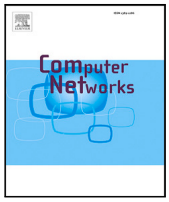
Computer Networks 197 (2021) 108327

Contents lists available at ScienceDirect

Computer Networks

journal homepage: www.elsevier.com/locate/comnet



Federated deep reinforcement learning based secure data sharing for Internet of Things

Qinyang Miao [a,b], Hui Lin [a,b,*], Xiaoding Wang [a,b,*], Mohammad Mehedi Hassan [c]

[a] College of Mathematics and Informatics, Fujian Normal University, Fuzhou, Fujian, 350117, China
[b] Engineering Research Center of Cyber Security and Education Informatization, Fujian Province University, Fuzhou, Fujian, 350117, China
[c] Information Systems Department, College of Computer and Information Sciences, King Saud University, Riyadh 11543, Saudi Arabia

ARTICLE INFO

Keywords:
Secure data sharing
Federated learning
Deep reinforcement learning
IoT

ABSTRACT

The increasing number of Internet of Things (IoT) devices motivate the data sharing that improves the quality of IoT services. However, data providers usually suffer from the privacy leakage caused by direct data sharing. To solve this problem, in this paper, we propose a Federated Learning based Secure data Sharing mechanism for IoT, named FL2S. Specifically, to accomplish efficient and secure data sharing, a hierarchical asynchronous federated learning (FL) framework is developed based on the sensitive task decomposition. In addition, to improve data sharing quality, the deep reinforcement learning (DRL) technology is utilized to select participants of sufficient computational capabilities and high quality datasets. By integrating task decomposition and participant selection, reliable data sharing is realized by sharing local data models instead of the source data with data privacy preserved. Experiment results show that the proposed FL2S achieves high accuracy in secure data sharing for various IoT applications.

1. Introduction

Internet of Things (IoT), as a promising communication paradigm, has a variety of applications in industry, transportation, environment, smart home and so on [1]. In IoT, various smart devices generate a large amount of data including medical data, environmental data, and security data. Through data sharing, IoT can provide high-quality services. However, direct data sharing in IoT faces the following challenges. For example, participants in the data sharing can hardly build trust with each other. Therefore, how to ensure the reliability of the shared data might be a challenge. In addition, due to the privacy concern [2], participants are not willing to share their own data without proper privacy preserving mechanisms. Thereby, both data reliability and data privacy preservation should be ensured in data sharing.

With the development of Artificial Intelligence (AI), there has been a growing interest in machine learning based privacy protection [3]. For example, as a solution to distributed secure data sharing, Federated Learning (FL) has received a widespread attention in a variety of industry applications. Compared with traditional machine learning technologies that require all data for model training in a central device, the FL [4] reduces the computational burden of centralized equipment by allowing each participant in the data sharing to complete the training task locally with data privacy preserved [5,6]. To be specific, each participant receives the initial model from the server and train it with own local data. Once the training is completed, the model parameters are sent to the server for aggregation. Eventually, the final aggregated model is shared, and the data of the participants are protected in the entire data sharing process without exposure. This provides a parallel data sharing scheme for users, organizations or institutions of equal status to realize fair cooperation, ensuring the participants to exchange information safely as individuals. However, the reliability of the final model is not guaranteed. Considering the computation resources required in secure data sharing, the edge computing [7–9] that provides computing, storage, and application platforms for services nearby can be integrated with the federated learning to address above problems with a secure data sharing framework built as shown in Fig. 1. As shown in Fig. 1, data are transmitted between various terminals and a cloud server in IoT [10]. The data of the terminal are stored locally. When data are shared, each terminal uses the local data for model training and sends the trained data model to the cloud server. The cloud server is responsible for aggregating the collected models to form a global model, and then share the global model to each user terminal with data requirements. Data sharing enables more institutions and organizations to make full use of existing data resources and saves the cost of resource collection. Based on this framework, in this paper, we

* Corresponding author.
E-mail addresses: 18291972715@163.com (Q. Miao), linhui@fjnu.edu.cn (H. Lin), wangdin1982@fjnu.edu.cn (X. Wang), mmhassan@ksu.edu.sa (M.M. Hassan).

https://doi.org/10.1016/j.comnet.2021.108327
Received 14 November 2020; Received in revised form 10 June 2021; Accepted 10 July 2021
Available online 17 July 2021

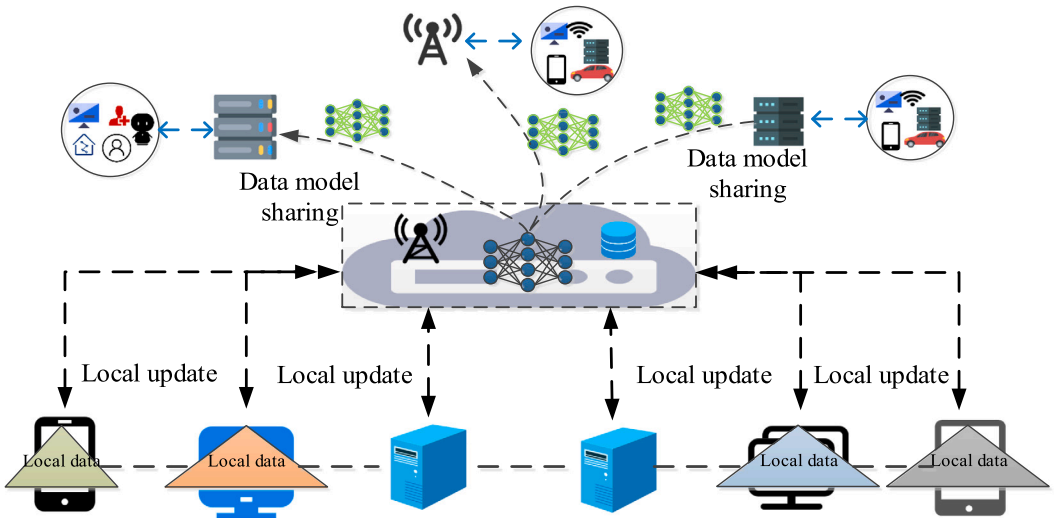


Fig. 1. Example of IoT data sharing scenario.

Table 1
Main notations and symbols.

Notations	Description
T_r	Data request task
w_0	Coordination server initialization model parameters
w_{t+1}^i	Model parameters after local update of data node
L_i	Data node location
C_s^i	Data node connection status
λ_i	Data node selection status
$a_\lambda(t)$	Action selection in deep reinforcement learning system
$R(t)$	Slot reward in DRL
C_i	Local test accuracy of each node
Q^π	DRL actor network
Q^Q	DRL critic network

propose a Federated Learning based Secure data Sharing mechanism (FL2S) for IoT with the privacy preservation. The contributions of this paper are summarized as follows:

- We propose a secure and reliable data sharing mechanism based on federated learning to realize data sharing in IoT. Specifically, an asynchronous multiple federated learning scheme using sub-task grading is proposed. Based on multiple sub-tasks, the deep reinforcement learning algorithm within the federated learning framework is used to select data nodes with high data quality and sufficient computational capability in order to improve the efficiency of data sharing and achieve the protection of the task privacy of the data requester.
- We combine homomorphic encryption into the model parameter aggregation process of federated learning to further protect the privacy of the data provider during the data sharing process.
- Experiment results show that the proposed FL2S achieves high accuracy in secure data sharing for various IoT applications.

The rest of this paper is organized as follows. The related work is introduced in Section 2. The system model is given in Section 3. In Section 4, the asynchronous federated learning with task privacy protection is elaborated. The performance evaluation is given in Section 5. Section 6 summarizes this paper.

2. Related work

In the process of realizing resource sharing, the cloud service model and mobile edge computing provide us with feasible solutions [11,12]. In [13], a framework for sharing medical data in a mobile cloud environment is proposed. The decentralized nature of the blockchain provides a solution for reliable and secure data sharing under mobile cloud computing. In [14], the author developed a new platform to analyze data resources in the Internet of Things by sharing data, so that both the demander and the provider can benefit. However, in actual applications, cloud services require high bandwidth due to data reaching the cloud, and the efficiency is limited when bandwidth resources are limited. Edge computing [15] can offload some tasks that need to be completed in the cloud, reducing the pressure on cloud computing, and improving the performance of applications based on cloud services. The combination of blockchain and edge computing can better realize edge network data storage and computing [16]. In the paper [17], a new distributed architecture was proposed to deploy large-scale applications, supporting scalability and realizing effective information sharing. In [18], the author considers the privacy and security of data sharing, and proposes a data sharing model based on edge computing, which realizes anonymous data sharing and access control. In [19], proposes a low-latency hybrid data sharing framework, which reduces the response delay by seeing the data location as the inner and outer parts of the sub-region. The development of machine learning has brought new opportunities for the intelligentization of edge computing. In the edge computing environment with high-latitude challenges, machine learning can help us quickly find solutions. Using deep learning in edge networks can ensure privacy and security and improve bandwidth efficiency [20]. In the paper [21], it is explained that machine learning enhances the function of mobile edge computing, allowing us to build more complex systems. However traditional machine learning with centralized data for model training is not suitable for distributed real-world scenarios. Although the current work has solved the efficiency problem of sharing resources, the privacy and security problems faced by users still exist.

Federated learning uses the idea of distributed machine learning to place model training locally on each participant for training, providing us with a new method of protecting privacy. In [22], the author proposed an efficient and privacy-protected federated learning scheme, which prevents data leakage while realizing data sharing. In [23], in order to meet the environmental requirements of federated learning, a new compression framework is proposed, which is suitable for training environments with limited bandwidth. In traditional federated learning, each participant receives the model of the server and uploads the model parameters to the server for aggregation after the local training is completed. This synchronization mode will inevitably affect the overall training efficiency. The existing scheme optimizes the efficiency of traditional federated learning. In [24], node screening is performed by

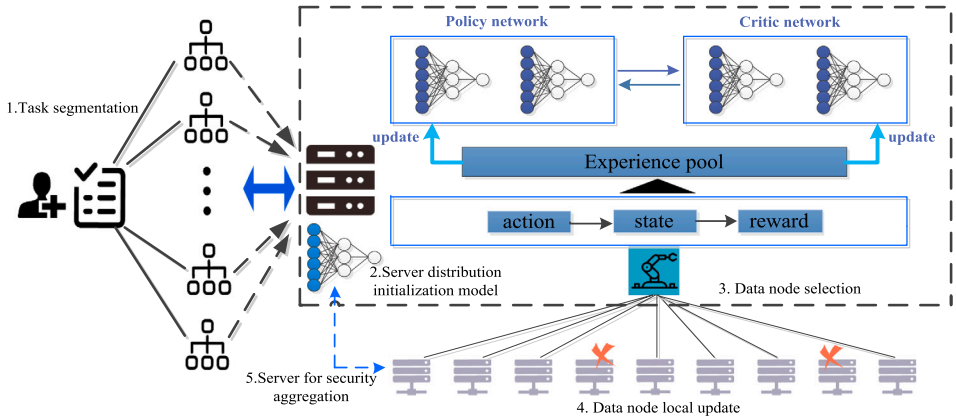


Fig. 2. The proposed system architecture.

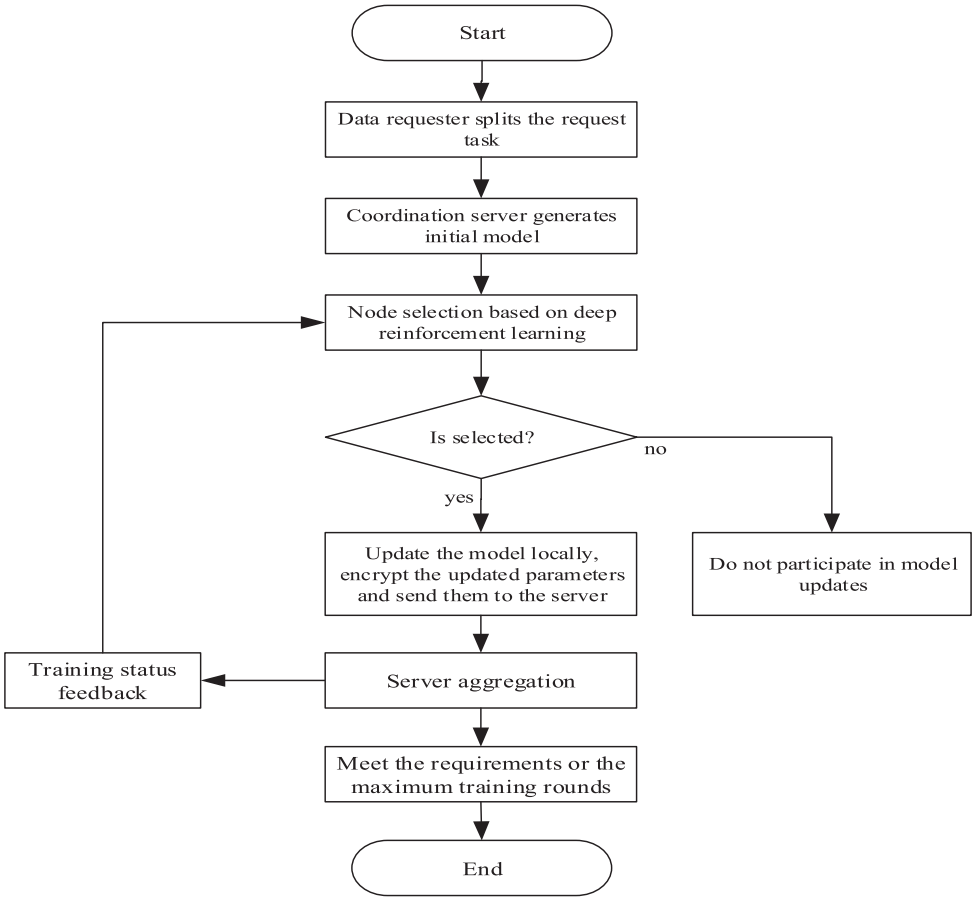


Fig. 3. Structure flow chart.

evaluating the feedback of participants and then iteratively updating the weight of customers. In [25], the author transformed the data sharing problem into a training model problem, and effectively protected data privacy by combining federated learning with blockchain. Although the existing work has contributed to data privacy, considering the actual scenario, the privacy of the request task issued by the data requester should also be paid attention to, in order to protect the privacy of the data provider and the request task issued by the data requester. In this paper, we propose a hierarchical asynchronous federated learning scheme, which not only guarantees the efficiency of data sharing and the privacy of the provider, but also protect the task privacy of the data requester.

3. System model

In this part, we first introduce the framework of the proposed system model, followed by the attack model. Table 1 summarizes the symbols used in this paper.

3.1. System architecture

In this paper, we consider an efficient and reliable multi-party data sharing scenario. The system includes the following three-party entities: the data requester, the data node, and the coordination server. The data requester is the user who made the data sharing request. Data nodes are users who own data and are willing to share. The aggregation server

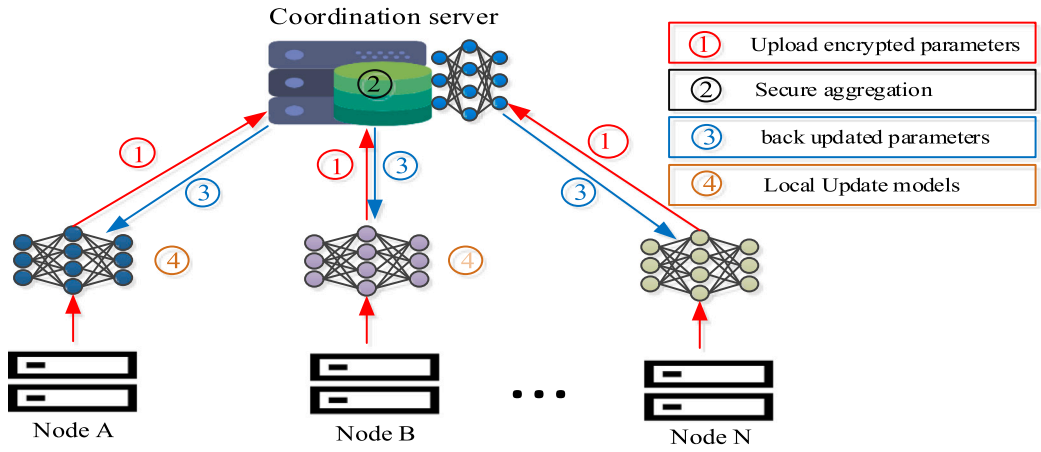


Fig. 4. Security aggregation model.

is responsible for cooperating with data nodes to realize the federated learning process. The data requester sends a data request to the server, the server forwards the request to each terminal node, and the node willing to share the data will respond to the request. The nodes with data in the proposed system model will not directly provide the local raw data to the data requester, but realize data sharing through the joint training of a data model. Considering that in actual situations, the data quality and the computing capability of each data node are not the same. For example, the quality of the data set is poor (i.e., the data are incomplete or the data are with inaccurate labels) and unreliable nodes with limited computational capability will have adverse effects on both speed and quality of the entire training process. On the other hand, to protect the privacy of task T_r of the data requester, data nodes and data request tasks are classified such that data nodes can only complete tasks of corresponding levels and the unselected data nodes in each task are forbidden to access the final task. Specifically, the data requester first divides the task into multiple subtasks $\{T_{r_1}, T_{r_2}, \ldots, T_{r_n}\}$ before sending the data request. Then, the deep reinforcement learning method and multiple subtasks are used to choose the data nodes to completes each subtask including the final task. The proposed federal learning architecture for task privacy protection is shown in Fig. 2.

Federal learning is a distributed machine learning framework. It connects the local training models of all participants with other than original data by technical means, so it solves the privacy problem of data sharing in distributed scenarios. Federation learning needs to coordinate the server to aggregate the training models of each participant for many times to form the global model, specifically, asynchronous aggregation in this paper. Because it is the collaborative training of participants for data sharing, the aggregation server will not conduct model training after generating the initial model, and only responsible for the aggregation process of the model. Therefore, in each round of federal learning process, the aggregation server will distribute the aggregated model to each participant for further training, and then deliver the model to the aggregation server after the local training of the participants is completed for aggregation. This process is repeated.

According to the architecture, we give the data sharing process as follows. The data requester divides the data request task into multiple subtasks before issuing the data request, and sends each subtask to the coordination server, then the server delivers the pre-trained model of each subtask to each data node. For each subtask, in each round of the federated learning process, the deep reinforcement learning method is used to select the data nodes participating in the model training. The model parameters are sent to the coordination server, and the server performs aggregation and then sends it out. The process of aggregation-distribution-aggregation is performed iteratively. If the accuracy requirements or the maximum number of training rounds are reached, the training will be stopped. For each subtask, this process is performed to select a batch of nodes for completing the final data request task. The flowchart of this process is shown in Fig. 3.

3.2. Attack model

In this paper, we considered the problem of privacy leakage. Since the server and the data provider are honest but curious, they are vulnerable to the following two privacy threats. The first one is against data privacy. Directly sharing the original data will cause the data provider's sensitive data to be exposed. The second one is against task privacy. If the requesting task is visible to all data providers, it will cause unnecessary privacy leakage. In order to solve the above-mentioned privacy problems, we use a sharing model that is not original data to complete data sharing and divide the data request task so that the final task is invisible to unselected data nodes.

4. The implementation of the proposed FL2S

In this section, the proposed FL2S consists of the task division and the secure Asynchronous Federated Aggregation Process.

4.1. Task division

In this solution we classify the data request tasks and data nodes, each data nodes can only complete tasks of their corresponding levels. For data nodes of any level, we use subtasks to select when completing a task. Specifically, the data requester first performs task segmentation on the requested task before issuing the data request. The first request is put forwarded based on multiple subtasks, and the final request task is raised at the end. We divide data request tasks into classification tasks and regression tasks. For multi-classification tasks, we can consider dividing the multi-classification tasks into multiple binary classification tasks. For example, for a classification task with N categories, pairwise matching can convert the $\frac{N(N-1)}{2}$ classification problem into a binary classification problem. Specifically, in this paper, we use the cifar10 data set for experimental verification. Cifar10 is a 10-classification task. When performing the first task segmentation, we divide the 10 classification into two classification tasks for identifying airplanes and cars. First, the selected data node is divided into the second task, the task of the segmentation is the two classification task of identifying birds and cats, and so on, the third task is the second classification task of identifying deer and dogs, and the segmentation task is the second classification task of identifying deer and dogs. The fourth task is the binary classification task of identifying frogs and horses. The experimental results will be given in the following chapters. For regression tasks, we can consider dividing the prediction tasks into short-term prediction tasks, mid-term prediction tasks, and future prediction tasks. For example, for power load forecasting tasks, the forecasting tasks can be divided into short-term load forecasting, mid-term and future forecasting. Short-term load forecasting includes load forecasting for

hours, days, and weeks, and mid-term load forecasting for months. Future load forecasting is a forecast for the next few years. The specific task division is performed by the data requester. The purpose of the adjudication is to screen out the data nodes that are most suitable for completing the data request task. Here we provide a way of dividing.

4.2. Secure asynchronous federated aggregation process

4.2.1. Secure federated aggregation

Federated learning aims to use distributed data sets to train a common model and break the dilemma that data owners are unwilling to share data due to concerns about privacy leakage. Since federated learning achieves the purpose of data sharing without leaking the privacy of data nodes in actual operation, it coincides with our purpose, so in this paper we use federated learning to complete the task model training of data requests. When data sharing is performed, the data request task is divided first. For each subtask T_{r_i}, multiple iterations of training are performed until the server checks that the model prediction accuracy meets the data requester's requirements or reaches the preset maximum training round. The federated learning stops iteration. The server initializes the model parameters w_0 and sends the original model parameters to each data node. In each execution round of federated learning, we use a deep reinforcement learning-based node selection algorithm to find reliable and efficient nodes, and only the selected nodes will update the model parameters. The specific algorithm will be introduced in the next section. In a time slot t, the data node will calculate $g_i = \nabla F_i(w_t)$, that is, the average gradient in the local data of the current model parameters w_t. According to formula (1), the data node uses local data and the existing model to further perform the gradient descent step and send the locally updated model parameters w^i_{t+1} to the server, and the server uses Eq. (2) to perform the model averaging operation:

$$w^i_{t+1} \leftarrow \bar{w}_t - \kappa g_i \tag{1}$$

$$\bar{w}_{t+1} \leftarrow \sum_{1}^{n} \frac{n_i}{n} w^i_{t+1} \tag{2}$$

Among them, κ is the learning rate of gradient descent during the local model training of the participants, $\bar{w}_t$ is the model parameter currently distributed by the aggregation server, and n is the number of participants. Next, the server sends the averaged results of the model to each data node, and each node performs parameter updates. This process is iteratively performed until the model meet the accuracy requirements or the maximum training round is reached.

Considering that in actual operation, during the process of data node interacting with the server, if the data is transmitted in plaintext, it is likely to be attacked by the attacker. Although the data obtained by the attacker is not the original data, it will be affected by the trained model parameters. Leaking the privacy of data nodes. In order to solve the problem of privacy leakage in the process of model parameter sharing, in this solution, we use additive homomorphic encryption to encrypt and transmit model parameters. As shown in Algorithm 1, the initialized model parameters are broadcast on the coordination server at the same time, the homomorphic encryption and decryption keys are issued to each data node, and the data nodes that receive the model and encryption and decryption keys perform iterative training locally to obtain the parameter updates of the local model, and then add the updated parameters the encrypted model parameters obtained by homomorphic encryption are sent to the coordination server. After the coordination server receives these model parameters, it aggregates, that is, averaging the encrypted model parameters, and then sends the encrypted model parameters to the data node. The process is repeated until the server check model meet the accuracy requirements or reaches the maximum training round. The secure aggregation model is shown in Fig. 4. A total of four steps are involved. The node locally updates the model and encrypts the model parameters and sends it to the server. The server aggregates, returns to the node after the aggregation is complete, and updates the node after local decryption. This process is repeated until the model reaches the required accuracy or reaches the maximum training round and then stops training.

The additive homomorphic encryption algorithm used in this paper is the paillier encryption algorithm. Since the ciphertext multiplication is equal to the plaintext addition, the coordination server will accumulate the received encrypted model parameters, the formula is as follows, among them, w^i_{t+1} is the model parameter of the ith participant updated locally in a time slot t.

$$\frac{1}{n} \Pi^n_i E\left(w^i_{t+1}\right) = \frac{1}{n} \sum_{i}^{n} w^i_{t+1} \tag{3}$$

Algorithm 1 Security aggregation of models

1: The server generates a model locally, initializes model parameter, and then broadcasts it to data nodes;
2: **for** global model iteration rounds $t = 1, T$ **do**:
3: the server selects data node η_i according to DRL;
4: **for** data node local update round $s = 1, S$ **do**:
5: receive model parameter $\bar{w}_t$,calculate gradient g_i;
6: update parameter $w^i_{t+1} \leftarrow \bar{w}_t - \kappa g_i$ locally;
7: execute addition homomorphic encryption to get $E(w^i_{t+1})$;
8: return $E(w^i_{t+1})$ to server;
9: **end for**
10: server checks whether reach accuracy or training rounds;
11: **end for**

4.2.2. Node selection based on deep reinforcement learning in federated learning

The training efficiency of the client–server federated learning architecture depends on the data set quality of each participating node and its computing power. Nodes with poor data set quality and weak computing power will have a negative impact on model training, so select the data set quality nodes with high computing power are necessary. Therefore, our goal is to select a subset of nodes from multiple nodes so that the training time is the shortest and the model is the best.

Deep reinforcement learning combines the perception ability of deep learning with the decision-making ability of reinforcement learning, and is an artificial intelligence method closer to human thinking. So in order to effectively select nodes, we use deep reinforcement learning [7] to achieve the goal. The node selection algorithm (DRNS) based on deep reinforcement learning is shown in Algorithm 2. The specific introduction is as follows:

The basic principle of DDPG is use value to update the strategy, using convolutional neural network as the simulation of the strategy function π and Q function, that is, the strategy network and Q network, and then use deep learning methods to train the above two networks. DDPG creates two neural networks for the policy network and the Q network respectively, called online network and target network. The policy network and the Q network are also called an actor network and a critic network, respectively. We use $\pi(s|Q^\pi)$ to denote the online actor network, use π' to denote the target actor network, use $Q(s|Q^Q)$ to denote the online critic network, and Q' used to denote the target critic network. Experience replay is used in DDPG to make the states independent of each other. Next, we will introduce the sampling process and the two network training processes. First, we need to make the following definitions, defining the relevant states, actions, reward functions, and strategy functions as follows:

State: In each time slot t of federated learning, the system state consists of the position of each node $L_i(i \in \{1, 2 \ldots N\})$, the connection state of each node $C^i_s(i \in \{1, 2, \ldots, N\})$, and the node selection state

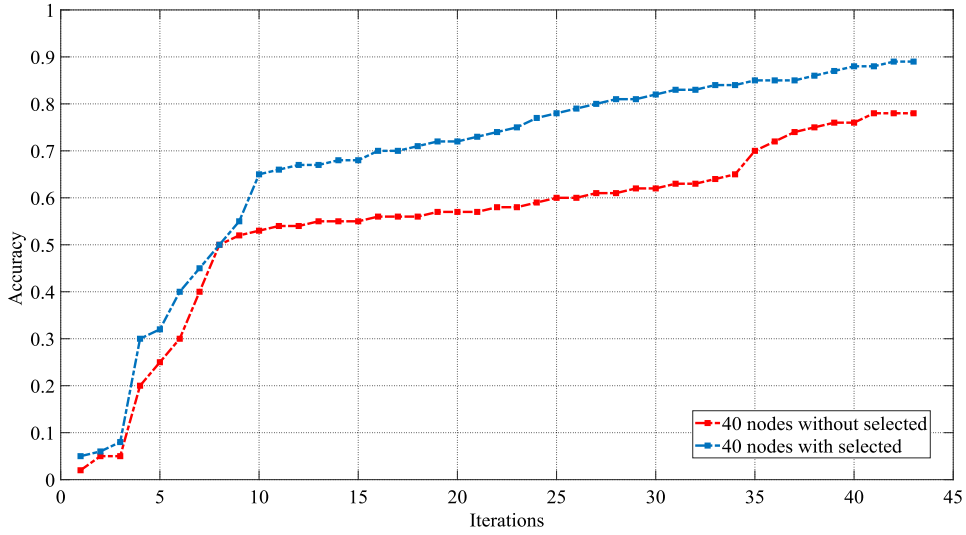


Fig. 5. Accuracy under 40 data nodes.

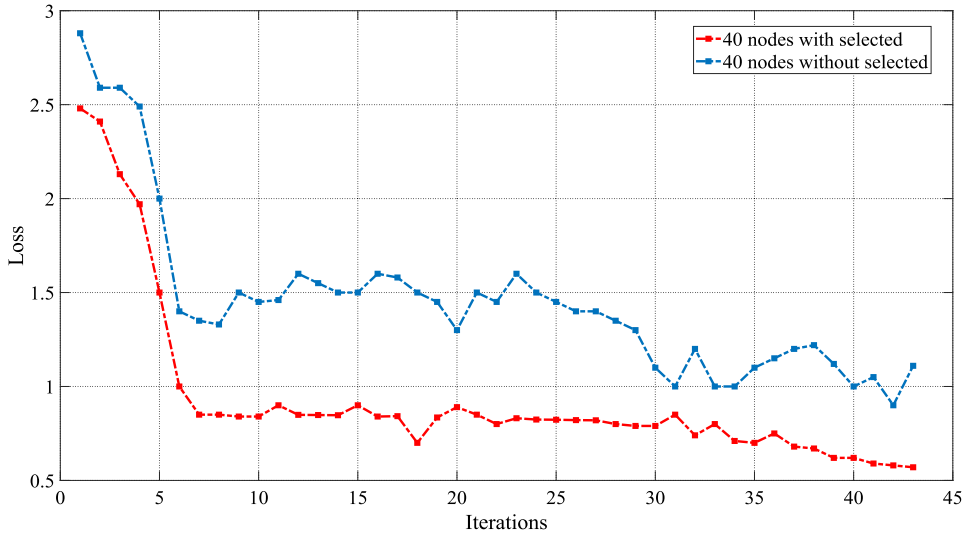


Fig. 6. Loss under 40 data nodes.

$\lambda_i (i \in \{1, 2, \ldots, N\})$. Therefore, the system state in a time slot t can be expressed as:

$$\begin{aligned} s(t) = [&L_1(t), L_2(t), \ldots, L_N(t), \\ &C_s^1(t), C_s^2(t), \ldots, C_s^N(t), \\ &\lambda_1(t), \lambda_2(t) \ldots, \lambda_N(t)] \end{aligned} \tag{4}$$

Action: In deep reinforcement learning, the agent needs to decide which nodes are selected to participate in the model training. There are two situations: nodes are selected or not, which can be reduced to a 0–1 problem, this action vector can be expressed by the formula.

$$a(t) = [a_{\lambda_1}(t), a_{\lambda 2}(t), \ldots, a_{\lambda N}(t)] \tag{5}$$

among them, $a_{\lambda_i}(t) = 1$ represents that this node is selected by the agent, and $a_{\lambda_i}(t) = 0$ represents that it is not selected by the agent. The agent randomly selects a certain number of data nodes to update the model according to the perceived node location, node connection status, and node selection status. The result of the selection is determined by the reward. The more nodes are not selected, the better. Considering the model training delay and model accuracy during the model update process, the reward algorithm will be given below.

Reward: The agent in deep reinforcement learning evaluates the effectiveness of the action based on the reward, and uses it to update the strategy, that is, in the time slot t, the agent adopts action $a(t)$ in the state $s(t)$ and the effect of $a(t)$ is evaluated based on rewards. In this paper, there are two main factors that affect the reward, one is the time that the node receives the model parameters from the server until it sends the updated parameters to the coordinator server, the other is the time that the coordinator sends the model parameters to check the convergence of the model parameters. In addition, considering the quality of the final model, model accuracy is also added to the reward function. Through the above analysis, the tasks of each time slot mainly include:

(1) Node P_i conducts model training based on the local data set and model, and sends the trained parameters to the server.

(2) The server performs parameter aggregation operations. This process is iterated until the model parameters converge. Therefore, the time slot reward can be expressed as:

$$R(t) = \alpha R_{p_i}^{time}(t) + \beta R_m^{qos}(t) \tag{6}$$

where $\alpha + \beta = 1$, α and β are debugging functions, and $R_{p_i}^{time}(t)$ is delay rewards, which are determined according to the convergence time of the final model parameters. The larger the delay, the smaller

the reward. $R_{p_i}^{time}(t)$ can be expressed as:

$$R_{p_i}^{time}(t) = -D_m^f - T_r \tag{7}$$

where D_m^f is the total time for the server to perform the weighted average of the model parameters, T_r is the time required for the server to receive the updated parameters of the last node from when the model parameters are delivered, $R_m^{qos}(t)$ is reward for model quality, which is determined according to the accuracy of the final model, $R_m^{qos}(t)$ can be expressed as:

$$R_m^{qos}(t) = \frac{1}{N}\sum_{1}^{n} \lambda_i C_i \tag{8}$$

where C_i is the local test accuracy of each node. N is the number of data nodes.

Strategy: Strategy π is the mapping from state to action: $s(t) \rightarrow a(t)$, in the time slot, the action to be taken can be calculated by the strategy calculation method. For deep reinforcement learning, actions are generated by a neural network. The input of the neural network is the state of the system, and the output is the action to be taken. The goal of reinforcement learning is to find strategies that maximize long-term return expectations: $\pi = \arg\max \sum_{t=0}^{T} R(t)$

The sampling process is to select an action $a_\lambda(t) = \pi(s(t)|\theta^\pi)$ in the state $s(t)$ and then execute this action to observe the reward $r(t)$. After observing the new state $s(t+1)$, it will be stored in the experience pool R. The essence of training is to optimize the actor network and the critic network. The goal of training is to maximize the reward function of the actor network and minimize the loss function of the critic network. The loss function of critic network is defined as:

$$L(\theta_Q) = \frac{1}{N}\sum_{t}(y(t) - Q(s(t), a_\lambda(t)|\theta^Q))^2 \tag{9}$$

The critic network is updated according to the Q value. The current state $s(t)$ is input in the online actor, and the current action $a_\lambda(t)$ is output. After the action $a_\lambda(t)$ is executed, the new state $s(t+1)$ and Q value are obtained:

$$Q(s(t), a_\lambda(t)) = E[r(s(t), a_\lambda(t)) + \gamma Q(s(t+1), \pi(s(t+1)))] \tag{10}$$

in formula (9), $y(t)$ is the target value of the target critic network, expressed as:

$$y(t) = r(t) + \gamma Q'\left(s(t+1), \pi'\left(s(t+1) \mid \theta^{\pi'}\right) \mid \theta^{Q'}\right) \tag{11}$$

The target value $y(t)$ can be calculated from the current reward and the Q value $Q'\left(s(t+1), \pi'\left(s(t+1) \mid \theta^{\pi'}\right) \mid \theta^{Q'}\right)$ of the next executed action, where is the action to be executed in the next state. γ is the attenuation factor. In addition, the actor online network is responsible for updating the strategy. It selects actions $a_\lambda(t)$ based on the current state $s(t)$. The actor target network selects the best next action $a_\lambda(t+1)$ based on the next state $s(t+1)$ sampled by the experience pool. The actor network is based on the following equation updated:

$$\nabla_{\theta^\pi} J \approx_{\pi} \left[\nabla_a Q\left(s, a \mid \theta^Q\right)\Big|_{s=s(t), a=\pi(s(t))} \nabla_{\theta_\pi} \pi\left(s \mid \theta^\pi\right)\Big|_{s=s(t)}\right] \tag{12}$$

For the same state, the online actor network will output different actions, as the input of the online critic network will get different feedback Q values, according to the feedback actor network will increase or decrease the probability of the corresponding action, in order to get a larger Q value. The update functions of the target actor network and the target critic network are as follows:

$$\begin{aligned} \theta^{Q'} &\leftarrow \tau\theta^Q + (1-\tau)\theta^{Q'} \\ \theta^{\pi'} &\leftarrow \tau\theta^\pi + (1-\tau)\theta^{\pi'} \end{aligned} \tag{13}$$

among them, τ is the soft update coefficient.

Algorithm 2 Node preferential algorithm based on DDPG

```
Initialization of neural network parameters of online network Q^π and Q^Q;
Initialize experience pool R;
for each episode do:
    Initialize the random process as the introduced noise;
    for t = 1, T do:
        Actor network selects an action a_λ(t) according to strategy π;
        Executing a_λ(t) returns R(t) and the new state s(t + 1);
        Store [s(t), a_λ(t), R(t), s(t + 1)] in experience pool R;
        Randomly sampling data from R as the next mini-batch;
        Update online critic network: Put L(θ_Q) in optimizer to minimize;
        Update online actor network: update Q^π with optimizer;
        Update target network parameter θ^{Q'} and θ^{π'};
    end for
end for
```

5. Performance evaluation

5.1. Experimental setup

The simulation is completed on a computer equipped with a Windows7 system, the machine is configured with an Intel core i7 processor, a CPU frequency of 6.4 GHZ, and the python programming language is used to verify the effectiveness of the proposed scheme. In this section, we will evaluate the performance of the proposed FL2S. First, we study the performance of asynchronous federated learning based on subtask classification. Secondly, we test the node selection algorithm based on deep reinforcement learning.

In order to test the effectiveness of the proposed scheme, we considered the data sharing process for the same data request task in two different situations. In one case, there are forty data nodes participating in the federated learning process in a network area. And the coordination server is responsible for aggregating and distributing the training parameters of these data nodes. Another situation is that there are 80 data nodes participating in federated learning in a region, and the coordination server ensures that the training process is carried out efficiently and securely. The data set size, quality, and local computing power of each data node are random. In addition, we divide a data request task into four subtasks.

We evaluate our proposed asynchronous multi-federated learning process on the cifar10 dataset. Cifar10 contains a total of 10 categories of RGB color pictures, namely: airplanes, cars, birds, cats, deer, dogs, frogs, horses, boats and trucks. The size of the picture is 32∗32. Because cifar10 is a multi-classification task. So we split the cifar10 classification task into four binary classification tasks to select the data nodes in stages. The data set is randomly divided and then distributed to data nodes participating in federated learning. The sharing task involved in this paper is to share the parameters of the local data training model of the shared data node. The coordination server uses the deep residual network to train the local model. In each iteration, there is a process of aggregation and delivery of model parameters. Then, we verified the data node selection process based on DDPG.

5.2. Numerical results

We have used the cifar10 dataset to evaluate the accuracy and loss of the scheme described in this article in a network area with forty data nodes and 80 data nodes. As shown in the figure, the model training results after the final task request is issued. The results show that the

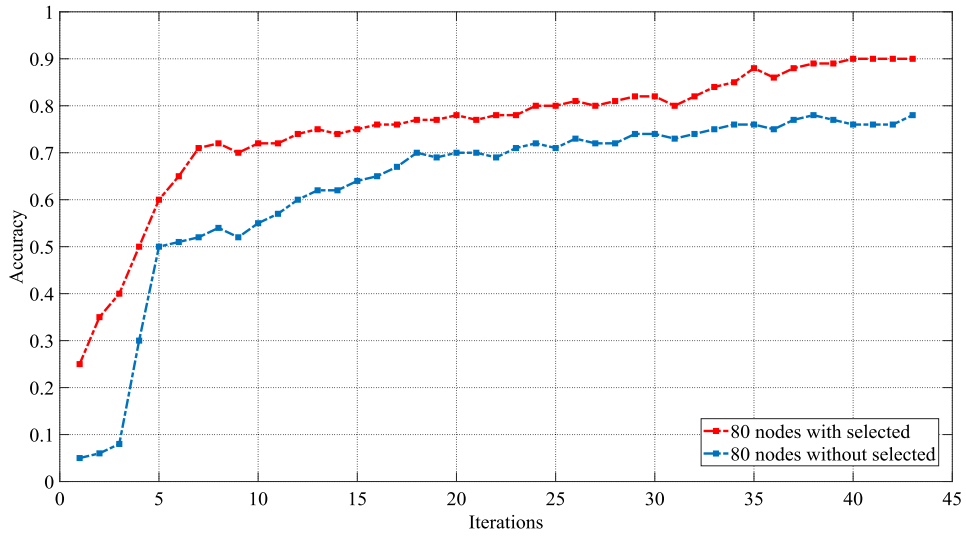


Fig. 7. Accuracy under 80 data nodes.

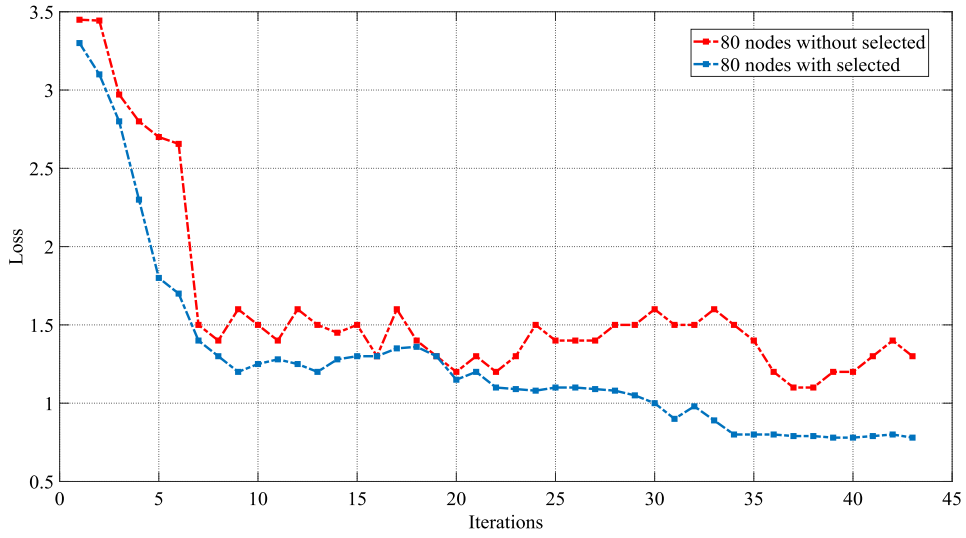


Fig. 8. Loss under 80 data nodes.

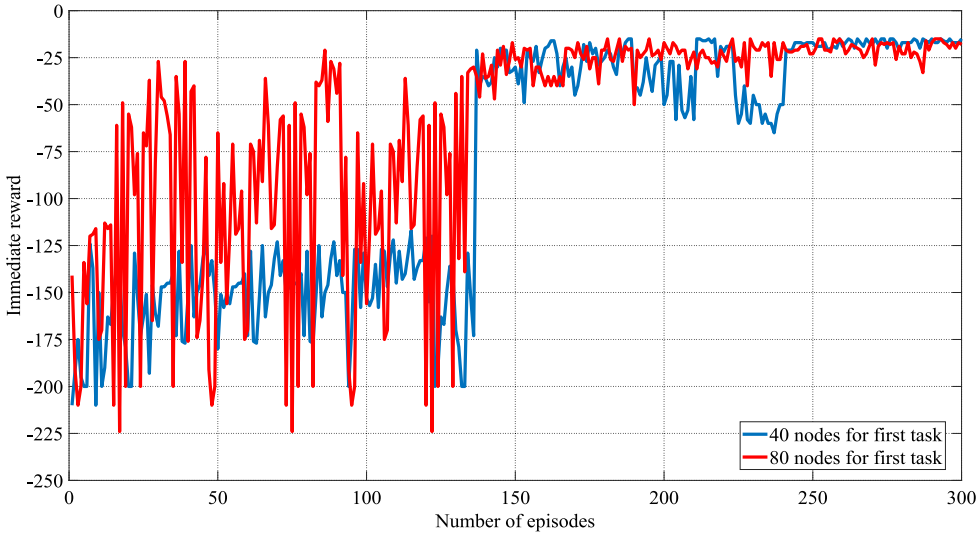


Fig. 9. DDPG average reward under the first subtask.

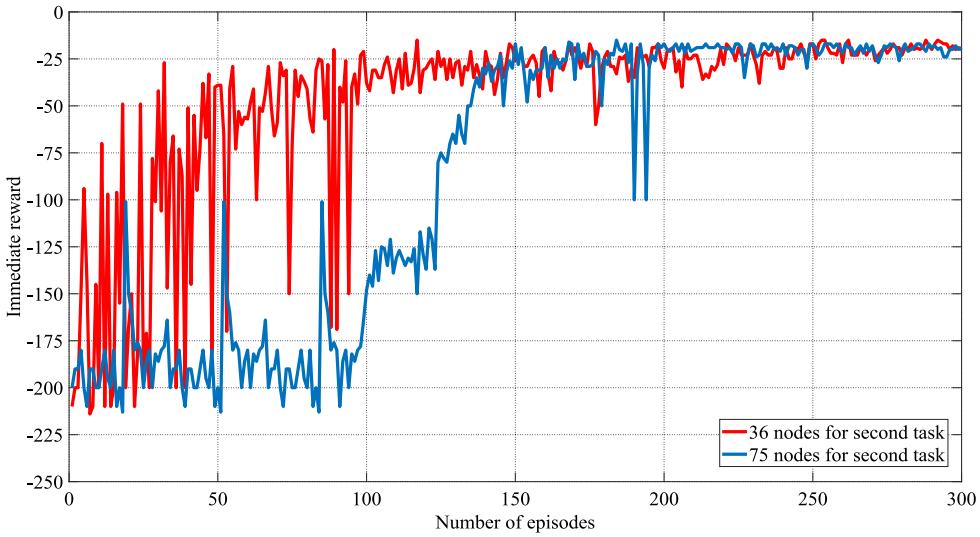


Fig. 10. DDPG average reward under the second subtask.

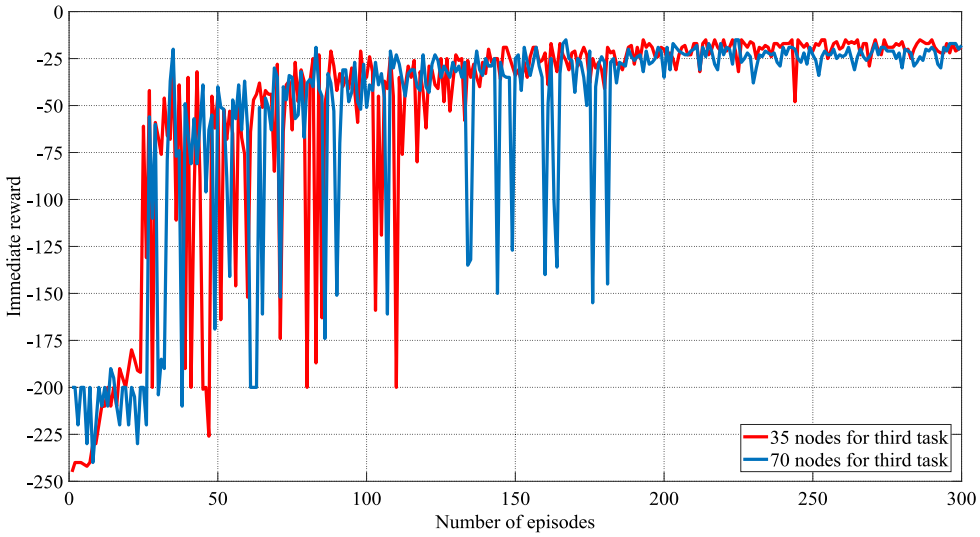


Fig. 11. DDPG average reward under the third subtask.

selection of data nodes can have better accuracy and convergence. In the two cases, there are data nodes with small data sets, poor data set quality and poor computing power in the network area, so there is no basis for subtasks. When selecting nodes, the accuracy of the final data request task will decrease, and the loss will be too large. On the contrary, the filtered data nodes perform better in learning. This is because in model training, high-quality data sets mean more complete and accurate data. The labels and features of the data are more reliable and effective, and the labels determine the final value of the model, and the features determine the capability boundary of the model. The results show that the proposed scheme can prevent inefficient data nodes from affecting the completion quality of the final requested task.

As shown in Fig. 5, in a network area with forty data nodes, for the same data request task, when the learning rate is 0.01, the global model trained by the selected data nodes has higher accuracy and more efficient. As shown in Fig. 6, the faster rate of loss reduction means that the global model reaches a higher accuracy faster.

As shown in Figs. 7 and 8, the experimental results show that the high-quality node training model selected in the network area with 80 data nodes has higher accuracy and better effect.

We further studied the node selection process based on the DDPG algorithm proposed in this paper. The learning rate is set to 0.01, the attenuation factor is set to 0.9, the experience pool size is set to 6000, and the batch size is 128. As shown in Fig. 9, the average instant reward of each segment during the first sub-task. Low-quality data nodes will affect the accuracy of the final model of the sub-task. After completing the first sub-task, we have 40 data nodes and 36 high-quality data nodes and 75 high-quality data nodes were selected from the network of 80 data nodes, and these high-quality nodes selected will continue to participate in the training of the next subtask.

As shown in Fig. 10, the average reward convergence is better than that of the first task, indicating that our proposed node selection scheme worked well. After completing the second subtask, we selected 35 and 70 high-quality data nodes respectively.

Then, perform the third subtask training process in the data nodes selected in the second time, and finally a batch of nodes are selected to complete the fourth sub task. As shown in Fig. 11, after completing the third subtask, 35 nodes and 70 nodes are selected respectively. As shown in Fig. 12, after completing the four subtasks, 35 nodes and 66 nodes are left respectively. The results show that the optimal return can be achieved in both cases.

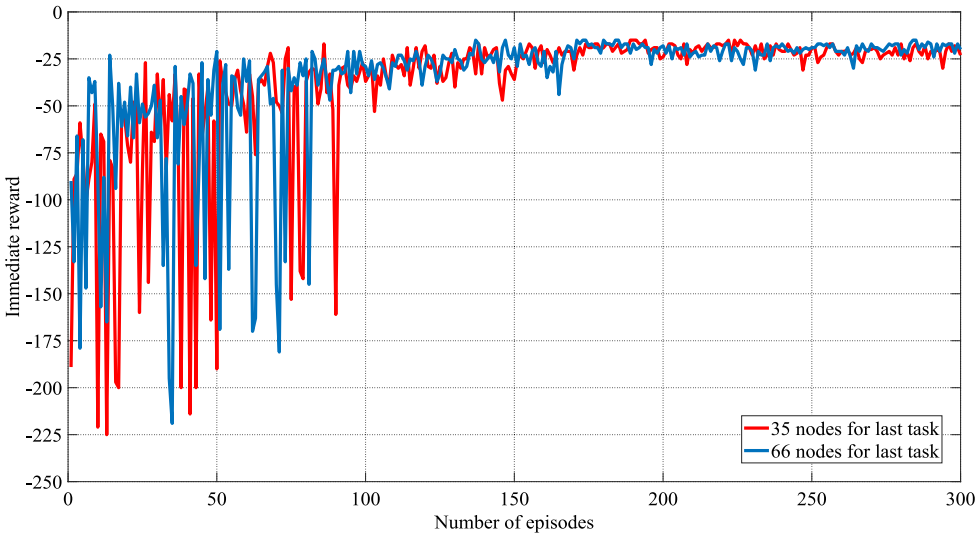


Fig. 12. DDPG average reward under the last subtask.

6. Conclusions

In this paper, we propose a secure data sharing method based on hierarchical asynchronous federated learning in IoT, which protects the data owner's data privacy when sharing data. Specifically, by dividing the data request task, the divided multiple subtasks are used as the task basis for selecting high-quality data nodes, and deep reinforcement learning is applied to the data node selection process to complete each subtask and the final task. And the task privacy of the data requester is also protected. Simulation experiments show that in the network area with different numbers of nodes, the data sharing method proposed in this paper achieves better privacy protection and data quality.

CRediT authorship contribution statement

Qinyang Miao: Developed mechanics modeling and analysis, Calculations. **Hui Lin:** Supervision. **Xiaoding Wang:** Performed sample preparation, Data analysis, Simulations using classical force fields. **Mohammad Mehedi Hassan:** Developed mechanics modeling and analysis, Calculations.

Declaration of competing interest

The authors declare that they have no known competing financial interests or personal relationships that could have appeared to influence the work reported in this paper.

Acknowledgment

This work was supported by King Saud University, Saudi Arabia, under esearchers Supporting Project number RSP-2021/18.

References

[1] S. Balaji, K. Nathani, R. Santhakumar, IoT technology, applications and challenges: A contemporary survey, Wirel. Pers. Commun. 108 (1) (2019) 363–388.

[2] F. Meneghello, M. Calore, D. Zucchetto, et al., IoT: Internet of Threats? A survey of practical security vulnerabilities in real IoT devices, IEEE Internet Things J. 6 (5) (2019) 8182–8201.

[3] T. Li, A.K. Sahu, A. Talwalkar, et al., Federated learning: Challenges, methods, and future directions, IEEE Signal Process. Mag. 37 (3) (2020) 50–60.

[4] J. Mills, J. Hu, G. Min, Communication-efficient federated learning for wireless edge intelligence in IoT, IEEE Internet Things J. 7 (7) (2020) 5986–5994.

[5] J. Mills, J. Hu, G. Min, Communication-efficient federated learning for wireless edge intelligence in IoT, IEEE Internet Things J. 7 (7) (2019) 5986–5994.

[6] Q. Yang, Y. Liu, T. Chen, et al., Federated machine learning: Concept and applications, ACM Trans. Intell. Syst. Technol. (TIST) 10 (2) (2019) 1–19.

[7] J. Wang, J. Hu, G. Min, et al., Fast adaptive task offloading in edge computing based on meta reinforcement learning, IEEE Trans. Parallel Distrib. Syst. 32 (1) (2020) 242–253.

[8] X. Li, X. Huang, C. Li, et al., EdgeCare: Leveraging edge computing for collaborative data management in mobile healthcare systems, IEEE Access 7 (2019) 22011–22025.

[9] M.A. Jan, W. Zhang, M. Usman, et al., SmartEdge: An end-to-end encryption framework for an edge-enabled smart city application, J. Netw. Comput. Appl. 137 (2019) 1–10.

[10] Z. Chen, J. Hu, G. Min, A. Zomaya, T. El-Ghazawi, Towards accurate prediction for high-dimensional and highly-variable cloud workloads with deep learning, IEEE Trans. Parallel Distrib. Syst. 31 (4) (2020) 923–934.

[11] L.M. Dang, M. Piran, D. Han, et al., A survey on Internet of Things and cloud computing for healthcare, Electronics 8 (7) (2019) 768.

[12] Q. Qi, F. Tao, A smart manufacturing service system based on edge computing, fog computing, and cloud computing, IEEE Access 7 (2019) 86769–86777.

[13] D.C. Nguyen, P.N. Pathirana, M. Ding, et al., Blockchain for secure ehrs sharing of mobile cloud based e-health systems, IEEE Access 7 (2019) 66792–66806.

[14] A. Yassine, S. Singh, M.S. Hossain, et al., IoT big data analytics for smart homes with fog and cloud computing, Future Gener. Comput. Syst. 91 (2019) 563–573.

[15] C.H. Hong, B. Varghese, Resource management in fog/edge computing: A survey on architectures, infrastructure, and algorithms, ACM Comput. Surv. 52 (5) (2019) 1–37.

[16] R. Yang, F.R. Yu, P. Si, et al., Integrated blockchain and edge computing systems: A survey, some research issues and challenges, IEEE Commun. Surv. Tutor. 21 (2) (2019) 1508–1532.

[17] S. Almajali, I. Dhiah el Diehn, H.B. Salameh, et al., A distributed multi-layer MEC-cloud architecture for processing large scale IoT-based multimedia applications, Multimedia Tools Appl. 78 (17) (2019) 24617–24638.

[18] Y. Sun, L. Yin, Z. Sun, et al., An IoT data sharing privacy preserving scheme, in: IEEE INFOCOM 2020-IEEE Conference on Computer Communications Workshops, INFOCOM WKSHPS, IEEE, 2020, pp. 984–990.

[19] J. Xie, D. Guo, X. Shi, et al., A fast hybrid data sharing framework for hierarchical mobile edge computing, in: IEEE INFOCOM 2020-IEEE Conference on Computer Communications, IEEE, 2020, pp. 2609–2618.

[20] J. Chen, X. Ran, Deep learning with edge computing: A review, Proc. IEEE 107 (8) (2019) 1655–1674.

[21] T.K. Rodrigues, K. Suto, H. Nishiyama, et al., Machine learning meets computation and communication control in evolving edge and cloud: Challenges and future perspective, IEEE Commun. Surv. Tutor. 22 (1) (2019) 38–67.

[22] M. Hao, H. Li, X. Luo, et al., Efficient and privacy-enhanced federated learning for industrial artificial intelligence, IEEE Trans. Ind. Inf. 16 (10) (2019) 6532–6542.
[23] F. Sattler, S. Wiedemann, K.R. Mller, et al., Robust and communication-efficient federated learning from non-iid data, IEEE Trans. Neural Netw. Learn. Syst. 31 (9) (2019) 3400–3413.
[24] A. Imteaj, M.H. Amini, Distributed sensing using smart end-user devices: Pathway to federated learning for autonomous IoT, in: 2019 International Conference on Computational Science and Computational Intelligence, CSCI, IEEE, 2019, pp. 1156–1161.
[25] Y. Lu, X. Huang, Y. Dai, et al., Blockchain and federated learning for privacy-preserved data sharing in industrial IoT, IEEE Trans. Ind. Inf. 16 (6) (2019) 4177–4186.

Miao Qinyang is now studying for a master's degree in Cyberspace Security. He is studying in the College of Computer and Cyber Security of Fujian Normal University. In 2019, he obtained a bachelor's degree in information security from Xi'an University of Posts and telecommunications. His research interests and directions include deep learning, intensive learning and network security.

Hui Lin is a professor in the College of Computer and Cyber Security at Fujian Normal University, FuZhou, China. He received his Ph.D. degree in Computing System Architecture from College of Computer Science of the Xidian University, China, in 2013. Now he is a M.E. supervisor in the College of Computer and Cyber Security at Fujian Normal University, FuZhou, China. His research interests include mobile cloud computing systems, blockchain, and network security. He has published more than 50 papers in international journals and conferences.

Xiaoding Wang received his Ph.D. in College of Mathematics and Informatics from Fujian Normal University in 2016, he is an Associate Professor with the School of Fujian Normal University, China. His main research interests include network optimization and fault tolerance.

Mohammad Mehedi Hassan is currently a Full Professor of Information Systems Department in the College of Computer and Information Sciences (CCIS), King Saud University (KSU), Riyadh, Kingdom of Saudi Arabia. He received his Ph.D. degree in Computer Engineering from Kyung Hee University, South Korea in February 2011. He has authored and co-authored more than 260+ publications including refereed journals (218+ SCI/ISI-Indexed Journal papers, 4+ ESI highly cited papers, 1 hot paper), conference papers, books, and book chapters. He has served as chair, and Technical Program Committee member in numerous reputed international conferences/workshops such as IEEE CCNC, ACM BodyNets, IEEE HPCC etc. He is a recipient of a number of awards including Distinguished Research Award from College of Computer and Information Sciences, KSU 2020, Best Conference Paper Award from IEEE Int'l Conf on Sustainable Technologies for Industry 4.0 (STI) 2020, Best Journal Paper Award from IEEE Systems Journal in 2018, Best Conference Paper Award from CloudComp in 2014 conference and the Excellence in Research Award from College of Computer and Information Sciences, KSU (2015 & 2016). He is listed as one of the top 2% Scientists of the world in Networking and Telecommunication field. He is one of the top computer scientists in Saudi Arabia as well. He is on the editorial board of several SCI/ISI-Indexed Journals. He has also played role of the guest editor of several international ISI-indexed journals. His research interests include Cloud/Edge computing, Internet of Things, Artificial intelligence, Body sensor network, Big data, Mobile computing, Cyber security, Smart computing, 5G/6G network, and social network. He is a Senior Member of the IEEE.