HELCFL: High-Efficiency and Low-Cost Federated Learning in Heterogeneous Mobile-Edge Computing

Yangguang Cui[†], Kun Cao[‡], Junlong Zhou[§], Tongquan Wei[†]
[†]Shanghai Key Laboratory of Trustworthy Computing, East China Normal University, Shanghai 200062, China
[†]State Key Laboratory of Mathematical Engineering and Advanced Computing, Wuxi 214125, China
[‡]College of Information Science and Technology, Jinan University, Guangzhou 510632, China
[§]School of Computer Science and Engineering, Nanjing University of Science and Technology, Nanjing 210094, China

***Abstract*—Federated Learning (FL), an emerging distributed machine learning (ML), empowers a large number of embedded devices (e.g., phones and cameras) and a server to jointly train a global ML model without centralizing user private data on a server. However, when deploying FL in a mobile-edge computing (MEC) system, restricted communication resources of the MEC system, heterogeneity and constrained energy of user devices have a severe impact on FL training efficiency. To address these issues, in this article, we design a distinctive FL framework, called HELCFL, to achieve high-efficiency and low-cost FL training. Specifically, by analyzing the theoretical foundation of FL, our HELCFL first develops a utility-driven and greedy-decay user selection strategy to enhance FL performance and reduce training delay. Subsequently, by analyzing and utilizing the slack time in FL training, our HELCFL introduces a device operating frequency determination approach to reduce training energy costs. Experiments verify that our HELCFL can enhance the highest accuracy by up to 43.45%, realize the training speedup of up to 275.03%, and save up to 58.25% training energy costs compared to state-of-the-art baselines.**

***Index Terms*—Federated learning, mobile-edge computing, user selection, frequency determination, high efficiency, low costs.**

I. Introduction

Nowadays, numerous embedded devices (e.g., phones and cameras) generate massive data which need to be fed to machine learning (ML) models for exploiting these data. The traditional ML training mode requires users to upload their generated data to a data platform (e.g., a server) for centralized training. However, due to privacy concerns, users are unwilling to upload their generated data to a server in practice, which makes it infeasible to accomplish ML centralized training [1].

To deal with this dilemma, an effective solution is federated learning (FL) [2]. FL, an emerging ML architecture, is capable of accomplishing distributed training across numerous users without sharing their own data to external entities. Considering a standard FL prototype in a mobile-edge computing (MEC) system, distributed training of FL works iteratively to obtain a global ML model with expected accuracy. In one training iteration, an edge server first broadcasts a global ML model to some users with a certain ratio. Then, these users conduct ML model updates on their own data and upload their updated models to the edge server. Finally, by integrating the uploaded models, this server creates an updated global ML model for the next iteration. Recently, FL has gained widespread research interest due to its distinctiveness in protecting data privacy [3].

However, when FL is coupled into a real MEC system, the training performance of FL is restricted by multiple actual challenges. The first challenge is insufficient communication resources of the MEC system [4]. Although no raw data are uploaded in FL training, instead, numerous ML models containing millions of parameters are conveyed frequently. Oftentimes, communication resources of the MEC system are insufficient to afford communication costs of a lot of users in FL training. To tackle this dilemma, many schemes have been developed to reduce FL communication costs from the perspective of model compression, such as sparseness [5], quantization [6], etc. Nevertheless, these schemes inevitably sacrifice model accuracy or introduce additional compression costs on individual users. Apart from model compression, researchers have designed some schemes to decrease communication costs at the expense of user computing costs. Typically, Wang et al. [7] presented an adaptive control technique that adjusts the number of local training iterations on users under the specific communication cost constraints. However, this scheme may induce long training delays and high computing energy costs on individual users.

For a FL system across a large number of users, an effective solution that can adapt to insufficient communication resources is to develop an efficient user selection strategy [8]. In a standard FL prototype, a random user selection approach with a specific ratio is provided to accommodate insufficient communication resources [9]. In this context, in each FL training iteration, following the synchronization rule of FL training, the selected user with the longest training delay decides the training delay of this iteration. Hence, another tricky bottleneck of FL coupled into a MEC system is user heterogeneity [3], [8]. Representatively, to deal with the dilemma of inefficient FL training caused by user heterogeneity, Nishio et al. [10] presented a greedy-enabled user selection strategy that always selects users with short training delays to participate in FL training. However, their greedy-enabled heuristic strategy re-

This work was partially supported by National Key Research and Development Program of China under Grant 2018YF B2101300, Shanghai Trusted Industry Internet Software Collaborative Innovation Center, the Open Project Program of the State Key Laboratory of Mathematical Engineering and Advanced Computing, the NSFC under Grants 62102164, 62172224, and 61802185, the China Postdoctoral Science Foundation under Grants 2021T140272 and 2021M691240, the Fundamental Research Funds for the Central Universities under Grant 21621025, and the Open Research Fund of the National Trusted Embedded Software Engineering Technology Research Center (East China Normal University). T. Wei is the corresponding author: tqwei@cs.ecnu.edu.cn.

stricts FL training from to gain the expected accuracy due to missing data samples on users with long training delays.

In addition, it is well known that ML model updates and parameter uploads of FL training on individual users are energy-intensive tasks. However, most of user devices are powered by batteries, thus, their energy is constrained in practice [11]. Therefore, the energy of user devices is quickly exhausted or even device shutdown occurs during FL training. Due to the above facts, there is an urgent need for an effective energy optimization mechanism to optimize user energy costs of FL training, thereby motivating users to participate in FL training more actively. By analyzing the energy and delay costs of FL training, Tran et al. [12] formulated the training costs of FL as an optimization problem, and offered a closed-form approach to balance training delay and energy costs. Nevertheless, the effect of user selection scheme on FL training efficiency has not been taken into account in [12]. Considering communication and computing costs of users, Wang et al. [13] developed a support vector machine-enabled user association and resource allocation scheme to implement energy-efficient FL training. However, user heterogeneity is ignored in [13].

This article makes the following outstanding contributions.

- To make FL well coupled into a MEC system, we propose a high-efficiency and low-cost FL framework, called HELCFL, which widely takes into account user heterogeneity, training delay, and training energy costs.
- Based on the analysis on the theoretical foundation of FL, our HELCFL first proposes a utility-driven and greedy-decay heuristic user selection strategy to address user heterogeneity issues and enhance FL training performance.
- By analyzing slack time in FL training, our HELCFL then develops a device operating frequency determination approach that exploits slack time to optimize training energy costs without degrading FL training performance.
- Experiments verify that our HELCFL can enhance up to 43.45% model accuracy, gain up to 275.03% training acceleration, and save up to 58.25% training energy costs compared to state-of-the-art baselines.

II. FL Prototype and Associated Models

A. FL Architecture in Mobile-Edge Computing

Consider a FL architecture in a mobile-edge computing (MEC) environment, in which a FL central controller (FLCC) cooperates with Q user devices $V = \{v_1, \cdots, v_Q\}$ to conduct FL training. Note that the FLCC is composed of a base station and an edge server. Each user $v_q (1 \leq q \leq Q)$ possesses a local dataset $D_q = \{\mathbf{s}_{q,b}, \mathbf{y}_{q,b}\}_{b=1}^{|D_q|}$ of size $|D_q|$, where $\mathbf{s}_{q,b}$ and $\mathbf{y}_{q,b}$ denote the bth data sample and its sample label on user v_q, respectively. Accordingly, the total dataset D_{total} of all users for FL training can be given as $D_{total} = \bigcup_{v_q \in V} D_q$.

Let M_q denote the ML model of each user v_q. Here, the target of local ML training can be defined as [14]

$$min : \{L_q(M_q, D_q) = \frac{1}{|D_q|} \sum_{b=1}^{|D_q|} l(M_q, \mathbf{s}_{q,b}, \mathbf{y}_{q,b})\}, \quad (1)$$

where $l(M_q, \mathbf{s}_{q,b}, \mathbf{y}_{q,b})$ denotes the loss function of the model M_q on the data sample $\{\mathbf{s}_{q,b}, \mathbf{y}_{q,b}\}$. Through the coordination between users and the FLCC, the target of FL is to obtain a global ML model M_G that can minimize the loss function $L_G(M_G, D_{tatal})$ on the total dataset D_{total} [15], that is,

$$min : \{L_G(M_G, D_{tatal}) = \frac{\sum_{q=1}^{Q} \sum_{b=1}^{|D_q|} l(M_q, \mathbf{s}_{q,b}, \mathbf{y}_{q,b})}{|D_{tatal}|}\}. \quad (2)$$

B. Local User Calculation Models

The well-known gradient descent (GD) algorithm, in this article, is employed to update the parameters of the local user model. Specifically, for each user v_q, its local model update in the jth training iteration can be calculated as [15]

$$M_q^{j+1} = M_q^j - \frac{\tau}{|D_q|} \sum_{b=1}^{|D_q|} \nabla l(M_q^j, \mathbf{s}_{q,b}, \mathbf{y}_{q,b}), \quad (3)$$

where τ denotes the learning rate.

In this context, the calculation delay and calculation energy costs for user model update can be outlined as follows. Let π represent the number of CPU cycles for each user to process a single data sample. This work considers Q users devices enabled by dynamic voltage/frequency scaling (DVFS). For each user v_q, its enabled CPU frequency f_q is taken from its CPU frequency range $[f_q^{\min}, f_q^{\max}]$. Note that the difference in the operating frequency ranges of user devices reflects their calculation heterogeneity. Here, for each user v_q, its calculation delay for model update can be defined as [12]

$$T_q^{cal} = \frac{\pi |D_q|}{f_q}, \quad (4)$$

where $\pi |D_q|$ is the total CPU cycles for user v_q to process its dataset, and f_q is the CPU operating frequency of user v_q.

Then, the energy costs E_q^{cal} of each user v_q introduced by the local model update can be calculated as [12]

$$E_q^{cal} = \frac{\alpha}{2} \pi |D_q| f_q^2, \quad (5)$$

where $\frac{\alpha}{2}$ indicates the effective switched capacitance which is associated with the chip architecture.

C. Local User Communication Models

In this article, we consider a time-division multiple access (TDMA) FL framework in the MEC system, and this MEC system has a total of Z resource blocks (RBs). Let p_q represent the transmission power of each user v_q, and the data upload rate that each user v_q can achieve can be defined as [16]

$$R_q = Z \log_2 (1 + \frac{p_q h_q^2}{N_0}), \quad (6)$$

where h_q and N_0 indicate the channel gain of this user and background noise, respectively. In this situation, the time duration T_q^{com} required for each user v_q to upload the model parameters can be calculated as [14]

$$T_q^{com} = \frac{C_{model}}{R_q}, \quad (7)$$

where C_{model} denotes the data size of the uploaded model parameters in bits. Accordingly, communication energy consumption E_q^{com} for uploading model parameters to the FLCC within the time period T_q^{com} can be formulated as [14]

$$E_q^{com} = p_q T_q^{com}. \tag{8}$$

In this case, in one training iteration, the total model update and upload delay T_q of each user v_q can be defined as [12]

$$T_q = T_q^{cal} + T_q^{com}. \tag{9}$$

D. Local User Models in One Training Iteration

Following the synchronization rule of FL training and assuming that the selected user set in the jth FL training iteration is Γ_j, the delay of this training iteration can be defined as [10]

$$T_{\Gamma_j} = \max_{v_q \in \Gamma_j} \{T_q^{cal} + T_q^{com}\}. \tag{10}$$

Accordingly, the entire energy consumption E_{Γ_j} of this FL training iteration can be defined as [14]

$$E_{\Gamma_j} = \sum_{v_q \in \Gamma_j} (E_q^{cal} + E_q^{com}). \tag{11}$$

Note that in the considered FL framework, due to the strong computational capacity and unconstrained energy of the edge server, the calculation delay and energy costs of the FLCC in ML model integration and parameter broadcasting are ignored.

III. Problem definition

The target of this article is to optimize FL training loss and total training delay, and simultaneously reduce training energy costs without degrading FL training performance. To be specific, the studied problem can be defined as

$$\text{Minimize: } L_G(M_G, D_{tatal}) \& \sum_{j=1}^{J} T_{\Gamma_j} \& \sum_{j=1}^{J} E_{\Gamma_j}$$

$$\text{Subject to: } \Gamma_j \subset V, \tag{12}$$

$$T_{\Gamma_j} \geq \max_{v_q \in \Gamma_j} \{T_q^{cal} + T_q^{com}\}, \tag{13}$$

$$\sum_{j=1}^{J} T_{\Gamma_j} \leq Deadline, \tag{14}$$

$$f_q^{\min} \leq f_q \leq f_q^{\max}, \quad v_q \in V, \tag{15}$$

where J denotes the given maximum number of FL training iterations. In the defined problem, constraints (12) and (13) guarantee that the selected user set in each FL training iteration is a subset of all users and FL training is synchronized. Constraint (14) ensures that the total FL training is terminated before the deadline. Constraint (15) defines the operating frequency range of each user.

IV. Our high-efficiency and low-cost FL

We propose a high-efficiency and low-cost FL framework, called HELCFL, in a MEC environment. Our HELCFL's 2-phase workflow, including the initialization phase and iterative training phase, is outlined as **Algorithm 1**.

Algorithm 1: Proposed high-efficiency and low-cost FL

Input: The FL central controller (FLCC), heterogeneous user set V of size Q, user selection fraction C;
Output: The final trained global ML model;
// initialization
1 FLCC broadcasts resource information requests to all users;
2 All users inform the FLCC of their resource information;
// FL iterative training
3 **for** $j \leftarrow 1, 2, 3, \cdots J-1$ **do**
4 Call **Algorithm 2** to determine the selected user set Γ_j and CPU operating frequency set F_{Γ_j} of this selected user set;
5 FLCC broadcasts its global ML model M_G^j to selected users;
6 **for** *Each selected user* v_q *in* Γ_j ***in parallel*** **do**
7 Each selected user v_q conducts its local model update at the determined operating frequency f_q according to Eq. (3);
8 Each selected user v_q uploads the updated model M_q^{j+1} to the FLCC in turn by using available Z RBs;
9 **end**
10 FLCC integrates the conveyed models to create a new global model M_G^{j+1} as $M_G^{j+1} = \frac{\sum_{v_q \in \Gamma_j} |D_q| M_q^{j+1}}{\sum_{v_q \in \Gamma_j} |D_q|}$;
11 **end**
12 **Return** the final global ML model M_G^J;

In the initialization phase, the FLCC first broadcasts resource information requests to all users in line 1, and then all users inform the FLCC of their resource information in line 2 (e.g., CPU frequency range and transmission power, etc.). In each iteration of the FL iterative training phase (lines 3-11), based on the informed resource information and training history records, **Algorithm 2** is called on the FLCC to determine the selected users in this training iteration and the CPU operating frequencies of these selected users (line 4). Then, the FLCC broadcasts its global ML model to these selected users (line 5). Afterwards, these selected users carry out their local model updates in parallel at the determined CPU operating frequencies, and then sequentially upload their updated models to the FLCC by using available RBs (lines 6-9). Finally, following the *FedAvg* formula in [9], the FLCC integrates the conveyed models to create a new global model (line 10). After the above workflow of each iteration, the FLCC checks whether this newly created global ML model converges in this iteration or the total training delay exceeds the deadline. If so, the training exits; otherwise, it returns to the second phase (line 3) for the next iteration.

Clearly, as outlined in **Algorithm 1**, the key part of our HELCFL framework is to select users for each FL iteration and determine CPU frequencies of these selected user devices, which dominates the FL training delay and energy costs. In the subsequent two sections, we describe in detail our user selection and CPU frequency determination solutions.

V. Utility-driven heuristic user selection

In this section, we first give an in-depth analysis on theoretical foundation of FL. Based on this analysis, we then design a utility-driven and greedy-decay user selection solution.

A. Analysis on Theoretical Foundation of FL

Here, we first outline the motivation and inspiration of our heuristic user selection solution. It is well known that the user selection problem of FL training is an NP-hard problem

[16]. Recently, to handle the dilemma of inefficient FL training caused by user heterogeneity, a highly-impacting literature [10] introduced a greedy-enabled user selection approach in which their greedy approach always selects users with short training delays to participate in FL training. We observe that although their approach enables FL training to quickly achieve low-standard accuracy, this approach restricts FL training from achieving high accuracy. Note that the corresponding experimental results in Section VII will prove our observation.

Then, to deal with this shortcoming in [10], here we conduct an in-depth analysis on the theoretical foundation of FL. Specifically, we consider the jth iteration of FL training and assume the selected user set Γ_j in this iteration is denoted as $\Gamma_j = \{v_1, \cdots, v_{|\Gamma_j|}\}$. First, at the beginning of this iteration, the FLCC broadcasts one same global ML model M_G^j to these selected users, thus, each selected user v_q possesses the same ML model M_q^j, that is,

$$M_1^j = M_2^j = \cdots M_{|\Gamma_j|}^j = M_G^j. \tag{16}$$

Second, by adopting GD algorithm, each selected user v_q conducts the model update on its local dataset D_q, that is,

$$M_q^{j+1} = M_q^j - \frac{\tau}{|D_q|} \sum_{b=1}^{|D_q|} \nabla l(M_q^j, \mathbf{s}_{q,b}, \mathbf{y}_{q,b}). \tag{17}$$

Third, the FLCC integrates these model parameters updated by selected users to create a new global ML model M_G^{j+1}. Following the *FedAvg* formula in [9], the model integration on the FLCC can be calculated as

$$M_G^{j+1} = \frac{\sum_{v_q \in \Gamma_j} |D_q| M_q^{j+1}}{\sum_{v_q \in \Gamma_j} |D_q|}. \tag{18}$$

Put Eq. (16) and Eq. (17) into Eq. (18), then Eq. (18) is reformulated as

$$\begin{aligned} M_G^{j+1} &= \frac{\sum_{v_q \in \Gamma_j} |D_q| M_q^{j+1}}{\sum_{v_q \in \Gamma_j} |D_q|}, \\ &= \frac{\sum_{v_q \in \Gamma_j} |D_q| (M_q^j - \frac{\tau}{|D_q|} \sum_{b=1}^{|D_q|} \nabla l(M_q^j, \mathbf{s}_{q,b}, \mathbf{y}_{q,b}))}{\sum_{v_q \in \Gamma_j} |D_q|}, \\ &= M_G^j - \frac{\tau \sum_{v_q \in \Gamma_j} (\sum_{b=1}^{|D_q|} \nabla l(M_G^j, \mathbf{s}_{q,b}, \mathbf{y}_{q,b}))}{\sum_{v_q \in \Gamma_j} |D_q|}, \\ &= M_G^j - \frac{\tau}{|D_{\Gamma_j}|} \sum_{b=1}^{|D_{\Gamma_j}|} \nabla l(M_G^j, \mathbf{s}_{q,b}, \mathbf{y}_{q,b}), \end{aligned} \tag{19}$$

where D_{Γ_j} ($D_{\Gamma_j} = \bigcup_{v_q \in \Gamma_j} D_q$) denotes the total dataset held by these selected users. Here, we can intuitively observe from Eq. (19) that the model update of this FL training iteration is essentially equivalent to conducting the centralized mini-batch SGD model update on the total dataset D_{Γ_j} held by the selected user set Γ_j. Therefore, we can confidently conclude: the reason why the traditional greedy approach in [10] cannot achieve high accuracy is that the data, on users with long training delays, are not incorporated within FL training.

Based on the above analysis on theoretical foundation of FL, we newly define a utility function u_q for each user v_q, that is,

$$u_q(\alpha_q, T_q^{cal}, T_q^{com}) = \eta^{\alpha_q} \times \frac{1}{(T_q^{cal} + T_q^{com})}, \tag{20}$$

where $\eta (0 < \eta < 1)$ denotes the decay coefficient and the appearance counter α_q counts the number of times that each user v_q appears in training iterations within overall FL training.

B. Utility-Driven and Greedy-Decay User Selection

Algorithm 2: Greedy-decay heuristic user selection

Input: The FLCC, heterogeneous user set V of size Q, user selection fraction C, training iteration index j, decay coefficient η;
Output: The selected user set Γ_j in this jth training iteration and operating frequency set F_{Γ_j} of this selected user set;

1 **if** $j == 1$ **then**
2 **for** $q \leftarrow 1$ *to* Q **do**
3 Derive model update delay T_q^{cal} of each user v_q at its maximum operating frequency $f_q^{\max}$ by using Eq. (4);
4 Derive transmission delay T_q^{com} of user v_q by using Eq. (7);
5 $\alpha_q \leftarrow 0$; `// Initialize the appearance counter`
6 **end**
7 **end**
8 **for** $q \leftarrow 1$ *to* Q **do**
9 Derive the utility u_q of user v_q by using Eq.(20);
10 **end**
11 $N \leftarrow \max(Q \cdot C, 1)$;
12 Initialize the selectable user set V' to V;
13 Initialize the selected user set Γ_j to $\emptyset$;
`// Selecting users`
14 **while** $N > 0$ **do**
15 $v_x \leftarrow \arg\max_{v_x \in V'} u_x(\alpha_x, T_x^{cal}, T_x^{com})$;
16 Delete user v_x from V', and add user v_x to Γ_j;
17 $N \leftarrow N - 1$;
18 $\alpha_x \leftarrow \alpha_x + 1$; `// Selected user's utility decay`
19 **end**
20 Call **Algorithm 3** to determine operating frequency set F_{Γ_j} of this selected user set Γ_j;
21 **Return** The selected user set Γ_j and its operating frequency set F_{Γ_j};

Based on our defined utility function derived from the theoretical foundation of FL, we design a greedy-decay heuristic user selection approach, as shown in **Algorithm 2**. For each user v_q, lines 1-7 first derive its training calculation delay T_q^{cal} at its maximum CPU frequency $f_q^{\max}$ and model upload delay T_q^{com}, and then initialize its appearance counter α_q to 0 in the first iteration. Lines 8-10 calculate the utility of each user and line 11 sets the number of selected users as N. Lines 12-13 initialize the selectable user set and the selected user set. Lines 14-19 greedily select the top N users with the largest utility to construct the selected user set. It is worth highlighting that line 18 updates the appearance counter of each selected user to decay its utility according to Eq. (20). In line 20, **Algorithm 3** is called to determine the CPU operating frequencies of these selected users. Line 21 returns the selected user set and its CPU operating frequency set of this training iteration.

Clearly, following the workflow of the above algorithm, our defined utility function can enable users with short training delays to be selected preferentially and also can incorporate users with long training delays within FL training, thereby conducting high-efficiency FL training.

VI. DVFS-enabled energy optimization in FL

In this section, we first introduce the fact that there is unnecessary energy waste in the traditional TDMA FL training. Then, we design a device operating frequency determination algorithm to avoid this unnecessary energy waste.

A. Observation on Energy Waste of Traditional FL

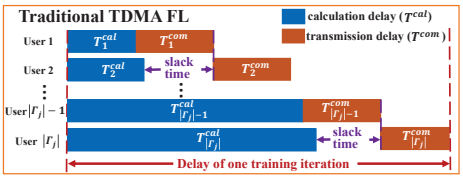

Fig. 1. Illustration of energy waste in traditional TDMA FL.

Assuming that users are at their maximum CPU frequency settings, the calculation delays of different users are different due to the heterogeneity of their computing capabilities. Based on the workflow of traditional TDMA FL, Fig. 1 shows an intuitive illustration of energy waste in this situation. As described in the figure, in one training iteration, if there is no user uploading the model when a user (e.g., user 1) completes its local model update, the FLCC assigns the time duration (e.g., T_1^{com}) to this user for uploading its model. Within this time duration (e.g., T_1^{com}), if a new user (e.g., user 2) completes the local model update, this new user (e.g., user 2) must stop and wait for the previous user (e.g., user 1) to finish uploading before starting to convey its model. Intuitively, there is the slack time for these users (e.g., user 2) who need to stop and wait to upload. Hence, these users do not need to conduct the local model update at their maximum operating frequencies and they only need to complete the local model update at the same time the previous users finish the model upload.

B. DVFS-Enabled Energy Optimization of FL training

Algorithm 3: Determine CPU operating frequencies of selected users

Input: The selected user set Γ_j in the jth training iteration;
Output: The CPU frequency set F_{Γ_j} of this selected user set;
1 Sort the selected user set Γ_j in ascending order the model update delay at their maximum CPU operating frequencies;
2 **for** $q \leftarrow 1$ *to* $|\Gamma_j|$ **do**
3 **if** $q == 1$ **then**
4 $f_q \leftarrow f_q^{\max}$; `// Set CPU frequency of 1st user`
5 **else if** $q == |\Gamma_j|$ **then**
6 **Break**;
7 **end**
  `// Determine the CPU frequency of next user`
8 Derive the total model update and upload delay T_q^j of the user v_q at the determined CPU frequency f_q by using Eq. (9);
9 Based on Eq. (4), derive the CPU frequency of next user v_{q+1} as $f_{q+1} = \frac{\pi |D_{q+1}|}{T_q^j}$;
10 **end**
11 **Return** The determined CPU frequency set F_{Γ_j} of the selected user set;

Based on the observation in the previous section, we design a device operating frequency determination algorithm by using dynamic voltage/frequency scaling (DVFS) method. The key idea of our algorithm is to utilize the slack time of selected users to adjust their operating frequencies, thereby avoiding unnecessary energy waste. The algorithm first sorts the selected users in ascending order the model update delay at their maximum CPU frequencies in line 1. Then, lines 2-10 sequentially determine the CPU frequencies of the selected users. Since there is no slack time, lines 3-4 set the CPU frequency of the first user as its maximum CPU frequency. Lines 5-7 check if the current user is the last selected user; if so, the algorithm terminates. Based on the operating frequency f_q of the current user v_q, line 8 first calculates the total model update and upload delay T_q^j of the current user v_q. According to the calculated delay T_q^j, line 9 then determines the CPU frequency f_{q+1} of the next user v_{q+1} by utilizing slack time. Line 11 returns determined CPU frequencies for selected users.

VII. Evaluation Results and Analysis

In this section, we first display simulation settings, and then evaluate our HELCFL framework.

A. Experimental Settings

This work considers a FL framework which consists of a FL central controller and 100 user devices in a MEC system [17]. The highest CPU operating frequencies of users are distributed at intervals (0.3, 2.0)GHz, and the lowest CPU frequency is 0.3GHz [12]. The effective switched capacitance α=2×10^{28} [12] and training parameter π=1×10^7. The total RBs of this MEC system is 2MHz [18]. The transmission power is deemed to be 0.2W [12]. The user selection fraction $C = 0.1$ [9]. For the ML model, the excellent SqueezeNet [19] is used for FL training. The common CIFAR-10 dataset is used to evaluate, and both the IID and Non-IID settings of CIFAR-10 dataset are considered [9], [10]. In the IID setting, training samples are randomly shuffled and evenly assigned to users. In the Non-IID setting, training samples are sorted by labels and cut into 400 pieces, and each four pieces are assigned a user.

Four state-of-the-art baselines are adopted for comparison. (1) **Classic FL** [9] denotes the classic FL scheme which randomly selects $100 \times C$ users in each iteration. (2) Constrained by the given deadline in each FL iteration, **FedCS** [10] always greedily selects a higher number of users with short training delays. (3) **FEDL** [12] analyzes the delay and energy costs of FL training and offers a closed-form scheme to balance delay and energy costs. (4) **SL** [4] is a separated learning method in which each user conducts its model update separately.

B. Results on Highest Accuracy

For the IID and non-IID settings of the CIFAR-10 dataset, Fig. 2 depicts the accuracy curves of our HELCFL and four baselines during the FL training iteration period. Here, the

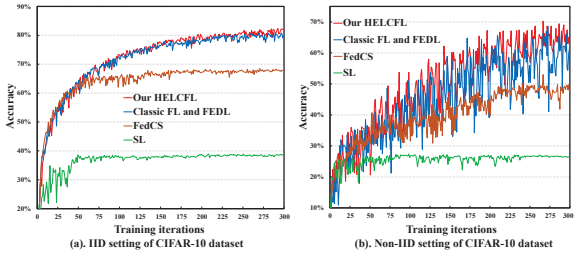

Fig. 2. Accuracy comparison of our HELCFL and 4 baselines.

maximum number of FL training iterations is given as 300. Clearly, in both the IID and non-IID settings, accuracy curves of our HELCFL shows superior performance compared to baselines. This is because our HELCFL ensures that all users are incorporated in FL training. Note that since FEDL takes the same user selection method as Classic FL, FEDL and Classic FL have equivalent accuracy curves following the *FedAvg* formula in [9]. Specifically, compared with Classic FL, FedCS, FEDL, and SL, our HELCFL can enhance 1.49%, 13.9%, 1.49%, and 43.45% accuracy in IID setting respectively and can enhance 1.65%, 17.24%, 1.65%, and 42.83% accuracy in Non-IID setting respectively.

C. Results on Training Delay to Obtain Desired Accuracy

TABLE I
COMPARE TRAINING DELAY TO OBTAIN DESIRED ACCURACY.

Desired accuracy in IID setting		60%	70%	80%
IID setting	Delay of our HELCFL	$6.82min$	**14.17**min	**36.48**min
	Delay of Classic FL	$10.31min$	$22.64min$	$62.89min$
	Delay of FedCS	**6.17**min	✗	✗
	Delay of FEDL	$9.86\ min$	$21.13min$	$59.26min$
	Delay of SL	✗	✗	✗
Desired accuracy in Non-IID setting		40%	50%	60%
Non-IID setting	Delay of our HELCFL	$9.79min$	**15.22**min	**25.34**min
	Delay of Classic FL	$16.28min$	$27.53min$	$48.23min$
	Delay of FedCS	**9.21**min	$41.86min$	✗
	Delay of FEDL	$14.97min$	$26.29min$	$45.65min$
	Delay of SL	✗	✗	✗

We also verify the performance of our HELCFL in terms of training delay to obtain desired accuracy, as shown in Table I. Here, the desired accuracy is set to 60%, 70%, 80% in IID setting and 40%, 50%, 60% in Non-IID setting respectively, and the notation ✗indicates that this scheme cannot achieve the given desired accuracy. For example, when the desired accuracy is specified as 80% in IID setting, FedCS and SL are unable to obtain 80% accuracy, and our HELCFL achieves speedups of 172.40% and 162.45% compared to Classic FL and FEDL. Besides, when the desired accuracy is specified as 60% in Non-IID setting, SL is unable to obtain this accuracy, and our HELCFL achieves speedups of 275.03%, 180.88%, and 173.73% compared to FedCS, Classic FL, and FEDL.

Besides, it is observed from Table I that the training delay of FedCS is shorter than that of our HELCFL when the desired accuracy is specified as 60% in Non-IID setting and 40% in Non-IID setting. The reason is that FedCS always greedily selects users with short training delays. However, FedCS cannot incorporate the data, on users with long training delays, within training, thus, it fails to gain higher accuracy. This fact is consistent with the theoretical analysis in subsection V-A.

D. Results on Energy Optimization

By utilizing slack time in FL training, Fig. 3 shows the energy reduction brought by our designed device operating frequency determination scheme. Clearly, in both IID and non-IID settings, our DVFS-enabled frequency determination method can significantly reduce training energy costs, as shown in Fig. 3. For instance, when the desired accuracy is set to 60% in IID setting, the training energy costs can be reduced by 58.25% via our DVFS-enabled solution.

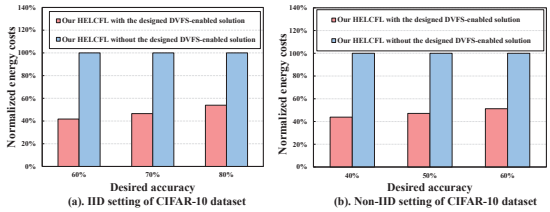

Fig. 3. Energy cost reduction via our DVFS-enabled solution.

VIII. CONCLUSIONS

We, in this article, aim to optimize FL training performance and energy costs for enabling emerging FL to be well coupled into MEC systems. To address user heterogeneity issues, we first design a utility-driven and greedy-decay heuristic user selection method based on the theoretical foundation of FL. With ensuring FL training performance, we then propose a device operating frequency determination approach to optimize training energy costs. Experiments verify that our work can obtain remarkable improvements in terms of model performance, training delay, and energy costs of FL training.

REFERENCES

[1] S. AbdulRahman, H. Tout, H. Ould-Slimane, A. Mourad, C. Talhi, and M. Guizani, "A Survey on Federated Learning: The Journey From Centralized to Distributed On-Site Learning and Beyond," *IEEE JIOT*, vol. 8, no. 7, pp. 5476-5497, 2021.

[2] X. Yin, Y. Zhu, and J. Hu, "A Comprehensive Survey of Privacy-preserving Federated Learning: A Taxonomy, Review, and Future Directions," *ACM CSUR*, vol. 54, no. 6, pp. 1-36, 2021.

[3] A. Imteaj, U. Thakker, S. Wang, J. Li, and M. H. Amini, "A Survey on Federated Learning for Resource-Constrained IoT Devices," *IEEE JIOT*, 2021.

[4] J. H. Ahn, O. Simeone, and J. Kang, "Wireless Federated Distillation for Distributed Edge Learning with Heterogeneous Data," *PIMRC*, pp. 1-6, 2019.

[5] F. Sattler, S. Wiedemann, K. R. Mller, and W. Samek, "Robust and Communication-Efficient Federated Learning From Non-i.i.d. Data," *IEEE TNNLS*, vol. 31, no. 9, pp. 3400-3413, 2019.

[6] N. Shlezinger, M. Chen, Y. C. Eldar, H. V. Poor, and S. Cui, "UVeQFed: Universal Vector Quantization for Federated Learning," *IEEE TSP*, vol. 69, pp. 500-514, 2020.

[7] S. Wang, T. Tuor, T. Salonidis, K. K. Leung, C. Makaya, T. He, and K. Chan, "Adaptive Federated Learning in Resource Constrained Edge Computing Systems," *IEEE JSAC*, vol. 37, no. 6, pp. 1205-1221, 2019.

[8] L. U. Khan, W. Saad, Z. Han, E. Hossain, and C. S. Hong, "Federated Learning for Internet of Things: Recent Advances, Taxonomy, and Open Challenges," *IEEE COMST*, vol. 23, no. 3, pp. 1759 - 1799, 2021.

[9] H. B. McMahan, E. Moore, D. Ramage, S. Hampson, and B. A. Y. Arcas, "Communication-Efficient Learning of Deep Networks from Decentralized Data," *AISTATS*, pp. 1273-1282, 2017.

[10] T. Nishio, and R. Yonetani, "Client Selection for Federated Learning with Heterogeneous Resources in Mobile Edge," *ICC*, pp. 1-7, 2019.

[11] P. Cong, J. Zhou, L. Li, K. Cao, T. Wei, and K. Li, "A Survey of Hierarchical Energy Optimization for Mobile Edge Computing: A Perspective from End Devices to the Cloud," *ACM CSUR*, vol. 53, no. 2, pp. 1-44, 2020.

[12] N. H. Tran, W. Bao, A. Zomaya, M. N. Nguyen, and C. S. Hong, "Federated Learning over Wireless Networks: Optimization Model Design and Analysis," *INFOCOM*, pp. 1387-1395, 2019.

[13] S. Wang, M. Chen, W. Saad, and C. Yin, "Federated Learning for Energy-Efficient Task Computing in Wireless Networks," *ICC*, pp. 1-6, 2020.

[14] Y. Zhan, P. Li, W. Leijie, and S. Guo, "L4L: Experience-Driven Computational Resource Control in Federated Learning," *IEEE TC*, 2021.

[15] M. Chen, N. Shlezinger, H. V. Poor, Y. C. Eldar, and S. Cui, "Joint Resource Management and Model Compression for Wireless Federated Learning," *ICC*, pp. 1-6, 2021.

[16] W. Shi, S. Zhou, and Z. Niu, "Device Scheduling with Fast Convergence for Wireless Federated Learning," *ICC*, pp. 1-6, 2020.

[17] H. Wang, Z. Kaplan, D. Niu, and B. Li, "Optimizing Federated Learning on Non-IID Data with Reinforcement Learning," *INFOCOM*, pp. 1698-1707, 2020.

[18] G. Reus-Muns, and K. Chowdhury, "Classifying UAVs with Proprietary Waveforms via Preamble Feature Extraction and Federated Learning," *IEEE TVT*, 2021.

[19] C. Schorn, A. Guntoro, and G. Ascheid, "An Efficient Bit-Flip Resilience Optimization Method for Deep Neural Networks," *DATE*, pp. 1507-1512, 2019.