Online Learning based Fast-Convergent and Energy-Efficient Device Selection in Federated Edge Learning

Cheng Peng, Qin Hu, Zhilin Wang, Ryan Wen Liu, Zehui Xiong

***Abstract*—As edge computing faces increasingly severe data security and privacy issues of edge devices, a framework called federated edge learning (FEL) has recently been proposed to enable machine learning (ML) model training at the edge, ensuring communication efficiency and data privacy protection for edge devices. In this paradigm, the training efficiency has long been challenged by the heterogeneity of communication conditions, computing capabilities, and available datasets at devices. Currently, researchers focus on solving this challenge via *device selection* from the perspective of optimizing energy consumption or convergence speed. However, the consideration of any one of them is insufficient to guarantee the long-term system efficiency and stability. To fill the gap, we propose an optimization problem to simultaneously minimize the total energy consumption of selected devices and maximize the convergence speed of the global model for device selection in FEL, under the constraints of training data amount and time consumption. For the accurate calculation of energy consumption, we deploy *online bandit learning* to estimate the CPU-cycle frequency availability of each device, based on an efficient algorithm, named *Fast-Convergent Energy-Efficient Device Selection (FCE^{2}DS)*, is proposed to solve the optimization problem with a low level of time complexity. Through a series of comparative experiments, we evaluate the performance of the proposed FCE^{2}DS scheme, verifying its high training accuracy and energy efficiency.**

***Index Terms*—Edge computing, federated learning, reinforcement learning, online bandit learning.**

I. Introduction

SINCE the traditional Internet of Things (IoT) mainly rely on network connections and computing services in the cloud or data center, sending data back and forth between IoT devices and the cloud can result in heavy response delay and low operation efficiency. As a distributed architecture for data storage and computing at the edge of the network, edge computing provides efficient computing services to edge devices with timely communications. Thanks to the short distance between IoT devices and the connected edge servers, edge computing can significantly reduce the communication delay, thereby improving computing and communication efficiency. However, since the data of edge devices will be exposed during transmission to the edge server, the risk of being tampered with or stolen makes devices vulnerable to various data security and privacy threats [1]. To protect the data privacy in edge computing, federated learning (FL) enables edge devices to only share machine learning (ML) model parameters to the edge server after training the global model using their own data. This can effectively resolve the security and privacy problems caused by direct data transmission from local devices to edge servers, thus being termed as federated edge learning (FEL). Combining edge computing and FL, FEL can complete ML model training tasks at the edge while ensuring communication efficiency and protecting data privacy for edge devices.

However, the generated data of devices in edge computing are usually non-independent and identically distributed (non-IID); besides, their communication conditions and computing capabilities are also heterogeneous, which greatly challenges the training efficiency of FEL. Considering these complicate factors, many studies are devoted to improving training efficiency through communication compression [2], [3] and resource allocation [4], [5]. Although these studies reduce the communication or computing time of the devices to a certain extent, the participation of devices with poor computing and communication capabilities or inadequate local data may reduce the training efficiency of FEL.

To fundamentally improve the training efficiency of FEL, it is necessary to filter participated devices before each communication round. Currently, researchers mainly focus on the perspectives of *energy consumption* [6]–[9] and *convergence speed* [10]–[17] to select devices for training. In general, both energy consumption and convergence speed are important factors affecting the performance of FEL systems, but almost all existing studies only consider one of them. If only the energy consumption is taken into account, it may take a long time to complete the training process, and the efficiency of the FEL system can be sacrificed. While, if only convergence rate is considered, devices with poor communication conditions and weak computing power consume too much energy to participate in local training, which may cause unexpected shutdowns of batter-powered devices and affect the performance of FEL.

To overcome these shortcomings, we formulate an optimization problem to simultaneously minimize the total energy consumption of selected devices and maximize the convergence speed of the global model in every communication round, under the constraints of total data amount and time consump-

This work is partly supported by the US NSF under grant CNS-2105004.

Cheng Peng, Qin Hu (Corresponding author), and Zhilin Wang are with the Department of Computer and Information Science, Indiana University-Purdue University Indianapolis, IN 46202, USA. E-mail: {cp16, qinhu,wangzhil}@iu.edu

Ryan Wen Liu is with the School of Navigation, Wuhan University of Technology, Wuhan 430063, China. E-mail: wenliu@whut.edu.cn

Zehui Xiong is with the Information Systems Technology and Design Pillar, Singapore University of Technology and Design, Singapore 119260. E-mail: zehui_xiong@sutd.edu.sg

tion. However, there exist two major challenges in solving the optimization problem. First, the accurate calculation of energy consumption is closely related to the CPU-cycle frequency of each device, which is often dynamically fluctuated as the CPU can be occupied by multiple tasks at the same time and cannot be known a priori. Second, as an aggregation result, the convergence speed reflects the contribution of all devices participating in the training for the global model, which makes it difficult to analyze the contribution of every single device for selection.

To address the first challenge, we use an online bandit learning method [18] to estimate the frequency of each device in a more accurate manner so as to better optimize the calculated energy consumption. For the second challenge, since devices with higher local losses can speed up the convergence of the global model due to their local data diversity as proved in [15], we are inspired to select devices based on their loss values for fast convergence.

However, after solving these challenges, we find that there is no trivial solution for the optimization problem. Since each edge device has one of two possible states in each communication round of FEL, i.e., selected or unselected, an arbitrary solution requires exponentially increasing time complexity as the number of devices increases, which burdens the server and reduce the overall efficiency. To solve the optimization problem with a lower time cost, we come up with an algorithm called Fast-Convergent Energy-Efficient Device Selection (FCE^{2}DS). The main contributions of this work are summarized as follows:

- To accelerate the convergence speed of the global FEL model, we propose an energy-efficient loss model that can be used by the server to collect the local loss value of each device.
- Considering the time limit of each communication round and the required data diversity for FEL training, we propose an optimization problem to minimize the total energy consumption of selected devices in the communication and computing processes, as well as maximize the convergence speed of the global model.
- To facilitate the accurate calculation of time and energy consumption, we introduce a reinforcement learning method to estimate the CPU-cycle frequency of each device.
- To achieve the optimization goal with reduced time complexity, we reformulate the proposed optimization problem and propose a solution named FCE^{2}DS with high efficiency to select devices in each communication round of FEL.
- To evaluate the performance of our proposed FCE^{2}DS scheme, a series of comparative experiments are conducted to verify its high accuracy and energy efficiency.

The rest of this paper consists of the following five sections. Section II summarizes the most related work about device selection in FEL. Section III introduces our system model and problem formulation, which is reformulated and solved in Section IV. Then our proposed scheme is evaluated in Section V. Section VI concludes the whole paper.

II. Related Work

With the wide deployment of FL at the edge, extensive research has been conducted to optimize the training efficiency of FEL system [2]–[5]. However, due to the heterogeneity of computing power, communication conditions and data sets owned by devices, it turns out to be unreasonable to randomly select a subset of devices or simply include all devices for local training in FEL. For devices with negligible impact on model convergence or consuming excessive energy and time, the performance of the FEL system will be significantly sacrificed. To this end, the selection of devices for each communication round of FEL has become an important research topic. Generally, the research about device selection in FEL can be realized by considering two factors: *energy consumption* [6]–[9] and *convergence speed* [10]–[16].

As for the research about device selection considering energy consumption, Zeng *et al.* [6] proposed a bandwidth allocation and scheduling scheme that provides selection priority for each device based on the the communication condition, so as to reduce energy consumption during communication. Kim *et al.* [7] proposed the AutoFL framework to optimize energy efficiency, selecting devices according to features such as data heterogeneity and runtime variance. Besides, AdaSplit was proposed in [8] to split and train the global models asynchronously, and an advantage function was designed to select devices so as to reduce energy consumption during data communication. In addition, Peng *et al.* [9] proposed a scheme named E^{2}DS to select devices considering energy consumption and time consumption in both communication and local computing processes.

With the consideration of the convergence speed, Wu *et al.* [10] proposed a scheme named FedProf to estimate the data dissimilarity of each device, which is used as the selection probability for training. Zhang *et al.* [11] analyzed the data importance of each device by studying the model parameters uploaded by devices, and then selected devices to participate in local training according to the data importance, so as to reduce the impact of non-IID data on convergence. By utilizing the superposition feature of wireless to achieve multiple access channels, Yang *et al.* [12] proposed a concurrent transmission device selection method to shorten the aggregation time and improve the training efficiency. Marnissi *et al.* [13] studied the relationship between gradients locally trained by devices and the impact on global model convergence, revealing that the device with the highest gradient norm is more important for the convergence of the global model. Tang *et al.* [14] proposed the FedGP algorithm to model the loss changes of devices for capturing data correlations, based on which devices are selected for training to accelerate the convergence speed. Besides, Cho *et al.* [15], [16] investigated the relationship between the loss value and convergence speed. By selecting devices with higher local training loss values, the convergence speed can be greatly accelerated and the communication efficiency can be improved. By solving the linear programming problem, Ko *et al.* [17] studied device selection and appropriate bandwidth allocation for devices in each round, so as to reduce the communication time and accelerate the convergence of the

global model.

However, it is clear that almost all the existing studies only consider either the effect of energy consumption or convergence speed on the FEL system. If only energy consumption is considered, the FEL system may select less number of devices with higher energy efficiency for local training, which would reduce the diversity of data, resulting in a slow convergence speed and reducing the training efficiency of the FEL system. While if only the convergence rate is considered, the energy required to complete the training for the selected devices can be uncontrollable, which can make the battery-powered devices exhausted quickly. To overcome these limitations, we propose a more comprehensive device selection scheme that considers energy and time consumption in the local computing and communication processes, as well as optimizes loss values of devices so as to increase the convergence speed in a more direct way. Overall, through the device selection process before local training, the total energy consumed during the FEL process will be significantly reduced with fast convergence of the global model.

III. System Model

In this paper, we consider a federated edge learning (FEL) system, consisting of one edge server in charge of aggregating the global model and multiple devices registered on the server to train local models based on their own private data. The set of devices on the edge is denoted by $\mathcal{K} = \{1, 2, \cdots, k, \cdots, K\}$. In fact, since there are a large number of devices on the edge with diverse communication and computing capabilities, device selection in each communication round becomes a feasible method to improve the training efficiency of FEL. Specifically, we consider the trade-off between the resource consumption of devices and the convergence speed of the global model to select devices, where the required number of data samples have to be provided by selected devices for model training while the local training and update time constraints need to be satisfied.

In the following, we introduce the FEL system and process in Section III-A, and then establish the time and energy consumption models in Section III-B. Besides, the loss model is presented in Section III-C, the convergence cost function is defined in Section III-D, and the overall problem formulation is elaborated in Section III-E.

A. FEL Process

In an FEL task, the time limit of waiting for devices to upload model updates is usually predefined as a constant parameter by the edge server, which can be denoted as T^{wait}. For the FEL system with device selection, six main steps are involved, as illustrated in Fig. 1, and can be introduced as follows:

- At the beginning of each communication round, the server needs to conduct the step of *Device Information Collection* to prepare essential device parameters for device selection, such as local data size, computing power, transmission power and bandwidth information.
- When the server receives necessary information from devices[1], the server calculates the best set of devices to optimize the FEL system performance in *Device Selection*. The detailed problem formulation and solution will be discussed in the following sections.
- After the determination of device selection in this communication round, the server sends the global model to all devices in the step of *Selection Results and Global Model Distribution*.
- Once devices receive parameters, they will use their local data to train the received global model in the step of *Local Learning at Devices*.
- In the *Learning Results and Local Model Uploading* step, after training the local model based on all the local data, each selected device sends the local model updates back to the server. While unselected devices send their estimated loss values to the server.
- In the step of *Global Model Aggregation*[2], the server aggregates the received model updates uploaded by T^{wait} to derive the updated global model. Then the server finishes this communication round of FEL and starts the next round from the first step, i.e., *Device Information Collection*.

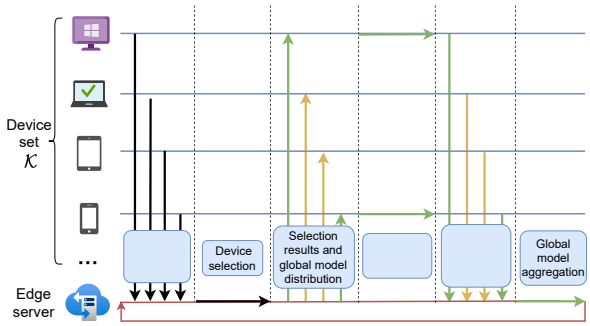


Fig. 1. The working process of an FEL communication round.

B. Time and Energy Consumption Models

To ensure training efficiency of the FEL system without wasting too much time for waiting, the server selects the optimal set of devices for local training in each communication round by limiting the maximum waiting time for devices submitting model updates as T^{wait}. Due to the relatively powerful computing and communication capabilities at the server, the time and energy consumption in multiple steps main involving the server's operation can be negligible, such as *Device Information Collection*, *Device Selection*, and *Global Model Aggregation*, which cannot be further optimized by selecting appropriated participated devices. While other steps closely related to devices are worthy of analysis and optimization, including the steps of *Selection Results and Global Model*

[1]For devices not returning required information, the server considers them as disconnected for this round of FEL training and will not include them for device selection.

[2]This step can be performed jointly with the step of *Device Information Collection* in the next communication round to reduce the time consumption of the overall FEL system.

Distribution, *Local Learning at Devices*, and *Learning Results and Local Model Uploading*.

Here we use $T_k^D(t)$ to denote the time spent by device k in round t during the *Selection Results and Global Model Distribution* step and $T_k^U(t)$ to represent the time spent in the *Learning Results and Local Model Uploading* step. In general, $T_k^D(t)$ and $T_k^U(t)$ are jointly determined by the size of the model parameters, denoted as D^p, and the transmission speeds in the downloading and uploading processes, which can be calculated as:

$$T_k^D(t) = \frac{D^p}{V_k^D(t)}, \tag{1}$$

$$T_k^U(t) = \frac{D^p}{V_k^U(t)}. \tag{2}$$

In the above equations, $V_k^D(t)$ and $V_k^U(t)$ are the downloading and uploading transmission speeds in round t that are closely related to the condition of the communication channel. Specifically, $V_k^D(t)$ and $V_k^U(t)$ can be calculated as:

$$V_k^D(t) = B_k^D(t)\log(1 + \frac{P_k(t)h_k(t)^2}{N_0}), \tag{3}$$

$$V_k^U(t) = B_k^U(t)\log(1 + \frac{P_k(t)h_k(t)^2}{N_0}). \tag{4}$$

In (3) and (4), $B_k^D(t)$ and $B_k^U(t)$ are respectively the downloading and uploading bandwidth of device k's communication channel in round t, $P_k(t)$ and $h_k(t)$ are respectively the transmission power and the channel gain [19], and N_0 is the background noise.

Then, we define the energy consumption of device k in steps of *Selection Results and Global Model Distribution* and *Learning Results and Local Model Uploading* in round t, denoted by $E_k^D(t)$ and $E_k^U(t)$, respectively. As the energy consumption during transmission is the product of the transmission power and the transmission time, $E_k^D(t)$ and $E_k^U(t)$ can be calculated as:

$$E_k^D(t) = T_k^D(t)P_k(t),$$

$$E_k^U(t) = T_k^U(t)P_k(t).$$

Next, to calculate the time and energy consumption of devices during the local learning process, we denote $c_k(t)$ as the number of CPU cycles for each device k to complete training using one data sample in round t, which can be measured locally and then reported to the server. Via defining D_k as the size of device k's local dataset, the total number of CPU cycles needed for device k to finish local training in round t is $c_k(t)D_k$. Besides, we denote the CPU-cycle frequency of device k in round t as $F_k(t)$. Based on these notations, the time spent in the local learning step of device k in round t, denoted by $T_k^{LC}(t)$, can be represented as [20]:

$$T_k^{LC}(t) = \frac{c_k(t)D_k}{F_k(t)}. \tag{5}$$

Since CPU can be occupied by multiple tasks at the same time, the CPU-cycle frequency may vary greatly during local computing and cannot be known a priori. To obtain an estimated value, we use the expected value $f_k(t)$ as the CPU-cycle frequency of device k in round t, which is shown as:

$$f_k(t) = \mathbb{E}[F_k(t)]. \tag{6}$$

Similar to the time consumption in the step of *Local learning at devices*, the energy consumption of device k in round t, denoted as $E_k^{LC}(t)$, is also closely related to its local data amount and CPU-cycle frequency. In detail, $E_k^{LC}(t)$ can be expressed as [21]:

$$E_k^{LC}(t) = \frac{\alpha_k}{2}c_k D_k F_k(t)^2,$$

where α_k is the effective capacitance coefficient of the computing chip-set in device k.

Overall, by combining the time spent in each step, the total time and energy consumption of device k in round t, denoted as $T_k(t)$ and $E_k(t)$, respectively, can be calculated as:

$$T_k(t) = T_k^D(t) + T_k^{LC}(t) + T_k^U(t), \tag{7}$$

$$E_k(t) = E_k^D(t) + E_k^{LC}(t) + E_k^U(t).$$

C. Loss Model

To facilitate the selection of devices, we define a binary variable $x_k(t) \in \{0, 1\}$ to indicate whether device k is selected or not in round t, where $x_k(t) = 1$ denotes that device k is selected for local training in this FEL round while $x_k(t) = 0$ means not being chosen.

In the environment of federated learning, the loss value of each device after local training can reflect the heterogeneity of local data to some extent, affecting the convergence speed of the global model [15], which can be used as a metric to help facilitate the selection of participated devices in the next round. If device k is selected in round t, the loss value can be generated after completing the local learning process, which is denoted by $G_k^c(t)$. For devices that are not selected in the t-th round, their accurate loss values cannot be derived without completing the local training. In order to obtain approximate values for these devices with less energy waste for computing, each unselected device is required to train a mini-batch of the local data to have an estimated loss value, denoted by $G_k^e(t)$, which is calculated as:

$$G_k^e(t) = \sum_{\xi \in \hat{\xi}_k(t)} \frac{L_k(w, \xi, t)}{|\hat{\xi}_k(t)|}.$$

In the above equation, $\hat{\xi}_k(t)$ is the mini-batch data owned by device k, which is sampled uniformly at random from D_k in round t. $L_k(w, \xi, t)$ is the loss function of device k with parameters of the global model w and the sample data ξ from $\hat{\xi}_k(t)$ in round t. Based on these two ways to efficiently collect loss values of all devices, we can calculate the loss value of device k in communication round t, denoted as $G_k(t)$, by

$$G_k(t) = \begin{cases} G_k^e(t), & \text{if } x_k(t) = 0, \\ G_k^c(t), & \text{if } x_k(t) = 1. \end{cases}$$

D. Convergence Cost Function

Given the fact that edge devices are usually battery-powered with limited energy supply, it is necessary to minimize the total energy consumption of all participating devices to avoid accidental device disconnection, thereby improving the cost-effectiveness of FEL. In addition, since devices with higher loss values can contribute more to the convergence of the trained global model due to their data heterogeneity [15], selecting these devices to join in the local training step can speed up the convergence in the real federated learning scenario.

Considering the impacts of the above two important factors on device selection, we define a convergence cost function $R_k(t)$ to describe the contribution of device k to the global model in round t, which can be expressed as:

$$R_k(t) = (\eta E_k(t) - G_k(t-1))x_k(t), \tag{8}$$

where $\eta > 0$ is a scalar for value balance. In detail, $R_k(t)$ is the difference between the energy consumption and the loss value of the selected device k, indicating that the device with a lower $R_k(t)$ contributes more to the convergence of the global model while consuming less energy. Note that since the loss value result can only be obtained after the local training, here we use $G_k(t-1)$ updated in round $t-1$ for calculating the convergence cost in round t.

E. Problem Formulation

As mentioned above, it is necessary to reduce the error for each device in calculating energy consumption caused by the fluctuation of the CPU cycle frequency while considering the loss value, which can be implemented by minimizing the time-averaged convergence cost $R_k(t)$ over time duration T.

Furthermore, the amount of training data will also significantly affect the performance of FEL, where too little data will lower the convergence speed and model precision. Here we define a parameter $a \in (0,1)$ for the edge server to adjust the ratio of data required in FEL to the total amount of data $D_{all} = \sum_{k=1}^{K} D_k$. Moreover, the time limit T^{wait} for the server to collect local model updates should also be a constraint considered for all devices.

To achieve the goal of speeding up the global model convergence with device energy consumed as less as possible while meeting constraints mentioned above, we formulate the following optimization problem:

$$\min: \ \frac{1}{T}\sum_{t=0}^{T-1}\mathbb{E}[\sum_{k=1}^{K} R_k(t)], \tag{9}$$

$$\text{s.t.}: \ x_k(t) \in \{0,1\}, \tag{9a}$$

$$T_k \le T^{wait}, \tag{9b}$$

$$\sum_{k=1}^{K} D_k x_k(t) \ge a \cdot D_{all}. \tag{9c}$$

In the above equations, the optimization goal (9) is to minimize the expected convergence cost of the selected devices. The first constraint (9a) ensures that the selection indicator $x_k(t)$ should only be 0 or 1. The constraint (9b) guarantees that the total time consumption of each selected device should be less than the maximum waiting time T^{wait} set by the server. And the last constraint (9c) specifies that the total data volume of the selected devices have to be sufficient to train the model in each communication round.

IV. Algorithm Design for Energy-efficient Device Selection with Fast Convergence

In this section, we mainly introduce a reinforcement learning method to approach the CPU-cycle frequency of each device, and then design an algorithm to solve the optimization problem (9) after reformulation. Firstly, we introduce the general idea of overcoming the uncertainty of CPU-cycle frequency based on the online bandit learning in Section IV-A, where the specific calculation is represented in Section IV-B. After that, we reformulate the problem in Section IV-C, and design the algorithm FCE^{2}DS in Section IV-D.

A. General Idea

Since the CPU of each device could be occupied by multiple tasks at the same time, the CPU-cycle frequency that can be used for the FEL task is more or less lower than the rated frequency in the practical situation. To obtain an accurate estimation of CPU-cycle frequency that can be used for local training at each device, we introduce the combinatorial multi-armed bandit (CMAB) algorithm to learn the CPU-cycle frequency for approaching the real situation.

In detail, the edge server is regarded as a player, each device is treated as an arm, and the action of selecting an appropriate set of devices by the server at the beginning of every communication round is considered as pulling multiple arms. Since the goal of our problem is to minimize the total convergence cost $\sum_{k=1}^{K} R_k(t)$, devices with smaller $R_k(t)$ are more likely to be selected. Thus, here the reward of pulling any arm, i.e., selecting a device, is $-R_k(t)$ in the CMAB problem, and the device with a larger reward should be more worthy of being selected for FEL training.

Specifically, by obtaining the communication and computing power of all devices at the *Device Information Collection* step, the server can calculate the energy and time consumption, as well as the loss value in the previous round, as a reference for device selection in the current communication round. In practice, however, the CPU-cycle frequency that affects time and energy consumption is unknown and only the rated value, instead of the frequency value dedicated to the FEL task, is available to the server. Therefore, after the local training in each communication round, the server calculates the true CPU-cycle frequency based on the information of time consumed by devices, and then updates the frequency information for each device based on the feedback.

In this case, to learn the CPU-cycle frequency of each device in every round more accurately, we need to solve the trade-off problem of "exploitation" and "exploration" in the CMAB problem. In this model, "exploitation" refers to choosing the arm (i.e., device) with a larger reward to participate in local training; "exploration" refers to choosing a new arm

(device) that has never been chosen before to participate in local training. After obtaining the trade-off reward value, we dynamically adjust the device selection strategy under the constraints of waiting time and training data size to maximize the sum of the trade-off reward value. Therefore, inspired by the CMAB problem, we can propose an algorithm to solve the problem of device selection with unknown CPU-cycle frequency.

B. Specific Calculation

We can get the real communication and calculation time of device k by recording the time difference between the server sending model parameters to this device in the *Selection Results and Global Model Distribution* step and receiving the result from device k in the *Learning Results and Local Model Uploading* step in round t, which is denoted as $T_k^{Final}(t)$. Then, according to the definition of $T_k(t)$ in (7), as well as other relevant equations (1), (2), and (5), the real value of CPU-cycle frequency of device k in round t, i.e., $F_k(t)$, is calculated as:

$$F_k(t) = \frac{c_k(t)D_k}{T_k^{Final}(t) - T_k^D(t) - T_k^U(t)}. \tag{10}$$

According to (10), the optimization goal in (9) can be rewritten as:

$$\begin{aligned}
&\frac{1}{T}\sum_{t=0}^{T-1}\mathbb{E}[\sum_{k=1}^{K}R_k(t)] \\
=&\frac{1}{T}\sum_{t=0}^{T-1}\mathbb{E}[\sum_{k=1}^{K}(\frac{\eta D^p}{B_k^D(t)\log(1+\frac{P_k(t)h_k^2(t)}{N_0})}P_k(t) \\
&+\frac{\alpha_k}{2}\eta c_k D_k F_k(t)^2 + \frac{\eta D^p}{B_k^U(t)\log(1+\frac{P_k(t)h_k(t)^2}{N_0})}P_k(t) \\
&- G_k(t-1))x_k(t)].
\end{aligned}$$

Under the consideration that the CPU-cycle frequency $F_k(t)$ is independent of other parameters and $f_k(t) = \mathbb{E}[F_k(t)]$ according to (6), the above equation can be updated as:

$$\begin{aligned}
&\frac{1}{T}\sum_{t=0}^{T-1}\mathbb{E}[\sum_{k=1}^{K}R_k(t)] \\
=&\frac{1}{T}\sum_{t=0}^{T-1}\sum_{k=1}^{K}\{\mathbb{E}[(\frac{\eta D^p}{B_k^D(t)\log(1+\frac{P_k(t)h_k^2(t)}{N_0})}P_k(t) \\
&+\frac{\eta D^p}{B_k^U(t)\log(1+\frac{P_k(t)h_k(t)^2}{N_0})}P_k(t) \\
&- G_k(t-1))x_k(t)] + \frac{\alpha_k}{2}\eta c_k D_k f_k(t)^2\mathbb{E}[x_k(t)]\}.
\end{aligned} \tag{11}$$

To minimize (11), it is required to learn the unknown CPU-cycle frequency of each device, i.e., $f_k(t)$. To aim this, we denote $n_k(t)$ as the number of communication rounds that device k is selected to participate in local training with this FEL task in the first t communication rounds, and it can be calculated as:

$$n_k(t) = \sum_{\tau=0}^{t-1} x_k(\tau). \tag{12}$$

Then, in order to record a stable CPU-cycle frequency for each device, we maintain an average value for the previous t rounds of device k, which is defined as $\bar{f}_k(t)$. According to the value of the CPU-cycle frequency of the previous t rounds $F_k(t)$ and the number of times that device k is selected in (12), we can calculate the average value $\bar{f}_k(t)$ as:

$$\bar{f}_k(t) = \frac{\sum_{\tau=0}^{T-1} F_k(\tau)x_k(t)}{n_k(t)}.$$

Therefore, after receiving the real CPU-cycle frequency $F_k(t)$ in each communication round, the server will update $\bar{f}_k(t)$ and $n_k(t)$ for each device, regardless of whether it is selected to participate in this round of training. Respectively, $n_k(t)$ and $\bar{f}_k(t)$ can be updated by:

$$n_k(t) = \begin{cases} n_k(t-1), & \text{if } x_k(t) = 0, \\ n_k(t-1)+1, & \text{if } x_k(t) = 1, \end{cases} \tag{13}$$

$$\bar{f}_k(t) = \begin{cases} \bar{f}_k(t-1), & \text{if } x_k(t) = 0, \\ \dfrac{n_k(t-1)\bar{f}_k(t-1) + F_k(t)}{n_k(t)}, & \text{if } x_k(t) = 1. \end{cases} \tag{14}$$

Once the average CPU-cycle frequency of each device is updated in each round, we complete the "exploitation" part of reinforcement learning. To allow the devices that have not been selected to have more opportunities to update the CPU-cycle frequencies for approximating their true values, which is the "exploration" part, we introduce the variable $\widetilde{f}_k(t)$ as the estimated value of the CPU-cycle frequency of device k in round t. In this article, we use the Upper Confidence Bound (UCB) [22] [23] algorithm to update $\widetilde{f}_k(t)$, which is shown below:

$$\widetilde{f}_k(t) = \bar{f}_k(t) + \sqrt{\frac{2\ln K}{n_k(t)}}.$$

In the above equation, $\widetilde{f}_k(t)$ is composed of two parts, considering both the "exploitation" and the "exploration" parts. Specifically, $\bar{f}_k(t)$ represents that with more times being selected, the confidence interval of the device will be narrower and the uncertainty of the estimation is lower. In the contrast, those devices with larger $\bar{f}_k(t)$ have more probabilities to be selected. Meanwhile, $\sqrt{\frac{2\ln K}{n_k(t)}}$ refers to that the fewer the number of attempts for a device, the wider the confidence interval and the higher the uncertainty, which means the device with a wider confidence interval tends to be selected multiple times.

Therefore, the variable $\bar{f}_k(t)$ represents the CPU-cycle frequency closest to the true value of the device, and $\widetilde{f}_k(t)$ gives suggestions for the next round of device selection, based on the perspectives of giving chances to all devices to update their real information. By doing so, the situation of selecting only the devices with better initial performance is avoided, and

the server also gives opportunity to devices with poor initial performance to be selected and update their information.

Base on this, we define $\widetilde{S}(T)$ as an average expected convergence cost that considers the whole FEL task from the beginning to the T-th round, which can be rewritten as:

$$\begin{aligned}\widetilde{S}(T) =& \frac{1}{T}\sum_{t=0}^{T-1}\mathbb{E}[\sum_{k=1}^{K}R_k(t)] \\ =&\frac{1}{T}\sum_{t=0}^{T-1}\sum_{k=1}^{K}\{\mathbb{E}[(\frac{\eta D^p}{B_k^D(t)\log(1+\frac{P_k(t)h_k^2(t)}{N_0})}P_k(t) \\ &+\frac{\eta D^p}{B_k^U(t)\log(1+\frac{P_k(t)h_k(t)^2}{N_0})}P_k(t) \\ &-G_k(t-1))x_k(t)]+\frac{\alpha_k}{2}\eta c_k D_k \widetilde{f}_k(t)^2\mathbb{E}[x_k(t)]\}.\end{aligned}$$

C. Problem Reformulation

Now, we can rewrite the optimization problem in Section III-E as follows:

$$\begin{aligned}\min:\ & \widetilde{S}(T), \\ \text{s.t.}:\ & \text{(9a)(9b)(9c)}.\end{aligned} \tag{15}$$

It is clear that the optimization problem is to select the optimal subset of devices that minimizes the total convergence costs under the constraints of time consumption and training data size. Thus, the above optimization problem requires complex combinatorial optimization. An intuitive solution to this problem is to traverse all device combinations and then compare them to obtain the best set of devices. However, considering that each device has two states, i.e., selected and unselected, it will take $O(2^K)$ time complexity to traverse all devices, which will not only consume a huge amount of time in each communication round but also severely reduce the training efficiency. To avoid this situation, we transform the above optimization problem into an effective maximization problem, and propose a solution with lower time complexity based on dynamic programming.

Since in each communication round, the devices registered on the edge are known, the calculated convergence cost of each device is fixed after the information of all devices is collected. Regarding the energy cost, we can find that the total energy consumption of all devices in each round is fixed, and there are only two choices for devices in each communication round, i.e., to be selected or not to be selected to participate in local training. Based on this, selecting a set of devices with lower energy consumption to participate in FEL is equivalent to excluding a set of devices with higher energy consumption and allowing the rest to participate in training. As for the convergence speed, after devices with lower loss values are selected, the rest of devices will have the higher loss values. Therefore, we can see that if a set of devices with a higher sum of convergence cost in round t can be selected and removed, the rest of devices are the optimal set of devices that solves our optimization problem.

As defined earlier, the state indicator of device $k \in \mathcal{K}$ is $x_k(t)$, which is equal to 1 when device k is selected in round t. To better distinguish the selected set and the unselected set of devices, we denote $y_k(t)$ to indicate the state of device k which is not selected for participating in FEL in round t, so we have $y_k = 1 - x_k$. Then, to solve the reformulated optimization problem in (15), we define $\widetilde{S}_y(T)$ as the average expected sum of cost convergence from the beginning to the T-th round of the FEL task, which can be calculated as:

$$\begin{aligned}\widetilde{S}_y(T) =&\frac{1}{T}\sum_{t=0}^{T-1}\sum_{k=1}^{K}\{\mathbb{E}[(\frac{\eta D^p}{B_k^D(t)\log(1+\frac{P_k(t)h_k^2(t)}{N_0})}P_k(t) \\ &+\frac{\eta D^p}{B_k^U(t)\log(1+\frac{P_k(t)h_k(t)^2}{N_0})}P_k(t) \\ &-G_k(t-1))y_k(t)]+\frac{\alpha_k}{2}\eta c_k D_k \widetilde{f}_k(t)^2\mathbb{E}[y_k(t)]\}.\end{aligned} \tag{16}$$

In this case, according to (16), the problem in (15) can be rewritten as:

$$\max:\ \widetilde{S}_y(T), \tag{17}$$
$$\text{s.t.}:\ y_k(t) \in \{0,1\}, \tag{17a}$$
$$T_k \le T^{wait}, \tag{17b}$$
$$\sum_{k=1}^{K} D_k y_k(t) \le (1-a)\cdot D_{all}, \tag{17c}$$

where the data size constraint (17c) means the total data size of selected devices not participating in FEL should be less than $(1-a)\cdot D_{all}$ to guarantee enough data size of devices participating in training.

After reformulating the optimization problem into a maximum problem with an upper limit of total training data size rather than a lower limit, we can solve the optimization using a dynamic programming method, which will be elaborated in the below.

D. Algorithm Design

As discussed above, the sum of convergence costs for all devices registered on the edge server in each communication round is fixed. Based on the complementary relationship of selected and unselected devices, we transform the minimization problem defined in (15) into a maximization problem in (17). To solve it, we propose a Fast-Convergent Energy-Efficient Device Selection (FCE^{2}DS) algorithm and specify it in Algorithm 1.

Generally, the above maximization problem with an upper limit constraint can be transferred to a 0-1 Knapsack problem[3]. In the 0-1 knapsack problem, there are many items with their respective values and a knapsack with limited capacity. By choosing whether putting each item into the backpack or not, the total value of the backpack can be maximized under the capacity constraint. Inspired by this problem, here

[3]For the classical 0-1 Knapsack problem, dynamic programming has been proved to be an optimal solution in many studies [24]. So we omit the proof details in this paper for brevity.

we propose a DeviceSelection algorithm to solve the problem of selecting devices with different convergence costs under the data size constraint defined in (17). In detail, for every communication round t, there are K different devices that can be selected or not for joining FEL, and the capacity is the maximum remaining data size $(1-a)\cdot D_{all}$ with an extra constraint of time consumption. For simplicity, we denote $D^{cap} = (1-a)\cdot D_{all}$ as the data size capacity of unselected devices, and $v_k(t) = R_k(t)$ as the value of device k in round t. Then a sequence $\mathcal{V}(t) = \{v_1(t), v_2(t), \cdots, v_K(t)\}$ is defined to contain the convergence cost values of all devices, while the sequence of data sizes is denoted by $\mathcal{D} = \{D_1, D_2, \cdots, D_K\}$.

After device selection, the selected devices will participate in one communication round of local computing to update the global model. Meanwhile, the real CPU-cycle frequency data of each selected device can be calculated, so as to learn the expected CPU-cycle frequency using the bandit learning method. The expected CPU-cycle frequency will continue to improve the accuracy of energy consumption calculation, and then impact the device selection in the next communication round.

Algorithm 1 FCE^{2}DS

Input: the number of current communication round t, the number of devices K, the size set of devices $\mathcal{D} = \{D_1, D_2, \cdots, D_K\}$, the data size capacity of unselected devices D^{cap}

1: **for** $k \in \{1,2,\cdots,K\}$ **do**
2: $\bar{f}_k(0) \leftarrow 0$
3: $n_k(0) \leftarrow 1$
4: **end for**
5: **for** communication round $t = 1, 2, \cdots$ **do**
6: Calculate the value set of devices $\mathcal{V}(t) = \{v_1(t), v_2(t), \cdots, v_K(t)\}$
7: $\{x_1, x_2, \cdots, x_K\} \leftarrow$ DeviceSelection$(K, \mathcal{V}, \mathcal{D}, D^{cap})$
8: **for** $k \in \{1,2,\cdots,K\}$ **do**
9: *Local learning at devices* according to x_k and update G_k
10: $n_k(t) = \begin{cases} n_k(t-1), & x_k(t) = 0 \\ n_k(t-1)+1, & x_k(t) = 1 \end{cases}$
11: $\bar{f}_k(t) = \begin{cases} \bar{f}_k(t-1), & x_k(t) = 0 \\ \dfrac{n_k(t-1)\bar{f}_k(t-1) + F_k(t)}{n_k(t)}, & x_k(t) = 1 \end{cases}$
12: $\widetilde{f}_k(t) = \bar{f}_k(t) + \sqrt{\frac{2\ln K}{n_k(t)}}$
13: **end for**
14: **end for**

As shown in Algorithm 1, there are four input parameters: the number of current communication round t, the number of devices K, the size set of devices $\mathcal{D} = \{D_1, D_2, \cdots, D_K\}$, and the data size capacity of unselected devices D^{cap}. To initiate the whole algorithm, for every device logged in the edge server, the average CPU-cycle frequency in the first communication round $\bar{f}_k(0)$ is set to 0 (Line 2). Since $n_k(t)$ will be used as a divisor in the following equation, we set its initial value to 1 (Line 3). Then, in each communication round, we start by calculating the convergence cost value of each device according to the function defined in (8) (Line 6), and next calculate the optimal device selection solution using DeviceSelection algorithm (Line 7), which will be specifically demonstrated in Algorithm 2. After selection, the selected device will participate in the local training process and update training results to the server (Line 8). Then, with the latest updated parameters, the number of each device being selected $n_k(t)$, the average frequency value $\bar{f}_k(t)$, and the estimated frequency value $\widetilde{f}_k(t)$ are learnt and updated according to (13) and (14) (Lines 9-12).

Algorithm 2 DeviceSelection

Input: the number of devices K, the value set of devices $\mathcal{V} = \{v_1, v_2, \cdots, v_K\}$, the size set of devices $\mathcal{D} = \{D_1, D_2, \cdots, D_K\}$, the data size capacity of unselected devices D^{cap}

Output: the state indicators of all devices $\{x_1, x_2, \cdots, x_K\}$

1: **for** $D \leftarrow 0$ to D^{cap} **do**
2: $DSA(0, D) \leftarrow 0$
3: **end for**
4: **for** $i \leftarrow 1$ to K **do**
5: $DSA(i, 0) \leftarrow 0$
6: **end for**
7: **for** $i \leftarrow 1$ to K **do**
8: **for** $D \leftarrow 1$ to D^{cap} **do**
9: **if** $D_i \le D$ **then**
10: **if** $v_i + DSA(i-1, D-D_i) > DSA(i-1, D)$ **then**
11: $DSA(i, D) \leftarrow v_i + DSA(i-1, D-D_i)$
12: **else**
13: $DSA(i, D) \leftarrow DSA(i-1, D)$
14: **end if**
15: **else**
16: $DSA(i, D) \leftarrow DSA(i-1, D)$
17: **end if**
18: **end for**
19: **end for**
20: **for** $i \leftarrow K$ to 1 **do**
21: **if** $DSA(i, D^{cap}) > DSA(i-1, D^{cap})$ **then**
22: $y[i] \leftarrow 1$
23: $D^{cap} \leftarrow D^{cap} - D[i]$
24: **else**
25: $y[i] \leftarrow 0$
26: **end if**
27: **end for**
28: **for** $i \leftarrow 1$ to K **do**
29: $x_i \leftarrow 1 - y_i$
30: **end for**
31: **return** $\{x_1, x_2, \cdots, x_K\}$

In Algorithm 2, the convergence cost value set of all devices $\mathcal{V}$ is input with K, $\mathcal{D}$, and D^{cap} to select devices. Firstly, a two-dimensional array $DSA(K, D^{cap})$ is used to store and update intermediate results of dynamic programming (Lines 1-19). Then, the output result of unselected devices are calculated according to $DSA(K, D^{cap})$ table (Lines 20-27). Finally, the indicator $y_k(t)$ of unselected devices is converted

to the indicator $x_k(t)$, which represents that device k will be selected in round t (Lines 28-31).

Overall, the computational overhead of our solution is mainly determined by the DeviceSelection part, so the time complexity of our FCE^{2}DS algorithm is $O(KD^{cap})$, where K is the number of devices that logged in the edge server in the FEL task, and D^{cap} is the data size capacity which is equal to $(1-a)\cdot D_{all}$.

V. Experimental Evaluation

A. FEL Environment Simulation

To evaluate the effectiveness and performance of our proposed scheme FCE^{2}DS, we establish a simulation environment of FEL and perform experimental verification. We also simulate the traditional FL (TFL) process without device selection and the energy-efficient device selection scheme E^{2}DS proposed in [9]. All three schemes are implemented on a desktop with Intel(R) Core(TM) i7-9750 CPU @2.60GHz and 16GB RAM running Windows 10 OS.

In order to make a reasonable and sufficient comparison, most of the parameter settings used in the E^{2}DS experiments are also applied here. Specifically, for the wireless communication simulation, a circular area with a radius of 50 meters is used as the coverage area of the edge server for the experiment, with the edge server located at the center. In the FEL task, there are 50 devices logged in the edge server, which are uniformly distributed with a range of 2 meters to 50 meters from the center of the circle. The channel gain h_k of device k follows the exponential distribution with the equation $g_0(d_0/d)^4$, where the reference distance $d_0 = 1$ meter, and $g_0 = -40$ dB [21]. The download bandwidth of each device B_k^D is set to follow the normal distribution with the mean and standard deviation being 5 MHz and 4 MHz, and as a practical bandwidth limitation, the upload bandwidth of devices B_k^U would be lower than the download bandwidth, which follows the normal distribution with the mean and standard deviation of 1 MHz and 0.1 MHz. The transmission power P_k is set as a normal distribution where the mean is 0.6 W and the standard deviation is 0.2 W. For other fixed values, we assume the background noise N_0 as 10^{-8} W, and the data size of model parameters is set as $D^p = 25,000$ nats, which is approximately equal to 4.5 KB.

For the local learning step, the training size D_k of each device is set as a normal distribution with the mean and standard deviation being 5 MB and 4 MB. To simulate the FEL scenario with non-IID data distribution, we distribute 30% of data from the same class to each device, and randomly select 70% of data from the remaining classes. We set the effective capacitance coefficient $\alpha_k = 2\times10^{-28}$. The number of CPU cycles c_k is normally distributed with the mean of 15 cycles/bits and the standard deviation of 10 cycles/bits, and the expected CPU-cycle frequency f_k follows the rule of a normal distribution with the mean of 0.5 GHz and the standard deviation of 0.1 GHz.

B. Model Training Settings

For the experiment of the FEL task, the dataset we use to train and test is MNIST which includes 60,000 handwritten digital images for training and 10,000 for testing with 10 classes. We compare our proposed FCE^{2}DS algorithm with the other two schemes, i.e., TFL and E^{2}DS [9]. In order to make the comparative experiment more convincing, we control the constraints unchanged, which means that all three schemes satisfy the constraints (17a), (17b) and (17c). Specifically, TFL randomly selects devices until the sum of data size reach $(1-a)\cdot D_{all}$ for training, and E^{2}DS refers to the optimization of energy consumption and the number of devices under time constraints when selecting devices.

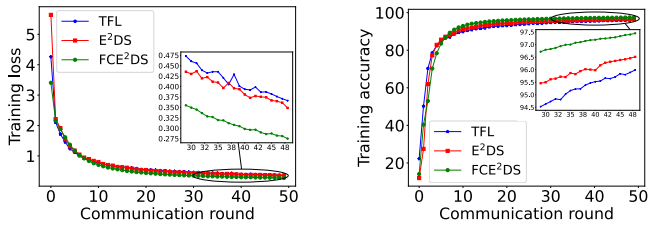
We use the same convolutional neural network as the global model for all three schemes, including three linear convolution layers. Specifically, the first layer contains 32 channels, while the second one has 64 channels and the third one has 128 channels. All layers are followed with 2×2 max pooling, which are activated by the ReLU function, and a final Softmax output layer afterward.

Then, we set the ratio of the necessary amount of data to the total amount of data for each round of FEL as $a = 0.75$ unless otherwise specified, which means there are three-quarters of the data used in each FEL round. In addition, the maximum waiting time in each round T^{wait} is set as 10 min unless specified. For the E^{2}DS scheme, we set the parameters according to [9]. In FCE^{2}DS, we set the weight of energy consumption as $\eta = 0.1$.

C. Evaluation Results

In this part, we first verify whether the FCE^{2}DS algorithm performs better by evaluating the loss trend, the accuracy trend, the number of devices and the energy consumption of the three schemes. Then, we evaluate the learning performance and cost of the FCE^{2}DS algorithm, which testifies the usability of our proposed scheme. To reduce the experimental error, we repeat each experiment 20 times and take the average value as the final result to ensure the reliability of experiments.

1) Comparison Experiments: To explore the effectiveness of FCE^{2}DS scheme, we study the performances of TFL scheme, E^{2}DS scheme, and FCE^{2}DS scheme.

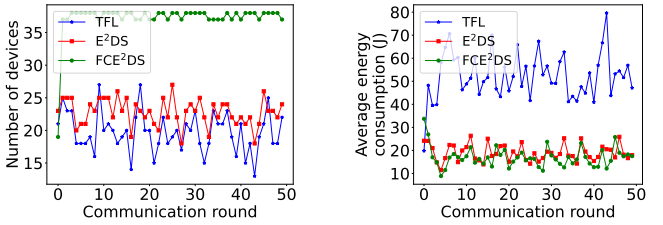


(a) Loss trend (b) Accuracy trend

Fig. 2. The comparison results of loss and accuracy trends in different schemes.

Firstly, we compare the loss and accuracy trends in different schemes, where the results are shown in Fig. 2. Specifically, as shown in Fig. 2(a), all three schemes converge after 50 communication rounds, but the loss of FCE^{2}DS is the smallest among them. In the final round 50, the loss of FCE^{2}DS reaches 0.274, while the losses of TFL and E^{2}DS are respectively 0.369 and 0.357, both larger than that of FCE^{2}DS. Since the

loss value is the most obvious result to show the convergence speed, it is clear to see that FCE^{2}DS has a better performance in convergence. Then, in Fig. 2(b), we can see that after 50 communication rounds, all the three schemes reach a high training accuracy. Separately, the accuracy of E^{2}DS scheme is 96.47%, which is a little larger than the accuracy of TFL (96.06%), while FCE^{2}DS scheme gets the highest accuracy of 97.48%. Above all, the FCE^{2}DS scheme shows the best performance and wins the comparison of not only the convergence speed but the accuracy.

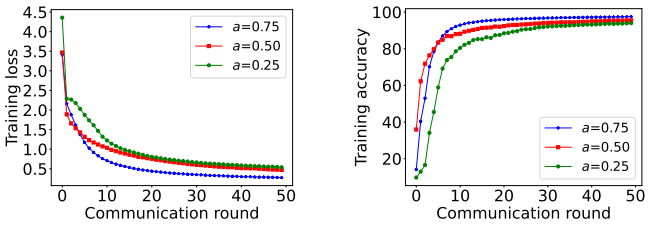


(a) Number of devices (b) Average energy consumption of each device

Fig. 3. The comparison results of device quantity and energy consumption in different schemes.

Then, Fig. 3 shows the comparison results of three schemes regarding the number of devices and the average energy consumption of each device. As shown in Fig. 3(a), FCE^{2}DS scheme selects more devices in each communication round, i.e., about 38 devices, which is larger than both the E^{2}DS and TFL schemes (about 10-25 devices in each round). We know that in a practical FEL environment, more devices mean that the model has more diverse data, which is better for the model to have a fast convergence performance. The result that FCE^{2}DS scheme selects more devices confirms that it can get the lowest loss value (shown in Fig. 2(a)). For the unusual number of selected devices in the FCE^{2}DS scheme for the first communication round, only 19 devices are selected, which is because the FCE^{2}DS scheme will randomly choose devices in the initialization stage to take part in the beginning round. For the energy consumption, Fig. 3(b) shows the average energy consumption of each device in different schemes. Since the number of selected devices is different for all the three schemes, it is more appropriate to compare the average energy consumption per device instead of the total energy consumption. We can see that the average energy consumption of both FCE^{2}DS scheme and E^{2}DS scheme are around 15-25 J per device, which is much less than the average energy consumption of TFL (about 40-70 J). By comparing FCE^{2}DS scheme and E^{2}DS scheme, in most of the communication rounds, the energy in FCE^{2}DS scheme consumes less and is more stable. In conclusion, FCE^{2}DS scheme has a better performance on energy consumption optimization.

2) Learning Performance and Cost: To explore the optimal parameters for training, we change the data size rate a to 0.25, 0.5, and 0.75, and see how the proposed FCE^{2}DS algorithm performs, with the results shown in Fig. 4. Specifically, Fig. 4(a) shows the difference of training loss values in each communication round. From the figure we can see that the FEL system has the fastest convergence speed when a = 0.75.



(a) Training Loss (b) Training Accuracy

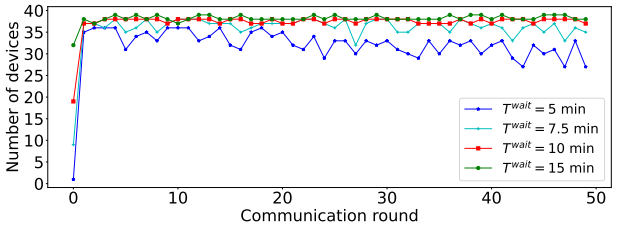
Fig. 4. Training loss and accuracy trends with different a.

Similarly, the FEL system gets the highest training accuracy when a = 0.75. Therefore, 75% of the data size rate to select devices is optimal for this FEL task to reach a high accuracy and fast convergence speed.

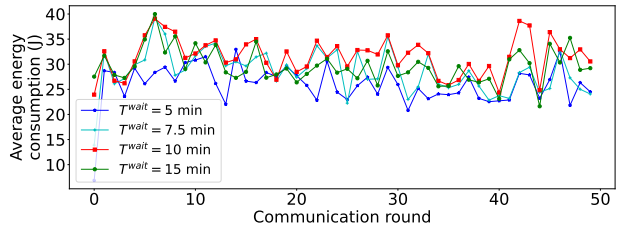
TABLE I
COMPARISON OF LEARNING RESULTS WITH DIFFERENT TIME LIMITS.

	T^{wait}	Communication Round t		
		$t = 10$	$t = 30$	$t = 50$
Accuracy	5 min	89.49%	96.14%	96.81%
	7.5 min	94.61%	96.99%	97.60%
	10 min	92.12%	96.71%	97.48%
	15 min	90.85%	94.37%	96.00%
Loss	5 min	0.88	0.41	0.35
	7.5 min	0.67	0.36	0.28
	10 min	0.76	0.35	0.27
	15 min	1.25	0.58	0.41

Then, by changing T^{wait} to 5 min, 7.5 min, 10 min, and 15 min, we study how the time limit influences the performance and energy consumption of the FEL system. The comparison results of learning performance with different time limits are shown in Table I, including accuracy and loss at different communication rounds. Besides, the comparison results of device quantity and energy in different time limits are shown in Fig. 5.



(a) Number of devices



(b) Average energy consumption of each device

Fig. 5. The comparison results of device quantity and energy consumption with varying time limits.

For detailed analysis, in Table I, the accuracy and loss values of different time limits are listed separately for various numbers of communication rounds. For the accuracy results, we can see that the FEL system could reach the highest accuracy and the lowest loss value when $T^{wait} = 7.5$ min through the whole task. In addition, in Fig. 5(a), it is clear that when $T^{wait} = 7.5$ min, there are more devices selected to take part in local training than the number of selected devices when $T^{wait} = 5$ min, which means the data amount used to train the model is greater as well. Combining Table I and Fig. 5(a) to analyze together, the limit of training data is the reason of the FEL system not reaching the best performance when $T^{wait} = 5$ min.

Then, for the experiments with longer T^{wait} (i.e., 10 min and 15 min), as the results shown in Table I, the FEL system does not perform as well as that of $T^{wait} = 7.5$ min. Until the 50-th round, the accuracy and convergence speed are still very slow, which means it needs more communication rounds to train the model and it is more difficult to have a good training result. This is because in an edge environment, the more required communication rounds, the more difficult to maintain the stability and durability of the edge devices. Moreover, in Fig. 5(b), the FEL system is more energy-efficient when $T^{wait} = 5$ min and 7.5 min, as the average energy consumption of selected devices is lower, compared with that of $T^{wait} = 10$ min and 15 min. Above all, $T^{wait} = 7.5$ min is the most suitable time limit for this FEL task.

VI. Conclusion

In this paper, to enable device selection in FEL for higher training efficiency, we propose an optimization problem to minimize the energy consumption of selected devices and maximize the convergence speed of the global model, under the constraints of time consumption and the total amount of data for training. Then, we take advantage of the CMAB learning algorithm to better estimate the CPU-cycle frequency of each device given its uncertainty nature due to multi-task processing at devices, which makes the calculation of the energy consumption more accurate. To the end, we design the FCE^{2}DS algorithm to solve the problem and verify its efficiency and performance through a series of experiments.

References

[1] J. Zhang, B. Chen, Y. Zhao, X. Cheng, and F. Hu, "Data security and privacy-preserving in edge computing paradigm: Survey and open issues," *IEEE access*, vol. 6, pp. 18 209–18 237, 2018.

[2] L. Li, D. Shi, R. Hou, H. Li, M. Pan, and Z. Han, "To talk or to work: Flexible communication compression for energy efficient federated learning over heterogeneous mobile edge devices," in *IEEE INFOCOM 2021-IEEE Conference on Computer Communications*. IEEE, 2021, pp. 1–10.

[3] F. Sattler, S. Wiedemann, K.-R. Müller, and W. Samek, "Robust and communication-efficient federated learning from non-iid data," *IEEE transactions on neural networks and learning systems*, vol. 31, no. 9, pp. 3400–3413, 2019.

[4] A. Taik, Z. Mlika, and S. Cherkaoui, "Data-aware device scheduling for federated edge learning," *arXiv preprint arXiv:2102.09491*, 2021.

[5] D. Ye, S. Chen, and C. Wang, "Fast convergence for federated learning in ofdma systems," in *2021 IEEE 32nd Annual International Symposium on Personal, Indoor and Mobile Radio Communications (PIMRC)*. IEEE, 2021, pp. 1–6.

[6] Q. Zeng, Y. Du, K. Huang, and K. K. Leung, "Energy-efficient radio resource allocation for federated edge learning," in *2020 IEEE International Conference on Communications Workshops (ICC Workshops)*. IEEE, 2020, pp. 1–6.

[7] Y. G. Kim and C.-J. Wu, "Autofl: Enabling heterogeneity-aware energy efficient federated learning," in *MICRO-54: 54th Annual IEEE/ACM International Symposium on Microarchitecture*, 2021, pp. 183–198.

[8] A. Chopra, S. K. Sahu, A. Singh, A. Java, P. Vepakomma, V. Sharma, and R. Raskar, "Adasplit: Adaptive trade-offs for resource-constrained distributed deep learning," *arXiv preprint arXiv:2112.01637*, 2021.

[9] C. Peng, Q. Hu, J. Chen, K. Kang, F. Li, and X. Zou, "Energy-efficient device selection in federated edge learning," in *2021 International Conference on Computer Communications and Networks (ICCCN)*. IEEE, 2021, pp. 1–9.

[10] W. Wu, L. He, W. Lin, R. Mao, C. Huang, and W. Song, "Fedprof: Optimizing federated learning with dynamic data profiling," *arXiv preprint arXiv:2102.01733*, 2021.

[11] W. Zhang, X. Wang, P. Zhou, W. Wu, and X. Zhang, "Client selection for federated learning with non-iid data in mobile edge computing," *IEEE Access*, vol. 9, pp. 24 462–24 474, 2021.

[12] K. Yang, T. Jiang, Y. Shi, and Z. Ding, "Federated learning via over-the-air computation," *IEEE Transactions on Wireless Communications*, vol. 19, no. 3, pp. 2022–2035, 2020.

[13] O. Marnissi, H. E. Hammouti, and E. H. Bergou, "Client selection in federated learning based on gradients importance," *arXiv preprint arXiv:2111.11204*, 2021.

[14] M. Tang, X. Ning, Y. Wang, Y. Wang, and Y. Chen, "Fedgp: Correlation-based active client selection strategy for heterogeneous federated learning," *arXiv preprint arXiv:2103.13822*, 2021.

[15] Y. J. Cho, J. Wang, and G. Joshi, "Client selection in federated learning: Convergence analysis and power-of-choice selection strategies," *arXiv preprint arXiv:2010.01243*, 2020.

[16] Y. J. Cho, S. Gupta, G. Joshi, and O. Yağan, "Bandit-based communication-efficient client selection strategies for federated learning," in *2020 54th Asilomar Conference on Signals, Systems, and Computers*. IEEE, 2020, pp. 1066–1069.

[17] H. Ko, J. Lee, S. Seo, S. Pack, and V. C. Leung, "Joint client selection and bandwidth allocation algorithm for federated learning," *IEEE Transactions on Mobile Computing*, 2021.

[18] R. Arora, O. Dekel, and A. Tewari, "Online bandit learning against an adaptive adversary: from regret to policy regret," *arXiv preprint arXiv:1206.6400*, 2012.

[19] T. Nishio and R. Yonetani, "Client selection for federated learning with heterogeneous resources in mobile edge," in *ICC 2019-2019 IEEE International Conference on Communications (ICC)*. IEEE, 2019, pp. 1–7.

[20] T. D. Burd and R. W. Brodersen, "Processor design for portable systems," *Journal of VLSI signal processing systems for signal, image and video technology*, vol. 13, no. 2, pp. 203–221, 1996.

[21] N. H. Tran, W. Bao, A. Zomaya, N. M. NH, and C. S. Hong, "Federated learning over wireless networks: Optimization model design and analysis," in *IEEE INFOCOM 2019-IEEE Conference on Computer Communications*. IEEE, 2019, pp. 1387–1395.

[22] P. Auer, N. Cesa-Bianchi, and P. Fischer, "Finite-time analysis of the multiarmed bandit problem," *Machine learning*, vol. 47, no. 2, pp. 235–256, 2002.

[23] A. Sengupta, S. Amuru, R. Tandon, R. M. Buehrer, and T. C. Clancy, "Learning distributed caching strategies in small cell networks," in *2014 11th International Symposium on Wireless Communications Systems (ISWCS)*. IEEE, 2014, pp. 917–921.

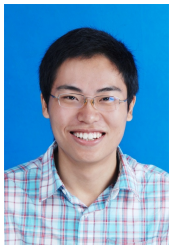
[24] P. Toth, "Dynamic programming algorithms for the zero-one knapsack problem," *Computing*, vol. 25, no. 1, pp. 29–45, 1980.

VII. Biography Section

Cheng Peng received the BS degree in Computer Science and Technology from Beijing University of Technology, Beijing, China, in 2019, and the MS degree in Computer and Information Sciences from Indiana University-Purdue University Indianapolis (IUPUI), Indiana, USA, in 2021. His current research focuses on edge computing, federated learning, and Internet of Things (IoT).

Qin Hu received her Ph.D. degree in Computer Science from the George Washington University in 2019. She is currently an Assistant Professor with the Department of Computer and Information Science, Indiana University-Purdue University Indianapolis (IUPUI). She has served on the Editorial Board of two journals, the Guest Editor of four journals, the TPC/Publicity Co-chair for several workshops/conferences, and the TPC Member for several international conferences. Her research interests include wireless and mobile security, edge computing, and blockchain.

Zhilin Wang received his B.S. from Nanchang University in 2020. He is currently pursuing his Ph.D. degree of Computer and Information Science In Indiana University-Purdue University Indianapolis (IUPUI). He is a Research Assistant with IUPUI, and he is also a reviewer of 2022 IEEE International Conference on Communications (ICC). His research interests include blockchain, federated learning, edge computing, and Internet of Things (IoT).



Ryan Wen Liu (M'15) received the B.Sc. degree (Hons.) in Information and Computing Science from the Department of Mathematics, Wuhan University of Technology, Wuhan, China, in 2009, and the Ph.D. degree from The Chinese University of Hong Kong, Hong Kong, in 2015. He is currently an Associate Professor with the School of Navigation, Wuhan University of Technology. He was a Visiting Scholar with the Agency for Science, Technology and Research, Singapore. His research interests include computer vision, data mining, and intelligent transportation system.

Zehui Xiong is currently an Assistant Professor in the Pillar of Information Systems Technology and Design, Singapore University of Technology and Design. Prior to that, he was a researcher with Alibaba-NTU Joint Research Institute, Singapore. He received the PhD degree in Nanyang Technological University, Singapore. He was the visiting scholar at Princeton University and University of Waterloo. His research interests include wireless communications, network games and economics, blockchain, and edge intelligence. He has published more than 140 research papers in leading journals and flagship conferences and many of them are ESI Highly Cited Papers. He has won over 10 Best Paper Awards in international conferences and is listed in the World's Top 2% Scientists identified by Stanford University. He is now serving as the editor or guest editor for many leading journals including IEEE JSAC, TVT, IoTJ, TCCN, TNSE, ISJ, JAS. He is the recipient of IEEE TCSC Early Career Researcher Award for Excellence in Scalable Computing, IEEE CSIM Technical Committee Best Journal Paper Award, IEEE SPCC Technical Committee Best Paper Award, IEEE VTS Singapore Best Paper Award, Chinese Government Award for Outstanding Students Abroad, and NTU SCSE Best PhD Thesis Runner-Up Award. He is the Founding Vice Chair of Special Interest Group on Wireless Blockchain Networks in IEEE Cognitive Networks Technical Committee.