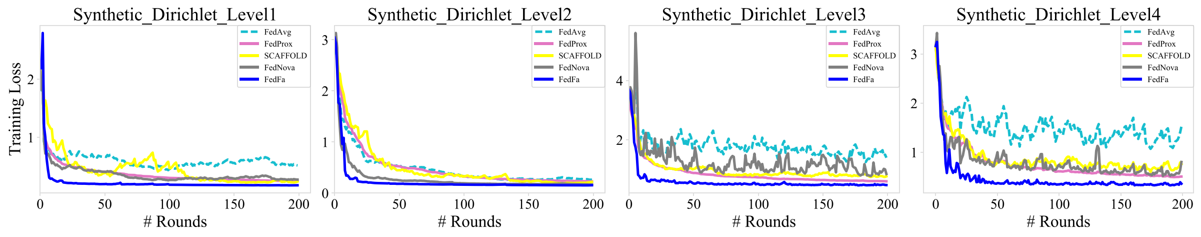
4.3. Effects of appropriate weighting strategies

In our following experiments, we verify that the proposed weighting strategy can lead to a fairer solution for federated datasets. We employ the weighting strategy on the four synthetic datasets, Femnist and Shakespeare. The accuracy weight α and frequency weight β are set to (0.5, 0.5), (0, 1) or (1, 0), respectively.

As shown in the following Table 2, our method is compared with FedAvg, Fedprox and q-Fair. The average accuracy, the accuracy of the worst 20% devices, the accuracy of the best 20% devices, and the variance of the final accuracy distribution are compared, respectively. In the IID synthetic dataset, our method is slightly inferior to FedAvg, but performs better than the FedProx and q-Fair. However, in federated networks, the general setup is in the form of Non-IID data distribution. It can be seen that the average accuracy is improved for all datasets except for the IID synthesis dataset. Moreover, the accuracy of the worst and best 20% devices is also increased. In addition, the variance of the final accuracy distribution is decreased.



(a) Testing Accuracy



(b) Training Loss

Fig. 6. The Training Loss and Test Accuracy of Experiments on Synthetic Datasets with Four Different Dirichlet Distributions (From left to right, the dataset has an increasing degree of Non-IID). The learning rate for FedAvg, FedProx, SCAFFOLD, and FedNova is 0.01. All settings of FedFa are kept fixed.

Table 2
Statistics of the test accuracy distribution for FedAvg, FedProx, q-Fair and FedFa.

Dataset	Method	Average	Worst 20%	Best 20%	Variance
Synthetic_iid	FedAvg	89.11%	78.17%	100.00%	68.04
	FedProx	71.92%	55.46%	86.09%	129.09
	q-Fair	83.22%	60.70%	99.21%	204.49
	FedFa	85.70%	71.46%	100.00%	98.74
Synthetic_0_0	FedAvg	54.67%	6.81%	98.18%	1112.46
	FedProx	75.02%	37.21%	100.00%	611.10
	q-Fair	73.79%	28.29%	100.00%	809.76
	FedFa	**78.25%**	**43.41%**	**100.00%**	**530.27**
Synthetic_0.5_0.5	FedAvg	40.24%	0.00%	98.18%	1448.46
	FedProx	67.84%	33.60%	100.00%	618.00
	q-Fair	71.67%	30.99%	100.00%	732.20
	FedFa	**73.30%**	**41.27%**	**100.00%**	**464.81**
Synthetic_1_1	FedAvg	54.78%	1.38%	96.85%	1069.37
	FedProx	64.75%	9.72%	100.00%	1088.87
	q-Fair	72.78%	25.24%	100.00%	887.05
	FedFa	**76.88%**	**37.03%**	**100.00%**	**603.69**
Femnist	FedAvg	70.96%	34.77%	100.00%	567.75
	FedProx	72.20%	40.50%	100.00%	449.83
	q-Fair	69.86%	21.07%	100.00%	918.78
	FedFa	**77.96%**	**48.99%**	**100.00%**	**368.93**
Sent140	FedAvg	68.51%	40.58%	93.22%	356.48
	FedProx	64.71%	34.02%	92.65%	431.89
	q-Fair	67.54%	29.58%	**100.00%**	609.63
	FedFa	**68.83%**	**42.66%**	95.71%	**319.04**

For instance, on the Synthetic_1_1 dataset, the heterogeneity between devices is relatively large. The model trained with FedAvg has an accuracy of 1.37% on the worst 20% devices. Similarly, on the Synthetic_1_1 dataset, the accuracy of the model trained with FedProx on the worst 20% devices is 9.72%. This can indicate that as the heterogeneity of data distribution among devices increases, the federated learning training process appears skewed or ignored for some clients, i.e., fairness issues. While the model trained on the Synthetic_1_1 dataset using q-Fair has an accuracy of 25.24% on the worst 20% devices significantly larger than FedAvg and FedProx, and the final accuracy variance is also smaller than FedAvg and FedProx. Thus, q-Fair plays a role in the fairness problem of federated learning. The model trained with FedFa, on the other hand, has an accuracy of 37.03% on the worst 20% devices which is 12.03% better than q-Fair, and the variance of the final accuracy distribution is lower than q-Fair. FedAvg and FedProx significantly have lower accuracy on the worst 20% devices than FedFa, but all four algorithms perform well in terms of accuracy on the best 20% devices. The results demonstrate that our approach allows clients in federated learning to participate more fairly in the training process, as well as to cope with the heterogeneity problem. The values of α and β determine the distribution of training accuracy and the number of clients involved in the training process to be taken more into account. It avoids a federated learning setup with preferences that result in high precision for some clients while others do not participate in training or have low precision.

In Fig. 7, we present the distribution of clients in terms of final precision for our method and the comparison method. We observe a significant increase in the number of clients with high precision distribution and a significant decrease in the number of clients with low precision or zero precision. Since our method is more fair, it allows more clients to participate in the training process and improve their accuracy.

Table 3 is a comparison experiment between FedFa-mo and FedFa, which further illustrates the effectiveness of the weighting strategy. FedFa-mo is a method that uses only the double momentum gradient. FedFa uses not only the double momentum gradient but also the weighting strategy. It can be seen that FedFa has better experimental results than FedFa-mo. Although the average accuracy of both is close, the variance of the performance distribution of the client becomes smaller and the performance of the worst 20% devices is improved, so the weighting strategy can make the performance of the devices in federated learning more balanced.

The basic setup of federated learning often leads to a disadvantage for certain clients in the training process. That is, some clients participate less frequently in the training process, or the accuracy of some clients cannot be improved. Therefore, our approach takes into account the effects of client participation frequency and accuracy to achieve a more fair distribution of accuracy among clients.

4.4. Discussion of weighted parameters

In this section, we discuss the turn parameter round b of the momentum gradient method and the parameters α (Accuracy weight) and β (Frequency weihgt) of the weighting strategy.

In order to analyze the effect of the training accuracy weight α and the number of participants weight β on the experiment, we increase or decrease these two parameters in steps of 0.1 on the synthetic dataset (Synthetic_1_1). However, we have to ensure that $\alpha + \beta = 1$. Under this constraint, Fig. 8 shows the effect of parameter values on the experimental results. We can observe the curves of the final accuracy, variance and average accuracy varying with the two parameters. On this dataset, the effect of parameters α and β is relatively symmetrical, i.e., α and β have a relatively close influence of the final test accuracy.

Fig. 9 illustrates the analysis of the effect of parameters α and β on the Mnist dataset. Similarly, we calculate the final accuracy, variance, and average accuracy of the dataset using a step size of 0.1. The figure demonstrates the curves of these metrics varying with the parameters. It is seen that a bias towards one parameter can lead to better experimental results.

The experimental results also demonstrate that the variance between accuracy of the clients is not always small when the average accuracy is large. Therefore, in federated learning, we should focus not only on the average accuracy but also on the variance between accuracy of the clients. In this way, we can make the federated learning framework more fair.

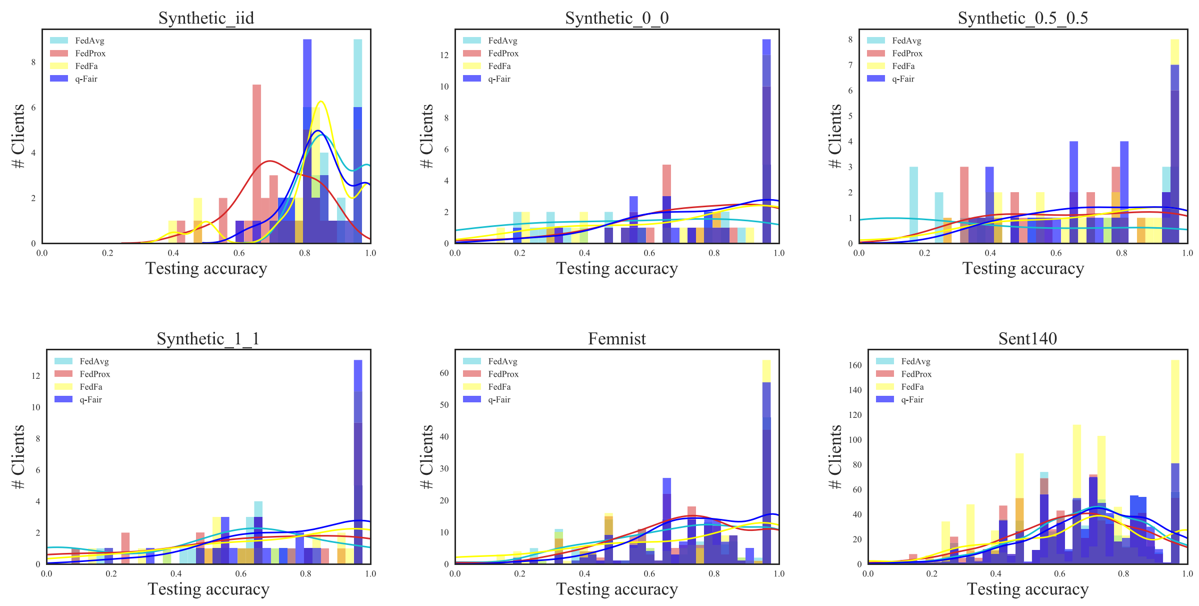
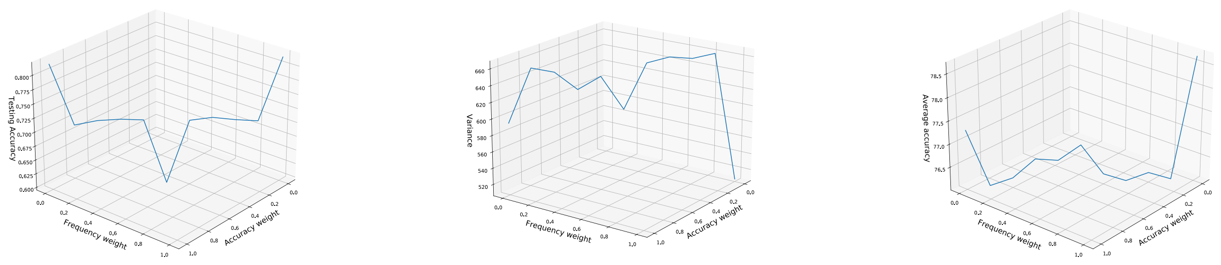


Fig. 7. The distribution of test accuracy for all clients of the four synthetic datasets, Femnsit, and Sent140.

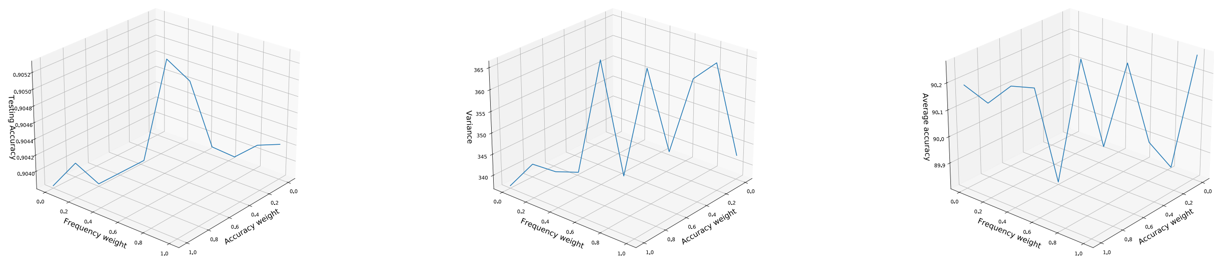
Table 3
A comparison between FedFa without using the weighting strategy (FedFa-mo) and FedFa using the weighting strategy.

Dataset	Method	Average	Worst 20%	Best 20%	Variance
Synthetic_iid	FedFa-mo	80.11%	60.90%	96.05%	151.76
	FedFa	**85.70%**	**71.46%**	**100.00%**	**98.74**
Synthetic_0_0	FedFa-mo	78.06%	24.21%	100.00%	971.18
	FedFa	**78.25%**	**43.41%**	**100.00%**	**530.27**
Synthetic_0.5_0.5	FedFa-mo	73.06%	31.12%	100.00%	740.16
	FedFa	**73.30%**	**41.27%**	**100.00%**	**464.81**
Synthetic_1_1	FedFa-mo	76.44%	27.48%	100.00%	818.90
	FedFa	**76.88%**	**37.03%**	**100.00%**	**603.69**
Femnist	FedFa-mo	74.75%	38.02%	100.00%	536.41
	FedFa	**77.96%**	**48.99%**	**100.00%**	**368.93**
Sent140	FedFa-mo	67.31%	40.98%	92.07%	331.34
	FedFa	**68.83%**	**42.66%**	**95.71%**	**319.04**



(a) Final testing accuracy (b) Variance (c) Average accuracy

Fig. 8. Analysis of the parameters on Synthetic_1_1. The accuracy weight is the value of α and the frequency weight is the value of β. These two parameters determine the final test accuracy, variance and average accuracy.



(a) Final testing accuracy (b) Variance (c) Average accuracy

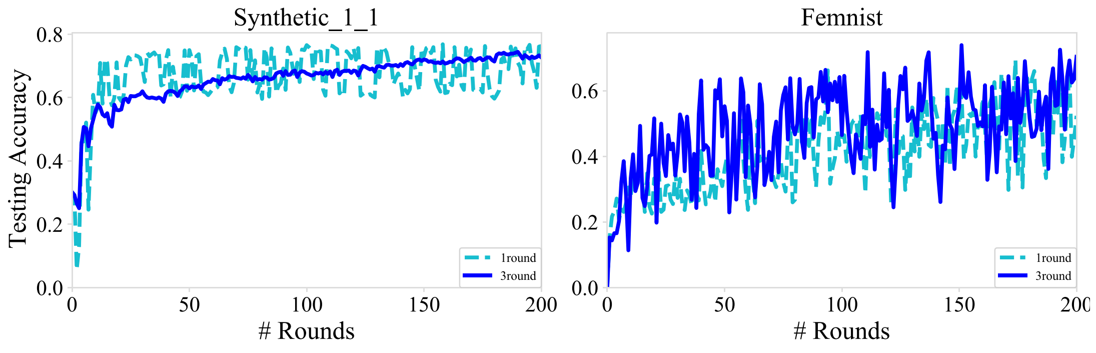
Fig. 9. Analysis of the parameters on Mnist. The accuracy weight is the value of α and the frequency weight is the value of β. These two parameters determine the final test accuracy, variance and average accuracy.

In Fig. 10, we conduct experiments using the double momentum gradient method of FedFa on synthetic_1_1 and Femnist. We compare the results of our experiments using the momentum gradient method after 1 round and 3 rounds of aggregation on the server side. We have verified that the momentum gradient method can be used at each round when the server-side performs the convergence gradient. It can be seen from Fig. 10 that when the number of rounds of the server is taken as 3, it will be more stable than using the momentum gradient method for each round.

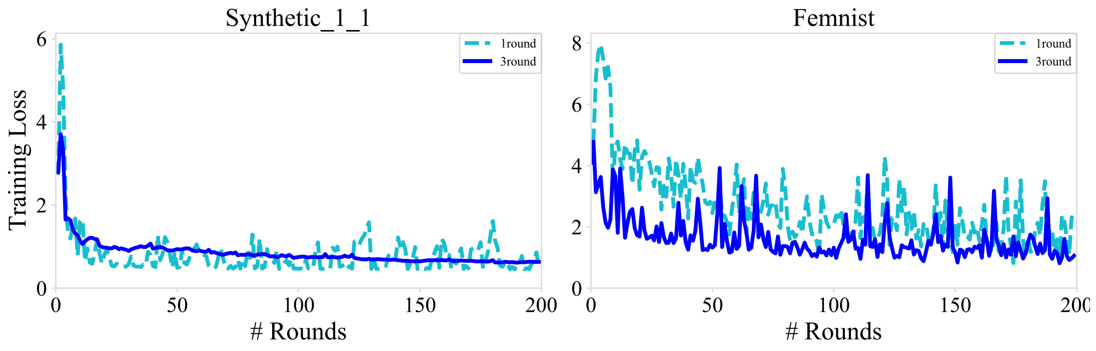
It is also possible to use the momentum gradient method on the server-side after round b. Since the problem of heterogeneity in the data distribution of these clients leads to different gradient optimization directions for each client. Therefore, we can wait for the server to collect the gradients of round b before using the momentum gradient method, which can better consider the influence of historical gradients and reduce the oscillation.

4.5. Discussion of communication overhead

Communication overhead analysis: Both our model and the models of the baseline are based on the fundamental framework FedAvg. In our proposed Algorithm 2, first, after the client is trained locally, each device $k \in S_t$ sends $w_k^{t+1}, Acc_k^{t+1}, f_k^{t+1}$



(a) Testing Accuracy



(b) Training Loss

Fig. 10. Analysis of the effects of rounds in Synthetic_1_1 and Femnist.

back to the server, where Acc_k^{t+1} and f_k^{t+1} are the accuracy and the number of participation rounds which are constants of negligible size, and w_k^{t+1} is the model updated parameter which is the same size as the parameter of the baseline model FedAvg. Furthermore, in the baseline model, Fedprox has the same size as the parameters of the baseline model FedAvg. SCAFFOLD passes not only the parameters of the model but also control variates (with the same dimension of the parameter vector) from the client to the Server in each round, which increases the size of the uploaded model updates (by a factor of 2). FedNova passes normalized gradient and τ_i from client to Server, where τ_i is the number of gradient updates in client i and is also a constant, and normalized gradient is the same size as the parameters of the baseline model FedAvg. In addition, we compare the communication rounds of FedFa and these baseline models in our experiments, and we can see that our communication rounds have an advantage. Thus, combining the cost of each round of communication and the required communication rounds, we can conclude that FedFa has an advantage in terms of communication overhead.

5. Conclusions

In this paper, we proposed a novel horizontal federated optimization algorithm, FedFa, which combines a double momentum gradient and a weighting strategy. The double momentum gradient takes into account the influence of historical gradient information at both client and server sides to improve the convergence of the algorithm. The weighting strategy aggregates the weights according to the training accuracy and the number of participants to create a fairer and more accurate horizontal federated learning algorithm. Through extensive experiments on federated datasets, we validated that our proposed approach significantly improves the accuracy and fairness of horizontal federated learning compared to existing benchmarks. In future research, we will develop new federated learning algorithms that can better adapt to the challenge of heterogeneity among clients, thus further improving learning performance and reducing communication costs. In addition, we will focus on theoretical analysis related to federated learning optimization algorithms, e.g., convergence analysis and convergence speed analysis.

Declaration of Competing Interest

The authors declare that they have no known competing financial interests or personal relationships that could have appeared to influence the work reported in this paper.

Acknowledgments

This research was supported by the National Key R&D Program of China (2019YFB2101802) and the National Natural Science Foundation of China (62176221, 62072106).

References

[1] Martín Abadi, Paul Barham, Jianmin Chen, Zhifeng Chen, Andy Davis, Jeffrey Dean, Matthieu Devin, Sanjay Ghemawat, Geoffrey Irving, Michael Isard, et al. Tensorflow: A system for large-scale machine learning. In Proceedings of the 12th {USENIX} Symposium on Operating Systems Design and Implementation ({OSDI} 16), pages 265–283, 2016..

[2] Sebastian Caldas, Sai Meher Karthik Duddu, Peter Wu, Tian Li, Jakub Konečnỳ H Brendan McMahan, Virginia Smith, and Ameet Talwalkar. Leaf: A benchmark for federated settings. arXiv preprint arXiv:1812.01097, 2018..

[3] D. Chai, L. Wang, K. Chen, and Q. Yang. Secure federated matrix factorization. IEEE Intelligent Systems, pages 1–1, 2020. DOI: 10.1109/MIS.2020.3014880..

[4] Yang Chen, Xiaoyan Sun, and Yaochu Jin. Communication-efficient federated deep learning with layerwise asynchronous model update and temporally weighted aggregation. IEEE Transactions on Neural Networks and Learning Systems, pp(99):1–10, 2019..

[5] G. Cohen, S. Afshar, J. Tapson, A. van Schaik, Emnist: Extending mnist to handwritten letters, in: Proceedings of the 2017 International Joint Conference on Neural Networks, 2017, pp. 2921–2926.

[6] Luca Corinzia and Joachim M Buhmann. Variational federated multi-task learning. arXiv preprint arXiv:1906.06268, 2019..

[7] Chelsea Finn, Pieter Abbeel, and Sergey Levine. Model-agnostic meta-learning for fast adaptation of deep networks. In Proceedings of the 34th International Conference on Machine Learning-Volume 70, pages 1126–1135, 2017..

[8] Hongchang Gao, Xu. An, Heng Huang, On the convergence of communication-efficient local sgd for federated learning, Proceedings of the AAAI Conference on Artificial Intelligence 35 (2021) 7510–7518.

[9] Alec Go, Richa Bhayani, and Lei Huang. Twitter sentiment classification using distant supervision. CS224N project report, Stanford, 1(12):2009, 2009..

[10] Li Huang, Yifeng Yin, Fu. Zeng, Shifa Zhang, Hao Deng, Dianbo Liu, Loadaboost: Loss-based adaboost federated machine learning with reduced computational complexity on iid and non-iid intensive care data, Plos One 15 (4) (2020) e0230706.

[11] Yutao Huang, Lingyang Chu, Zirui Zhou, Lanjun Wang, Jiangchuan Liu, Jian Pei, and Yong Zhang. Personalized cross-silo federated learning on non-iid data. In Proceedings of the AAAI Conference on Artificial Intelligence, volume 35, pages 7865–7873, 2021..

[12] Zhouyuan Huo, Qian Yang, Bin Gu, Lawrence Carin Huang, et al. Faster on-device training using new federated momentum algorithm. arXiv preprint arXiv:2002.02090, 2020..

[13] Yihan Jiang, Jakub Konečnỳ, Keith Rush, and Sreeram Kannan. Improving federated learning personalization via model agnostic meta learning. arXiv preprint arXiv:1909.12488, 2019..

[14] Sai Praneeth Karimireddy, Satyen Kale, Mehryar Mohri, Sashank Reddi, Sebastian Stich, and Ananda Theertha Suresh. Scaffold: Stochastic controlled averaging for federated learning. In International Conference on Machine Learning (ICML), pages 5132–5143, 2020..

[15] Andreas Kotsios, Matteo Magnani, Davide Vega, Luca Rossi, Irina Shklovski, An analysis of the consequences of the general data protection regulation on social network research, ACM Transactions on Social Computing 2 (3) (2019) 1–22.

[16] Y. Lecun, L. Bottou, Y. Bengio, P. Haffner, Gradient-based learning applied to document recognition, Proceedings of the IEEE 86 (11) (1998) 2278–2324.

[17] Junyu Li, Ligang He, Shenyuan Ren, and Rui Mao. Developing a loss prediction-based asynchronous stochastic gradient descent algorithm for distributed training of deep neural networks. In Proceedings of the 49th International Conference on Parallel Processing, pages 1–10, 2020a..

[18] Qinbin Li, Yiqun Diao, Quan Chen, and Bingsheng He. Federated learning on non-iid data silos: An experimental study. arXiv preprint arXiv:2102.02079, 2021a..

[19] Qinbin Li, Bingsheng He, and Dawn Song. Practical one-shot federated learning for cross-silo setting. In Proceedings of the Thirtieth International Joint Conference on Artificial Intelligence (IJCAI), pages 1484–1490, 2021b..

[20] Tian Li, Maziar Sanjabi, Ahmad Beirami, and Virginia Smith. Fair resource allocation in federated learning. arXiv preprint arXiv:1905.10497, 2019..

[21] Tian Li, Anit Kumar Sahu, Manzil Zaheer, Maziar Sanjabi, Ameet Talwalkar, and Virginia Smith. Federated optimization in heterogeneous networks. Proceedings of Machine Learning and Systems, 2:429–450, 2020b..

[22] Tian Li, Shengyuan Hu, Ahmad Beirami, and Virginia Smith. Ditto: Fair and robust federated learning through personalization. In Proceedings of the Thirtieth International Joint Conference on Artificial Intelligence, pages 6357–6368. PMLR, 2021c..

[23] Wei Liu, Li Chen, Yunfei Chen, Wenyi Zhang, Accelerating federated learning via momentum gradient descent, IEEE Transactions on Parallel and Distributed Systems 31 (8) (2020) 1754–1766.

[24] Brendan McMahan, Eider Moore, Daniel Ramage, Seth Hampson, and Blaise Aguera y Arcas. Communication-efficient learning of deep networks from decentralized data. In Artificial Intelligence and Statistics, pages 1273–1282, 2017..

[25] Mehryar Mohri, Gary Sivek, and Ananda Theertha Suresh. Agnostic federated learning. arXiv preprint arXiv:1902.00146, 2019..

[26] Takayuki Nishio, Ryo Yonetani, Client selection for federated learning with heterogeneous resources in mobile edge, in: Proceedings of the 2019 IEEE International Conference on Communications, IEEE, 2019, pp. 1–7.

[27] Jeffrey Pennington, Richard Socher, Christopher Manning, Glove: Global vectors for word representation, in: Conference on Empirical Methods in Natural Language Processing, 2014, pp. 1532–1543.

[28] Neta Shoham, Tomer Avidor, Aviv Keren, Nadav Israel, Daniel Benditkis, Liron Mor-Yosef, and Itai Zeitak. Overcoming forgetting in federated learning on non-iid data. arXiv preprint arXiv:1910.07796, 2019..

[29] Virginia Smith, Chao-Kai Chiang, Maziar Sanjabi, Ameet S Talwalkar, Federated multi-task learning, Advances in Neural Information Processing Systems 30 (2017) 4424–4434.

[30] Hongyi Wang, HMikhail Yurochkin, Yuekai Sun, Dimitris Papailiopoulos, and Yasaman Khazaeni. Federated learning with matched averaging. In International Conference on Learning Representations, pages 1–10, 2020a..

[31] Hongyi Wang, Mikhail Yurochkin, Yuekai Sun, Dimitris Papailiopoulos, Yasaman Khazaeni, Federated learning with matched averaging, in: International Conference on Learning Representations (ICLR), 2020.

[32] Jianyu Wang, Qinghua Liu, Hao Liang, Gauri Joshi, and H Vincent Poor. Tackling the objective inconsistency problem in heterogeneous federated optimization. arXiv preprint arXiv:2007.07481, 2020c..

[33] Ming Xie, Guodong Long, Tao Shen, Tianyi Zhou, Xianzhi Wang, and Jing Jiang. Multi-center federated learning. arXiv preprint arXiv:2005.01026, 2020..

[34] Jing Xu, Sen Wang, Liwei Wang, and Andrew Chi-Chih Yao. Fedcm: Federated learning with client-level momentum. arXiv preprint arXiv:2106.10874, 2021..

[35] Qiang Yang, Yang Liu, Tianjian Chen, Yongxin Tong, Federated machine learning: Concept and applications, ACM Transactions on Intelligent Systems and Technology 10 (2) (2019) 1–19.
[36] Qiang Yang, Yang Liu, Yong Cheng, Yan Kang, Tianjian Chen, Yu. Han, Federated learning, Synthesis Lectures on Artificial Intelligence and Machine Learning 13 (3) (2019) 1–207.
[37] Xin Yao, Chaofeng Huang, Lifeng Sun, Two-stream federated learning: Reduce the communication costs, in: Proceedings of the 2018 IEEE Visual Communications and Image Processing, IEEE, 2018, pp. 1–4.
[38] Xin Yao, Tianchi Huang, Chenglei Wu, Rui-Xiao Zhang, and Lifeng Sun. Federated learning with additional mechanisms on clients to reduce communication costs. arXiv preprint arXiv:1908.05891, 2019a..
[39] Xin Yao, Tianchi Huang, Chenglei Wu, Ruixiao Zhang, and Lifeng Sun. Towards faster and better federated learning: A feature fusion approach. In Proceedings of the 2019 IEEE International Conference on Image Processing, pages 175–179. IEEE, 2019b..
[40] Austine Zong Han Yapp, Hong Soo Nicholas Koh, Yan Ting Lai, Jiawen Kang, Xuandi Li, Jer Shyuan Ng, Hongchao Jiang, Wei Yang Bryan Lim, Zehui Xiong, and Dusit Niyato. Communication-efficient and scalable decentralized federated edge learning. In Proceedings of the Thirtieth International Joint Conference on Artificial Intelligence (IJCAI), pages 5032–5035, 2021..
[41] Yue Zhao, Meng Li, Liangzhen Lai, Naveen Suda, Damon Civin, and Vikas Chandra. Federated learning with non-iid data. arXiv preprint arXiv:1806.00582, 2018..
[42] Yuanshao Zhu, Shuyu Zhang, Yi Liu, Dusit Niyato, J.Q. James, Robust federated learning approach for travel mode identification from non-iid gps trajectories, in: 2020 IEEE 26th International Conference on Parallel and Distributed Systems (ICPADS), IEEE, 2020, pp. 585–592.