electronics

MDPI

Article

Adaptive Quantization Mechanism for Federated Learning Models Based on DAG Blockchain

Tong Li [1,2], Chao Yang [3,4], Lei Wang [3], Tingting Li [5,*], Hai Zhao [1] and Jiewei Chen [5]

1 College of Computer Science and Engineering, Northeastern University, Shenyang 110819, China
2 Electric Power Research Institute, State Grid Liaoning Electric Power Co., Ltd., Shenyang 110055, China
3 State Grid Liaoning Electric Power Co., Ltd., Shenyang 110004, China
4 College of Electrical and Electronic Engineering, North China Electric Power University, Beijing 102206, China
5 State Key Laboratory of Networking and Switching Technology, Beijing University of Post and Telecommunications, Beijing 100087, China
* Correspondence: litingting@bupt.edu.cn; Tel.: +86-18203781194

Abstract: With the development of the power internet of things, the traditional centralized computing pattern has been difficult to apply to many power business scenarios, including power load forecasting, substation defect detection, and demand-side response. How to perform efficient and reliable machine learning tasks while ensuring that user data privacy is not violated has attracted the attention of the industry. Blockchain-based federated learning (FL), proposed as a new decentralized and distributed learning framework for building privacy-enhanced IoT systems, is receiving more and more attention from scholars. The framework provides some advantages, including decentralization, scalability, and data privacy, but at the same time its consensus mechanism consumes a significant amount of computational resources. Moreover, the number of model parameters has increased dramatically, leading to an increasing amount of transmitted data and a vast communication overhead. To reduce the communication overhead, we propose an FL framework in the directed acyclic graph (DAG)-based blockchain system, which achieves efficient and trusted sharing of FL networks. We design an adaptive model compression method based on k-means to compress the FL model and reduce the communication overhead of each round in FL training. Meanwhile, the original accuracy-based tips selection algorithm is optimized, and a tips selection algorithm considering multi-factor evaluation is proposed. Simulation experimental results show that the method proposed in this paper reduces the total bytes of communication of the blockchain-based federated learning system while balancing the accuracy of the FL model compared to previous work.

Keywords: federated learning; DAG; communication overhead; adaptive model quantization

check for updates

Citation: Li, T.; Yang, C.; Wang, L.; Li, T.; Zhao, H.; Chen, J. Adaptive Quantization Mechanism for Federated Learning Models Based on DAG Blockchain. *Electronics* **2023**, *12*, 3712. https://doi.org/10.3390/electronics12173712

Academic Editor: Imran

Received: 3 August 2023
Revised: 22 August 2023
Accepted: 28 August 2023
Published: 2 September 2023

1. Introduction

With the deep integration of internet of things (IoT) technology and the power grid, the intelligent development of power IoT has gradually attracted people's attention. Coordinated scheduling between the power generation side, user side, and distribution network is the key to the development of power systems. The scenarios include intelligent inspection, power load forecasting, and demand-side response. These tasks require a power system with trusted data sharing capabilities and big data mining capabilities. However, the development of new power systems faces some problems and challenges. First, the traditional centralized computing framework is vulnerable to attacks by third parties, and the data transmission process is at risk of data leakage and tampering [1]. Second, with the development of artificial intelligence technology, the number of model parameters increases significantly, and the limited resources in IoT devices make adapting to the development of large models challenging. Third, people's awareness and concerns about privacy are growing. Governments have implemented data privacy legislation, such as the European

Commission's General Data Protection Regulation (GDPR) [2] and the U.S. Consumer Privacy Bill of Rights [3].

In recent years, federated learning (FL) has been proposed as a distributed learning framework for building a data privacy-enhanced power system. The authors in [4] suggested that FL can solve the privacy problem among data owners. Lu et al. [5] proposed a decentralized and secure FL model based on blockchain. This model integrates FL into the consensus process of blockchain, which improves the system's security without the need for centralized trust. However, the traditional consensus mechanism causes extreme resource consumption.

To avoid the extra resource consumption caused by blockchain, Li et al. [6] proposed DAG consensus, a consensus mechanism designed based on the structure of a directed acyclic graph (DAG). A blockchain system using such a consensus mechanism is called a DAG blockchain. Compared with proof-of-work (PoW) and proof-of-stake (PoS), that have been widely used in blockchain, the consensus mechanism designed on DAG consensus can overcome some shortcomings, such as high resource consumption, high transaction fee, low transaction throughput, and long confirmation delay. An important aspect of DAG-based consensus mechanisms is the tip selection algorithm [7,8]. The algorithm determines the selection of tips that should be approved when the next transaction is issued. That is the parent node of the published new transaction connection. In the DAG consensus-based scheme, the traditional tip selection algorithm always chooses the highest weight when selecting, and its traditional transaction weight is calculated by counting the number of approved transactions. Cao et al. [9] first combined the DAG blockchain with FL (DAG-FL). DAG-FL adopts asynchronous FL to approve the nodes by verifying the accuracy of the tips and the local models, and the local models with considerable accuracy of the local models are selected to construct the global model.

However, there are two main problems in the current DAG blockchain-based federated learning framework: first, the system communication overhead increases as the number of federated learning model participants increases. Second, the model transmission process is vulnerable to gradient leakage attacks [10–12]. Therefore, how to realize an efficient and trustworthy federated learning framework with balanced learning accuracy has become an urgent problem to be solved.

We note that there is a gradient compression approach to compress the uploaded gradients [13] to reduce the burden of communication in each round. Model compression refers to using various techniques to reduce the size, complexity, and computation of machine learning models so that they can be deployed and run on resource-constrained devices. Model compression can effectively reduce the storage and computing resource requirements of the model, and improve the inference speed and efficiency of the model, so as to realize efficient machine learning applications in resource-constrained scenarios such as mobile devices and edge devices. In a federated learning system, due to the frequent transmission of gradients, the communication overhead between an aggregation node and multiple clients in distributed training is large, which limits the training efficiency of federated learning. Luo et al. [14] verifies the applicability of model compression methods in distributed deep learning. In addition, Beilharz et al. [7] proposed an accuracy-based tips selection algorithm for specialization implicitly. Inspired by this, we offer a k-means-based adaptive model quantization scheme to reduce the communication overhead in an FL training task. Meanwhile, a grading-based tips selection algorithm is proposed to make specific changes to the direction of FL iterations, taking into account the accuracy and overall communication overhead of the system. The main contributions of this paper are as follows:

(1) A k-means-based adaptive model compression scheme is proposed to reduce the communication overhead. Instead of transmitting the original model, the compressed model is transmitted. It not only reduces the communication overhead of the nodes but also improves the security of the system.

(2) A grading-based tips selection algorithm is proposed, which integrates the accuracy of each iteration during FL training and the overall communication overhead of the system, so that the FL model iterates in the direction of high accuracy.
(3) The primary convolutional neural network (CNN) model is trained for the handwriting recognition tasks by using the MNIST dataset. Extensive experiments show that our CDAG-FL scheme saves approximately 16% of the communication overhead compared to the baseline method.

The rest of the paper is organized as follows: Section 2 introduces the related work, Section 3 describes the system framework and workflow of the system, Section 4 analytically models the communication overhead problem in DAG-based federated learning systems, Section 5 proposes a k-means-based adaptive model quantization scheme and a grading-based tips selection method, and Section 6 performs experimental simulation verification. The last chapter summarizes the thesis work.

2. Related Work

2.1. Convergence Framework for DAG Blockchain and Federated Learning

Scholars have researched and proposed a framework for converging DAG blockchain and FL. The earliest one is DAG-FL, proposed by Mingrui Cao et al. [9] to solve the problem of device asynchrony and anomaly detection in an FL framework, avoiding the extra resource consumption brought by the blockchain. It proposes a framework for FL using a blockchain based on the direct acyclic graph (DAG), which achieves better performance in terms of training efficiency and modeling accuracy compared with the existing typical on-device FL systems. However, it needs to address the problem of communication overhead. Based on this, Beilharz et al. [7] proposed a framework called directed acyclic graph federated learning (SDAGFL). It not only overcomes the challenges of device heterogeneity, single point of failure, and poisoning attacks, but also creates a unified solution for decentralized and personalized FL. But again, it dose not consider the communication overhead.

However, in IoT scenarios, the computing nodes have limited computing and communication resources with strict energy constraints. In order to optimize the SDAG-FL system for IoT, Xiaofeng Xue et al. [15] proposed an energy-efficient SDAG-FL framework based on an event-triggered communication mechanism, i.e., ESDAG-FL. The ESDAG-FL can reasonably achieve the balance between the training accuracy and specialization of the model and reduces nearly half of the energy consumption. Inspired by this, this paper proposes a new SDAG-FL scheme for efficient communication, called CDAG-FL. We design an adaptive model compression method based on the k-means mechanism and an improved tips selection algorithm for the CDAG-FL system in power IoT. The relevant research analysis of blockchain and federated learning architecture is shown in Table 1.

Table 1. Comparing our proposed scheme with existing methods.

	Optimization of Training Efficiency	Optimization of Model Accuracy	Scalability	Model Compression	Optimization of Tips Selection
Literature [6]	✓	✓	×	×	×
Literature [11]	×	×	✓	×	×
Literature [12]	✓	✓	×	✓	✓
CDAG-FL	✓	✓	✓	✓	✓

2.2. Communication Overhead Issues

To address the problem of how to reduce the communication overhead in FL, Mingzhe Chen et al. [16] used stochastic gradient quantization to compress the local gradient. He optimized the quantization level of each device under the multi-access channel capacity constraints to minimize the optimality gap, which reduces the communication overhead of FL. However, we must consider the reduction of convergence speed brought by model

compression to FL. Wei Yang et al. [13] analyzed the effect of fixed compression rate in model compression on the number of iterations and training error in the training process, proved that a suitable compression rate can better perform the compression algorithm, and proposed an adaptive gradient compression algorithm, which provides a unique compression rate for each client according to the actual characteristics of each client, to improve the communication performance. However, it does not consider the influence of the client training process. Peng Luo et al. [14] proposed a new ProbComp-LPAC algorithm. The ProbComp-LPAC algorithm selects the gradient with a probability equation and uses different compression rates in different layers of a deep neural network. In the same layer, the more parameters, the lower the compression rate with higher accuracy. ProbComp-LPAC is not only faster in training speed but also has high accuracy. However, the compression rate of each layer needs to be adjusted manually, and its effect is limited.

3. System Model

3.1. System Framework Architecture

The DAG-based efficient communication federated learning system (CDAG-FL) proposed in this paper is an asynchronous FL system, including the application layer, DAG layer, and FL layer. A more detailed illustration of this system is shown in Figure 1.

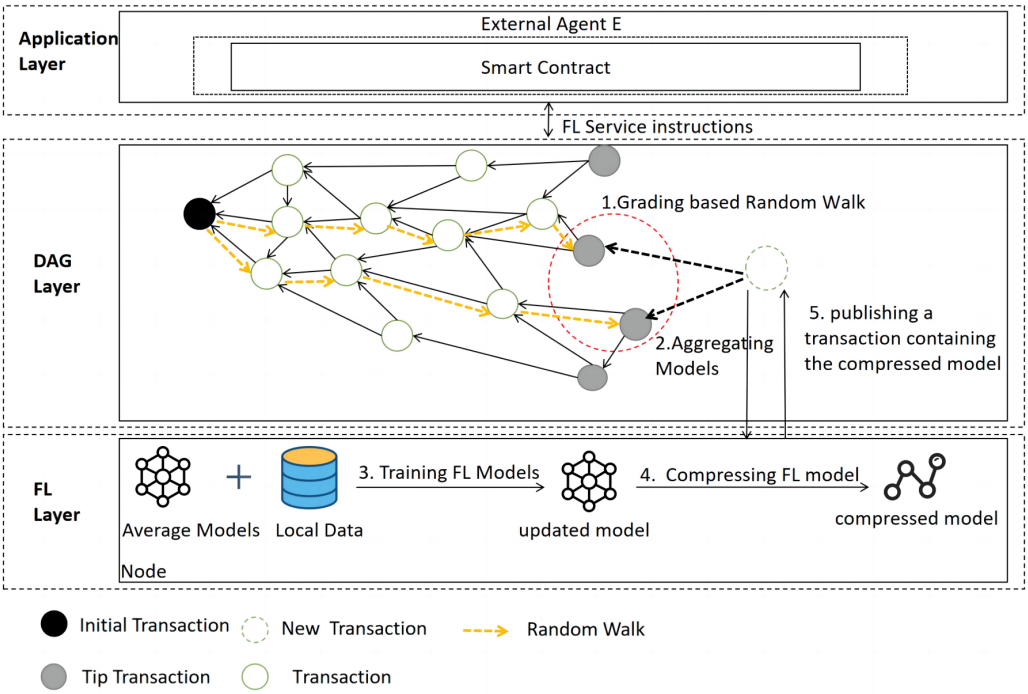

Figure 1. The DAG-based efficient communication federated learning system (CDAG-FL) framework diagram.

Application Layer: the application layer is deployed at the top layer of the system to manage the system process and provide interfaces for external agents through intelligent contracts. Firstly, external agents can issue FL tasks to nodes through intelligent contracts. Secondly, during the FL process, the smart contract observes the transactions on the DAG and comes to the test at regular intervals to determine whether the target model has been reached. Finally, when the FL iteration termination requirement is reached, the application layer issues an iteration termination command to the node through the smart contract.

DAG Layer: at the DAG layer, each node maintains the global DAG, where the published transactions in the DAG contain authentication information, local model parameters, and approval connections. Using P2P technology, nodes update the DAG by broadcasting over the wireless network.

FL Layer: the FL layer provides the function of FL model training. The FL layer obtains the latest DAG by interacting with the DAG layer. It locally selects the models to be aggregated using the tips selection algorithm, after which the models stored in the transactions on the DAG are aggregated using the Federated Averaging algorithm to form the Average Models. After further training using its dataset, it will encapsulate a newly trained model (Updated Models) into a transaction on the DAG for publishing and processing.

3.2. System Workflow

The DAG-based efficient communication federated learning(CDAG-FL) system is an asynchronous FL system without a centralized server, which deploys on mobile device nodes such as smartphones, wearables, and IoT devices under a wireless network, and these nodes collectively maintain a public DAG blockchain. We assume that these nodes can communicate with each other, as shown in Figure 1.

Suppose the set of power IoT devices denote as $D = \{1, 2, 3, \ldots, N_D\}$, where N_D is the number of nodes and D_i is the *i*-th node in D. Denote the training dataset on the D_i as S_i, where N_i is the number of samples of S_i. The local model trained by node D_i at moment *t* can denote ω_i^t. Assume that the Transaction in the DAG blockchain structure divides into four classes, and a complete model is stored in each class.

1. Initial Transaction: The initial transaction contains FL tasks and the initial model parameters ω_0.
2. Transaction: The primary transaction type contains the published model.
3. Tip Transaction: A unique transaction that contains the latest model and has not been approved by other nodes.
4. New Transaction: The new node contains the model to be published.

In the CDAG-FL, the task publisher of federated learning can represent an external agent, *E*, which holds a virtual machine to run smart contracts. The external agent, *E*, publishes the initial global model, ω_0, and federated learning tasks and observes the FL process. If the model meets the requirements, it issues a stop iteration command, all nodes stop iteration immediately, and the federated learning task ends.

The node D_i will repeat the following five steps during this period to update the model, as shown in Figure 1. Assume the node will update the model at the t_0 start of the FL iteration.

Step 1: After obtaining the latest DAG, the node receives *k* Tip Transactions by performing *k* times of Grading-based Random Walks. The selection process is represented by the orange dashed line in Figure 1. The *k*-selected Tip Transactions are marked with red dashed lines in Figure 1.

Step 2: Then the local model, ω, and the models in these *k* Tip Transactions, $\omega_{d_1}^{t_1}, \omega_{d_2}^{t_2}, \omega_{d_3}^{t_3}, \ldots, \omega_{d_k}^{t_k}$, where the parameters satisfy $(t_1, t_2, \ldots, t_k \leq t_0,\ d_1,\ d_2, \ldots, d_k \in D)$, are aggregated into one model, ω^{t_0}, using the Federated-Averaging algorithm.

$$\omega^{t_0} = \frac{1}{k+1}\left(\sum_{j=1}^{k} \omega_{d_j}^{t_j} + \omega\right) \tag{1}$$

Step 3: Next, the node D_i uses m randomly selected samples from the local dataset as a small batch of the Z_i dataset in the local dataset, S_i, to the aggregation model obtained above ω^{t_0} to perform μ rounds of training to receive a new model, ω_i. The samples in Z_i can be represented as (x_i, y_i), where x_i is the feature set and y_i is the label set. Then, in each FL iteration, the loss function of the local model, ω_i, of the D_i node can be expressed as $f_{z_i}(\omega_i) = l(x_i, y_i; \omega_i)$, which represents the prediction error of (x_i, y_i) in the ω_i. Every time FL iteratively trains the model, it hopes that the local model, ω_i, can minimize $F_i(\omega_i)$.

$$F_i(\omega_i) = f_{Z_i}(\omega_i) \tag{2}$$

Step 4: After that, the models in the obtained k Tip Transactions, $\omega_{\mathrm{d}_1}^{\mathrm{t}_1}, \omega_{\mathrm{d}_2}^{\mathrm{t}_2}, \omega_{\mathrm{d}_3}^{\mathrm{t}_3}, \ldots, \omega_{\mathrm{d}_k}^{\mathrm{t}_k}$, are used as reference models, and the node compares the new model, ω_i, with the reference models. If the test loss of the new model is lower than that of all the reference models, it meets the requirements of the release. After that, the new model by the k-means-based adaptive quantization method is ω_i compressed into ω_i'.

Step 5: The client publishes the compressed model, ω_i', encapsulated into a New Transaction published to the public DAG blockchain, updating the DAG structure with its k Tip Transactions as its parents. Otherwise, the model is not published, and the process starts again.

Throughout the iterative process, the external agent, E, expects to obtain the global model, ω, through the smart contract after several FL iterations to minimize $F(\omega)$, as follows:

$$\min F(\omega) = \frac{1}{\mathrm{N}_D} \sum_{i \in D} F_i(\omega_i) \tag{3}$$

3.3. Problem Formulation

After understanding the framework structure and workflow of the DAG-based efficient communication federated learning system, we analyze its communication overhead. According to Section 3.2, the system model training process can be divided into five steps: node acquisition Tip Transactions, model aggregation, model training, model comparison, and model release. According to the analysis in [10–12], we analyze the communication overhead in the system as follows. The communication overhead metric in the system is defined by the total amount of data communicated by all the nodes during the communication process, which is constantly interacting and updating. In each round of communication, the communication overhead of each node mainly comes from these two processes.

Node Acquisition Tip Transactions: The communication overhead of this process is determined by the number of models it downloads and the size of the models it downloads, i.e., the number of bits it transmits. It assumes that each node $\mathrm{D_i}$ performs k times on the DAG blockchain structure Grade-based Random Walk during each iteration. Suppose that during the Grade-based Random Walk, the average number of models downloaded per Grade-based Random Walk is n. $\mathrm{B}_{\mathrm{D}_i}$is the average number of bits of the downloaded models by node $\mathrm{D_i}$ during this process, which is also its moderate size. Therefore, the node $\mathrm{D_i}$ obtaining k Tip Transactions has the following communication overhead.

$$C_{D_i}^{\mathrm{down}} = k \times n \times B_{D_i} \tag{4}$$

Model Comparison and Publishing: The communication overhead of this process is determined by the size of its publishing model, i.e., its number of bits. It assumes that the publish model condition $x = 1$ is satisfied; otherwise, $x = 0$. The node $\mathrm{D_i}$ publishes a model of size $\mathrm{B}_{\mathrm{D_i}}^{\omega}$. The communication overhead of the model comparison and publishing process is shown below.

$$C_{D_i}^{\mathrm{up}} \begin{cases} \mathrm{B}_{D_i}^{\omega} & x = 1 \\ 0 & x = 0 \end{cases} \tag{5}$$

From the above analysis, we can obtain the node $\mathrm{D_i}$ communication overhead during one FL iteration as follows.

$$\begin{aligned} C_{\mathrm{D}_i}^{\mathrm{total}} &= C_{D_i}^{\mathrm{down}} + C_{D_i}^{up} \\ &= k \times n \times B_{D_i} + C_{D_i}^{up} \end{aligned} \tag{6}$$

Assume that the number of communication rounds required for federated learning to reach the predefined performance metric is N, during which a total of M nodes participate. Therefore, its total amount of communication data is:

$$C = \sum_{\mathrm{i}=1}^{\mathrm{M}} \sum_{j=1}^{N} C_{D_{i,j}}^{\mathrm{down}} + C_{D_{i,j}}^{up} \tag{7}$$

From the steps of updating the model by the nodes, we can see [13] that many model parameters must be constantly exchanged between the participants. From the analysis of the communication overhead in the system, we can see that the number of communication rounds in the FL iteration and the amount of data transferred between nodes in a single communication round are the main reasons for the high communication overhead. By analyzing the primary sources of communication overhead in the steps, we learn that reducing the amount of data in a single round of communication can rely on appropriate model compression of the node models to reduce the size of the transmitted models to reduce the bandwidth occupied by the communication to reduce its communication overhead. On the other hand, with model compression on the node models, the size of the models published by the nodes on the DAG blockchain may be different. In order to reduce the total amount of transmitted data in the FL iteration process, it can rely on the change of the method of selecting Tip Transaction in the FL iteration process from only considering the accuracy [7] to also considering the bit size of the model, thus prompting the federated learning to iterate in the direction of the small bit size of the model and high accuracy as a way of reducing the total amount of transmitted data in the whole federated learning process.

Taken together, the optimization objective equation for this work is:

$$\text{Minimize}\left[C, \min_{\omega \in R^d} \mathrm{F}(\omega)\right] \tag{8}$$

In other words, the optimization objective minimizes the federated learning training loss function while minimizing the communication overhead during federated learning.

4. Optimization Methods

With the system communication overhead analysis and its established optimization objective in Section 4, we propose a k-means-based adaptive quantization model compression method and a grading-based tips selection strategy to optimize this objective.

4.1. Adaptive Quantization Model Compression Based on k-Means

Model weight sharing quantization is a way of model compression that compresses the parameters in the model from high bit bits to low bit bits, which can reduce the number of bits in the model transmission and, in this way, reduce the amount of data transmitted in a single round of communication in federated learning. Therefore, we propose an adaptive quantization model compression method, which performs adaptive model quantization compression on the model after the nodes have trained the model and before posting the model to the DAG blockchain.

In deep learning, researchers [17] use quantization and weight-sharing methods for model compression to reduce the number of bits of weights stored in the model. Weight sharing means that multiple connections use the same weights. The general steps are first to model the transmitted ω weights of each layer by clustering them into k classes by a clustering algorithm, and all the weights in each category share the same value, i.e., the weight of the cluster center. The model weights are stored through a clustering index, and the clustering center is stored in a codebook data structure, as shown in Figure 2.

After the network model clusters, when storing and transmitting the model, the weights of the model only need to be represented by the indexes of the clusters. During model

transmission, only the indexes of model weights and the structure of the corresponding codebook need to be transmitted, which significantly reduces the communication overhead.

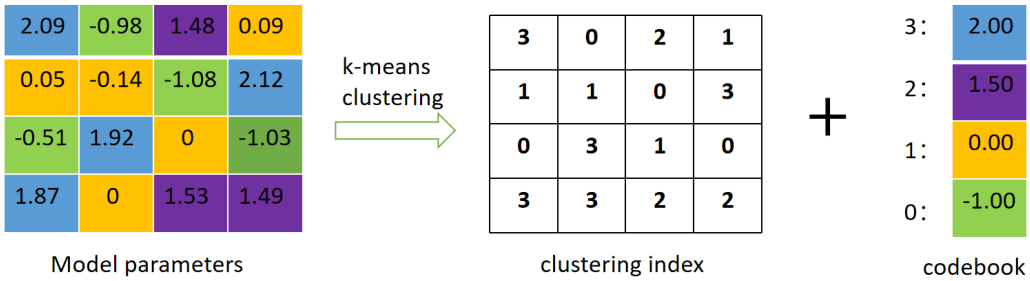

Figure 2. Quantization of the k-means model for a certain layer with a clustering number of 4 [17].

4.1.1. Selection of Clustering Centers and Number of Clusters k

In the model quantization method based on k-means clustering, the choice of the clustering number k is essential, affecting the size of the model quantization and the effect after quantization. The selection of the traditional clustering number k, which sets artificially, is usually 32 or 64. However, due to the different data quality of nodes in federated learning, data heterogeneity, data distribution, etc., all nodes set to the same number of clusters will affect the effect of federated learning. Similarly, the number of weights for each layer of the network in the deep learning model is different, and setting the same number of clusters for each layer will affect the effect of the model, and each manual setting is very time-consuming.

Similarly, cluster center initialization has generally random uniform initialization, density-based initialization, and linear method-based initialization. Its clustering centers sample for density-based initialization, where the weight values distribute more. This is because larger weights in neural networks are more important and are usually in the minority. Therefore, this method produces a negligible probability of selecting essential weights. In addition, for random uniform initialization, the selection is random and unstable. Therefore, considering the above, in this paper we use a linear method-based initialization that selects the clustering centers in a way that covers all the weights very uniformly.

Through the above analysis, this paper adopts an adaptive model quantization scheme to replace the original time-consuming and laborious method of manually setting the k value, which first uses the adaptive model quantization scheme to learn the number of clusters, k, at each model layer. It then searches for the clustering centers of the weights according to the number of clusters at each layer based on the k-means algorithm.

During a particular round of iterations of federated learning, suppose the model released by node $\mathrm{D_i}$ is ω; for a certain layer of the model, we use the k-means algorithm for clustering, assuming that the adaptive model quantization scheme yields that the weights of the layer are classified as $(\mathrm{L_1}, \mathrm{L_2}, \ldots, \mathrm{L_i}, \ldots, \mathrm{L_k})$ a total of k classes, where $\mathrm{i} \in [1, \mathrm{k}]$.$\mathrm{u_i}$ is the cluster center of class $\mathrm{L_i}$. w denotes the parameters of the model. By optimizing Equation (9), we can obtain the clustering center of the layer $(\mathrm{u_1}, \mathrm{u_2}, \ldots, \mathrm{u_k})$.

$$\arg\min \sum_{i=1}^{k} \sum_{w \in L_i} \|w - u_i\|^2 \tag{9}$$

After obtaining the number of clusters, k, and the clustering centers of the layer $(\mathrm{u_1}, \mathrm{u_2}, \ldots, \mathrm{u_k})$, the model can be quantized using weight sharing, where the initial parameters of the model are represented by the index of the class it belongs to. A codebook records all the classes and their clustering centers, as shown in Figure 2.

4.1.2. Calculation of Quantized Partial Model Bits

After quantization through clustering and weights, the number of bits of the model are calculated. Generally, deep network model parameters are represented by floating point numbers of 32 or 64 bits.

In this paper, we assume that the model's parameters are represented by b bits and that the current layer has a total of n parameters with a clustering number of k. The current layer is represented by the number of bits B = nb. After quantization, the model's initial parameters are represented by the class index to which they belong. Since it is binary encoded in the computer, the number of bits required to represent the index of the k classes is $\log_2 \mathrm{k}$. In addition to this, the values of the k clustering centers are stored, and their values are represented by b bits, as are the model parameters. Thus, for the current layer, the number of bits after quantization is:

$$B' = n \log_2 k + kb \tag{10}$$

Since the number of clusters in each layer of the deep learning model differs, the number of model parameters in each layer is also different. Assuming that the model has t layers, the model ω has a bit number of:

$$B_{\omega} = \sum_{i=1}^{t} n_i b \tag{11}$$

where $\mathrm{n_i}$ denotes the number of parameters in the i-th layer of the model, where $\mathrm{k_i}$ indicates the number of clusters in the i-th layer of the model; after quantization, the model ω' of the bit number is:

$$\mathrm{B}_{\omega'} = \sum_{i=1}^{t} n_i \log_2^{k_i} + k_i b \tag{12}$$

After five steps of FL iteration, it assumes that the model to be released by node $\mathrm{D_i}$ is ω_i. In order to obtain the clustering number, k, of each network layer of the model, the number of clusters of the model is obtained according to the mapping Equation (13).

$$\mathrm{k_{all}} = \mathrm{round}(((\mathrm{max_k} - \mathrm{min_k}) * acc_i) + \mathrm{min_k}) \tag{13}$$

where $\mathrm{max_k}$ denotes the maximum number of clusters (the maximum number of clusters is set to 1024) and $\mathrm{min_k}$ indicates the minimum number of clusters; setting the minimum number of clusters to $4.\mathrm{acc_i}$ denotes the node $\mathrm{D_i}$ precision on the local test set. the round function represents the integer value of the obtained floating point number within a certain range.

Suppose the model ω_i has a total of t layers, respectively $(1, 2, \ldots i, \ldots, t)$. For the i-th layer of the model, assume that the number of parameters in this layer is $\mathrm{n_i}$, the number of zero parameters in the layer is $\mathrm{zero_i}$, and the mean of the parameters is $\mathrm{x_i}$. After calculation, the number of absolute values of the parameter of this layer greater than $\mathrm{x_i}$ is $\mathrm{m_i}$. For the i-th layer of the model, the number of clusters is:

$$k_i = \mathrm{round}\left(\frac{m_i}{n_i - zero_i} * k_{\mathrm{all}}\right) \tag{14}$$

According to Equation (14), we can obtain the number of clusters of the t-layer network for the model ω_i, which is $(\mathrm{k_1}, \mathrm{k_2}, \ldots, \mathrm{k_i}, \ldots, \mathrm{k_t})$. After that, each layer network is clustered separately according to the k-means clustering algorithm to obtain the quantized model, as shown in Figure 3.

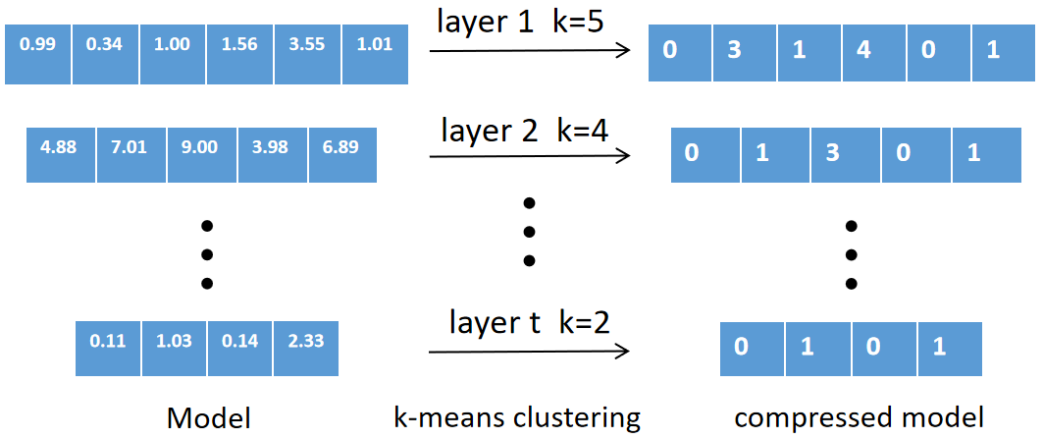

Figure 3. Example of k-means-based adaptive quantization model compression method with layer number t.

4.2. Grading-Based Tips Selection Strategy

In the DAG-based consensus scheme, the traditional tips selection algorithm randomly traverses the DAG in the opposite direction of approvals, as shown in Figure 4, with the orange line indicating its random wandering process. In the process of random wandering, when selecting the next transaction the weight of the transaction is calculated by counting the number of approved transactions, and this is used as the criterion for selection, always selecting the one with the highest weight until the Tip Transaction is found.

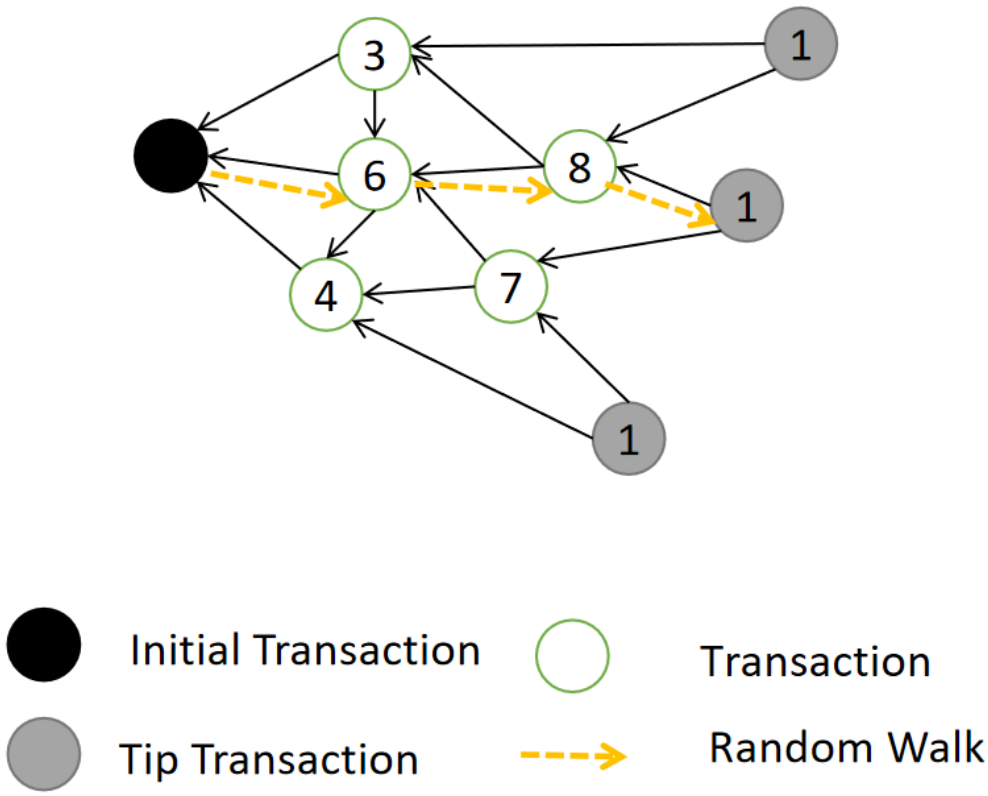

Figure 4. Example of traditional weight-based tips selection method [7].

Inspired by [7,13], we propose a grading-based tips selection strategy. In this system, after k-means-based adaptive model compression, the models stored in the transactions are different sizes. Considering the communication overhead, the grade integrates the metrics of model accuracy, size, and the number of times the model is approved. It uses them as

the criteria for selecting the next transaction during the random wandering process. In the selection of the next transaction in the randomized wandering process, it assumes that a total of n transactions exist with $\{\omega_1, \omega_2, \ldots, \omega_i, \ldots, \omega_n\}$. There are a total of n potential following models whose transactions label as $\{1, 2, \ldots \mathrm{i}, \ldots, \mathrm{n}\}$, the sizes of the models in their transactions are, respectively, $\{\mathrm{B}_1,\ \mathrm{B}_2, \ldots, \mathrm{B}_\mathrm{i}, \ldots, \mathrm{B}_\mathrm{n}\}$, and the number of times the model is approved is, respectively, $\{\mathrm{L}_1,\ \mathrm{L}_2, \ldots \mathrm{L}_\mathrm{i}, \ldots, \mathrm{L}_\mathrm{n}\}$. Where m denotes the number of all clients participating in federated learning, randomly select x samples in the local dataset S_i of node D_i as a small batch of dataset Z_i to test all those mentioned above next-step latent models. The accuracies obtained are, respectively, $\{\mathrm{acc}_1, \mathrm{acc}_2, \ldots, \mathrm{acc}_i, \ldots, \mathrm{acc}_n\}$. Assuming that the size of the initial model for federated learning is B_{ω_0}, for the sake of comparison, we use $\left(1 - \frac{B_i}{B_{\omega_0}}\right)$ as a measure of how much the model compressed. The same $\frac{\mathrm{L_i}}{\mathrm{m}}$ denotes how well the model is approved. By Equation (13), we can obtain the grade of model ω_i, after which the model selects the next step based on this grade, as shown in Algorithm 1.

$$\mathrm{S}_i = \alpha * acc_i + \beta * \left(1 - \frac{B_i}{B_{\omega_0}}\right) + \gamma * \left(\frac{\mathrm{L}_i}{m}\right) \tag{15}$$

Among them, α, β, and γ are three variables to control and balance the impact of their corresponding indicators on the scores.

Algorithm 1: Grading-based tips selection strategy

Input: DAG structure:$G = (V, E)$, $InitialNode$, Initial model size:D_0, Number of FL nodes:m
Output: $TipNode$

```
nextNode = InitialNode, TipNode = Null;
while GetChild(nextNode) ≠ Null do
    //get the number of child nodes
    n = len(nextNode);
    child[n] = GetChild(nextNode);
    Initial acc[n] = 0;  Score[n] = 0;D[n] = 0; L[n] = 0;
    for (i = 0; i < n; i++) do
        //get the relevant indicators of each child node
        acc[i] = GetAccBasedLocalData(Child[i]);
        D[i] = GetChildDataSize(Child[i]);
        L[i] = GetChildApprovalNumber(Child[i]);
    for (i = 0; i < n; i++) do
        Score_i = α * acc_i + β * (1 - D_i/D_0) + γ * L_i/m;
    //get the subscript with the maximum rating
    x = GetMaxIndexScore(Score[n]);
    nextNode = child[x];
TipNode = nextNode;
return TipNode;
```

5. Simulation Analysis

5.1. Experiment Setting

Unlike traditional FL, the DAG blockchain-based FL system is an entirely asynchronous framework without any centralized server. In order to simulate and validate the effectiveness of the proposed k-means-based adaptive model quantization scheme and grading-based tips selection strategy in solving the communication overhead problem, this paper uses the DAG blockchain-based FL framework to model the environment of the communication overhead problem. It involves and uses the open-source Python framework

Pytorch to simulate and experiment with the tips selection algorithm, and the adaptive model quantization scheme was manufactured and tested. The experiments simulate the scenarios of an adaptive model compression scheme during FL training and the tips selection algorithm during model transmission in the DAG blockchain environment.

Two quantization schemes simulate as a baseline for the adaptive model quantization scheme to compare with the k-means-based adaptive quantization scheme proposed in this paper. (1): No quantization: after nodes train the model, there is no model compression, e.g., traditional FL system, DAG-FL, SDAG-FL, etc., and the FL process under DAG-FL is simulated in this paper. (2): Fixed k-value: in this scheme, model quantization with the same k-value applies to the model published by any node in the system to mimic the case of manually setting the k-value, where the k-value takes an intermediate value.

Different accuracy-based methods for the tips selection algorithm are used in DAG blockchain-based FL frameworks such as SDAG-FL and DAG-FL. In this paper, we simulate the accuracy-based preference tips selection algorithm in SDAG-FL to compare it with the grading-based tips selection algorithm proposed in this paper. Since the premise of using the grading-based tips selection algorithm is that the nodes upload different numbers of model bits, we used the adaptive quantization scheme proposed in this paper for the model first.

5.1.1. Datasets and Models

In an efficient DAG-based FL system for communication (CDAG-FL), we evaluated a handwriting recognition task on the MNIST dataset with a 9:1 ratio of training and test sets for each node.

Dataset: The MNIST dataset is a subset of the NIST (National Institute of Standards and Technology) dataset, containing 60,000 figures for training and 10,000 for testing. The numbers have been size normalized and located in the center of the images, which are of fixed size (28×28 pixels) and have values from 0 to 1. For simplicity, each image has been flattened and converted to a one-dimensional numpy array of 784 (28×28) features.

Model: A convolutional neural network (CNN) is used for the handwriting recognition task. It contains two convolutional layers and two fully connected layers. The kernel size of the convolutional layers is 5 using the RELU activation function. A maximum pooling layer with a pool size and step size of 2 follows each convolutional layer. The two fully connected layers contain 2048 and 10 neurons, respectively.

5.1.2. Training Hyperparameters Setting

The training hyperparameter settings in the experiment are shown in Table 2. The first five are for FL algorithm simulation experiments. The first parameter is the number of local iterations, which refers to the number of iterations of each FL node when training the FL model. The second parameter is the local batch, which represents the number of batches into which the local dataset of the FL node is divided. During training, the dataset is trained batch by batch. The third parameter refers to the number of samples in each batch. The fourth parameter refers to the learning rate of FL training. The fifth parameter indicates that three nodes are selected each time to publish transactions.

Moreover, α, β, γ are the hyperparameters in the multi-factor evaluation scoring method. These three parameters are for the convenience of adjusting the impact of accuracy, communication overhead, and number of approvals on the tip selection algorithm. The ninth parameter is the value of the fixed k value when performing the simulation experiment. The tenth parameter is the maximum number of clusters in formula (13), and the eleventh parameter is the minimum number of clusters in formula (13). The maximum number of clusters is set to 1024, because generally the deep network model parameters are represented by floating-point numbers, 32-bit or 64-bit. In the model quantification method of this paper, the maximum number of clusters is set to 1024, which means that the maximum number of quantized model parameters is represented by 10 bits, the minimum number of clusters is 4, and the minimum number of quantized model parameters is represented by 2 bits.

Table 2. Training hyperparameter settings.

Parameter	Set Up
Local epochs	3
Local batches	32
Batch size	32
Learning rate	0.001
Client per round	10
Choose the number of tips per round	3
α	60
β	35
γ	5
Fixed k value	514
$\max_k$	1024
$\min_k$	4

5.1.3. Results Analysis

Training accuracy and loss are two standard metrics for evaluating model performance. Accuracy is the percentage of correct predictions, and loss is the cross-entropy loss. First, we assess the average accuracy and loss of the handwriting recognition task on the test dataset. The evaluation results are shown in Figures 5 and 6.

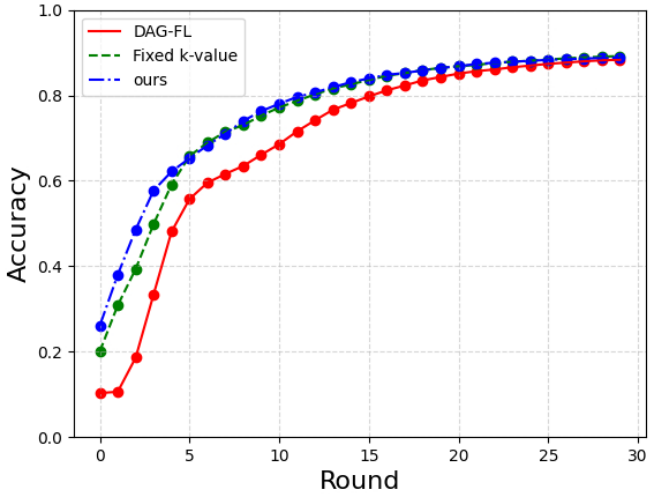

Figure 5. The average training accuracy of the three different methods on the MNIST dataset.

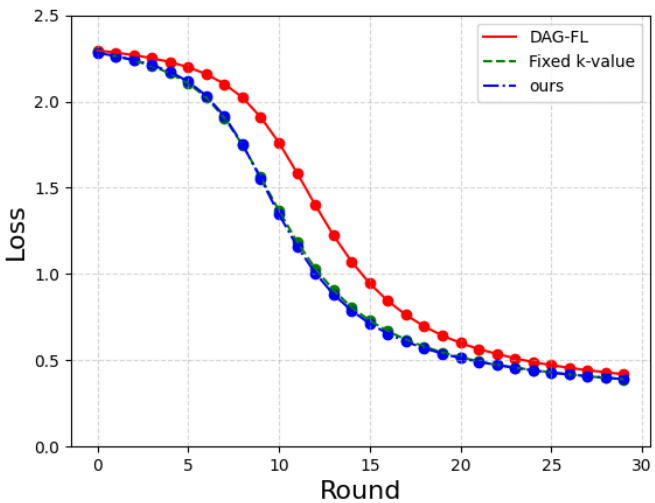

Figure 6. The average training loss of the three different methods on the MNIST dataset.

DAG-FL represents the scheme without quantization, and the Fixed k-value represents the scheme with a fixed k-value. The evaluation results show that the adaptive quantization scheme achieves similar accuracy and approximate loss as the baselines, demonstrating its applicability. In addition, compared to the two baselines, our adaptive quantization scheme achieves the targeted accuracy and loss faster and more rapidly.

To measure the communication overhead during the whole FL process, by using Equation (7) in Section 4 and Equation (12) in Section 5, we can calculate the total communication volume (transmission data volume) of the three quantization schemes in federated learning, that is, the number of bits transmitted by the models in MB, as shown in Figure 7.

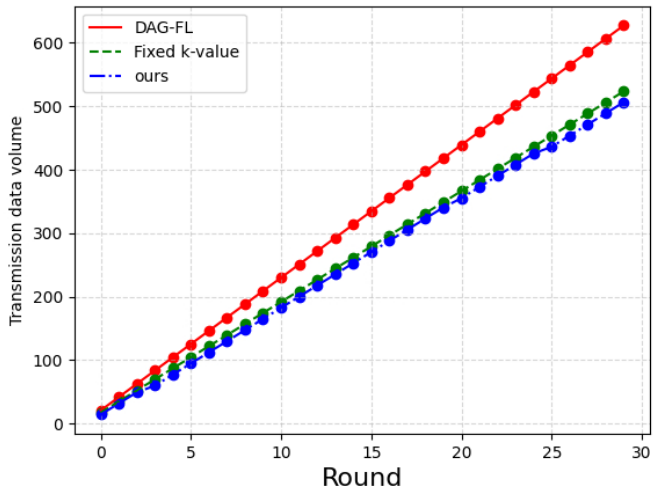

Figure 7. Evaluation of the overall total communication volume for the three quantization schemes.

From Figure 7, we can learn that in federated learning, as the number of rounds increases, compared to the two baselines, our adaptive quantization scheme reduces the total amount of total communication and reduces the communication overhead, especially for the baseline without quantization. From Figures 5–7, it can be seen that with the increase of iterations, the degree of reduction in the communication overhead of the CDAG-FL algorithm gradually decreases. Therefore, we believe that in each round of iteration, the method of obtaining the optimal k value needs to be optimized. More advanced methods can be considered to dynamically obtain the optimal k value, such as reinforcement learning. The detailed experimental data comparison is shown in Table 3.

Table 3. Experimental results of model quantization compression on MNIST dataset on CNN.

Method	Acc (%)	Acc ↑ (%)	Transmission Bits (MB)	Transmission Bits ↓ (%)
DAG-FL (Baseline)	88.32	0	627.29	0
Fixed k-value	89.19	0.98	527.44	15.91
Ours	88.82	0.56	508.86	16.07

Based on the quantization of the model for the tips selection algorithm, we have carried out an experimental comparison of the two schemes, as shown in Figure 8, where SDAG-FL represents the typical accuracy-based tips selection scheme.

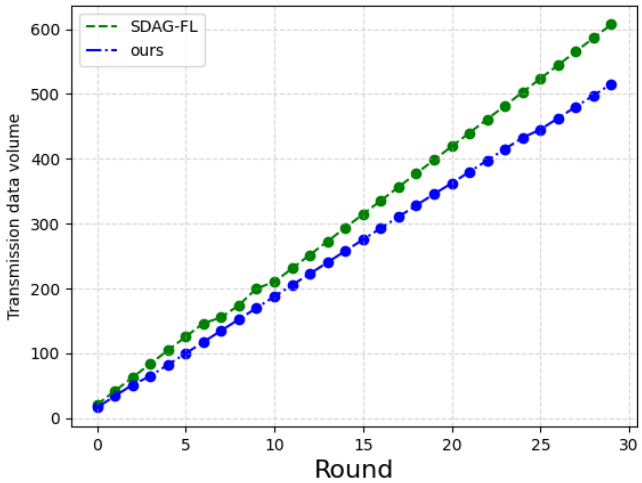

Figure 8. Evaluation of the overall total communication volume of the two tips selection algorithms.

Through Figure 8, we can see that as the number of rounds increases, relative to the baseline, our grading-based tips selection scheme reduces the total amount of communication in the system and reduces the communication overhead of the system, and its effect is more obvious as the number of rounds increases.

6. Conclusions

This paper proposes an efficient communication federated learning framework based on the DAG blockchain (CDAG-FL). The CDAG-FL system overcomes the problem of excessive communication overhead in existing blockchain-based FL systems. In the FL model uploading phase, this paper proposes a k-means-based adaptive model quantization scheme to quantize and compress the FL models. In addition, we design an improved tips selection algorithm during the DAG-based consensus process and propose a grading-based tips selection mechanism. Finally, we conducted simulation experiments to prove that the proposed scheme can reduce the total amount of system communication by approximately 21.6% compared with the baseline method.

7. Discussion

This paper proposes an efficient federated learning framework (CDAG-FL) based on DAG blockchain, in which the model compression method and tip selection algorithm based on optimal k-value clustering have certain limitations. In this section, we discuss how to develop the CDAG-FL system from two aspects, giving directions for future work.

7.1. Selection of the Optimal k Value

From the simulation results, it can be seen that with the iteration increases, the space for the proposed scheme to reduce the communication cost decreases. The reason for this is that, in each round, the formula for dynamically obtaining the optimal k value has yet to be optimized. More advanced methods can be considered to dynamically and intelligently obtain the optimal k value, such as reinforcement learning.

7.2. Tip Selection Algorithm with Multi-Factor Evaluation

In this paper, the three values of model size, accuracy, and number of batches are used to evaluate the tip of a certain transaction to improve the consensus efficiency of the system. However, hyperparameter settings in the tip selection algorithm need to be manually set empirically. In future work, more automated methods can be used to control these factors, and more factors can be introduced to improve the efficiency of the system.

Author Contributions: Conceptualization, T.L. (Tong Li) and C.Y.; methodology, T.L. (Tingting Li); software, T.L. (Tingting Li); validation, L.W., H.Z., and J.C.; formal analysis, J.C.; investigation, C.Y.; resources, J.C.; data curation, T.L. (Tong Li); writing—original draft preparation, T.L. (Tingting Li); writing—review and editing, J.C.; visualization, T.L. (Tong Li); supervision, C.Y.; project administration, T.L. (Tong Li); funding acquisition, H.Z. All authors have read and agreed to the published version of the manuscript.

Funding: This research was funded by the Research program of State Grid Liaoning Electric Power Research Institute OF FUNDER grant number 2022YF-97 (SGLNDK00NYJS2200142).

Data Availability Statement: The data provided in this study are available on the official website of the MNIST dataset at http://yann.lecun.com/exdb/mnist/.

Conflicts of Interest: The authors declare no conflict of interest.

References

1. Cui, L.; Yang, S.; Chen, Z.; Pan, Y.; Xu, M.; Xu, K. An Efficient and Compacted DAG-Based Blockchain Protocol for Industrial Internet of Things. *IEEE Trans. Ind. Inform.* **2020**, *16*, 4134–4145. [CrossRef]
2. Custers, B.; Sears, A.M.; Dechesne, F.; Georgieva, I.; Tani, T.; Van der Hof, S. *EU Personal Data Protection in Policy and Practice*; T.M.C. Asser Press: The Hague, The Netherlands, 2019; Volume 29.
3. Gaff, B.M.; Sussman, H.E.; Geetter, J. Privacy and Big Data. *Computer* **2014**, *47*, 7–9. [CrossRef]
4. Qi, Y.; Hossain, M.S.; Nie, J.; Li, X. Privacy-preserving blockchain-based federated learning for traffic flow prediction. *Future Gener. Comput. Syst.* **2021**, *117*, 328–337. [CrossRef]
5. Lu, Y.; Huang, X.; Dai, Y.; Maharjan, S.; Zhang, Y. Blockchain and Federated Learning for Privacy-Preserved Data Sharing in Industrial IoT. *IEEE Trans. Ind. Inform.* **2020**, *16*, 4177–4186. [CrossRef]
6. Li, Y.; Cao, B.; Peng, M.; Zhang, L.; Zhang, L.; Feng, D.; Yu, J. Direct Acyclic Graph-Based Ledger for Internet of Things: Performance and Security Analysis. *IEEE/ACM Trans. Netw.* **2020**, *28*, 1643–1656. [CrossRef]
7. Beilharz, J.; Pfitzner, B.; Schmid, R.; Geppert, P.; Arnrich, B.; Polze, A. Implicit Model Specialization through DAG-Based Decentralized Federated Learning. In Proceedings of the Middleware'21: 22nd International Middleware Conference, Québec City, QC, Canada, 6–10 December 2021; pp. 310–322. [CrossRef]
8. Schmid, R.; Pfitzner, B.; Beilharz, J.; Arnrich, B.; Polze, A. Tangle Ledger for Decentralized Learning. In Proceedings of the 2020 IEEE International Parallel and Distributed Processing Symposium Workshops (IPDPSW), New Orleans, LA, USA, 18–22 May 2020; pp. 852–859. [CrossRef]
9. Cao, M.; Zhang, L.; Cao, B. Toward On-Device Federated Learning: A Direct Acyclic Graph-Based Blockchain Approach. *IEEE Trans. Neural Networks Learn. Syst.* **2023**, *34*, 2028–2042. [CrossRef] [PubMed]
10. Zhu, L.; Liu, Z.; Han, S. Deep Leakage from Gradients. *arXiv* **2019**, arXiv:1906.08935.
11. Zhao, B.; Mopuri, K.R.; Bilen, H. idlg: Improved Deep Leakage from Gradients. *arXiv* **2020**, arXiv:2001.02610.
12. Geiping, J.; Bauermeister, H.; Drge, H.; Moeller, M. Inverting Gradients—How Easy Is It to Break Privacy in Federated Learning? *arXiv* **2020**, arXiv:2003.14053.
13. Yang, W.; Yang, Y.; Dang, X.; Jiang, H.; Zhang, Y.; Xiang, W. A Novel Adaptive Gradient Compression Approach for Communication-Efficient Federated Learning. In Proceedings of the 2021 China Automation Congress (CAC), Beijing, China, 22–24 October 2021; pp. 674–678. [CrossRef]
14. Luo, P.; Yu, F.R.; Chen, J.; Li, J.; Leung, V.C.M. A Novel Adaptive Gradient Compression Scheme: Reducing the Communication Overhead for Distributed Deep Learning in the Internet of Things. *IEEE Internet Things J.* **2021**, *8*, 11476–11486. [CrossRef]
15. Xue, X.; Mao, H.; Li, Q.; Huang, F.; Abd El-Latif, A.A. An Energy Efficient Specializing DAG Federated Learning Based on Event-Triggered Communication. *Mathematics* **2022**, *10*, 4388. [CrossRef]
16. Chen, M.; Yang, Z.; Saad, W.; Yin, C.; Poor, H.V.; Cui, S. A Joint Learning and Communications Framework for Federated Learning Over Wireless Networks. *IEEE Trans. Wirel. Commun.* **2021**, *20*, 269–283. [CrossRef]
17. Han, S.; Mao, H.; Dally, W.J. Deep compression: Compressing deep neural networks with pruning, trained quantization and huffman coding. *arXiv* **2015**, arXiv:1510.00149.