Electronic Research Archive

ERA, 32(3): 1731–1748.
DOI: 10.3934/era.2024079
Received: 29 November 2023
Revised: 29 January 2024
Accepted: 03 February 2024
Published: 27 February 2024

http://www.aimspress.com/journal/ERA

Research article

Research on incentive mechanisms for anti-heterogeneous federated learning based on reputation and contribution

Xiaoyu Jiang, Ruichun Gu* and Huan Zhan

School of Information Engineering, Inner Mongolia University of Science and Technology, Baotou 014010, China

* **Correspondence:** Email: reachcool@imust.edu.cn.

Abstract: An optimization algorithm for federated learning, equipped with an incentive mechanism, is introduced to tackle the challenges of excessive iterations, prolonged training durations, and suboptimal efficiency encountered during model training within the federated learning framework. Initially, the algorithm establishes reputation values that are tied to both time and model loss metrics. This foundation enables the creation of incentive mechanisms aimed at rewarding honest nodes while penalizing malicious ones. Subsequently, a bidirectional selection mechanism anchored in blockchain technology is developed, allowing smart contracts to enroll nodes with high reputations in training sessions, thus filtering out malicious clients and enhancing local training efficiency. Furthermore, the integration of the Earth Mover's Distance (EMD) mechanism serves to lessen the impact of non-IID (non-Independent and Identically Distributed) data on the global model, leading to a reduction in the frequency of model training cycles and an improvement in model accuracy. Experimental results confirm that this approach maintains high model accuracy in non-IID data settings, outperforming traditional federated learning algorithms.

Keywords: federated learning; blockchain; incentive mechanism; reputation; EMD

1. Introduction

With the rapid development of computing and storage capabilities, IoT devices are now widely used in several fields, and machine learning is widely used in industries such as industrial production, autonomous driving, healthcare, and retail [1–4]. Federated Learning (FL) [5] has emerged to fully

utilize the data of each enterprise for model training to obtain more accurate results while protecting data privacy. Federated learning, as an effective method to solve the problem of data silos [6], can make full use of data stored on different devices for model training without pooling the data.

Although federated learning offers numerous benefits, it is not without its challenges. First, the conventional federated learning architecture typically comprises a central server and numerous distributed devices. The aggregation of the global model in the FL process is carried out by a central server. A single central server is very fragile and has the potential for a single point of failure [7], and the federated learning process will be forced to be interrupted if it is attacked. In contrast, blockchain technology is decentralized, immune to the failure of any single node, and maintains data that is not controlled by any single entity. Consequently, replacing traditional central aggregation servers with blockchain integration can eliminate the risk of a single point of failure and defend against malicious attacks. Second, in traditional federated learning, the quality of the global model is influenced by the local data held by these clients. Given that real-world data is highly correlated, non-IID (non-Independent and Identically Distributed) [8] data with varying probability distributions are a common challenge in federated learning. These data distribution and characteristic discrepancies across participants can hinder the federated learning model's training and impact the accuracy of the model. Last, the widely used Gradient Average Aggregation Algorithm (FedAvg) fails to consider the individual contributions of clients in the training process. This oversight necessitates multiple training rounds to achieve the desired accuracy, thereby increasing training time and diminishing the efficiency of model training. In practical scenarios, the time allotted for model training by the task publisher is limited, and for those requiring urgent results, exceeding this time frame renders the outcomes meaningless. Hence, there is a pressing need to optimize the federated learning training model to enhance efficiency.

In this paper, based on existing research, a client incentive mechanism under the blockchain framework is designed. First, blockchain technology is used to solve the single-point-of-failure problem. Second, the efficiency of model training is improved by designing the incentive mechanism and optimizing the data quality. Considering the dynamic nature of model accuracy, this paper designs a reputation value calculation system related to the completion time, and gives clients rewards based on their specific contributions to each round of training. Furthermore, it is based on the characteristics of edge nodes with low latency and high storage, to enhance the efficiency of local training; Finally, the EMD mechanism is introduced, by calculating the EMD distance of each client, it can eliminate the nodes with too large EMD value in time, to effectively reduce the influence of Non-IID data on the global model.

The major contributions of this paper are as follows:

1) Design a reputation value mechanism related to both model loss and time. Based on this reputation value, a reward mechanism is established to incentivize clients with high-quality data and high performance to join the training, and only honest and efficient clients can get high rewards.

2) The EMD mechanism is adopted, through the calculation of EMD distance can filter out the nodes with higher data quality, to improve the accuracy of the global model as well as the efficiency of model training.

3) Finally, it is proved by experiments that this scheme can improve the accuracy of the global model in the Non-IID environment compared with the traditional FL method, and it can reach convergence in a shorter time, which improves the efficiency of the federated learning process.

2. Related work

In 2016, Google defined federated learning as a distributed training paradigm for machine learning, which does not require the pooling of data and realizes model training by passing only parameters between models in each round of training. This advantage of federated learning makes it widely used in fields such as self-driving cars, innovative healthcare, and earthquake prediction [9] to solve the problem of dispersed and private data. Although it has many advantages, it inevitably has some issues.

First, due to the single point of failure, the centralized federated learning server has been increasingly challenged and questioned. To solve the single point of failure of the centralized server, the combination of federated learning and blockchain systems has become an emerging research hotspot. Blockchain, with its high stability and security, has become the best solution to solve the problem of single point of failure under the federated learning framework. Rahman et al. [10] proposed the BlockFlow architecture, which initially implements responsible and privacy-preserving FL through a contribution-scoring procedure. Shao et al. [11] proposed a blockchain-assisted decentralized federated learning, which can well solve the single-point-of-failure problem that exists in the traditional FL system but does not solve the risk of privacy leakage. Wu et al. [12] designed a blockchain-based chain framework, which is decentralized and can operate normally without a master node with good privacy and robustness. Zhang et al. [13] proposed a hierarchical federation learning algorithm based on dynamic weight and a hierarchical federation learning algorithm based on time-effective aggregation. They integrated edge computing technology into the federated learning box, which alleviated the problem of heavy redundancy in the communication process, and improved the convergence speed and training efficiency.

Second, due to the self-interest of clients, clients with high-quality data are reluctant to participate in model training, which will affect the accuracy and rounds of training and reduce the efficiency of model training [14]. To solve this problem, Francis et al. [15] proposed an EOS[1]-based FL framework to record and reward the contributions of federated learning participants, which uses off-chain to directly store the gradient values of local model training, and on-chain to store the hash values of the gradient values obtained from local model training to prevent data tampering. However, the reward size is only based on the amount of data consumption, which is not proven to be fair and reliable. Liu et al. [16] introduced Shapley values[2] into joint learning to calculate the motivation of each joint learning member, but did not consider the limitations of Shapley value calculation. In federated learning, the data leakage risk of each participating data holder is different, which is unequal and does not take into account the historical integrity of FL participants, which is also an important factor in successfully constructing federated learning. Xiong et al. [17]. designed an efficient multi-weight subjective logic model that can improve the reliability of selecting work nodes based on reputation values and interaction history. Guo et al. [18] proposed a federal learning incentive mechanism based on the principle of compound interest. In the proposed plan, different interest rates were set for different types of participants to ensure that the more costs paid, the greater the benefits obtained. In addition, to suppress malicious participants, the concept of joint behavior tolerance is proposed to kick out participants whose malicious behavior exceeds the tolerance level.

[1] EOS blockchain, developed to promote efficient and scalable decentralized applications

[2] By considering the contributions made by each agent, the cooperative benefits are fairly distributed. The Shapley value of agent *i* is the average expected contribution of *i* to a cooperative project

Finally, real-life heterogeneous data can likewise seriously jeopardize the accuracy of the FL global model. Considering the impact on model accuracy and convergence speed under Non-IID data, in 2017, an improved algorithm called FedAvg [19] was proposed, which allows clients to perform several rounds of SGD synchronously before uploading the model to a central server for model aggregation, thus effectively reducing the number of communication rounds and thus improving the efficiency of joint learning. However, Zhao et al. [20] found that the performance and efficiency of FedAvg decreased significantly with increasing statistical heterogeneity. Moreover, the relationship between EMD of statistical heterogeneity was also determined, showing that EMD can be used as an ideal indicator of statistical heterogeneity. In addition to this, optimizing the working client selection method can likewise reduce the impact of Non-IID data on the global model. Luo et al. [21] designed a client sampling strategy to reduce the total training time. Their approach provides a new upper bound on the convergence of arbitrary client selection probabilities and generates a non-convex training time minimization problem. Wu et al. [22] identify the optimal subset of local model updates by excluding unfavorable local updates and propose a probabilistic node selection framework, FedPNS, which is based on the device's contribution to the model associated with the data distribution determined using the outputs of the aggregation algorithm contribution to dynamically adjust the selection probability of the client. Chen et al. [23] proposed an indicator to measure the heterogeneity of client data and training performance, called cumulative model strength. Clients with lower cumulative model strength or poorer training performance are more likely to be selected, allowing the federated model to learn global statistical knowledge faster. Lv et al. [24] proposed a new federated learning algorithm FedRDS, which adds regularization terms to the local loss function of the client to gradually approach the global model, thereby minimizing the impact of client drift.

Aiming at the above research, this paper combines the above three aspects and designs a scheme to optimize the global model from multiple aspects to improve the model training efficiency. Using the decentralization of blockchain as well as the feature of tampering, it replaces the central service area to solve the single-point-of-failure problem; through the incentive mechanism based on reputation value, it attracts more honest and reliable clients to join the model training process; and finally, using the EMD mechanism, it eliminates the clients with too much heterogeneity of the data, to reduce the impact of Non-IID data on the global model.

3. Model framework

The system model is shown in Figure 1, which consists of 4 major parts, which are blockchain, edge node, work node, and task publisher.

1) Blockchain. Deploy smart contracts, responsible for selecting work nodes, selecting suitable clients as work nodes based on their reputation values and data heterogeneity, and participating in federated learning model training; Second, distribute rewards based on the performance of work nodes during training; In addition, as nodes in the blockchain, federated learning participants aggregate the global model on the chain.

2) Edge nodes. Verify the accuracy contribution of the model. The work node sends the locally trained model updates to the edge node. Edge nodes validate ratings evaluate contributions, and then upload model updates to the blockchain.

3) Work nodes. Work nodes use their private data to train global models and generate model updates. Then submit the model updates to the edge layer for evaluation and validation.

4) Task Publisher. Publish training tasks and pay compensation to the working nodes.

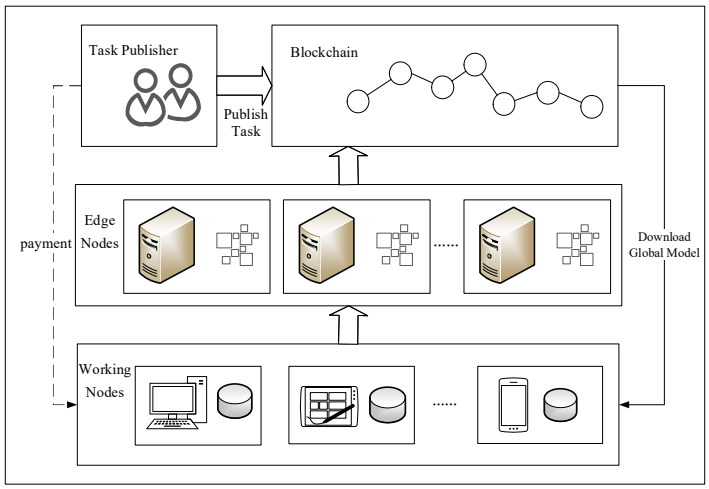

Figure 1. Model framework diagram.

4. Schematic design

In this scheme, first, a reputation value mechanism based on time and model loss degree is designed, and then an incentive scheme is designed on this basis, which can attract more high-quality clients to join the FL model training and only efficient and honest nodes can be rewarded; finally, an EMD distance mechanism is adopted, which can timely eliminate the clients with too high data heterogeneity, and reduce the impact of Non-IID data on the global model. The incentive mechanism process of this plan is shown in Figure 2, mainly divided into the following six parts.

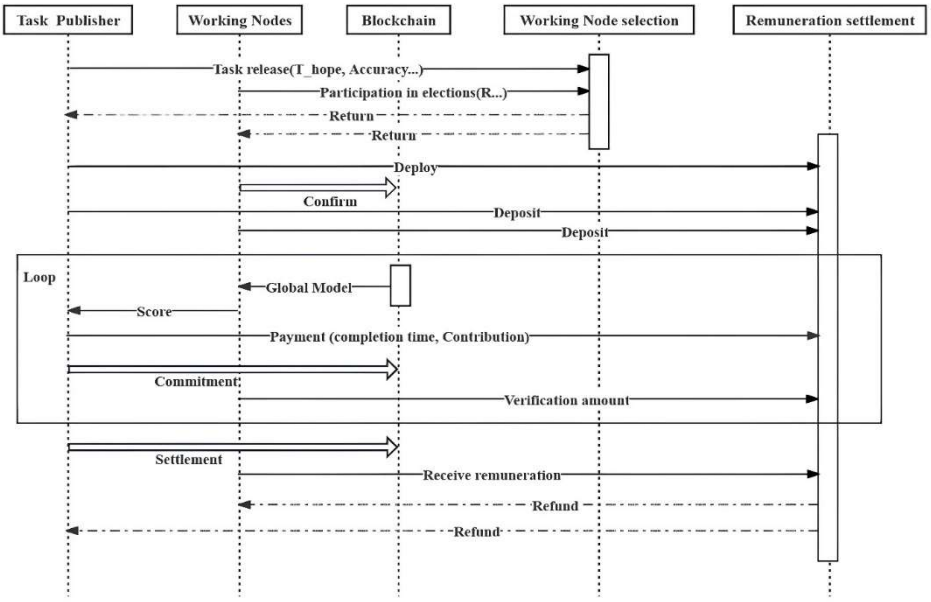

Figure 2. Incentive mechanism flowchart.

1) Task release. The task publisher releases the training task and releases the reward rules and resource requirements (including the size of the local data to join the training, the time threshold for local training, the type of data, the final accuracy of the training needs to be achieved, the desired training time); the client judges whether it can profit from the client according to its actual situation and chooses whether to sign a contract or not, and both parties need to pay a deposit for signing a contract.

Algorithm 1. Federated learning process

Inputs: Distributed client list $Client = \{D_1, D_2, \cdots, D_N\}$, local small sample size B, iteration rounds E, learning rate η

Output: global model ω

Task publisher releases FL training tasks
// Select working nodes
Select M clients to form a working node set *WorkN*[].
Initialize the global model ω_0
for each round t = 0, 1, 2, … do
    Client downloads the latest global model ω_t
    for $node \in WorkN[]$ do
        $\omega_{t+1}^k \leftarrow$ Client-side local updates (k, ω_t)
    end for
    Send ω_{t+1}^k to edge node
    // Edge node validation model update
    if Edge nodes have sufficient computing resources
        Calculate the model accuracy contribution *Cont*
            $Cont_t = Loss_{t-1} - Loss_t / Loss_{t-1}$
    else
        Unloading computing tasks
    end if
        Payment by the task publisher
    Uploading model updates to the blockchain
    // Chain polymerization
    $\omega_{t+1} = \sum_{K \in U_t} \frac{n_k}{n} \omega_{t+1}^k$
end for
Update the reputation value of a worker node R
Task publisher submits the settlement transaction
Getting paid for work nodes & & Refunding deposits to both parties

2) Work node selection. Clients seeking to participate in FL model training are required to submit their data resource details, encompassing the size of the data volume, the type of data, and reputation value information, and the smart contract deployed on the chain calculates the EMD distance of the client. It then selects the most appropriate client based on this assessment, designating them as a working node to integrate into the FL training process.

3) Submitting deposit. The client elected as the working node needs to return a confirmation

transaction after the task publisher is subjected to the confirmation transaction, and both parties submit the deposit, after which the model training process begins.

4) Model training. The working node downloads the latest global model and starts local training, after completion, it submits the model update to the edge node for verification and scoring, the model update verified by the edge node is packaged and uploaded to the chain, and when the model update on the blockchain reaches the threshold, it triggers the model aggregation work, generates a new global model, and starts a new round of FL process.

5) Reputation value update. After each round of training, the edge nodes validate and score the model updates submitted by the working nodes.

6) Remuneration settlement. After each model training round, upon receiving the validation results, the task publisher remunerates the blockchain rather than paying directly to the participating node. The task publisher initiates a payment transaction to the blockchain following the completion of the payment. This transaction allows the working node to verify the accuracy of the payment amount. Once the model training task is finished, the task publisher submits a settlement transaction. The working node can then collect all due compensation through this settlement transaction. Only after the working node's settlement transaction is processed will both parties' deposits be returned.

The scheme process is shown in Algorithm 1.

4.1. EMD calculation

EMD distance is a probability distribution-based measure of distance, a method for evaluating the similarity between two multidimensional distributions in the feature space, and is commonly used as a measure of image similarity in image retrieval. In general, there are many features in an image, and their distribution can be represented by a set of clusters, where each cluster is represented by its mean value and the percentage of the distribution belonging to the cluster, i.e., the weight, and this representation is called the signature of the distribution of image features. Let S be a signature, M a feature, and W the weight of the feature; then, the signature can be written as S = (W, M). Different signatures have different sizes, such as the signature of a simple distribution is shorter than the signature of a complex distribution.

Let $P=\{(p1,wp1), (p2,wp2) \cdots (pm,wpm)\}$ be a signature with m clusters, $Q=\{(q1,wq1), (q2,wq2) \cdots (qn,wqn)\}$ a signature with n clusters, $D=[dij]$ is the distance between individual features, and each dij term denotes the distance from pi and qj, so D is an M*N matrix. For P and Q, let the distance flow matrix be $F=[fij]$, where each fij term denotes the number of flows pi from to qj, then the EMD distance can be further transformed into a linear programming problem, i.e., to find a flow in the matrix F that minimizes the global cost of going from P to Q. The formula is as follows:

$$WORK(P,Q,F)=\sum_{i=1}^{m}\sum_{j=1}^{n} f_{ij}d_{ij} \ . \tag{1}$$

And the formula needs to comply with the following constraints:

$$\begin{cases} f_{ij} \geq 0, \ 1 \leq i \leq m, \ 1 \leq j \leq n, & (2) \\ \sum_{i=1}^{m} f_{ij} \leq q_i, \ 1 \leq j \leq n, & (3) \\ \sum_{j=1}^{n} f_{ij} \leq p_i, \ 1 \leq i \leq m \ , & (4) \\ \sum_{i=1}^{m}\sum_{j=1}^{n} f_{ij} = \min\left(\sum_{i=1}^{m} p_i \ , \sum_{j=1}^{n} q_i\right), & (5) \end{cases}$$

where Eq (2) restricts the direction of the flow F to flow from P to Q, but not vice versa; Eqs (3) and (4) stipulate that the flow of f cannot be greater than the number of p, q; and Eq (5) is to ensure that the distance of the flow is as small as possible. Therefore, the EMD distances of two different distributions P, Q are shown in Eq (6):

$$EMD(P,Q) = \frac{\sum_{i=1}^{m}\sum_{j=1}^{n} f_{ij} d_{ij}}{\sum_{i=1}^{m}\sum_{j=1}^{n} f_{ij}}. \tag{6}$$

According to the calculation method of the Earth Mover's Distance (EMD) mentioned above, this scheme defines EMD_k as the measure of the heterogeneity of data on client k. It quantifies the difference between the local data distribution q_k and the global data distribution p. The specific calculation method is as follows:

$$EMD_k = \inf_{\gamma \sim (p, q_k)} E_{(x,y)\sim\gamma}[\| x - y \|_2].$$

In the above formula, (p, q_k) represents the set of possible joint distributions of distribution p and q_k, γ Represents a specific joint distribution of (p, q_k). Afterwards, regarding the joint distribution γ Perform sampling, represented by x and y, and calculate the distance between the two. Afterwards, calculate the expected distance $E_{(x,y)\sim\gamma}[\| x - y \|_2]$ between the joint distribution samples, and finally, take the smallest expected value as the EMD distance of the distributions p and q_k.

4.2. Select working nodes

Upon the task publisher posting a model training task, interested clients submit their attribute information to the task publisher. The task publisher then selects clients to participate in the FL process based on their requirements, with the chosen clients designated as working nodes. Given that clients possess varying attributes, such as data types and volumes, this results in a significant disparity in performance and data heterogeneity among clients. By assessing client attributes to identify suitable working clients, this scheme effectively pinpoints more honest and reliable high-performance nodes. Consequently, it diminishes the wait time for participants, enhances the iteration pace, mitigates the risk of malicious attacks, and bolsters the system's security. The specific process is outlined as follows.

1) Publishing tasks. The task publisher submits task requirements (including data volume size, time threshold for local training, data type, compensation information, etc.).

2) Bidding. Clients that meet the requirements decide whether to participate in the bidding based on the local training overhead submission, and clients that wish to join the FL model training submit their data resource information. According to self-interest, clients will not participate in the bidding when the overhead is larger than the reward received, so some low-performance nodes will be eliminated, and it can constrain the task publisher so that it cannot mark too low a reward amount.

3) Determining the winner. The task publisher selects the working nodes to participate in the FL model training based on the attribute information submitted by the clients and the data heterogeneity level of each client.

$$\begin{cases} U_i = \alpha \cdot DataV_i + \beta \cdot rep_i, \ \ EMD_i \le K \\ U_i = 0, \ \ EMD_i > K \\ \alpha + \beta = 1 \end{cases}.$$

In the equation, Ui denotes the priority level at which a client is elected, $DataV_i$ denotes the data volume scale information of client i, rep_i denotes the reputation value of client i, and EMD_i denotes the data offset level of client i. When the EMD level is greater than K, the client will not be elected, and when the EMD level is lower than K, the client's priority is determined by the data volume scale as well as the reputation value together, and α and β denote the weights of the two, and the task publisher is capable of dynamically adjusting the values of α and β in case of different demands.

Algorithm 2. Select working nodes

Input: client list $Client = \{D_1, D_2, \ldots, D_n\}$, task publisher TP

Output: working nodes list $WorkN[] = [W_1, W_2, \ldots, W_m]$

1: if the number of candidate clients is less than N

2: return FAIL

3: end if

4: // client submits attribute information

5: for $D_i \in Client[]$

6: send $D_i[DataV_i, rep_i]$

7: end for

8: TP calculation Priority U_i

9: eliminate clients with excessive data heterogeneity

10: sort the remaining clients according to priority

11: take the first M clients to form a list of working nodes $WorkN[]$

4) Local training. The selected work node downloads the latest global model and uses its private dataset for local model training. After completion, the model updates are uploaded to the edge layer for contribution evaluation by the edge node. Subsequently, the edge nodes package the model updates onto the chain. The aggregation model work is carried out on the blockchain, and the aggregation formula is shown below to generate a new global model and start the next iteration.

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

To mitigate the risk of malicious clients compromising the accuracy and efficiency of the model during training, we propose a threshold and penalty framework. Participants are required to place a security deposit. In the event that a client introduces subpar data or deliberately prolongs training sessions, thus negatively impacting model training performance, their deposit will be forfeited, and they will be ineligible for any additional rewards. This strategy aims to minimize the costs borne by task publishers, while preventing malicious clients from entering the training process, and deterring clients from extending training time improperly.

The specific process is shown in Algorithm 2.

4.3. Reputation mechanism

Federated learning is a type of distributed machine learning commonly used for knowledge acquisition to improve global models collaboratively to protect user data, and model performance improvement is as important as security improvement. Unreliable workers may launch malicious attacks and send malicious updates, so reputation calculation is needed to select honest working nodes. Nodes below a particular threshold are not allowed to participate in the federated learning process. This scheme analyses the forward and reverse aspects [25] and combines both parts to determine the final reputation value of the working node.

Reputation assessment is categorized into positivity, accuracy contribution, and timeliness.

(i) Positivity: It is expressed by the fact that the higher the positivity, the more the interaction, and according to the content of the exchange, it is divided into positive and negative interactions. The calculation process is shown in Eqs (7) and (8).

$$Intp = \frac{Hp}{H}, \tag{7}$$

$$Intn = \frac{Hn}{H}. \tag{8}$$

The formula H indicates the total number of interactions, $Intp$ the proportion of positive interactions, and $Intn$ the proportion of negative interactions.

(ii) Accuracy contribution: The accuracy contribution is denoted by $Cont$. The degree of improvement in the accuracy of the model is regarded as another factor affecting the reputation. Assuming that the loss of the model before and after the t th interaction is denoted as $Loss_{t-1}$ and $Loss_t$, respectively, the accuracy contribution of the working node at the t th interaction is calculated as shown in Eq (9):

$$Cont_t = \frac{Loss_{t-1} - Loss_t}{Loss_{t-1}}. \tag{9}$$

When the model loss is reduced, $Loss_{t-1} > Loss_t$, indicates that the updated model is useful. $Cont > 0$, made a positive contribution; Conversely, as model losses increase, $Loss_{t-1} < Loss_t$, indicates a negative contribution.

(iii) Timeliness: timeliness is denoted by *TimeL*. As the training process advances step by step, the model is constantly updated, the contribution of the model accuracy to the reputation value is not always the same, and it is considered that the contribution of the more recent model accuracy should occupy a higher weight. Then the timeliness is calculated as shown in Eq (10):

$$TimeL = \sum_{t=1}^{H} Cont_t \times e^{-(T_c - T_t)}, \tag{10}$$

where T_c is the current time and T_t is the time of the t th sharing of the local model. This is a weighting function where model contribution is considered as a weighting factor, jointly considering the importance of model contribution and timeliness. Interaction timeliness can also be categorized into positive and negative values as shown in Eqs (11) and (12):

$$TimeLp = \sum_{t=1}^{Hp} Contp_t \times e^{-(T_c - T_t)}, \tag{11}$$

$$TimeLn = \sum_{t=1}^{Hn} Contn_t \times e^{-(T_c - T_t)}, \tag{12}$$

where *TimeLp* and *TimeLn* denote the positive model accuracy contribution and the negative model accuracy contribution.

Considering the above factors, then the overall reputation is calculated as shown in Eq (13):

$$R = Intp \times TimeLp + Intn \times TimeLn, \tag{13}$$

where R denotes the overall reputation, the positive behavior of the end node will increase the value of $Intp \times TimeLp$, which is always not less than 0. On the contrary, $Intn \times TimeLn < 0$, so the more negative behavior, the larger the absolute value of $Intn \times TimeLn$. Combine these two components to calculate the reputation value of the end node.

4.4. Incentive mechanism

When a task publisher releases a model training task, it will attach task requirements, including the desired training time for each round *T_hope*, the desired accuracy, and the reputation value. The client submits an application based on its attribute conditions, and after both parties have decided, it needs to pay a deposit. According to rational analysis, to prevent malicious task publishers from cheating the deposit of working nodes, the deposit paid by task publishers is defined to be much larger than the deposit of working nodes.

Following the completion of one iteration cycle, the worker node is required to upload the model update to the edge node layer for contribution scoring. The worker node then submits the final scoring result to the task publisher. The task publisher compensates the worker node based on its contribution value and completion time but does not disburse the payment directly. Instead, the task publisher initiates a promise-to-pay transaction, which the worker node receives to verify the accuracy of the proposed payment amount. Upon the completion of the FL task, the task publisher sends a payment settlement transaction. Using this transaction, the worker node can claim the full payment due. Only

after the worker node's settlement transaction is processed will both parties' deposits be returned.

The iteration time for the local update of the work node is shown in Eq (14):

$$T_i = \frac{c_i D_i}{f_i}. \tag{14}$$

In the formula, *ciDi* denotes the CPU cycles required for one iteration; f_i denotes the frequency of client *i*.

The local energy consumption of the working node is shown in Eq (15):

$$E_i(f_i) = \frac{\alpha_i}{2} c_i D_i f_i^2. \tag{15}$$

It can be obtained from Eqs (14) and (15):

$$E_i(f_i) = \frac{\alpha_i}{2} c_i D_i f_i^2 = \frac{\alpha_i}{2} \frac{(c_i D_i)^3}{T_i^2}. \tag{16}$$

Algorithm 3. Incentive Mechanism

Input：Work node list *WorkN*[]， Number of training rounds *N*
Output：Work node benefits *Income*
1： while $j \leq$ N
2： Work node training FL model
3： Submit to edge layer validation
4： if ($Loss_{j-1} < Loss_j$) // Negative contribution, deducting deposit
5： Calculate benefits $P_i^j = \max[T\_hope, t_j]Contn$
6： else if（$T\_hope > t_j$） // Positive contribution without timeout
7： Calculate benefits $P_i^j = (\frac{1}{\quad} - \frac{nope}{\quad})^2 Contp$
8： else // Positive contribution, but timeout
9： Calculate benefits $P_i^j = 0$
10： end if
11: Task publisher submits committed payment transactions
12: Model update on chain aggregation
13： j++
14： end while
15： Task publisher submits compensation settlement transactions
16： Receive compensation based on settlement transactions at work nodes
17： Refund of deposit from both parties

From Eq (16), the energy consumption of a client is inversely proportional to the square of the update iteration time [26]; thus, this scheme specifies that the payoff that client *i* receives after the completion of the *j* th round of training is shown below:

$$P_i^j = \begin{cases} (\frac{T\_hope}{t_j})^2 Contp \ , & (T\_hope > t_j) \text{ and } (Loss_{j-1} > Loss_j) \\ \max[T\_hope,\ t_j]Contn \ , & (Loss_{j-1} < Loss_j) \\ 0 \ , & (T\_hope < t_j) \text{ and } (Loss_{j-1} > Loss_j) \end{cases}, \tag{17}$$

where *tj* denotes the actual time taken by the client for a round of iteration, only the work node that completes within the specified time and has a positive contribution to the model loss can get rewarded, and when the completion time is shorter, the more rewarded the reward is; when a work node does not complete within the specified time and makes a positive contribution, the node's gain is 0 and will not be penalized; if a work node makes a negative contribution, which makes the loss of the model go up, then the node will face a penalty, and the deposit handed over will be deducted.

In summary, the final benefit of the work node is:

$$Income_i = \sum_{j=1}^{N} p_i^{\ j} - \sum_{j=1}^{N} E_i(f_i) .$$

The specific algorithmic flow of the motivation process is shown in Algorithm 3.

5. Experimental analysis

5.1. Experimental environment

Ubuntu 20.04 is used as the base operating system for the experiment, in terms of blockchain this program uses the FISCO blockchain network, which is an open-source federated blockchain platform, and through the information ports provided by it, it is possible to check the transaction as well as the details of the smart contract. Moreover, the smart contract is written using the Solidity programming language, and the learning model is implemented using Python 3.7 and TensorFlow 1.14.0 implementation. This scheme uses a CNN model for training, and working node will train the local model for 1 epoch with a learning rate of 0.05 and a batch size of 128.

The incentive framework in this scheme is divided into four parts, as shown in Figure 1 in Section 2. The first part is the end-working node layer, which is responsible for the local model update; the second part is the edge layer, which is responsible for verifying the model update as well as the block generation work; and the third part is the blockchain layer, which is responsible for the aggregation of the update and the generation of the new global model.

To mimic the data distribution of Non-IID in the real scene, take the MNIST dataset as an example, the MNIST training set of 60,000 handwritten digit images is sorted according to the labels 0 to 9 in the order from smallest to largest, and then the training set is divided into 200 slices in order, and each slice contains 300 images. The images contained in each slice are all the same digit. Distribute two slices of training data for each work node. There are only two possibilities for the training dataset to be distributed to each worker node: 600 images containing only one digit and 300 images containing each of the two digits.

5.2. Experimental results and evaluation

The relationship between data heterogeneity and model accuracy is illustrated in Figure 3.

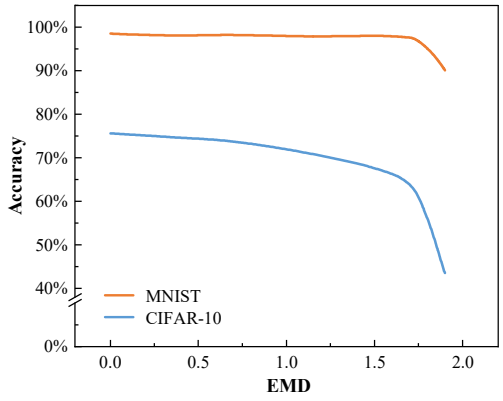

Figure 3. Effect of Non-IID data on model accuracy.

In this experiment, two datasets, MNIST and CIFAR-10, were employed. A greater EMD distance signifies a higher degree of data heterogeneity. As heterogeneity levels rise, the accuracy of the global model tends to decline. This decline is attributed to the dissimilar characteristics and distributions of data across different participants, which impedes effective data synergy and diminishes the model's generalization capacity. Consequently, the model's accuracy is significantly compromised.

To reduce the harm caused by Non-IID data, this scheme introduces the EMD mechanism, which measures the heterogeneity level of the data by calculating the EMD distance and eliminates the clients with too high EMD distance, so that it can reduce the impact of Non-IID data on the global model.

As shown in Figure 4, this paper verifies the superiority of this scheme using four datasets, namely, MNIST, EMNIST, FEMNIST, and CIFAR-10. This experiment compares this scheme with FedAvg and FedProx [27], which proves that in the Non-IID data environment, this scheme slightly outperforms both the FedAvg and the FedProx two federated learning algorithms, as this scheme can eliminate the clients with high data heterogeneity in time so that it can maintain high accuracy. Thus, this scheme can effectively reduce the harm of Non-IID data to the global model.

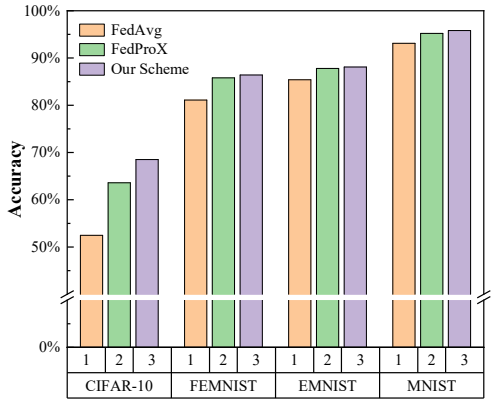

Figure 4. Comparison of accuracy with different datasets.

As shown in Figure 5, this figure shows in detail how the model accuracy changes with the number of training rounds under the CIFAR-10 dataset. In this experiment, by comparing the performance of the present scheme with that of the FedAvg algorithm, we have concluded that the present scheme is able to maintain high model performance in heterogeneous environments. As can be seen from the figure, in the Non-IID environment, the model accuracy of the proposed scheme gradually approaches the accuracy in the IID environment. This indicates that the proposed scheme can achieve better performance in practical applications, even in the case of heterogeneous data distribution. This advantage comes from the fact that this scheme makes effective use of data from different clients in the training process while achieving better model generalization across clients.

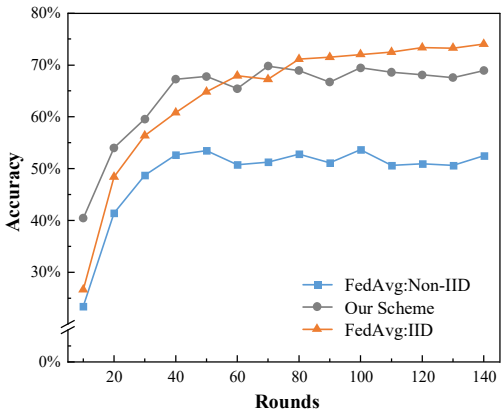

Figure 5. Comparison of our scheme with FedAvg.

As shown in Figure 6, this experiment uses the MNIST dataset and compares the training times of FedAvg, FedProx, and the present scheme when they reach different levels of accuracy in the Non-IID environment. The FedAvg algorithm with the longest convergence time. The convergence performance of the present scheme is slightly improved compared to the FedProx algorithm, and the scheme can pay remuneration incentives for the honest working nodes, thus attracting more clients to join the training process and having higher security.

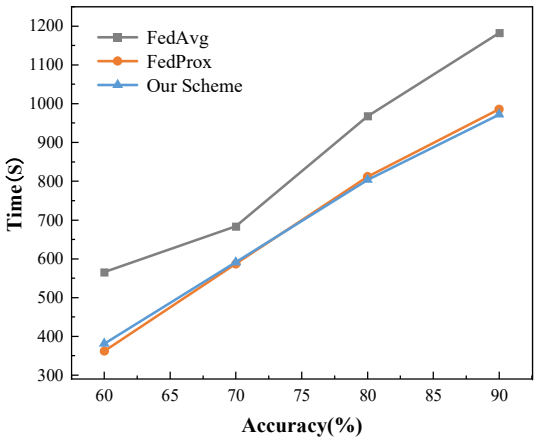

Figure 6. Comparison of convergence time.

6. Conclusions

With the growing problems of data silos and privacy breaches, federated learning techniques enable distributed model training without data going out of the local area, and it is an emerging technology that balances data privacy protection and AI development. Existing studies usually have an optimistic assumption that mobile devices all contribute their computational resources unconditionally, and since model training incurs resource costs, which is unrealistic in real-life scenarios, mobile terminals will be reluctant to participate in federated learning tasks without effective incentives. In addition, real-life isomorphic data likewise seriously affects the accuracy of the federated learning global model. To address the above problems, we propose a reputation mechanism that correlates the model contribution with the completion time, and takes the reputation value as a standard to formulate a compensation incentive mechanism, which combines the model contribution accuracy of the working nodes as well as the local training time in order to comprehensively evaluate the benefits of the terminal nodes, so that only the honest nodes with a high contribution value and a high training efficiency can obtain high compensation. Second, an EMD distance mechanism is introduced, through which the EMD distance, the data heterogeneity level of clients, can be easily measured, and then the clients with too high heterogeneity level can be eliminated in time to reduce the influence of heterogeneous data on the global model of federated learning. Finally, experiments show that this scheme is able to maintain a higher accuracy in Non-IID environments compared with traditional federated learning algorithms.

Use of AI tools declaration

Artificial intelligence (AI) tools were not used during the writing of this article.

Acknowledgments

This work was supported by Inner Mongolia Natural Science Foundation Project (2021LHMS06003) and Inner Mongolia University Basic Research Business Fee Project (114).

Conflict of interest

The authors declare that there are no conflicts of interest.

References

1. I. S. Candanedo, E. H. Nieves, S. R. González, M. T. S. Martín, A. G. Briones, Machine learning predictive model for industry 4.0, in *Knowledge Management in Organizations: 13th International Conference*, Springer International Publishing, Žilina, Slovakia, (2018), 501–510. https://doi.org/10.1007/978-3-319-95204-8_42
2. M. A. Khan, H. El Sayed, S. Malik, M. T. Zia, N. Alkaabi, J. Khan, A journey towards fully autonomous driving-fueled by a smart communication system, *Veh. Commun.*, **36** (2022), 100476. https://doi.org/10.1016/j.vehcom.2022.100476

3. C. J. Haug, J. M. Drazen, Artificial intelligence and machine learning in clinical medicine, *N. Engl. J. Med.*, **388** (2023), 1201–1208. https://doi.org/10.1056/NEJMra2302038
4. A. A. Shaikh, K. S. Lakshmi, K. Tongkachok, J. Alanya-Beltran, E. Ramirez-Asis, J. Perez-Falcon, Empirical analysis in analysing the major factors of machine learning in enhancing the e-business through structural equation modelling (SEM) approach, *Int. J. Syst. Assur. Eng. Manage.*, **13** (2022), 681–689. https://doi.org/10.1007/s13198-021-01590-1
5. J. Konečný, H. B. McMahan, F. X. Yu, P. Richtárik, A. T. Suresh, D. Bacon, Federated learning: strategies for improving communication efficiency, preprint, arXiv:1610.05492. https://doi.org/10.48550/arXiv.1610.05492
6. Q. Li, Y. Diao, Q. Chen, B. He, Federated learning on non-iid data silos: an experimental study, in *2022 IEEE 38th International Conference on Data Engineering (ICDE)*, (2022), 965–978. https://doi.org/10.1109/ICDE53745.2022.00077
7. E. T. M. Beltrán, M. Q. Pérez, P. M. S. Sánchez, S. L. Bernal, G. Bovet, M. G. Pérez, et al., Decentralized federated learning: fundamentals, state of the art, frameworks, trends, and challenges, *IEEE Commun. Surv. Tutorials*, 2023. https://doi.org/10.1109/COMST.2023.3315746
8. Y. Huang, L. Chu, Z. Zhou, L. Wang, J. Liu, J. Pei, et al., Personalized cross-silo federated learning on non-iid data, in *Proceedings of the AAAI Conference on Artificial Intelligence*, **35** (2021), 7865–7873. https://doi.org/10.1609/aaai.v35i9.16960
9. V. Mugunthan, R. Rahman, L. Kagal, Blockflow: An accountable and privacy-preserving solution for federated learning, preprint, arXiv:2007.03856. https://doi.org/10.48550/arXiv.2007.03856
10. Q. Wang, Y. Guo, L. Yu, P. Li, Earthquake prediction based on spatio-temporal data mining: an LSTM network approach, *IEEE Trans. Emerging Top. Comput.*, **8** (2017), 148–158. https://doi.org/10.1109/TETC.2017.2699169
11. J. Li, Y. Shao, K. Wei, M. Ding, C. Ma, L. Shi, et al., Blockchain assisted decentralized federated learning (BLADE-FL): Performance analysis and resource allocation, *IEEE Trans. Parallel Distrib. Syst.*, **33** (2021), 2401–2415. https://doi.org/10.1109/TPDS.2021.3138848
12. X. Wu, Z. Wang, J. Zhao, Y. Zhang, Y. Wu, FedBC: blockchain-based decentralized federated learning, in *IEEE International Conference on Artificial Intelligence and Computer Applications (ICAICA)*, (2020), 217–221. https://doi.org/10.1109/ICAICA50127.2020.9182705
13. W. Zhang, Y. Zhao, F. Li, H. Zhu, A hierarchical federated learning algorithm based on time aggregation in edge computing environment, *Appl. Sci.*, **13** (2023), 5821. https://doi.org/10.3390/app13095821
14. J. Guo, Z. Liu, S. Tian, F. Huang, J. Li, X. Li, et al., TFL-DT: a trust evaluation scheme for federated learning in digital twin for mobile networks, *IEEE J. Sel. Areas Commun.*, **41** (2023), 3548–3560. https://doi.org/10.1109/JSAC.2023.3310094
15. I. Martinez, S. Francis, A. S. Hafid, Record and reward federated learning contributions with blockchain, in *2019 International Conference on Cyber-Enabled Distributed Computing and Knowledge Discovery (CyberC)*, (2019), 50–57. https://doi.org/10.1109/CyberC.2019.00018
16. Y. Liu, Z. Ai, S. Sun, S. Zhang, Z. Liu, H. Yu, Fedcoin: a peer-to-peer payment system for federated learning, in *Federated Learning: Privacy and Incentive*, Cham: Springer International Publishing, (2020), 125–138. https://doi.org/10.1007/978-3-030-63076-8_9
17. J. Kang, Z. Xiong, D. Niyato, Y. Zou, Y. Zhang, M. Guizani, Reliable federated learning for mobile networks, *IEEE Wireless Commun.*, **27** (2020), 72–80. https://doi.org/10.1109/MWC.001.1900119

18. J. Guo, L. Xiong, J. Li, J. Liu, S. Tian, H. Li, An incentive mechanism for horizontal federated learning based on the principle of compound interest, *Phys. Commun.*, **60** (2023), 102128. https://doi.org/10.1016/j.phycom.2023.102128
19. B. McMahan, E. Moore, D. Ramage, S. Hampson, B. A. y Arcas, Communication-efficient learning of deep networks from decentralized data, in *Artificial Intelligence and Statistics*, PMLR, (2017), 1273–1282. 10.48550/arXiv.1602.05629
20. Y. Zhao, M. Li, L. Lai, N. Suda, D. Civin, V. Chandra, Federated learning with non-iid data, preprint, arXiv:1806.00582. https://doi.org/10.48550/arXiv.1806.00582
21. B. Luo, W. Xiao, S. Wang, J. Huang, L. Tassiulas, Tackling system and statistical heterogeneity for federated learning with adaptive client sampling, in *IEEE INFOCOM 2022-IEEE Conference on Computer Communications*, (2022), 1739–1748. https://doi.org/10.1109/INFOCOM48880.2022.9796935
22. H. Wu, P. Wang, Node selection toward faster convergence for federated learning on non-iid data, *IEEE Trans. Network Sci. Eng.*, **9** (2022), 3099–3111. https://doi.org/10.1109/TNSE.2022.3146399
23. A. Chen, Y. Fu, Z. Sha, G. Lu, An emd-based adaptive client selection algorithm for federated learning in heterogeneous data scenarios, *Front. Plant Sci.*, **13** (2022), 908814. https://doi.org/10.3389/fpls.2022.908814
24. Y. Lv, H. Ding, H. Wu, Y. Zhao, L. Zhang, FedRDS: federated learning on non-iid data via regularization and data sharing, *Appl. Sci.*, **13** (2023), 12962. https://doi.org/10.3390/app132312962
25. J. Kang, Z. Xiong, D. Niyato, S. Xie, J. Zhang, Incentive mechanism for reliable federated learning: A joint optimization approach to combining reputation and contract theory, *IEEE Internet of Things J.*, **6** (2019), 10700–10714. https://doi.org/10.1109/JIOT.2019.2940820
26. N. H. Tran, W. Bao, A. Zomaya, M. N. Nguyen, C. S. Hong, Federated learning over wireless networks: optimization model design and analysis, in *IEEE INFOCOM 2019-IEEE Conference on Computer Communications*, (2019), 1387–1395. https://doi.org/10.1109/INFOCOM.2019.8737464
27. T. Li, A. K. Sahu, M. Zaheer, M. Sanjabi, A. Talwalkar, V. Smith, Federated optimization in heterogeneous networks, *Proc. Mach. Learn. Sys.*, **2** (2022), 429–450. https://doi.org/10.48550/arXiv.1812.06127

AIMS Press