[1]*Xide Wang
[2]Qingyan Fang

cFL：Data distribution edge clustering algorithm based on deep Federated learning

Abstract: - Federated learning (FL) allows end-devices to train local models on their respective local datasets and collaborate with the server to train a global predictive model, thus achieving the goal of machine learning while protecting the privacy and sensitive data of end-devices. However, simultaneous access to the server by a large number of end devices may result in increased transmission latency and some local models may have malicious behavior, converging in the opposite direction to the global model. As a result, additional communication costs can occur during the federation learning process. Existing research has mainly focused on reducing the number of communication rounds or cleaning dirty local data. In order to decrease the overall amount of local updates, we provide an edge-based model cleaning and device clustering strategy in this study. By computing the cosine similarity between local update parameters and global model parameters over a multi-dimensional space, the approach assesses if local updates are necessary while preventing pointless communications. In addition, end devices are grouped together based on where they are connected to the network and connect to the cloud via mobile edge nodes in clusters, therefore lowering the latency associated with high concurrent server access. Convolutional neural networks and Soft max regression are used, for instance, to perform MNIST handwritten digit recognition, and the effectiveness of the suggested approach to enhancing communication efficiency is confirmed. According to experimental findings, the edge-based model cleaning and device clustering technique decreases the quantity of local updates by 60% and speeds up the model's convergence by 10.3% when compared to the conventional FL approach.

Keywords: mobile edge computing; model cleaning; clustering; cosine similarity

I. INTRODUCTION

The Many industries experience data silos as a result of competitive competition and the requirement to preserve data privacy. It is extremely difficult to integrate data from separate organizations, which is almost impossible in practice, even within the same corporation, let alone between different divisions. Data privacy and security are becoming increasingly important on a worldwide scale as Big Data continues to evolve. There are many difficulties with traditional machine learning techniques that transmit data from end devices to the cloud for deep learning [1]. FL, one of the core technologies of artificial intelligence, has emerged as a promising approach to address this challenge. In FL, the cloud server maintains a global model that is shared by all end devices [2]. The end devices only need to train the local model using local datasets and upload the trained local updates to the cloud server to participate in global aggregation, and then iterate the process [3]. The learning process of FL does not involve the transmission of data, and is therefore able to protect the privacy and security of the data and achieve the goals of machine learning while protecting data privacy.

FL still has a lot to learn about effective communication [4]. On the one hand, complicated neural networks are widely used in advanced machine learning applications that are installed on end devices, leading to local updates that frequently have huge gradient vectors. The network between end devices and cloud servers, in contrast, has two issues: first, a limited amount of network bandwidth and the high cost of servers for high bandwidth services; and second, an asymmetric network connection between local and cloud where the upstream speed is typically much slower than the downstream speed. Therefore, when a large number of end devices are involved in federation learning, high concurrent access increases the communication latency of model transmission, and the instability of the network can lead to bottlenecks in the training process [5]. On the other hand, there is heterogeneity in the devices involved in federation learning, and their local data tend to obey a non-independent homogeneous distribution [6].Because of this, the local models created using these tools and data are frequently of poor quality and are referred to as "dirty models." The accuracy of the global model will be significantly impacted if local updates from filthy models are sent to the cloud to participate in aggregation. This will also result in increased communication costs. Thus, it is essential to lessen FL's network footprint [7].

[1] Harbin University, Harbin,Heilongjiang, China

[2] Harbin Guangsha College,Harbin,Heilongjiang China.

*Corresponding author:miyeerf@126.com

This work analyzes how edge computing and communication resources can be effectively used to obtain the best federation learning performance in order to address the aforementioned problems [8]. The usage of mobile edge nodes deployed at various network locations to serve as hubs for communication between distant clouds and end devices is a common mobile edge computing design [9]. End devices are split into several clusters according to their LAN addresses, as seen in Figure 1. Each end device calculates the cosine similarity between the local update parameters and the global model parameters during the local update phase [10]. The local model is not transferred to the server for aggregation if the similarity is low, indicating that it is a malicious or useless update and preventing the need for further communication [11]. Mobile edge nodes installed in each LAN gather and aggregate essential or non-malicious local updates during the edge aggregation phase because each global aggregation uses network connection and cloud computing resources. Once the edge aggregated models have been delivered to the cloud server for global aggregation, delay associated with high concurrent access to the server is avoided, and the utilization of the computing resources of the mobile edge nodes is rationalized to relieve the load on the cloud [12].

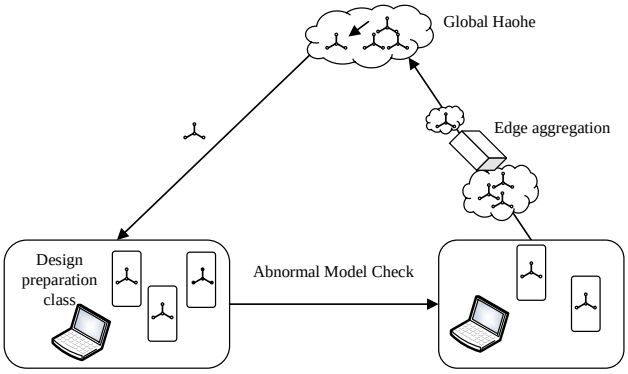


Figure 1 System framework

The main contributions of this paper are as follows:

By computing the cosine similarity between global model parameters and local update parameters and contrasting their disparities in parameter values and model convergence directions, a method for cleaning up models is given to get rid of pointless updates and minimize the amount of local updates.

The introduction of a clustering technique based on the network location of end devices. By installing mobile edge nodes in each cluster as a communication hub between the cloud server and the end devices to gather and aggregate local updates, the latency issue caused by high concurrent server access is addressed.

On two FL models, the usefulness of the suggested strategy to communication optimization is confirmed. When compared to conventional federation learning, the suggested strategy reduced the number of network updates by 60% and sped up the model's convergence by 10.3%, according to extensive testing using the MNIST dataset [13].

II. RELATED WORKS

In recent years, communication optimization work in FL has received continuous attention from researchers, and a series of optimization methods have been proposed, mainly including reducing the number of communication rounds and terminal equipment cleaning [14].

A. Reducing the number of communication rounds

To reduce the communication overhead, existing work has focused on increasing edge computation and optimization algorithms. Increasing the amount of edge computing refers to using edge resources to compute local models with better convergence and model accuracy, thus reducing the computational burden on the cloud server and accelerating the convergence of the global model [15]. For example, the optimization scheme proposed in [16] focuses on increasing the number of end devices and improving the computational power of end devices to speed up the convergence of the global model. In addition, [17] proposed a three-tier federation learning system which

aggregates models at both the edge server and cloud server levels to achieve a good compromise between communication and computation.

In terms of algorithm optimization, [18] suggests a control algorithm that chooses the best balance between local updates and global parameter aggregation for a given resource budget in order to minimize the loss function, taking into account that federation learning involves data distributed across multiple end devices. By determining whether local updates are consistent with the overall trend to prevent irrelevant updates, a federation learning technique called communication mitigation is proposed in [19]. This method lowers the total number of communication rounds for FL.

B. Data Clustering

Malicious end devices may upload damaging models during the federation learning training process and stop the process altogether. These malevolent devices may unintentionally produce low-quality data or purposefully carry out unpredictable updates, such as data poisoning assaults. The convergence speed of the global model will be slowed down, and a great deal of superfluous communication will result, if the local models trained by these devices are uploaded straight to the cloud for aggregation. Studies have suggested a method for choosing trustworthy endpoint devices and established the idea of reputation as a metric to address this problem. Additionally, [20] developed a technique that makes use of a pre-trained anomaly detection model to identify end device abnormal behavior and get rid of its negative effects on the overall model. The technique specifically produces low-dimensional agent vectors and makes use of them for anomaly identification. [21] suggests a federated learning framework for clustering end devices into groups in order to weed out aberrant players. This framework is based on cosine similarity clustering between parameter updates. [22] concentrated on cleaning up local anomaly data to lessen the effect of anomalous data on the overall model.

In contrast to the research mentioned above, the communication delays brought on by high server concurrency and the additional communication issues brought on by the uploading of pointless local updates to the cloud are the main topics of this paper. This study suggests a method to eliminate unnecessary updates whose differences are greater than a predetermined threshold, compare the differences between local update parameters and global model parameters, and cluster devices with the same network position. By relocating the edge nodes to connect with the server as clusters, FL's communication efficiency is increased.

III. RELATED CONCEPTS AND PROBLEM DESCRIPTIONS

A. Federal Learning

A horizontal FL system consisting of a server and N end devices, and a global model trained collaboratively using the Fed Avg algorithm. The FL scheme explored is limited to the mobile communication network scenario, considering that existing FL has a large number of participants and is widely distributed. Assuming that each end device $i \in N$ has an independent local data set D_i, for any data sample $\{x_j, y_j\}$, where x_j represents the input to the model, the learning task of the device is to find a model parameter w that describes y_j and minimizes the loss function $f_j(w)$ of the model. The loss function is used to evaluate the difference between the model prediction and the true situation, and the smaller the difference, the better the model. For example, for a linear regression model, the number of losses can be expressed as $f_j(w) = \frac{1}{2}\left(x_j^T w - y_j\right), y_j \in R$. For convenience, this paper uses $|D_i|$ to represent the size of the data set and $D = \sum_{i-1}^{N} D_i$ to define the total size of the data involved in learning. Thus, in federal learning, the loss function of end device i on its dataset is defined as

$$F_i(w) = \frac{1}{|D_i|} \sum_{j \in D_i} f_j(w) \quad (1)$$

According to the Fed Avg algorithm, the global loss function on the server can be defined as

$$F\left(w\right)=\sum_{i-1}^{N}\frac{D_i}{D}F_i\left(w\right) \quad (2)$$

The learning objective of federal learning is to minimize the global loss function represented by equation (2).

B. Model parameters

A global model parameter is initialized by the server at the start of FL training and optimized by the end device. FL typically need T global iterations to get the loss function to converge. Similar to this, end device i must perform numerous rounds of local training on its local dataset D_{i-} during each global iteration to determine the ideal model parameters.

$$w_i^{(t)}=\arg\min F\left(w\right) \quad (3)$$

The complexity inherent in most machine learning models is usually addressed by using Stochastic Gradient Descent (SGD) in equation (3). The best local update parameters $w_i^{(t)}$ trained by these end devices on their local datasets need to be sent to the server to participate in global aggregation, which according to the Fed Avg algorithm, can be expressed as

$$w_i^{(t+1)}=\sum_{i-1}^{N}\frac{D_i}{D}w_i^{(t)} \quad (4)$$

The goal of global aggregation is to minimize the global loss function $F\left(w\right)$ in equation (2), and then the server broadcasts $w^{(t+1)}$ to all end devices as the global model parameter for the next iteration. After several global iterations, the global model converges, and finally a stable global model accuracy $\varepsilon\left(||\nabla F\left(w^{(t)}\right)||\leq\varepsilon\leq||\nabla F\left(w^{(t-1)}\right)||^{[22]}\right)$ is obtained.

C. Problem description

Each end device reduces its loss function across its local data set during each global iteration. The disparities between the local and global models are substantial because of the variability of the end devices and the non-independent homogenous distribution of the local data [23]. Some local models' gradients might run counterclockwise to those of the global model. In other words, certain locally trained models with poor training may contaminate the global model $w^{(t+1)}$ during the model aggregation phase. The accuracy of the global model is impacted if these local models are transferred to the cloud for aggregation, and they also use up a lot of network capacity and slow down model transmission.

In this paper, δ_i is defined to denote the size of the update uploaded from terminal i in the t st global iteration, and due to the dimensional consistency of the model parameters, δ_i is assumed to be constant. For a given number of end devices N , then the total number of data uploaded from end devices in the t st global iteration is $\Upsilon_t=\sum_{i-1}^{T}\delta_i$. Assuming that $w*$ is the final target global model parameter, this paper seeks a solution to the following problem.

$$\min \sum_{i-1}^{T} \Upsilon_t \qquad (5)$$
$$s.t.w^* = \arg\min F\left(w\right)$$

IV. EDGE-BASED MODEL CLEANING AND CLUSTERING METHODS

In this section, this paper proposes a solution to improve the efficiency of FL communication, namely edge- based model cleaning and device clustering in FL (ecFL). The solution to the problem presented in the previous section is given in this paper.

A. Anomalous model detection

Anomaly local models or filthy models are terms used to describe local models that are unrelated to the convergence of the global model. In order to avoid downloading filthy models and cut down on wasteful communication expenses, this paper's goal is to identify them. As previously indicated, CMFL measures the significance of local updates by counting the number of parameters that have the same sign between global and local updates, for example, those satisfying $\frac{1}{N}\sum_{j-1}^{N} I\left(\mathrm{sgn}\left(u_j\right)=\mathrm{sgn}\left(\bar{u}_j\right)\right) < Threshold$ represents the global modelling parameters of the preceding global iteration, $\bar{u}_j$ denotes the local modelling parameters in the current global iteration, and N denotes the total number of parameters.

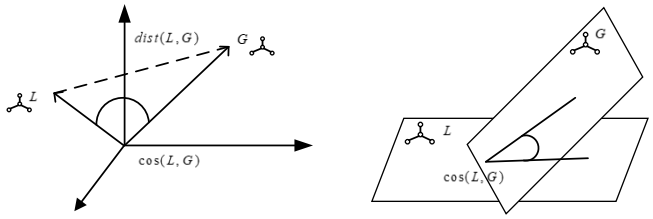
While the parameter sign of the model update identifies the direction of improvement (increase or decrease) of the model parameters in each dimension, the values of the parameters also reflect the extent to which the model parameters change in different directions.For instance, the values of the model parameters in a standard Soft max regression model can be understood as the Soft max probability values for each category. As a result, the local update's and the global model's parameter values ought to be comparable. We infer that there is no association between the two model parameters if the global and local models both have the same sign of the relevant parameters but have considerably different parameter values. This naturally prompts the query, "Are there any other methods to find the correlation between the global update and the local update?"

The basics of federated learning, machine learning and edge computing, are the focus of this study. Euclidean distance, cosine similarity, and Jaccard similarity coefficient are three common similarity measurements used in machine learning. While cosine similarity measures the angle of spatial vectors, which more clearly displays the difference in direction, Euclidean distance measures the absolute distance of each point in space, which is directly tied to the point's coordinates. This work makes the assumption that cosine similarity is preferable than Euclidean distance for anomaly model identification because model parameters suggest the direction of convergence of the model.

We also found an Angle-Based Anomaly Detection method (ABOD) in our edge computing based data cleaning algorithm. ABOD contains a set of data point sets, for any point o. ABOD calculates the angles $\angle xoy$ of a pair of points $\left(x, y\right)$ and o satisfying $x \neq o$ and $y \neq o$, and determines whether the data are anomalous based on the angular variance of each point. The findings in edge computing confirm the conjecture of this paper that cosine similarity algorithms focusing on angular variance are more suitable as anomaly model detection algorithms.

As illustrated in Figure 2, where G stands for the global model and signifies the local model, this research abstracts the model parameters in geometric space as two-dimensional vectors and high-dimensional spatial planes. The cosine distance, which measures the angle of the spatial vector and is more a reflection of the difference in direction than in position, may be seen in the figure 2. For the given models L and G in Figure 2(a), the distance $dist\left(L, G\right)$ between them is absolute. If the position of L remains constant and the parameters of G change in any dimension, the cosine distance will change at that point (because the angle changes). For the given models L

and G in Figure 2(b), the angle between the two planes represents their cosine distance; the smaller the angle, the smaller the cosine distance, indicating that the two models are more similar.



(a) Model parameters in low dimensional space

(b) Model parameters in high-dimensional space

Figure 2 Abstract representation of model parameters in different dimensional spaces

In order to calculate the similarity between the model parameters for any terminal device involved in training, this paper first vectorizes the local update and the global model by using digital image processing methods. Assuming the local update $L_l = [l_1, l_2, ..., l_m]$ in the current iteration of the device and the global update $G_{l-1} = [g_1, g_2, ..., g_m]$ in the previous iteration, the similarity between the global model and the local model can be calculated based on the cosine similarity algorithm, i.e:

$$\text{similarity}_{(L_l, G_{l-1})} = \cos(L_l, G_{l-1}) = \frac{\langle L_l, G_{l-1} \rangle}{|L_l||G_{l-1}|} = \frac{\sum_{j-1}^{m}(l_j \times g_j)}{\sqrt{\sum_{j-1}^{m}(l_j)^2} \times \sqrt{\sum_{j-1}^{m}(g_j)^2}} \quad (6)$$

The Trigonometric Theorem states that the cosine value range is [-1,1]. When the cosine value is close to 1, it denotes a closer relationship between the two vectors; when it goes to -1, it denotes a greater difference in direction; and when it is close to 0, it denotes a nearly orthogonal relationship between the two models. In general, similarity values below 0 do not correspond to readers' reading preferences. As a result, this paper normalizes the models' degree of similarity to the interval [0,1].

$$Similarity_{(L_l, G_{l-1})} = 0.5 \times \cos(L_l, G_{l-1}) + 0.5 \quad (7)$$

Similarly, when $Similarity_{(L_l, G_{l-1})}$ tends to zero, the validity of the local model parameters is lower; conversely, the validity is higher.

In this study, a threshold value is established, and the model parameter is set to NULL to indicate that the model update is incorrect and does not take part in global aggregation when the correlation $Similarity_{(L_l, G_{l-1})}$ between the local model and the global model is less than the set threshold value. The local training and implementation of the anomaly model detection in the end device are described in detail in Algorithm 1. The experimental part includes a detailed description of threshold setting.

Algorithm 1 Anomaly model detection algorithm.

Input: dataset D of end device z, global model G of the last iteration and threshold f, learning rate λ

Output: local update L

1) Partition the dataset D into multiple minibatches to obtain the minibatch set Bt;

2) FOR b IN Bt DO

3) $L \leftarrow L - \lambda \nabla F(L)$

4) END FOR

5) $Similarity_{(L,G)} = 0.5 \times \cos(L,G) + 0.5$;//Calculate the cosine similarity between the local and global update parameters

6) IF $Similarity_{(L,G)} < f$ THEN

7) $L = NULL$;

8) END IF

B. Edge clustering

The uploading of local updates and the downloading of the global model take up the majority of FL's communication time. The download time of the global model is trivial and is not taken into account in this work because, due to the network's design, the downstream bandwidth is far greater than the upstream capacity. However, because the server's network capacity is constrained, there can only be a certain number of connections active at once. The data transfer rate would be inversely proportional to the number of devices in the network and energy consumption would grow if all end devices were to communicate with the server directly. Given Shannon's theorem's restrictions, we can assume that the model has

$$\upsilon = reverse(\mu) \times B1n\left(1 + \frac{S}{N_p}\right) \quad (8)$$

Where B and $\frac{S}{N_p}$ denote the uplink bandwidth and signal-to-noise ratio, respectively, μ denotes the number of devices connected to the server, and $reverse(\mu)$. denotes the function with μ . Assuming that the signal-to-noise ratio $\frac{S}{N_p}$ of the channel remains constant during the learning process of FL, and the size of the model update w_i is defined as M in this paper, and assuming that this value is constant, then the communication time for the end devices to participate in a global training can be expressed as

$$T_{com} = \frac{M}{\upsilon} = \frac{M}{reverse(\mu) \times B1n\left(1 + \frac{S}{N_p}\right)} \quad (9)$$

This shows that the communication time T_{com} increases monotonically with the number of device connections μ , i.e. the more devices connected to the server, the longer the communication time for federal learning.

Based on the fact that LAN bandwidth is much larger than WAN bandwidth, the idea of this paper is to allow end devices to communicate within the LAN as much as possible. As computing resources in the LAN are limited, large-scale global aggregation tasks still need to be done at the remote cloud. Therefore, this paper plans to implement joint communication of FLs between the LAN and the WAN. In short, this can be done using a clustering approach where end devices are aggregated in some way to form clusters and communicate with the cloud server as clusters to reduce model transmission latency and achieve the goal of federation learning.

Assuming that A_i is the LAN address of end device i and $clustsr_m$ is the cluster of devices with address A_m, the device clustering process can be expressed as

$$clustsr_m = \{i \mid A_i = A_m\} \quad (10)$$

In other words, depending on the LAN where the device is located, it can be divided into clusters, each of which is independent of the other. Once the clustering is complete, the question of who will communicate between the clusters and the cloud becomes a further consideration.

The first idea of this paper is to select a cluster head device in a local area network, inspired by the traditional wireless sensor networks [24]. The cluster head in a sensor network is in charge of combining the data gathered by the sensors and transmitting it to the aggregation nodes. The use of a cluster head device to gather all local updates in a LAN and transfer them to a cloud server for global aggregation is also taken into consideration in this article. A theoretical examination is conducted to see if the theory presented in this work holds true in real-world situations. For example, if there are p end devices in the LAN and each one has a local update size of k, the cluster head device must communicate $p \times k$ bytes of data in total. The size of the packets sent between a single device and the server, in other words

The cluster header reduces the number of devices directly connected to the server μ and increases the data transmission rate υ, but also increases the amount of data to be transmitted M. According to the definition of communication time $T_{com} = M / \upsilon$ in the previous section, it is difficult to determine the trend of communication time T_{com} when M and υ are changing at the same time. Therefore, the idea of clustering not only does not reduce the training communication time, but also may increase the transmission delay of the model.

Considering that a complex DL model training may contain millions of parameters per update, the local model contains a large gradient vector; and due to the clustering mechanism, the size of the data transferred between the edge and the cloud changes from k to $p \times k$, the size of the local update is even more non-negligible. Therefore, in order to ensure the consistency of the model variables in the communication process, this paper considers using the computing resources of the mobile edge nodes to calculate the weighted average of the local models in the cluster, i.e., the local updates satisfying $Similarity > Threshold$ in the local model will be transferred to the mobile edge nodes to participate in the edge aggregation and obtain the cluster model. In this paper, the process of edge aggregation is represented as

$$F_m(w) = \sum_{i \in \text{ clusser } m} \frac{D_i}{D_m} f_i(w) \quad (11)$$

Where $D_m = \sum_{i \in \text{ clusser } m} D_i$, all cluster models are uploaded to the cloud to take part in global aggregation to reduce the overall loss function before moving on to the following training cycle. The specifics of the edge aggregation procedure are shown in Algorithm 2. The ecFL model cleaning workflow is shown in Figure 3.

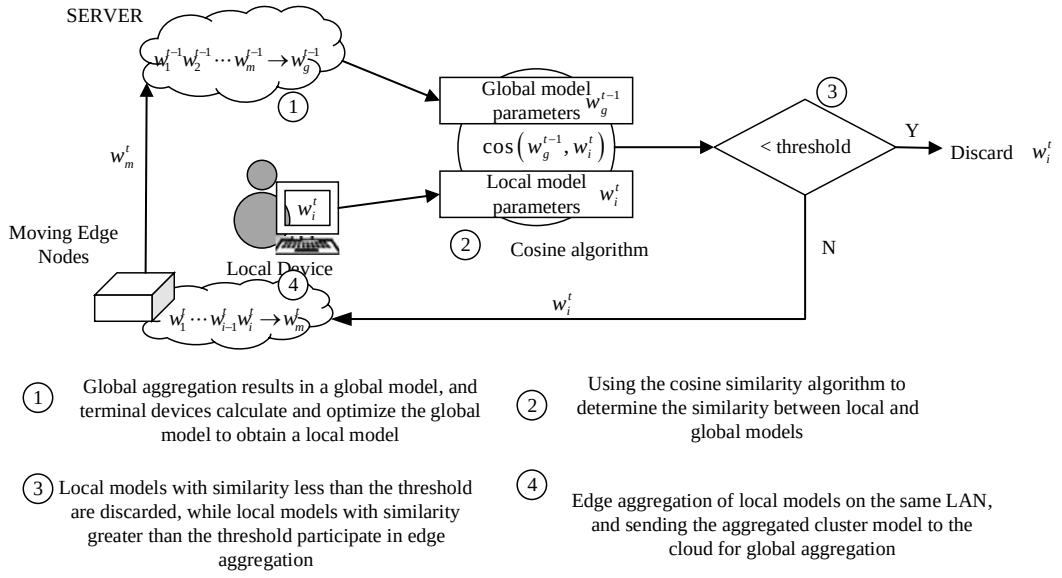


Figure 3 Workflow of model cleaning

Algorithm 2. Model cleaning and device clustering algorithms.

Input: set of end devices N , LAN location $\Lambda address(i)$ where device i is located

Output: global model G

1) FOR $iteration = 1, 2, ...$ DO

2) FOR devices i IN N DO

3) // Clustering based on device LAN addresses

4) $Cluster_m = \{i \mid i\, in \quad \Lambda address(m)\}$;

5) END FOR

6) FOR $m = 1, 2, ...$ DO

7) FOR Equipment i IN $Cluster_m$ DO

8) Execute the anomaly model detection algorithm to obtain local updates L_i ;

9) // Collect local updates within the cluster

10) $Cluster \bmod el_m = \{L_i \mid L_i\, is \quad NOT \quad NULL\}$;

11) END FOR

12) $Edge \bmod el_m = \sum_{i \in \text{clusser } m} \frac{D_i}{D_m} L_i$; // Get the edge model

13) END FOR

14) $G = \sum \frac{D_m}{D} Edge \bmod el_m$ // Transfer the edge model to the cloud server to participate in global aggregation to obtain the global model G for the next iteration

15) END FOR

C. *Analysis of the algorithm*

Minimizing the communication time for federal learning while ensuring learning convergence. Given that the dataset of end devices obeys a non-independent homogeneous distribution, minimizing the communication time is still an open problem. Therefore, this paper only proves theoretically that the proposed algorithm is guaranteed to converge, and relies on simulation experiments to illustrate the communication improvement of the proposed scheme. Instead of adding additional communication overhead, this process reduces the communication overhead. Quantitatively, assuming that there are p end devices participating in learning, and the local update size is fixed and the communication cost is x for a single upload from the local to the cloud server, the total communication cost for a global update in the traditional federation learning mode is $p \times x$; in contrast, assuming that the communication cost for a local update to the mobile edge node is y, and a total of $z\left(z < p\right)$ mobile edge nodes are set up, as the mobile edge nodes are closer to the $y << x$, then the total communication cost of a global update in ecFL is $z \times x + p \times y$, which is much smaller than the communication overhead in the traditional mode.

In addition, assume that the loss value of federal learning training to achieve the target accuracy is $L\left[w^*\right]$,i.e. $L\left[w^*\right] = |F\left(w\right) - F\left(w^*\right)|$. Then the time complexity of training the federal learning model can be limited to $\lim_{T\to\infty} \frac{1}{T} L\left[w^*\right] = \frac{1}{T}\left[O\left(\frac{1}{\lambda T}\right) - O\left(\sum_{t-1}^{T} f_t\right)\right]$, and since f is constant and the number of training iterations T can be reduced after model cleaning, this paper basically maintains a similar time complexity to traditional federal learning in terms of communication.

V. EXPERIMENTAL ANALYSIS

A. *Dataset description and experimental preparation*

In this study, we assess the MNIST dataset training performance of two distinct models for federal learning. Convolutional neural networks and Soft max regression are among the models.

MNIST is a handwritten digital dataset of 10,000 test samples and 60,000 training samples. The MNIST dataset consists of images that are each 28 by 28 pixels in size, with a grey value assigned to each pixel.

Soft max regression. The modeling process starts with converting the MNIST test images into probability distributions using the $Soft\max(\ )$ function, then calculating the cross-entropy between the true and predicted distributions, and finally updating the parameters using gradient descent to achieve convergence of the loss function.

MLP. i.e. multilayer perceptron neural network. In this experiment, two hidden layers are set.

CNN Convolutional layer (Convolutions, Cl), pooling layer (Subsampling, S2), convolutional layer (C3), pooling layer (Subsaglng, S4), convolutional layer (C5), fully connected layer (F6), and output layer (OUTPUT) make up the model's eight network layers in total. The convolutional kernel's dimensions are 5 by 5, and the model has eight network layers in total. ReLU serves as the activation function.

The TensorFlow framework was used to implement the experimental portion of this paper. The MNIST CNN and MNIST Soft max models are employed in the experiments. The MNIST training samples are evenly divided to 20

end devices for the MNIST Soft max model, with each end device receiving 3000 examples. Additionally, each end-device underwent 93 rounds of local training on its local dataset for each global iteration utilizing.

In order to simulate the effect of dirty data on the model in a real learning scenario, this paper attempts to do so by modifying the labels in the dataset. Previous studies have shown that the accuracy of the model is only affected when the proportion of dirty labels is greater than 0.7. To test this idea, this paper randomly modifies a certain proportion of the dataset labels in the training of the CNN model using different proportions of noisy datasets. Figure 4 shows the results of the change in model accuracy when the proportion of noisy labels was varied from 0% to 100%. When the proportion of noise labels is below 70%, the accuracy of the model still reaches 0.9; while when the proportion of noise labels exceeds 70%, the accuracy of the model drops sharply. Therefore, in order to evaluate the effectiveness of the ecFL algorithm in model cleaning, this paper randomly modified the sample labels in the MNIST dataset with more than 70% to simulate the dirty data situation in the FL environment.

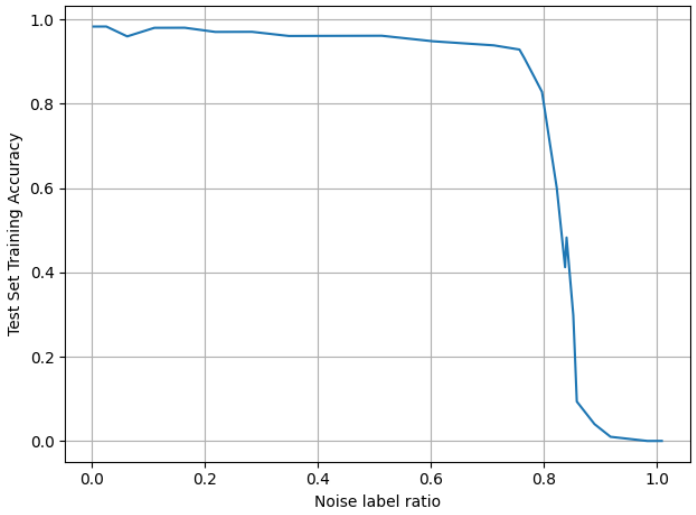
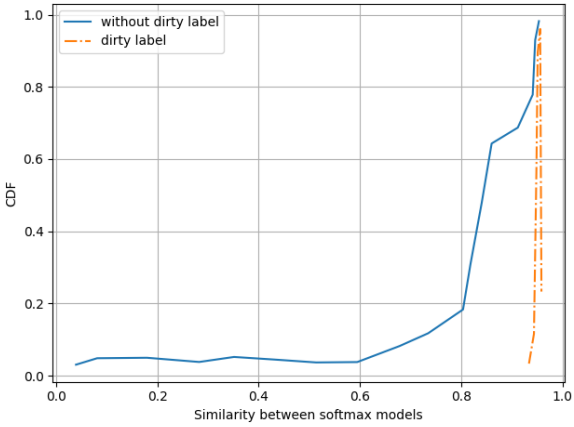


Figure 4 Accuracy of models trained on datasets with different noise label ratios

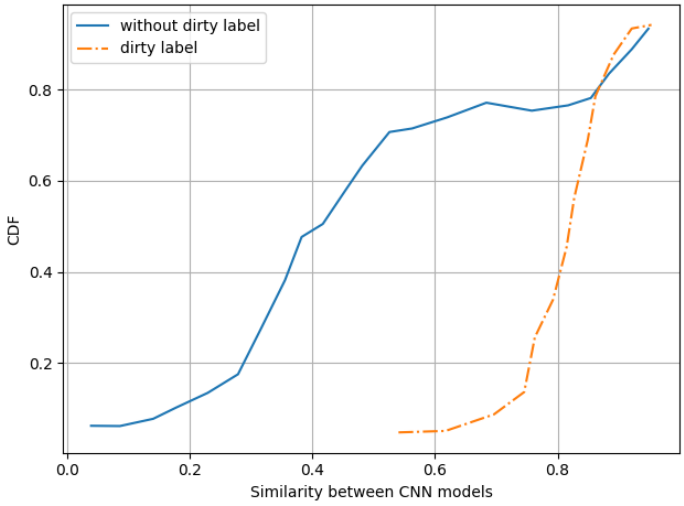
B. Kan value setting

In order to fully analyse the performance of ecFL in terms of model cleaning and improving the efficiency of FL communication, this paper conducted experiments and measured the similarity between the global and local parameters of the Soft max regression model, CNN model and MLP model during each iteration. Their cumulative distributions are illustrated in Figure 4. A comparison is made between the traditional federal learning approach using a dataset trained without the addition of noise labels and with the addition of noise labels. In the Soft max regression model, the similarity between the global model and the local model is close to 1.0 when trained with the noise-labelled dataset, whereas with the noise-labelled, the similarity between the two is distributed between 0.7 and 1.0, which is significantly lower than the result of training with the noise-labelled dataset. A similar situation was observed in the CNN model and the MLP model.

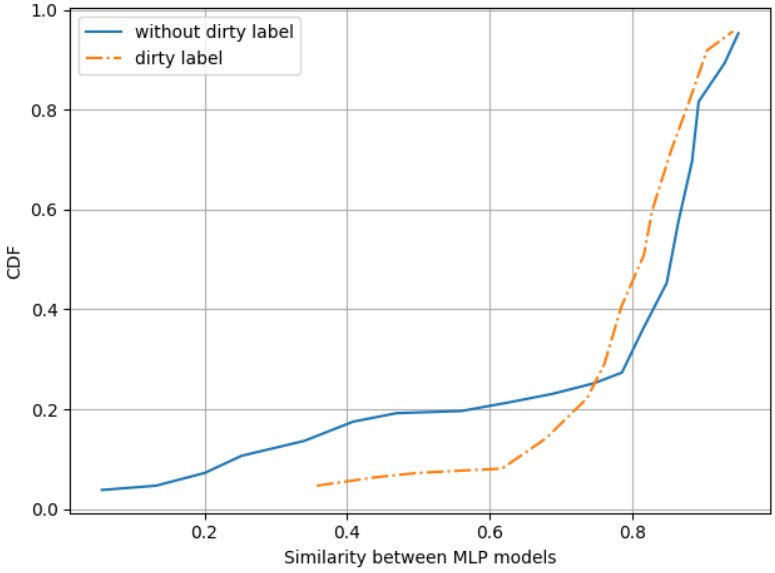
In order to verify that the ecFL algorithm can accurately remove dirty models in the presence of dirty models and achieve the training accuracy in the dirty model-free condition, a set of tests around specific thresholds is performed in this paper. In the Soft max regression model, tests were conducted around 0.97, with the set of thresholds {0.95,0.955,0.96,0.965,0.97,0.975,0.98,0.985} set; In the CNN model, the set of thresholds {0.9995,0.9996,0.9997,0.9998,0.9999} was set around 0.9998 for the test; in the MLP model, the set of thresholds {0.9990,0.9991,0.9992,0.9993,0.9994. 0.9995}. The experimental results showed that the ecFL algorithm was able to obtain the best performance when the threshold values were set to 0.9999, 0.9995 and 0.98 for the MNIST CNN model, MNIST MLP model and MNIST Soft max regression model, respectively. (Figure 5)



A softmax



B CNN



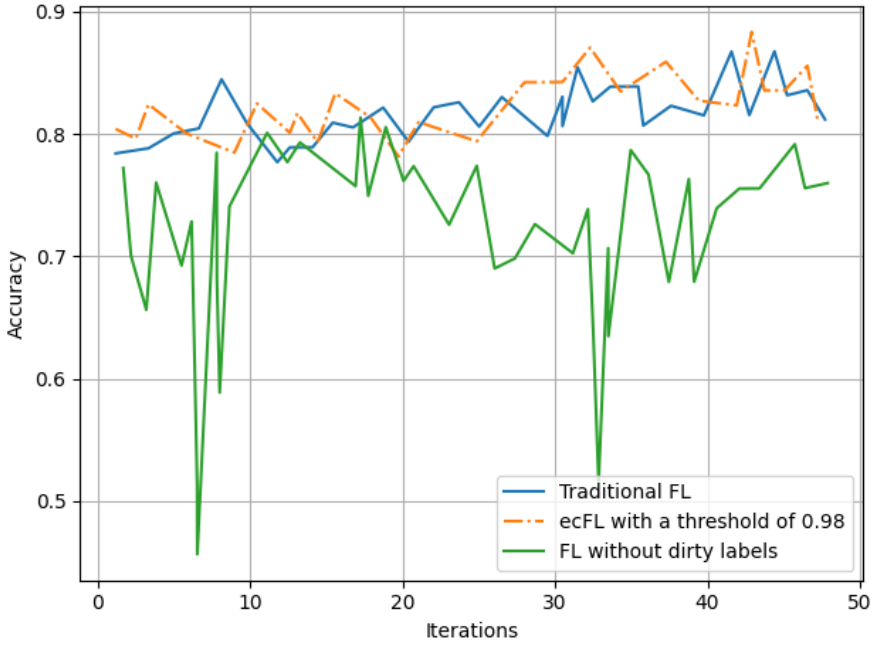
C MLP

Figure 5 Cumulative distribution of cosine similarity between global and local model parameters

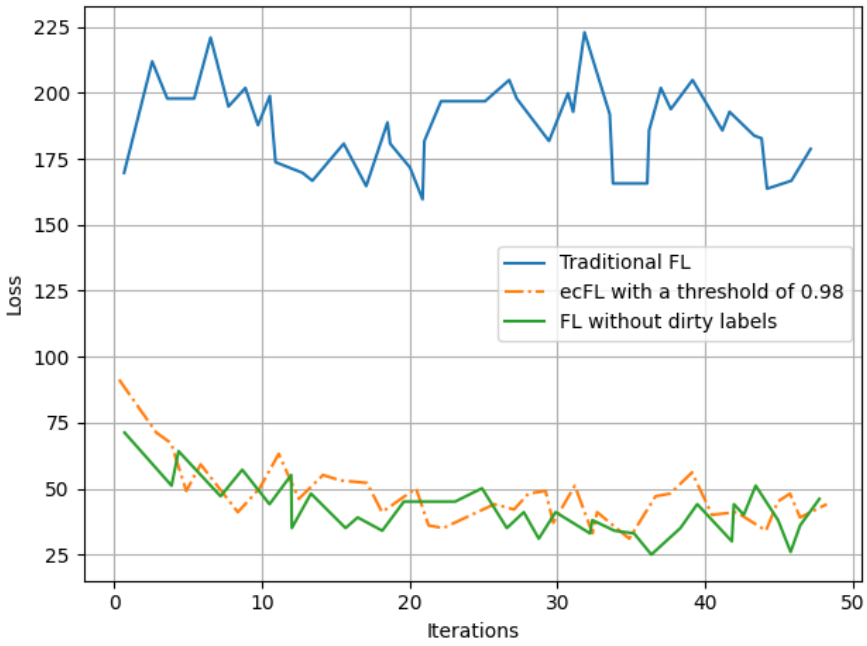
C. Results and analysis

A whole model was first trained on a dataset without dirty labels as a control in order to assess the effectiveness of this paper's method for cleaning models and lowering network footprint. The Soft max regression model then underwent experiments, with the outcomes being examined.

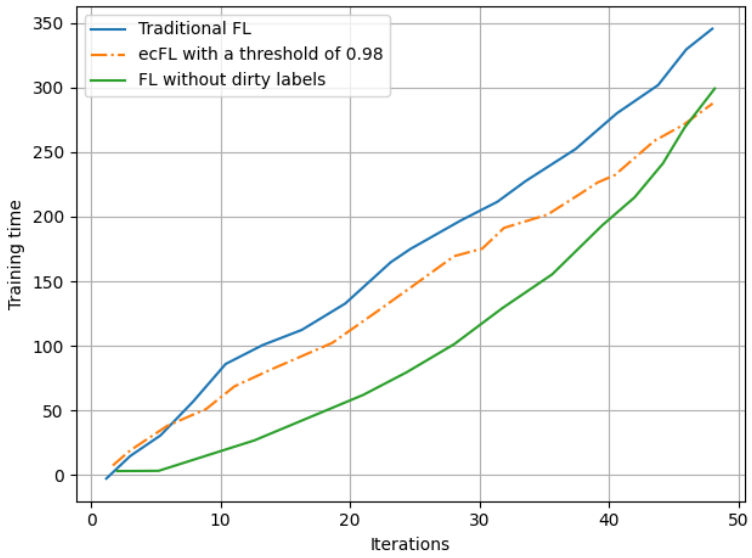
In the studies, the dirty tags are evenly distributed across 16 end devices, and each device is given a random set of LAN addresses. The model's learning rate was set to λ = 0.001. Figures 6(a) and 6(b) show the relationship between the accuracy and training loss of the model during the global iterations as measured in this study.



A The accuracy of softmax through different learning methods



B The loss value of softmax through different learning methods



C Training time for different learning methods of softmax

Figure 6 Performance of Soft max regression models learned by different means

Due to the high local training round epoch of 93 for the end device, the global model accuracy has reached above 0.8 at the first global aggregation of the server. Overall, ecFL was effective in model cleaning, with a steady improvement in model accuracy and a significant reduction in training loss. In contrast, model accuracy under the traditional FL approach without model cleaning was extremely unstable, with no significant downward trend in training loss. Both models trained under the ecFL approach had significantly lower loss values compared to models trained under the FL approach without dirty labels, and there was only a 1% decrease in accuracy, and both were significantly higher than models trained under the traditional FL approach without model cleaning.

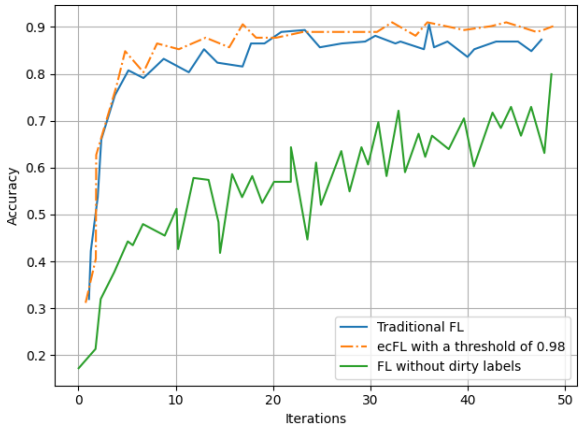
This research adds random delays to explain the transmission process of the model in the network based on the distance between various devices and the server and the status of the server devices, and plots them in Figure 6(c) in order to replicate the training period of FL. According to the experimental findings, the ideal training time for the same number of repetitions is achieved by the ecFL technique with device clustering and edge aggregation, whereas the training times for the conventional FL and the ideal FL without dirty labels are comparable. The proposed ecFL approach requires 10.3% less training time than the conventional FL [25].

D. Experimental analysis on CNN models

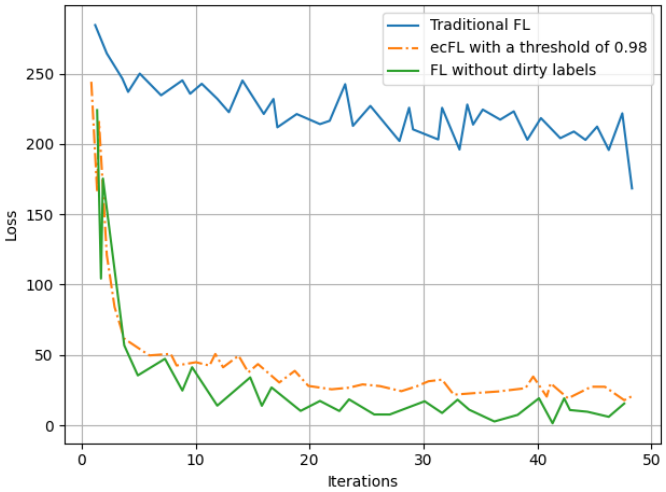
The CNN model was subjected to similar testing, with 8 malicious endpoints and a $\lambda = 0.001, droput = 0.5$ learning rate. This study evaluated how well the three learning techniques—conventional FL, ecFL, and FL without dirty labels—performed when used to train the Soft max regression model and the CNN model. With a data noise ratio of 0.7, the findings in Figures 7(a) and 7(b) show that the ecFL method of model accuracy is much superior to the traditional FL without model cleaning. After only 5 global iterations, ec FL reduces the model's training loss by 80%, hastening the model's convergence. The accuracy of the conventional FL technique without model cleaning is only 0.7 when looking at the graph vertically, under the identical conditions, after 50 global iterations, with just a 33% decrease in the loss value. The ecFL approach only needs two iterations to attain the same accuracy that the standard FL method does, but the traditional FL method without model cleaning needs thirteen [26,27].

Compared to the FL training method without dirty labels, the ecFL method was close in accuracy, with only a 2.1% drop. Furthermore, the training results of the conventional FL method without model cleaning were much worse than the results of the ecFL method with model cleaning. This also indirectly indicates that the removal of local model parts that do not match the global trend does not affect the correctness of the overall model results. At the

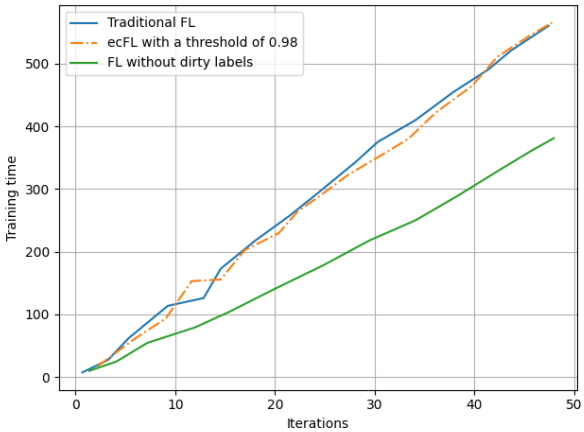
same time, this paper introduces a delay in the traditional FL model update process to simulate the network conditions in real scenarios. According to the experimental results in Figure 7(c), the training time of the ecFL method with device clustering and edge aggregation is 30.8% shorter than that of the conventional FL method for the same number of iterations.



A The accuracy of CNN through different learning methods



B The loss value of CNN through different learning methods



C Training time for different learning methods of CNN

Figure 7 Performance of CNN models learning by different means

E. Communication efficiency evaluation

In Table 1, a comparison of the performance of conventional FL, ecFL and FL without dirty labels in terms of reducing communication latency is summarized, where the number of communication iterations is 50 for all three. Overall, when the number of iteration rounds is the same, the communication time required for conventional FL and FL without dirty labels is similar, indicating that the presence of a dirty model has a greater impact on the model training time, but a smaller impact on the communication time. The proposed ecFL method requires significantly less communication time than conventional FL when the number of iterations is the same. this indicates that through device clustering and edge aggregation, the ecFL method is able to significantly reduce the time required for model uploading and downloading, thus effectively reducing the network latency in the federation learning process.

Table 1 Communication time consumed by different learning methods

	Traditional FL	e FL	FL without dirty labels
Soft max regression model	344.1	296.2	290.3
CNN model	616.8	411.7	620.1

In order to get a more intuitive view of the performance of the proposed ecFL in optimizing FL communication efficiency, a metric called $effect$ is designed in this paper. For a given learning accuracy, $effect$ is defined as the normalized value of the total number of local updates that need to be uploaded under conventional FL compared to the total number of local updates that need to be uploaded under ecFL mode.

$$effect = \frac{\sigma_t - \sigma_e}{\sigma_t} \quad (12)$$

Where σ_t denotes the total number of local updates to be uploaded without clearing the model, and σ_e denotes the total number of local updates to be uploaded in ecFL mode. Intuitively, the larger $effect$ is, the better ecFL performs in terms of improving communication efficiency. In this paper, we measure the $effect$ of ecFL in MNIST Soft max regression model and MNISTCNN model with different precision. The results of $effect$ for different model accuracies are given in Table 2.

Table 2 Results for $sffsct$ on different model accuracies

	σ_i	σ_e	$sffsct$
MNIST CNN model with an accuracy of 0.6	100	30	0.74
MNIST CNN model with an accuracy of 0.8	330	40	0.92
Soft max regression model with an accuracy of 0.85	10	18	0.7
Soft max regression model with an accuracy of 0.87	230	20	0.96

For the Soft max regression model, this paper investigates the impact of the ecFL learning approach for model accuracies of 0.85 and 0.87. At 0.85 accuracy, conventional FL requires 10 model updates to be uploaded, while ecFL only requires 18, which means that ecFL reduces the network footprint by 60%. When accuracy is increased

to 0.87, conventional FL requires 230 model updates to be uploaded while ecFL only requires 20. With model cleaning, ecFL eliminates 96% of local updates, significantly reducing communication costs.

For the CNN model, due to its slow convergence rate, this paper examines the performance of ecFL in improving the efficiency of federal learning communication at 0.6 and 0.8 precision. At 0.6 precision, ecFL reduces the number of local updates by 74% compared to conventional FL, which is equivalent to 90 fewer updates. When the precision was increased to 0.8, conventional FL required 330 updates to be uploaded, while ecFL only required 40 updates to be uploaded, meaning that ecFL reduced the number of model updates by 92%, significantly improving communication efficiency.

Based on the experimental results in Figure 8, an evaluation of the execution time of 1,000 tasks was performed without using the paper's scheme clustering method and with scheduling of the clustered tasks. The results show that the average execution time of the tasks on the terminal was 5.804 seconds without the use of the clustering method of this paper's scheme, whereas with the use of this method the average execution time of the tasks was reduced to 3.761 seconds, a reduction of approximately 35.2%.

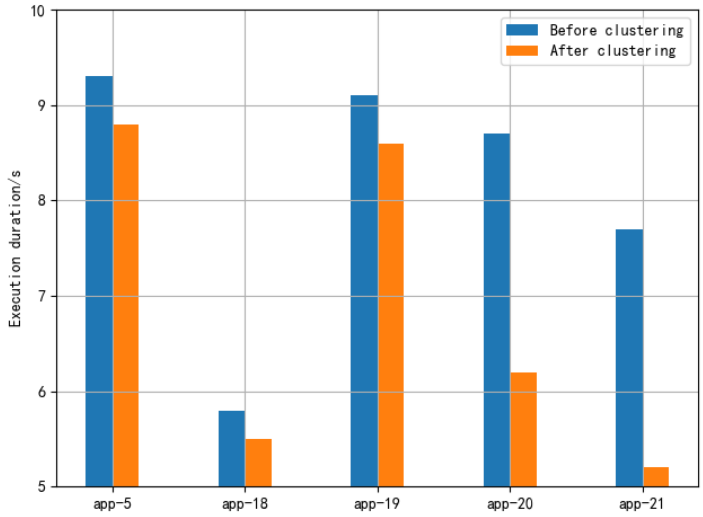


Figure 8 Comparison of task execution time before and after ecFL clustering

Figure 9 shows the task scheduling time before and after adding the task clustering algorithm for task numbers of 1,000, 3,000, 5,000, 7,000 and 10,000. As the number of tasks increases, the advantage of task scheduling via task clustering becomes increasingly apparent. This is because when the number of tasks is large, the resources that match the tasks can be found faster for scheduling when the tasks are clustered using this paper's scheme. By clustering tasks, the efficiency and accuracy of task scheduling can be improved, thus optimizing the performance of the whole system. The rescheduled tasks are executed at a more even time, reducing the waste of resources and load imbalance.

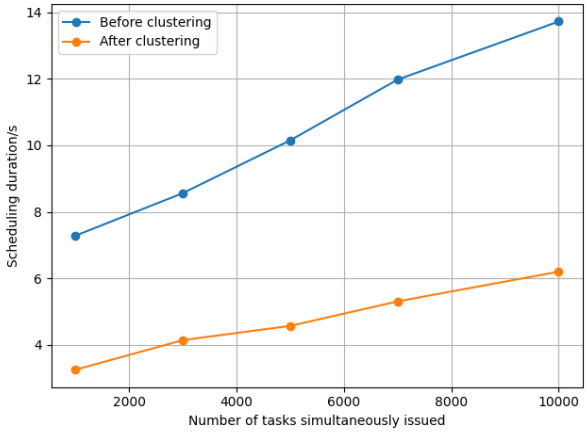


Figure 9 Time spent per 100 tasks scheduled before and after ecFL clustering

VI. Conclusion

In order to increase federation learning's communication effectiveness and model resilience, this research presents an edge-based federation learning framework dubbed ecFL. By using mobile edge nodes as the main nodes for cloud-edge communication, ecFL introduces the idea of clustering based on device network locations and overcomes the latency issue brought on by high server concurrency. Calculating the cosine similarity between the global model parameters is the key concept of ecFL. To ascertain whether the local update complies with the global model's convergence tendency, the cosine similarity between the parameters of the global model and the local update is computed. The mobile edge nodes will only gather and aggregate the local updates when the similarity rises beyond a predetermined threshold, accomplishing the model cleaning objective. In order to show the effectiveness of ecFL in terms of model cleaning and communication efficiency improvement, experiments on the two learning models Soft max regression and CNN are conducted in this study. With ecFL, the network footprint was effectively decreased by 95% compared to classical federated learning, and the convergence performance was improved to a 30.8% improvement over the CNN model. The use of federated learning in a distributed setting has been greatly enhanced by the introduction of the ecFL framework.

Acknowledgment

There is no specific funding to support this research.

References

[1] Pan, H. , Lei, Y. , & Yin, S. . (2021). K-means clustering algorithm for data distribution in cloud computing environment. International journal of grid and utility computing(3), 12.

[2] Fan, L. , Shen, L. , & Zuo, X. . (2021). Feature extraction and recognition of medical ct images based on mumford-shah model. Advances in Mathematical Physics(Pt.3), 2021.

[3] Harakannanavar, S. S. , Rudagi, J. M. , Puranikmath, V. I. , Siddiqua, A. , & Pramodhini, R. . (2022). Plant leaf disease detection using computer vision and machine learning algorithms. Global Transitions Proceedings, 3(1), 305-310.

[4] Shang, R. , Liu, M. , Lin, J. , Feng, J. , & Jiao, L. . (2021). Sar image segmentation based on constrained smoothing and hierarchical label correction. IEEE Transactions on Geoscience and Remote Sensing, PP(99), 1-16.

[5] Shan, P. , Li, Z. , Li, Y. , & Fu, W. . (2021). Spectral-spatial hyperspectral image classification using robust dual-stage spatial embedding. IEEE Access, PP(99), 1-1.

[6] Wang, H. , Tang, G. , Wu, K. , & Wang, J. . (2021). Plver: joint stable allocation and content replication for edge-assisted live video delivery. IEEE Transactions on Parallel and Distributed Systems, PP(99), 1-1.

[7] Wang, Z. , Wan, L. , Xiong, N. , Zhu, J. , & Ciampa, F. . (2021). Variational level set and fuzzy clustering for enhanced thermal image segmentation and damage assessment. NDT & E International, 118(8), 102396.

[8] Wang, J. C. F. . (2021). Variational level set and fuzzy clustering for enhanced thermal image segmentation and damage assessment. NDT & E International: Independent Nondestructive Testing and Evaluation, 118(1).

[9] Shuja, J. , Humayun, M. A. , Alasmary, W. , Sinky, H. , & Khan, M. K. . (2021). Resource efficient geo-textual hierarchical clustering framework for social iot applications. IEEE Sensors Journal, PP(99), 1-1.

[10] Sami, H. , Otrok, H. , Bentahar, J. , & Mourad, A. . (2021). Ai-based resource provisioning of ioe services in 6g: a deep reinforcement learning approach. IEEE Transactions on Network and Service Management, PP(99), 1-1.

[11] Zhang, C., Roh, B. H., & Shan, G. (2023, December). Poster: Dynamic Clustered Federated Framework for Multi-domain Network Anomaly Detection. In Companion of the 19th International Conference on emerging Networking EXperiments and Technologies (pp. 71-72).

[12] Xu, J. , Liu, L. , Zhang, R. , Xie, J. , & Shi, L. . (2021). Ifts: a location privacy protection method based on initial and final trajectory segments. IEEE Access, 9, 18112-18122.

[13] Uykan, Z. . (2021). Shadow-cuts minimization/maximization and complex hopfield neural networks. IEEE Transactions on Neural Networks and Learning Systems, 32(3), 1096-1109.

[14] Wang, F. , & Hu, H. . (2021). Coverage hole detection method of wireless sensor network based on clustering algorithm. Measurement, 179(5), 109449.

[15] Xu, M. , Fu, P. , Liu, B. , & Li, J. . (2021). Multi-stream attention-aware graph convolution network for video salient object detection. IEEE Transactions on Image Processing, PP(99), 1-1.

[16] Stephanie, V. , Chamikara, M. A. P. , Khalil, I. , & Atiquzzaman, M. . (2021). Privacy-preserving location data stream clustering on mobile edge computing and cloud. Information Systems(2), 101728.

[17] Sandoval, C. , Pirogova, E. , & Lech, M. . (2021). Adversarial learning approach to unsupervised labeling of fine art paintings. IEEE Access, PP(99), 1-1.

[18] Zhang, L. , Sadler, B. M. , Blum, R. S. , & Bhattacharya, S. . (2021). Inter-cluster transmission control using graph modal barriers. IEEE Transactions on Signal and Information Processing over Networks, PP(99), 1-1.

[19] Karaaslanl, A. , & Aviyente, S. . (2021). Community detection in dynamic networks: equivalence between stochastic blockmodels and evolutionary spectral clustering. IEEE Transactions on Signal and Information Processing over Networks, 7, 130-143.

[20] Feng, C. , Ye, J. , Hu, J. , & Yuan, H. L. . (2021). Community detection by node betweenness and similarity in complex network. Complexity, 2021(5), 1-13.

[21] Yang, L. , Lu, Y. , Cao, J. , Huang, J. , & Zhang, M. . (2021). E-tree learning: a novel decentralized model learning framework for edge ai. IEEE Internet of Things Journal, PP(99), 1-1.

[22] Yu, Y. . (2021). Application of mobile edge computing technology in civil aviation express marketing. Wireless Communications and Mobile Computing, 2021(3), 1-11.

[23] Guo, X. , Zhu, Y. , Zhang, J. , Hai, Y. , & Liu, S. . (2021). Intelligent pointer meter interconnection solution for data collection in farmlands. Computers and Electronics in Agriculture, 182(13), 105985.

[24] C. Zhang, M. Li and D. Wu, "Federated Multidomain Learning With Graph Ensemble Autoencoder GMM for Emotion Recognition," in IEEE Transactions on Intelligent Transportation Systems, vol. 24, no. 7, pp. 7631-7641, July 2023, doi: 10.1109/TITS.2022.3203800.