# N-FedAvg: Novel Federated Average Algorithm Based on FedAvg

Shuaikun Xing
*Faculty of Information Technology*
*Beijing University of Technology*
Beijing, China
xingshuaikun@163.com

Zhenhu Ning
*Faculty of Information Technology*
*Beijing University of Technology*
Beijing, China
ningzhenhu@bjut.edu.cn

Jin Zhou
*Faculty of Information Technology*
*Beijing University of Technology*
Beijing, China
broinniub@gmail.com

Xue Liao
*Faculty of Information Technology*
*Beijing University of Technology*
Beijing, China
1506824775@qq.com

Jiawei Xu
*Faculty of Information Technology*
*Beijing University of Technology*
Beijing, China
1045243191@qq.com

Wei Zou
*Faculty of Information Technology*
*Beijing University of Technology*
Beijing, China
18821705441@163.com

2022 14th International Conference on Communication Software and Networks (ICCSN) | 978-1-6654-5328-8/22/$31.00 ©2022 IEEE | DOI: 10.1109/ICCSN55126.2022.9817607

***Abstract*—As a safe and efficient distributed machine learning technology, federated learning has been widely used in all walks of life. However, various problems in the traditional FedAvg algorithm have brought a lot of inconvenience to the implementation of federated learning in related industries. Therefore, this paper addresses related problems by proposing a novel federated learning algorithm (N-FedAvg) based on FedAvg. N-FedAvg selects clients in sequence before each epoch, reducing the randomness when selecting clients, so that all clients' data can have the opportunity to participate in federated learning, and prevent local data of some clients from participating in the federation with a low probability. Also, when using gradient descent to optimize the objective function and approaching the global minimum of the loss function value, the learning rate should be made smaller to get the model as close as possible to this point, and cosine annealing can be achieved by the cosine function and produces good results. Finally, the sparsity strategy is essentially a model compression method, which not only transmits a small number of parameters, but also reduces the network bandwidth between the server and the clients, and can also prevent the leakage of global model parameters. Through comparative experimental analysis, this paper finds that the N-FedAvg algorithm proposed by us is 1.34% more accurate than the traditional FedAvg algorithm on the CIFAR-10 dataset, and the loss function value is 2.77% lower.**

***Keywords*—Federated Learning, Client Selection, Dynamic Learning Rate, Model Compression***

## I. INTRODUCTION

With the advent of the 5G network era, various smart devices such as smartphones, computers, sensor nodes, and autonomous vehicles are generating massive amounts of data all the time [1]. According to reports, there are currently about 3 billion smart devices [2] and 7 billion IoT devices [3] working continuously around the world, and the data formats, quality, dimensions, etc. generated by them are significantly different, and the geographical location of storage is different [4]. Therefore, distributed machine learning can process off-site data through intelligent and efficient optimization algorithms, and mine valuable information from it [5, 6]. However, in recent years, data laws and regulations aimed at protecting relevant users have been continuously improved, such as the EU's General Data Protection Regulation (GDPR) [7], the People's Republic of China Cybersecurity Law [8] and Personal Information Protection Law [9], etc. These laws have already been implemented. They pose a big challenge for distributed machine learning technologies, as they require service providers not to pass collected user data to any other entity. As a result, federated learning (FL) emerged as a safe and efficient distributed machine learning technology [10].

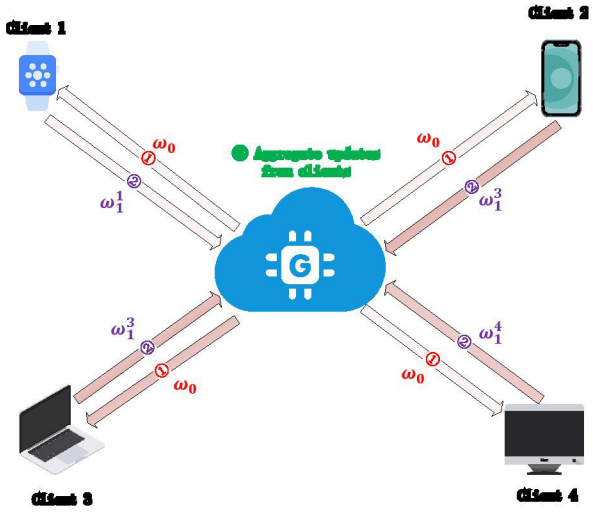


Fig. 1. Federated Learning Architecture.

FL was first proposed by McMahan et al. in 2016 [11]. It mainly learns relevant data on the client through a centralized or distributed server and obtains a global statistical model $f(\omega)$. The general process is shown in Figure 1. First, the server sends the initialized global model to the selected client. Then, the client calculates the local privacy data and global model by using

the corresponding FL algorithm, obtains the local model parameters of this round, and sends them to the server. Next, the server performs a weighted average of all the model parameters sent by the client to obtain the global model of the next round, and repeats the above process in a loop until the objective function $F(\omega)$ reaches the minimum value, as shown in formula (1).

$$\min_{\omega} F(\omega) = \min_{\omega} \sum_{i=1}^{N} p_i F_i(\omega) \tag{1}$$

Where $F_i(\omega) = \frac{1}{n_i} \sum_{j_i=1}^{n_i} f_{j_i}(\omega; x_{j_i}, y_{j_i})$ is the objective function characterizing the empirical risk of local data, $N$ is the number of all participating clients, $n_i$ is the number of data points available at the client $i$, $p_i = \frac{n_i}{n}$ is the relative impact of each client, and $n = \sum_k n_k$ is the number of all data samples.

At present, there are many mainstream FL platforms. They have slightly different implementation mechanisms and programming languages, but basically all of them implement related technologies, e.g., homomorphic encryption, privacy protection, authentication, access control and so on. For example, FATE[12], FedML[16], Pysyft[13], TensorFlow Federated[14], PaddlePaddle[15], etc. These platforms have their own characteristics. The promotion of the above platforms has enabled FL to be well applied in the fields of finance [17], medical care [18], industrial Internet [19], and computer vision [20], and promoted the implementation of related technologies.

However, FL currently faces many problems, e.g., Non-IID [21, 22], client heterogeneity [23], communication [24, 25], storage and computing differences [26], etc. These brings a big problem to FL. In addition, the security problems of FL cannot be ignored, and these problems may lead to leakage of user privacy data [27, 28], Byzantine attacks [29], backdoor attacks [30], etc. They maybe lead to many various problems. Under the influence of COVID-19 and the changing world situation, the number of cyberspace security attacks is increasing day by day, and the attack behaviors are also very different. Therefore, the security issue of FL should arouse the attention of relevant researchers.

The N-FedAvg algorithm proposed in this paper is mainly dedicated to solving the problems of random client selection, static learning rate (LR) and model compression. The introduction is as follows.

The original distributed machine learning system assumed full participation of all clients, i.e., all clients participated in each round of training. In existing FL systems, the number of clients participating in each update epoch is usually fixed. Because it is usually limited by factors such as client status and network conditions, FL only randomly selects a small number of clients in each epoch of training. However, due to the large number of heterogeneous clients in a FL setting, this random selection can exacerbate the adverse effects of data heterogeneity [31-35]. These issues delay the aggregation steps required for subsequent central servers to continue the training process. Sequential selection can reduce the randomness when selecting clients, so that the data of all clients can have the opportunity to participate in the FL, and prevent the local data of some clients from participating in the FL with a low probability.

When using the gradient descent algorithm (SGD) to optimize the objective function and approaching the global minimum of the loss function, LR should become smaller to make the model as close as possible to this point, so a reasonable attenuation of LR is required. Since the weights of the model are randomly initialized at the beginning of training, if a larger LR is selected at this time, the model may oscillate [36]. And cosine annealing can reasonably change LR through the cosine function [37]. As the independent variable increases, the value of the cosine function first decreases slowly, then decreases rapidly, and then decreases slowly again. The combination of this decreasing mode and LR is beneficial to speed up the convergence speed of the model and obtain a better model.

Research has shown that most deep neural network models have the problem of redundant weight parameters in the process of model training, i.e., among all parameters, there are often only a small part of them that play an important role in model performance (such as 5%) [38]. In deep learning, we often use model compression methods such as pruning [39], quantization [40], distillation [41] to compress models to achieve a balance between model performance and model complexity. Similar problems exist in the training process of FL. Therefore, we can use the idea of model compression to transmit part of the parameter data in the process of transmission. On the one hand, with the reduction of the amount of transmitted data, it can effectively reduce the bandwidth consumption of network transmission. On the other hand, it can prevent the model parameters from being stolen. Even if the attacker obtains some parameter data, it is difficult to use the model inversion attack to invert the original data, thereby effectively improving the security of the system [42].

We perform related experiments on the N-FedAvg algorithm on Google Colab, which provides free GPU computing resources and program execution environment. Since we use the pre-trained model and data augmentation method in torchvision, the final model converges quickly, so our parameter settings are slightly different from [43]. Through the analysis of the experimental results, it is found that the N-FedAvg algorithm proposed in this paper is 1.34% more accurate than the traditional FedAvg algorithm on the CIFAR-10 dataset, and the loss function value is 2.77% lower. To the best of our knowledge, our proposed N-FedAvg is the first algorithm to comprehensively apply client selection, dynamic LR and model compression in FedAvg algorithm.

In this research work, we make the following contributions:

- In order to solve the problem that the local data of some clients participates in FL with a low probability, we change the random selection of clients in traditional FL to sequential selection of clients.
- The static LR cannot be reasonably changed with the change of the global model, we introduce cosine annealing LR to obtain a better model as much as possible.

- There is a redundancy problem of weight parameters in the model training process, and our sparsity strategy can not only compress the model to reduce the communication pressure between the server and the client, but also resist the model inversion attack.
- We propose the N-FedAvg algorithm based on FedAvg, which alleviates the above three problems to a certain extent.
- All experiments in this paper demonstrate the efficiency, speed and security of our N-FedAvg algorithm.

The subsequent sections of this paper are arranged as follows: Section II mainly introduces the related work in the research process of the paper. Section III describes the theoretical basis and detailed implementation of N-FedAvg algorithm. A series of experiments are carried out by using N-FedAvg algorithm in Section IV, and then the experimental results are evaluated and analyzed. Section V summarizes the full text.

## II. Related Work

Since McMahan et al. proposed the FedAvg algorithm in [43], the research on FL in all walks of life has been uninterrupted. In the N-FedAvg algorithm in this paper, we mainly focus on client selection, dynamic LR and model compression.

### A. Client Selection

In terms of client selection, the common FL algorithms FedAvg [43], FedProx [44], FedAsync [45], etc., randomly select some clients before each round of training. Because there will be a large number of heterogeneous clients in an actual FL environment, the way of randomly selecting clients will exacerbate the adverse effects of data heterogeneity. The heterogeneity in the FL environment mainly includes:

(1) Each client device has heterogeneity in terms of storage, computing and communication capabilities [26].

(2) The problem of data heterogeneity caused by the non-independently and identically distributed (Non-IID) of local data in each client device [21, 22].

(3) The model heterogeneity required by each client device according to its application scenario. These heterogeneities in client devices often affect the training efficiency of the global model, and clients may not be able to receive FL training or testing at the same time [46]. For example, when some clients have limited computing resources, they take longer time to update the model. In addition, if the wireless channel condition of the client is not good, the update time will also be prolonged [47]. All of these issues delay the aggregation steps required for subsequent central servers to continue the training process.

Chai et al. proposed a hierarchical-based FL system [31], the core of which is to adaptively select clients with similar training time to participate in the training during each round of training. When the accuracy of the model is not affected, they can alleviate the problem of client data heterogeneity. Ribero et al. argued that only identifying and transmitting client-side updates that are considered to be informative in each round of training reduces the transmission pressure [32], but loses some valuable data. Similar to [32], Lai et al. in [33] propose a participant selection framework (Oort) that can identify and pick valuable clients for training and testing. In [34], Wang et al. proposed an experience-driven FL framework (Favor) based on reinforcement learning, which intelligently selects client devices participating in each round of FL to offset the bias introduced by Non-IID data, and speed up the convergence.

In the above-mentioned articles we introduced, many authors have analyzed and verified the problems of increasing communication time, decreasing model accuracy, and increasing the value of loss function caused by randomly selecting clients for FL. We can see that it is very important to selectively determine the clients participating in each round of training in practical application scenarios to improve the performance of FL.

### B. Dynamic Learning Rate

The SGD requires us to specify a LR as a control factor for the weight update step. Common LR s are 0.01, 0.001, and 0.0001, etc. The larger the LR, the faster the weight update. Generally speaking, we hope that the LR is larger in the early stage of training, so that the network converges quickly, and the LR is smaller in the later stage of training, so that the network can better converge to the optimal solution. Four commonly used strategies decay types are shown in Figure 2: Linear, Concave, Convex, Cosine annealing [37].

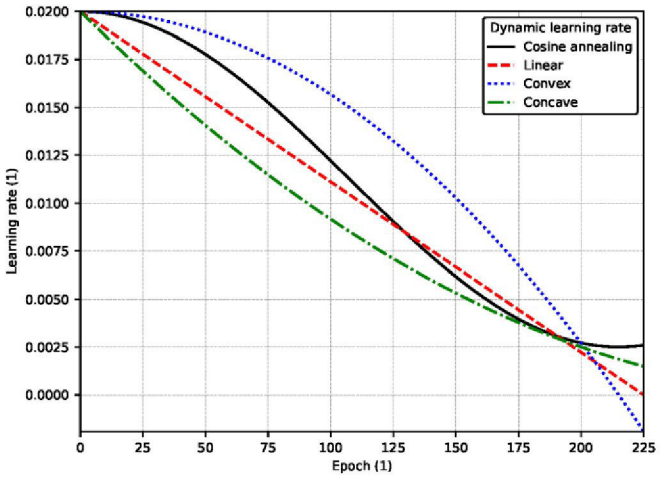


Fig. 2. Curves of four strategies for dynamically adjusting the learning rate.

In this paper, cosine annealing is used to dynamically adjust the LR during training. The characteristic of the cosine function is that with the increase of the independent variable $x$, the value of the cosine function first decreases slowly, then accelerates and decreases, and then decelerates, so the cosine function is often used to reduce the LR, which is called cosine annealing. The attenuation of the LR is performed according to the following formula, and the principle is shown in formula (2).

$$\eta_t = \eta_{\min} + \frac{1}{2}(\eta_{\max} - \eta_{\min})(1 + \cos(\frac{T_{cur}}{T}\pi)) \quad (2)$$

Meaning of characters in expressions:

(1) $\eta_{\max}$ and $\eta_{\min}$ represent the maximum and minimum values of the LR respectively, and define the range of the LR. The paper [11] mentions that the values of $\eta_{\max}$ and $\eta_{\min}$ are reduced after each restart, but for simplicity, this paper keeps $\eta_{\max}$ and $\eta_{\min}$ unchanged after each restart.

(2) $T_{cur}$ indicates how many iterations are currently executed, but $T_{cur}$ is updated after each batch is run, and one iteration has not been executed at this time, so the value of $T_{cur}$ can be a decimal. For example, the total number of samples is 100, and the size of each batch is 20. Then in one iteration, the batch will be read 5 times. Then after the first batch is executed in the first iteration, the value of $T_{cur}$ is updated to 20/100=0.2, and so on.

(3) $T$ represents the total number of iterations in the running process.

As the independent variable increases, the value of the cosine function first decreases slowly, then decreases rapidly, and then decreases slowly again. This decreasing mode can be combined with the LR to produce good results in a very computationally efficient way [10].

### C. Model Compression

The sparsity strategy is essentially a kind of model compression. It also reduces the network bandwidth between the server and the client by transmitting a small number of parameters, and on the other hand, it can also prevent the leakage of global model parameters [42]. The principle is as follows:

Assuming that it is currently in the $t-th$ iteration, the model structure of the current client $C_i$ for local iterative training is $G_t=[g_1,g_2,...,g_L]$, where $g_i$ represents the model parameters of the $i-th$ layer of $G_t$, and the client $C_i$ performs local iterative training, and the model will change from $G_t$ to $L_i^{t+1}$. In traditional FL, the client $C_i$ will upload the model parameters $(L_i^{t+1}-G_t)$ to the server.

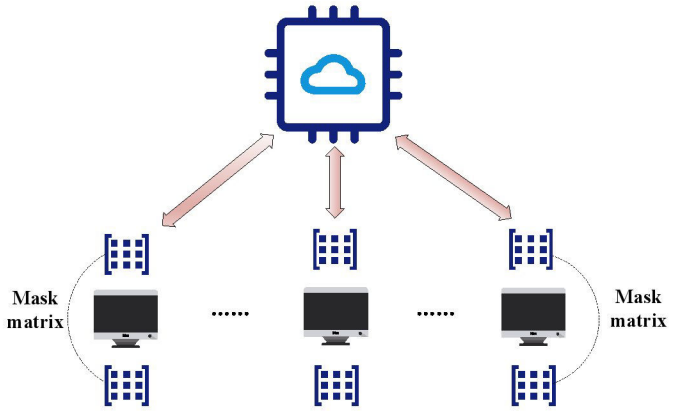


Fig. 3. Schematic diagram of parameter sparsity.

As shown in Figure 3, sparsity means that in the process of local federated training, each client saves a mask matrix $R=[r_1,r_2,...,r_L]$, $r_i$ is a parameter matrix with the same shape and size as $g_i$, and only consists of 0 and 1. After the local training of the client ends, the result of the product of the model parameter $(L_{t+1}-G_t)$ and the mask matrix will be uploaded to the server, as shown in formula (3).

$$\omega_i^{t+1}=(L_i^{t+1}-G_t)\times R_i \tag{3}$$

To sum up, the unreasonable selection of the client will lead to problems such as the increase of communication time, the reduction of model accuracy and the increase of the loss function value. The static LR may cause the model to oscillate, and the direct transmission of the client's parameter model is not conducive to the improvement of communication efficiency. And it is possible to use model inversion attacks to reverse the original data. Therefore, we need to improve the traditional FedAvg algorithm and finally form the N-FedAvg algorithm to solve the above problems.

## III. N-FedAvg Algorithm

This section mainly includes two parts. First, we give a general overview of the N-FedAvg algorithm, so that readers can generally grasp the overall framework of the algorithm. We then provide a detailed introduction and formal description of the workflow for training the algorithm.

### A. Algorithmic Framework

The goal of the N-FedAvg algorithm is to obtain the smallest objective function, and the overall process is shown in Figure 4. First, the server uses the pre-trained model as the first-round global model $\omega_0$ and delivers it to the sequentially selected client set $S_t$. Then, the sequentially selected client $C_i,(i=1,2,...,N)$ calculates the local model $\omega_1^i$ of this round through the global model $\omega_0$ and local privacy data $D_i$ of this round, and sends it to the server. Next, the server aggregates the local model $\omega_1^i$ sent back by the client $C_i$ to obtain the next round of global model $\omega_1$, and finally repeats the above training process until the cross-entropy loss function reaches the minimum value.

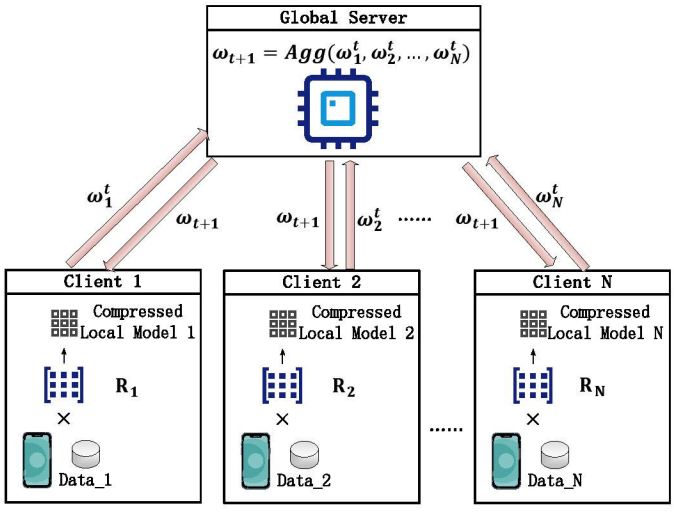


Fig. 4. N-FedAvg algorithm framework.

Before introducing the specific operation flow details of the actual training algorithm in Section 3.2, we first define the meaning of the following symbols.

Note: $C=\{C_1,...,C_N\}$ represents the $N$ clients participating in FL, where $C_i$ represents the $i-th$ client in $C$. During each round of epoch training, $S_t \subset C$ is the set of clients sequentially selected by the server. In addition, $\omega_i$ represents the $i-th$ global model on the server, and $\omega_i^t$ is the local model parameters obtained by the client $C_t$ training its local data $D_i$ in the $i-th$ round. In addition, $GE$ is the number of global epochs, i.e., the number of communication rounds, $LE$ is the number of local iterations, $\eta$ is the static LR, $B$ is the number of samples in each round of local training, and $K$ is the number of clients sequentially selected by the server from the client set $C$ in each round of iteration, and $P$ represents the proportion of the number of 1 in the mask matrix $R$.

*B. Training Algorithm*

Recently, many FL algorithms are optimized from FedAvg, but the optimization starting point of different algorithms is different, involving the reasonable selection of the client [31-34], the aggregation of heterogeneous models on the server [46, 47], The processing of client-side Non-IID data [21, 22], as well as the research on the global model of asynchronous computing [45] and so on.

The main function of the FL server is to perform model aggregation on the local models uploaded by the selected clients. However, it should be noted that, in fact, for a fully functional FL framework, such as the FATE platform of WeBank, the functions of the server are much more complicated. For example, the server needs to monitor the network of each client node and send a reconnection signal to the failed node [12]. Since this section is simulated locally and does not involve the details of network communication and failures, it does not discuss the details of these functions, but only involves the model aggregation function.

In the N-FedAvg algorithm, the client applies SGD when optimizing the global model parameters through local private data. The cross entropy function is a convex function with continuous differentiability. Here we use it as the loss function, the definition is shown in formula (4).

$$\ell(\omega)=-\frac{1}{N}\sum_{i=1}^{N}\sum_{j=1}^{M}(y_{i,j}\log(p_{i,j})) \tag{4}$$

Meaning of characters in expressions:

Where $\omega$ represents the model parameters and $M$ represents the number of data label types. $y_{i,j}$ is whether category $i$ and category $j$ are the same, the same is 1, and the difference is 0. And $p_{i,j}$ represents the predicted probability that the observed sample $i$ belongs to the category $j$.

The N-FedAvg algorithm in this paper has some improvements to the traditional FedAvg algorithm, and the main process is shown in Algorithm 1.

**Algorithm 1**: N-FedAvg

**Data**: The global model of pre-training is $\omega_0$. $S_t$ is a collection of clients selected in order and indexed by $k$. $B$ is the number of samples in each round of local training. $R_i$ is the mask matrix saved by the client $C_i$. $GE$ and $LE$ represent global and local epochs, respectively. $\eta$ is the learning rate.

**Function** *MODELAGGREGATION*

$\omega_0$=(Initialize global model)

**While** $t<GE$ **do**

$S_t$=(Sequentially selected clients)

**While** $k \in S_t$ **do**

$\omega_{t+1}^k = LOCALTRAIN(k,\omega_t)$

**end While**

$\omega_{t+1}=\sum_{k=1}^{K}\frac{n_k}{n}\times\omega_{t+1}^k$

**end While**

**Return** $\omega_{GE}$

**end Function**

**Function** *LOCALTRAIN*

$\Theta$=(split data $D_i$ into batched of size $B$)

**While** $i<LE$ **do**

$\eta$=(Cosine annealing value of current epoch)

**While** $b \in \Theta$ **do**

$\omega=\omega-\eta\times\ell(\omega,b)$

**end While**

**end While**

**Return** $R_i\times\omega$

**end Function**

For the server, first, the parameters of the pre-trained model torchvision.models.resnet50 utilized by the server are used as the global model $\omega_0$ for the first round. Then, in the outermost while loop, the number of communications $t$ between the client and the server changes from 0 to $GE$, and the server sequentially selects the clients in the FL in each round, and puts the selected client in the set $S_t$. Next, the program enters the inner while loop, and performs client local training for each client in the set $S_t$. After the inner while loop is executed, the server aggregates the updated $\omega_i^t$ obtained in the set $S_t$.

For the client, first, the client $C_i$ generates a 0-1 mask matrix locally using Bernoulli distribution, divides the local data $D_i$

with batch size $B$ and stores it in the set $S_B$. Then, in the outer while loop, the client's local iteration number $i$ changes from 0 to $LE$, and then the client calculates the cosine annealing value $\cos\_\eta$ of the LR for the current iteration number. Next, the program enters the inner while loop and performs gradient descent for each data batch $b, b \in S_B$ belonging to the specified client $C_i$. After both layers of while loops are finished, the client $C_i$ sends the compressed model parameters $(\omega_{i+1}^t - \omega_i^t) \times R_i$ to the server.

## IV. Experiment

We have now implemented the N-FedAvg algorithm using Pytorch, and conducted multiple simulation experiments to obtain the average value of the N-FedAvg data under given experimental conditions.

### A. Dataset

The CIFAR-10 dataset has a total of 60,000 color images, which are 32×32×3 pixels in size, and contain a total of 10 classes with 6,000 images per class. Among them, 50,000 images are used for training, forming 5 training batches of 10,000 images each. The other 10,000 images are used for testing, forming a separate batch. The data of the test batch comes from each of the 10 categories, and 1000 images are randomly selected from each category. The rest of the graphs are randomly arranged to form the training batch. Please note that the number of images of each type in a training batch is not necessarily the same. In total, each class of the training batch has 5000 images [48].

### B. Experimental Settings

Pytorch has many convenient and easy-to-use modules. The torchvision module includes three packaged packages: torchvison.datasets, torchvision.models, torchvision.transforms. We obtain the predefined datasets CIFAR-10 by calling the torchvision.datasets package, obtain the pre-trained model Resnet50 [49] through torchvision.models, and use torchvision.transforms to call the predefined data augmentation methods. These methods can be called directly to simplify our modeling process, and can also be used as a reference for us to learn or build new models.

The simulation experiments are carried out on the free GPU server and development environment provided by Google Colab. There are many advantages to use this method: no need to configure any development environment, free use of GPU resources, and easy data sharing. First, we upload the local code to Google Drive, then create a new Jupiter file main.ipynb in Colab and change the device type to GPU, then write code in main.ipynb to mount Google Drive, and run the python code.

In our experiments, 20 user devices participating in FL are simulated, and each device is simulated by a thread and has exclusive GPU computing resources. The server selects 2 clients in each epoch, which means that there will be 2 threads per epoch to train on local private data. We evaluate the performance of the N-FedAvg algorithm in terms of global model accuracy and loss function value on the constructed CIFAR-10 dataset.

We compared N-FedAvg with Centralized, FedAvg. Among them, Centralized is a machine learning method in which the server centrally stores and trains the data of all clients. In this experiment, the number of clients participating in FL can be set to $K=1$, so that the effect of Centralized training can be simulated. FedAvg is a classic FL algorithm. During each epoch, a randomly selected client obtains the global model from the server, then uses local private data for training and obtains local model parameters, then uploads it to the server, and finally performs next epoch.

In Algorithm 1, we set the following hyperparameters: $GE=215$, $LE=5$, $K=20$, $B=50$, $\eta=0.01$, $P=0.6$. Also, the same settings were used for the method of comparative analysis. Additionally, we evaluate the performance of the corresponding algorithm using the classification accuracy and loss function values on the test set data for each client.

### C. Experimental Results

First, we use the FedAvg algorithm as the baseline to conduct experiments on the complete training set of CIFAR-10, and find that the global model accuracy and loss function values tend to be stable after 140 epochs. The N-FedAvg algorithm adopts sequential client selection, dynamic LR and sparsity strategy. After 81 epochs, the experimental parameter values tend to be stable, and the global model accuracy is 1.34% higher than FedAvg. The loss function value is relatively 2.77% lower. Finally, due to the centralized training of the data, Centralized tends to stabilize around 39 epochs. Figures 5 and 6 show the relationship between the number of epochs and the accuracy of the global model and the value of the loss function.

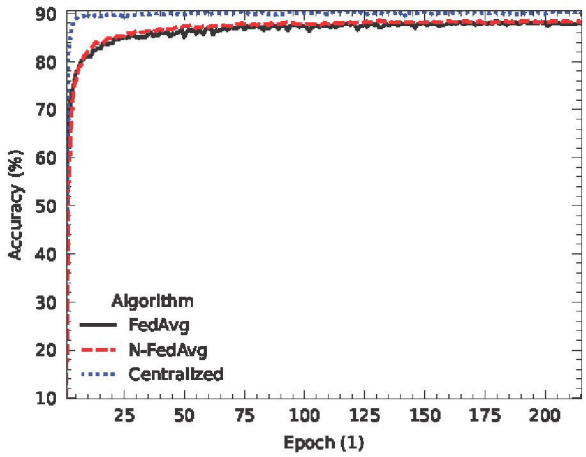


Fig. 5. Change curve of epoch and accuracy of the algorithm.

From Figures 5 and 6, we can find that the experiments on the CIFAR-10 dataset show that the overall training effect of the Centralized algorithm is better, but this situation of centralized training on the data seriously violates the FL for data privacy. prerequisite for protection. In addition, the effect of the N-FedAvg is better than the experimental effect of the FedAvg, not only the accuracy is 1.3% higher than that of FedAvg, but also the final loss function value is lower than the other two algorithms.

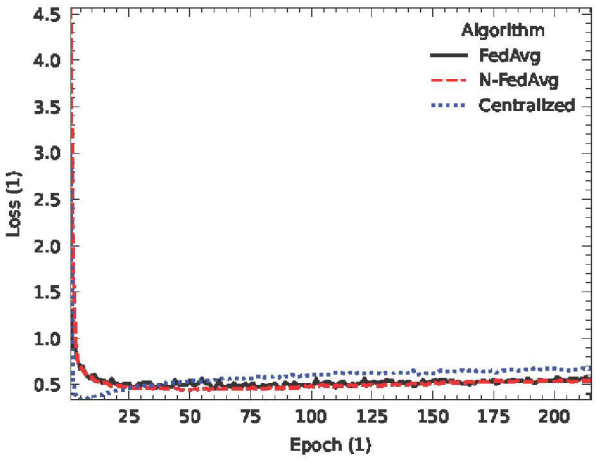


Fig. 6. The change curve of the number of epochs of the algorithm and the value of the loss function.

We make a table of epochs versus global accuracy relative to the baseline FedAvg with the same hyperparameter settings as in Section 4.2, given the specified target test set accuracy.

TABLE I. THE NUMBER OF EPOCH FOR THE RELATED ALGORITHM TO OBTAIN THE SPECIFIED ACCURACY

| | 80% | 83% | 86% | 88% |
|---|---|---|---|---|
| FedAvg | 7 | 16 | 36 | 140 |
| N-FedAvg | 8 | 12 | 27 | 74 |
| Centralized | 2 | 3 | 3 | 4 |

## D. Evaluation and Analysis

### 1) Sequential vs random selection of clients

Sequential selection of clients has a slightly different selection strategy than random selection of clients. Sequential selection of clients can reduce the randomness when selecting clients, so that all client data can have the opportunity to participate in FL, avoiding local data of some clients can't participate in FL, or the local data of some clients participate in FL with lower probability.

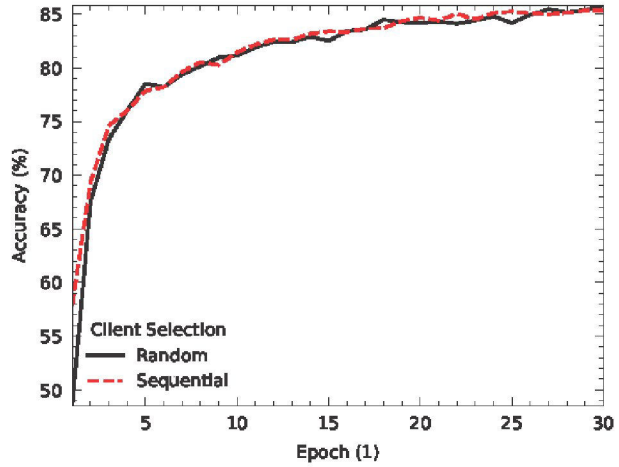


Fig. 7. Accuracy curve of sequential and random selection of clients.

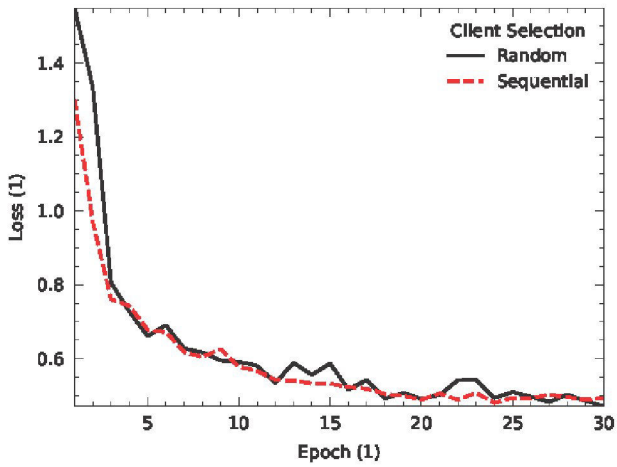


Fig. 8. The change curve of the loss function value of sequentially and randomly selected clients.

However, the sequential selection of clients does not take into account the heterogeneity among numerous client devices and the distribution of Non-IID data among clients, including differences in storage, CPU computing power, network transmission and other aspects. The configuration makes the computing time of the devices different, and even causes individual devices to be directly disconnected, and eventually the running time of the entire FL task will be longer [26].

As shown in Figures 7 and 8, through the comparative analysis of randomly selected clients and sequentially selected clients, we found that at the beginning of training, the global model accuracy of sequentially selected clients is higher than that of randomly selected clients, but the two eventually both the model accuracy and the loss function value are within 0.1% deviation. However, the accuracy of sequentially selected clients is higher at the beginning, and the value of loss function is lower. Therefore, we have reason to believe that in a real FL environment, selecting clients sequentially is slightly better than randomly selecting clients, which can prevent the local data of some clients from participating in FL with low probability. Sometimes a very simple method can bring unexpected results.

### 2) Cosine Annealing vs Static Learning Rate

When we use the SGD to optimize the objective function, as we get closer to the global minimum value of the loss function value, the LR should become smaller to make the model as close as possible to this point, and cosine annealing can pass the cosine function to reduce the LR [37]. In the cosine function, as the independent variable increases, the cosine value first decreases slowly, then decreases rapidly, and then decreases slowly again. This descending mode works well with the LR, and the loss function value is also small, producing good results in a very computationally efficient way.

As shown in Figures 9 and 10, through the analysis of the training effect of the cosine annealing LR and the static LR, when we use the cosine annealing dynamic LR to optimize the iterative process, the accuracy of the global model will be improved by 1.3%, and The loss function value is 9.1% lower than the static LR. Therefore, cosine annealing is a good way to dynamically change the LR.

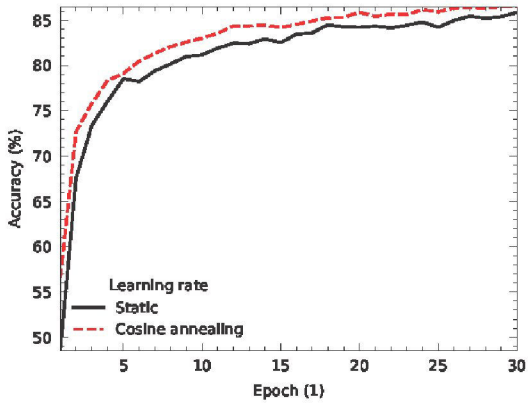


Fig. 9. Accuracy curve of static learning rate and cosine annealing learning rate.

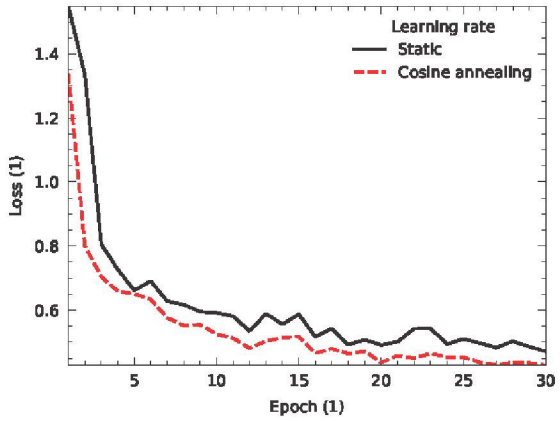


Fig. 10. The change curve of the loss function value of the static learning rate and the cosine annealing learning rate.

*3) Sparse vs normal model*

For clients participating in FL, model compression has become a very necessary choice. Because the model compression can reduce the communication pressure between the server and the client, thereby speeding up the overall process of FL, and also preventing the leakage of global model parameters [42].

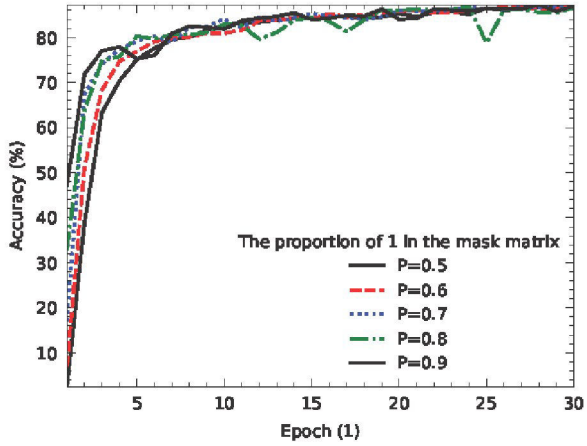


Fig. 11. The relationship between the accuracy and the number of 1 in the mask matrix.

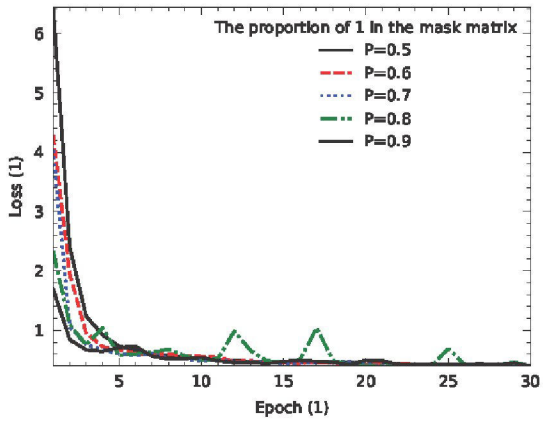


Fig. 12. The relationship between the loss function value and the number of 1 in the mask matrix.

Let's look at the performance of the model after parameter sparsity is shown in Figures 11 and 12. It can be seen that as the number of 1 in the mask matrix increases, the performance of the sparse model decreases at the beginning of the epochs, and the loss function value is relatively small. However, with the change of epoch, the performance of the model and the value of the loss function gradually return to a normal state.

## V. Conclusion

In this paper, we first introduce the basic concepts of client selection, dynamic learning rate and model compression, and then describe the detailed execution process of our N-FedAvg algorithm based on FedAvg. Then, we use the N-FedAvg algorithm to experiment on CIFAR-10 dataset, and compare it with Centralized and FedAvg algorithms to show the robustness and fast convergence of our proposed algorithm. The relevant experimental results were further evaluated and analyzed. Experiments also show that the N-FedAvg algorithm can prevent the local data of some clients from participating in FL with a low probability by sequentially selecting clients, and the cosine annealing learning rate can make the global model converge to the optimal target at a relatively fast speed, the sparsity method can improve the communication efficiency between the server and the client and prevent the leakage of global model parameters to a certain extent. Through the comparative analysis with the FedAvg algorithm in the CIFAR-10 dataset, we found that the N-FedAvg algorithm can improve the accuracy of the final converged global model by 1.34%, and reduce the loss function value by 2.77%, which has a good experimental effect.

Under the influence of the current changing world situation and the outbreak of COVID-19, FL has broad application prospects in medical, finance, insurance, smart industry and other related fields, and the N-FedAvg algorithm proposed in this paper can be used for FL in related industries. We hope to improve the following aspects of the N-FedAvg algorithm in the next stage, e.g., dynamic global epoch and batch size, reasonable selection of heterogeneous clients, proper processing of Non-IID data, embedding the proposed strategy in other FL Algorithms and combined trusted computing [50-52] conduct further related experiments on real datasets.

## REFERENCES


[1] "Data volume of internet of things (iot) connections worldwide," https://www.statista.com/statistics/1017863/worldwide-iotconnecteddevices-data-size/, accessed: 2021-02-17.

[2] J. Kooistra, Newzoo's 2018 global mobile market report: Insights into the world's 3 billion smartphone users, Newzoo 11 (2018).

[3] Lim, W. Y. B., Luong, N. C., Hoang, D. T., Jiao, Y., Liang, Y. C., Yang, Q., ... & Miao, C. (2020). Federated learning in mobile edge networks: A comprehensive survey. IEEE Communications Surveys & Tutorials, 22(3), 2031-2063.

[4] Miao, X., Nie, X., Shao, Y., Yang, Z., Jiang, J., Ma, L., & Cui, B. (2021, June). Heterogeneity-Aware Distributed Machine Learning Training via Partial Reduce. In Proceedings of the 2021 International Conference on Management of Data (pp. 2262-2270).

[5] Li, M., Andersen, D. G., Park, J. W., Smola, A. J., Ahmed, A., Josifovski, V., ... & Su, B. Y. (2014). Scaling distributed machine learning with the parameter server. In 11th USENIX Symposium on Operating Systems Design and Implementation (OSDI 14) (pp. 583-598).

[6] Li, M., Zhou, L., Yang, Z., Li, A., Xia, F., Andersen, D. G., & Smola, A. (2013, December). Parameter server for distributed machine learning. In Big learning NIPS workshop (Vol. 6, p. 2).

[7] EU. 2016. REGULATION (EU) 2016/679 OF THE EUROPEAN PARLIAMENT AND OF THE COUNCIL on the protection of natural persons with regard to the processing of personal data and on the free movement of such data, and repealing directive 95/46/EC (general data protection regulation). Retrieved December 26, 2018 from https://eur-lex.europa.eu/legal-content/EN/TXT.

[8] Bai, X., & Zhong, L. (2021, August). Legal Protection of Blockchain from the Perspective of the Cybersecurity Law: Legislation and Practice of China. In International Conference on Blockchain and Trustworthy Systems (pp. 585-592). Springer, Singapore.

[9] Cui, Shujie, and Peng Qi. "The legal construction of personal information protection and privacy under the Chinese Civil Code." Computer Law & Security Review 41 (2021): 105560.

[10] Yang, Q., Liu, Y., Chen, T., & Tong, Y. (2019). Federated machine learning: Concept and applications. ACM Transactions on Intelligent Systems and Technology (TIST), 10(2), 1-19.

[11] McMahan, H. B., Moore, E., Ramage, D., & y Arcas, B. A. (2016). Federated learning of deep networks using model averaging. arXiv preprint arXiv:1602.05629.

[12] Liu, Y., Fan, T., Chen, T., Xu, Q., & Yang, Q. (2021). FATE: An industrial grade platform for collaborative learning with data protection. Journal of Machine Learning Research, 22(226), 1-6.

[13] Ryffel, T., Trask, A., Dahl, M., Wagner, B., Mancuso, J., Rueckert, D., & Passerat-Palmbach, J. (2018). A generic framework for privacy preserving deep learning. arXiv preprint arXiv:1811.04017.

[14] Bonawitz, K., Eichner, H., Grieskamp, W., Huba, D., Ingerman, A., Ivanov, V., ... & Roselander, J. (2019). Towards federated learning at scale: System design. Proceedings of Machine Learning and Systems, 1, 374-388.

[15] Brock, A., Donahue, J., & Simonyan, K. (2018). Large scale GAN training for high fidelity natural image synthesis. arXiv preprint arXiv:1809.11096.

[16] He, C., Li, S., So, J., Zeng, X., Zhang, M., Wang, H., ... & Avestimehr, S. (2020). Fedml: A research library and benchmark for federated machine learning. arXiv preprint arXiv:2007.13518.

[17] Alsaleem, M. Y., & Hasoon, S. O. (2020). Predicting Bank Loan Risks Using Machine Learning Algorithms. AL-Rafidain Journal of Computer Sciences and Mathematics, 14(1), 149-158.

[18] Kaissis, G. A., Makowski, M. R., Rückert, D., & Braren, R. F. (2020). Secure, privacy-preserving and federated machine learning in medical imaging. Nature Machine Intelligence, 2(6), 305-311.

[19] Nguyen, D. C., Ding, M., Pathirana, P. N., Seneviratne, A., Li, J., Niyato, D., & Poor, H. V. (2021). Federated learning for industrial internet of things in future industries. IEEE Wireless Communications.

[20] He, C., Shah, A. D., Tang, Z., Sivashunmugam, D. F. N., Bhogaraju, K., Shimpi, M., ... & Avestimehr, S. (2021). FedCV: A Federated Learning Framework for Diverse Computer Vision Tasks. arXiv preprint arXiv:2111.11066.

[21] Zhao, Y., Li, M., Lai, L., Suda, N., Civin, D., & Chandra, V. (2018). Federated learning with non-iid data. arXiv preprint arXiv:1806.00582.

[22] Hsieh, K., Phanishayee, A., Mutlu, O., & Gibbons, P. (2020, November). The non-iid data quagmire of decentralized machine learning. In International Conference on Machine Learning (pp. 4387-4398). PMLR.

[23] Diao, E., Ding, J., & Tarokh, V. (2020). Heterofl: Computation and communication efficient federated learning for heterogeneous clients. arXiv preprint arXiv:2010.01264.

[24] Mao, Y., Zhao, Z., Yan, G., Liu, Y., Lan, T., Song, L., & Ding, W. (2021). Communication Efficient Federated Learning with Adaptive Quantization. arXiv preprint arXiv:2104.06023.

[25] Rothchild, D., Panda, A., Ullah, E., Ivkin, N., Stoica, I., Braverman, V., ... & Arora, R. (2020, November). Fetchsgd: Communication-efficient federated learning with sketching. In International Conference on Machine Learning (pp. 8253-8265). PMLR.

[26] Li, T., Sahu, A. K., Talwalkar, A., & Smith, V. (2020). Federated learning: Challenges, methods, and future directions. IEEE Signal Processing Magazine, 37(3), 50-60.

[27] Lam, M., Wei, G. Y., Brooks, D., Reddi, V. J., & Mitzenmacher, M. (2021, July). Gradient disaggregation: Breaking privacy in federated learning by reconstructing the user participant matrix. In International Conference on Machine Learning (pp. 5959-5968). PMLR.

[28] Geiping, J., Bauermeister, H., Dröge, H., & Moeller, M. (2020). Inverting gradients-how easy is it to break privacy in federated learning?. Advances in Neural Information Processing Systems, 33, 16937-16947.

[29] Rodríguez-Barroso, N., Martínez-Cámara, E., Luzón, M. V., & Herrera, F. (2022). Dynamic defense against byzantine poisoning attacks in federated learning. Future Generation Computer Systems.

[30] Sun, Z., Kairouz, P., Suresh, A. T., & McMahan, H. B. (2019). Can you really backdoor federated learning?. arXiv preprint arXiv:1911.07963.

[31] Chai, Z., Ali, A., Zawad, S., Truex, S., Anwar, A., Baracaldo, N., ... & Cheng, Y. (2020, June). Tifl: A tier-based federated learning system. In Proceedings of the 29th International Symposium on High-Performance Parallel and Distributed Computing (pp. 125-136).

[32] Ribero, M., & Vikalo, H. (2020). Communication-efficient federated learning via optimal client sampling. arXiv preprint arXiv:2007.15197.

[33] Lai, F., Zhu, X., Madhyastha, H. V., & Chowdhury, M. (2020). Oort: Informed participant selection for scalable federated learning. arXiv preprint arXiv:2010.06081.

[34] Wang, H., Kaplan, Z., Niu, D., & Li, B. (2020, July). Optimizing federated learning on non-iid data with reinforcement learning. In IEEE INFOCOM 2020-IEEE Conference on Computer Communications (pp. 1698-1707). IEEE.

[35] Cho, Y. J., Wang, J., & Joshi, G. (2020). Client selection in federated learning: Convergence analysis and power-of-choice selection strategies. arXiv preprint arXiv:2010.01243.

[36] Brownlee, J. (2019). Understand the impact of learning rate on neural network performance. Machine Learning Mastery.

[37] Loshchilov, I., & Hutter, F. (2016). Sgdr: Stochastic gradient descent with warm restarts. arXiv preprint arXiv:1608.03983.

[38] Ji, R., Lin, S., Chao, F., Wu, Y., & Huang, F. (2018). Deep neural network compression and acceleration: a review. Journal of Computer Research and Development, 55(9), 1871.

[39] Li, G., Liu, F., & Xia, Y. (2020, November). Overview of deep convolutional neural network pruning. In 2020 International Conference on Image, Video Processing and Artificial Intelligence (Vol. 11584, p. 1158419). International Society for Optics and Photonics.

[40] Krishnamoorthi, R. (2018). Quantizing deep convolutional networks for efficient inference: A whitepaper. arXiv preprint arXiv:1806.08342.

[41] Hinton, G., Vinyals, O., & Dean, J. (2015). Distilling the knowledge in a neural network. arXiv preprint arXiv:1503.02531, 2(7).

[42] Yang Qiang, Huang Anbu, Liu Yang, Chen Tianjian. "Practicing Federated Learning". Publishing house of electronics industry (2021).

[43] McMahan, B., Moore, E., Ramage, D., Hampson, S., & y Arcas, B. A. (2017, April). Communication-efficient learning of deep networks from

decentralized data. In Artificial intelligence and statistics (pp. 1273-1282). PMLR.

[44] Li, T., Sahu, A. K., Zaheer, M., Sanjabi, M., Talwalkar, A., & Smith, V. (2020). Federated optimization in heterogeneous networks. Proceedings of Machine Learning and Systems, 2, 429-450.

[45] Xie, C., Koyejo, S., & Gupta, I. (2019). Asynchronous federated optimization. arXiv preprint arXiv:1903.03934.

[46] Hu, L., Yan, H., Li, L., Pan, Z., Liu, X., & Zhang, Z. (2021). MHAT: An efficient model-heterogenous aggregation training scheme for federated learning. Information Sciences, 560, 493-503.

[47] Xiao, J., Du, C., Duan, Z., & Guo, W. (2021, July). A Novel Server-side Aggregation Strategy for Federated Learning in Non-IID situations. In 2021 20th International Symposium on Parallel and Distributed Computing (ISPDC) (pp. 17-24). IEEE.

[48] Krizhevsky, A., & Hinton, G. (2009). Learning multiple layers of features from tiny images.

[49] He, K., Zhang, X., Ren, S., & Sun, J. (2016). Deep residual learning for image recognition. In Proceedings of the IEEE conference on computer vision and pattern recognition (pp. 770-778).

[50] Yang, M., & Fu, F. (2020, June). Secure Big Data Computing Based on Trusted Computing and Key Management. In International Conference on Applications and Techniques in Cyber Security and Intelligence (pp. 1114-1118). Springer, Cham.

[51] Wagner, P. G., Birnstill, P., & Beyerer, J. (2019, September). Challenges of Using Trusted Computing for Collaborative Data Processing. In International Workshop on Security and Trust Management (pp. 107-123). Springer, Cham.

[52] Dong, S., Zeng, D., Gu, L., & Guo, S. (2020, December). Offloading Federated Learning Task to Edge Computing with Trust Execution Environment. In 2020 IEEE 17th International Conference on Mobile Ad Hoc and Sensor Systems (MASS) (pp. 491-496). IEEE.