Effective Blockchain-Based Asynchronous Federated Learning for Edge-Computing

Zhipeng Gao, Huangqi Li(✉), Yijing Lin, Ze Chai, Yang Yang, and Lanlan Rui

State Key Laboratory of Networking and Switching Technology, Beijing University of Posts and Telecommunications, Beijing 100876, China
lhq320@bupt.edu.cn

Abstract. Since massive data are generated at the network's edge, the Internet of Things devices can exploit edge computing and federated learning to train artificial intelligence (AI) models while protecting data privacy. However, heterogeneous devices lead to low efficiency and single-point-of-failure. Moreover, malicious nodes may affect training accuracy. Therefore, we propose FedLyra, an effective blockchain-based asynchronous federated learning architecture, to improve the efficiency of aggregation and resist malicious nodes in a trusted and decentralized manner. We then propose a reputation mechanism that combines historical behaviors and the quality of local updates to resist disagreements and adversaries. With the help of the reputation mechanism, we propose a council-based decentralized aggregation mechanism to exclude malicious nodes. Experiments show that FedLyra can resist malicious nodes and ensure the accuracy of training results.

Keywords: Federated learning · Blockchain · Edge-computing · Asynchronous architecture · Decentralization

1 Introduction

The vast amount of data generated by edge devices can enhance various AI applications [5]. Edge devices can deliver computation-intensive AI tasks to edge servers without transferring large amounts of data to distant data centers [17,26]. However, data transferred from edge devices to edge servers involves privacy risks [26]. Federated learning is dedicated to solving the privacy problem in distributed learning. An edge computing-based federated learning system can learn a global statistical model with localized data on edge devices [13].

Every coin has two sides. First, federated learning suffers the single-point-of-failure due to the need for a central server. Second, there may be data inconsistency in the trustless environment. Third, heterogeneous devices make aggregation processes inefficient. Forth, attackers may disrupt the model aggregation process by hijacking devices or injecting malicious parameters.

The decentralization and asynchronous of federated learning can solve the above security problems. As a price, compared with the centralized counterpart,

H. Gao et al. (Eds.): CollaborateCom 2022, LNICST 460, pp. 514–532, 2022.
https://doi.org/10.1007/978-3-031-24383-7_28

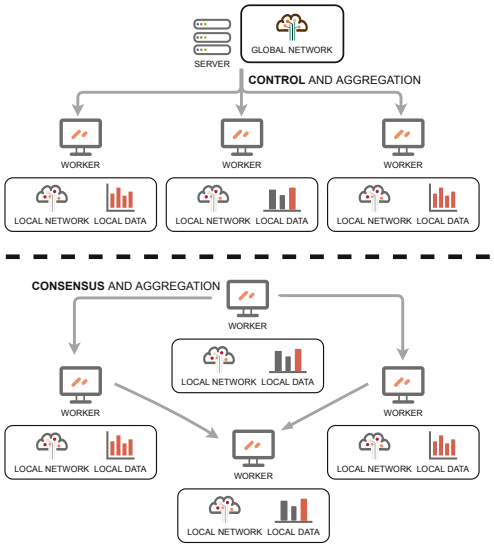


Fig. 1. Centralized vs. decentralized federated learning

decentralized asynchronous federated learning requires additional consideration. As shown in Fig. 1, centralized federated learning requires scheduling and aggregation through a central server. In contrast, in a decentralized network, all nodes are equal, and there is no authority responsible for control and storage. Therefore, the decentralized and asynchronous architecture needs to coordinate the nodes and ensure the consistency of global information such as global model, history, etc.

In recent years, blockchain has gradually become a promising solution for trusted decentralized systems with the characteristics of decentralization, tamper-proof, and transparency [1]. Many scholars have attempted to solve the aforementioned problems of federated learning by using blockchain to build consensus among untrusted workers, thus ensuring the consistency and reliability of the global model. However, there are several points often ignored. First, it is inefficient when reaching a consensus in a large-scale network [22]. Since each aggregation operation requires consensus in blockchain, adopting an inefficient consensus algorithm like Proof of Work (PoW) would severely reduce the aggregation efficiency. Second, limited resources for edge devices restrict the throughput of the blockchain. Furthermore, there may be huge delays caused by stragglers in the learning process [20]. Third, global synchronization can degrade the system's efficiency due to the unstable communication links [22]. Fourth, it is hard to find and exclude malicious nodes because traditional consensus cannot ensure the correctness of uploaded models, meaning that a poisoned model may be recorded in the blockchain since consensus will not verify the model itself.

In this paper, we propose FedLyra, an effective blockchain-based asynchronous federated learning framework to adapt to heterogeneous edge networks.

FedLyra restricts the consensus among a small group of particular nodes to reduce the consensus overhead and improve throughput. To reduce the latency caused by synchronous communication and stragglers, FedLyra adopts asynchronous aggregation, which enables FedLyra to process local updates uploaded by edge devices in a real-time manner. The main contributions of this paper are summarized as follows.

1. We propose FedLyra, a novel blockchain-based decentralized asynchronous federated learning architecture that improves the security and efficiency of federated learning under unstable edge networks.
2. In order to identify and prevent malicious nodes from interfering with the federated learning, we quantify the reputation of a node based on historical data and the quality of local updates, and exclude the malicious node through the reputation threshold.
3. To improve the aggregation efficiency of the federated learning, we design a council consensus mechanism based on reputation to increase the consensus speed by reducing the consensus scale.

This paper is organized as follows. Section 2 discusses the research related to blockchain and federated learning. Section 3 presents the core framework and mechanism proposed in this paper. Section 4 evaluates the performance of the system. Section 5 concludes the paper.

2 Related Work

Federated learning has attracted much attention from researchers since McMahan *et al.* proposed FedAvg, a federated learning method that learns a shared model between different devices through a synchronized weighted average method [18]. Li *et al.* [14] proposed FedProx, which limits the effect of non-i.i.d. data on the global model by introducing a proximal term. Reisizadeh and others proposed FedPAQ algorithm [19]. FedPAQ uses cycle averaging instead of global synchronous averaging. Xie *et al.* [24] proposed an asynchronous federated learning architecture FedAsync in which Worker and Updater work in parallel. Chen *et al.* [3] further improved on [24] by proposing FedSA, which accelerates training and improves communication efficiency by a two-stage training strategy and dynamical hyperparameters.

Since federated learning is privacy-preserving and can work between heterogeneous devices, some studies have attempted to combine federated learning with edge computing to enable distributed learning on edge heterogeneous networks. Wang *et al.* [21] combines deep reinforcement learning with federated learning to intelligently optimize communication and computing resources in the mobile edge computing. Jin *et al.* [7] explored approaches to adaptive adaptation of federated learning with limited resource budgets. Chen *et al.* [4] proposed an asynchronous federated learning method that can process continuous stream data on edge devices. Khan *et al.* [9] propose a Stackelberg-game-based incentive mechanism for federated learning. The above researches focus on optimizing

the performance of federated learning or resource allocation in edge computing. They do not consider how to solve the trust between edge nodes.

As a decentralized immutable distributed digital ledger system [25], blockchain is suitable for processing distributed transactions in an untrusted environment. Kim *et al.* [10] proposed a blockchain-based distributed federated learning architecture BlockFL, in which each device will connect to a miner, who will verify the local updates uploaded by the device, and the miner who successfully mines a new valid block will aggregate the local updates and add the results to the blockchain. Li *et al.* [15] proposed a consensus approach based on a trusted small committee that can effectively avoid the influence of malicious nodes. Kang *et al.* [8] used reputation as a measure of device's reliability and trustworthiness and implemented a reputation-based reliable worker selection scheme, in addition to which they proposed an effective incentive mechanism that combines reputation with contract theory.

However, the above blockchain-based approaches implement in the synchronous scheme, which is unsuitable for edge environments with limited network bandwidth and low equipment reliability. Liu *et al.* [16] apply blockchain-based asynchronous federated learning methods to edge computing, but they lack consideration of practical conditions such as limited edge network resources and unstable devices. Feng *et al.* [6] proposed a blockchain-based asynchronous federated learning scheme, which is an improvement of [10] with an entropy-based node evaluation method for determining the weight of local update. These blockchain-based asynchronous federated learning methods attempt to improve efficiency by asynchronization. But, these methods adopt all devices to participate in consensus, which requires many resources for message synchronization. It may cause low throughput and significant delays due to the limited resource of edge devices and stragglers.

3 Core Technology of FedLyra

FedLyra employs a decentralized asynchronous federated learning mechanism based on council and reputation, which consists of asynchronous federated learning process, reputation quantification, and council consensus. This section first demonstrates the architecture of FedLyra, introduce the workflow, and then designs the three core mechanisms of FedLyra.

3.1 Architecture

In FedLyra, all devices in the edge network are treated as peer nodes and jointly maintain the blockchain ledger, as shown in Fig. 2. Nodes entitled to participate in learning are divided into common nodes and council nodes. Common nodes undertake local training and update uploading. Council nodes are a special type of common node that is additionally responsible for receiving updates from other nodes and performing aggregation. They will package aggregation results, upload updates, verification scores and reputation into blocks, then upload the blocks to

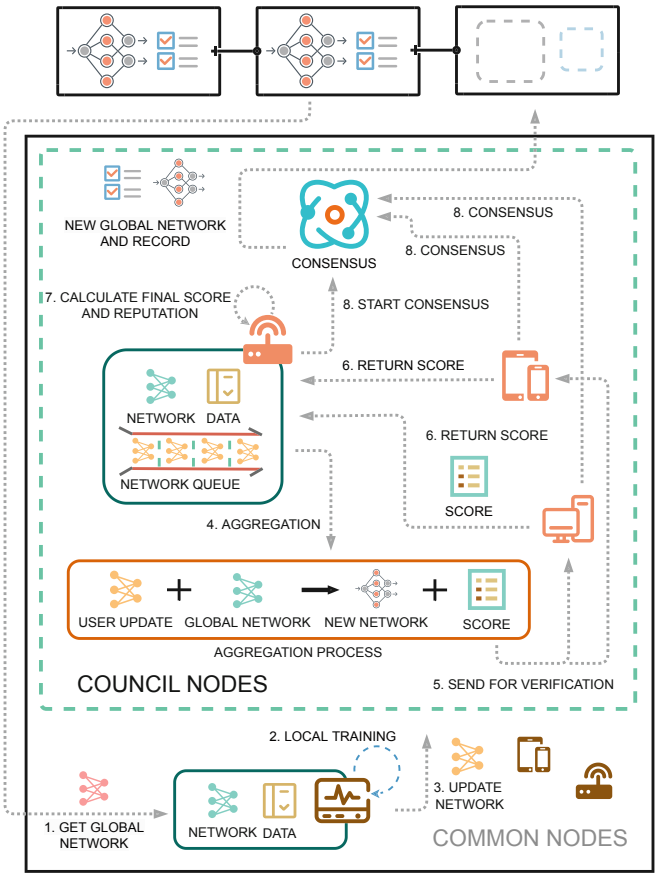


Fig. 2. Structure of FedLyra

blockchain after internal consensus among the council. It is worth noting that the council nodes are also common nodes, meaning that they still need to conduct model training and uploading. Each aggregation of FedLyra must be determined by consensus. By dividing the nodes, FedLyra limits the consensus process to a small council to reduce latency. Furthermore, since nodes outside the council do not have to perform aggregation, most nodes can concentrate on local training.

From a system-level perspective, FedLyra is divided into off-chain learning and on-chain verification. In the off-chain learning part, common nodes train the model based on local data, and council nodes aggregate the updates uploaded by common nodes. In the on-chain verification part, council nodes validate the global models and quantify model scores and node reputation. Blockchain stores the global model generated by each aggregation, local updates uploaded by nodes and other related records. Based on these records, we can quantitatively evaluate nodes' behaviors, assign roles to nodes and adjust the weight of node updates in

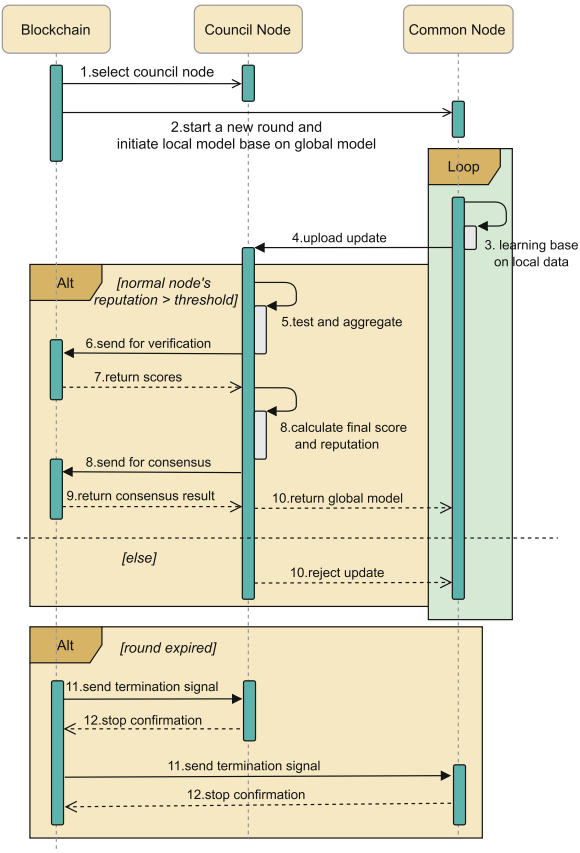


Fig. 3. Workflow of FedLyra in a round

aggregation. We can also penalize nodes with poor reputations, excluding them from the learning process.

3.2 Workflow

Figure 3 shows the workflow of FedLyra in a training round. Each common node first takes the latest global model from the local blockchain ledger and trains it according to the local dataset. After the training is completed, the common node broadcasts the local update to the council nodes and waits for them to complete the model aggregation. Once a council node receives the local update, it will test and aggregate the model, then broadcast the new model to other council nodes for scoring and validation. After the verification, this council node generates the final score of the common node's update based on the model scores quantified by others in council, then calculates the reputation of local update uploader based on the final score and the historical reputation. The new reputation and

global model will be updated to blockchain through a successful consensus. The common node then continues local training and model uploading until the smart contract signal the end of the current round. Once a round ends, all nodes stop training, the learning process proceeds to the next round, and new council nodes will be determined. The learning process continues until a specified number of rounds are reached or the expected accuracy is met.

3.3 Asynchronous Training and Aggregation

The traditional federated learning method implements cross-device model training based on multi-node and a single central server, which fully leverages local data from different nodes while preventing data leakage. Denote the devices in the edge network by $D = \{1, 2, 3, \ldots, N_D\}$. $|D| = N_D$, which is the number of edge devices. The overall goal of federated learning is typically minimizing the objective function $\underset{w \in \mathbb{R}^{N_D}}{min} f(w)$ where $f(w) := \frac{1}{n} \sum_{n}^{i=1} f_i(w)$. Here, $f_i(w)$ is the local objective function of device i.

In asynchronous federated learning, the server and worker perform asynchronous aggregation and local training, respectively. A worker continuously performs non-blocking local training and sends updates to the server once local training is completed. The server performs aggregation immediately after receiving the updates from the worker and sends the aggregated results to the corresponding worker. By combining asynchronous aggregation and non-blocking communication, asynchronous federated learning mitigates the influence of stragglers, prevents other devices from being blocked due to waiting for slow training devices, thus improving device utilization.

Inspired by the Delegated Proof of Stake (DPoS) consensus algorithm, we set up a council which cooperates with smart contracts to implement the scheduling and aggregation process in federated learning. The control function of the smart contract is shown in Algorithm 1. K is the number of council members. Upon the start of each round, the control function initializes the round termination flag in the contract, selects the council node by random or reputation sorting, and sends the node identity and the start signal to the corresponding node. It subsequently waits until $\frac{2}{3}$ of common nodes have completed the learning process and then reverses the round termination flag in the contract to alert all active nodes that the current round should be terminated.

The running process of common nodes is shown in Algorithm 2. In FedLyra, common nodes are only responsible for asynchronous training and local update upload meanwhile monitoring the round termination flag in the smart contract. Nodes can use the Adam algorithm for local training [11]. The global model used for node training can be obtained directly from the local blockchain records. When conducting an update upload, the node sends w_i, the local update, τ_i, the timestamp of the start of training, and d_i, the size of local dataset, to each council node.

Different from the common node, the council node needs to perform the duty of the common node (training and uploading) while aggregating model updates

Algorithm 1. Council selection and training control contract

Require: $round, K$

for $i = 0, 1, 2, ..., round - 1$ **do**
  Change termination flag in contract to False
  Randomly choose common node N_{common}
  if is the first time this function invoked and $i = 0$ **then**
    Randomly select K Nodes from N_{common} as $N_{council}$ with Verifiable Random Function
  else
    Sort N_{common} by reputation
    Select $N_{council}$ from top K in N_{common}
  end if
  Send identity and start signal to all node in N_{common}
  Wait until $^2/_3$ nodes in N_{common} have completed the learning process
  Change termination flag in contract to True
end for

Algorithm 2. Worker updating in FedLyra

Require: $epoch, \gamma$

Get latest w_{global} from blockchain with its timestamp t_{global}
$\tau_i \leftarrow t_{global}$
$w_i \leftarrow w_{global}$
$d_i \leftarrow$ size of dataset
while Termination flag in contract is False **do**
  for $i = 0, 1, 2, ..., epoch - 1$ **do**
    Update model using Adam algorithm with γ
  end for
  Broadcast (w_i, τ_i, d_i) to council
  Wait for aggregation
end while

from others, as well as evaluating and recording the updates. The aggregation process of council nodes is described in Algorithm 3. The receptor thread of the council node will store the data in the message queue if it receives the uploaded information from common nodes. Meanwhile, it will validate the aggregation result if it receives an aggregation massage from a council node, which will be described in the following section. The aggregator thread will take the model updates from the message queue in first-in-first-out (FIFO) order. If the uploader's reputation exceeds the threshold, the aggregator will test the model

Algorithm 3. Council aggregation in FedLyra

Require: t_{max}, ζ, a, b
Process: Council
Initial *queue*
Run Receptor thread and Aggregator thread in parallel
Thread: Receptor
while Termination flag in contract is False **do**
  if Receive (w_i, τ_i, d_i) from common node **then**
    Push (w_i, τ_i, d_i) as msg_i into *queue*
  else if Receive $(w_i, s_i^j, w'_{global})$ from other council node **then**
    $s_i^k \leftarrow$ validates score of w_i by current node k
    $s_{g'}^k \leftarrow$ validates score of w'_{global} by current node k
    $s_g^k \leftarrow$ score of current global model w_{global}
    if $\left|s_i^j - s_i^k\right| < \varepsilon$ and $s_{g'}^k \geq s_g^k - \delta$ **then**
      Sign and return s_k^j to node j
    end if
  end if
end while
Thread: Aggregator
while Termination flag in contract is False **do**
  Get msg_i that has not been recorded in blockchain from *queue*
  Get (w_i, τ_i, d_i) from msg_i
  Get reputation r_i for node i from blockchain
  if $r_i \geq$ reputation threshold **then**
    Test w_i with local data and get correspond score s_i^j
    $w'_{global} \leftarrow$ Aggregate w_i according to equation (1)
    Send $(w_i, s_i^j, w'_{global})$ to other council node for verification
    Receive test score S_i from others and calculate r_i according to equation (5)
    Pack $(w'_{global}, msg_i, S_i, r_i)$ to start consensus
  end if
end while

using the local data, and record its test score. The score can be the average absolute error (MAE) or other indicators that can measure the quality of the model. Model aggregation is performed after the test according to the following equation:

$$w_{global} \leftarrow \alpha \times w_i + (1 - \alpha) \times w_{global} \tag{1}$$

where α is the coefficient of aggregation, which is calculated according to the following formula:

$$\alpha \leftarrow \alpha_0 \times l\,(t - \tau) \times n(\frac{d_i}{d_{all}}) \times r_i \tag{2}$$

where $l(t-\tau)$ is a function to measure the degree of staleness of the update. t is current time. τ is the generation time of the global model used by node i. $l(t-\tau)$ is represented by $l(t-\tau) = \begin{cases} \frac{1}{a(t-\tau-b)+1}, & t-\tau < b \\ 1, & else \end{cases}$. r_i is the reputation value of the node i, which will be described in detail below. $n(\frac{d_i}{d_{all}})$ is a factor related to the amount of local data on node i. $n(\frac{d_i}{d_{all}})$ is represented by

$$n(\frac{d_i}{d_{all}}) = \beta \times arctan\left(\gamma \times \frac{d_i}{d_{all}}\right) \tag{3}$$

where d_i is the dataset size of node i, d_{all} is the dataset size of all nodes, and β and γ are scale factors.

After the aggregation of updates from node i, council node j packages the generated global model parameters w_g^j, node's local update w_i, test scores s_i^j, and original message of node i into a transaction and broadcasts it to others in council.

Other council nodes will extract local update and global model in the transaction and test them by local data respectively. Suppose that council node k receives a transaction from council node j and validates the aggregation result of common node i. Node k will check if the gap between the test score s_i^k for w_i and the score s_i^j in the transaction is within the specified range, i.e., $\left|s_i^j - s_i^k\right| < \varepsilon$, and if the test score of w_g^j is greater than or similar to the test score of the latest global model in the blockchain. If both conditions hold, the transaction passes the verification. Node k will return s_k^j to node j that proposed the transaction. Node j calculates the final test score and the reputation of node i based on the test scores of other council nodes. The detailed calculation for score and reputation is described in the next subsection.

3.4 Reputation of Node

All nodes have an initial reputation value at their first participation in the task. Suppose that in the subsequent learning process, when node k uploads a local update, the council node i processes this update. To prevent malicious nodes in council from deliberately providing wrong scores, node i sorts $S_k = \{s_k^j | j \in N_{council}\}$ in descending order, and exclude the top 1/6 and bottom 1/6 scores, then calculates the final test score s_k of this update for node k as follow

$$s_k = \frac{1}{|S_k|} \times \sum_{s' \in S_k} s' \tag{4}$$

The calculation of reputation is based on the final test score. Considering that a malicious node may increase its reputation, reputation must be iterative based on the historical reputation and current score

$$r_k = \zeta \times r_k + (1-\zeta) \times (s_k / s_{compare})^2 \tag{5}$$

where ζ is the coefficient that balances the historical data and the current data. $s_k/s_{compare}$ is the normalized score. In the initial stage, the scores of local updates are low due to insufficient training. If the score is directly used as the base of the squared term, it will lead to a low reputation. In the later stage, the scores are high and need to be normalized to reflect the relative training quality. $s_{compare}$ is calculated as follows

$$s_{compare} = \begin{cases} s_{mid}, & |S| > 1/3 \cdot |N_{common}| \\ s_{compare}\ in\ last\ round, & else \end{cases} \tag{6}$$

where $|N_{common}|$ is the number of common nodes. s_{mid} is the median score of S, $S = \{(j, s_j)|node\ j\ participated\ in\ learning\}$, which is a set of final score for all participating learning nodes in current round. S is maintained in the blockchain and updated through transaction as follow.

After the calculation of the reputation of node k, council node will update (k, s_k), the score of node k into S, generating a new score set S'. Then, it will package S', S_k, s_k together with other information mentioned above into a transaction and initiate a consensus in council. If consensus is successfully reached, the aggregation result will be permanently written to the blockchain.

3.5 Council Selection and Consensus

In FedLyra, only council members will participate in the aggregation and generation of the global model, while common nodes have no right to handle the local updates uploaded by other nodes. Meanwhile, common nodes are also excluded from the blockchain consensus, and only the council nodes will participate in the consensus. Since the council has the privilege to aggregate updates and write data to the blockchain by consensus, the council nodes must be selected cautiously. In principle, we want to grant these powers to good nodes, while malicious nodes are excluded from the council as much as possible. In FedLyra, the selection of council is mostly based on reputation, which is described in Lines 4 to 9 of Algorithm 1.

In the first round of the first learning task, the selection of council nodes is performed by random sampling due to the lack of historical data. With techniques such as Verifiable Random Function (VRF), FedLyra can complete council selection in a decentralized scenario. In the subsequent learning process, the selection of council nodes is performed based on node reputation generated in the last round. FedLyra sorts the reputation of nodes through smart contracts. At the beginning of the next round, nodes with the top K reputations are selected as the council nodes. It is worth noting that the reputations of nodes in the previous learning task can be applied to the next task. The council nodes in the first round of the next task can be selected by reputation ranking instead of random sampling.

After finishing model aggregation and reputation quantification and passing the council's validation, the council node that handles the uploaded update will pack the local update, update's score, the validated new global model, and node reputation into blocks and initiate consensus. Consensus is the final step

of model aggregation, only global models that reach consensus can be released to others in the network for local training, so the efficiency of consensus has a significant impact on the system's throughput. Most blockchain-based federated learning architectures adopt the PoW consensus algorithm for consensus. Still, PoW suffers from low throughput rates, and consumes a lot of energy and computational resources, which is not suitable for energy-scarce edge devices. To further increase the throughput rate and reduce the energy consumption of edge devices, the consensus among council nodes adopts the Practical Byzantine Fault Tolerance (PBFT) algorithm [2]. PBFT has a high throughput in small-scale networks that can handle 100 transactions per second [22], more than ten times the PoW. PBFT can tolerate malicious nodes that account for less than 1/3 of all nodes. Furthermore, PBFT does not require mining, which effectively reduces the energy consumption caused by consensus. In PBFT, each node takes turns to be the primary, and the primary will receive the request and launch a broadcast process to send the request to other nodes in the network. In FedLyra, the master node also needs to filter duplicate requests. A duplicate request is defined as an aggregation of the same local updates.

4 Evaluation Results

In this section, we evaluate the feasibility of FedLyra. We first analyze the training performance of FedLyra with different sizes of networks and different proportions of malicious nodes. Then, we analyze the impact of different reputation thresholds on the speed of detection and the risk of misclassification.

4.1 Experiment Setting

We conduct experiments on two benchmarks: Fashion-MNIST [23] and Cifar-10 [12]. We transform both into non-i.i.d. datasets, and each node will train based on its local dataset. We use Fashion-MNIST to construct evenly assigned dataset, meaning each node has the same number of local images. Fashion-MNIST contains a training set of 60,000 samples and a test set of 10,000 samples. We sort the training set by label, divide it into 200 slices, and distribute them equally to each node. In addition, we use Cifar-10 to construct unevenly assigned datasets. In experiments based on Cifar-10, we simulate 50 nodes and distribute the training set containing 50,000 images to these nodes unevenly. Figure 4 shows the size of the local dataset owned by each node. Nodes' training performance may have huge differences in experiments based on unevenly assigned datasets.

In the experiments, we adopt convolutional neural network (CNN) as the local training model[1,2]. The time limit for each round is 10 s. β and γ, the

[1] The CNN network trained on Cifar has 6 layer with the following structure: $3 \times 3 \times 64/128/256/512$ Convolutional $\rightarrow$ 2×2 MaxPool $\rightarrow$ 2048 Fully connected $\rightarrow$ SoftMax.

[2] The CNN network trained on Fashion-MNIST has 5 layer with the following structure: $3 \times 3 \times 16/32/64$ Convolutional $\rightarrow$ 2×2 MaxPool $\rightarrow$ 576 Fully connected $\rightarrow$ SoftMax.

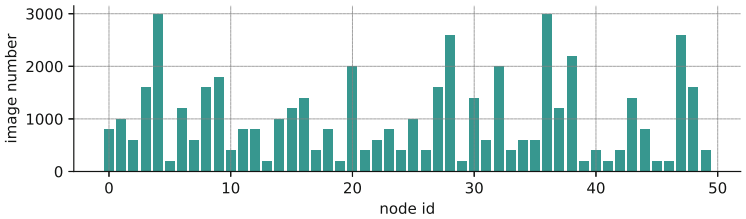


Fig. 4. Number of images owned by 50 nodes in experiment based on Cifar-10

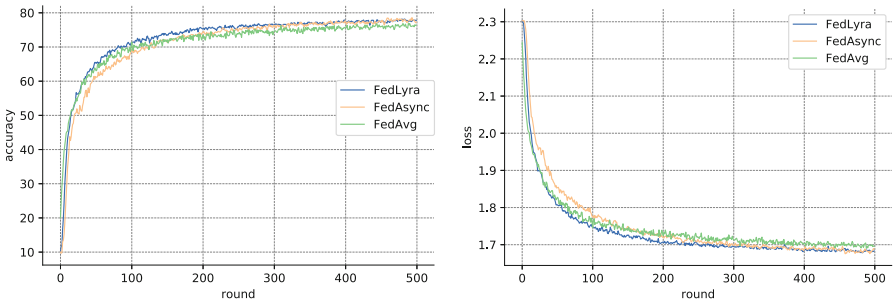
weight of the dataset size in $n(\frac{d_i}{d_{all}})$, are fixed to 2 and 10, respectively. For the weight of historical reputation, $\zeta = 0.3$. Reputation threshold is 0.3 The initial weight of the aggregation α_0 is set to 0.9. The node learns 5 epochs locally before uploading the local update. The learning rate is set as 0.001.

In the experiment, we tested FedLyra's ability to resist malicious attacks by simulating malicious nodes. Malicious nodes will intentionally upload a neural network composed of random parameters. The parameters of the network are uniformly distributed random integers ranging from 0 to 10. These useless models may be involved in the aggregation, thereby disrupting the global model of the system.

4.2 Learning Performance of FedLyra

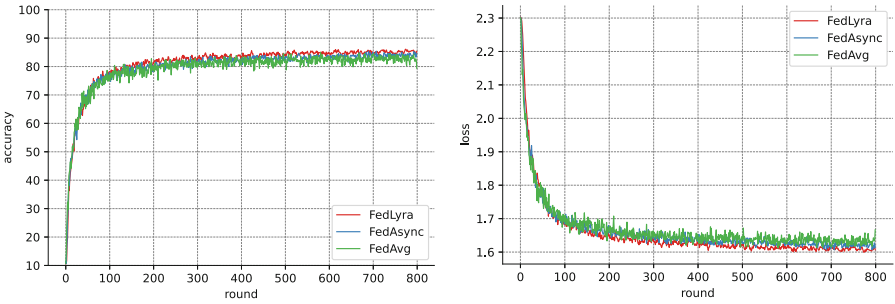
We first test the performance of FedLyra in scenarios where no malicious parties are involved. The FedAvg algorithm proposed by McMahan et al. [18] and the FedAsync algorithm proposed by Xie et al. [24] are adopted as the benchmark for synchronous federated learning and asynchronous federated learning, respectively. Figure 5 shows the performance of the three algorithms for the federated learning task base on unevenly assigned Cifar-10 in the absence of malicious nodes. The number of nodes is 50. As shown in Fig. 5, although the precision and loss of the three are not much different after the model approaches convergence, compared to the other two algorithms, FedLyra has a faster learning speed and the best learning effect. This is because FedLyra optimizes the weights of node updates in the aggregation process based on the size of local data. The weights of updates from nodes with more local data will be increased, and the quality of these updates tends to be better than updates from nodes with only a small amount of data. Figure 6 shows the accuracy and loss of the three algorithms for federated learning tasks on FashionMNIST with 100 goods nodes. We can conclude that FedLyra has a performance that is not inferior to the benchmark algorithm in a scenario with no malicious node interference.

We conduct experiments with malicious node participation based on an evenly assigned dataset, thereby better examining our proposed reputation mechanism by eliminating the influence of differences in the size of local data. Figure 7 shows the performance of each algorithm on FashionMNIST for federated learning tasks when 10% of the nodes are malicious nodes. The number of total nodes is 100. It is observed that FedAsync and FedAvg can hardly train an available model even



(a) Accuracy with Cifar (b) Loss with Cifar

Fig. 5. Learning performance based on unevenly assigned dataset without malicious nodes

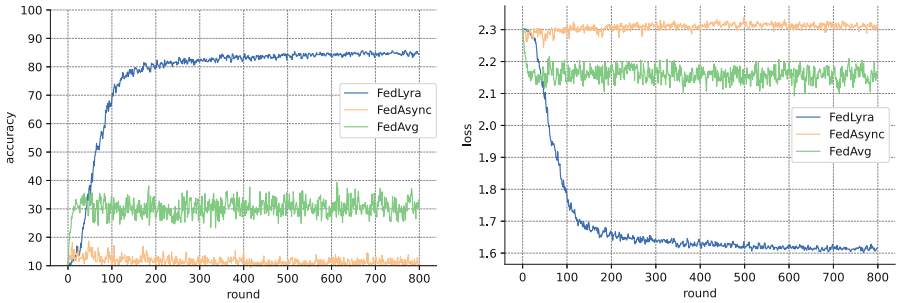


(a) Accuracy with FashionMNIST (b) Loss with FashionMNIST

Fig. 6. Learning performance based on evenly assigned dataset without malicious nodes

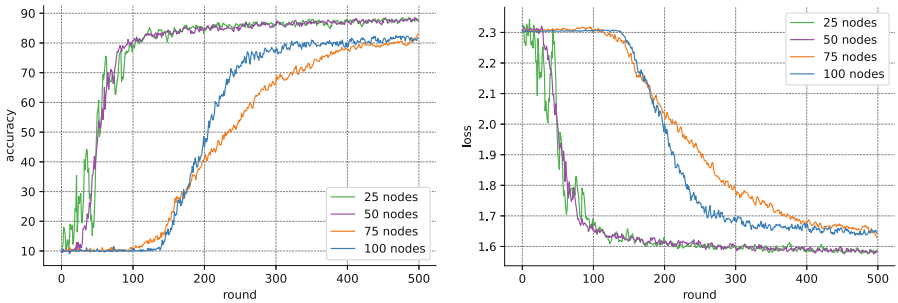
if 90% of the nodes are good nodes. Compared to the two benchmark algorithms, FedLyra performs much better. After screening and excluding malicious nodes, FedLyra can perform federated learning based on the remaining common nodes, reaching accuracy rates of nearly 85% on FashionMNIST. In contrast, the accuracy rate of FedAsync and FedAvg cannot even reach 40%.

To verify the reliability of FedLyra in networks of different scales, we test FedLyra in networks with 25, 50, 75, and 100 nodes based on FashionMNIST. In these networks, 30% of the nodes are malicious. We distribute the dataset equally to each node. It can be seen from Fig. 8 that FedLyra can accomplish the learning task in networks of different sizes. As the number of nodes decreases, the learning performance improves, and the time affected by malicious nodes is relatively reduced. This phenomenon is due to 1) Compared with large-scale networks, small networks have fewer malicious nodes (a 100-node network contains 30 malicious nodes, compared to 25 nodes with only 7 malicious nodes) 2) In a small-scale network, the local data of a single node is larger, thus the quality of the trained model of the common node is higher, making the inefficient model uploaded by the malicious node more noticeable (in a 25-node network, each



(a) Accuracy with FashionMNIST (b) Loss with FashionMNIST

Fig. 7. Learning performance with 10% malicious nodes

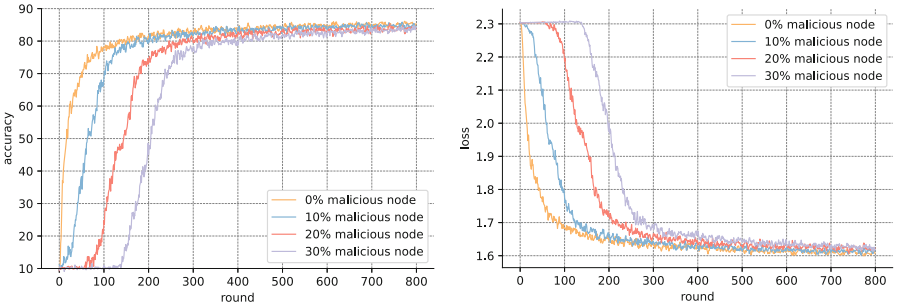


(a) Accuracy with FashionMNIST (b) Loss with FashionMNIST

Fig. 8. Learning performance under different size networks

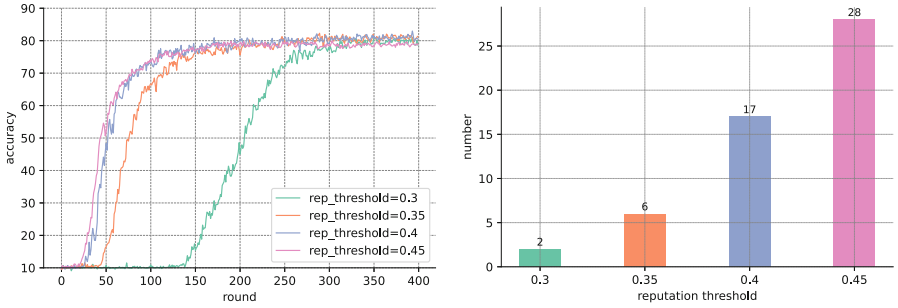
node has 1200 Pictures, and in a network of 100 nodes, each node has only 400 pictures).

In order to further analyze the influence of malicious nodes, we continue to conduct experiments on FashionMNIST to analyze the influence of different proportions (0% to 30%) of malicious nodes on model training. From Fig. 9, it can be seen that as the proportion of malicious nodes continues to rise, the number of rounds FedLyra needs to achieve the same accuracy continues to rise, meaning that more malicious nodes will lead to more time spent on eliminating them, which is also consistent with intuition. Meanwhile, after the model converges, the effect of learning with a smaller proportion of malicious nodes will be slightly better than the environment with more malicious nodes. However, the difference is not evident on the FashionMNIST dataset due to the relatively low difficulty of the FashionMNIST dataset. This influence will be more pronounced on more difficult datasets such as cifar100.



(a) Accuracy with FashionMNIST (b) Loss with FashionMNIST

Fig. 9. Influence of different proportions of malicious nodes



(a) Influence of reputation threshold on learning speed (b) The number of mistaken node under different reputation thresholds

Fig. 10. Influence of different proportions of malicious nodes

4.3 Impact of Reputation Threshold

In FedLyra, the strength of detection of malicious nodes can be adjusted by reputation threshold, i.e., a higher reputation threshold means that there is a greater possibility of judging a suspicious node with low reputation as a malicious node. We conducted experiments based on FashionMNIST to explore the impact of different reputation thresholds on the ability to protect against malicious nodes. The proportion of malicious nodes is fixed at 30% in the experiment. As shown in Fig. 10(a), the higher the reputation threshold, the faster FedLyra detects malicious nodes. For example, achieving 50% classification accuracy takes more than two hundred rounds with a threshold of 0.3, but less than fifty rounds when the threshold is 0.45. However, this does not mean that the higher the reputation threshold, the better the learning effect, because an excessive reputation threshold may lead to mistaking a large number of common nodes. Figure 10(b) shows the number of good nodes that are mistaken for malicious nodes under different reputation thresholds. It can be seen from the figure that the higher the reputation threshold, the greater the number of nodes that are misjudged, i.e.,

the probability of misjudgment continues to increase. Mistaking common nodes means inevitably losing a portion of the dataset, which in turn reduces model accuracy.

5 Conclusion

To improves the security and efficiency of federated learning under unstable edge networks, we propose FedLyra, a decentralized blockchain-based asynchronous federated learning architecture. FedLyra achieves trustworthiness and reliability through blockchain. To increase the system throughput and avoid the latency caused by synchronous communication, we adopt an asynchronous mechanism and the council-based consensus. We also propose a quantification mechanism for node reputation. The reputation determines the weight of the uploaded updates and the permissions of a node. We evaluate the performance of the proposed algorithm based on two types of datasets, and the statistical results show that FedLyra can resist malicious nodes and performs well in terms of learning accuracy. For future work, we will focus on exploring a more flexible and state-aware reputation quantification mechanism that reduces the probability of a common node being treated as a malicious node. In addition to this, we will incorporate device capabilities as a reference factor in council selection to further improve aggregation efficiency.

Acknowledgment. This work is supported by National Natural Science Foundation of China (62072049).

References

1. Ali, M.S., Vecchio, M., Pincheira, M., Dolui, K., Antonelli, F., Rehmani, M.H.: Applications of blockchains in the Internet of Things: a comprehensive survey. IEEE Commun. Surv. Tutor. **21**(2), 1676–1717 (2019)
2. Castro, M., Liskov, B.: Practical byzantine fault tolerance. In: Seltzer, M.I., Leach, P.J. (eds.) Proceedings of the Third USENIX Symposium on Operating Systems Design and Implementation (OSDI), New Orleans, Louisiana, USA, 22–25 February 1999, pp. 173–186. USENIX Association (1999)
3. Chen, M., Mao, B., Ma, T.: FedSA: a staleness-aware asynchronous Federated Learning algorithm with non-IID data. Future Gener. Comput. Syst. **120**, 1–12 (2021)
4. Chen, Y., Ning, Y., Slawski, M., Rangwala, H.: Asynchronous online federated learning for edge devices with non-IID data. In: 2020 IEEE International Conference on Big Data (Big Data), pp. 15–24 (2020)
5. Dong, S., Wang, P., Abbas, K.: A survey on deep learning and its applications. Comput. Sci. Rev. **40**, 100379 (2021)
6. Feng, L., Zhao, Y., Guo, S., Qiu, X., Li, W., Yu, P.: Blockchain-based asynchronous federated learning for Internet of Things. IEEE Trans. Comput. 1 (2021)

7. Jin, H., Yan, N., Mortazavi, M.: Simulating aggregation algorithms for empirical verification of resilient and adaptive federated learning. In: 2020 IEEE/ACM International Conference on Big Data Computing, Applications and Technologies (BDCAT), pp. 124–133 (2020)
8. Kang, J., Xiong, Z., Niyato, D., Xie, S., Zhang, J.: Incentive mechanism for reliable federated learning: a joint optimization approach to combining reputation and contract theory. IEEE Internet Things J. **6**(6), 10700–10714 (2019)
9. Khan, L.U., et al.: Federated learning for edge networks: resource optimization and incentive mechanism. IEEE Commun. Mag. **58**(10), 88–93 (2020)
10. Kim, H., Park, J., Bennis, M., Kim, S.L.: Blockchained on-device federated learning. IEEE Commun. Lett. **24**(6), 1279–1283 (2020)
11. Kingma, D.P., Ba, J.: Adam: a method for stochastic optimization. In: Bengio, Y., LeCun, Y. (eds.) 3rd International Conference on Learning Representations, ICLR 2015, San Diego, CA, USA, 7–9 May 2015, Conference Track Proceedings (2015). http://arxiv.org/abs/1412.6980
12. Krizhevsky, A., Hinton, G., et al.: Learning multiple layers of features from tiny images (2009)
13. Li, T., Sahu, A.K., Talwalkar, A., Smith, V.: Federated learning: challenges, methods, and future directions. IEEE Sig. Process. Mag. **37**(3), 50–60 (2020)
14. Li, T., Sahu, A.K., Zaheer, M., Sanjabi, M., Talwalkar, A., Smith, V.: Federated optimization in heterogeneous networks. In: Proceedings of Machine Learning and Systems 2020, MLSys 2020, Austin, TX, USA, 2–4 March 2020. mlsys.org (2020)
15. Li, Y., Chen, C., Liu, N., Huang, H., Zheng, Z., Yan, Q.: A blockchain-based decentralized federated learning framework with committee consensus. IEEE Netw. **35**(1), 234–241 (2021)
16. Liu, Y., Qu, Y., Xu, C., Hao, Z., Gu, B.: Blockchain-enabled asynchronous federated learning in edge computing. Sensors **21**(10), 3335 (2021)
17. Mao, Y., You, C., Zhang, J., Huang, K., Letaief, K.B.: A survey on mobile edge computing: the communication perspective. IEEE Commun. Surv. Tutor. **19**(4), 2322–2358 (2017)
18. McMahan, B., Moore, E., Ramage, D., Hampson, S., Aguera y Arcas, B.: Communication-efficient learning of deep networks from decentralized data. In: Proceedings of the 20th International Conference on Artificial Intelligence and Statistics. Proceedings of Machine Learning Research, vol. 54, pp. 1273–1282. PMLR, 20–22 April 2017
19. Reisizadeh, A., Mokhtari, A., Hassani, H., Jadbabaie, A., Pedarsani, R.: FedPAQ: a communication-efficient federated learning method with periodic averaging and quantization. In: Proceedings of the Twenty Third International Conference on Artificial Intelligence and Statistics. Proceedings of Machine Learning Research, vol. 108, pp. 2021–2031. PMLR, 26–28 August 2020
20. Tak, A., Cherkaoui, S.: Federated edge learning: design issues and challenges. IEEE Netw. **35**(2), 252–258 (2021)
21. Wang, X., Han, Y., Wang, C., Zhao, Q., Chen, X., Chen, M.: In-Edge AI: intelligentizing mobile edge computing, caching and communication by federated learning. IEEE Netw. **33**(5), 156–165 (2019)
22. Wu, M., Wang, K., Cai, X., Guo, S., Guo, M., Rong, C.: A comprehensive survey of blockchain: from theory to IoT applications and beyond. IEEE Internet Things J. **6**(5), 8114–8154 (2019)
23. Xiao, H., Rasul, K., Vollgraf, R.: Fashion-MNIST: a novel image dataset for benchmarking machine learning algorithms. CoRR abs/1708.07747 (2017). http://arxiv.org/abs/1708.07747

24. Xie, C., Koyejo, S., Gupta, I.: Asynchronous federated optimization. CoRR abs/1903.03934 (2019). http://arxiv.org/abs/1903.03934
25. Yaga, D., Mell, P., Roby, N., Scarfone, K.: Blockchain technology overview. arXiv preprint arXiv:1906.11078, October 2018
26. Zhou, Z., Chen, X., Li, E., Zeng, L., Luo, K., Zhang, J.: Edge intelligence: paving the last mile of artificial intelligence with edge computing. Proc. IEEE **107**(8), 1738–1762 (2019)