IEEE Access
Multidisciplinary | Rapid Review | Open Access Journal

Received May 28, 2021, accepted July 6, 2021, date of publication July 9, 2021, date of current version July 16, 2021.

Digital Object Identifier 10.1109/ACCESS.2021.3095915

Adaptive Client Selection in Resource Constrained Federated Learning Systems: A Deep Reinforcement Learning Approach

HANGJIA ZHANG[1], ZHIJUN XIE[1], ROOZBEH ZAREI[2], TAO WU[1], AND KEWEI CHEN[3]

[1]Faculty of Electrical Engineering and Computer Science, Ningbo University, Ningbo 315000, China
[2]School of Information Technology, Deakin University, Melbourne, VIC 3217, Australia
[3]Faculty of Mechanical Engineering and Mechanics, Ningbo University, Ningbo 315000, China

Corresponding author: Zhijun Xie (18057423757@163.com)

This work was supported in part by the Nature Science Foundation of China under Grant U20A20121; in part by the Ningbo International Science and Technology Cooperation Program under Grant 2016D10008; in part by the Ningbo Key Science and Technology Plan (2025) Project under Grant 2019B10125, Grant 2019B10028, and Grant 20201ZDYF020077; in part by the Special Research funding from the Marine Biotechnology and Marine Engineering Discipline Group, Ningbo University, under Grant 422004582; in part by the Project of Research and Development of Intelligent Resource Allocation and Sharing Platform for Marine Electronic Information Industry under Grant 2017GY116; and in part by the Key Science and Technology Projects of Zhejiang Province under Grant 2020C03064.

ABSTRACT With data increasingly collected by end devices and the number of devices is growing rapidly in which data source mainly located outside the cloud today. To guarantee data privacy and remain data on client devices, federated learning (FL) has been proposed. In FL, end devices train a local model with their data and send the model parameters rather than raw data to server for aggregating a new global model. However, due to the limited wireless bandwidth and energy of mobile devices, it is not practical for FL to perform model updating and aggregation on all participating devices in parallel. And it is difficulty for FL server to select apposite clients to take part in model training which is important to save energy and reduce latency. In this paper, we establish a novel mobile edge computing (MEC) system for FL and propose an experience-driven control algorithm that adaptively chooses client devices to participate in each round of FL. Adaptive client selection mechanism in MEC can be modeled as a Markov Decision Process in which we do not need any prior knowledge of the environment. We then propose a client selection scheme based on reinforcement learning that learns to select a subset of devices in each communication round to minimize energy consumption and training delay that encourages the increase number of client devices to participate in model updating. The experimental results show that the unit of energy required in FL can be reduced by up to 50% and training delay required can be reduced by up to 20.70% compared to the other static algorithms. Finally, we demonstrate the scalability of MEC system with different tasks and the influence of different non independent and identically distributed (non-IID) settings.

INDEX TERMS Client selection, federated learning, mobile edge computing, reinforcement learning.

I. INTRODUCTION

Currently, artificial intelligence has become an essential part of our lives today, following the recent successes of machine learning (ML) in several domains, e.g., image recognition and natural language processing [1], [2]. ML model requires a wealth of data to update the model to achieve the desirable accuracy [3]. In traditional training of ML model, a cloud-centric based approach is adopted whereby data collected by mobile devices is centralized and ML model training occurs in a powerful cloud server or data center [4]. However, this approach is no longer sustainable in three main aspects: privacy leaking that data owners are increasingly privacy sensitive, long propagation delay which incurs unacceptable delay for real-time decisions, and backbone network burden which the transfer of data to the center occupancy quite some time and bandwidth [5]. These are intensified by the fact that cloud-centric training approach is relatively reliant on wireless communications.

With data increasingly collected by end devices and the number of end devices is growing rapidly, which means data source mainly located outside the cloud today. Hence,

The associate editor coordinating the review of this manuscript and approving it for publication was David Flynn.

Mobile Edge Computing has naturally been proposed as a solution to train distributed ML model which brings ML training model closed to data source [6]. In conventional MEC ML training collaborative paradigm, cloud server first send training data an ML model to edge servers for training a low-level model, before more computation intensive ML model training process is offloaded to the cloud server. However, this collaborative paradigm incurs significant communication costs and computation offloading, which involves the transmission of potentially sensitive personal data. This will prevent privacy sensitive clients to participate in ML model training processing, or even violate personal privacy laws.

To guarantee that data remains in clients own devices and to facilitate the complex ML model training among distributed devices, [7] proposed a promising collaborative decentralized ML paradigm called federated learning. In FL, FL server first send ML model to clients who will participate in model training this round. Clients use their own privacy data to train a personal ML model. They then send upload the model updates, i.e., the ML model's weights, to the FL server. FL server aggregates all models' weights uploaded by clients through synchronization aggregation algorithm [8]. These steps are continuously repeated multiple rounds until ML model achieves the desirable accuracy. As compared to conventional cloud-centric ML model training approach, the implementation of FL in MEC for ML model training features highly efficient use of clients' bandwidth which will ease pressure on backbone network, client privacy and low delay.

Given the aforementioned advantages, FL has being increasingly an enabling technology for ML model training at MEC system optimization. Traditional optimization approaches that are built on static models fare relatively poorly in modelling dynamic MEC network, which devices are constrained by computation, storage and energy. In FL global model training progress, FL server selects clients to take part in ML model updating in this training round and offloads the newest global model when they complete training task. In general, FL server randomly selects a set of clients to participate incurs high ML model training time which limited by the slowest client, i.e., stragglers. However, recent studies have paid more attention to purely FL or MEC system, rather than effectively combining the two fields for research. At the same time, for FL in MEC, the independent resource status of each client needs to be considered. The existing methods are more static algorithms, and intelligent algorithms are rarely used for this field. In this paper, we propose a data-driven Deep Reinforcement Learning (DRL) based approach for optimizing resource allocation and selecting clients through amending decision making in dynamic MEC environment.

FL server first observes all clients' resource information such as wireless channel states, computation capacities, and real-time energy states. According to clients' resource information, FL server estimates the time and energy required for broadcasting ML model, training local ML model, uploading local model and aggregating model received. FL server repeats these steps multiple rounds until ML model achieves a certain desired performance. However, clients are constrained by personal energy and computation that may reduce efficiency of ML training tasks and clients' bandwidth limitation that increases communication cost and model training latency. To solve the two limitations, FL server should select eligible clients that equipped with energy unit, storage and CPU to participate in ML model training aims to minimize the energy consumption and training latency. However, it is challenging for FL server to determine optimal decisions since the MEC environment is stochastic in which clients' resource information are uncertain. In this paper, we thus propose to use the Deep Q-Learning (DQL) technique that enables FL server to find the optimal client selection policy in MEC system through FL without any priori knowledge of network dynamics [9]. We first formulate the dynamic MEC system as a Markov Decision Process (MDP). We then adopt the DQL based on the Double Deep Q-Network (DDQN) to make the optimal decision that selecting adaptive clients to train ML model for FL server [10]. Simulation results show that our proposed DDQN algorithm outperforms the static algorithms in terms of energy units consumption and delay.

We conclude our contributions as follows.

1) We establish a MEC model for FL which contains FL server and client devices with energy, CPU, data storage, bandwidth and wireless charging.
2) We model the MEC system as MDP which need not any prior knowledge. Then we use DDQN to adaptively select clients to take part in global model updating to minimize resource consumption such as energy and bandwidth.
3) We demonstrate our proposed algorithm in MEC system on MNIST and Fashion-MNIST datasets in which greatly reduce the energy consumption and the model updating delay.
4) We explore the influence of different edge server tasks and non-IID settings on our MEC system.

II. RELATED WORK

In this section, we introduce the related work about FL and recent work for the implementation in MEC system.

A. FEDERATED LEARNING

FL is an emerging decentralized ML technique in which a number of clients use their privacy data to train a shared model that reminds their own data locally. Reference [7] proposed the concept of FL and Federated Averaging algorithm (FedAvg) which could achieve desirable accuracy even when data is non-IID across clients. Reference [11] invites specified clients to train ML model leveraging on implicit association between raw data distribution and local model's weights. Reference [12] proposed MOCHA algorithm in which an alternating optimization approach to solve the minimization problem of resource constraints of clients. But it

cannot effectively apply non-concave deep learning models. Reference [13] proposed Loss-based AdaBoost Federated ML (LoAdaBoost) algorithm in which client should retrain local ML model before aggregation if the current cross-entropy loss is higher than the median loss from the previous training round so as to increase training efficiency. However, the method does not consider the resource consumption of participating clients, which may occupy client device resources for a long time. Reference [14] proposed structured and sketched updates to reduce communication costs through reducing the size of ML model weights. Reference [15] proposed q-Fair FL (q-FFL) algorithm in which assigns higher weights in the loss function to clients with higher loss so as to lower the variance of performance of FL model across clients. But, the algorithm did not consider the actual resource situation of the client. However, the exiting studies usually treat FL and MEC separately.

B. FEDERATED LEARNING IN MOBILE EDGE COMPUTING

In cloud computing, recent research addressed the problems of massive amount of energy consumption and service level agreements violation with concentrate on CPU utilization [16], [17]. References [18] and [19] proposed energy-aware host overload detection algorithm and virtual machines selection algorithms to reduce the energy consumption of mobile cloud data centers. While in MEC, recent research pays more attention to client resources with intelligent algorithms. Reference [20] and [21] model the computation offloading task as a MDP and maximize the long-term utility performance with DQL to make optimal decisions. Reference [22] studies the computation offloading for clients with energy harvesting. However, most existing surveys on MEC do not consider FL as a potential solution to preserve client privacy in computation offloading.

In addition, the implementation of FL for MEC system's resource optimization mostly do not focus on client selection policy. Reference [23] proposed a resource allocation approach based on DRL with clients are mobile and uncertainty to maximize the success rate of broadcasting global model. However, it did not consider the convergence problem of the global model. Reference [9] proposed a new Federated Learning with Client Selection (FedCS) protocol, which selects clients with higher computing capabilities to participate. However, the agreement cannot be a good choice for suitable clients when the system is large, that is, when there are many participants. Reference [24] modelled the interaction between clients and FL server as Stacklberg game in which each client could non-cooperatively decide on its own profit maximization price. However, the simulation environment of the game only involves a small number of devices, and its effectiveness has not been proven for a large number of devices. Reference [25] proposed an incentive mechanism in which FL server set price for client to join at will.

In summary, most exiting surveys rarely consider MEC system resource optimization and non-IID data of clients together, and little surveys concentrate on the challenges of client selection to reduce resource consumption and speed up ML model convergence.

III. SYSTEM MODEL AND ASSUMPTION

In this section, we briefly introduce the general workflow of FL and assumption about MEC system.

A. FEDERATED LEARNING AND ASSUMPTION

FL allows clients to collaboratively train a global model while keeping personal data on their own devices to protect privacy. Therefore, FL can serve as a enabled technology for ML training. ML models' success based on large of training data. A training data sample j consists of two parts: a feature vector x_j as the input of the ML model and a label y_j that is regarded as the output of the ML model. We train a two-layer CNN model with client data determined by σ which means σ percent of client data comes from a random dominant class and the remaining belong to other labels.

FL system contains two main entities, i.e., clients that is regarded as the data owner and the model owner that is desired FL server. Assume that there is a set $N = \{1, 2, \ldots, n\}$ of clients in MEC system. Each round FL server decides the ML model training task, i.e., the number of local training iterations ε and the corresponding training data requirements μ. After FL server broadcasts the initialized global model w_0^0 to the selected m clients, where w_G^0 denotes the global model parameters in first iteration m denotes a subset of N. And in i communication round, client k respectively uses its own local data and device resource to train update global model w_k^{i}, where w_k^{i} is local model parameters of client k in communication round i. The goal of client k is to find optimal local model parameters w_k^i that minimize the loss function $l_k(w)$. The updated local model parameters are subsequently sent to FL server. FL server aggregates the local models from m clients and sends the updated global model parameters $w_G^{\mathrm{i}+1}$ to the selected m clients. The aggregation function in FL server is defined as

$$w_G^{i+1} = \frac{1}{\sum_{k=1}^{m} |\mu_k|} \sum_{k=}^{m} |\mu_k| w_k^i \tag{1}$$

where $|\mu_k|$ denotes the data size of clients k and i denotes the current iteration index [7]. According to clients willing and FL server aggregation efficiency, we defines the corresponding training data requirements is μ. Therefore, the aggregation function in FL server is

$$w_G^{i+1} = \frac{1}{m} \sum_{k=1}^{m} w_k^i \tag{2}$$

FL training task is iterated till the global loss function converges, or achieved desirable accuracy Ω.

B. SYSTEM MODEL AND ASSUMPTION

Fig. 1 shows a typical architecture of MEC system in which contains our method for FL. MEC system consists of an edge server that broadcasts global model, a radio frequency source (RF) that charging for clients, and n clients that

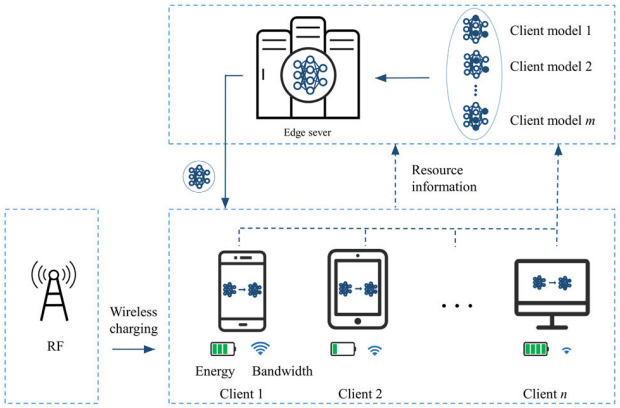

FIGURE 1. Illustration of adaptively client selection in MEC system.

equipped with CPU, energy unit and storage. Instead of randomly selecting clients, we propose a DQN-based client selection approach. First, n clients inform edge server with their resource information. And then edge server estimates MEC system energy consumption and transmission delay in next global model training progress that selecting a specific set of $N' = \{1, 2, \ldots, m\}$ clients that utilizing resource information. We assume the limitation of time required for broadcasting global model, updating local model and uploading local model is $L_{\max}$, i.e.,

$$L_{max} \geq \max\{T_k^{local} + T_k^{trans}\}, \quad \text{where } k \in N' \tag{3}$$

We assume that each client k has independent CPU-cycle frequency f_k, energy units e_k, wireless bandwidth r_k and wireless charging. The selected clients update local model in parallel with corresponding training data requirements μ that requires μG CPU cycles each iteration where G is the training required number of CPU cycles for each bit of data. Therefore, the local computing time required is

$$T_k^{local} = \frac{\mu G}{f_k}, \quad \text{where } k \in N' \tag{4}$$

Then, selected clients upload local model parameters which rely on wireless communication channels. Therefore, the other time required for global model updating which called transmission time can be calculated as

$$T_k^{trans} = \frac{D}{r_k}, \quad \text{where } k \in N' \tag{5}$$

where D is the size of the global model. We assume time required L for global model transmission and local model training depend on local model training delay and local model upload delay in which the Internet upload speed is typically much slower than download speed [14]. L_{max} is defined as

$$L_{max} \geq \max\{\frac{\mu G}{f_k} + \frac{D}{r_k}\}, \quad \text{where } k \in N' \tag{6}$$

Edge server aggregates updated local models' parameters that uploading by selected clients with FedAvg algorithm and evaluates the new global model performance with certain validation data set. Edge server repeats above steps multiple iterations until global model achieves a desirable accuracy Ω or arrives the final deadline.

IV. PROBLEM FOMULATION

The problem of adaptive client selection in MEC system for training a shared model could be formulated as MDP that defined as $\langle S, A, P, R \rangle$, where S, A, P, and R are state space, action space, state transition probability and reward function, respectively.

A. STATE SPACE

Let the state space of MEC system be represented by n clients resource information that is defined as

$$S = \prod_{k=1}^{n} S_k \tag{7}$$

where $\prod$ is the Cartesian product, and S_k is the state of client k. The state space of client k is expressed as

$$S_k = \{f_k, e_k, r_k; f_k \leq F, e_k \leq E, r_k \leq R\} \tag{8}$$

where F, E, R are the limitation of CPU-cycle frequency, energy units and wireless bandwidth, respectively.

B. ACTION SPACE

The action space of MEC system is the combination of the selection policy by edge server which includes n clients. Action space could represent

$$A = \prod_{k=1}^{n} A_k \tag{9}$$

where $A_k = \{0\} \cup \{1\}$ that the action state of client k. $A_k = 0$ means that client k does not participate in updating global model this round, whereas $A_k = 1$ means that client k take part in this round to train a local model.

C. STATE TRANSITION PROBABILITY

The state transition of MEC system from the current state s to the next state s' is determined based on the transitions of all clients' states. For the energy unit state, each client that taking part in updating global model required consumes energy units which consists of local training and transmission. In this paper, we assume energy unit consumption B_k depending on training which the transmission power is relatively low. The energy unit consumption each iteration can be calculated as

$$B_k = f_k^2 \tau \mu G \tag{10}$$

where f is assumed to follow the uniform distribution and τ is determined by the effective switched capacitance that depends on the chip architecture of client [26].

At each iteration, client k has e_k energy units and could be charged with C_k energy units from environment. The energy transition from the current energy state e_k to the next energy state e_k is calculated as

$$e_k{}' = \max\{e_k - B_k + C_k, 0\} \tag{11}$$

Charging energy units C_k is assumed to follow the Poisson distribution, i.e.,

$$\mathrm{P}(C_k = t) = e^{\omega}\frac{\omega^t}{t!} \tag{12}$$

where ω is the average energy arrival rate.

D. REWARD FUNCTION

Edge server selects adaptive clients with adequate resources to take part in model updating till achieves the desirable accuracy in which consume CPU-cycle frequencies and energy units. Edge server makes the best decision to maximum resource utilization of MEC system.

We demonstrate the effectiveness of selecting more clients to participate in updating global model to speed up convergence. In this experiment, we train a two 16×16 layers with each layer followed by 2×2 max pooling convolutional neural network model on MNIST dataset which contains 60,000 training samples. We consider training data of each client $C = 1$ MB, set non-IID setting σ is 30, make the desirable accuracy Ω is 99%, and each updating process on a different thread. As shown in Fig. 2, FL server selects 1 clients randomly needs more than 200 rounds to achieve desirable accuracy Ω. And FL server selects 2 to 4 clients randomly needs 135, 106, 82 communication rounds respectively. It is evident that the convergence speed of 4 clients participate in global model training than other setting in early round, which means more clients take part in FL process will lead a faster convergence speed.

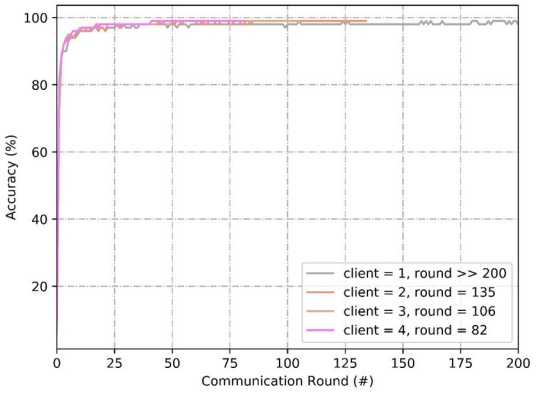

FIGURE 2. Accuracy with communication rounds.

Therefore, reward function should be proportional to the number of clients and inversely proportional to energy consumption and training delay. The reward function could be defined as

$$R(s,a) = \alpha_n\frac{m}{n} - \alpha_e\frac{E}{E_{max}} - \alpha_l\frac{L}{L_{max}} \tag{13}$$

where α_n, α_e, α_l are the scale factors. E is the total energy units consumption by MEC system in each iteration, i.e.,

$$E = \sum_{k=1}^{m} B_k, \quad \text{where } k \in N' \tag{14}$$

TABLE 1. List of notations.

Symbol	Quantity
μ	Training data
m	Number of clients participating in updating model
n	Number of clients
w_G^i	Global model parameters in iteration i
w_k^i	Local model parameters of client k in iteration i
$l_k(w)$	Loss function of client k
σ	Non-IID setting
ε	Number of local training iterations
Ω	Desirable accuracy
G	Number of CPU cycles required to process one bit local data
f_k	CPU-cycle frequencies for local training
e_k	Energy level of client k
r_k	Wireless bandwidth of client k
C_k	Amount of harvested energy of client k
B_k	Energy consumed by client k
D	Global model size
τ	Effective switched capacitance
α_n	Scale factor for client number
α_e	Scale factor for energy units
α_l	Scale factor for delay

E_{max} is the total energy units of system, i.e.,

$$E_{max} = \sum_{k=1}^{n} E, \quad \text{where } k \in N \tag{15}$$

L is the maximum delay for clients that taking part in this iteration, i.e.,

$$L = \max\{(\frac{\mu G}{f_k} + \frac{D}{r_k})\} \quad \text{where } k \in N' \tag{16}$$

For ease of reference, our commonly used symbols are summarized in Table 1.

E. DEEP REINFORCEMENT LEARNING BASED ON CLIENT SELECTION

According to the current state of all clients $s \in S$, edge server selects $a \in A$ to maximize the long-term cumulative reward of the entire system. Edge server makes the best decision which is defined as $\pi^* : S \rightarrow A$. In order to find the best decision, standard Q-Learning is often used [10]. Q-Learning constructs a continuously updated Q-value table with action-state pairs and then looks up Q-table to obtain the best decision which namely $\mathrm{Q}(s, a)$ [27]. Edge server updates Q-value based on experience replay, as follows:

$$Q'(s,a) = (1-\beta)\, Q(s,a) + \beta[R(s,a) + \gamma \max_{a' \in A} Q(s', a')] \tag{17}$$

where β is learning rate, γ is discount factor [10].

After updating Q(s, a), the server can rely on Q-table to proceed. According to any state s, edge server could select action a with the largest cumulative reward to make the optimal decision π^*. However, Q-table will occupy large storage resource in which the state space is always cursed by dimensionality that the time of searching the optimal policy in Q-table is very long. Traditional Deep Q-Network (DQN) uses a single neural network (NN) to make the optimal decision for edge server to reduce Q-table's storage and speed up searching. However, the single NN has the overoptimistic Q-value estimate problem since the same derived policy network to select and to evaluate an action for client selection policy.

Therefore, we propose Double Deep Q-Network (DDQN) that consists an action-value online network Y and an action-value target network Y'. The online NN updates its weights θ based on experience replay $\langle s, a, r, s' \rangle$, and the target NN resets its weights $\theta' = \theta$ regularly with gradient descent algorithm [10]. The loss function is defined as

$$L\theta = \mathbb{E}[\text{y-Q}(s, a; \theta)]^2 \tag{18}$$

where target value y is defined as

$$y = R(s, a) + \gamma Q'(s', \arg\max_{a' \in A} Q(s', a'; \theta); \theta') \tag{19}$$

DDQN selects an action for edge server based on online NN and evaluates the action with target NN that prevents the policy overoptimistic problem [28]. Algorithm 1 shows the DDQN training process for making an optimal decision that finds proper clients to update global model in each round.

Algorithm 1 DDQN-Based Client Selection

Initialize the replay memory M, action-value function with random weights θ and target action-value function with random weights $\theta' = \theta$, learning rate β, discount factor γ, global model weights.
for episode $i = 1$ to U **do**
  Initialize clients states $\in sS$.
  for iteration $k = 1$ to achieve the desirable accuracy **do**
    Select an action a randomly with probability ϵ or $a = \arg max_{a \in A} Q(s_k, a, \theta)$ with probability $(1 - \epsilon)$.
    Execute action a and update global model, observe reward r and next state s'.
    Store experience $\langle s, a, r, s' \rangle$ in M.
    Sample a minibatch size of m experiences from M.
    Update θ with gradient descent by m experiences with (18).
    Regularly resets $\theta' = \theta$.
  end for
end for

Before training process, FL server initializes replay memory M, learning rate β and discount factor γ, sets online action-value function with random weights θ, sets target action-value function weights $\theta' = \theta$. Each training process, online NN selects an action randomly with probability $\in$ or executes all actions. FL server observes all selected clients' reward r and next state s' to find the optimal decision while storing experience $\langle s, a, r, s' \rangle$ in M. Then, a mini-batch size of m experiences from M is sampled to update the online NN. FL server sets the target action-value weights $\theta' = \theta$. Agent is updated till global model achieve the desirable accuracy Ω.

V. EXPERIMENTAL RESULTS

In this section, we provide a performance evaluation to demonstrate the effectiveness of our proposed method. We measure the algorithm with the all MEC system's energy consumption and delay. We use Q-learning algorithm, random algorithm, and greedy algorithm as baselines.

A. GENERAL SETUP

We assume that the MEC system consists of 1 edge server and 4 IOT devices. Edge server initializes global model in which consists of two 32×32 convolution layers followed by 2×2 max pooling and two 64×64 layers followed by 2×2 max pooling. Clients upload their resource information such as wireless channel states and energy states. Each client has privacy data $C = 1$ MB in which the non-IID setting σ is 30% and the dataset is MNIST. Client device's CPU-cycle frequency f_k follow the uniform distribution U[0, 1] and wireless bandwidth r_k follow the uniform distribution U[0, 2]. Edge server employs DDQN, Q-learning, random and greedy algorithms to select clients participating in updating global model. DDQN will train an agent to select clients adaptively to update global model. The mechanism of Q-learning is creating a Q-table for searching the most decision. Random algorithm will select clients randomly to take part in model training process. Greedy algorithm will invite all clients to update the global model. All parameters are shown in Table 2.

B. MNIST EXPERIMENTS

Our initial study includes three consumption on MNIST datasets and set $\Omega = 99\%$. The tasks that clients need to handle are continuously sent from other clients or servers in which task queue is always full. Even if client can charge from a nearby RF source, but also needs to properly arrange their own energy to avoid additional consumption, resulting in inefficient task process. Therefore, we regard system energy planning as the most important influencing factor. As shown in Figure 3, in each iteration, DDQN, Q-learning, random and greedy algorithms needs to consume about 2.0, 2.5, 3.5, 4.0 energy units after convergence. It is conspicuous that DDQN algorithm reduces extra energy consumption in model training by up to 50% compare with greedy algorithm in MNIST dataset. DDQN also has greatly development compared with random algorithm. The reason is that greedy algorithm will choose as many clients as possible to participate in

TABLE 2. Simulation parameters.

Parameters	Value
n	4
μ	1 MB
σ	30%
D	20 Mbit
G	7000
ω	1
τ	10^{-28}
E	5
F	2 GHz
R	2 Mbps
L_{max}	500 s
NN size	$64 \times 64 \times 64$
ϵ - greedy	$0.1 \rightarrow 0$
α_n	3
α_e	2
α_l	2

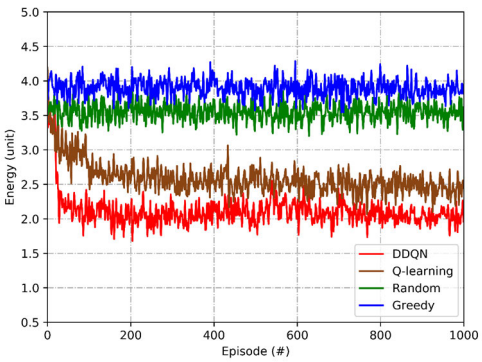

FIGURE 3. Energy consumption comparison.

model training, as long as the clients have sufficient energy to support the global model training. In this case, greedy algorithm cannot achieve an effective balance between CPU-cycle frequency and energy consumption, resulting in low energy utilization efficiency. On the contrary, in terms of energy-saving, DDQN can effectively select suitable clients adaptively from heterogeneous clients to reduce extra energy units consumption.

Energy is the basis of the client training model. In addition to energy, delay determines the iteration speed and convergence speed of the global model. For the MEC system, less delay means less time required for each model update. In other words, the algorithm with less delay will iterate the global model more times at the same time. As shown in Figure 4, DDQN, Q-learning, random and greedy algorithms take 230, 240, 260, 290 seconds separately after

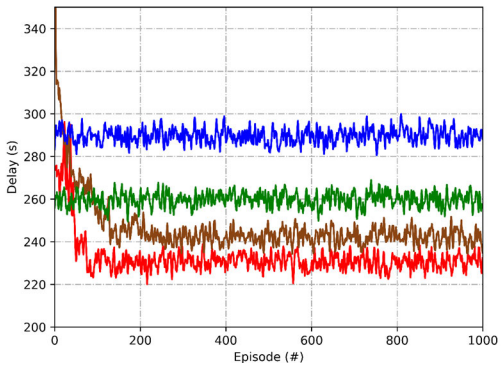

FIGURE 4. Delay consumption comparison.

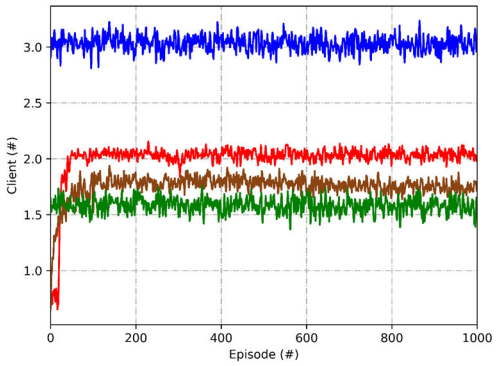

FIGURE 5. Client consumption comparison.

convergence. It is obvious that DDQN reduces the average delay of each model update by up to 20.70% compared to greedy algorithm. Compared with another static algorithms, random algorithm also has a little improvement. The reason is that greedy algorithm will select as many clients as possible within the limited delay L_{max}. This leads to high delay of greedy algorithm and is greatly affected by the CPU-cycle frequencies and wireless bandwidth. In other words, the delay of each client in the system will vary greatly and will fluctuate greatly. This is unfavorable for the system to select clients, because it cannot make good use of devices which has rich computing resources, wireless bandwidth resources, and energy units. Equipment with high training time delay drags down the training efficiency of the entire system. In contrast, DDQN can select clients with similar delay in the system. This not only prevents poorly-resourced equipment from affecting the whole system, but also improves the efficiency.

As shown in Figure 5, DDQN, Q-learning, random and greedy algorithms select average 2, 1.7, 1.5, 3.0 clients to take part in global model training process. It is clearly that greedy algorithm selects 75% of all clients to participate in global model training each iteration, while DDQN can select 52% of the clients to participate. We also studied the performance of global model communication rounds after convergence. Fig. 6 shows that center, DDQN, Q-learning, random and greedy algorithms select clients with their own

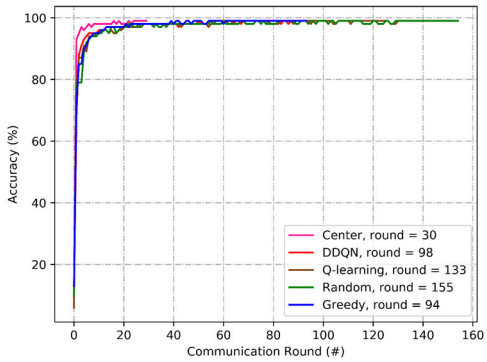

FIGURE 6. Accuracy with communication rounds after convergence.

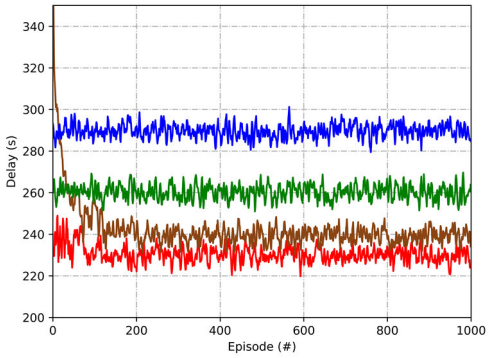

FIGURE 8. Delay consumption comparison on fashion-MNIST.

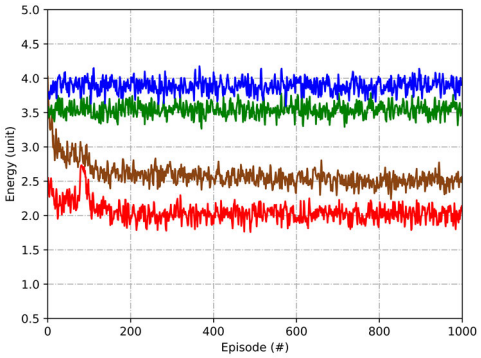

FIGURE 7. Energy consumption comparison on fashion-MNIST.

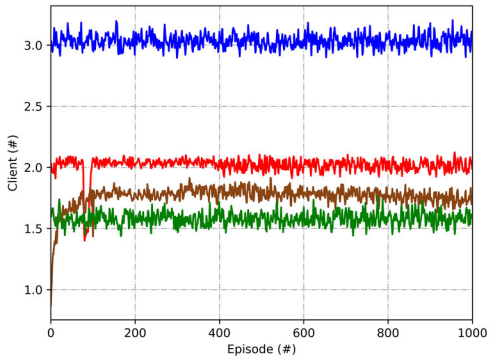

FIGURE 9. Client consumption comparison on fashion-MNIST.

mode needs 30, 98, 133, 155 and 94 communication rounds to achieve desirable accuracy. Though greed algorithm only faster a little than DDQN algorithm. We infer the reason for the similar performance between greed and DDQN algorithm is that DDQN will be giving priority to clients which are more benefit for the convergence with reducing the resource consumption.

C. DIFFERENT EDGE SERVER TASKS

We also ran experiments on different FL server tasks which including the Fashion-MNIST dataset to further validate our algorithm in which $\Omega = 89\%$. As shown in Figure 7-10, we ran experiments on the Fashion-MNIST dataset. In each iteration, DDQN, Q-learning, random and greedy algorithms needs to consume about 2.0, 2.5, 3.6, 3.9 energy units separately which DDQN algorithm reduces energy unit computation by up to 48.7% compare with greedy algorithm after convergence. And DDQN, Q-learning, greedy and random algorithms needs about 233, 239, 261 and 287 seconds, respectively. Our algorithm reduces the average delay in each iteration by up to 18.82% compared with greedy algorithm while average engagement clients are 2.1, 1.7, 1.6, 3.0, respectively and communication rounds are 172, 186, 251, 151, separately in which need more communication rounds to achieve desirable accuracy compare with center algorithm.

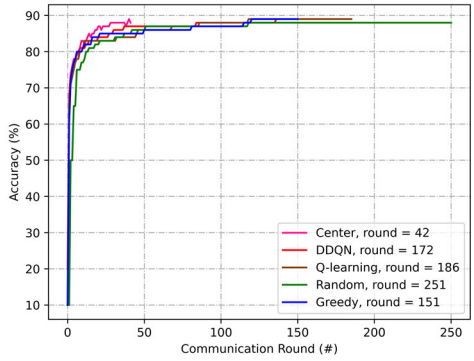

FIGURE 10. Accuracy with communication rounds after convergence.

In different edge server tasks, it is obvious that the performance and the improvement of MEC system resources of our algorithm are similar. From experiments on Fashion-MNIST dataset and MNIST dataset, the MEC system proposed in this paper has strong scalability to accommodate different edge server tasks, while effectively scheduling and optimizing the resources in which clients' computation, energy and wireless bandwidth resources are limited. However, it can be clearly seen that the DDQN algorithm could reach the optimal state faster on the Fashion-MNIST dataset than it trained on the MNIST dataset in which the state could optimize MEC system resources more effectively. We conjecture this is largely due to edge server requires more

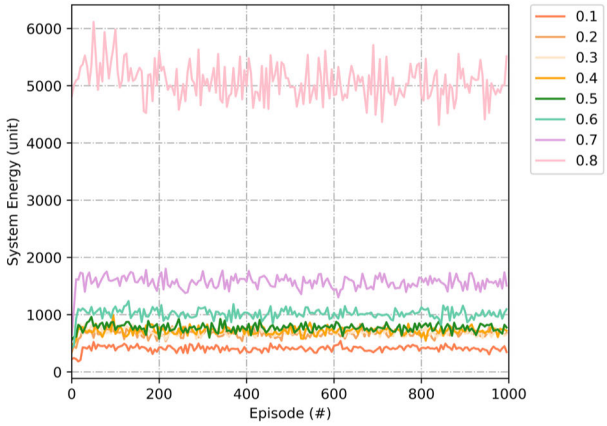

FIGURE 11. MEC system energy consumption with different levels of non-IID setting.

communication rounds to achieve the desirable accuracy on the Fashion-MNIST dataset in each iteration in which give the experience-driven decision-making algorithm more training rounds. But, it increases averagely the required MEC system energy units by up to 40% compared to it trained on the MNIST dataset.

D. DIFFERENT LEVELS OF NON-IID SETTING

FL has been shown to work well approximating the model trained on centrally collected data in which the data and label distributions are independent and identically distributed on MEC system. However, in reality, FL is unstable and may even diverge when clients' data distributions are non-IID in which the different performances and usage patterns. In this secession, we explore the influence of different levels of non-IID settings on Fashion-MNIST dataset. As shown in Figure 11, MEC system will consume more energy units with the setting of non-IID increases and even lead convergence failure. This is due to the inconsistency between the locally performances, which aims to minimize the loss value on local model and edge server aims to minimize the overall loss on the MEC system. As we keep fitting models on different clients to heterogeneous local data of clients, the divergence among the weights of local models will be accumulated and eventually degrades the performance of FL which will lead to more communication rounds before global model converges or global model reach the desirable accuracy and more MEC system energy units consumption.

VI. CONCLUSION

In this paper, we focus on the research for resource optimization of adaptive client selection for training global model in MEC system with each client resources is changed in real time each model updating process while recent researches pay more attention to select fixed number of clients. We then model global model training process in MEC system as a MDP and demonstrate a faster convergence speed with more parallel clients. In order to solve the formula MDP, we propose an adaptive client selection algorithm based on DDQN. The algorithm can not only withstand the curse of the dimensionality of the state space, but also does not require any prior information about MEC system. In the simulation, we found that our DDQN-based client selection algorithm reduces the energy unit consumption by up to 50% and the delay by up to 20.70% compared with greedy algorithm, while the communication rounds is just increased 4%. Comparing with other baseline, our algorithm represents better performance. Then, we explore the impact of different edge server tasks and demonstrate the scalability of our proposed MEC system. Finally, we discuss the influence of different levels of non-IID setting and find the divergence of local models will degrade the performance of FL and even make more energy consumption. In the future, we will concentrate on solving the curse of action space to extend our algorithm could withstand the rapidly gowning number of client devices.

REFERENCES

[1] A. Krizhevsky, I. Sutskever, and G. E. Hinton, "ImageNet classification with deep convolutional neural networks," in *Proc. Annu. Conf. Neural Inf. Process. Syst.*, 2012, pp. 1097–1105.

[2] T. Mikolov, M. Karafiát, L. Burget, J. Černocký, and S. Khudanpur, "Recurrent neural network based language model," in *Proc. 11th Annu. Conf. Int. Speech Commun. Assoc.*, 2010, pp. 1045–1048.

[3] C. Sun, A. Shrivastava, S. Singh, and A. Gupta, "Revisiting unreasonable effectiveness of data in deep learning era," in *Proc. IEEE Int. Conf. Comput. Vis. (ICCV)*, Oct. 2017, pp. 843–852.

[4] R. Yadav, W. Zhang, O. Kaiwartya, H. Song, and S. Yu, "Energy-latency tradeoff for dynamic computation offloading in vehicular fog computing," *IEEE Trans. Veh. Technol.*, vol. 69, no. 12, pp. 14198–14211, Dec. 2020.

[5] H. Li, K. Ota, and M. Dong, "Learning IoT in edge: Deep learning for the Internet of Things with edge computing," *IEEE Netw.*, vol. 32, no. 1, pp. 96–101, Jan./Feb. 2018.

[6] M. Chiang and T. Zhang, "Fog and IoT: An overview of research opportunities," *IEEE Internet Things J.*, vol. 3, no. 6, pp. 854–864, Dec. 2016.

[7] B. McMahan, E. Moore, D. Ramage, S. Hampson, and B. A. Y. Arcas, "Communication-efficient learning of deep networks from decentralized data," in *Proc. 20th Int. Conf. Artif. Intell. Statist.*, in Proceedings of Machine Learning Research, vol. 54, A. Singh and J. Zhu, Eds. Fort Lauderdale, FL, USA: PMLR, Apr. 2017, pp. 1273–1282.

[8] K. Bonawitz, H. Eichner, W. Grieskamp, and D. Huba, "Towards federated learning at scale: System design," *CoRR*, vol. abs/1902.01046, 2019. [Online]. Available: http://arxiv.org/abs/1902.01046

[9] T. Nishio and R. Yonetani, "Client selection for federated learning with heterogeneous resources in mobile edge," in *Proc. IEEE Int. Conf. Commun. (ICC)*, May 2019, pp. 1–7.

[10] H. Van Hasselt, A. Guez, and D. Silver, "Deep reinforcement learning with double Q-learning," in *Proc. AAAI Conf. Artif. Intell.*, 2016, pp. 2094–2100.

[11] H. Wang, Z. Kaplan, D. Niu, and B. Li, "Optimizing federated learning on non-IID data with reinforcement learning," in *Proc. IEEE Conf. Comput. Commun. (INFOCOM)*, Jul. 2020, pp. 1698–1707.

[12] V. Smith, C.-K. Chiang, M. Sanjabi, and A. S. Talwalkar, "Federated multi-task learning," in *Proc. Adv. Neural Inf. Process. Syst.*, 2017, pp. 4424–4434.

[13] L. Huang, Y. Yin, Z. Fu, S. Zhang, H. Deng, and D. Liu, "LoAdaBoost: Loss-based AdaBoost federated machine learning with reduced computational complexity on IID and non-IID intensive care data," 2018, *arXiv:1811.12629*. [Online]. Available: http://arxiv.org/abs/1811.12629

[14] J. Konečný, H. Brendan McMahan, F. X. Yu, P. Richtárik, A. Theertha Suresh, and D. Bacon, "Federated learning: Strategies for improving communication efficiency," 2016, *arXiv:1610.05492*. [Online]. Available: http://arxiv.org/abs/1610.05492

[15] T. Li, M. Sanjabi, and V. Smith, "Fair resource allocation in federated learning," *CoRR*, vol. abs/1905.10497, 2019. [Online]. Available: http://arxiv.org/abs/1905.10497
[16] R. Yadav, W. Zhang, K. Li, C. Liu, M. Shafiq, and N. K. Karn, "An adaptive heuristic for managing energy consumption and overloaded hosts in a cloud data center," *Wireless Netw.*, vol. 26, no. 3, pp. 1905–1919, Apr. 2020.
[17] R. Yadav, W. Zhang, O. Kaiwartya, P. R. Singh, I. A. Elgendy, and Y. Tian, "Adaptive energy-aware algorithms for minimizing energy consumption and SLA violation in cloud computing," *IEEE Access*, vol. 6, pp. 55923–55936, 2018.
[18] R. Yadav and W. Zhang, "MeReg: Managing energy-SLA tradeoff for green mobile cloud computing," *Wireless Commun. Mobile Comput.*, vol. 2017, pp. 1–11, 2017.
[19] R. Yadav, W. Zhang, H. Chen, and T. Guo, "MuMs: Energy-aware VM selection scheme for cloud data center," in *Proc. 28th Int. Workshop Database Expert Syst. Appl. (DEXA)*, Aug. 2017, pp. 132–136.
[20] L. Huang, S. Bi, and Y.-J.-A. Zhang, "Deep reinforcement learning for online computation offloading in wireless powered mobile-edge computing networks," *IEEE Trans. Mobile Comput.*, vol. 19, no. 11, pp. 2581–2593, Nov. 2020.
[21] X. Chen, H. Zhang, C. Wu, S. Mao, Y. Ji, and M. Bennis, "Performance optimization in mobile-edge computing via deep reinforcement learning," in *Proc. IEEE 88th Veh. Technol. Conf. (VTC-Fall)*, Aug. 2018, pp. 1–6.
[22] M. Min, L. Xiao, Y. Chen, P. Cheng, D. Wu, and W. Zhuang, "Learning-based computation offloading for IoT devices with energy harvesting," *IEEE Trans. Veh. Technol.*, vol. 68, no. 2, pp. 1930–1941, Feb. 2019.
[23] H. T. Nguyen, N. Cong Luong, J. Zhao, C. Yuen, and D. Niyato, "Resource allocation in mobility-aware federated learning networks: A deep reinforcement learning approach," 2019, *arXiv:1910.09172*. [Online]. Available: http://arxiv.org/abs/1910.09172
[24] S. Feng, D. Niyato, P. Wang, D. I. Kim, and Y.-C. Liang, "Joint service pricing and cooperative relay communication for federated learning," in *Proc. Int. Conf. Internet Things (iThings) IEEE Green Comput. Commun. (GreenCom) IEEE Cyber, Phys. Social Comput. (CPSCom) IEEE Smart Data (SmartData)*, Jul. 2019, pp. 815–820.
[25] Y. Zhan and J. Zhang, "An incentive mechanism design for efficient edge learning by deep reinforcement learning approach," in *Proc. IEEE Conf. Comput. Commun.*, Aug. 2020, pp. 2489–2498.
[26] T. D. Burd and R. W. Brodersen, "Processor design for portable systems," *J. VLSI Signal Process. Syst.*, vol. 13, nos. 2–3, pp. 203–221, Aug. 1996.
[27] C. J. C. H. Watkins and P. Dayan, "Technical note Q-learning," *Mach. Learn.*, vol. 8, no. 3, pp. 279–292, May 1992.
[28] V. Mnih, K. Kavukcuoglu, D. Silver, A. A. Rusu, J. Veness, M. G. Bellemare, A. Graves, M. Riedmiller, A. K. Fidjeland, G. Ostrovski, S. Petersen, C. Beattie, A. Sadik, I. Antonoglou, H. King, D. Kumaran, D. Wierstra, S. Legg, and D. Hassabis, "Human-level control through deep reinforcement learning," *Nature*, vol. 518, pp. 529–533, Feb. 2015.

HANGJIA ZHANG received the B.E. degree from the School of Marine Engineering Equipment, Zhejiang Ocean University, Zhejiang, China, in 2018. He is currently pursuing the M.E. degree with Ningbo University, Zhejiang. His research interests include mobile edge computing and machine learning.

ZHIJUN XIE was born in 1974. He received the Ph.D. degree in computer science from Renmin University, Beijing, China. He is currently an Associate Professor with the Department of Information Science and Engineering, Ningbo University, China. He is also currently leading a Research Team in the area of wireless body area sensor networks for human centric sensing. His research interests include wireless sensor networks and body area sensor networks.

ROOZBEH ZAREI received the B.Sc. degree in electrical and electronics engineering from Azad University, Iran, in 2006, the M.E. degree in electrical–electronics and telecommunications engineering from Universiti Teknologi Malaysia (UTM), Malaysia, in 2012, and the Ph.D. degree in computer science from Victoria University, Australia, in 2017. He is currently an Associate Research Fellow with the School of Information Technology, Deakin University, Melbourne, Australia. His current research interests include data stream mining, anomaly detection, pattern recognition, and data analytics (statistics and machine learning).

TAO WU was born in 1994. He is currently pursuing the M.E. degree with Ningbo University, Zhejiang, China. His research interests include wireless sensor networks and robotic path planning.

KEWEI CHEN is currently a Professor with Ningbo University. His main research interests include intelligent manufacturing, robotics, and computational intelligence.