Hindawi
Wireless Communications and Mobile Computing
Volume 2022, Article ID 1657558, 10 pages
https://doi.org/10.1155/2022/1657558

Research Article

PPEFL: An Edge Federated Learning Architecture with Privacy-Preserving Mechanism

Zhenpeng Liu,[1,2] **Zilin Gao**,[1] **Jingyi Wang**,[1] **Qiannan Liu**,[1,3] **and Jianhang Wei**[2]

[1]*School of Electronic Information Engineering, Hebei University, Baoding 071002, China*
[2]*Information Technology Center, Hebei University, Baoding 071002, China*
[3]*School of Cyber Security and Computer, Hebei University, Baoding 071002, China*

Correspondence should be addressed to Jianhang Wei; wei@hbu.edu.cn

Received 4 May 2022; Revised 15 September 2022; Accepted 23 September 2022; Published 11 November 2022

Academic Editor: Zhiguo Qu

Copyright © 2022 Zhenpeng Liu et al. This is an open access article distributed under the Creative Commons Attribution License, which permits unrestricted use, distribution, and reproduction in any medium, provided the original work is properly cited.

The emergence of federal learning makes up for some shortcomings of machine learning, and its distributed machine learning paradigm can effectively solve the problem of data islands, allowing users to collaboratively model without sharing data. Clients only need to train locally and upload model parameters. However, the computational power and resources of local users are frequently restricted, and ML consumes a large amount of computer resources and generates enormous communication consumption. Edge computing is characterized by low latency and low bandwidth, which makes it possible to offload complicated computing tasks from mobile devices and to execute them by the edge server. This paper is dedicated to reducing the communication cost of federation learning, improving the communication efficiency, and providing some privacy protection for it. An edge federation learning architecture with a privacy protection mechanism is proposed, which is named PPEFL. Through the cooperation of the cloud server, the edge server, and the edge device, there are two stages: the edge device and the edge server cooperate to complete the training and update of the local model, perform several lightweight local aggregations at the edge server, and upload to the cloud server and the cloud server aggregates the uploaded parameters and updates the global model until the model converges. The experimental results show that the architecture has good performance in terms of model accuracy and communication consumption and can well protect the privacy of edge federated learning.

1. Introduction

The rapid development of machine learning (ML) [1] has led to more and more intelligent IoT devices around people. Services such as Internet of Vehicles [2], recommendation systems, and smart cities require devices to collect a large amount of user privacy data, and people are often reluctant to share personal private data to complete ML. Many security and privacy issues are exposed by traditional ML frameworks. Therefore, federated learning [3] (FL) is proposed, which is a method to collaboratively train shared models without direct access to the raw data. It can solve the problem of data silos very well. In the current mainstream federated averaging algorithm [4] (FAVG), an ML model is trained locally with the user's private data, and then the model parameters are uploaded. The central server aggregates the uploaded parameters and updates the global model. The attacker cannot directly access the user's private data, which can protect the user's private data to a certain extent.

Unfortunately, federated learning incurs huge communication overhead [5]. First, there are too many communication rounds. In order to achieve the required model accuracy, hundreds of communications are required, which will result in huge data transmission from the local devices to the cloud server. Second, the resources of the edge devices may not be able to withstand the heavy computational demands of local training. Faced with these problems, some people use data compression [6] to solve them. However, the disadvantage of this method is that information is lost, resulting in reduced model accuracy.

As a distributed computing framework, edge computing [7] distributes the centrally processed data to edge nodes, which significantly improves the processing and transmission speed of data. Deploying FL on the distributed

architecture of edge computing can make full use of the feature that edge servers can connect to cloud servers and edge devices, reduce the computing pressure on cloud servers and edge devices, and improve the communication efficiency of FL.

Deploying federated learning in an edge computing environment still needs to address some challenges. One is to adopt the edge federated learning architecture, which does not change the current edge computing paradigm, but offloading all calculations to the edge server will lead to insufficient computing resources of the edge server [8]. Second, in edge federated learning, when parameters are exchanged between edge devices and edge servers, there is a risk of parameter leakage [9–11]. To address this issue, it is necessary to add additional privacy-preserving schemes for edge federated learning.

In order to reduce the communication cost of edge federated learning and find a balance between communication efficiency, edge device computing pressure and privacy protection, in this paper, we propose an edge federated learning architecture with a privacy protection mechanism (PPEFL). It optimizes hierarchical edge federated learning and applies the split learning idea to local training to reduce communication overhead. A privacy protection module is added to the edge client to effectively protect data privacy. In this way, the communication overhead is reduced and the privacy protection is unified. The contributions of this paper are summarized as follows:

(1) An optimized edge federation learning framework is proposed, where users do not need to upload local data but only need to train the model on the client-side and upload parameters, avoiding leakage of the user's own information and without modifying the existing edge computing paradigm

(2) To reduce the calculation amount of edge devices, complex computing tasks are offloaded to edge servers for calculation, which can effectively improve the computing efficiency of edge devices, and a lightweight local aggregation method is proposed to effectively reduce communication overhead

(3) A lightweight encryption algorithm is used to improve privacy protection when passing model parameters

The rest of this paper is organized as follows. Section 2 describes some related work. Section 3 describes the system framework. Section 4 describes the process of implementing PPEFL, and Section 5 discusses the effectiveness of PPEFL by conducting an experimental evaluation. Finally, Section 6 gives the conclusions.

2. Related Work

The advent of federated learning allows users to complete collaborative training without uploading local data, but only uploading model parameters for aggregation, enabling "data availability and invisibility" [12]. However, it will cause huge communication consumption, and to address this problem, Wang et al. proposed a framework to reduce the communication overhead of federated learning, CMFL [13], where the client checks whether its updates match the feedback provided by CMFL to the client to avoid transmitting irrelevant updates to reduce the communication overhead. Paragliola and Coronato proposed a novel federated learning approach TFedAvg [14], which reduces the communication overhead by evaluating two learning strategies, the FullNet strategy and the PartialNet strategy, to reduce the communication cost. Due to its distributed architecture and edge servers close to the device, edge computing can relieve certain computing pressure for edge devices. Wang et al. applied federation learning to edge computing to propose in-edge AI [15]. Through federated learning, to intelligentize edge computing, near-optimal performance can be achieved with relatively low learning overhead. Paragliola and Coronato proposed an efficient communication federated learning method for vehicular edge computing in 6G communication networks, namely, FedCPF [16]. It improves the convergence speed through a customized local training strategy. The communication cost is reduced by uploading partial training results, and a flexible aggregation strategy is proposed to further reduce the communication overhead. Ye et al. proposed an edge federated learning (EdgeFed) [17], which uses a segmentation technique to offload part of the computation from the mobile client to the edge server, reducing the computation cost for the user and also reducing the global communication overhead.

Federated learning has some privacy-preserving efficacy due to its locally trained architecture, which can avoid direct data sharing. But when information is shared between clients and servers, it still leads to privacy leaks. In modern society, people attach great importance to the security of private information, such as identity, data, location, and other private information, and the leakage of these data may lead to extremely serious consequences. Thence, to solve the problem of gradient leakage due to user uploaded data in federation learning, M.A.P., Chamikara et al. proposed a distributed perturbation algorithm DISTPAB [18], which perturbs the data locally before the edge devices have to perform federation learning to achieve improved privacy-preserving performance by disturbing the original data. In addition, some authors have used differential privacy to improve the privacy-preserving performance of federation learning. Wei et al. [19] proposed a differential privacy-based framework that adds artificial noise to the client's parameters before aggregation and later proposed a user-level differential privacy algorithm [20], which can effectively improve the training efficiency. Wang et al. proposed a three-plane framework [21] and adopted local differential privacy on the user plane to solve the problem of user-level privacy leakage under the cross-silo federated learning framework and designed a data reconstruction algorithm on the edge plane, which makes it impossible for malicious attackers to directly access or infer user data. Asad et al. proposed a noninteractive zero-knowledge proof homomorphic cryptosystem (NIZKP-HC) [22] for protecting local gradient updates. Chen et al. [23] proposed a mutual information-based federation learning parameter aggregation algorithm

that can effectively resist malicious node attacks. Federated learning enables users to train locally due to its unique architecture, reaching the point where uniform model training can be accomplished across multiple mobile devices without aggregating users' original data and avoiding attacks on mobile users' information during delivery by collecting users' uploaded model gradients or weights. The data link between the edge server and edge devices in edge computing is short, with sufficient bandwidth resources and relatively fast data transmission speed. Therefore, combining the advantages of both, some authors propose edge federated learning. Zhang et al. proposed a federated learning framework supporting mobile edge computing, FedMEC [24], which achieves differential privacy by adding Laplacian random noise to locally trained data features to provide privacy protection, but this will reduce the accuracy of the model, while adding additional communication overhead. In order to reduce the communication overhead as much as possible, reduce the computing cost at the device, improve the privacy protection capability of federated learning, and make it better applied to real-world scenarios, we propose PPEFL, an edge federated learning framework with a privacy protection mechanism.

3. System Structure

3.1. Federated Learning Based on Edge Computing. The PPEFL architecture combines federal learning with edge computing, which is designed on the basis of edge computing. It is a 3-2-2 model architecture, which consists of three layers: cloud server, edge server, and edge device; two-tier aggregation, local training by edge devices and edge servers, federated learning model by edge servers and cloud servers; two-layer encryption, the key is composed of the key of the edge device and the key of the cloud server; and complete decryption requires two decryptions, as shown in Figure 1.

The architecture includes a cloud server, L edge servers and K edge devices, where the edge server is denoted by l, the edge device is denoted by C_i^l, where $1 \le i \le K$, the edge client contains the distributed dataset D_i^l; use D^l to represent the aggregated data set under server l, $w_i^l(t)$ to represent the model parameters of user i, x_r to represent the output of the layer r, and loss function to be represented by $F(w)$. As the control center, the cloud server initializes the weights and parameters of the global model, broadcasts the calculation tasks to the clients participating in the training, aggregates the new information uploaded by the local clients, updates the global model, and then assigns it to the edge equipment participating in the training. Deployed in an environment at the edge of the network, edge servers are servers between the cloud and edge devices, such as routers, microbase stations, or microclouds in regional data centers, which are used to transmit data, storage, computing, etc. The edge device is the user's mobile device which stores a large number of user privacy data.

3.2. Threat Model. In PPEFL, it is assumed that edge devices are credible, while cloud servers and edge servers are honest but curious; that is, all entities can complete the calculation process of federated learning well, but they will try to reason for the private information of the training data. In response to this assumption, the designed architecture should be able to protect the information of the participants during the transmission process, and at the same time, have good accuracy and efficiency.

4. Scheme Realization

The basic idea of PPEFL is that the edge server and the edge device are trained together to complete the training and update of the local model. After the edge server completes several local model aggregations, it is uploaded to the cloud server to complete the global model aggregation and updating. For the benefit of enhancing the privacy protection capability during transmission, a lightweight encryption is added to encrypt the data to protect the data.

By jointly completing local training on edge devices and edge servers, the computational overhead on the client can be effectively reduced. At the same time, an optimized hierarchical edge federated learning mechanism is proposed, which is essentially a cycle strategy, that is, adding several local updates and aggregations before transmitting to the cloud server, reducing the number of communications between the edge server and the cloud server, and reducing communication overhead.

4.1. Local Training. PPEFL selects a convolutional neural network as the model framework. Due to the high computational overhead of complete CNN training, training directly on edge devices may lead to excessive training time and insufficient edge device resources. Compared with edge devices, edge servers have more powerful computing and storage resources, as well as a more stable power supply. The data link between the edge server and the edge device is short, the bandwidth resources are sufficient, and the data transmission speed is relatively fast. Offloading some tasks to the edge server can reduce the computing cost of edge devices and improve communication efficiency.

A complete neural network consists of many hidden layers, and the input of each hidden layer can be regarded as the output of the previous layer. Therefore, these layers can be divided into two parts by using model segmentation technology, namely, the edge device area and the edge server area, which are called C-CNN and E-CNN, respectively, are shown in Figure 2. Among them, C-CNN includes the convolutional layers of the local CNN, and E-CNN includes the rest.

The front-end part of the CNN structure is deployed on edge devices to facilitate feature extraction from raw data and local training. The rest of the CNN structure is deployed in the edge server area to perform forward and backward propagation procedures in order to update the model parameters. In this way, the edge device and the edge server can work together to complete the training. Here, C-CNN includes the first convolutional layer and pooling layer, and after the raw data is processed through C-CNN, the output is sent to the edge server. In order to reduce the amount of data transmission, O clients are randomly selected, and

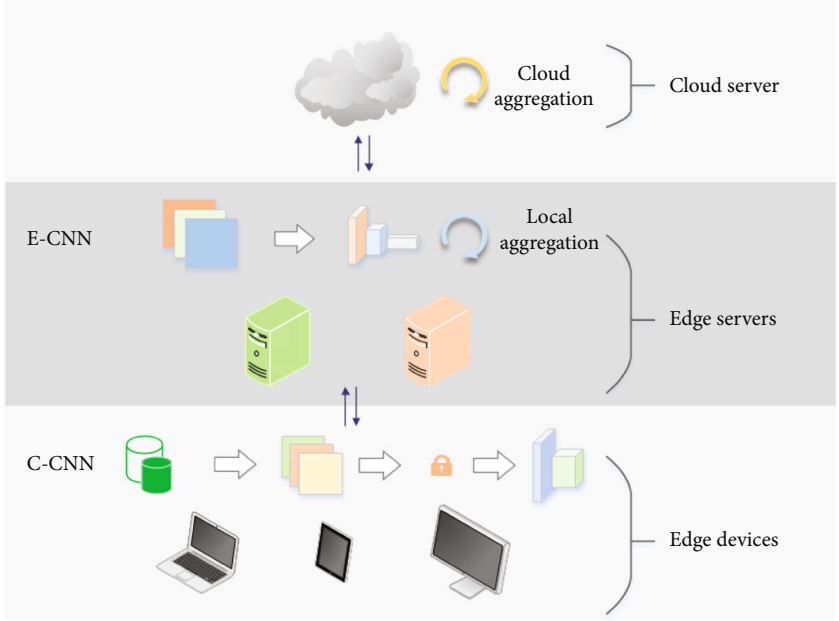

Figure 1: PPEFL architecture.

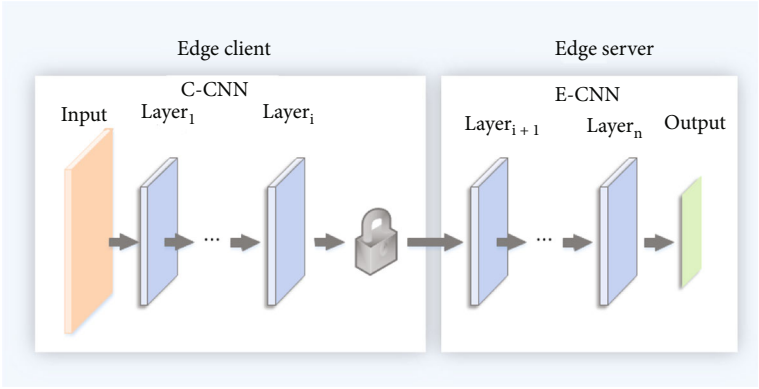

Figure 2: C-CNN and E-CNN.

the selected clients transmit the encrypted data to the edge server to complete E-CNN. Dividing the complete CNN into two parts and computing them in different devices can effectively reduce the computation of the edge client. The algorithm of the C-CNN part is as follows.

4.2. Model Aggregation. We take inspiration from the hierarchical federated averaging algorithm (HierFAVG) [25] and propose an optimized edge federation algorithm to further reduce the communication energy consumption. The edge server is used as an intermediate parameter aggregator to reduce the communication overhead between the edge server and the cloud server and avoid excessive communication delay. The basic idea is that after the edge client completes τ_1 local updates, the edge server aggregates the model, and after completing τ_2 edge aggregation, the edge server uploads it to the cloud server for aggregation.

In each iteration of the optimization algorithm, the k users participating in the training download the C-CNN model from the edge server L to the local and entered the local dataset it owns into the C-CNN to generate features and then encrypted them. After encryption, it is transmitted to the edge server to complete the E-CNN. In order to reduce the number of communication rounds and reduce the communication cost, an optimized edge federated learning mechanism is used instead of federated learning. Using the edge server as the intermediate parameter aggregator, after several local joint updates, the edge server is locally aggregated several times and then uploaded to the central server for aggregation, thereby greatly reducing the communication cost.

In order to further reduce communication consumption, a local aggregation mechanism is designed in the local aggregation stage to reduce the resource occupancy of the edge server. The edge server aggregates all the models updated on this server, which is called the edge model $w^l(t)$. After τ_2 edge aggregations, it is uploaded to the cloud server for aggregation to complete the update of the global model, that

Procedure EDGE DEVICES
  Initialized all clients with parameter w_0
  For $t = 1, 2, \cdots, T$ **do**
    For each client $i = 1, 2, \cdots, N$ in parallel **do**
      Train C-CNN
      Encrypt x_r and send features to edge servers
    End for
  End for
End procedure

Algorithm 1: C-CNN in edge devices.

is, $w^c(t)$, which means that the edge server and the cloud server are locally updated every τ_1 and τ_2. The aggregation algorithm is shown in Algorithm 2.

In the local aggregation stage, the local aggregation mechanism is applied to the edge server to complete τ_1 local updates and aggregate the models several times. In order to reduce the amount of computation, the edge server adopts the local aggregation mechanism to randomly select a part of the parameters according to the proportion θ in the parameters updated by each edge device to complete the aggregation of the edge model. The calculation formula of local aggregation is as follows:

$$w^l(t) = \frac{\sum_{i\epsilon\theta} |D_i^l| w_i^l(t)}{|D_\theta^l|}. \tag{1}$$

After τ_2 local aggregations, the cloud server uploads the model to the cloud server to complete global aggregation. The calculation is as follows:

$$w^c(t) = \frac{\sum_{l=1}^{L} |D^l| w^l(t)}{|D|}. \tag{2}$$

After the cloud server completes the aggregate update, it sends the updated model to the edge server to complete the next round of updates until the model converges.

4.3. Lightweight Encryption. Federated learning can provide some basic privacy protection for users' original data due to the attributes of local training. However, clients participating in training may still receive attacks from malicious users, resulting in information leakage. Therefore, an encryption mechanism is required to protect the participating users';data privacy.

Homomorphic encryption (HE) can operate the calculation performed on the ciphertext intact on the plaintext, which means that after homomorphic encryption of the client's data, it can still be inputted into the convolutional neural network to complete various calculations without losing security and accuracy. Among them, ElGamal encryption is a homomorphic encryption with a multiplication mechanism, which has higher computational efficiency and stronger encryption capability. In the traditional homomorphic encryption algorithm, if the client leaks his private key to

Procedure AGGREGATION IN EDGE SERVERS
EVENT: Receive x_r and features
  For each edge $l = 1, \cdots, L$ in parallel **do**
    Train E-CNN
    $w_i^l(t) \longleftarrow w_i^l(t-1) - \gamma \nabla F_i(w_i^l(t-1))$
    If $t|\tau_1 = 0$ **then**
    $w^l(t) \longleftarrow$ Local Aggregation
      If $t|\tau_1\tau_2 \neq 0$ **then**
        For each client $i \in C^l$ in parallel **do**
          $w_i^l(t) \longleftarrow w^l(t)$
        End for
      End if
  End for
End procedure
Procedure AGGREGATION IN CLOUD SERVERS
  If $t|\tau_1\tau_2 = 0$ **then**
    $w^c(t) \longleftarrow$ Global Aggregation
    For each client $i = 1, 2, .., N$ in parallel **do**
      $w_i^l(t) \longleftarrow w^l(t)$
    End for
  End if
End procedure

Algorithm 2: Aggregation in edge servers and cloud server.

the client or server with the ciphertext, the plaintext can be easily inferred, resulting in information leakage. So this paper alleviates this situation by introducing secret sharing technology and double key. Using Feldman's Verifiable Secret Sharing (VSS) [26] to generate keys can successfully and securely verify key sharing and key generation without referring to a third-party trust organization. In order to strengthen the ability of privacy protection, we modify the traditional key generation and the number of keys and propose verifiable double-key encryption.

According to Feldman's VSS combined with ElGamal encryption [27], to complete the encryption of features, the encryption method is as follows:

Parameter generation: generate three parameters p, q, and g, where q is the prime order of cyclic group G; p is a large prime satisfying $q|p-1$, and g is the generator of G.

Key generation: in the t-th round, each participating user i generates a random polynomial $f_i(x) = \sum_{j=0}^{t-1} a_{ij} x^j$, such that $s = z_i = a_{i0} = f_i(0)$ is the locally stored private key. Taking $s_{ij} = f_i(j) \bmod p$ as a share, calculate and broadcast $\alpha_{ij} = g^{a_{ij}} \bmod p, i = 0, 1, \cdots, t-1$; P_j verifies that $g^{s_{ij}} = \prod_{i=0}^{t-1} \alpha_{ij}^{j^i} \bmod p, j = 1, 2, \cdots, n$. If the equation does not hold, then the shared s_{ij} received by the user is invalid. When the sharing of all collaborators is verified as valid, the collaborators can calculate the secret s according to the Lagrangian polynomial interpolation method. The specific recovery key calculation method is as follows:

$$s = \sum_{i=1}^{t} f_i(j) \prod_{1 \le j \le t, j \neq i} \frac{i}{i-j}. \tag{3}$$

The server can generate a global public key by broadcasting α_{i0} from all clients in equation (4), where x is the global private key.

$$\begin{aligned} \mathrm{pk}_i &= \alpha_{i0} = g^{z_i} (\mathrm{mod}\ p), \\ \mathrm{pk} &= \prod_i \mathrm{pk}_i = g^{\sum_i z_i} = g^x (\mathrm{mod}\ p). \end{aligned} \tag{4}$$

The cloud server selects a random number $r, r \in Z_p^*$ as the cloud key ck = r, and the public key is pk = g^{xck} mod p.

Encryption: encrypt message m, compute ciphertext $c_1 = g^{\mathrm{ck}} (\mathrm{mod}\ p)$, $c_2 = \mathrm{mpk}^{\mathrm{ck}} (\mathrm{mod}\ p)$.

Decryption: decrypt the plaintext message m by formula (5).

$$\frac{c_2}{c_1} = \frac{\mathrm{mpk}^{\mathrm{ck}}}{\left(g^{\mathrm{ck}}\right)^x} = \frac{\mathrm{mg}^{\mathrm{xck}}}{\mathrm{gxck}} (\mathrm{mod}\ p) \equiv m. \tag{5}$$

Decryption can only be completed after the cloud server and the edge device are decrypted separately. After the server is aggregated, the decryption is performed, and then part of the decrypted ciphertext is downloaded to the edge server and edge device, and the edge device can restore the plaintext after decryption. In this way, no one can decrypt by holding the global private key.

Since EIGamal encryption is a homomorphic encryption of a multiplication mechanism, the model aggregation in FL performs an addition operation. In order to make the encryption method better applied in FL, the encrypted message m is transformed into $\widehat{m} = g^m (\mathrm{mod}\ p)$. Through this transformation, homomorphic encryption can be well applied in FL, as shown in

$$\begin{aligned} \mathrm{Enc}(m_1) * \mathrm{Enc}(m_2) &= g^{m_1} \mathrm{key}^{1*} g^{m_2} \mathrm{key}^2 (\mathrm{mod}\ p) \\ &= g^{m_1 + m_2} \mathrm{key}^{1+2} (\mathrm{mod}\ p). \end{aligned} \tag{6}$$

When the plaintext is recovered, the discrete logarithm problem is solved by Pollard's rho algorithm adopted in [27]. This can solve discrete logarithm problems in tens of milliseconds on a moderately capable computer. It is worth noting that the assumption is that the participants voluntarily provide the labeled data to perform supervised learning, and privacy in the labels is not expected, so the privacy in the labels of the training data samples is not considered by us.

4.4. Security Analysis. Since edge servers and cloud servers are honest but curious, this requires that the security requirements of the proposed encryption scheme include verifiability and privacy. Verifiability requires no proof as the server always strictly enforces the program. In terms of privacy, it is necessary to ensure data privacy security and parameter privacy security. For data privacy, in PPEFL, clients do not need to upload local parameters and do not share them with other users. Model training is always performed locally, which ensures the privacy and security of user data. For parameter privacy, the proof is given in Theorem 1.

Theorem 1. *PPEFL meets the requirements of parameter privacy.*

Proof. During training, there is no additional information on edge devices revealed except the output of a specific layer. If the server tries to expose information, it needs to resolve g^m. However, information is protected in two ways. When ck is obtained from g^{ck}, it is protected by the hardness of discrete logarithm. When g^{ck} and g^{μ} is obtained from $\mathrm{g}^{\mu\mathrm{ck}}$, it is protected by the hardness of the computational Diffie-Hellman problem. It is almost impossible to obtain the information before encryption. Therefore, the proposed encryption scheme meets the requirements of parameter privacy. It is almost impossible to obtain the information before encryption. As a result, the proposed encryption scheme meets the requirements of parameter privacy. □

5. Analysis of Experimental Results

5.1. Experimental Setup. The experimental dataset is the MNIST handwritten digit classification dataset, which includes 60 k training samples and 10 k test samples, including ten different label categories from 0 to 9, and the UCI Human Activity Recognition (HAR) dataset, including 7 k training samples and 3 k test samples, containing 50 dimensional features.

The network structure used is a convolutional neural network (CNN), which consists of two 5×5 convolutional layers, two 2×2 pooling layers, a fully connected layer, and a final softmax output layer. The weights in the model are initialized to random values sampled from the normal distribution N (0, 0.022) and the biases are initialized to 0. We simulated a setup of 100 devices, 10 edge servers, and a central server. Discussed the scenario where each edge server has the same number of edge devices uploading information, and each edge device has the same amount of data. For local updates on each client, minibatch stochastic gradient descent with a batch size of 10 is employed. Use the pickle module to convert gradient parameters into a stream of bytes for transmission. The experiments were performed on a Windows 10 machine with an Intel i7-8700K CPU, GTX 1080 T GPU, and 16GB RAM.

5.2. Experimental Results

5.2.1. Influence of Parameters. Firstly, the parameters in the model are simulated to discuss the performance of the model under different parameters. In optimizing the hierarchical edge federated learning architecture, the key to the optimization technique lies in the number and proportion of local updates and local aggregations. We first analyze the impact of the number of local and global aggregations on model accuracy and then discuss the problem of local update ratios. It is worth noting that in order to better test the impact of parameters on performance, the homomorphic encryption method is not used. It is assumed in this paper that $O = 50$.

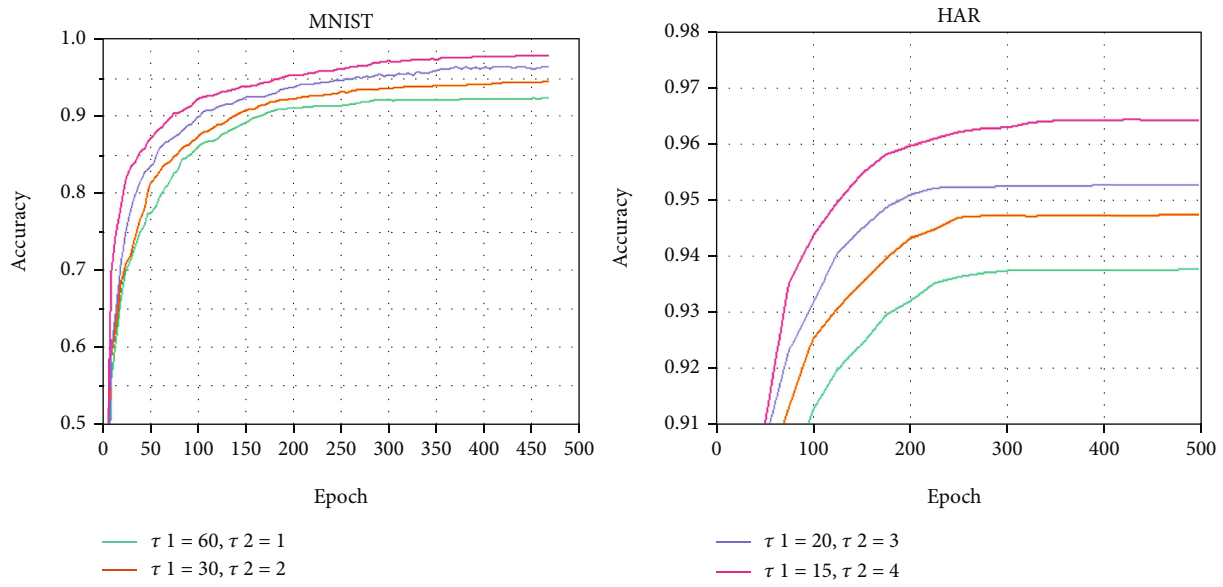

Figure 3: Test accuracy for different local updates and local aggregation times.

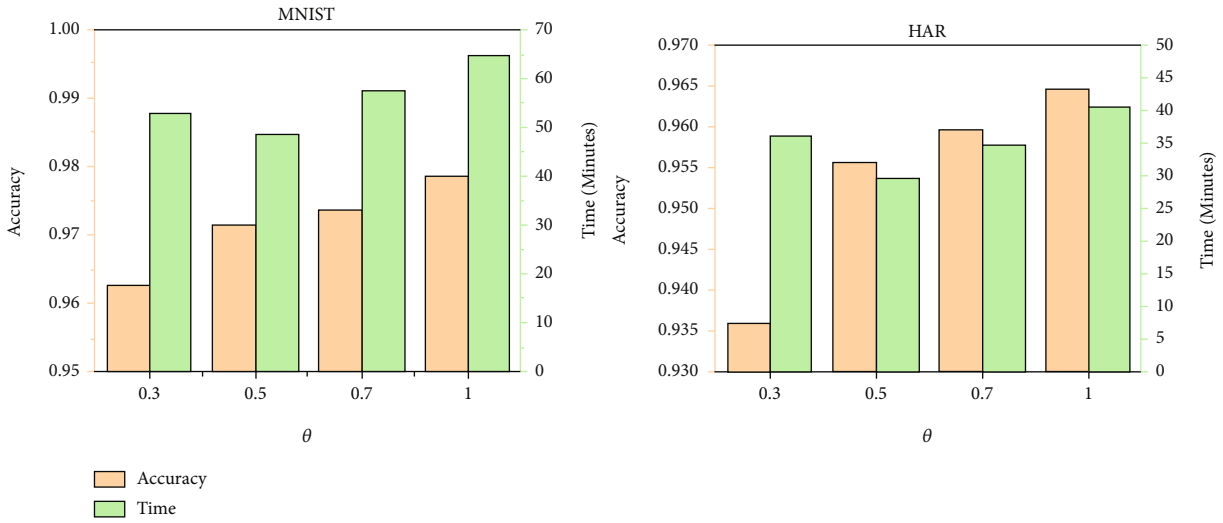

Figure 4: Test accuracy at different scales.

When $\tau_2 = 1$ and all local aggregations, the optimized hierarchical federated learning is the same as FAVG, so $\tau_2 = 1$ is used as the baseline. As shown in Figure 3, when $\tau_1 = 15$ and $\tau_2 = 4$, the best efficiency can be achieved in the least time, which is also consistent with the experimental results in [25]. There are similar experimental results on both datasets.

Experiments show that, without affecting the performance of the model, increasing the number of local gradient aggregation can reduce the communication cycle with the cloud, achieve better accuracy in a shorter time, and help reduce communication overhead.

In addition, we also found that increasing the threshold when local aggregation is updated can further reduce the computational overhead without affecting the model accuracy. The experimental results are shown in Figure 4. When $\theta = 1$, all local aggregations increase the computing pressure of the edge server, which also means longer computing time, so we choose the ratio of $\theta = 0.5$. Because under this ratio, not only can better results be achieved but also the communication time is shorter and the communication cost is reduced.

5.2.2. Execution Time. By testing the time it takes to build a model on the dataset, it reflects the efficiency of the algorithm. Taking the traditional FedAvg algorithm as the benchmark and comparing EdgeFed, which also adopts the idea of segmentation learning; we test the time required by the three parties under the same model accuracy. According to [28], the latency of one round communication between edge server and cloud server is set to 100 ms, the latency of one round of communication between edge device and edge server is 10 ms, and the latency of one local update on the device is

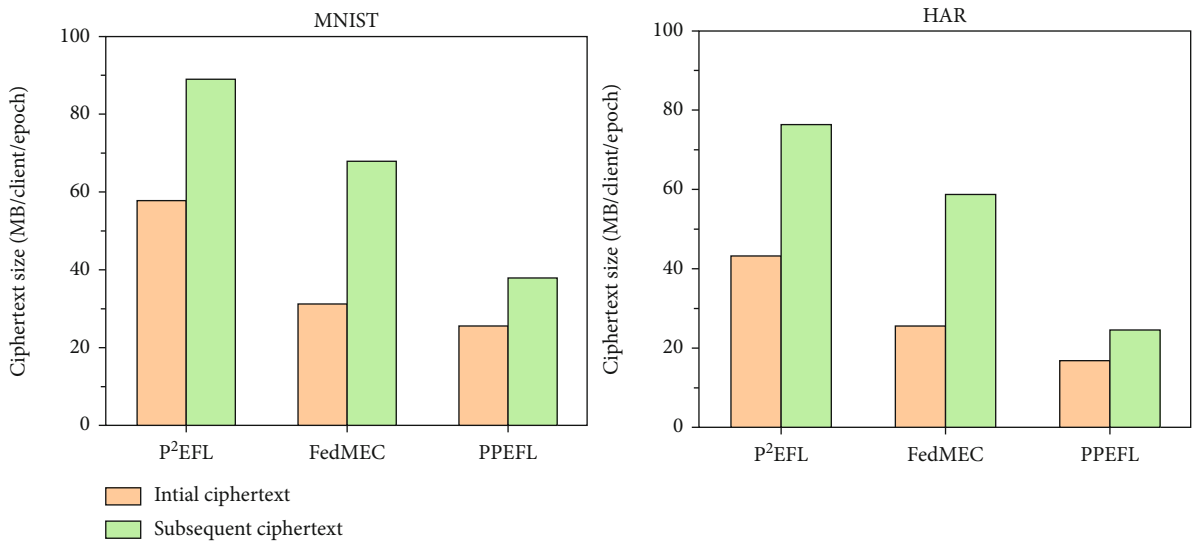

Figure 5: Ciphertext size transmitted per epoch.

1 ms. Taking the MNIST dataset results as an example, it takes 62.7 min, 60.5 min and 106.1 min to train on PPFEL, EdgeFed, and FedAvg to achieve 97% accuracy, respectively. This shows that adopting segmentation learning can not only reduce the computational pressure of edge devices but also achieve better performance in less time. PPEFL optimizes the model aggregation, reduces the number of communication rounds between the edge server and the cloud server, and thus reduces the computational overhead, which is not much different than the EdgeFed running time but adds additional privacy protection capabilities, which means that the proposed edge local aggregation method is effective.

5.2.3. Comparison with Other Models. The comparison scheme is as follows:

EdgeFed [17]: applying federated learning to edge computing, all participants complete training on local and edge servers, upload updates to cloud servers to complete aggregation, and reduce device computing costs and global communication consumption, but have no other privacy protection methods.

FedMEC [24]: adopt the federated edge learning framework to further protect privacy security through differential privacy.

P²FEC [29]: also adopts the federated edge learning framework to ensure user privacy through secure multiparty computation.

PPEFL: our scheme adopts an optimized edge federated learning framework to protect privacy through lightweight encryption.

The four comparison schemes were completed under the same settings, and the experimental results are as follows.

(1) Data transmission

The local update composition is completed by the edge device and the edge server, which will inevitably bring additional communication costs. After multiple local aggregations are performed at the edge server, the global aggregation is completed. By reducing the number of global aggregations, the purpose of reducing the communication pressure can be achieved. In PPEFL, to provide 128-bit security level, the ElGamal key size is set to 256 and the group size is set to 3072. To evaluate the transmission efficiency, the size of the encryption parameters was measured, and the Figure 5 shows the size of the ciphertext transmitted in one global epoch. By limiting the number of clients participating in uploading gradients, the size of the transmitted ciphertext is reduced.

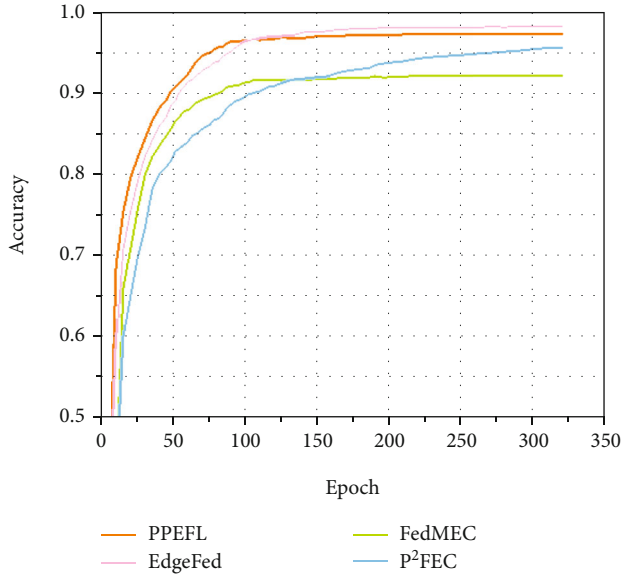

Figure 6: Accuracy comparison experiment of the four schemes in MNIST.

(2) Model training

The time of model convergence is used to reflect the size of communication overhead because the increase of communication rounds will inevitably bring a large amount of communication overhead. Experiments show that our proposed scheme can achieve good accuracy with fewer communication rounds. The model has a shorter convergence time and lower communication overhead. As can be seen from Figure 6, the number of communications required by this method on the MNIST dataset is less than the EdgeFed, which can reduce the communication overhead, but the accuracy rate is reduced.

Under the same training time, the model accuracy of EdgeFed scheme using no privacy protection scheme is slightly higher than other schemes. Using the FedMEC scheme is slightly lower than the other schemes, which is due to the addition of additional noise, resulting in a reduction in model accuracy. The test accuracy of the PPEFL scheme is slightly lower than that of the EdgeFed scheme in the comparison, because some parameters are discarded in the local aggregation process, resulting in a decrease in accuracy. Although the accuracy is slightly lower than the EdgeFed scheme, privacy protection is very important for information, so it is more suitable for the edge computing network environment. Compared with the same precision, using the P^2FEC scheme takes longer to execute, because SMC requires more rounds of communication in each global update.

6. Summary

In order to protect clients' data privacy and reduce communication consumption, a federated learning based on edge computing framework (PPFEL) is proposed that can improve both efficiency and privacy. The proposed framework is based on optimizing hierarchical edge federated learning. Local updates are implemented on the edge client and edge server, and a local aggregation is completed on the edge server after several updates. After several local aggregations are completed, upload to the cloud server to complete the global aggregation, and the global model update is sent, and the global model updates to edge servers until the model converges. In an effort to reduce the computational pressure of the edge client, a segmentation model is used to divide the complete convolutional neural network into two parts, which are deployed on the edge device and the edge server, respectively. With the purpose of ensuring the privacy of data transmission, a lightweight encryption algorithm that does not require a third-party trusted organization is used to provide sufficient privacy assurance. Finally, the effectiveness of the proposed method is proved by experiments.

Data Availability

The experimental data used to support the findings of this study are available from the corresponding author upon request.

Conflicts of Interest

The authors declare that they have no conflicts of interest.

Acknowledgments

This research was supported by the Natural Science Foundation of Hebei Province, China under Grant No. F2019201427 and the Fund for Integration of Cloud Computing and Big Data, Innovation of Science and Education of China under Grant No. 2017A20004.

References

[1] S. Singh, S. Rathore, O. Alfarraj, A. Tolba, and B. Yoon, "A framework for privacy-preservation of IoT healthcare data using federated learning and blockchain technology," *Future Generation Computer Systems*, vol. 129, pp. 380–388, 2022.

[2] C. Chen, L. Liu, S. Wan, X. Hui, and Q. Pei, "Data dissemination for industry 4.0 applications in internet of vehicles based on short-term traffic prediction," *ACM Transactions on Internet Technology*, vol. 22, no. 1, pp. 1–18, 2021.

[3] L. Gao, L. Li, Y. Chen, C. Xu, and M. Xu, "FGFL: A blockchain-based fair incentive governor for federated learning," *Journal of Parallel and Distributed Computing*, vol. 163, pp. 283–299, 2022.

[4] C. Palihawadana, N. Wiratunga, A. Wijekoon, and H. Kalutarage, "FedSim: similarity guided model aggregation for federated learning," *Neurocomputing*, vol. 483, pp. 432–445, 2022.

[5] T. Cao, T. Truong-Huu, H. Tran, and K. Tran, "A federated deep learning framework for privacy preservation and communication efficiency," *Journal of Systems Architecture*, vol. 124, article 102413, 2022.

[6] F. Sattler, S. Wiedemann, K. R. Muller, and W. Samek, "Robust and communication-efficient federated learning from non-IID data," *IEEE Transactions on Neural Networks and Learning*, vol. 31, pp. 3400–3413, 2019.

[7] N. Harth, C. Anagnostopoulos, H. Voegel, and K. Kolomvatsos, "Local & federated learning at the network edge for efficient predictive analytics," *Future Generation Computer Systems*, vol. 134, pp. 107–122, 2022.

[8] C. Chen, Y. Zhang, Z. Wang, S. Wan, and Q. Pei, "Distributed computation offloading method based on deep reinforcement learning in ICV," *Applied Soft Computing*, vol. 103, article 107108, 2021.

[9] Z. Zhang, Y. Zhang, D. Guo, L. Yao, and Z. Li, "SecFedNIDS: robust defense for poisoning attack against federated learning-based network intrusion detection system," *Future Generation Computer Systems*, vol. 134, pp. 154–169, 2022.

[10] N. Rodríguez-Barroso, E. Martínez-Cámara, M. Luzón, and F. Herrera, "Backdoor attacks-resilient aggregation based on robust filtering of outliers in federated learning for image classification," *Knowledge-Based Systems*, vol. 245, article 108588, 2022.

[11] N. Rodríguez-Barroso, E. Martínez-Cámara, M. Victoria Luzón, and F. Herrera, "Dynamic defense against byzantine poisoning attacks in federated learning," *Future Generation Computer Systems*, vol. 133, pp. 1–9, 2022.

[12] C. Fang, Y. Guo, J. Ma, H. Xie, and Y. Wang, "A privacy-preserving and verifiable federated learning method based on blockchain," *Computer Communications*, vol. 186, pp. 1–11, 2022.

[13] L. Wang, W. Wang, and B. Li, "Cmfl: mitigating communication overhead for federated learning," in *2019 IEEE 39th International Conference on Distributed Computing Systems*, pp. 954–964, Dallas, TX, USA, 2019.

[14] G. Paragliola and A. Coronato, "Definition of a novel federated learning approach to reduce communication costs," *Expert Systems with Applications*, vol. 189, article 116109, 2022.

[15] X. Wang, Y. Han, C. Wang, Q. Zhao, and M. Chen, "In-edge ai: intelligentizing mobile edge computing, caching and communication by federated learning," *IEEE Network*, vol. 33, no. 5, pp. 156–165, 2019.

[16] S. Liu, J. Yu, X. Deng, and S. Wan, "FedCPF: An efficient-communication federated learning approach for vehicular edge computing in 6G communication networks," *IEEE Transactions on Intelligent Transportation Systems*, vol. 23, no. 2, pp. 1616–1629, 2022.

[17] Y. Ye, S. Li, F. Liu, Y. Tang, and W. Hu, "EdgeFed: optimized federated learning based on edge computing," *IEEE Access*, vol. 8, article 209191, 2020.

[18] M. Chamikara, P. Bertok, I. Khalil, D. Liu, and S. Camtepe, "Privacy preserving distributed machine learning with federated learning," *Computer Communications*, vol. 171, pp. 112–125, 2021.

[19] K. Wei, J. Li, M. Ding et al., "Federated learning with differential privacy: algorithms and performance analysis," *IEEE Transactions on Information Forensics and Security*, vol. 15, pp. 3454–3469, 2020.

[20] K. Wei, J. Li, M. Ding et al., "User-level privacy-preserving federated learning: analysis and performance optimization," *IEEE Transactions on Mobile Computing*, vol. 5, pp. 3454–3469, 2020.

[21] C. Wang, X. Wu, G. Liu, T. Deng, K. Peng, and S. Wan, "Safeguarding cross-silo federated learning with local differential privacy," *Digital Communications and Networks*, vol. 8, no. 4, pp. 446–454, 2022.

[22] M. Asad, A. Moustafa, and M. Aslam, "CEEP-FL: a comprehensive approach for communication efficiency and enhanced privacy in federated learning," *Applied Soft Computing*, vol. 104, article 107235, 2021.

[23] N. Chen, Y. Li, X. Liu, and Z. Zhang, "A mutual information based federated learning framework for edge computing networks," *Computer Communications*, vol. 176, pp. 23–30, 2021.

[24] J. Zhang, Y. Zhao, J. Wang, and B. Chen, "FedMEC: improving efficiency of differentially private federated learning via mobile edge computing," *Mobile Networks and Applications*, vol. 25, no. 6, pp. 2421–2433, 2020.

[25] L. Liu, J. Zhang, S. Song, and K. Letaief, "Client-edge-cloud hierarchical Federated learning," in *ICC 2020-2020 IEEE International Conference on Communications*, pp. 1–6, Dublin, Ireland, 2020.

[26] H. Zhu, R. Wang, Y. Jin, K. Liang, and J. Ning, "Distributed additive encryption and quantization for privacy preserving federated deep learning," *Neurocomputing*, vol. 463, pp. 309–327, 2021.

[27] C. Fang, Y. Guo, N. Wang, and A. Ju, "Highly efficient federated learning with strong privacy preservation in cloud computing," *Computers & Security*, vol. 96, article 101889, 2020.

[28] G. Premsankar, M. D. Francesco, and T. Taleb, "Edge computing for the internet of things: a case study," *IEEE Internet of Things Journal*, vol. 5, no. 2, pp. 1275–1284, 2018.

[29] G. Liu, C. Wang, X. Ma, and Y. Yang, "Keep your data locally: Federated-learning-based data privacy preservation in edge computing," *IEEE Network*, vol. 35, no. 2, pp. 60–66, 2021.