Local Performance Trade-Off in Heterogeneous Federated Learning with Dynamic Client Grouping

Yingchi Mao[a, b], Jun Wu[b], Yangkun Cheng[b], Ping Ping[a, b], and Jie Wu[c]
[a] Key Laboratory of Water Big Data Technology of Ministry of Water Resources, Hohai University, Nanjing, China
[b] School of Computer and Information, Hohai University, Nanjing, China
[c] Center of Networked Computing, Temple University, Philadelphia, USA
yingchimao@hhu.edu.cn, 1606010225@hhu.edu.cn, verne_kon@foxmail.com
pingpingnjust@163.com, jiewu@temple.edu

***Abstract*—As an emerging machine learning paradigm, federated learning typically employs client selection to reduce computational and communication overheads of the learning process in mobile edge environment. Although client selection can coordinate large-scale clients for efficient training, the inherent characteristic of the scheme, i.e., the partial client participation can lead to a performance bias of the global model among clients. Besides, the system heterogeneity and data heterogeneity in mobile edge environment can exacerbate the negative impact of such bias on federated learning. To improve unbalanced local performances on different clients of the global model caused by client selection, this paper proposes a Federated Learning Clients Dynamic Grouping Method based on Local Computational Efficiency (FedGLCE). Considering the differences of computational power between clients and combining the idea of grouping clients, FedGLCE separates the framework of federated learning into several client groups, each with cohesive local computing efficiency, to balance the client participation in local training while guaranteeing that clients are not discarded. Specifically, FedGLCE calculates the local computational efficiency of clients based on the number of local data and updated local training time. Then, to complete client grouping, a polynomial distribution set is constructed and mapped to different groups, and one client is randomly selected from each group to participate in the local training. Compared with state-of-the-art client grouping approaches, FedGLCE effectively balances the participation of clients and improves the performance bias of the global model among clients under evaluations on MNIST-Fed and CIFAR-10-Fed heterogeneous datasets in federated learning.**

***Index Terms*—federated learning, computational efficiency, client grouping, client participation.**

I. Introduction

The development of IoT technology and the popularity of mobile edge devices have provided a plethora of data sources and data types for building complex machine learning models with realistic applications. The data gathered by edge devices is uniformly transferred to a central server for processing and computational [1] in the Mobile Edge Computing (MEC) framework [2]. However, because this centralized learning approach is risky in terms of data privacy [3], storing data and executing local training on mobile edge devices with increasing process capacities, such as drones [4], smart cameras [5], and smartphones [6] is becoming more appealing.

Federated learning [7] is an emerging machine learning paradigm that utilizes the local data and computational power of mobile edge devices (called clients) for training local models. The central server in federated learning aggregates all local models to construct the global model while maintaining the security constraint of local data storage. In resource-constrained mobile edge environment, federated learning frequently employs partial rather than full client participation in training (called client selection) to reduce the computational and communication overhead of the learning process [8]. Client selection has been shown to minimize the communication cost [9] of federated learning. Fig. 1 depicts a schematic diagram of the federated learning process based on client selection.

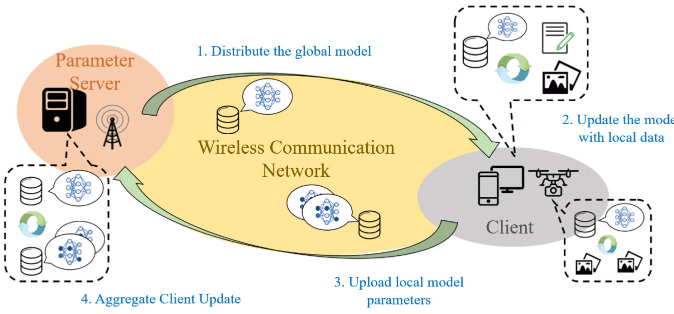
Parameter Server
1. Distribute the global model
2. Update the model with local data
Wireless Communication Network
Client
3. Upload local model parameters
4. Aggregate Client Update

Fig. 1. The federated learning process based on client selection

While client selection scheme in federated learning can coordinate large-scale clients for efficient training, the inherent partial client participation caused by this scheme can lead to performance biases of the global model among clients such as local test accuracy [10] [11] [12]. The severity of such biases is further enhanced in mobile edge environment with high system and data heterogeneity. In terms of system heterogeneity, generic client selection approaches like FedAvg [7], FedProx [13], and FedDrop [14] discard a portion of clients based on the computing efficiency, leaving the global model unable to characterize these clients' local performances. In terms of data heterogeneity, both the random selection strategy and the equal treatment that all clients participate in the training make the global model skewed towards common data distributions and underpowered to characterize particular data distributions. Despite the high overall performance of the global model, clients with weak computing power rarely have the opportunity to participate in federated training, and even if they do, their training results can be discarded by the central server. These clients fail to meet the needs of user-sensitive applications due to the performance bias [15] [16].

Based on the above analysis, the decisive factor for whether a client is discarded or not lies in the computing power of the client, i.e., local computational efficiency. With high local computational efficiency, one client can complete the computation in a short time even if the data volume is large. However, if a client has low local computational efficiency, the client must consume a long computation time even though the data volume is small. To the best of our knowledge, existing client grouping schemes fail to accurately reflect differences in computational efficiency between clients. Furthermore, present approaches only produce results of the static grouping, which are incompatible with mobile edge scenarios where clients' computational capacities fluctuate. Therefore, to enhance the participation of weaker clients (clients with poor computational efficiency) in federated learning, this paper proposes a Federated Learning Clients Dynamic Grouping Method based on Local Computational Efficiency (FedGLCE), starting from the difference in computational power of clients and under the guarantee of convergence provided by the clustering construction of polynomial distribution. Specifically, FedGLCE firstly characterizes the computational power of clients by their local computational efficiency, and constructs a polynomial distribution set for mapping to client groups based on their local computational efficiency. Clients with similar local computational efficiency are divided into the same group, and then randomly select one client from each group to participate in the training. Meanwhile, because the computational power of clients varies in mobile edge environment, clients in FedGLCE dynamically update their local computational efficiency after completing their training. Finally, the central server periodically updates client grouping results in specified iteration rounds according to clients' local computational efficiency. In general, FedGLCE improves the overall participation of weaker clients by grouping them based on their local computational efficiency, and equalizes the performance differences of the global model among clients at the client level.

The main contributions of this paper include,

- The local computational efficiency is employed to characterize the computational power of clients, and a polynomial distribution set is constructed for mapping to groups of clients based on their local computational efficiency. Clients with similar local computational efficiency are divided into the same group.
- One client from each group is randomly selected to participate in the training and dynamically updates its local computational efficiency after completing training. Besides, considering variations in the computational power of clients in the mobile edge environment, the central server adjusts the client grouping results within a specified iteration round.
- Evaluations on several heterogeneous datasets in federated learning demonstrates that FedGLCE outperforms FedAvg, FedDrop, TiFL, and FedSS by about 1.30 times, 1.86 times, 1.49 times, and 1.23 times in client participation variance, respectively, and in local test accuracy variance by about 1.24 times, 1.04 times, 1.40 times and 1.01 times, respectively.

The remainder of this paper is organized as follows. Section II presents the related work of client grouping. The system model of FedGLCE is shown in Section III. In Section IV, FedGLCE is discussed in detail. Then, FedGLCE is compared with state-of-the-art client grouping approaches in Section V. At last, conclusions are drawn in Section VI.

II. Related Work

Grouping clients prior to federated training is one of the effective approaches to address the performance bias of the global model among clients. The essence of client grouping is to divide clients into different groups based on their own characteristics, with similar client characteristics within the group and distinct client characteristics between groups. After clients are grouped, client selection is performed within groups to guarantee that clients, particularly those who would otherwise be discarded, receive access to the local training.

For example, the FRL framework [17] groups clients based on the similarity of user behavior performance and constructs a secondary global model for each group to receive intra-group updates from clients, reducing the negative impact of low-relevance data samples on the global model's performance. However, FRL cannot specify the evaluation criteria and grouping strategy for the client similarity metric. Ghosh et al. [18] employ the clustering of data distribution to complete client grouping and divide clients with similar local datasets into the same group, thereby alleviating the data heterogeneity problem in federated learning. However, this approach requires clients to share their local metadata with the central server, which is contrary to the nature of privacy protection for federated learning. Fraboni et al. [9] also introduce clustered sampling for client grouping, that is, taking the number of local data samples into account for grouping to achieve a more uniform client selection effect. However, this approach only groups clients during the initialization phase in federated learning, and it is not applicable to the mobile edge environment, where local computational power varies dynamically.

Furthermore, some researchers consider employing the local training time of clients for client grouping. TiFL [19] groups clients based on their local training time to ensure that clients with long training time have the opportunity to participate in training, but TiFL ignores the effect of data samples on training time, resulting in local performance biases of the global model. Unlike TiFL, FedSS [9] examines client training time indirectly from the perspective of local data volume, and groups clients with similar amount of local data into the same client group, allowing clients with long training times, i.e., large data volume to participate in training. However, FedSS also groups clients during the federated learning's

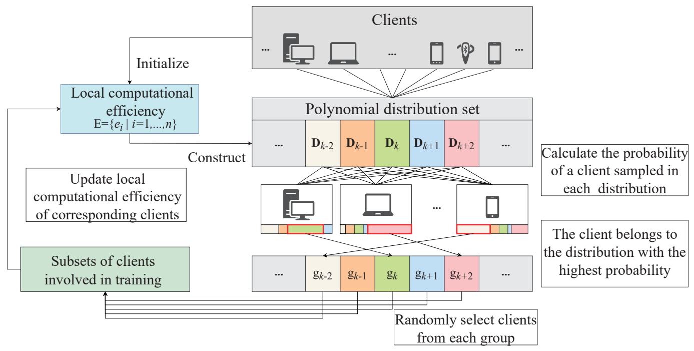
Clients
Initialize
Local computational efficiency E={e_i | i=1,...,n}
Polynomial distribution set
D_{k-2} D_{k-1} D_k D_{k+1} D_{k+2}
Construct
Calculate the probability of a client sampled in each distribution
Update local computational efficiency of corresponding clients
The client belongs to the distribution with the highest probability
Subsets of clients involved in training
g_{k-2} g_{k-1} g_k g_{k+1} g_{k+2}
Randomly select clients from each group

Fig. 2. The framework of FedGLCE

initialization phase. Additionally, if the computational power of clients with large data volume is generally poor, FedSS greatly increases the time cost in federated learning.

III. System model

Fig. 2 depicts the framework of Federated Learning Clients Dynamic Grouping Method based on Local Computational Efficiency (FedGLCE). The clients dynamic grouping process of FedGLCE can be split down into four steps as follows,

(1) Calculate local computational efficiency: Input clients' local data set $\{|d_i||i = 1, ..., n\}$ of length n, the number of client groups m, and obtain clients' local computation efficiency set $E = \{e_i|i = 1, ..., n\}$. Then, sort the set E in descending order.

(2) Construct polynomial distribution set: A polynomial distribution set $\{\mathrm{D}_k^r\}_{k=1}^m$ of length m is built based on the local computational efficiency set E, and the distribution $\{\mathrm{D}_k^r\}_{k=1}^m$ corresponds to the client group g_k one by one.

(3) Group clients: The probability $\{q_{i,k}^r\}_{k=1}^m$ of a client being sampled in distribution $\{\mathrm{D}_k^r\}_{k=1}^m$ is calculated sequentially. If a client has the maximum sampling probability in distribution $\{\mathrm{D}_k^r\}_{k=1}^m$, i.e., $\forall k \in [1, k'-1]\cup[k'+1, m], q_{i,k'}^r > q_{i,k}^r$, then client c_i is allocated to group $g_{k'}^r$. The grouping result is output after all clients have been assigned.

(4) Update local computational efficiency: A random client from each group forms a subset S^r of clients participating in training based on the client grouping. Following the local training phase, the local computational efficiency of clients in S^r is updated using the training time $t_{i'}^r$.

For computational cost considerations, steps (2)(3) are performed every r_u iteration rounds, dynamically updating the grouping results on a regular basis. Steps (1)(4) are executed in each iteration round until the final iteration round comes.

Further, we specify the working mechanism of FedGLCE using the form of variables. Allow n to be the total number of clients in federated learning and define the set of clients as $\{c_i|i = 1, ..., n\}$. e_i represents the local computational efficiency of client c_i for a local dataset d_i with the number of samples $|d_i|$. The client grouping problem is equivalent to assigning n clients to m client groups $\{g_k^r|k = 1, ..., m\}$ $(m < n)$. In the r-th iteration round, m independent polynomial distributions $\{\mathrm{D}_k^r\}_{k=1}^m$ are utilized to map one by one to client groups $\{g_k^r|k = 1, ..., m\}$, and each distribution in $\{\mathrm{D}_k^r\}_{k=1}^m$ offers various weights to n clients based on client's local computational efficiency e_i. Then, the probability $q_{i,k}^r$ of client c_i being sampled in group g_k^r is computed based on the distribution $\{\mathrm{D}_k^r\}_{k=1}^m$, where $q_{i,k}^r \propto e_i$. Consider client c_i to be assigned to group $g_{k'}^r$ when the probability of c_i being sampled in the k'-th group is the highest, i.e., $\forall k \in [1, k'-1] \cup [k'+1, m], q_{i,k'}^r > q_{i,k}^r$. After the client grouping process, each client group selects a client at random to participate in the local training for the iteration round and updates selected clients' local computational efficiency. Since the local computational efficiency changes over time, the set of polynomial distributions $\{\mathrm{D}_k^r\}_{k=1}^m$ utilized to map client grouping results is also dynamic.

A list of main symbolic parameters of the client grouping is offered in Table 1 for a better presentation in this paper.

TABLE I
Main symbolic parameters of the client grouping

Symbol	**Definition**
n	Number of clients
m	Number of client groups
d_i	Local dataset of client c_i
ω_i^r	Local model of client c_i for the r-th iteration
ω^r	Global model for the r-th iteration
g_k^r	The k-th client group of the r-th iteration
$q_{i,k}^r$	Sampling probability of client c_i in D_k^r
r	Iterative round index
c_i	The i-th client
e_i	Local computational efficiency of client c_i
$t_i^{r_{latest}}$	Latest local training time for client c_i
w_i	Aggregation weight of client c_i
D_k^r	The k-th polynomial distribution for the r-th iteration
S^r	Subset of clients in the r-th iteration

IV. Federated Learning Clients Dynamic Grouping Method based on Local Computational Efficiency

This section describes the functional modules of FedGLCE in detail. The implementation of the algorithm and complexity analysis are given accordingly.

A. Compute Local Computational Efficiency

The local computational efficiency e_i of client c_i is calculated using the local training time t_i and the training sample size of local data $|d_i|$ as,

$$e_i = \frac{|d_i|}{t_i}. \tag{1}$$

Considering the dynamic nature of mobile edge environment, the computational resources available to clients fluctuate at different iteration rounds. Assuming a constant amount of local data $|d_i|$, the local computational efficiency of client c_i in the r-th iteration round is denoted as $e_i = \frac{|d_i|}{t_i^{r_{latest}}}$, where r_{latest} is the most recent iteration round, $r_{latest} < r$. Since all clients are not trained locally during the initialization phase, the local data quantity $|d_i|$ of client c_i is employed to initialize the local computational efficiency e_i as,

$$e_i = \begin{cases} |d_i|\,, c_i(r_{latest}=0) \\ \\ |d_i|\,/t_i^{r_{latest}}, c_i r_{latest}(r_{latest}\neq 0) \end{cases}, i \in \{1,...,n\}. \tag{2}$$

In the general case, since the initialized local computational efficiency of a client is substantially higher than its true local computational efficiency, that is $|d_i| \gg e_i = |d_i|\,/t_i^{r_{latest}}$. Therefore, at the beginning of the iteration, distributions of clients who have not yet completed the federated training are more dispersed among groups than those who have. In other words, clients that fail to complete the training are more likely to be selected, thus achieving a fast access to the real local computational efficiency.

B. Construct Polynomial Distribution Set

FedGLCE randomly picks one client from each group to participate in the local training and updates that client's local computational efficiency based on the client grouping. As a result, the client grouping procedure not only influences the client participation in federated learning, but it may also induce client drift [20] [21] due to the participation of some clients, resulting in the non-convergence of the global model. To avoid this issue, this section constructs a reasonable polynomial distribution set $\{\mathrm{D}_k^r\}_{k=1}^m$ so that the grouping results have a guarantee of the global model convergence.

Assumption 1 (Unbiasedness). If the expected value of the local model aggregation of clients selected for training is identical to the global model aggregation when all clients are included, we define this client selection is unbiased as,

$$_{S^r}[\omega^r] = {}_{S_r}[\sum_{i'\in S_r} w_{i'}\omega_{i'}^r] := \sum_{i=1}^{n} w_i\omega_{i'}^r, \tag{3}$$

where S^r is the subset of clients involved in training for iteration round r, $w_{i'}$ is the aggregation weight of client $c_{i'}$ with respect to S^r, and $\omega_{i'}^r$ is the local model parameter of $c_{i'}$.

Based on the above assumption, a polynomial distribution set $\{\mathrm{D}_k^r\}_{k=1}^m$ of length m is constructed in iteration round r. The sampling probability $q_{i,k}^r$ of clients in each distribution is computed so as to divide n clients into m groups. According to this construction, $q_{i,k}^r$ needs to satisfy the following,

$$\forall k \in \{1,...,m\}, \sum_{i=1}^{n} q_{i,k}^r = 1, q_{i,k}^r \geq 0. \tag{4}$$

Equation (4) ensures the feasibility of dividing clients into m groups. When sampling clients using a polynomial distribution D_k^r, the expected value of the global model is,

$$_{\mathrm{D}_k^r}[\sum_{i'\in \mathrm{D}_k^r} w_{i'}\omega_{i'}^r] := \sum_{i=1}^{n} q_{i,k}^r\omega_{i'}^r. \tag{5}$$

Since the expected value possesses a linear property, the next iteration round's expected value of global model is the average results derived from Equation (5) over the set of polynomial distributions $\{\mathrm{D}_k^r\}_{k=1}^m$. That is,

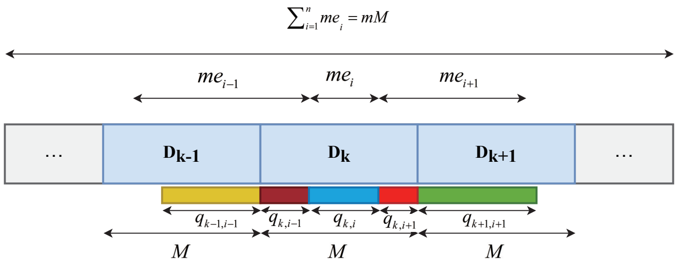

Fig. 3. The construction of polynomial distribution set

$$_{S^r}[\omega^r] = \sum_{k=1}^{m}\frac{1}{m}\sum_{i=1}^{n} q_{i,k}^r\omega_{i'}^r. \tag{6}$$

Extending the unbiasedness assumption of Equation (3) to m independent polynomial distributions $\{\mathrm{D}_k^r\}_{k=1}^m$ yields the property as,

$$\forall i \in \{1,...,n\}, \sum_{k=1}^{m} q_{i,k}^r = mw_i. \tag{7}$$

As illustrated in Fig. 3, if the value of the polynomial distribution D_k^r is M, i.e. $|\mathrm{D}_k^r| = M$, then the distribution set has Mm elements. The local computational efficiency set $E = \{e_i | i = 1,...,n\}$ is arranged in descending order, and the priority of assigning the client's local computation efficiency is offered to the polynomial distribution that has not yet reached M. To put it another way, after a client completes the efficiency assignment, the number of elements in all but at most one distribution should be 0 or M.

The total efficiency value assigned to each client on $\{\mathrm{D}_k^r\}_{k=1}^m$ is $e_i{}' = me_i$. $e_i{}' = M\alpha_i + \beta_i$ means that client c_i has a sampling probability of 1 on all α_i distributions, and the remaining efficiency value β_i is assigned to $(m - \sum \alpha_i)$ distributions, i.e., $Mm = \sum\limits_i e_i^{'} = M(\sum\limits_i \alpha_i) + \sum\limits_i \beta_i$. From $|\mathrm{D}_k^r| = M$, we can obtain that the construction satisfies Equation (4) and clients can be divided into m groups. Because of $e_i{}' = me_i$, the proportion of each client on all distributions in $\{\mathrm{D}_k^r\}_{k=1}^m$ can be expressed as $m\frac{e_i}{E} = m\frac{|d_i|/t_i^{r_{latest}}}{|d_i|/\sum\limits_{i=1}^{n} t_i^{r_{latest}}} \sim mw_i$, satisfying Equation (7).

Substituting Equation (7) into Equation (5), we can get the following equation,

$$_{S^r}[\omega^r] = \sum_{k=1}^{m}\frac{1}{m}\sum_{i=1}^{n} q_{i,k}^r\omega_{i'}^r = \sum_{i=1}^{n} w_i\omega_{i'}^r. \tag{8}$$

It is observed that Equation (8) yields the same expected value of the global model as Equation (3), satisfying the unbiasedness assumption. As a result, after grouping clients according to the above construction, the global model has a guarantee of convergence in federated learning.

Algorithm 1 Client Grouping

```
Input: $n$: number of clients, $m$: number of client groups, $r$: current iteration round, $r_u$: iteration round scheduled for updating client grouping results, $\{e_i|i=1,...,n\}$: client local computational efficiency set
Output: $\{q_{i,k}^{r_u}|i=1,...,n;\ k=1,...,m\}$: probability that a client belongs to each client group
1: if $r\%r_u == 0$ then
2:     define $k=1$
3:     define $count = 0$
4:     define $M=\sum_{i=1}^{n} e_i$
5:     for $i=1$ to $n$ do
6:         $count = count + me_i$
7:         $count = M\alpha_i + \beta_i$
8:         if $\alpha_i > k$ then
9:             $(q_{i,k}^{r_u})' = M - \beta_{i-1}$
10:            $\forall l \geq k+1 s.t. (\alpha_i - 1) - l \geq 0, (q_{i,k}^{r_u})' = M$
11:        end if
12:        $(q_{i,\alpha_i}^{r_u})' = \beta_i$
13:        $k = \alpha_i$
14:    end for
15:    return $\{q_{i,k}^{r_u} = \frac{(q_{i,k}^{r_u})'}{M}|i=1,...,n;\ k=1,...,m\}$
16: end if
```

C. Algorithm Design and Complexity Analysis

Two algorithms are utilized by FedGLCE, including Client Grouping algorithm and Local Computational Efficiency Update algorithm. We give the detailed design process and complexity analysis of these two algorithms.

The client grouping takes the current iteration round r and clients' local computational efficiency $\{e_i|i=1,...,n\}$ as inputs. Then build a set of polynomial distributions for each client group, and output the probability $\{q_{i,k}^{r}|i=1,...,n; k=1,...,m\}$ that a client belongs to each polynomial distribution (client group). Algorithm 1 gives the detailed steps.

Because each step in the building of the polynomial distribution set $\{\mathrm{D}_k^r\}_{k=1}^m$ necessitates an action on the client local computational efficiency or polynomial distribution, of which the complexity is $O(n+m)$. Since grouping makes sense only when the number of client groups m is fewer than the number of clients n, the overall complexity of the Client Grouping algorithm is $O(n+m)=O(n)$.

The Local Computational Efficiency Update algorithm is employed to update the local computational efficiency of selected clients in federated training. Specifically, during the initialization phase, the local computational efficiency of all clients is unknown, and Algorithm 2 initializes their local computational efficiency using clients' local data quantity. At the end of the local training phase, the local computational efficiency of clients participating training is updated. In summary, Algorithm 2 receives as input the number of client local data and the subset of clients participating in training for that iteration round, and outputs the current descending sequence of client local computation efficiency, thus providing input to Algorithm 1. The detailed steps are shown in Algorithm 2.

Algorithm 2 Local Computational Efficiency Update

```
Input: $r$: current iteration round, $\{|d_i|\,|i=1,...,n\}$: number of local data on each client, $S^r$: subset of clients involved in training in iteration round $r$
Output: $\{e_i|i=1,...,n\}$: Local computational efficiency set of each client
1: define $E=\{e_i=0|i=1,...,n\}$
2: if $r == 1$ then
3:     for $i=1$ to $n$ do
4:         $e_i = d_i$
5:     end for
6: else
7:     $S^r = \mathrm{ClientGrouping}(E)$
8:     $\{t_{i'}^r|c_{i'} \in S^r\} = \mathrm{Train}()$
9:     for $c_{i'}$ in $S^r$ do
10:        $e_{i'} = |d_{i'}|/t_{i'}^r$
11:    end for
12:    $\mathrm{Descend}(E)$
13: end if
14: return $E=\{e_i|i=1,...,n\}$
```

TABLE II
Key Performance Parameters in Federated Learning

Parameter Server			
CPU	AMD Ryzen5-3600@1.8GHz 12 cores		
GPU	NVIDIA GeForce RTX 2060 CUDA 1920 cores		
RAM	16GB		
Client			
Type	computational Resources	RAM	Number
Raspberry Pi 3B+	Cortex-A53@1.4GHz CPU×1	1GB	5
Nvidia Jetson Nano	Maxwell CUDA 128 cores GPU×1	4GB	10
0.8 CPU Docker	Ryzen5-3600@1.8GHz CPU×0.8	3GB	15
1.6 CPU Docker	Ryzen5-3600@1.8GHz CPU×1.6	3GB	15
2.4 CPU Docker	Ryzen5-3600@1.8GHz CPU×2.4	3GB	10
3.2 CPU Docker	Ryzen5-3600@1.8GHz CPU×3.2	3GB	10
4.0 CPU Docker	Ryzen5-3600@1.8GHz CPU×4.0	3GB	10
4.8 CPU Docker	Ryzen5-3600@1.8GHz CPU×4.8	3GB	10
5.6 CPU Docker	Ryzen5-3600@1.8GHz CPU×5.6	3GB	10
Nvidia Jetson TX2	Pascal CUDA 256 cores GPU×1	8GB	5

For Algorithm 2, the startup phase requires traversal of the client set $\{c_i|i=1,...,n\}$, with complexity $O(n)$. In the training phase, after client grouping is completed, one client from each group is selected to participate in the iterative round of local training, i.e. $|S^r|=m$, with complexity $O(m)$. Besides, because the complexity of the descending order for the local computational efficiency $E=\{e_i|i=1,...,n\}$ is $O(n\log(n))$, the overall complexity of Algorithm 2 is $O(n\log(n))$.

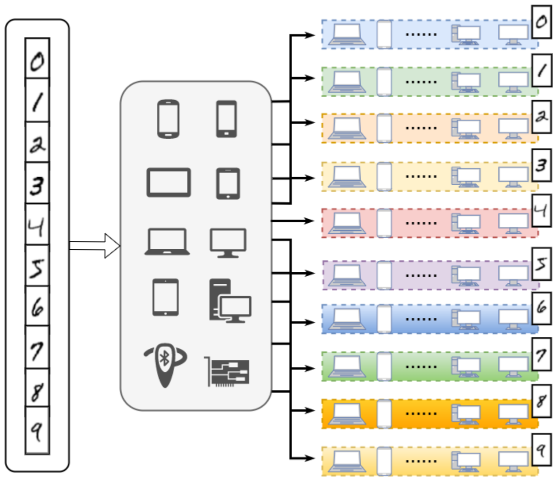

Fig. 4. The generation of MNIST-Fed dataset

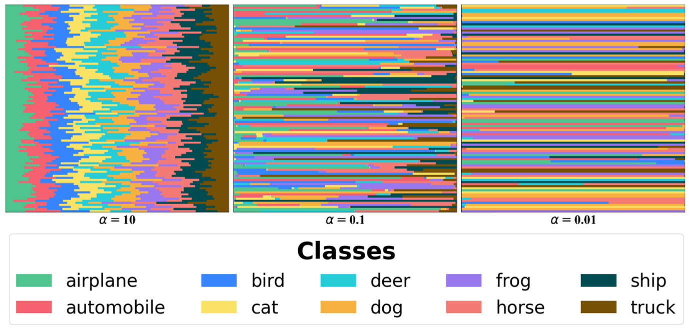

Fig. 5. The percentage of local data belonging to categories for 100 clients in CIFAR-10-Fed dataset

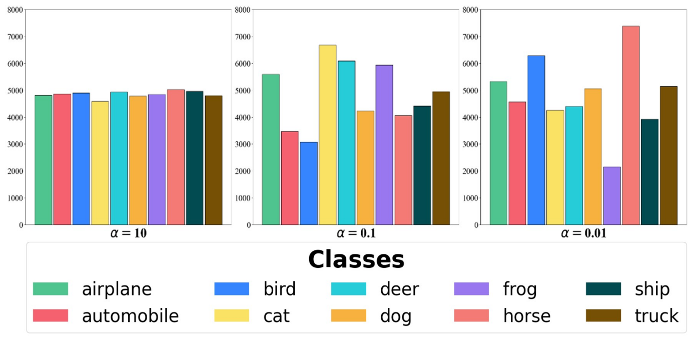

Fig. 6. The distribution of samples in each category in CIFAR-10-Fed dataset

V. EXPERIMENTS

A. Experiment Settings

1) Federated learning environment: In this paper, a federated learning framework with one central server and $n = 100$ clients is created to simulate the real mobile edge environment. We also adopt docker containers to virtualize server resources and increase the sorts of clients. Specifically, the central server in this federated learning framework is a workstation with GPU, and clients consist of 5 Raspberry Pi 3B+, 10 Nvidia Jetson Nano, 5 Nvidia Jetson TX2, as well as 80 docker containers with computational power ranging from 0.8 to 5.6 core CPUs. The exact parameters are listed in Table 2.

The clients and server are both on the same LAN and communicate each other via the PySyft's WebSocket protocol. Since this paper is concerned with the difference in computational power of clients, to shield the influence of communication conditions, clients are trained in a serial manner to complete the federated learning.

2) Federated datasets and models: The MNIST-Fed and CIFAR-10-Fed datasets are created using a set of 100 clients and the MNIST and CIFAR-10 public datasets. In MNIST-Fed, as shown in Fig. 4, 100 clients are sorted into 10 groups (matching to the 10 categories), and each group is given the same data category, with 500 training examples and 100 test samples for each client. The dirichlet distribution $Dir(\alpha)$ is employed to generate a cross-category CIFAR-10-Fed dataset for 100 clients, where $\alpha \in [0, +\infty)$ reflects the degree of heterogeneity of CIFAR-10-Fed. As α has a larger value, it indicates a more consistent data distribution among clients and $\alpha = 0$ indicates that a client is assigned to only one category of data. Fig. 5 and Fig. 6 show data distributions with different α in CIFAR-10-Fed dataset. The 10, 30, 30, 20, and 10 clients in federated learning, respectively, hold 100, 250, 500, 750, and 1000 training samples and 20, 50, 100, 150, and 200 test samples.

The CNN model is chosen as the experimental model for the MNIST-Fed dataset. Two convolutional layers, two maximum pooling layers, and two fully connected layers make up the model. Another CNN model, which consists of three convolutional layers with Dropout, two maximum pooling layers, and two fully connected layers, is also utilized as the experimental model for the CIFAR-10-Fed dataset.

3) Baselines and parameter settings: FedAvg [7] and FedDrop [14] are chosen as the experimental baselines. In addition, FedGLCE is compared against FedSS [9], a client grouping method based on the number of local data, and TiFL [19], a client grouping method based on the local training time.

Set the iteration round $R = 200$, batch size $batch\_size = 20$, learning rate $lr = 0.1$, and number of clients $n = 100$ in the MNIST-Fed dataset experiments. In the CIFAR-10-Fed dataset experiments, change $R = 300$ and $lr = 0.05$, while other parameters remain the same.

B. Analysis of Hyperparameter Selection for FedGLCE

FedGLCE's hyperparameter r_u indicates that the result of client grouping is updated after each r_u iteration rounds, and m indicates that clients are separated into m client groups. r_u impacts the process times of client grouping, thus affecting the degree to which the grouping results reflect clients' real computational capacity. Because one client from each group must be chosen for local training after grouping, m influences the number of clients who participate in local training in iterative rounds. As a result, we must select suitable r_u and m for following experiments. The experimental outcomes are presented in Tables 3 to 6.

TABLE III
EXPERIMENTAL RESULTS OF FEDGLCE IN DIFFERENT GROUP UPDATE PERIODS (MNIST-FED, 5 GROUPS)

Group update period r_u	Variance of participation	Variance of local accuracy	Accuracy of global test
5	19.54	262,48	81.82%
10	17.92	61.41	86.99%
20	16.54	**37.47**	**91.14%**
50	**11.76**	80.41	90.74%

TABLE IV
EXPERIMENTAL RESULTS OF FEDGLCE IN DIFFERENT GROUP UPDATE PERIODS (MNIST-FED, 10 GROUPS)

Group update period r_u	Variance of participation	Variance of local accuracy	Accuracy of global test
5	60.00	24.81	94.93%
10	58.40	24.93	95.08%
20	53.94	19.71	94.51%
50	**43.42**	**16.36**	**95.46%**

The following analysis can be made by comparing the experimental data in Tables 3 to 6,

- Under different data heterogeneity conditions, the period r_u of FedGLCE for updating groups has varying degrees of influence on the performance indexes. Specifically, a larger r_u should be taken under high data heterogeneity conditions while a smaller r_u should be taken under low data heterogeneity conditions.
- Different r_u values of FedGLCE have a greater impact on the variance of participation and local accuracy, indicating that r_u does affect FedGLCE in balancing the differences of client participation and local performance of the global model. However, r_u has little impact on global test accuracy.
- The variance of participation, the variance of local accuracy, and the accuracy of global test are all affected by the number of groups m used in FedGLCE. In particular, when it comes to the variance of participation, m fundamentally fails to affect the number of clients who attend local training. In terms of local accuracy variance, the global model with groups $m = 10$ has the largest value among clients, and the grouping results are closest to the true distribution of client computational capacity. When the data heterogeneity is high, the setting of $m = 5$ and $m = 10$ can effectively improve the global test accuracy. However, the global test accuracy is not statistically different from the above setting when the data heterogeneity is low. This is mostly due to FedGLCE's ability to select a larger number of clients for training, reducing the detrimental impact of high data heterogeneity on global test accuracy.

Through the above experiments and analysis of hyperparameter selection, considering the existence of data heterogeneity in mobile edge environment, r_u is determined as 20 and m is determined as 10, i.e., the client grouping is re-performed every 20 iteration rounds, and 100 clients are divided into 10 groups. This setup id applied in the following comparison experiments.

C. Client Participation

Table 7 shows results of client participation variance on the MNIST-Fed dataset and CIFAR-10-Fed datasets ($\alpha = 10$,$\alpha = 0.1$,$\alpha = 0.01$) with varying degrees of data heterogeneity under the above hyperparameter settings for five approaches, i.e. FedAvg, FedDrop, TiFL, FedSS, and FedGLCE.

As shown in Table 7, the variance of client participation for FedGLCE surpasses FedAvg, FedDrop, and FedSS by roughly 16.1%, 74.3%, and 31.2%, respectively, for the MNIST-Fed dataset experiments, but is poorer than TiFL. Because FedSS based on the sample number is incapable of effectively grouping clients, the client selection strategy devolves from random selection within each group to random selection across all clients, with a selection effect comparable to FedAvg. FedGLCE proposed in this paper utilizes the number of data and local computational efficiency, which is related to training time, for client grouping, therefore, the client participation variance is better than FedSS, but is worse than TiFL because FedGLCE adopts the local data quantity to initialize the local computational efficiency.

Table 7 also shows the experimental performance of the aforementioned five approaches on the CIFAR-10-Fed dataset with varying degrees of data heterogeneity in columns 4-6. FedGLCE offers the best variance of client participation among five methods under three data heterogeneity settings. When $\alpha = 10$, in addition to the increase in dataset complexity, the number of client local data differs, and thus all results show a significant rise in client participation variance. TiFL, which only considers training time for grouping, cannot distinguish between clients with different data amounts and thus has poor experimental results. Whereas FedSS, which is based on sample number for grouping, has better experimental results but is slightly inferior to FedGLCE because the local training time is not considered and the grouping results are static. Furthermore, FedDrop performs the poorest in terms of client participation variance, reflecting huge disparities in the number of local training by clients. The variance of client participation for the same approach in $\alpha = 0.1$ and $\alpha = 0.01$ settings is not large, owing to the fact that increasing data heterogeneity has little effect on the results of client grouping based on system heterogeneity.

D. Local Test Accuracy

The local test accuracy variance of FedGLCE exceeds FedAvg, FedDrop, and FedSS by roughly 17.9%, 15.5%, and 3.4%, respectively, for the MNIST-Fed dataset evaluation, as shown in Table 8, but is poorer than TiFL. The experimental results are generally consistent with the variation of client participation, indicating that increasing the variance of client participation is helpful for obtaining a global model with more balanced local test accuracy. Because the MNIST dataset

TABLE V
EXPERIMENTAL RESULTS OF FEDGLCE IN DIFFERENT GROUP UPDATE PERIODS (CIFAR-10-FED, 5 GROUPS)

Group update period r_u	**CIFAR-10-Fed($\alpha = 10$)**			**CIFAR-10-Fed($\alpha = 0.1$)**			**CIFAR-10-Fed($\alpha = 0.01$)**		
	Var (par)	**Var (local)**	**Acc (global)**	**Var (par)**	**Var (local)**	**Acc (global)**	**Var (par)**	**Var (local)**	**Acc (global)**
5	**249.92**	**35.80**	**71.40%**	264.68	109.23	62.77%	267.72	526.39	48.56%
10	262.56	38.94	70.79%	259.04	144.64	62.19%	250.56	406.60	49.75%
20	265.04	38.59	69.84%	255.08	125.86	**63.47%**	269.62	405.93	**50.87%**
50	270.86	44.25	69.83%	**237.14**	**96.54**	63.36%	**241.34**	**387.73**	50.38%

TABLE VI
EXPERIMENTAL RESULTS OF FEDGLCE IN DIFFERENT GROUP UPDATE PERIODS (CIFAR-10-FED, 10 GROUPS)

Group update period r_u	**CIFAR-10-Fed($\alpha = 10$)**			**CIFAR-10-Fed($\alpha = 0.1$)**			**CIFAR-10-Fed($\alpha = 0.01$)**		
	Var (par)	**Var (local)**	**Acc (global)**	**Var (par)**	**Var (local)**	**Acc (global)**	**Var (par)**	**Var (local)**	**Acc (global)**
5	**1096.90**	**29.94**	**71.12%**	1172.14	137.26	64.38%	1196.06	385.10	53.05%
10	1126.62	36.00	70.54%	1133.98	103.86	65.32%	1168.56	**321.12**	53.04%
20	1134.32	35.29	71.02%	**1022.92**	**94.08**	**65.44%**	1116.88	373.48	53.72%
50	1165.22	41.05	70.94%	1115.62	113.21	65.10%	**1094.64**	353.47	**53.77%**

TABLE VII
EXPERIMENTAL RESULTS OF CLIENT PARTICIPATION DIFFERENCES FOR CLIENT GROUPING APPROACHES

Comparison approaches	**Evaluation metrics**	**Dataset**			
		MNIST-Fed	**CIFAR-10-Fed**		
			Dir(10)	***Dir(0.1)***	***Dir(0.01)***
FedAvg	$Var(Fre)$	64.30	1323.4	1326.86	1322.5
FedDrop		210.02	1887.22	1900.14	1818.46
TiFL		**17.02**	1537.14	1527.5	1547.14
FedSS		78.36	1258.06	1259.86	1252.56
FedGLCE		53.94	**1134.52**	**1022.92**	**1094.64**

TABLE VIII
EXPERIMENTAL RESULTS OF LOCAL TEST ACCURACY DIFFERENCES FOR CLIENT GROUPING APPROACHES

Comparison approaches	**Evaluation metrics**	**Dataset**			
		MNIST-Fed	**CIFAR-10-Fed**		
			Dir(10)	***Dir(0.1)***	***Dir(0.01)***
FedAvg	$Var(Acc)$	24.02	**34.29**	116.28	406.91
FedDrop		23.33	37.74	97.80	549.22
TiFL		**15.77**	38.79	132.16	408.10
FedSS		20.40	37.56	95.10	520.88
FedGLCE		19.71	35.29	**94.08**	**373.48**

is simple, the local test accuracy variance of the above approaches cannot differ much.

Table 8 also shows the experimental performance of the above five approaches on the CIFAR-10-Fed dataset with varying degrees of data heterogeneity in columns 4-6. In the high data heterogeneity situation, FedGLCE has the best local test accuracy variance for the three data heterogeneity settings ($\alpha = 0.1$, $\alpha = 0.01$). TiFL, which simply considers training time for grouping, has the worst experimental performance for $\alpha = 10$. With increased data heterogeneity in $\alpha = 0.1$ and $\alpha = 0.01$, there is a significant increase in the local test accuracy variance of the same approach, despite the trend in the variance of client participation, which is primarily due to differences in learning performance among clients with different data distributions. Under high data heterogeneity, FedGLCE can better discriminate between clients and hence provide a more uniform local test accuracy distribution. Furthermore, the local test accuracy variance of FedDrop is much larger than that of FedGLCE under high data heterogeneity, confirming our research motivation of this work, which is to balance the local performance variance of the global model among clients by ensuring client participation in a mobile edge environment.

E. Total Hours of Federated Training

As can be seen in Fig. 7, the overall distribution of the total training time for five approaches remains constant for different datasets as well as data heterogeneity settings. FedAvg, which randomly selects clients for training, has a more balanced total training time among five approaches, whereas FedDrop, which directly discards some clients, achieves the shortest training time in all settings. FedGLCE, which considers local computational efficiency, has the shortest total training time of the three grouping methods. Because FedGLCE takes into account a variety of computationally efficient clients, while both FedSS and TiFL obtain a subset of participating clients with a high proportion of slow clients.

F. Simulation Summary

The above experimental results show that FedGLCE can balance the participation of clients in the local federated training through the dynamic client grouping approach based on local computing efficiency under the premise of ensuring the local test accuracy. Besides, performance biases of the global model on clients are improved. Furthermore, FedGLCE also reduces the total training time of clients in federated learning compared with state-of-the-art client grouping approaches.

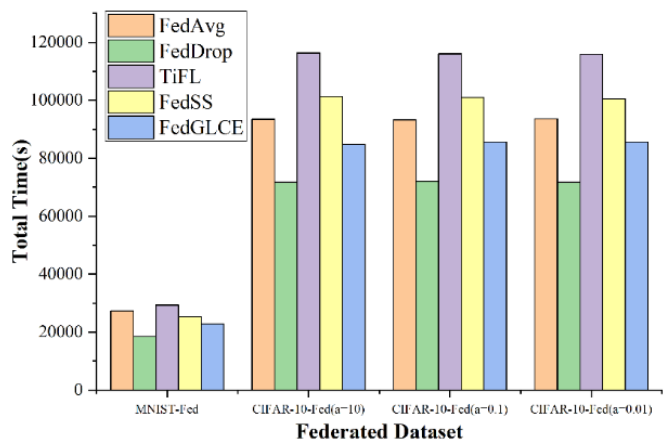

Fig. 7. Experimental results of total federated learning training hours for different client grouping approaches

VI. CONCLUSIONS

With the storage and computing power of mobile edge devices grows, it becomes possible to train mobile edge devices using federated learning. To reduce the computational and communication overhead incurred in federated training, client grouping approaches have proven to be an effective way. However, the global model trained by extant methods show performance biases among clients, mainly because computationally weak clients are often not selected for training. To ensure that clients are not discarded while balancing the participation of clients in local training, this paper proposes a Federated Learning Clients Dynamic Grouping Method based on Local Computational Efficiency (FedGLCE). The latest local training time of clients is used for the periodic updating process of client grouping. The experimental results show that FedGLCE outperforms FedAvg, FedDrop, TiFL and FedSS by about 1.30 times, 1.86 times, 1.49 times and 1.23 times in client participation variance, and surpasses FedAvg, FedDrop, TiFL and FedSS by about 1.24 times, 1.04 times, 1.40 times and 1.01 times in local test accuracy variance, respectively, balancing the performance bias of the global model among clients.

VII. ACKNOWLEDGEMENT

This work is supported by The National Natural Science Foundation of China (No. 61902110, 61832005), Postgraduate Research & Practice Innovation Program of Jiangsu Province (No. KYCX22_0610), Key Research and Development Project of Jiangsu Province (No. BE2020729), Science Technology Achievement Transformation of Jiangsu Province (No. BA2021002), Key Technology Project of China Huaneng Group (No.HNKJ19-H12, HNKJ20-H64).

REFERENCES

[1] G. R. Sreekanth, S. Ahmed, M. Sarac, I. Strumberger, N. Bacanin, and M. Zivkovic, ”Mobile fog computing by using SDN/NFV on 5G edge nodes,” *Computer Systems: Science & Engineering*, vol. 41, no. 2, pp. 751-765, Feb. 2022.

[2] X. Chu, H. Jiang, B. Li, D. Wang, and W. Wang, ”Editorial: Advances in mobile, edge and cloud computing,” *Mobile Networks and Applications*, vol. 27, no. 1, pp. 219-221, Jan. 2022.

[3] K. Wei, J. Li, M. Ding, C. Ma, H. Yang, F. Farokhi, S. Jin, T. Quek, and H. Poor, ” Federated learning with differential privacy: Algorithms and performance analysis,” *IEEE Transactions on Information Forensics and Security*, vol. 15, pp. 3454-3469, Apr. 2020.

[4] A. Nayyar, B. Nguyen, and N. Nguyen, ”The Internet of drone things (IoDT): Future envision of smart drones,” in *First international conference on sustainable technologies for computational intelligence*, pp. 563-580, Nov. 2019.

[5] C. Hu, R. Lu, and D. Wang, ”FEVA: A federated video analytics architecture for networked smart cameras,” *IEEE Network*, vol. 35, no. 6, pp. 163-170, Jun. 2021.

[6] T. Koutny, and M. Ubl, ”Parallel software architecture for the next generation of glucose monitoring,” *Procedia Computer Science*, vol. 141, pp. 279-286, 2018.

[7] B. McMahan, E. Moore, D. Ramage, S. Hampson, and B. Arcas, ”Communication-efficient learning of deep networks from decentralized data,” in *Proceedings of the 20th International Conference on Artificial Intelligence and Statistics*, pp. 1273-1282, Apr. 2017.

[8] T. Huang, W. Lin, W. Wu, L. He, K. Li, and A. Zomaya, ”An Efficiency-boosting client selection scheme for federated learning with fairness guarantee,” *IEEE Transactions on Parallel and Distributed Systems*, vol. 32, no. 7, pp. 1552-1564, Jul. 2021.

[9] Y. Fraboni, R. Vidal, L. Kameni, and M. Lorenzi, ”Clustered sampling: Low-variance and improved representativity for clients selection in federated learning,” in *International Conference on Machine Learning*, pp. 3407-3416, Jul. 2021.

[10] J. Hong, Z. Zhu, S. Yu, Z. Wang, H. Dodge, and J. Zhou, ”Federated adversarial debiasing for fair and transferable representations,” in *Proceedings of the 27th ACM SIGKDD Conference on Knowledge Discovery Data Mining*, pp. 617-627, Aug. 2021.

[11] J. Lu, H. Liu, Z. Zhang, J. Wang, S. K. Goudos, and S. Wan, ”Toward fairness-aware time-sensitive asynchronous federated learning for critical energy infrastructure,” *IEEE Transactions on Industrial Informatics*, vol. 18, no. 5, pp. 3462-3472, May. 2022.

[12] S. Horvath, S. Laskaridis, M. Almeida, I. Leontiadis, S. Venieris, and N. Lane, “Fjord: Fair and accurate federated learning under heteogeneous targets with ordered dropout,” *Advances in Neural Information Processing Systems*, vol. 34, 2021.

[13] T. Li, A. Sahu, M. Zaheer, M. Sanjabi, A. Talwalkar, and V. Smith, ”Federated optimization in heterogeneous networks,” in *Proceedings of Machine Learning and Systems*, pp. 429-450, Mar. 2020.

[14] K. Bonawitz, H. Eichner, and W. Grieskamp, ”Towards federated learning at scale: System design,” in *Proceedings of Machine Learning and Systems*, pp. 374-388, Apr. 2019.

[15] S. Baghersalimi, T. Teijeiro, D. Atienza, and A. Aminifar, ”Personalized real-time federated learning for epileptic seizure detection,” *IEEE Journal of Biomedical and Health Informatics*, vol. 26, no. 2, pp. 898-909, Feb. 2022.

[16] Y. Zhou, Y. Fu, Z. Luo, M. Hu, D. Wu, Q. Z. Sheng, and S. Yu, ”The role of communication time in the convergence of federated edge learning,” *IEEE Transactions on Vehicular Technology*, vol. 71, no. 3, pp. 3241-3254, Mar. 2022.

[17] C. Nadiger, A. Kumar, and S. Abdelhak, ”Federated reinforcement learning for fast personalization,” in *IEEE Second International Conference on Artificial Intelligence and Knowledge Engineering (AIKE)*, pp. 123-127, Jun. 2019.

[18] A. Ghosh, J. Hong, D. Yin, and K. Ramchandran, ”Robust federated learning in a heterogeneous environment,” *CoRR*, vol. abs/1906.06629, 2019.

[19] Z. Chai, A. Ali, S. Zawad, S. Truex, A. Anwar, N. Baracaldo, Y. Zhou, H. Ludwig, F. Yan, and Y. Cheng, ”Tifl: A Tier-based Federated Learning System,” in *Proceedings of the 29th International Symposium on High-Performance Parallel and Distributed Computing*, pp. 125-136, Jun. 2020.

[20] X. Xu, S. Duan, J. Zhang, J. Zhang, Y. Luo, and D. Zhang, ”Optimizing Federated Learning on Device Heterogeneity with A Sampling Strategy,” in *29th IEEE/ACM International Symposium on Quality of Service*, pp. 1-10, Jun. 2021.

[21] S. Karimireddy, S. Kale, M. Mohri, S. Reddi, S. Stich, and A. Suresh, ”SCAFFOLD:Stochastic Controlled Averaging for Federated Learning,” in *ICML*, pp. 5132-5143, Jul. 2020.