Edge-Based Communication Optimization for Distributed Federated Learning

Tian Wang, Yan Liu, Xi Zheng, Hong-Ning Dai, Weijia Jia, and Mande Xie

***Abstract*—Federated learning can achieve distributed machine learning without sharing privacy and sensitive data of end devices. However, high concurrent access to cloud servers increases the transmission delay of model updates. Some local models may be unnecessary with an opposite gradient from the global model, thus incurring many additional communication costs. Existing work mainly focuses on reducing communication rounds or cleaning local defect data, and neither takes into account latency associated with high server concurrency. To this end, we study an edge-based communication optimization framework to reduce the number of end devices directly connected to the parameter server while avoiding uploading unnecessary local updates. Specifically, we cluster devices in the same network location and deploy mobile edge nodes in different network locations to serve as hubs for cloud and end devices communications, thereby avoiding the latency associated with high server concurrency. Meanwhile, we propose a method based on cosine similarity to filter out unnecessary models, thus avoiding unnecessary communication. Experimental results show that compared with traditional federated learning, the proposed scheme reduces the number of local updates by 60%, and the convergence speed of the evaluated model increases by 10.3%.**

***Index Terms*—Federated learning, Communication optimization, Mobile edge nodes, Model filtering, Clustering.**

I. Introduction

MACHINE learning (ML) [1] has been successfully applied in a wealth of practical artificial intelligence (AI) applications in the field of computer vision, natural language processing, healthcare, finance, robotics and many others. These applications enhance the operational efficiency of the entire manufacturing process and generate huge amounts of data daily [2] [3]. As a result of industry competition and data privacy, data is not shared in most industries [4] [5]. Even in the same company, data integration between different departments is faced with massive resistance [6], not to mention integrating data from various agencies, which is almost impossible in reality. Besides, with the further development of big data, the emphasis on data privacy and security has become a worldwide trend. As the essential technology of AI, federated learning (FL) [7] [8] is a promising approach to resolve this challenge. End devices participating in federated learning only need to train the model locally and send the trained model to the cloud server for aggregation. In a word, FL can achieve machine learning under the condition of protecting data privacy.

However, the communication efficiency of FL still faces many challenges [9] [10]. On the one hand, the advanced ML applications deployed in end devices are increasingly using the complex neural network, so the local updates usually contain a large gradient vector. In contrast, the network between end devices and the central server usually exists two problems: a) The bandwidth of the network is limited, and high-bandwidth server services are costly; b) The asymmetric property of internet connections: the uplink is typically much slower than the downlink. Therefore, when a large number of end devices participate in FL, there is bound to be a large number of communication delays. On the other hand, the devices and data participating in FL have the problems of heterogeneity and Non-IID, respectively [11], so the local models trained by these devices and data might not be up to expectations. There may be updated parameters from unnecessary models. If unnecessary models are sent to the cloud for aggregation, it will not only seriously affect the accuracy of model training, but also increase the additional communication cost. Therefore, it is critical to minimize the number of end devices directly connected to the server while avoiding uploading unnecessary local updates, thus reducing the high communication costs of FL.

To this end, we discuss how to effectively leverage the computation and communication resources at the edge to obtain the best FL performance. We consider a typical mobile edge computing architecture in which mobile edge nodes [12] are interconnected with the remote cloud and end devices. End devices are divided into clusters based on their local area network (LAN) address. In the local update phase, each end device calculates the cosine similarity between the local

The above work was supported in part by grants from The Natural Science Foundation of Fujian Province of China (No. 2020J06023), The National Natural Science Foundation of China (NSFC) under Grant No. 61872154 and No. 61972352, UIC Start-up Research Fund under Grant No. R72021202 and the Key Research and Development Program of Zhejiang Province under Grant No. 2021C03150. (*Corresponding author: Mande Xie.*)

Tian Wang is with BNU-UIC Institute of Artificial Intelligence and Future Networks, Beijing Normal University (BNU Zhuhai), Zhuhai 519000, China, and also with the Guangdong Key Lab of AI and Multi-Modal Data Processing, BNU-HKBU United International College, Zhuhai 519000, China, and the College of Computer Science and Technology, Huaqiao University, Xiamen 361021, China (e-mail: cs_tianwang@163.com).

Yan Liu is with the College of Computer Science and Technology, Huaqiao University, Xiamen 361021, China (e-mail: lyliuxixi@163.com).

Xi Zheng is with the department of computing, Macquarie University, NSW, 2109, Australia (e-mail: james.zheng@mq.edu.au).

Hong-Ning Dai is with Faculty of Information Technology at Macau University of Science and Technology, Macau 519000, China (e-mail: hndai@ieee.org).

Weijia Jia is with BNU-UIC Institute of Artificial Intelligence and Future Networks, Beijing Normal University (BNU Zhuhai), Zhuhai 519000, China, and also with the Guangdong Key Lab of AI and Multi-Modal Data Processing, BNU-HKBU United International College, Zhuhai 519000, China (e-mail: weijiaj@gmail.com).

Mande Xie is with the School of Information and Electronic Engineering, Zhejiang Gongshang University, Hangzhou 310018, China (e-mail: xiemd@zjgsu.edu.cn).

model parameters and the global model parameters. If the similarity between the two is lower than a threshold, the local model is considered an unnecessary update, thereby avoiding additional communication costs. In the edge aggregation phase, the mobile edge nodes deployed in each LAN collect and aggregate necessary updates from local models and then send the aggregated model to the cloud server for global aggregation, thereby avoiding the latency associated with concurrent server connections. Each edge aggregation consumes computing resources of mobile edge nodes, and each global aggregation consumes network communication resources between edge nodes and the cloud server only.

Overall, our main contributions are as follow:

- We introduce a clustering method based on the network location of end devices. Mobile edge nodes are deployed in each cluster as communication hubs to decrease the latency associated with high server concurrency.
- We compare the differences in parameter values and convergence directions and use this difference as a metric to filter out unnecessary models, thus avoiding unnecessary communication costs.
- We validate our framework in two FL settings. Extensive experiments on the MNIST dataset demonstrate that our approach reduces the number of updates in the network by 60%, and the convergence speed of the models is accelerated by 10.3% compared to traditional FL.

The remainder of the paper is organized as follows: After presenting related work in Section II, we preliminary introduce federated learning and present the problem definition in Section III. The proposed Edge-based communication optimization method of federated learning is presented in Section IV. Experimental evaluation and conclusion are given in Section V and Section VI, respectively.

II. Related Work

The existing communication optimization work of FL includes two aspects of reducing communication and end device cleaning. We summarize the approaches along with references in Table I.

A. Reduce Communication

The existing work of reducing communication mainly realizes by increasing the computation on edge and algorithm optimization. It is proposed in [13] that focuses on increasing the computation on end devices or increasing the number of end devices to accelerate the convergence speed of the global model, while it ignores the delay caused by a large number of end devices concurrently accessing the server. Besides, a three-layer FL system is proposed in [14] that aggregates models on two layers of edge servers and cloud servers, which achieves a good compromise in communication calculations, but defaults to uploading all local models. As for algorithm optimization, considering that the data involved in FL is distributed on multiple end devices, a control algorithm is proposed in [15] [16] that determines the best trade-off between local update and global parameter aggregation under a given resource budget.

However, existing methods of reducing communication assume that all local updates (good or bad) must be uploaded to the cloud. They do not take into account the fact that some updates may be unnecessary or malicious.

B. End Device Cleaning

During the training process of FL, unnecessary models may be uploaded by end devices, leading to the high communication cost of FL. The dirty devices may perform erratic updates intentionally, such as the data poisoning attack, or produce low-quality data unintentionally. It is proposed in [17] for the first time that considers the existence of unreliable participants (i.e., participants with low data quality), and proposes a solution to reduce the impact of these participants while protecting their privacy. In response to this problem, each participant trains a local model with its data and only shares model parameters with other participants, and it uses a functional mechanism to perturb the objective function of the neural network in the training process to achieve differential privacy. Besides, It is proposed in [18] [19] that achieved reliable end devices selection scheme, but they focus more on safety issues during training. An approach proposed in [20] leverages a pre-trained anomaly detection model to detect abnormal client behaviors and eliminate their adverse impacts. It seems to speed up the training speed of FL. A clustered federated learning framework based on the pairwise cosine similarity clustering between parameter updates is proposed in [21], which excludes abnormal participants by dividing customers into different groups and focuses on the impact of bad local updates on global models. Also, another work [11] that focuses on data cleaning reduces the risk of poisoning the global model but ignores the communication efficiency.

Although end devices with better performance can speed up the training speed of federated learning, they may also affect the global model's accuracy due to insufficient training data.

C. Local Update Optimization

Different from most of the previously published methods that focused on the optimization of a single issue, it is proposed in [22] that takes into account three issues in federation learning, namely, the local client-side computation complexity, the communication cost, and the test accuracy. It develops an algorithm named Loss-based Adaptive Boosting FederatedAveraging (LoAdaBoost FedAvg), where the local models with high cross-entropy loss are further optimized before model averaging on the server. A work called communication-mitigated federated learning (CMFL) proposed in [23] that avoids irrelevant updates by checking whether local updates conform to the global trend, and ultimately reduces the number of accumulated communication rounds. However, its judgment method for the importance of local models is one-sided, and we also discuss this method later.

In contrast to the above research, our work focuses more on the communication delay caused by the high concurrency of the server and the additional communication cost caused by uploading unnecessary local updates. Our solution is to compare the differences between local update parameters and

TABLE I
SOLUTIONS TO COMMUNICATION OPTIMIZATION IN FL

Solutions	Ref.	Key Ideas	Limitations
Reduce Communication	[12]	Increase the computation on end devices before communication.	Not take into account the fact that some updates may be unnecessary or malicious.
	[13]	Asynchronous aggregation in the edge layer and cloud server.	
	[14][15]	Determine the best trade-off between local update and global parameter aggregation under a given resource budget.	
End Device Cleaning	[16]	Reduce the impact of unreliable participants while protecting privacy.	High-performance end devices may reduce the accuracy of the global model due to insufficient training data.
	[17][18]	End device trains with its data and only shares model parameters with other participants.	
	[19]	Reliable end devices selection scheme.	
	[20]	Propose an anomaly detection model to detect abnormal client.	
	[10]	Exclude abnormal participants by dividing customers into different groups.	
Local Update Optimization	[21]	The local model with large cross-entropy loss is further optimized before global aggregation.	The method of judging the importance of local models is one-sided.
	[22]	Avoid irrelevant updates by checking whether local updates conform to the global trend.	

global model parameters in multiple dimensions. Besides, we cluster devices with the same network location to improve the communication efficiency of learning. We will state our problem definition in the next section, and give solutions and experimental results in subsequent sections.

III. Preliminaries and Definitions

A. Federated Learning

We consider a horizontal FL [24] system consisting of a server and N end devices that use the FedAvg algorithm [25] to train a model collaboratively. Assume that each end device i ($i \in N$) has an local dataset D_i. For a sample data $\{x_j, y_j\}$ with input x_j, the task is to find the $model\ parameter\ w$ that characterizes y_j with the loss function $f_j(w)$. An example of the loss function is $f_j(w) = \frac{1}{2}(x_j^T w - y_j)$, $y_j \in$ R for linear regression. The loss function on the dataset of end device i is defined as

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

For convenience, we use $|D_i|$ to denote the size of the dataset and define the total size by $D = \sum_{i=1}^{N} D_i$. According to the FedAvg algorithm, the learning task of the server is to minimize the following global loss function

$$F(w) = \sum_{i=1}^{N} \frac{D_i}{D} F_i(w). \tag{2}$$

B. Model Parameters

Machine learning algorithms generally require T iterations to achieve convergence of the loss function. At the beginning of the training, the server initializes a global model parameter. In each global iteration, end device i takes multiple local iterations to calculate its local model parameters using its local training data D_i. The local model parameters of end device i are defined as

$$w_i^{(t)} = w_i^{(t-1)} - \lambda F_i(w), \tag{3}$$

where λ is the learning rate. Due to the inherent complexity of most machine learning models, (3) is often solved using gradient-descent techniques [26]. Then the local parameters $w_i^{(t)}$ are transmitted to the server. At the server, all local parameters

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

are aggregated to minimize its objective $F(w)$ in (2) with a global accuracy ε, and then broadcasts $w^{(t+1)}$ to all end devices for next iteration (i.e.,$\| \nabla F(w^{(t)}) \| \leq \varepsilon \leq \| \nabla F(w^{(t-1)}) \|$ [27]).

C. Problem Definition

The communication efficiency of FL is related to the transmission time (or what we call the communication time) when the model is uploaded from the local to the cloud and the total bits transferred between the local and the cloud. Therefore, our optimization goal will revolve around these two factors.

The communication time of FL mainly comes from the upload time of the local update and the download time of the global model. Since the downlink bandwidth of the network is much larger than the uplink bandwidth, the download time is negligible compared to upload time and thus is not considered in this work. We mentioned earlier that the network bandwidth of the server is limited, so the number of concurrent connections to the server is also limited. If all end devices communicate with the server directly, this may increase the energy consumption [28]. To verify this point of view, we do a simple theoretical analysis. Assume that the data transmission rate is inversely proportional to the number of devices in the network. Constrained by Shannon's theorem, the transmission rate of the model in the network is defined as

$$v = inverse(\mu) * B * \ln(1 + \frac{S}{N_p}), \tag{5}$$

where B is the uplink bandwidth, S/N_p is the signal to noise ratio and μ is the number of devices connected to the server, and $inverse(\mu)$ represent a function inversely proportional to μ. We assume that S/N_p is constant during the learning time of FL. Denote the model size of w_i by M. Because the dimensions of vectors w_i are fixed, we assume that model size is constant throughout the FL. Thus, the communication time of each end device in each global iteration is

$$T_{com} = \frac{M}{v} = \frac{M}{inverse(\mu) * B * \ln(1 + \frac{S}{N_p})}. \quad (6)$$

It is easy to see that T_{com} is monotonically decreasing in μ. Therefore, a natural question is how to efficiently utilize a given amount of network resources to maximize the communication efficiency of model training? Based on the above analysis, we can narrow the problem down to reducing the number of devices (μ) connected to the server.

In addition, in each global iteration, each end device minimizes its objective $w_i^{(t)}$ in (3) using local training data. Due to the heterogeneity of end devices and the non-IID of local data, the local models vary greatly [23]. The gradient of some models may be in the opposite direction of the global model. In other words, in the model aggregation phase, some local models with unsatisfactory training effects can not only fail to accelerate the convergence of the model but also occupy a large amount of network transmission resources. Therefore, it is desirable to filter the unnecessary local model from the end devices to cut the additional communication cost.

Formally, we define δ_i as the bits uploaded from the end device i in the t^{th} iteration. For a given number of end devices N, the total bits uploaded from local in the t^{th} iteration is then $\Upsilon_t = \sum_{i=1}^{N} \delta_i$. Let w^* denote the target global model aprameter. We seek the solution to the following problem:

$$\begin{aligned} \min \quad & \sum_{t=1}^{T} \Upsilon_t, \\ s.t. \quad & w^* = \arg\min F(w). \end{aligned} \quad (7)$$

IV. Edge-based Model filtering and device clustering of Federated Learning

We present our solution to improve the communication efficiency of FL, which we call *Edge-based Communication Optimization for Distributed Federated Learning (eFL)*. For the problems raised in the previous section, we have given corresponding solutions.

A. Model filtering

Models that are independent of the convergence of the global model are called malicious local models or dirty models. During transmission, we refer to this as unnecessary models. Our purpose is to detect unnecessary models and avoid uploading unnecessary models, thereby reducing unnecessary communication costs. In related work, we mentioned that the CMFL determines the importance of local updates by calculating the number of identical symbolic parameters between global and local updates, i.e., any updates satisfying $\frac{1}{N}\sum_{j=1}^{N} I(sgn(u_j) = sgn(\bar{u}_j)) < threshold$ are considered insignificant and will not be uploaded, where $\bar{u}_j$ is the parameter of the global model in the last iteration, and u_j is the parameter of the local model in the current iteration.

Intuitively, though the sign of parameter in an update determines the direction (increase or decrease) to which the model should be improved along the dimension of that parameter, the value of the parameter also reflects how much the parameter changes in each direction. For example, in a typical softmax regression model, the value of the model parameters can be understood as the softmax probability value of each category, so the corresponding parameter values between the local update and the global model should be similar. If the signs of the corresponding parameters are the same, but the values are very different, unquestionable, we think that the two model parameters are not related. A natural question is: is there any other way to determine the correlation between global and local updates?

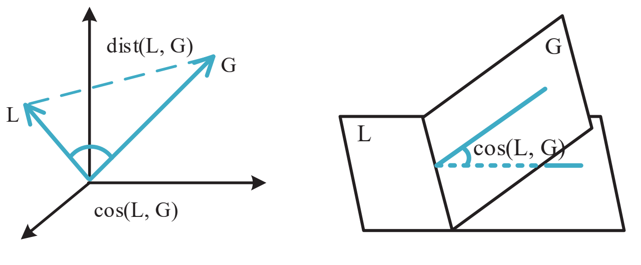


Fig. 1. Illustration of cosine similarity between model parameters in different dimensions. Low-dimensional model parameters are represented as vectors, and high-dimensional model parameters are represented as planes.

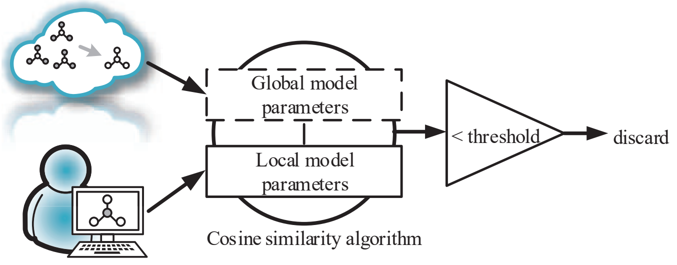


Fig. 2. The workflow of model filtering.

We turn our attention to machine learning and edge computing, which are the basis of federated learning. Euclidean distance and cosine similarity algorithms are commonly used in machine learning to judge similarity. Euclidean distance measures the absolute distance of each point in space, which is directly related to the coordinates of each point, while cosine distance measures the included angle of the space vector, which is more reflected in the difference in direction. Considering that the model parameters imply the convergence direction of the model, the cosine similarity is more suitable for detecting the unnecessary model than the Euclidean distance.

Coincidentally, we also found an angle-based anomaly detection method (ABOD) in a data cleaning algorithm based on edge computing [29]. ABOD contains a set of points to form a cluster, for each point o, ABOD examines the angle $\angle xoy$ for each pair of points x, y such that $x \neq o$ and $y \neq o$, then according to the variance of the angle at a point to determine whether the data is an outlier. The discovery in edge computing confirms our conjecture that the cosine similarity focusing on angle change is more suitable as a model filtering algorithm.

To eliminate local updates that are not related to global convergence, our first task is to determine the trend of the global model. Since the local training is based on the global model parameters of the previous iteration, end devices cannot know the global trend of the current iteration before global aggregation. Fortunately, the difference between two consecutive global updates verified in [23] is about 0.05, and the maximum does not exceed 0.21, so the global model parameters of the previous iteration can be used as a good evaluation of the new update. So far, we have transformed the model filtering problem into the problem of calculating the cosine similarity between the local update parameters in the current iteration and the global model parameters in the previous iteration.

In order to better understand the model parameters in the geometric space, we abstract the model parameters into two-dimensional vectors and higher-dimensional space planes, as shown in Fig. 1. We use G for global model parameters and L for local model parameters. As can be seen from the figure, the cosine distance is more reflected in the difference in direction than in position. If the position of L is kept unchanged, and the parameters of G change in each dimension, then the cosine distance is changing at this time (because the angle has changed).

To facilitate calculation, all end devices participating in FL need to vectorize the local update parameters and the global model parameters before model filtering. Assuming that the current local update parameters trained by the device are $L_t = [l_1, l_2, ..., l_s]$, and the global update parameters in the previous iteration are $G_{t-1} = [g_1, g_2, ..., g_s]$, then their cosine distance can be calculated as follows:

$$\begin{aligned} Similarity_{(L_t,G_{t-1})} &= cos(L_t, G_{t-1}) \\ &= \frac{\langle L_t, G_{t-1} \rangle}{|L_t||G_{t-1}|} \\ &= \frac{\sum_{j=1}^{s}(l_j \times g_j)}{\sqrt{\sum_{j=1}^{s}(l_j)^2} \times \sqrt{\sum_{j=1}^{s}(g_j)^2}}, \end{aligned} \tag{8}$$

where $\langle \, , \, \rangle$ is the scalar product operator.

According to the trigonometric function theorem, the range of the cosine value is between [-1,1]. The more cosine approaches 1, the closer the two vectors are; The more cosine goes to -1, the more they go in the opposite direction; When cosine is close to 0, it means that these two vectors are almost orthogonal. In general, it is not consistent with our reading habits if the similarity is less than 0, so the similarity is usually normalized within the range of [0,1]. That is, $Similarity_{(L_t,G_{t-1})} = 0.5 \times cos(L_t, G_{t-1}) + 0.5$. Clearly, when $Similarity_{(L_t,G_{t-1})}$ approaches 0, it indicates that the effectiveness of local model is higher. Conversely, the effectiveness of the local model is lower.

The model parameter is set to NULL only when the $Similarity_{(L_t,G_{t-1})}$ of the local model is less than the threshold, indicating that the model update is unnecessary and will not participate in the subsequent calculation. Algorithm 1 details the local training process in function LOCAL UPDATE. Fig. 2 illustrates the workflow of filtering out unnecessary models in federated learning. The setting of the threshold will be described in detail in the experimental section.

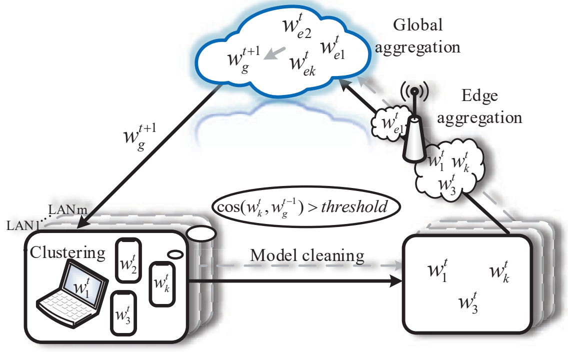


Fig. 3. The architecture of edge-based communication optimization for distributed federated learning.

B. Clustering at the Edge

Compared with LAN bandwidth, WAN bandwidth is a very scarce resource. Recent work shows that WAN bandwidth between network sites is, on average, 15 times smaller than LAN bandwidth within a site, and in the worst case is 60 times smaller. It confirms the scarcity of WAN bandwidth between different regions and gives us another piece of information: LAN bandwidth is much greater than WAN bandwidth.

Taking advantage of LAN bandwidth that is much greater than WAN bandwidth, we consider having end devices communicate as much as possible within the LAN. Therefore, we plan to realize the joint communication of FL in LANs and WANs. In short, it is the idea of clustering, which enables end devices to form clusters in a way and then communicate with the cloud server in clusters.

Our first option is to select an end device in the LAN that acts as a cluster head. The inspiration comes from the traditional wireless sensor network (WSN) [30] [31], in which the cluster head is responsible for collecting the information collected by the sensor and sending it to the sink node. From this, we envision deploying cluster head devices to collect all local updates on the LAN and send them to the cloud server for global aggregation. To verify if our insight holds in practice, we do a theoretical analysis. Suppose there are p end devices in a LAN, and the local update size at the end device is k, then the total bits that the cluster head needs to transmit to the cloud are $p \times k$. In other words, the size of the data packet communicated between a single device and the server has changed from k to pk. According to the definition of communication time in the foregoing, $T_{com} = \frac{M}{v}$, we can see that the greater the M, the greater the communication time T_{com}. Therefore, the cluster head device can not reduce the training communication time but increase the transmission delay.

For the issue that existed in the first option, our second option considers deploying mobile edge nodes to collect local updates by using their computing resources in the LAN. Mobile edge computing (MEC) [32] [33] is a network architecture that provides services and cloud computing resources required by users on the edge side, which can accelerate the rapid download of various applications in the network. In the previous study, the mobile edge node was proposed, and a

Algorithm 1 Edge-based communication optimization of federated learning(eFL)

```
1:  procedure GLOBAL AGGREGATION
2:      Input: Devices set $N$, LAN address $A_i$ of device $i$.
3:      Initialize: global model parameters $G_0$.
4:      for each iteration $t = 1, 2, ...$ do
5:          for all device $i \in N$ do
6:              $cluster_m \leftarrow \{i \mid A_i = A_m\}$;
7:          end for
8:          $G^t \leftarrow \{\}$;
9:          for all $cluster_m$ do in parallel
10:             $G^t \leftarrow$ EDGE AGGREGATION($cluster_m$, $G_{t-1}$);
11:         end for
12:         $G_t \leftarrow \frac{1}{|G^t|} \sum_{C_{m,t} \in G^t} C_{m,t}$;
13:     end for
14: end procedure
15: function EDGE AGGREGATION($cluster_m$, $G_{t-1}$)
16:     for all device $i \in cluster_m$ do in parallel
17:         $L_i \leftarrow$ LOCAL UPDATE($i$, $G_{t-1}$);
18:         $C^m \leftarrow \{L_i \mid L_i$ is not NULL$\}$;
19:     end for
20:     $C_{m,t} \leftarrow \frac{1}{|C^m|} \sum_{L_i \in C^m} L_i$;
21:     return $C_{m,t}$
22: end function
23: function LOCAL UPDATE($i$, $G_{t-1}$)
24:     Split local dataset $D_i$ to minibatch set $B_i$.
25:     for each local epoch $j = 1, 2, ...$ do
26:         for each $b \in B_i$ do
27:             $L_i \leftarrow L_i - \lambda \nabla F(L_i)$;
28:         end for
29:     end for
30:     if $Similarity_{(L_i, G_{t-1})} < f$ then
31:         $L_i \leftarrow$NULL;
32:     end if
33:     return $L_i$
34: end function
```

moving route was designed for the edge node to maximize the throughput and minimize the transmission latency [34]. Therefore, we can use the computing resources of mobile edge nodes to solve the problems mentioned above.

Assuming that the LAN address of end device i is A_i, then the device clustering process can be represented as follows

$$cluster_m = \{i \mid A_i = A_m\}. \tag{9}$$

That is, according to the different LAN where the devices are located, they can be divided into multiple clusters, and each cluster is independent of the other. Algorithm 1 details the clustering process in procedure GLOBAL AGGREGATION.

In each global iteration, local models from the same cluster are uniformly uploaded to the cloud by mobile edge nodes. Also, considering that the mobile edge node has computing resources, we use its computing resources to do edge aggregation of local models in the cluster. Therefore, each mobile edge node will get a cluster model after edge aggregation. The cluster model of $cluster_m$ is defined as

$$F_m(w) = \sum_{i \in cluster_m} \frac{D_i}{D_m} f_i(w), \tag{10}$$

where $D_m = \sum_{i \in cluster_m} D_i$. After this edge aggregation, all the cluster models of LAN will be sent to the cloud for aggregation to minimize the global loss function, then the next round of training iteration is performed. Algorithm 1 details the edge aggregation process in procedure EDGE AGGREGATION. Fig. 3 details the architecture of eFL.

C. Algorithm Analysis

Assuming that the target global model parameters are w^*, and the global model parameters we finally trained are w. Therefore, we can calculate the loss value of federated learning as $L(w^*) = |F(w) - F(w^*)|$, where $F(.)$ represents the global loss function. As the number of training iterations (T) approaches infinity, the time complexity of Algorithm 1 is limited to $\lim_{T \to +\infty} L(w^*) = \frac{1}{T}[O(\frac{1}{\eta T}) + O(\sum_{t=1}^{T} sh_t)]$, where η is the learning rate and sh represents the threshold of model filtering. Since sh is constant and T can be reduced after filtering out unnecessary models, so Algorithm 1 maintains the time complexity similar to that of traditional federated learning in communication.

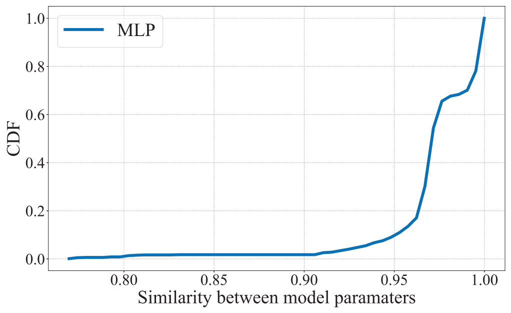
V. EXPERIMENTAL EVALUATION

A. Dataset and Preparation

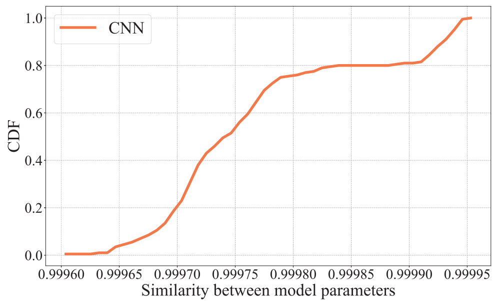
1) Dataset and models: We evaluate the training of two different models on the MNIST dataset. The models include multilayer perceptron (MLP) and convolutional neural networks (CNN).

- *MNIST:* MNIST [35] is a handwritten digits dataset, in which the training set contains 60,000 samples, and the test set contains 10,000 samples. Each image in the MNIST dataset is composed of 28 x 28 pixels, and each pixel is represented by a gray value.
- *MLP:* MLP is a forward-structured artificial neural network that maps a set of input vectors to a set of output vectors. In our experimental setup, the MLP consists of an input layer, a hidden layer, and an output layer (softmax layer). The number of neurons in the hidden layer is 300, and ReLU is used as the activation function.
- *CNN:* The structure of the CNN [36] [37] model is as follows: input layer, convolutional layer, pooling layer (max pooling), convolutional layer, pooling layer (max pooling), fully connected layer, output layer (softmax layer), where the size of the convolution kernel is 5*5 and the active function is ReLU.

2) Preparation: In our experiments, the model architectures are built upon TensorFlow. We train two models: MNIST CNN and MNIST MLP. For the MNIST MLP model, We divide the MNIST training samples evenly among 20 end devices, and each end device gets about 2750 samples. Besides, we set the number of local epochs each end device makes over its local dataset on each iteration as 87 and the local minibatch size used for the local updates as 32. Similarly, the number of end



(a) Similarity between model parameters in MLP.



(b) Similarity between model parameters in CNN.

Fig. 4. Distribution of similarity.

devices and local epochs is set as 10 and 15 for the MNIST CNN model, respectively.

We simulate the dirty data by modifying the labels in the dataset, and it is shown to have a significant impact on the training of FL. Research showed that only when the proportion of dirty labels is greater than 0.7 can the model accuracy be affected. Through experiments, the author proved that when the noise ratio of the dataset is less than 0.7, the accuracy of the CNN model can reach as high as 85%. In contrast, when the noise ratio is greater than 0.7, the accuracy of the model is reduced by 30%. Therefore, to reflect the eFL algorithm's effectiveness in filtering out dirty models, we modify more than 70% sample labels in the MNIST dataset to simulate noise in the FL environment.

B. Threshold Settings

In order to do a thorough analysis of eFL despite the influence of the threshold, we measure the similarity between global model parameters and local model parameters in each iteration and depict their CDF distributions in Fig. 4. Since we add 70% noise in the experiment, we convince that local models with low similarity to the global model should account for 70% and local models with high similarity to the global model should account for 30%. As shown in the results in Fig. 4, the slope of the solid line in the figure changes suddenly around CDF=0.7. In the MLP experiment, the local model whose similarity to the global model is less than 0.97 accounts for 70%. In contrast, in the CNN experiment, the local model whose similarity to the global model is less than 0.9998 accounts for 70%. The reason for this result is related to the noise ratio we added of 0.7, so we test a set of 8 relevance threshold values around 0.97 for the eFL on MLP model: $\{0.95, 0.955, 0.96, 0.965, 0.97, 0.975, 0.98, 0.985\}$, and another set of 5 relevance threshold values around 0.9999 on CNN model: $\{0.9995, 0.9996, 0.9997, 0.9998, 0.9999\}$. The result shows that the best performance is obtained when setting the relevance threshold value as 0.9999 for the MNIST CNN model and 0.98 for the MNIST MLP model.

C. Results and Analysis

We set up three sets of experiments: traditional FL, eFL, and FL without dirty labels.

1) Evaluation of the MNIST MLP: We evenly distribute the dirty labels among 16 end devices, and we randomly match a set of LAN addresses to all devices. The Adam optimizer is used with the learning rate $\lambda = 0.001$. We measure the validation accuracy and training loss over iteration and depict their relationship in Fig. 5(a) and Fig. 5(b).

Since the regression model converges quickly and the number of local epochs of the end device is as high as 87, the accuracy of the global model has reached 0.8 or more after the first aggregation on the server. After the second iteration, the accuracy of the model trained under eFL we mentioned rose steadily and remained above 0.9, while the accuracy of the model trained under traditional FL is extremely unstable, even falling below 0.5 in the 6^{th} iteration.

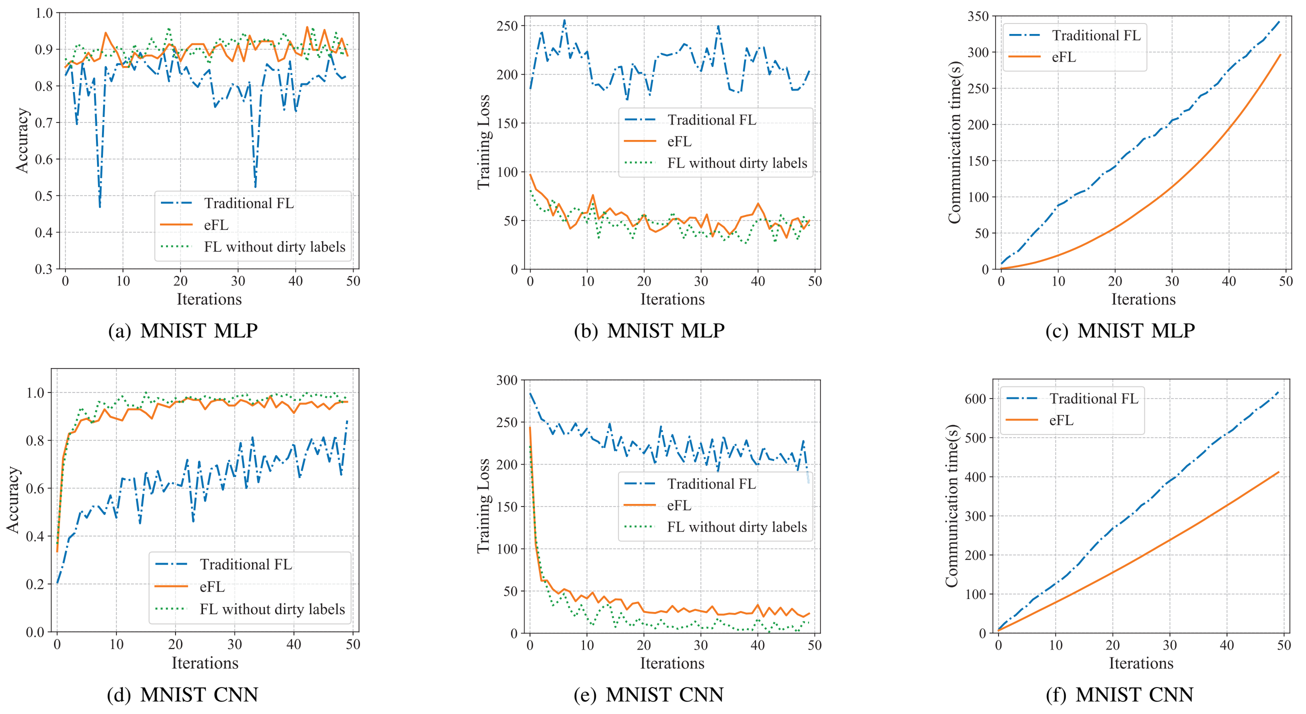
In comparison, the model trained under eFL and the model trained under FL without dirty labels are close in accuracy, and both are significantly higher than traditional FL. Similarly, the model trained under eFL and the model trained under FL without dirty labels have similar training loss, and both are significantly lower than traditional FL.

We randomly add delays to the network according to the distance between different devices and the server and the status of the server equipment, thus measuring the communication time of FL and showing it in Fig. 5(c). From iteration 0 to iteration 50, the communication time of federated learning under eFL is significantly lower than that under traditional FL. The results show that the communication time of eFL is reduced by 10.3% under the same number of iterations.

2) Evaluation of the MNIST CNN: A similar experiment is performed on the CNN model, where the dirty labels are evenly distributed to 8 end devices. The Adam optimizer is used with the learning rate $\lambda = 0.001$ and the ratio for dropout $keep\_prob = 0.5$.

As shown in Fig. 5(d) and Fig. 5(e), under the uniform condition of a data noise ratio of 0.7, the model accuracy trained under eFL is much higher than that trained under traditional FL. Besides, the convergence rate of the model trained under eFL is very fast, with the training loss of the model reduced by 80% after only about 5 global iterations.

From the vertical perspective of the figure, due to the influence of unnecessary models, the accuracy of the model trained under traditional FL is only 0.7 with the same iteration times, and the loss value only decreases by 33%. From the horizontal perspective of the figure, the learning trained under traditional FL needs 13 iterations when the accuracy of the global model reaches 0.6, while the eFL needs only 2 iterations to achieve the same accuracy.



(a) MNIST MLP (b) MNIST MLP (c) MNIST MLP

(d) MNIST CNN (e) MNIST CNN (f) MNIST CNN

Fig. 5. The performance of eFL compared to traditional FL and FL without dirty labels on MNIST MLP and MNIST CNN.

Compared with the model trained under FL without dirty labels, the model trained under eFL is close to it in accuracy, and the convergence of the model is slightly lower. However, the model trained under traditional FL is far less accurate than the two methods, and the convergence is not satisfactory.

Similarly, we add delays in the model update process to simulate the network conditions in the real scene. Similar to the evaluation of MLP, the communication time of CNN trained under eFL is much lower than that of traditional FL. The experimental results in Fig. 5(f) show that under the same number of iterations, the communication time of learning trained under eFL is reduced by 30.8% compared with traditional FL.

3) Communication Efficiency Assessment: Table II shows the communication time for 50 training rounds of traditional FL and eFL. Overall, the communication time required by the proposed eFL is far less than that of traditional FL when iterating the same number of rounds. That is, after device clustering and edge aggregation, the time needed for model uploading and downloading is significantly reduced.

To observe the performance of our proposed eFL in optimizing FL communication more intuitively, we define communication-saving as, for a given learning accuracy, as the total number of local updates that need to be uploaded under traditional FL normalized by that under eFL, i.e.,

$$communication\ saving = \frac{\varrho_t - \varrho_e}{\varrho_t}, \tag{11}$$

where ϱ_t represents the total number of local updates that need to be uploaded by traditional FL, and ϱ_e represents that number under eFL. Intuitively, the greater $communication\ saving$ is the better performance of eFL in reducing communication. We measure the $communication\ saving$ of eFL in MNIST MLP and MNIST CNN under different accuracies. Table III gives a statistical summary of the $communication\ saving$ when reaching the target accuracy.

TABLE II
COMMUNICATION TIME CONSUMED BY DIFFERENT LEARNING METHODS.

	Traditional FL	eFL
MNIST MLP (seconds)	344.0	296.1
MNIST CNN (seconds)	616	411.6

TABLE III
SUMMARY OF *communication saving* FOR DIFFERENT LEARNING ACCURACIES IN MNIST MLP AND MNIST CNN.

	ϱ_t	ϱ_e	$communication\ saving$
MNIST CNN 60% accuracy	110	20	0.73
MNIST CNN 80% accuracy	340	30	0.91
MNIST MLP 85% accuracy	20	8	0.6
MNIST MLP 87% accuracy	240	10	0.95

For MLP, when the model's accuracy rises to 0.85, the number of models that need to be uploaded under traditional FL is 20 (including some unnecessary models), and eFL reduces this number to 8, saving 60% of the communication. More surprisingly, when the accuracy rises to 0.87, the number of models that need to be uploaded under traditional FL is as high as 240, and eFL reduces this number to 10. Furthermore, for CNN, when the accuracy raises to 0.6, eFL reduced the number of model updates by 73%, and when the accuracy increases to 0.8, eFL reduced the number of model updates by 91%. Such experimental results undoubtedly prove the significant effect of eFL in improving communication efficiency.

VI. Conclusion

In this paper, we propose an edge-based federated learning framework (eFL), which aims to improve the communication efficiency of FL and strengthen the robustness of models. eFL introduces the idea of clustering based on devices' network location. It deploys mobile edge nodes in each cluster as the hub between cloud and edge communication. At the same time, to avoid extra communication caused by uploading unnecessary models, eFL calculates the similarity between the local model parameters and the global model parameters, only the local update whose value of similarity is greater than the set threshold will be collected by mobile edge nodes and then participate in edge aggregation. Experiments in MLP and CNN two learning models verify the effectiveness of eFL in filtering out unnecessary models and improving learning efficiency. Compared with traditional federated learning, eFL reduces network footprint by 95%, and the convergence speed of the CNN model accelerates by 30.8%.

References

[1] H. Hu, D. Wang, and C. Wu, "Distributed machine learning through heterogeneous edge systems." in *proceeding of the AAAI Conference on Artificial Intelligence*, 2020, pp. 7179–7186.

[2] M. Hao, H. Li, X. Luo, G. Xu, H. Yang, and S. Liu, "Efficient and privacy-enhanced federated learning for industrial artificial intelligence," *IEEE Transactions on Industrial Informatics*, vol. 16, no. 10, pp. 6532–6542, 2020.

[3] T. Wang, Y. Lu, J. Wang, H.-N. Dai, X. Zheng, and W. Jia, "Eihdp: Edge-intelligent hierarchical dynamic pricing based on cloud-edge-client collaboration for iot systems," *IEEE Transactions on Computers*, 2021, DOI: 10.1109/TC.2021.3060484.

[4] P. Kairouz, H. B. McMahan, B. Avent, A. Bellet, M. Bennis, A. N. Bhagoji, K. Bonawitz, Z. Charles, G. Cormode, R. Cummings *et al.*, "Advances and open problems in federated learning," *arXiv preprint arXiv:1912.04977*, 2019.

[5] T. Wang, Y. Mei, X. Liu, J. Wang, H.-N. Dai, and Z. Wang, "Edge-based auditing method for data security in resource-constrained internet of things," *Journal of Systems Architecture*, 2020, DOI: 10.1016/j.sysarc.2020.101971.

[6] J. Wang, Y. Yang, T. Wang, R. S. Sherratt, and J. Zhang, "Big data service architecture: a survey," *Journal of Internet Technology*, vol. 21, no. 2, pp. 393–405, 2020.

[7] B. McMahan and D. Ramage, "Federated learning: Collaborative machine learning without centralized training data," *Google Research Blog*, vol. 3, 2017.

[8] Q. Yang, Y. Liu, T. Chen, and Y. Tong, "Federated machine learning: Concept and applications," *ACM Transactions on Intelligent Systems and Technology (TIST)*, vol. 10, no. 2, pp. 1–19, 2019.

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

[10] T. Li, A. K. Sahu, A. Talwalkar, and V. Smith, "Federated learning: Challenges, methods, and future directions," *IEEE Signal Processing Magazine*, vol. 37, no. 3, pp. 50–60, 2020.

[11] Y. Han and X. Zhang, "Robust federated learning via collaborative machine teaching." in *proceeding of the AAAI Conference on Artificial Intelligence*, 2020, pp. 4075–4082.

[12] Y. Wu, H. Huang, Q. Wu, A. Liu, and T. Wang, "A risk defense method based on microscopic state prediction with partial information observations in social networks," *Journal of Parallel and Distributed Computing*, vol. 131, pp. 189–199, 2019.

[13] B. McMahan, E. Moore, D. Ramage, S. Hampson, and B. A. y Arcas, "Communication-efficient learning of deep networks from decentralized data," in *proceeding of the Artificial Intelligence and Statistics*. PMLR, 2017, pp. 1273–1282.

[14] L. Liu, J. Zhang, S. Song, and K. B. Letaief, "Edge-assisted hierarchical federated learning with non-iid data," *arXiv preprint arXiv:1905.06641*, 2019.

[15] S. Wang, T. Tuor, T. Salonidis, K. K. Leung, C. Makaya, T. He, and K. Chan, "Adaptive federated learning in resource constrained edge computing systems," *IEEE Journal on Selected Areas in Communications*, vol. 37, no. 6, pp. 1205–1221, 2019.

[16] S. Wang, T. Tuor, T. Salonidis, K. K. Leung, C. Makaya, T. He, and K. Chan, "When edge meets learning: Adaptive control for resource-constrained distributed machine learning," in *proceeding of the IEEE Conference on Computer Communications*, 2018, pp. 63–71.

[17] L. Zhao, Q. Wang, Q. Zou, Y. Zhang, and Y. Chen, "Privacy-preserving collaborative deep learning with unreliable participants," *IEEE Transactions on Information Forensics and Security*, vol. 15, pp. 1486–1500, 2019.

[18] J. Kang, Z. Xiong, D. Niyato, Y. Zou, Y. Zhang, and M. Guizani, "Reliable federated learning for mobile networks," *IEEE Wireless Communications*, vol. 27, no. 2, pp. 72–80, 2020.

[19] T. Nishio and R. Yonetani, "Client selection for federated learning with heterogeneous resources in mobile edge," in *proceeding of the IEEE International Conference on Communications (ICC)*, 2019, pp. 1–7.

[20] S. Li, Y. Cheng, Y. Liu, W. Wang, and T. Chen, "Abnormal client behavior detection in federated learning," *arXiv preprint arXiv:1910.09933*, 2019.

[21] F. Sattler, K.-R. Müller, T. Wiegand, and W. Samek, "On the byzantine robustness of clustered federated learning," in *proceeding of the IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP)*, 2020, pp. 8861–8865.

[22] L. Huang, Y. Yin, Z. Fu, S. Zhang, H. Deng, and D. Liu, "Loadaboost: Loss-based adaboost federated machine learning with reduced computational complexity on iid and non-iid intensive care data," *Plos one*, vol. 15, no. 4, 2020, DOI: 10.1371/journal.pone.0230706.

[23] W. Luping, W. Wei, and L. Bo, "Cmfl: Mitigating communication overhead for federated learning," in *proceeding of the IEEE 39th International Conference on Distributed Computing Systems (ICDCS)*, 2019, pp. 954–964.

[24] F. Liu, X. Wu, S. Ge, W. Fan, and Y. Zou, "Federated learning for vision-and-language grounding problems." in *proceeding of the AAAI Conference on Artificial Intelligence*, 2020, pp. 11 572–11 579.

[25] W. Y. B. Lim, N. C. Luong, D. T. Hoang, Y. Jiao, Y. Liang, Q. Yang, D. Niyato, and C. Miao, "Federated learning in mobile edge networks: A comprehensive survey," *IEEE Communications Surveys Tutorials*, vol. 22, no. 3, pp. 2031–2063, 2020.

[26] Z. Wang and H. Li, "Edge-based stochastic gradient algorithm for distributed optimization," *IEEE Transactions on Network Science and Engineering*, vol. 7, no. 3, pp. 1421–1430, 2020.

[27] N. H. Tran, W. Bao, A. Zomaya, N. M. NH, and C. S. Hong, "Federated learning over wireless networks: Optimization model design and analysis," in *proceeding of the IEEE Conference on Computer Communications*, 2019, pp. 1387–1395.

[28] Q. Zhou, S. Guo, H. Lu, L. Li, M. Guo, Y. Sun, and K. Wang, "Falcon: Addressing stragglers in heterogeneous parameter server via multiple parallelism," *IEEE Transactions on Computers*, 2020, DOI: 10.1109/TC.2020.2974461.

[29] T. Wang, H. Ke, X. Zheng, K. Wang, A. K. Sangaiah, and A. Liu, "Big data cleaning based on mobile edge computing in industrial sensor-cloud," *IEEE Transactions on Industrial Informatics*, vol. 16, no. 2, pp. 1321–1329, 2019.

[30] J. Wang, Y. Gao, C. Zhou, S. Sherratt, and L. Wang, "Optimal coverage multi-path scheduling scheme with multiple mobile sinks for wsns," *Computers, Materials & Continua*, vol. 62, no. 2, pp. 695–711, 2020.

[31] D. Cao, B. Zheng, B. Ji, Z. Lei, and C. Feng, "A robust distance-based relay selection for message dissemination in vehicular network," *Wireless Networks*, vol. 26, no. 3, pp. 1755–1771, 2020.

[32] X. Liu, P. Lin, T. Liu, T. Wang, A. Liu, and W. Xu, "Objective-variable tour planning for mobile data collection in partitioned sensor networks," *IEEE Annals of the History of Computing*, 2020, DOI: 10.1109/TMC.2020.3003004.

[33] T. Wang, P. Wang, S. Cai, X. Zheng, Y. Ma, W. Jia, and G. Wang, "Mobile edge-enabled trust evaluation for the internet of things," *Information Fusion*, 2021, DOI: 10.1016/j.inffus.2021.04.007.

[34] X. Liu, M. S. Obaidat, C. Lin, T. Wang, and A. Liu, "Movement-based solutions to energy limitation in wireless sensor networks: State of the art and future trends," *IEEE Network*, pp. 1–6, 2020, DOI: 10.1109/MNET.011.2000445.

[35] Y. LeCun, "The mnist database of handwritten digits," *http://yann. lecun. com/exdb/mnist/*, 1998.

[36] S. Zhou and B. Tan, "Electrocardiogram soft computing using hybrid deep learning cnn-elm," *Applied Soft Computing*, vol. 86, 2020, DOI: 10.1016/j.asoc.2019.105778.

[37] M. Long and Y. Zeng, "Detecting iris liveness with batch normalized convolutional neural network," *Computers, Materials & Continua*, vol. 58, no. 2, pp. 493–504, 2019.

Tian Wang received his BSc and MSc degrees in Computer Science from the Central South University in 2004 and 2007, respectively. He received his PhD degree in City University of Hong Kong in in Computer Science in 2011. Currently, he is a professor in the Institute of Artificial Intelligence and Future Networks, Beijing Normal University & UIC. His research interests include internet of things, edge computing and mobile computing. He has 27 patents and has published more than 200 papers in high-level journals and conferences. He has more than 7000 citations, according to Google Scholar. His H-index is 43. He has managed 5 national natural science projects (including 2 sub-projects) and 4 provincial-level projects.

Yan Liu received her BSc degree in Computer Information Engineering from Jiangxi Normal University in 2019. Currently, she is a master candidate in the National Huaqiao University of China. Her research interests include edge intelligence, mobile computing and edge computing.

Xi Zheng got PhD in Software Engineering from UT Austin. He specialised in Service Computing, IoT Security and Reliability Analysis. Published more than 40 high quality publications in top journals and conferences (PerCOM, ICSE, IEEE IoT Journal, ICCPS, IEEE Systems Journal, ACM Transactions on Embedded Computing Systems, IEEE Transactions on Vehicular Technology). Awarded the best paper in Australian distributed computing and doctoral conference in 2017. Awarded Deakin Research outstanding award in 2016. Active reviewer for top journals and conferences.

Hong-Ning Dai is currently with Faculty of Information Technology at Macau University of Science and Technology as an associate professor. He obtained the Ph.D. degree in Computer Science and Engineering from Department of Computer Science and Engineering at the Chinese University of Hong Kong. His research interests include Internet of Things, big data analytics and blockchain. He is a senior member of the Institute of Electrical and Electronics Engineers (IEEE).

Weijia Jia is currently a Chair Professor in the Institute of Artificial Intelligence and Future Networks, Beijing Normal University & UIC. He has been Zhiyuan Chair Prof at Shanghai Jiaotong University, China (where he received 2013 China 1000 Talent Award). He received BSc/MSc from Center South University, China in 82/84 and Master of Applied Sci./PhD from Polytechnic Faculty of Mons, Belgium in 92/93, respectively, all in computer science. From 93-95, he joined German National Research Center for Information Science (GMD) in Bonn (St. Augustine) as research fellow. From 95-13, he worked in City University of Hong Kong as a full professor in Computer Science Dept. His research interests include smart city; next generation IoT, knowledge graph constructions; multicast and anycast QoS routing protocols, wireless sensor networks and distributed systems. In these fields, he has over 500 publications in the prestige international journals/conferences and research books and book chapters. His contributions can be summarised from the aspects of vertex cover and efficient anycast for optimal placement and routing of severs/sensors in many applications of IoT/sensor/wireless networks and the Internet. He has received Best Product Awards from the International Science&Tech. Expos (Shenzhen) in 2011/2012 and 1st- Prize of Scientific Research Awards from Ministry of Education of PR China in 2017 (list 2). He has served as area editor for various prestige international journals, chair and PC member/keynote speaker for many top international conferences. He is the Senior Member of IEEE and the Member of ACM.

Mande Xie is currently a Professor in the Zhejiang Gongshang University. He received the PhD degree in Circuit & System from Zhejiang University in 2006. His research interests include Wireless Sensor Networks (WSNs), Social Network and Privacy Preservation.