Article

A DQN-Based Multi-Objective Participant Selection for Efficient Federated Learning

Tongyang Xu, Yuan Liu, Zhaotai Ma, Yiqiang Huang and Peng Liu *

College of Computer and Control Engineering, Northeast Forestry University, Hexing Road 26, Harbin 150040, China; xuty210942@nefu.edu.cn (T.X.); 2021116210@nefu.edu.cn (Y.L.); ma1774888939@nefu.edu.cn (Z.M.); 504176134@nefu.edu.cn (Y.H.)

* Correspondence: liupeng@nefu.edu.cn

Abstract: As a new distributed machine learning (ML) approach, federated learning (FL) shows great potential to preserve data privacy by enabling distributed data owners to collaboratively build a global model without sharing their raw data. However, the heterogeneity in terms of data distribution and hardware configurations make it hard to select participants from the thousands of nodes. In this paper, we propose a multi-objective node selection approach to improve time-to-accuracy performance while resisting malicious nodes. We firstly design a deep reinforcement learning-assisted FL framework. Then, the problem of multi-objective node selection under this framework is formulated as a Markov decision process (MDP), which aims to reduce the training time and improve model accuracy simultaneously. Finally, a Deep Q-Network (DQN)-based algorithm is proposed to efficiently solve the optimal set of participants for each iteration. Simulation results show that the proposed method not only significantly improves the accuracy and training speed of FL, but also has stronger robustness to resist malicious nodes.

Keywords: federated learning; node selection; deep reinforcement learning; multi-objective; model performance

check for updates

Citation: Xu, T.; Liu, Y.; Ma, Z.; Huang, Y.; Liu, P. A DQN-Based Multi-Objective Participant Selection for Efficient Federated Learning. *Future Internet* **2023**, *15*, 209. https://doi.org/10.3390/fi15060209

Academic Editor: Wei Yu

Received: 20 April 2023
Revised: 25 May 2023
Accepted: 31 May 2023
Published: 8 June 2023

1. Introduction

In recent years, artificial intelligence (AI) applications such as ChatGPT are experiencing surging development, which not only benefits from the advance of machine learning and deep learning algorithms but also from the accumulation of enormous data and the support of computing power. However, the traditional AI paradigm has to gather extensive data at the powerful central cloud for centralized model training. This paradigm faces the difficulty of safeguarding data privacy and brings high data transmission costs. To thoroughly exploit the data without leaking privacy, federated learning (FL) has emerged [1], which enables clients to collaboratively learn a global model without sharing their raw data [2,3]. Specifically, the distributed clients train the local model using their own data, and then upload the local model to the central parameter server for global model aggregession. Then, the aggregated model is returned to each client for the next-round iteration. In this way, the global model can be learned iteratively in a distributed and privacy-preserving manner.

Despite the potential to exploit data in a privacy-preserving way and reduce the communication cost, federated learning still faces technical challenges for wide application. Most of all, the large number of distributed clients in FL typically involves heterogeneous data distribution and hardware conditions. How to select the optimal participant set is a vital issue that will not only affect the training efficiency and model performance of the federated learning dramatically [4], but will also be able to enhance the preservation of privacy [5]. However, node selection is still a challenging problem. Firstly, the heterogeneity of hardware and data makes node selection a complex task [6,7]. Secondly, the process of node selection normally involves sensitive personal information [8] which can not be obtained easily. Moreover, it is difficult to balance multiple objectives in node selection [9]; it is tough to get a selection outcome that has shorter training time, higher model accuracy, and stronger trustworthiness. Finally, node selection is mostly an NP difficult problem but it requires high latency. An excellent algorithm needs to provide the optimal selection result with the shortest time cost, which seems to be a contradiction.

In this paper, we propose a node selection method for federated learning based on deep reinforcement learning. It considers training quality and efficiency of heterogeneous devices, and balances multiple objectives to guarantee higher model accuracy and shorter training delay of federated learning. By leveraging deep reinforcement learning techniques, our method intelligently selects the most suitable participants based on their capabilities and contributions, leading to improved model convergence and overall performance in federated learning scenarios. Simulation results demonstrate that the proposed method enhances the accuracy and training speed of federated learning while facing different datasets and malicious nodes. The contributions of this paper are as follows:

- We develop a multi-objective node selection optimization model that takes into account accuracy, robustness, and latency simultaneously.
- We formulate the multi-objective node selection as a Markov decision process (MDP), defining the state space, action space, and reward function.
- Based on the multi-objective and deep reinforcement learning, we design a DQN-based algorithm to solve the node selection problem.

In Section 2, we provide a comprehensive discussion on the related work in the field, highlighting the existing approaches and their limitations. We then present our proposed system model in Section 3 and our selection method in Section 4. Finally, in Section 5, we present the experimental results and analysis, followed by the conclusion and future work in Section 6.

2. Related Work

The last few years have witnessed rapid developments in distributed machine learning, especially in the area of federated learning. Federated learning has been widely applied in various fields such as Internet of Vehicles [10,11] and healthcare [12,13]. Despite its numerous advantages and successful applications, federated learning still faces several challenges and limitations, such as insufficient training accuracy and lengthy training time, as well as security and privacy concerns. These issues severely hinder the further development of federated learning.

To address these issues, numerous approaches have been proposed, as shown in Table 1. In [14], the authors proposed a novel Federated Learning by Aerial-Assisted Protocol (FLAP) that enables higher accuracy for an image classification model. In [15], the authors design a federated learning model to coordinate x-applications to improve learning efficiency. In [16], the authors present a federated learning security and privacy model enabled by blockchain technology to ensure that it can be used normally for the protection of user data. Among these approaches, optimizing node selection is the most intuitive solution. A well-designed node selection strategy has the potential to improve accuracy, accelerate training speed, and enhance privacy protection, thereby mitigating the limitations of federated learning. Some node selection methods aim to optimize for accuracy. S. Xin et al. [17] propose a node selection algorithm based on reputation (NSRA) to improve accuracy. Li et al. [18] propose a federated learning client selection algorithm based on cluster label information (FedCLS) which realizes efficient federated learning by optimizing the selection of clients in each round of training. Shen et al. [19] propose a simple and effective approach named FedShift which adds the shift on the classifier output during the local training phase to alleviate the negative impact of class imbalance. Their experiments indicate that FedShift significantly outperforms other approaches regarding accuracy. Meanwhile, others prioritize fairness; for instance, Travadi et al. [20] propose a novel incentive mechanism that includes a client selection process to guarantee a fair distribution of rewards. Carey et al. [21] propose Fair Hypernetworks (FHN), a personalized federated learning architecture based on hypernetworks that gives clients the freedom to personalize the fairness metric enforced during local training. Huang et al. [22] propose E3CS, a stochastic client selection scheme, as a solution that balances the joint consideration of effective participation and fairness. In addition, there are also some node selection methods that aim to optimize for training time. Ami et al. [23] present a novel approach for client selection based on the multi-armed bandit (MAB) algorithm, which minimizes training latency without compromising the model's ability to generalize and provide reliable predictions for new observations. Abyan et al. [24] introduce the first availability-aware selection strategy called MDA, which aims to improve training time by taking into account the resources and speed of individual clients. Yin et al. [25] propose a decentralized FL framework by grouping clients with similar computing and communication performance, named federated averaging-inspired group-based federated learning (FGFL).The simulation results verify the effectiveness of FGFL for accelerating the convergence of FL with heterogeneous clients.

Table 1. Comparison of different IoV intrusion detection methods.

Targets	Research Work	Contribution	Limitation
Signature-based	Xin et al. [17]	Using NSRA to select nodes with high reputation prediction value.	Selecting nodes just by the historical reputation.
	Li et al. [18]	Optimizing the selection of clients through label information of each cluster.	Obtaining such cluster labels may not always be feasible or may introduce additional computational overhead.
	Shen et al. [19]	Applying FedShift to alleviate the negative impact of class imbalance and improve the accuracy of the model.	The approach's applicability to other forms of non-IID data needs to be further explored.
Fairness	Travadi et al. [20]	Designing an incentive mechanism to remove low-quality clients and ensure a fair reward distribution.	In practice, the cost may not be easily accessible and the utility can be determined by multiple factors.
	Carey et al. [21]	Utilizing FHN to make clients to personalize the fairness metric enforced during local training freely.	Not all fairness metrics can be formalized.
	Huang et al. [22]	Employing E3CS as selection decision under joint consideration of effective participation and fairness.	Real-time contributions of individual clients are not taken into account in the design of fairness factors.
Training time	Ami et al. [23]	Minimize the training latency without harming the ability of the model to generalize through MAB-based approach.	Lack of consideration for client heterogeneity.
	Abyan et al. [24]	Propose MDA to speed up the learning process.	The generalization capability of the strategy across different FL scenarios should be assessed.
	Yin et al. [25]	Accelerating the convergence of the FL model using FGFL.	When the data size is large, the scalability is not good.
Multi-Objective	Chen et al. [26]	Applying GAN to address the conflicting objectives of training time and energy consumption.	Approach has limited generalization to diverse scenarios.
	Tu et al. [27]	Minimize the AMSE of the aggregation and maximize the long-term energy efficiency of the system via DRL-based framework.	Lack of comprehensive performance evaluation.
	Banerjee et al. [28]	Maximize the processing power, memory, and bandwidth capacity of devices by Fed-MOODS.	Lack of simulations with a large number of devices.
	Hu et al. [29]	FedMGDA+ is proposed to guarantee fairness among users and robustness against malicious attackers.	Absence of privacy guarantees to ensure the privacy of user data.

While the previous studies focus on single-objective node selection, recent research has shifted towards addressing multi-objective node selection. In [26], the authors optimize a multi-objective problem that contains bandwidth and computing resource allocation to obtain a trade-off between the training time and energy consumption. In [27], the authors propose a deep reinforcement learning-based framework to minimize the AMSE of the over-the-air aggregation for different communication rounds and maximize the long-term energy efficiency of the system. In [28], the authors propose Fed-MOODS, a Multi-Objective Optimization-based Device Selection approach that significantly improves the model's convergence and performance. In [29], motivated by ensuring fairness and robustness, the authors formulate federated learning as multi-objective optimization and propose a new algorithm, FedMGDA+.

Nonetheless, the implementation of these algorithms in the absence of complete information presents significant challenges, and their efficacy may not always align with the stringent performance requirements of practical applications.

3. System Model

To improve the model training rate of IoT devices and reduce the communication cost of model aggregation in federated learning (FL), a deep reinforcement learning (DRL)-assisted FL framework is proposed, which leverages existing artificial intelligence (AI) techniques. The DRL algorithm is well-suited for solving high-dimensional decision problems, which makes it ideal for selecting high-quality local IoT device models for aggregation based on their decision-making capabilities [30]. In [31], the authors leverage the Deep Deterministic Policy Gradient (DDPG) to find the optimal solution for node selection in the asynchronous federated learning. In [32,33], the authors propose a node selection algorithm based on Distributed Proximal Policy Optimization (DPPO) to solve the optimization problem. These methods have limitations in exploring non-deterministic policies and ensuring stability in discrete action spaces. In this paper, we propose a DRL-assisted FL algorithm for federated learning devices that balances data privacy and efficiency. We use MNIST and CIFAR-10 datasets to represent the data generated by industrial IoT, leveraging the data features of federated learning devices. Unlike traditional FL distributed training architectures, we improve the model aggregation module by incorporating DRL-based node selection, which enables us to select devices with strong computational power and high training quality for model aggregation before weight aggregation, thereby improving the FL performance. The system architecture we propose is depicted in Figure 1.

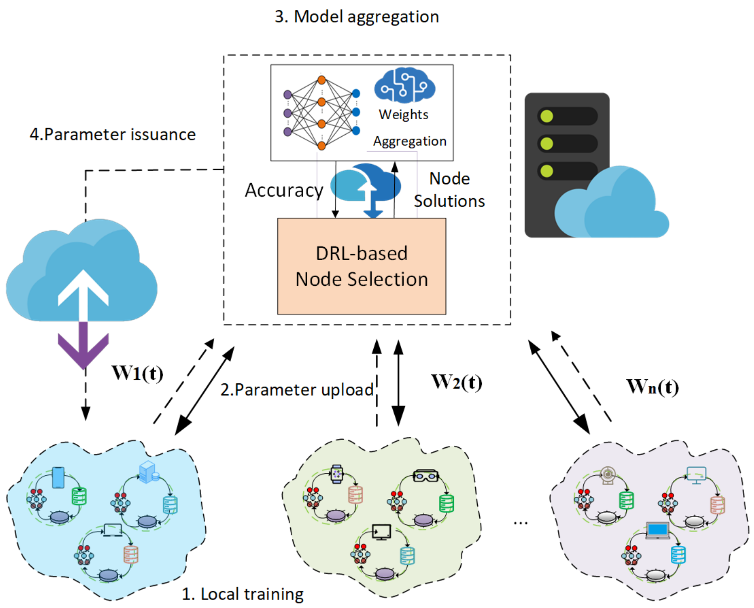

Figure 1. DRL-based FL architecture.

3.1. Network Model

The network consists of end devices and servers. The servers have powerful computing and communication resources and are used for federated learning by connecting to several end devices. The set of terminals is denoted by N, and $H_d = \{x_d, y_d\}$ represents the dataset of terminals d participating in federated learning. For learning tasks $i \in I$ such as path selection and image recognition, the goal is to learn a model M related to the task from the dataset $H_\mathrm{d} = \{x_d, y_d\}$ of terminals. In this paper, the attribute set of FL task i is defined as $Z_i = \{N_i, C_i, M_i^0\}$, where N_i represents the set of terminals related to task i, C_i denotes the number of CPU cycles required to compute the data for FL task i, and M_i^0 denotes the initial model for FL task i.

The specific system parameters are established and presented in Table 2.

Table 2. System parameters.

Parameters	Meaning
H_d	The dataset of terminal d participating in federal learning
R_i	Total dataset related to FL task i
$\|H_d\|$	The size of the dataset for terminal d participating in FL tasks
$\|R_i\|$	The size of the total dataset for this federal learning task
Z_i	Collection of attributes for FL task i
C_i	The number of CPU cycles required for calculating a set of data in dataset
M_i^0	Initial model of FL task i
l_d^i	Loss function of device d when performing local training of FL task i
ω_d^n	The model parameters of device d at the nth training session
η	Learning rate d of device when performing local training of FL task i
A_i	Sum of loss functions for the FL task i test dataset
T_d^local	The local computation time of FL task i
T_d^{trans}	The transmission time of FL task i
f_d	The CPU cycle frequency of device d in executing FL task i
r_d	The transfer rate of task i in FL between the device and the server
G	The number of CPU cycles required per data for device in the FL task i
s_i^t	The environmental state at time t
H_i^{t-1}	The dataset of terminal at the previous time step
a_i^t	The action space at time t
$\beta_{i,d}$	Whether device d is selected to participate in FL task i
T_i	The max delay of clients participating in this iteration
π	A policy is a mapping from a state space to an action space
α^t	Discount factor

3.2. Workflow for Federated Learning Training

The core objective of FL is to enable efficient model training among multiple terminals, while ensuring the security and privacy of data communication. After training the model using the local data, the terminals need to upload the local model to the central server for aggregation aggregating the global model. The steps are as follows:

(1) Participants are selected for this training round. For a federated learning task $i \in I$, the dataset collection within that task is denoted by $\sum\limits_{d \in N} H_d$.

(2) The selected end devices are trained using local data in a local training process, and the locally trained model is obtained by minimizing the loss function of the local training. The loss function $l_d^i(x_d, y_d; \omega_d)$ of end device d during the local training process for federated learning task i is defined as the distinction between its predicted and actual values on sample dataset H_d. Therefore, the loss function of federated learning task i on all datasets can be defined as $L^i(\omega) = \frac{1}{|R_i|} \sum\limits_{d \in N} l_d^i(x_d, y_d; \omega_d)$, where ω denotes the weight of

the current model to be trained and $|H_i|$ denotes the size of the total dataset for the task, also $|R_i| = \sum_{d \in N} |H_d|$.

(3) The devices involved in the training process upload their locally generated models to the server, which aggregates them to produce a global model.

(4) After aggregating the local models, the central server redistributes the resulting global model to the end devices, which then update their own model parameters and initiate the subsequent round of local training. The objective of federated learning (FL) is to optimize the global model parameters, denoted as $\omega = \arg\min L^i(\omega)$, by minimizing the loss function $L^i(\omega)$ associated with the task. In this paper, we adopt the stochastic gradient descent (SGD) method for updating the parameters of the FL model, where one randomly selected data point $\{x_d, y_d\}$ from the dataset is used for each update. This approach significantly reduces the computational effort required for training, but due to the stochastic nature of the method, the local models need to be trained with a sufficient volume of data to ensure model quality. The update of the model parameters is denoted by $\omega_d^n = \omega_d^{n-1} - \eta \nabla l\left(\omega_d^{n-1}\right)$, with η representing the learning rate and $n \in Z$ representing the number of training iterations conducted during the parameter update process.

(5) Repeat the steps described in Steps 2 and 3. Upon completion of sufficient local training by the end devices, the central server aggregates the locally trained models to obtain the global model. This specific weight aggregation process is denoted by $\omega_g' = \omega_g + \sum_{d \in N} \frac{|H_d|\left(\omega_d' - \omega_d\right)}{|R_i|}$, where $|H_d|$ illustrates the size of the dataset of the terminal d participating in the FL task, $|H_i|$ represents the total size of the dataset used in this federated learning task, and the set of end devices is denoted by N.

3.3. Problem Formulation for Node Selection

The selection of nodes is influenced by multiple factors. Firstly, the differential computing and communication capabilities of end devices have a direct impact on local training and data transmission latency. Secondly, the datasets carried on the end devices vary in size, and the data may not satisfy the independent homogeneous distribution property, which can cause variations in the training quality of local models. The purpose of incorporating DRL into FL is to intelligently leverage the collaboration between end devices and servers to exchange learning parameters, thereby improving the training of local models.

Therefore, in this paper, we propose a model that achieves optimal accuracy through node selection.

Accuracy: For an FL task $i \in I$, the training quality is determined by the test accuracy of the aggregated global model on the test dataset. In this paper, the test accuracy is represented by the sum of the loss functions computed on the test dataset.

$$A_i = L^i\left(x_{test}, y_{test}; \omega_g\right) \tag{1}$$

Time delay: Let there be n clients, and after node selection, we obtain client $N' = \{1, 2, \ldots, d\}$ corresponding to FL task $i \in I$. We assume that the time required to broadcast the global model, update the local model, and upload the local model is T_i, denoted by

$$T_i \geq \max\left\{T_d^{\text{local}} + T_d^{\text{trans}}\right\}, \text{where i} \in I, d \in N' \tag{2}$$

We assume that each client has an independent CPU cycle frequency denoted by f_d and a wireless bandwidth denoted by r_d. The selected client and the corresponding training data require the local model to be updated in parallel, and each iteration requires $|H_d|G$ CPU cycles, where G is the number of CPU cycles required per data. Therefore, the required local computation time is denoted by:

$$T_d^{\text{local}} = \frac{|H_d|G}{f_d}, \text{ where } d \in N' \tag{3}$$

The additional time required for the global model update (also known as transmission time) can be denoted as:

$$T_d^{\text{trans}} = \frac{D}{r_d}, \text{ where } d \in N' \tag{4}$$

where D represents the size of the global model.

We assume that the time required for global model transfer and local model training depends on the local model training latency and the local model upload latency, denoted as:

$$T_i \geq \max\left\{\frac{|H_d|G}{f_d} + \frac{D}{r_d}\right\}, \text{ where i} \in I, d \in N' \tag{5}$$

For an FL task $i \in I$, the node selection problem aims to choose a set of nodes $Z_i \in Z$ at each iteration to optimize the training accuracy, i.e., the total loss function is minimized, while ensuring that the training and transmission latency remain within a certain range.

4. FL Node Selection Algorithm Based on DQN

In complex and variable edge networks, node selection policies need to adapt to changes in environmental state information. The DRL-based node selection framework introduced in this paper consists of three components: the environment, the agent, and the reward. It enables the agent to interact with the environment to learn an effective node selection policy that maximizes the reward. Figure 2 illustrates the architecture of our proposed DRL-based node selection framework.

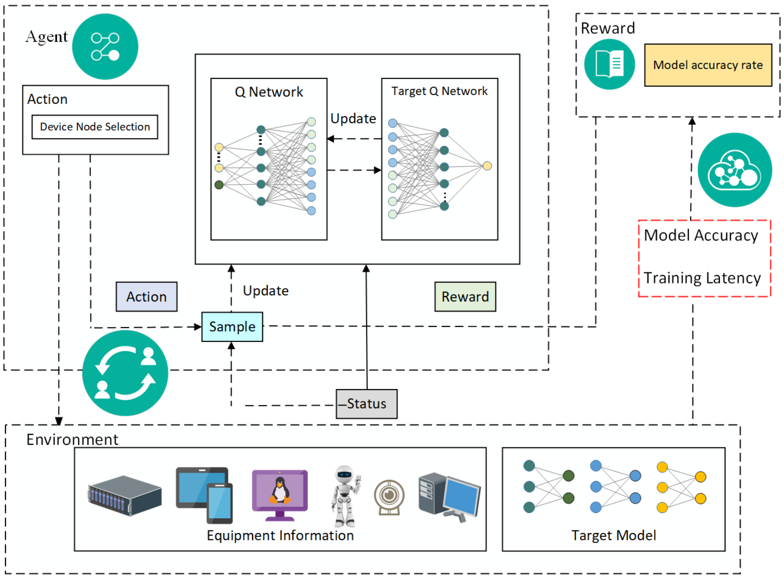

Figure 2. DRL-based node selection framework.

The environment comprises network states, devices, and target model information. The agent interacts with the environment, selects actions based on its policy distribution from a given state, and receives rewards [34,35]. The actions, rewards, and environment states obtained by the agent are used to update the Q-network and the Target Q-network.

We propose a DQN-based approach for client selection in FL. Initially, clients send information about their resources to the server. The server then selects a specific $N^{'} = \{1, 2, \cdots d\}$ to estimate the transmission delay in the next global model training process. The FL client selection problem based on DRL can be formulated as an MDP model [36]. Subsequently, a DQN-based node selection algorithm is designed to solve this problem as follows.

4.1. MDP Model

4.1.1. State Space

The state space will be denoted by a federated learning resource information, defined as:

$$S = \prod_{i=1}^{I} s_i^t \tag{6}$$

where Π represents the Cartesian product, and s_i^t represents the environmental state at time t.

s_i^t can be represented by a quadruple $s_i^t = \left\{ f_i^t, r_i^t, H_i^{t-1}, a_i^{t-1} \right\}$, where f_i^t denotes the independent CPU cycle frequency of each client, r_i^t denotes the wireless bandwidth available to the terminal for the federated learning task i at time t, H_i^{t-1}denotes the dataset of the terminal at the previous time step, and a_i^{t-1} denotes the node selection scheme used in the previous moment.

4.1.2. Action Space

The action space is composed of a set of selection strategies for federated learning tasks i, and it can be denoted by:

$$A = \prod_{i=1}^{I} a_i^t \tag{7}$$

At each action selection step, the agent is allowed to employ only one node selection scheme. If the value is equal to 1, it indicates that the device with the corresponding ID is selected in this node selection. Conversely, if the value is equal to 0, then the device is not selected. The action space can be represented as follows:

$$a_i^t = \{\beta_{i,1}, \beta_{i,2}, \beta_{i,3}, \ldots, \beta_{i,d}\}, \beta_{i,d} \in \{0, 1\} \tag{8}$$

4.1.3. Reward Function

The server selects end devices with sufficient resources, such as CPU cycle frequency, to participate in the model update until the desired accuracy is achieved. The server selects the optimal end devices via the reward function to make the best decision.

When an agent takes a one-step action based on a node selection policy, the environmental information changes and a reward value is obtained to evaluate the action. In this paper, we propose a reward function based on federated learning's test accuracy, while considering a maximum time delay constraint during action selection.

$$R(s, a) = -\alpha_l \frac{T_i}{T_i^{\max}} - \frac{1}{\sum\limits_{d \in N'} \beta_{i,d}} L^i \left(x_{test}, y_{test}; \omega_g^i \right) \tag{9}$$

The maximum latency T_i of the clients participating in the iteration (2).

The execution action source described above is a policy π, which is a mapping from the state space to the action space, resulting in the selection of appropriate actions for different states, i.e.,

$$a_i^t = \pi\left(s_i^t\right) \tag{10}$$

The goal of the MDP model is to obtain an optimal policy that maximizes the expected cumulative reward of reinforcement learning when taking appropriate actions based on the corresponding states, i.e.,

$$\pi^* = \arg\max E\left[\sum_{t=0}^{\infty} \alpha^t r_i^t \right] \tag{11}$$

where α^t is the discount factor, whose value decreases with time.

The discount factor plays a crucial role in deep reinforcement learning by weighting the importance of future rewards. It determines to what extent the agent discounts future rewards when making a decision. The value of the discount factor typically decreases over time, since future rewards are uncertain and risky, and the agent's prediction of future rewards becomes increasingly unreliable over time. Furthermore, more distant future rewards generally have higher uncertainty than more immediate rewards, since they are influenced by more factors. Therefore, decreasing the value of the discount factor can help reduce the effect of distant rewards and focus more on current rewards, making deep reinforcement learning more robust and reliable.

4.2. DQN-Based Algorithm for FL Node Selection

To identify the best course of action, standard Q-Learning is commonly used.

Q-Learning constructs a constantly updated Q-value table with action states, and then looks up the Q-value table to get the best decision for $Q(s_i^t, \alpha_i^t)$ [37]. The edge server updates the Q-values based on the empirical replay as follows:

$$Q'\left(s_i^{t+1}, a_i^{t+1}\right) = (1-\eta)Q\left(s_i^t, a_i^t\right) + \eta\left[R\left(s_i^t, a_i^t\right) + \alpha^t \max_{d_i' \in A} Q\left(s_i^t, a_i^t\right)\right] \tag{12}$$

where η denotes the learning rate and α^t denotes the discount rate.

Following an update to the $Q(s_i^t, \alpha_i^t)$, the server can utilize it to carry out its operations. Using any given state s_i^t, the serve can select the optimal decision π^* by choosing the action with the highest cumulative reward α_i^t [38]. However, the Q-table can be resource-intensive, and the search time for the optimal policy within the table can be prolonged. To address this, we propose the use of the DQN which employs a single neural network (NN) for optimal decision-making, thereby reducing the storage requirements for the Q-value table and accelerating the search process.

Convolutional neural networks (CNNs) serve as a function approximation for Q-Tables in high-dimensional and continuous state. However, in function optimization problems, the conventional approach of supervised learning involves first determining the loss function, followed by finding the gradient and updating the parameters using techniques such as stochastic gradient descent. In contrast, the DQN algorithm is built on Q-Learning to determine the loss function. Here, we define the loss function as follows:

$$L(\theta) = E\left[\left(\text{Target}\, Q - Q\left(s_i^t, a_i^t; \theta\right)\right)^2\right] \tag{13}$$

where θ is the network parameter. The definition of Target Q is :

$$\text{Target}\, Q = R\left(s_i^t, a_i^t\right) + \alpha^t \max_{d_i'} Q\left(s_i^{t'}, a_i^{t'}; \theta\right) \tag{14}$$

The DQN algorithm utilizes the Experience Replay mechanism to store learned data in a cache pool, which is then randomly sampled for subsequent training.

When new nodes join the federated learning, we add a new list to store the new nodes. Each training session preferentially selects a node from this list, and deletes the new node from the new list after its selection. When a new node is added, the new node executes a startup script to add itself to the new list. Node failure in this experiment is defined as failure to return training results after training. After each training session, faulty nodes are judged and processed offline, that is, the nodes are removed from the Q-Table.

Upon conducting an in-depth analysis of the DQN, we present a DQN-inspired federated learning (FL) node selection algorithm. This approach consists of two main phases: multi-threaded interactions and global network updates.

(1) Multi-threaded interactions

Step 1 Each worker thread is assigned a replica of the environment and a local copy of the DQN network. In the context of FL tasks, the environment emulates the behavior and performance of client devices, while the network serves to implement policies within the local environment.

Step 2 Each worker thread independently interacts with a replica of its environment to gather empirical data including states, actions, rewards, and new states. This information is used to train the DQN network. Threads can interact concurrently with their assigned environments, thereby speeding up the data collection process.

Step 3 The experience data collected by individual threads is stored in a shared experience replay buffer. This buffer can be used to randomly sample batch data.

(2) Global Network Updates

Step 1 Upon completing a predetermined number of iterations, the parameters of the global DQN network are synchronized with the local network of each thread, ensuring consistency and updated information across all instances.

Step 2 The global DQN network is trained by sampling a random batch of data from the shared experience replay buffer. The training process is executed concurrently, distributing the computational load across multiple threads for increased efficiency.

Step 3 Every specified number of steps (circle), update the target network with the parameters of the global DQN network to ensure consistency and continued learning progress.

Step 4 Iterate through steps 1 to 3 until the model converges.

After the global network model converges, the agent utilizes the trained model to determine suitable actions according to different environmental states. It then selects an optimal set of nodes to participate in the aggregation process of federated learning. Algorithm 1 provides a detailed outline of this procedure.

The computational complexity of the DQN-based node selection algorithm is primarily determined by the size of the DQN network, the number of episodes and sub-episodes, and the complexity of the reward calculation and Q-value calculation steps.

Algorithm 1: DQN-based node selection algorithm

```
Input: FL Task Information Q network initial state
Output: Node selection scheme
Initialize network, edge device and task information, along with system state and
  experience replay buffers.
for episode $\in \{1, \cdots, EP\}$ do
    for sub_episode $\in \{1, \cdots EP_s\}$ do
        Each agent executes node selection action $\alpha_{i,t}$ according to global DQN
          policy $\alpha_{i,t} = \pi(s_t)$
        Each agent calculates the reward $r_i$ and the next state $S_{t+1}$ according to
          expression (8) and stores the result as a new tuple
        Update current network and device status information
    end
    Each agent uploads the collected data to the global network service in
      synchronization
    Calculate the Q-estimate of the current state and the maximum Q-estimate of
      the next state
    Calculate the target Q value according to expression (14)
    Update network parameters using mean square error (MSE) loss function
end
```

5. Simulation Analysis

5.1. Experimental Settings

In this study, we simulate and validate the algorithm using Python 3.7 and TensorFlow 2.2.0 environments. Our experiments emulate a distributed FL training scenario with various categories of devices. The setup consists of an aggregation server and 10–80 devices.

To demonstrate the robustness of the proposed approach, malicious nodes are introduced in the experiments to simulate devices with subpar training quality. During server aggregation, the parameters of a node can be modified to random values, simulating the presence of a malicious node in local training [39]. We initially select the MNIST dataset for training. The dataset is uniformly partitioned and allocated to the nodes as the local dataset. In addition, the CIFAR dataset and the Fashion Mnist are used in this study to validate the efficacy of the proposed algorithm.

In our simulation, the client device's CPU cycle frequency f_d adheres to a uniform distribution $U[0,1]$ while the wireless bandwidth r follows a uniform distribution $U[0,2]$, $\alpha_i = 2$ [40] This setup allows for a diverse range of computational and communication capabilities among the devices.

A convolutional neural network (CNN) serves as the FL training model, featuring a structure that consists of six convolutional layers, three pooling layers, and one fully connected layer. The DQN algorithm uses four threads to interact with the environment and collect empirical data. The reward discount factor is set to 0.9, and the learning rate of the Q network (value network) is configured at 0.0001. The target network is updated with the parameters of the Q network after every 100 rounds of agent training. In addition, the buffer size for experience replay is set to 10,000. The settings of specific experimental parameters are shown in Table 3.

Table 3. Simulation parameter setting.

Parameter Type	Parameter	Parameter Description	Parameter Value
Equipment and model parameters	E	Number of terminals	100
	f_d	CPU cycle frequency	[0, 1]
	r_d	Wireless Bandwidth	[0, 2]
	H_d	Local datasets	600
	ζ	Local Iteration	2
	N	Minimum sample size	10
	α	Learning Rate	0.01
	Node	Number of nodes involved	[10, 80]
	f_{bt}	Number of CPU cycles required for training per data bit	7000
	$\lvert R_i \rvert$	Global Model Size	20 Mbit
DQN parament	A	Agents	4
	s	Training steps	1000
	Target Q	Q Network	0.0001
	α^t	Bonus Discount Factor	0.9
	circle	Strategy Update Steps	100
	E_r	Experience replay buffer	10,000
	B	Batch-size	64

In this study, we compare the proposed algorithm (FL-DQN) with three alternative approaches:

(1) FL-Random: This algorithm does not utilize deep reinforcement learning for node selection during each iteration of FL training. Instead, it selects nodes at random.

(2) FL-Greedy: The algorithm selects all participating nodes for model aggregation in each iteration of the FL training.

(3) Local Training: This approach does not incorporate any FL mechanism, and the model is solely trained on individual local devices [36].

These comparisons help to assess the relative effectiveness and efficiency of the FL-DQN algorithm.

5.2. Analysis of Results

The experiments evaluate four algorithms, including accuracy, loss function, and time delay. In the experiment, the accuracy of the MNIST dataset is calculated as the ratio of correctly classified samples to the total number of samples, considering it is a classification problem. This metric allows for a comprehensive comparison of the performance of the algorithms.

We divide the experiment into three groups to present the comparison of accuracy, loss function, and delay under different conditions.

Figure 3 presents the accuracy of four algorithms under different conditions of changing the number of iterations, the number of nodes, and the proportion of malicious nodes. Figure 3a illustrates the accuracy variation of the four algorithms when 20% of the nodes are malicious. From Figure 3a, it is evident that the accuracy of the models obtained through the four mechanisms is low during the early stages of training, suggesting that sufficient training iterations are necessary to ensure model accuracy. Upon reaching eight iterations, the accuracy of the models trained by FL-DQN, FL-Random, and FL-Greedy mechanisms tends to stabilize. When the number of iterations reaches 25, the accuracies of FL-DQN, FL-Random, FL-Greedy, and Local Training stabilize around 0.98, 0.96, 0.96, and 0.95, respectively. The FL-DQN algorithm maintains strong training performance when faced with a limited number of malicious nodes and varying data quality.

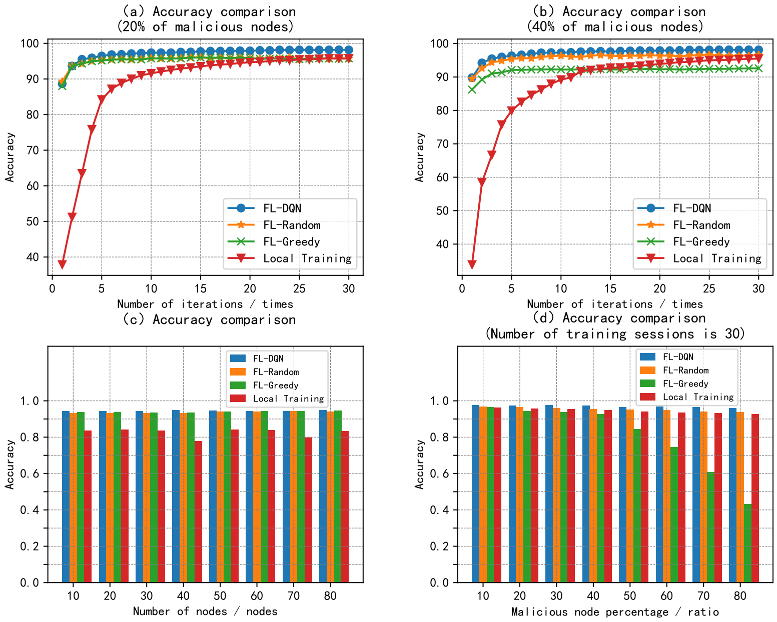

Figure 3. Accuracy experimental group.

Figure 3b displays the accuracy variation of the four algorithms when 40% of the device nodes are malicious. As observed in Figure 3b, FL-DQN rapidly converges to the highest accuracy (0.98) when confronted with a large number of malicious nodes. In contrast, the model quality obtained by FL-Random decreases due to the influence of malicious nodes and stabilizes around 0.95, which is comparable to the training performance of Local Training. For the FL-Greedy algorithm, the accuracy decreases to 0.93. The FL mechanism proposed in this study successfully balances data quality and device training, effectively ensuring optimal model quality.

Figure 3c presents the model accuracy achieved by the four algorithms for different numbers of nodes. The FL-DQN algorithm achieves the highest accuracy when dealing with various node quantities. For example, when 40 nodes are considered, the accuracy of the four algorithms is 0.967, 0.938, 0.932, and 0.754, respectively. The accuracy of FL-DQN algorithm improves by 3.0% and 22.0% compared to FL-DQN and Local Training, respectively. The results also demonstrate that the proposed method exhibits strong scalability in terms of node size, maintaining peak performance as the number of nodes increases.

Figure 3d presents the accuracy of the models obtained by the four algorithms for a fixed number of training rounds (30 rounds) and different percentages of malicious nodes (ranging from 10% to 80%). We observe that FL-DQN can efficiently filter out high-quality nodes for model aggregation when dealing with different proportions of malicious nodes, in contrast to FL-Random, FL-Greedy and Local Training. This filtering process ensures the quality of the overall model, leading to the highest accuracy and smallest loss function values in FL-DQN. As a result, it can be concluded that the proposed method exhibits excellent robustness.

Figure 4 presents the loss functions of four algorithms under different conditions of changing the number of iterations, the number of nodes, and the proportion of malicious nodes. Figure 4a presents the variation of the loss functions for the four algorithms when 20% of the nodes are malicious. The FL-DQN algorithm converges faster than the remaining four algorithms and exhibits the lowest value for the loss function. This also highlights the advantages of the FL-DQN approach in terms of convergence and loss reduction.

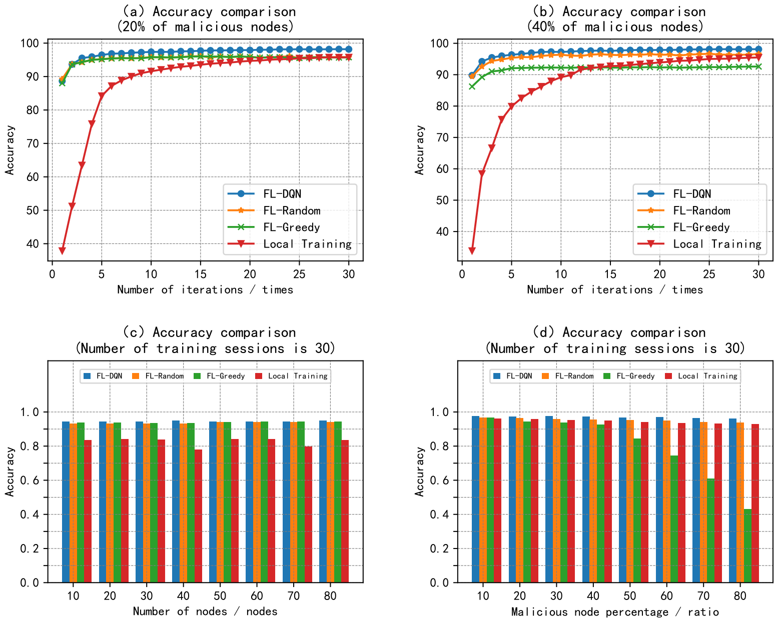

Figure 4. Loss function experimental group.

Figure 4b shows the variation of the loss functions of the four algorithms when 40% of the malicious nodes are present. Similar to the convergence in accuracy, the FL-DQN algorithm converges faster and has the smallest loss function value than the remaining three algorithms, while FL-Random, FL-Greedy, and Local Training always have higher loss function values due to the presence of malicious nodes. Comparing the above four sets of simulation results, it can be seen that FL-DQN always converges quickly to the highest accuracy with different numbers of malicious nodes and has the lowest loss function compared to FL-Random, FL-Greedy and Local Training. At the same time, for FL-DQN algorithm, the accuracy rate of 0.98 is maintained for both 20% and 40% of malicious nodes. Therefore, it can be concluded that the proposed method in this paper is remarkably robust.

Figure 4c presents the loss function achieved by the four algorithms for different numbers of nodes. The FL-DQN algorithm achieves the highest accuracy when dealing with multiple numbers of nodes. The difference between FL-Random and FL-Greedy loss functions is not significant. Figure 4d shows the loss function values of the four algorithms obtained at the end of training under the same conditions. Compared to FL-Random, FL-Greedy, and Local Training, FL-DQN can handle different proportions of malicious nodes to guarantee the quality of the whole model and thus obtain the minimum of the loss function.

Figure 5 presents the latency of four algorithms under different conditions of changing the number of iterations, the number of nodes, and the proportion of malicious nodes. Figure 5a shows the changes in latency of four algorithms with 20% malicious nodes. Figure 5b shows the changes in latency of four algorithms with 40%malicious nodes. Compared to the remaining three algorithms, the FL-DQN algorithm is able to complete the training task faster and has the smallest variation in the time used per round.

As shown in Figure 5c, the FL-DQN algorithm can guarantee low latency in dealing with various numbers of nodes, as it can effectively select high-quality training devices for model aggregation. Taking the number of nodes as 40, the latency values for the four algorithms are 11.3 s, 13.8 s, 17.4 s, and 16.4 s, respectively. The FL-DQN algorithm reduces the latency by 18%, 35%, and 31% compared to FL-Random, FL-Greedy, and Local Training, respectively. These results indicate that the proposed algorithm can efficiently complete FL training.

Figure 5d presents the latency changes of four algorithms with fixed training rounds (30 rounds) and different percentages of malicious nodes (from 10% to 80%). From Figure 5d, it can be observed that even when facing a large number of malicious nodes, the FL-DQN algorithm can still complete the training task relatively quickly.

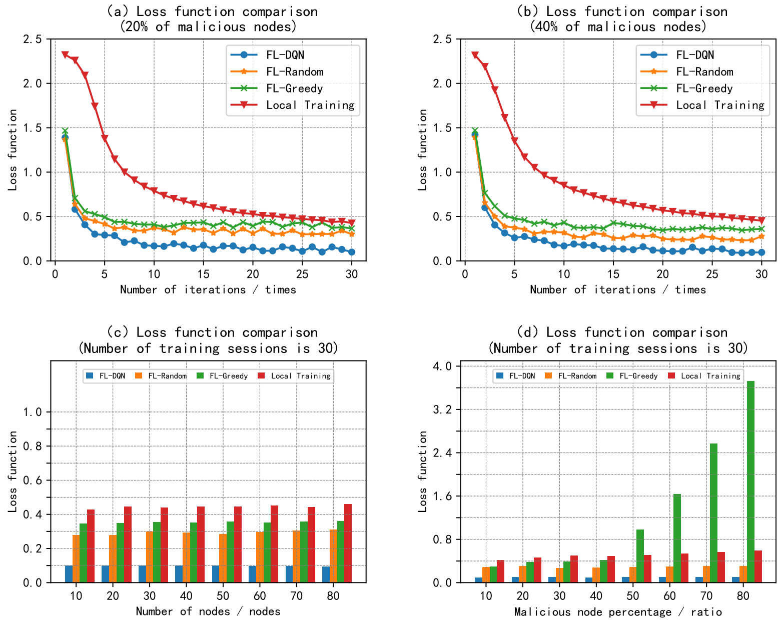

Figure 5. Latency experimental group.

The following section compares and validates four algorithms using the CIFAR dataset. Figure 6 illustrates the accuracy variations of the four algorithms with 20% malicious device nodes. (CIFAR) The CIFAR dataset requires significantly more training iterations than the MNIST dataset. When the number of iterations reaches 60, the accuracy of the models trained by the four algorithms stabilizes. The accuracies of FL-DQN, FL-Random, FL-Greedy, and Local Training stabilize at 0.80, 0.71, 0.68, and 0.58, respectively. The FL-DQN algorithm demonstrated good training performance in handling malicious nodes and differential data quality.

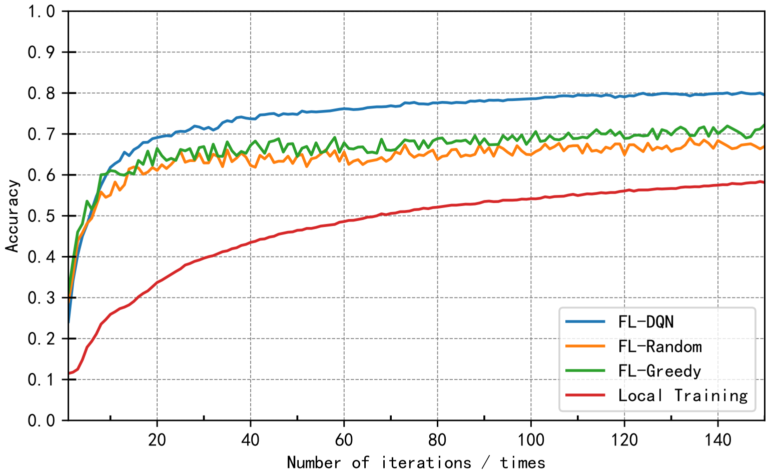

Figure 6. Accuracy comparison of CIFAR dataset.

Figure 7 illustrates the accuracy variations of the four algorithms with 20% malicious device nodes. (Fashion MNIST) The FL-DQN algorithm continued to demonstrate excellent training performance with higher accuracy compared to other algorithms.

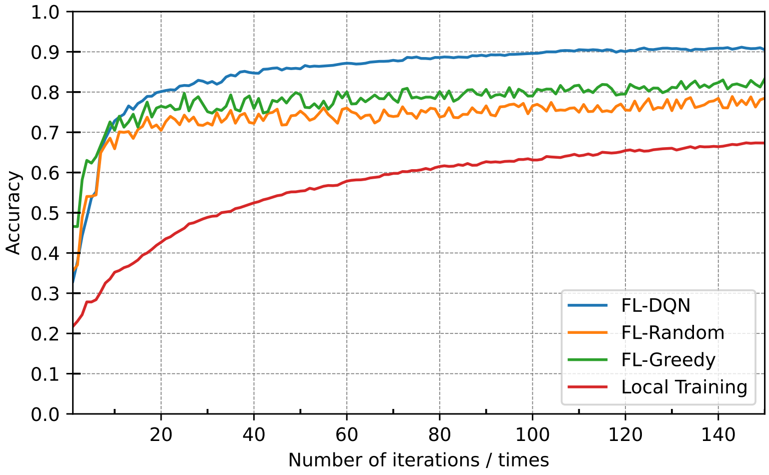

Figure 7. Accuracy comparison of Fashion MNIST dataset.

Figure 8 displays the loss function variations of the four algorithms with 20% malicious device nodes. (CIFAR) Figure 9 presents the loss functions of four algorithms with 20% malicious device nodes. (Fashion MNIST) The FL-DQN algorithm achieves faster convergence and has the smallest loss function value. These results demonstrate that the FL-DQN algorithm outperforms the FL-DQN and Local Training algorithms in terms of loss function convergence.

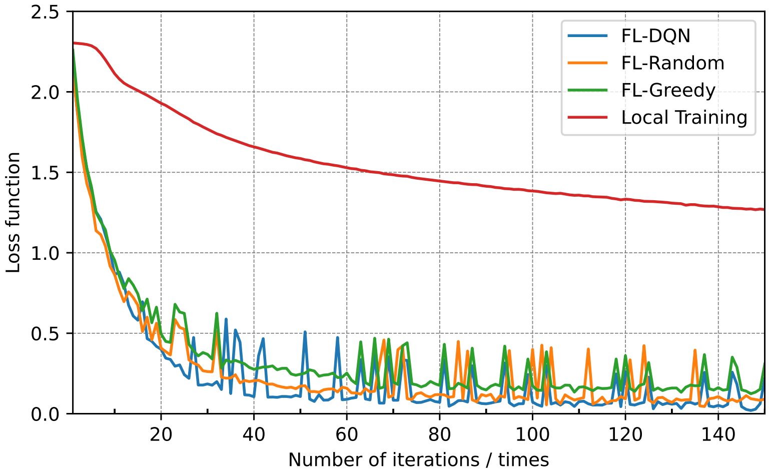

Figure 8. Comparison of loss functions for CIFAR dataset.

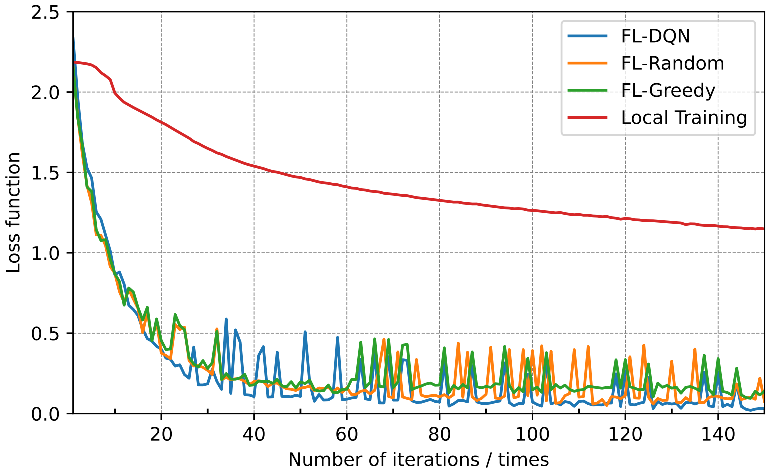

Figure 9. Comparison of loss functions for Fashion MNIST dataset.

6. Conclusions

In this paper, we propose a novel multi-objective node selection approach that leverages deep reinforcement learning. Our proposed approach can efficiently solve MOP and provides an efficient solution for FL node selection. First, we construct a model that fully takes into account device training delay, model transmission delay, and accuracy rate to optimize node selection. Then, we formulate the problem as an MDP model and design a node selection algorithm based on the DQN algorithm to select a reasonable set of devices for model aggregation before each training iteration. Finally, the simulation results demonstrate that the proposed approach significantly improves the accuracy and training speed of federated learning while maintaining good resistance to malicious nodes. However, it is important to note that further research is needed to address the limitations related to optimizing the aggregation process while considering communication overhead. In future work, we plan to explore the application of our algorithm in real-world scenarios, where factors such as network conditions, device heterogeneity, and varying data distributions may pose additional challenges. In addition, we aim to enhance our node selection strategy by considering factors such as node reputation and reliability. By incorporating these additional factors, we can further optimize the node selection process and improve the overall performance of federated learning.

Author Contributions: Software, T.X. and Z.M.; Validation, Y.L. and P.L.; Investigation, T.X.; Resources, Y.H.; Data curation, Y.L.; Writing—original draft, T.X. and Y.H.; Writing—review and editing, Z.M.; Supervision, P.L. All authors have read and agreed to the published version of the manuscript.

Funding: This research was funded by the Natural Science Foundation of Heilongjiang Province of China (No. LH2022F003), and the Key Research and Development Program Heilongjiang Province of China (No. 2022ZX01A35).

Data Availability Statement: The MNIST dataset used in this study is publicly available and can be accessed from the official website of the Modified National Institute of Standards and Technology (MNIST) at http://yann.lecun.com/exdb/mnist/. The CIFAR-10 dataset used in this study is publicly available and can be accessed from the official website of the Canadian Institute for Advanced Research (CIFAR) at https://www.cs.toronto.edu/~kriz/cifar.html. The Fashion MNIST dataset used in this study is publicly available and can be accessed from the official website of the Zalando Research at https://github.com/zalandoresearch/fashion-mnist.

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

Abbreviations

The following abbreviations are used in this manuscript:

MDP	Markov decision process
DQN	Deep Q-Network
DRL	Deep Reinforcement Learning
SGD	Stochastic Gradient Descent
CNN	Convolutional Neural Network
MSE	Mean Square Error

References

1. Yang, Q.; Liu, Y.; Chen, T.; Tong, Y. Federated machine learning: Concept and applications. *ACM Trans. Intell. Syst. Technol. (TIST)* **2019**, *10*, 1–19. [CrossRef]
2. Kairouz, P.; McMahan, H.B.; Avent, B.; Bellet, A.; Bennis, M.; Bhagoji, A.N.; Bonawitz, K.; Charles, Z.; Cormode, G.; Cummings, R.; et al. Advances and open problems in federated learning. *Found. Trends Mach. Learn.* **2021**, *14*, 1–210. [CrossRef]
3. Banabilah, S.; Aloqaily, M.; Alsayed, E.; Malik, N.; Jararweh, Y. Federated learning review: Fundamentals, enabling technologies, and future applications. *Inf. Process. Manag.* **2022**, *59*, 103061. [CrossRef]
4. Mora, A.; Fantini, D.; Bellavista, P. Federated Learning Algorithms with Heterogeneous Data Distributions: An Empirical Evaluation. In Proceedings of the 2022 IEEE/ACM 7th Symposium on Edge Computing (SEC), Seattle, WA, USA, 5–8 December 2022; pp. 336–341. [CrossRef]
5. Chathoth, A.K.; Necciai, C.P.; Jagannatha, A.; Lee, S. Differentially Private Federated Continual Learning with Heterogeneous Cohort Privacy. In Proceedings of the 2022 IEEE International Conference on Big Data (Big Data), Osaka, Japan, 17–20 December 2022; pp. 5682–5691. [CrossRef]
6. Han, J.; Khan, A.F.; Zawad, S.; Anwar, A.; Angel, N.B.; Zhou, Y.; Yan, F.; Butt, A.R. Heterogeneity-Aware Adaptive Federated Learning Scheduling. In Proceedings of the 2022 IEEE International Conference on Big Data (Big Data), Osaka, Japan, 17–20 December 2022; pp. 911–920. [CrossRef]
7. Wu, H.; Wang, P. Node selection toward faster convergence for federated learning on non-iid data. *IEEE Trans. Netw. Sci. Eng.* **2022**, *9*, 3099–3111. [CrossRef]
8. Deer, A.; Ali, R.E.; Avestimehr, A.S. On Multi-Round Privacy in Federated Learning. In Proceedings of the 2022 56th Asilomar Conference on Signals, Systems, and Computers, Pacific Grove, CA, USA, 31 October–2 November 2022; pp. 764–769. [CrossRef]
9. Liu, W.; Chen, L.; Zhang, W. Decentralized federated learning: Balancing communication and computing costs. *IEEE Trans. Signal Inf. Process. Over Netw.* **2022**, *8*, 131–143. [CrossRef]
10. Issa, W.; Moustafa, N.; Turnbull, B.; Sohrabi, N.; Tari, Z. Blockchain-based federated learning for securing internet of things: A comprehensive survey. *ACM Comput. Surv.* **2023**, *55*, 1–43. [CrossRef]
11. Chi, J.; Xu, S.; Guo, S.; Yu, P.; Qiu, X. Federated Learning Empowered Edge Collaborative Content Caching Mechanism for Internet of Vehicles. In Proceedings of the NOMS 2022–2022 IEEE/IFIP Network Operations and Management Symposium, Budapest, Hungary, 25–29 April 2022; pp. 1–5. [CrossRef]
12. Patel, V.A.; Bhattacharya, P.; Tanwar, S.; Gupta, R.; Sharma, G.; Sharma, P.N.; Sharma, R. Adoption of federated learning for healthcare informatics: Emerging applications and future directions. *IEEE Access* **2022**, *10*, 90792–90826.. [CrossRef]
13. Moon, S.H.; Lee, W.H. Privacy-Preserving Federated Learning in Healthcare. In Proceedings of the 2023 International Conference on Electronics, Information, and Communication (ICEIC), Singapore, 5–8 February 2023; pp. 1–4. [CrossRef]

14. Vrind, T.; Pathak, L.; Das, D. Novel Federated Learning by Aerial-Assisted Protocol for Efficiency Enhancement in Beyond 5G Network. In Proceedings of the 2023 IEEE 20th Consumer Communications & Networking Conference (CCNC), Las Vegas, NV, USA, 8–11 January 2023; pp. 891–892. [CrossRef]
15. Zhang, H.; Zhou, H.; Erol-Kantarci, M. Federated deep reinforcement learning for resource allocation in O-RAN slicing. In Proceedings of the GLOBECOM 2022-2022 IEEE Global Communications Conference, Rio de Janeiro, Brazil, 4–8 December 2022; pp. 958–963. [CrossRef]
16. Guo, X. Implementation of a Blockchain-enabled Federated Learning Model that Supports Security and Privacy Comparisons. In Proceedings of the2022 IEEE 5th International Conference on Information Systems and Computer Aided Education (ICISCAE), Dalian, China, 23–25 September 2022; pp. 243–247. [CrossRef]
17. Xin, S.; Zhuo, L.; Xin, C. Node Selection Strategy Design Based on Reputation Mechanism for Hierarchical Federated Learning. In Proceedings of the 2022 18th International Conference on Mobility, Sensing and Networking (MSN), Guangzhou, China, 14–16 December 2022; pp. 718–722. [CrossRef]
18. Li, C.; Wu, H. FedCLS: A federated learning client selection algorithm based on cluster label information. In Proceedings of the 2022 IEEE 96th Vehicular Technology Conference (VTC2022-Fall), London, UK, 26–29 September 2022; pp. 1–5. [CrossRef]
19. Shen, Y.; Wang, H.; Lv, H. Federated Learning with Classifier Shift for Class Imbalance. *arXiv* **2023**, arXiv:2304.04972.
20. Travadi, Y.; Peng, L.; Bi, X.; Sun, J.; Yang, M. Welfare and Fairness Dynamics in Federated Learning: A Client Selection Perspective. *arXiv* **2023**, arXiv:2302.08976.
21. Carey, A.N.; Du, W.; Wu, X. Robust Personalized Federated Learning under Demographic Fairness Heterogeneity. In Proceedings of the 2022 IEEE International Conference on Big Data (Big Data), Osaka, Japan, 17–20 December 2022; pp. 1425–1434. [CrossRef]
22. Huang, T.; Lin, W.; Shen, L.; Li, K.; Zomaya, A.Y. Stochastic client selection for federated learning with volatile clients. *IEEE Internet Things J.* **2022**, *9*, 20055–20070. [CrossRef]
23. Ami, D.B.; Cohen, K.; Zhao, Q. Client Selection for Generalization in Accelerated Federated Learning: A Bandit Approach. In Proceedings of the ICASSP 2023-2023 IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP), Rhodes Island, Greece, 4–10 June 2023; pp. 1–5. [CrossRef]
24. Eslami, Abyane, A.; Drew, S.; Hemmati, H. MDA: Availability-Aware Federated Learning Client Selection. *arXiv* **2022**, arXiv:2211.14391.
25. Yin, T.; Li, L.; Lin, W.; Ma, D.; Han, Z. Grouped Federated Learning: A Decentralized Learning Framework with Low Latency for Heterogeneous Devices. In Proceedings of the 2022 IEEE International Conference on Communications Workshops (ICC Workshops), Seoul, Republic of Korea, 16–20 May 2022; pp. 55–60. [CrossRef]
26. Yin, B.; Chen, Z.; Tao, M. Predictive GAN-powered Multi-Objective Optimization for Hybrid Federated Split Learning. *arXiv* **2022**, arXiv:2209.02428.
27. Tu, X.; Zhu, K. Learning-based Multi-Objective Resource Allocation for Over-the-Air Federated Learning. In Proceedings of the GLOBECOM 2022-2022 IEEE Global Communications Conference, Rio de Janeiro, Brazil, 4–8 December 2022; pp. 3065–3070. [CrossRef]
28. Banerjee, S.; Vu, X.S.; Bhuyan, M. Optimized and Adaptive Federated Learning for Straggler-Resilient Device Selection. In Proceedings of the 2022 International Joint Conference on Neural Networks (IJCNN), Padua, Italy, 18–23 July 2022; pp. 1–9. [CrossRef]
29. Hu, Z.; Shaloudegi, K.; Zhang, G.; Yu, Y. Federated learning meets multi-objective optimization. *IEEE Trans. Netw. Sci. Eng.* **2022**, *9*, 2039–2051. [CrossRef]
30. Jarwan, A.; Ibnkahla, M. Edge-Based Federated Deep Reinforcement Learning for IoT Traffic Management. *IEEE Internet Things J.* **2022**, *10*, 3799–3813. . [CrossRef]
31. Lu, Y.; Huang, X.; Zhang, K.; Maharjan, S.; Zhang, Y. Blockchain empowered asynchronous federated learning for secure data sharing in internet of vehicles. *IEEE Trans. Veh. Technol.* **2020**, *69*, 4298–4311. [CrossRef]
32. Wang, R.; Tsai, W.T. Asynchronous federated learning system based on permissioned blockchains. *Sensors* **2022**, *22*, 1672. [CrossRef]
33. Shen, Y.; Gou, F.; Wu, J. Node screening method based on federated learning with IoT in opportunistic social networks. *Mathematics* **2022**, *10*, 1669. [CrossRef]
34. Neves, M.; Neto, P. Deep reinforcement learning applied to an assembly sequence planning problem with user preferences. *Int. J. Adv. Manuf. Technol.* **2022**, *122*, 4235–4245. [CrossRef]
35. Li, X.; Fang, J.; Du, K.; Mei, K.; Xue, J. UAV Obstacle Avoidance by Human-in-the-Loop Reinforcement in Arbitrary 3D Environment. *arXiv* **2023**, arXiv:2304.05959.
36. He, W.; Guo, S.; Qiu, X.; Chen, L.; Zhang, S. Node selection method in federated learning based on deep reinforcement learning. *J. Commun.* **2021**, *42*, 62–71. [CrossRef]
37. Xuan, Z.; Wei, G.; Ni, Z. Power Allocation in Multi-Agent Networks via Dueling DQN Approach. In Proceedings of the 2021 IEEE 6th International Conference on Signal and Image Processing (ICSIP), Nanjing, China, 22–24 October 2021. [CrossRef]
38. Lin, J.; Moothedath, S. Federated Stochastic Bandit Learning with Unobserved Context. *arXiv* **2023**, arXiv:2303.17043.

39. Kim, H.; Doh, I. Privacy Enhanced Federated Learning Utilizing Differential Privacy and Interplanetary File System. In Proceedings of the 2023 International Conference on Information Networking (ICOIN), Bangkok, Thailand, 11–14 January 2023; pp. 312–317. [CrossRef]
40. Zhang, H.; Xie, Z.; Zarei, R.; Wu, T.; Chen, K. Adaptive client selection in resource constrained federated learning systems: A deep reinforcement learning approach. *IEEE Access* **2021**, *9*, 98423–98432. [CrossRef]

Disclaimer/Publisher's Note: The statements, opinions and data contained in all publications are solely those of the individual author(s) and contributor(s) and not of MDPI and/or the editor(s). MDPI and/or the editor(s) disclaim responsibility for any injury to people or property resulting from any ideas, methods, instructions or products referred to in the content.