Mao YC, Shen LJ, Wu J *et al.* Federated Dynamic Client Selection for Fairness Guarantee in Heterogeneous Edge Computing. JOURNAL OF COMPUTER SCIENCE AND TECHNOLOGY 33(1): 1–22 March 2023. DOI 10.1007/s11390-015-0000-0

Federated Dynamic Client Selection for Fairness Guarantee in Heterogeneous Edge Computing

Ying-Chi Mao[1,2] (毛莺池), *Senior Member*, *CCF*, Li-Juan Shen[1,2] (沈莉娟), Jun Wu[1,2,*] (吴俊), *Graduate Student Member*, *IEEE*, Ping Ping[1,2] (平萍), and Jie Wu[3] (吴杰), *Fellow*, *IEEE*

[1]*Key Laboratory of Water Big Data Technology of Ministry of Water Resources, Hohai University, Nanjing 211100, China*
[2]*School of Computer and Information, Hohai University, Nanjing 211100, China*
[3]*Center of Networked Computing, Temple University, Philadelphia PA 19122-6096, U.S.A*

E-mail: yingchimao@hhu.edu.cn; 211307040040@hhu.edu.cn; 1606010225@hhu.edu.cn; pingpingnjust@163.com; jiewu@temple.edu

Abstract Federated learning has emerged as a distributed learning paradigm by training at each client and aggregating at a parameter server. System heterogeneity hinders stragglers from responding to the server in time with huge communication costs. Although client grouping in federated learning can solve the straggler problem, the stochastic selection strategy in client grouping neglects the impact of data distribution within each group. Besides, current client grouping approaches make clients suffer unfair participation, leading to biased performances for different clients. In order to guarantee the fairness of client participation and mitigate biased local performances, we propose a Federated Dynamic Client Selection Method based on Data Representativity (FedSDR). FedSDR clusters clients into groups correlated with their own local computational efficiency. To estimate the significance of client datasets, we design a novel data representativity evaluation scheme based on local data distribution. Furthermore, the two most representative clients in each group are selected to optimize the global model. Finally, the DYNAMIC-SELECT algorithm updates local computational efficiency and data representativity states to regroup clients after periodic average aggregation. Evaluations on real datasets show that FedSDR improves in client participation by 27.4%, 37.9%, and 23.3% compared with FedAvg, TiFL, and FedSS, taking fairness into account in federated learning. In addition, FedSDR surpasses FedAvg, FedGS, and FedMS by 21.32%, 20.4%, and 6.90%, in local test accuracy variance, balancing the performance bias of the global model across clients.

Keywords federated learning fairness, computational efficiency, data distribution, client selection, client grouping

1 Introduction

As an emerging machine learning paradigm, federated learning [1] utilizes the local data and computing capacity of clients for training locally. The parameter server aggregates local models to generate a global model while maintaining the security constraint of local data storage [2]. Federated learning is capable of handling large-scale data [3] in the real-world scenarios and uses edge device resources to provide efficient machine learning in distributed computing.

However, there are some inherent characteristics in federated learning, such as low-speed broadband, various computing capacity of clients, unstable availability of the network, and so on [4, 5]. Some clients are called stragglers for delaying in computation or communication. Huge communication cost [6] is inevitable when the server fails to receive responses from stragglers in the aggregation phase. To reduce the communication cost caused by stragglers, FedAvg [1] drops stragglers directly but loses partial information [7] from stragglers.

Regular Paper

This work is supported by the Key Research and Development Program of China under Grant No.2022YFC3005401, Key Research and Development Program of China, Yunnan Province under Grant No.202203AA080009, Transformation Program of Scientific and Technological Achievements of Jiangsu Provence under Grant No.BA2021002 and the Key Research and Development Program of China, Jiangsu Province under Grant No.BE2020729.

*Corresponding Author

FedProx [8] decreases communication rounds by limiting the iteration frequency of partial clients. FedDrop [9] discards a portion of clients with low computational efficiency. Nonetheless, FedAvg [1], FedProx [8], and FedDrop [9] neglect the participation of stragglers, leaving the global model unable to effectively assess the value of local data. The three common client selection approaches perform high accuracy across some clients but low accuracy across others. Unfair opportunity for clients to participate in training leads to the performance bias [10]. Client grouping methods can improve fairness [11] by guaranteeing the equitable participation of various clients in federated learning. Unfortunately, the existing client grouping methods cannot accurately reflect the computational efficiency [12] differences from clients and only provide static grouping results [13]. Therefore, the above methods are not tailored to the edge computing where client computing capacity changes dynamically [14].

In addition, data distribution is also regarded as a vital factor in the fairness of federated learning [15], with the performance bias [16] of the global model. In the real-world scenarios, clients have different usage patterns, data samples and labels following different distributions [17]. The stochastic selection strategy and the inequal participation in the training make the global model skewed towards the common data distributions, failing to characterize the particular data distributions. Ozdayi *et al.* [18] demonstrated that non-IID data negatively impacts model performance and fairness negatively. As shown in Fig.1, the client B is assumed to be unsuitable for general scenarios. B is selected by the server due to its large data size. Although the client C performs well in general scenarios, the server discards it owning to its slow computational rate [19]. The global model loses valuable information from the discarded clients when the server aggregates locally-computed updates. As a result, the global model presents biased performances across local clients, especially for heterogeneous data applications. The degree of heterogeneity in data increases as the data distribution becomes more unbalanced [1]. If we can determine the heterogeneity degree of data according to distribution, selecting certain representative clients to train can minimize the divergence between global and local optimums.

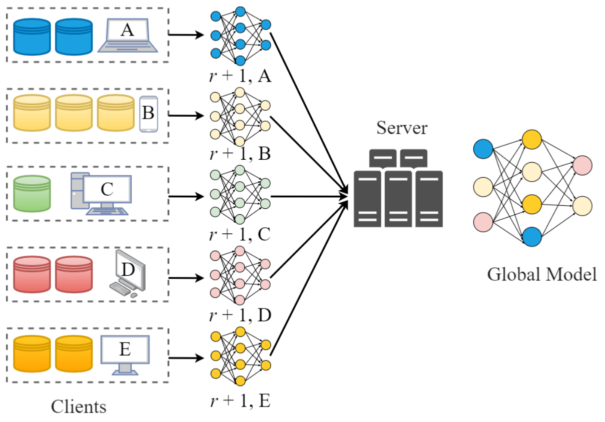


Fig.1. Fairness of federated learning.

Motivated by the above problems, we propose a Federated Clients Dynamic Selection Method based on Data Representativity (FedSDR) in this paper. FedSDR constructs a multinomial distribution set by the local computational efficiency of clients and maps the multinomial distribution to client groups. Clients with similar local computational efficiency are divided into the same group. Meanwhile, since clients in each group have varying data, we use data representativity to interpret the significance of heterogeneous data on the model's performance. Each client's data representativity is estimated based on the similarity between the local data distribution and the uniform distribution. To enhance the fairness of the client selection, FedSDR utilizes the designed DYNAMIC-SELECT algorithm to select clients with the highest data representativity from each group to participate in training. And the subsets of clients involved in training are updated dynamically at every iteration. Finally, the central server periodically updates the grouping results at the specified iteration rounds. In general, from the perspective of computing capacity and data distribution, FedSDR pledges fairness for all clients and balances the performance bias in heterogeneous systems.

The main contributions are summarized as follows.

- A multinomial distribution is constructed to map the local computational efficiency to their corresponding computing capacity of clients. Clients with similar computational efficiency are allocated into the same group, possessing fair opportunities to engage in training.

• A local data representativity evaluation scheme is proposed based on the similarity between the local and global data distribution. The scheme considers the impact of data heterogeneity on client performance in client selection.

• Furthermore, FedSDR utilizes the designed DYNAMIC-SELECT algorithm to execute client selection procedure and reselects clients dynamically for the next iteration to optimize the global model.

• Experiments on different heterogeneous datasets demonstrate that FedSDR outperforms in client participation by 27.4%, 37.9%, and 23.3% compared with FedAvg [1], TiFL [20], and FedSS [7], respectively. FedSDR surpasses FedAvg, FedGS, and FedMS by 21.32%, 20.4%, and 6.90% in local test accuracy variance. Moreover, FedSDR achieves good performance in training loss and test accuracy of the global model, within acceptable training time.

The remainder of this paper is organized as follows. Section 2 presents the related work of client grouping and client selection. The framework description and problem formulation are shown in Section 3. The design details of FedSDR are discussed in Section 4. The experiments and analysis are given in Section 5. At last, conclusions are drawn in Section 6.

2 Related Work

The current major approach to improve the fairness of federal learning is client grouping, but it ignores the differences of clients within the group and leads to unbiased client performance. Many researches measure client importance to select certain clients for training, as a way to improve the overall performance.

2.1 Client Grouping

As edge computing tends to contain high systems heterogeneity [21], stochastic client selection discards stragglers with poor computing capacity and causes biased performance across local clients. Skirpan *et al.* [19] researched model training bias in the context of machine learning settings and argued that the disadvantaged clients (with limited computing or memory, unbalanced data, etc.) should be taken into account along with the overall performance of the model. A series of researches considering disadvantaged clients are carried out to tackle the performance bias of global models across clients [22]. Grouping clients prior to federated training is one of the effective approaches. The client grouping methods divide clients into groups so that each group is a cluster of clients with similar characteristics [23]. Therefore, the dropout clients are guaranteed the opportunity to participate in local training, enhancing the fairness of client participation.

Current work on client grouping prefers to utilize techniques such as multinomial distribution clustering [7], and hierarchical clustering [24] to accomplish the grouping process. For example, the FRL framework [25] groups clients based on the similarity of users' behavior and generates a secondary global model for each group to receive intra-group updates from clients, reducing the negative impact of low-relevance data on the global model's performance. However, FRL does not specify the evaluation criteria and similarity metrics for grouping strategy. Fraboni *et al.* [7] employed clustered sampling and group clients according to their relevant data ratio, to achieve more uniform sampling results. Unfortunately, this method is executed only once in the initial phase, therefore it is not applicable to edge computing where the computing capacity changes dynamically. TiFL [20] groups clients based on their training time to ensure stragglers have the potential to participate in training. However, TiFL ignores the impact of data size on the training time and leads to biased performance across local models. FedSS [7] divides clients with similar amount of local data into the same group, so that clients with long training time (large data size) have the opportunity to participate in training. However, FedSS only groups clients in the initial phase, which is not suitable for the mobile edge environment where the local computing capacity changes dynamically.

2.2 Client Selection

Most of the grouping methods select clients randomly from each group and the clients with common distributed data are likely to be sampled [26, 27]. The stochastic selection strategy triggers biased performance and even a decline in accuracy without sufficient attention on the unbalanced data [28]. Especially in case of high statistical heterogeneity, the randomly selected dataset cannot reflect realistic data distribution of the global model [29, 30].

To address the above drawbacks associated with the stochastic client selection, a typical solution is determining the sample probability of the client by assessing the client's importance. Mohammed *et al.*[31] aimed to optimize client selection with an online heuristic algorithm based on the test accuracy, inspired by the secretary problem [32]. ISFedAvg [28] constructs a composite probability distribution combining client selection with local data selection, and derives the optimal importance sampling strategy considering the variability of data. FedCS [33] and MAB-CS [34] investigate client selection optimization in heterogeneous and uncertain systems, respectively. All the above approaches conduct client selection based on importance assessment but neglect the correlation among clients.

In addition, another solution is to group clients [34] and select a fraction of clients within each group. The strategy can provide a more uniform selection result [35]. FedGS [36] integrates clients with complementary classes into a uniformly distributed subset for local training, achieving similar learning results to all clients participated in training. However, the global model generated by FedGS exhibits low accuracy on clients with unbalanced data. Moreover, FedGS requires the transmission of specific information about the data distribution so that it does not strictly adhere to the data privacy settings in federated learning. Considering the mapping relationship between data and model, FedMS [37] clusters clients with similar data distribution by the model segmentation algorithm. When FedMS demands all clients to train periodically, the computation and communication burden surges greatly.

3 Framework of FedSDR and Problem Formulation

3.1 Framework Description

As shown in Fig.2, FedSDR allocates clients with similar local computational efficiency into the same group based on multinomial sampling. Then, FedSDR modifies the client selection strategy based on the data representativity within each group. For the unknown data distribution of clients within each group, we design a novel data representativity evaluation scheme to determine each client's data distribution. Then, FedSDR selects clients with the highest representativity from each group to participate in training, which can afford a fair engagement for each client.

The process of FedSDR can be broken down into the following detailed steps. First, we create a multinomial distribution set. Each client's computational efficiency is assigned to the multinomial distribution. The probability of a client sampled from the group relies on its proportion across all the multinomial distributions. Clients are divided into several groups with similar local computational efficiency. Subsequently, data representativity of each client is defined by the similarity of the local data distribution to the global data distribution. The data representativity evaluation scheme identifies the data representativity and the importance of each client in descending order. FedSDR selects the most representative clients logically. Moreover, after the selected clients participate in the training, their model parameters are uploaded for global aggregation. Finally, the DYNAMIC-SELECT algorithm updates each client's computational efficiency and selection weights dynamically according to the latest training time until the global model converges.

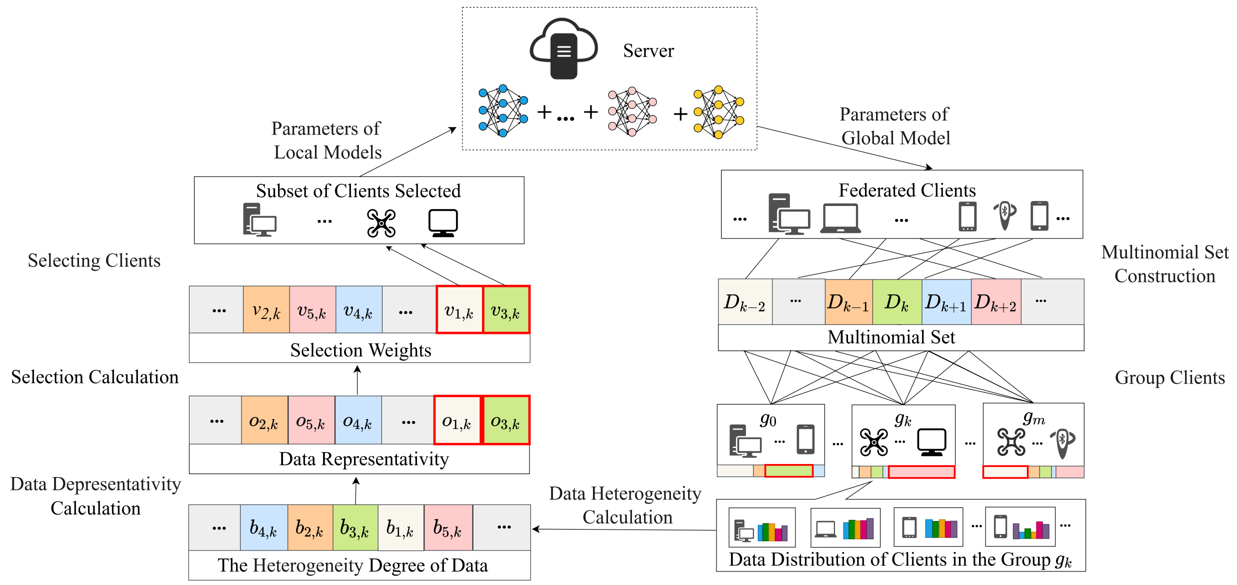

Server
Parameters of Local Models
Parameters of Global Model
Subset of Clients Selected
Federated Clients
Selecting Clients
Multinomial Set Construction
$v_{2,k}$ $v_{5,k}$ $v_{4,k}$ $v_{1,k}$ $v_{3,k}$
Selection Weights
D_{k-2} D_{k-1} D_k D_{k+1} D_{k+2}
Multinomial Set
Selection Calculation
Group Clients
$o_{2,k}$ $o_{5,k}$ $o_{4,k}$ $o_{1,k}$ $o_{3,k}$
Data Representativity
g_0 g_k g_m
Data Depresentativity Calculation
Data Heterogeneity Calculation
$b_{4,k}$ $b_{2,k}$ $b_{3,k}$ $b_{1,k}$ $b_{5,k}$
The Heterogeneity Degree of Data
Data Distribution of Clients in the Group g_k

Fig.2. Framework of FedSDR.

3.2 Problem Formulation

We assume that there are a total of n clients in the federated learning system. The total n clients are assigned to m groups $\{g_k^r \,|\, k = 1,...,m\}$. The set of all clients is defined as $\{c_i \,|\, i = 1,...,n\}$ and e_i represents the local computational efficiency of client c_i with the local data d_i. The global data $\sum_{k=1}^{m}\sum_{i=1}^{|g_k^r|} d_{i,\,k}^r$ is the total data of all clients. For client groups g_k^r, the clients in this group are denoted as $\{c_{i,\,k}^r \,|\, i = 1,...,|g_k^r|\}$. Clients with the most representative data are selected from each group to participate in training and upload their model parameters to aggregate the global model.

The fairness about client participation can be formulated as the variance in the amount of times a client is selected to train, i.e.,

$$Var(Fre) = \frac{1}{n}\sum_{i=1}^{n}(Fre_i - \overline{Fre})^2, \tag{1}$$

where Fre_i is the amount of times client c_i participated in local training and $\overline{Fre}$ is the average times of all clients participated in training.

In particular, we need to balance the global model's biased performance on local clients, i.e., to minimize the variance in local test accuracy as

$$Var(Acc) = \frac{1}{n}\sum_{i=1}^{n}(Acc_i - \overline{Acc})^2, \tag{2}$$

where Acc is the local test accuracy (the test accuracy of the local model on a client). The test accuracy is defined as dividing the number of correct predictions (compared with the ground truth) by the total number of test samples. $\overline{Acc}$ is the average local test accuracy of all the clients. Lower $Var(Acc)$ indicates a smaller difference in the test accuracy of clients improving federated learning fairness. The goal of our work is to minimize both (1) and (2).

At the r-th iteration round, n clients are assigned to m independent multinomial distributions $\{D_k^r\}_{k=1}^m$ based on each client's computational efficiency. Distributions in $\{D_k^r\}_{k=1}^m$ offer the sample weights of n clients based on each client's local computational efficiency e_i. And at two different iterations, multinomial distributions can vary. Then, the probability $q_{i,\,k}^r$ of clients c_i sampled from group g_k^r is computed based on the distribution $\{D_k^r\}_{k=1}^m$. When the probability $q_{i,\,k}^r$ is the highest in the k'-th group, client c_i is assigned to the group $g_{k'}^r$, i.e., $\forall k \in [1, k'-1] \cup [k'+1, m], q_{i,\,k'}^r > q_{i,\,k}^r$.

After all clients have been assigned, we need to calculate the data balance of each client in turn. The local data $d^r_{i,k}$ distributes as $A^r_{i,k}$ and the global data $\sum_{k=1}^{m}\sum_{i=1}^{|g^r_k|} d^r_{i,k}$ obeys a uniform distribution U. The similarity between local data and global data distribution is expressed as $b^r_{i,k}$. Due to U usually being set as a uniform distribution [38], $b^r_{i,k}$ represents $d^r_{i,k}$'s data balance. Therefore, the balance degree sequence of data is $\{b^r_{i,k} \,|\, i = 1, ..., |g^r_k|\}$. And $c^r_{i',k}$ will be given the highest sampled weight in group $g^r_{k'}$ when $A^r_{i',k}$ is determined as the most unbalanced or the most balanced local data distribution within the group, i.e., $\forall i \in [1, i'-1] \cup [i'+1, |g^r_k|], b^r_{i',k} < b^r_{i,k} \,|\, b^r_{i',k} > b^r_{i,k}$. Each group selects the two clients with the highest weights to participate in local training at the r-th iteration round. The global model aggregates local models of clients $S^r = \{S^r_k \,|\, k = 1, ..., m\}$ selected from all groups $\{g^r_k \,|\, k = 1, ..., m\}$. Since the local computational efficiency is updated over time, the set of selected clients S^r changes dynamically. The server repeats the progress above until the final iteration.

A list of the main symbolic parameters of FedSDR is presented in Table 1.

Table 1. Key Symbolic Parameters of FedSDR

Symbol	Definition
n	Number of clients
U	Global data distribution
m	Number of groups
r	Iterative round index
c_i	The i-th client
ω^r_i	Local model of client c_i for the r-th iteration
d_i	Local dataset of client c_i
e_i	Local computational efficiency of client c_i
t_i	Training time of client c_i
ω^r	Global model of client c_i for the r-th iteration
$t_i^{r_{\text{latest}}}$	Latest local training time of client c_i
S^r	Subset of clients in the r-th iteration
$q^r_{i,k}$	Sampling probability of client c_i in D^r_k
g^r_k	The k-th client group for the r-th iteration
$c^r_{i,k}$	The i-th client in group g^r_k
$d^r_{i,k}$	Local dataset of client $c^r_{i,k}$
$A^r_{i,k}$	Data distribution of dataset $d^r_{i,k}$
$b^r_{i,k}$	Data balance degree of client $c^r_{i,k}$
$o^r_{i,k}$	Data representativity of client $c^r_{i,k}$
$v^r_{i,k}$	Sampling weight of client $c^r_{i,k}$
S^r_k	Subset of clients participating in training in g^r_k

4 Federated Dynamic Client Selection Method based on Data Representativity

In this section, we introduce the design of FedSDR to solve the optimization problem as presented in (1) and (2). We first elaborate the client grouping process before client selection. Then we illustrate local data representativity evaluation scheme to measure data distributions of clients. Next, we select clients from each group based on local data representativity. Finally, we present the DYNAMIC-SELECT algorithm to to dynamically adjust clients participated in training for better performance.

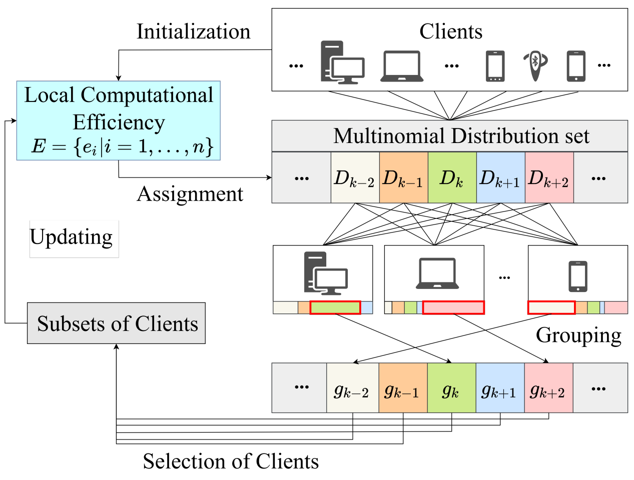

Initialization
Clients
Local Computational Efficiency
$E = \{e_i | i = 1, \ldots, n\}$
Multinomial Distribution set
D_{k-2} D_{k-1} D_k D_{k+1} D_{k+2}
Assignment
Updating
Subsets of Clients
Grouping
g_{k-2} g_{k-1} g_k g_{k+1} g_{k+2}
Selection of Clients

Fig.3. Client grouping of FedSDR.

4.1 Client Dynamic Grouping Method based on Local Computational Efficiency

The client grouping of FedSDR is presented in Fig.3, and we illustrate the detailed procedure below.

4.1.1 *Local Computational Efficiency*

The local computational efficiency e_i of client c_i is calculated using the local training time t_i and the local data size $|d_i|$ as (3)

$$e_i = \frac{|d_i|}{t_i}. \tag{3}$$

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 the 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

$$e_i = \begin{cases} |d_i|\,, \text{if } r_{\text{latest}} = 0, \\ & i \in \{1, ..., n\}. \\ |d_i|\,/t_i^{r_{\text{latest}}}, \text{if } r_{\text{latest}} \neq 0, \end{cases} \tag{4}$$

In the initial phase, because the training time of a client is 0 as in (4), we use the amount of data to initialize local computational efficiency, i.e., $|d_i| \gg e_i = |d_i|\,/t_i^{r_{\text{latest}}}$. 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 with large e_i are more likely to be selected to participate in training. Thus, the computational efficiency of these clients, which received the global model parameters, is much closer to that of training in a real-world environment.

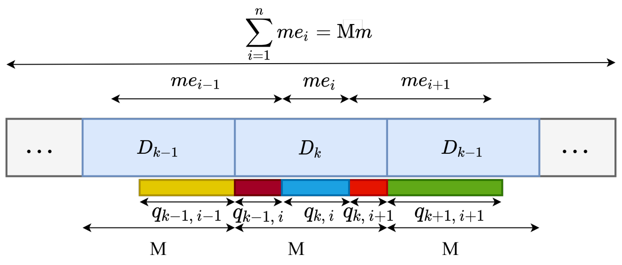


Fig.4. Construction of multinomial distribution set.

4.1.2 *Multinomial Distribution Set Construction*

FedSDR picks clients from each group to participate in the local training and updates 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 [37] due to the participation of some clients, resulting in the non-convergence of the global model. To avoid this issue, FedSDR constructs a reasonable multinomial distribution set $\{D_k^r\}_{k=1}^m$ so that the grouping results have the 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 in, we define this client selection is unbiased as

$$\mathbb{E}_{S^r}[\omega^r] = {}_{S^r}[\sum_{i' \in S^r} w_{i'} \omega_{i'}^r] := \sum_{i=1}^{n} w_i \omega_{i'}^r, \tag{5}$$

where E is the mathematic expectation, 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 multinomial distribution set $\{D_k^r\}_{k=1}^m$ 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,

$$\forall k \in \{1, ..., m\}, \sum_{i=1}^{n} q_{i,\,k}^r = 1, q_{i,\,k}^r \geq 0. \tag{6}$$

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

$$\mathbb{E}_{D_k^r}[\sum_{i' \in D_k^r} w_{i'} \omega_{i'}^r] := \sum_{i=1}^{n} q_{i,\,k}^r \, \omega_{i'}^r. \tag{7}$$

Since the expected value possesses a linear property, the next iteration round's expected value of global model is the average results derived from (7) over the set of multinomial distributions $\{D_k^r\}_{k=1}^m$, satisfying (8) as

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

Extending the unbiasedness assumption of (5) to m independent multinomial distributions $\{D_k^r\}_{k=1}^m$ yields the property as

$$\forall i \in \{1, ..., n\}, \sum_{k=1}^{m} q_{i,\,k}^r = m w_i. \tag{9}$$

As illustrated in Fig.4, if the amount of the multinomial distribution D_k^r is a constant M, i.e., $|D_k^r| = \mathrm{M}$, then the distribution set has Mm elements. The local computational efficiency set $E = \{e_i \mid i = 1, ..., n\}$ is arranged in descending order, and the priority of assigning the client's local computation efficiency is offered to the multinomial distribution that has not yet reached M. After a client completes the efficiency assignment, each of the remaining distributions should be 0 or M elements, except for at most one multinomial distribution.

The total efficiency value assigned to each client on $\{D_k^r\}_{k=1}^m$ is $e_i{}' = me_i$. And $e_i{}' = \mathrm{M}\alpha_i + \beta_i$ 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., $\mathrm{M}m = \sum\limits_i e_i^{'} = \mathrm{M}(\sum\limits_i \alpha_i) + \sum\limits_i \beta_i$. Due to $|D_k^r| = \mathrm{M}$, the construction $\{D_k^r\}_{k=1}^m$ satisfies (6) and clients can be divided into m groups. Because of $e_i{}' = me_i$, the proportion of each client on all distributions $\{D_k^r\}_{k=1}^m$ 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 (9). Substituting (9) into (7), we can get the following equation:

$$\mathbb{E}_{\mathrm{S}^r}[\omega^r] = \sum_{k=1}^{m} \frac{1}{m} \sum_{i=1}^{n} q_{i,\,k}^r\, \omega_{i'}^r = \sum_{i=1}^{n} w_i\, \omega_{i'}^r. \qquad (10)$$

It is observed that (10) yields the same expected value of the global model as (5), satisfying the unbiasedness assumption. As a result, after grouping clients according to the above construction, the global model has the guarantee of convergence in federated learning.

4.1.3 Client Grouping of FedSDR

The dynamic client grouping process of FedSDR can be split down into four steps as follows .

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

2) *Constructing multinomial distribution set.* A multinomial distribution set $\{D_k^r\}_{k=1}^m$ of length m is built based on the local computational efficiency set E, and the distribution $\{D_k^r\}_{k=1}^m$ corresponds to the client group g_k one by one.

3) *Grouping clients.* The probability $\{q_{i,\,k}^r\}_{k=1}^m$ of a client being sampled in distribution $\{D_k^r\}_{k=1}^m$ is calculated sequentially. If a client has the maximum sampling probability in distribution $\{D_k^r\}_{k=1}^m$, 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) *Updating local computational efficiency.* A random client from each group forms a subset S^r of clients participating in training. 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 and 3 are performed in every iteration round, dynamically updating the grouping results on a regular basis. Steps 1 and 4 are executed in each iteration round until the final iteration round comes.

4.1.4 Algorithm Design and Complexity Analysis

The GROUP algorithm performs steps for the multinomial distribution construction and client grouping at a federated learning iteration. And the hyperparameters are specified in the GROUP algorithm, including the number of clients n, the number of client groups m, and the group update period r_u, as detailed in Algorithm 1.

For the GROUP algorithm, its complexity is $O(n + m)$, since each step in the construction of a multinomially distributed set $\{D_k^r\}_{k=1}^m$ requires an operation on the local computational efficiency or multinomial distribution. For reason that this algorithm only makes sense when the number of client groups is smaller than the number of clients ($m < n$), its overall complexity is $O(n + m) = O(n)$.

The UPDATE algorithm is used to update the local computational efficiency of the clients. Specifically, the local computational efficiency of all the clients is unknown in the initial phase, hence the UPDATE algorithm initializes each client's local computational efficiency using the local data size. In the local training phase, the local computational efficiency is updated by the UPDATE algorithm at each iteration for the clients participating in training. In summary, the UPDATE algorithm receives local data size and the subset of clients participating in training as input, and outputs the current descending sequence of local computational efficiency. Thus, the UPDATE algorithm provides input to the GROUP algorithm, with the following detailed steps in Algorithm 2.

For the UPDATE algorithm, the initial phase requires traversal of the client set $\{c_i \mid i = 1, ..., n\}$ with a complexity of $O(n)$. In the training phase, each client of S^r is selected from each group to participate in the local training at the iteration, i.e., $|S^r| = 2m$ with the complexity of $O(2m)$. Because the complexity of the descending order operation on the local computational efficiency $E = \{e_i \mid i = 1, ..., n\}$ of the clients is $O(n \log(n))$, the overall complexity of the UPDATE algorithm is $O(n \log(n))$.

Algorithm 1 GROUP

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

Algorithm 2 UPDATE Algorithm

```
Input:
    $r$: current iteration round
    $\{|d_i| \mid i = 1, ..., n\}$: local data size on each client
    $S^r$: subset of selected clients
Output:
    $\{e_i \mid i = 1, ..., n\}$: local computational efficiency set

 1: define $E = \{e_i = 0 \mid i = 1, ..., n\}$
 2: if $r == 1$ then
 3:     for $i = 1$ to $n$ do
 4:         $e_i = d_i$
 5:     end for
 6: else
 7:     $S^r = GROUP(E)$
 8:     $\{t^r_{i'} \mid c_{i'} \in S^r\} = Train()$
 9:     for $c_{i'}$ in $S^r$ do
10:         $e_{i'} = |d_{i'}| / t^r_{i'}$
11:     end for
12:     descending $E$
13: end if
14: return $E = \{e_i \mid i = 1, ..., n\}$
```

4.2 Local Data Representativity Evaluation Scheme

The local data $d^r_{i,k}$ of client $c^r_{i,k}$ is assumed to obey the probability distribution $A^r_{i,k}$. The local data's information entropy is denoted as $H(d^r_{i,k}) = -\sum A^r_{i,k} \log(A^r_{i,k}) = \sum A^r_{i,k} \log \frac{1}{A^r_{i,k}}$ according to the information theory [39]. When the local data $d^r_{i,k}$ is represented by the global data U, its added information satisfies Kullback–Leibler divergence [39] of $A^r_{i,k}$ relative to U, i.e.,

$$\begin{aligned} &\mathrm{D_{KL}}(A^r_{i,k}, U) \\ &= -\sum A^r_{i,k} \log(U) - (-\sum A^r_{i,k} \log(A^r_{i,k})) \\ &= -\sum A^r_{i,k} \log \frac{U}{A^r_{i,k}} \\ &= \sum A^r_{i,k} \log \frac{A^r_{i,k}}{U} \end{aligned} . \quad (11)$$

Since $-\log(x)$ is a convex function, (11) is expressed as

$$\begin{aligned} &\mathrm{D_{KL}}(A^r_{i,k}, U) = \sum A^r_{i,k} \log \frac{A^r_{i,k}}{U} = \mathbb{E}_{A^r_{i,k}} \log \frac{A^r_{i,k}}{U} \\ &= -\mathbb{E}_{A^r_{i,k}} \log \frac{U}{A^r_{i,k}} \\ &\geq -\log \left(\sum A^r_{i,k} \frac{U}{A^r_{i,k}} \right) = -\log \left(\sum U \right) = 0 \end{aligned} , \quad (12)$$

where the KL divergence of $A^r_{i,k}$ relative to U is always bigger than 0. Meanwhile, larger KL divergence states that the distribution $A^r_{i,k}$ is more different relative to U. Besides, $\mathrm{D_{KL}}(A^r_{i,k}, U)$=0 when $A^r_{i,k}$ satisfies $A^r_{i,k}$=U.

Due to U obeying the uniform distribution, using (12) characterizes the balance degree of local data $d^r_{i,k}$, i.e.,

$$b^r_{i,k} = B(U, A^r_{i,k}) = e^{-\mathrm{D}_{KL}(A^r_{i,k}, U)}, \qquad (13)$$

where $b^r_{i,k}$ values $(0,1]$. If $b^r_{i,k}$ becomes smaller, the local data distributes more special. Conversely, the local data distribution is close to the uniform distribution (i.e., the local data distribution is more balanced). In particular, the local and global data has the same distribution while $b^r_{i,k} = 1$.

On the basis of client grouping, in the r-th iteration round, clients in group g^r_k use (13) to calculate their balance degree of data and then are sorted as $\{b^r_{i,k} \,|\, i = 1, ..., |g^r_k|\}$ in ascending order. Evidently, the data representativity of the client $c^r_{i',k}$ is defined as

$$o^r_{i',k} = \left(b^r_{i',k} - \frac{b^r_{1,k} + b^r_{|g^r_k|,k}}{2}\right)^2 + \varepsilon, \qquad (14)$$

where ε is the minimal value tending to 0, for avoiding $o^r_{i',k} = 0$.

Furthermore, the selection weight is determined based on (14) as,

$$v^r_{i',k} = o^r_{i',k} / \sum_{i=1}^{|g^r_k|} o^r_{i,k}. \qquad (15)$$

The server prefers to select clients with high representativity from the group g^r_k. Based on the sequence of selection weights $\{v^r_{i,k} \,|\, i = 1, ..., |g^r_k|\}$ as (15), a subset of clients is carried out from the group g^r_k. Consequently, we attain the client selection results at the r-th iteration via merging the selection results of each group.

4.3 Client Selection Based on Local Data Representativity

4.3.1 *Client Selection Process of FedSDR*

To ensure the engagement of data with different distributions, FedSDR is divided into four steps as follows.

1) *Calculating the degree balance of data.* According to the client grouping results, the balance degree of local data $b^r_{i,k}$ is measured by the similarity of $b^r_{i,k}$ and the global data U. By the way, the client group $\{g^r_k \,|\, k = 1, ..., m\}$ contains the clients $\{c^r_{i,k} \,|\, i = 1, ..., |g^r_k|\}$. In order to guarantee the privacy of the local data in federated learning, $b^r_{i,k}$ is calculated locally by the client. Hence, the central server cannot have access to the exact distribution of client data.

2) *Evaluating data representativity.* The balance degree of local data obtained from step 1 is sorted in ascending order in each group. From the client balance degree sequence $\{b^r_{i,k} \,|\, i = 1, ..., |g^r_k|\}$, each client's data representativity is evaluated utilizing (14). Therefore, the data representativity of clients in the group g^r_k is $\{o^r_{i,k} \,|\, i = 1, ..., |g^r_k|\}$.

3) *Constructing client selection weights and select clients.* In each group, construct client selection weights $\{v^r_{i,k} \,|\, i = 1, ..., |g^r_k|\}$ deriving from the data representativity sequence $\{o^r_{i,k} \,|\, i = 1, ..., |g^r_k|\}$. Based on the sequence $\{v^r_{i,k} \,|\, i = 1, ..., |g^r_k|\}$ obtained from step 2, the DYNAMIC-SELECT algorithm selects two clients from each group in turn to get a subset of clients participating in training. Moreover, these two clients serve the most peculiar data distribution and the most general data distribution in the group.

4) *Selecting clients and update models.* The subset of sampled clients $S^r = \{S^r_k \,|\, k = 1, ..., m\}$ consists of the selection results of each group. And the selected clients train models locally. The server aggregates local models on average and updates the local computational efficiency using the UPDATE algorithm. And thus the client selection results will vary dynamically with the updated computational efficiency.

The balance degree of local data is determined by the data distribution while the data representativity and selection weights of the client are relative to the client grouping results. As a result, the execution period of steps 1 and 2 is the same as the update period of the client grouping; steps 3 and 4 are executed at each iteration until the final iteration.

Table 2. Sampling Probability Distributions of Clients

Client	Sampling Probability Distribution
client_{05}	[0.080, 0.070, 0.090, 0.090, 0.170, 0.050, 0.070, 0.130, 0.070, 0.180]
client_{12}	[0.060, 0.050, 0.120, 0.090, 0.090, 0.130, 0.090, 0.140, 0.110, 0.120]
client_{27}	[0.130, 0.090, 0.080, 0.130, 0.100, 0.100, 0.090, 0.090, 0.080, 0.110]
client_{33}	[0.100, 0.050, 0.110, 0.050, 0.080, 0.200, 0.090, 0.080, 0.080, 0.160]
client_{39}	[0.080, 0.060, 0.060, 0.100, 0.110, 0.080, 0.090, 0.070, 0.130, 0.220]
client_{50}	[0.108, 0.068, 0.140, 0.068, 0.120, 0.072, 0.148, 0.100, 0.096, 0.080]
client_{71}	[0.148, 0.096, 0.096, 0.096, 0.096, 0.096, 0.124, 0.060, 0.116, 0.072]

4.3.2 Example of Client Selection

To understand the process described in Subsection 4.3.1 more intuitively, we give an example of the client selection process for client group g_k^r as the following.

1) *Calculating the balance degree of data.*

We input a group of clients $\{\text{client}_{05}, \text{client}_{12}, \text{client}_{27}, \text{client}_{33}, \text{client}_{39}, \text{client}_{50}, \text{client}_{71}\}$, the global uniform distribution $[0.1, 0.1, 0.1, 0.1, 0.1, 0.1, 0.1, 0.1, 0.1, 0.1]$ and the local data to the DYNAMIC-SELECT algorithm. The local data obeys the following sampling probability distribution in Table 2 respectively.

The data balance of data in each client included in this group is calculated as $\{\text{client}_{05} : 0.919, \text{client}_{12} : 0.959, \text{client}_{27} : 0.985, \text{client}_{33} : 0.912, \text{client}_{39} : 0.917, \text{client}_{50} : 0.964, \text{client}_{71} : 0.971\}$.

2) *Evaluating the data representativity and constructing client selection weights.*

Sort clients in ascending order by the balance degree of the corresponding data, as $\{\text{client}_{33} : 0.912, \text{client}_{39} : 0.917, \text{client}_{05} : 0.919, \text{client}_{12} : 0.959, \text{client}_{50} : 0.964, \text{client}_{71} : 0.971, \text{client}_{27} : 0.985\}$. Based on this ascending sequence, the data representativity of each client in the group is calculated using (14) as $\{\text{client}_{33} : 14.32, \text{client}_{39} : 11.20, \text{client}_{05} : 9.51, \text{client}_{12} : 2.05, \text{client}_{50} : 3.27, \text{client}_{71} : 6.20, \text{client}_{27} : 14.32\}$.

From the results above, $client_{33}$ and $client_{27}$ have the most unbalanced and the most balanced data distribution in the group, respectively. Further, the corresponding client selection weights for the group are constructed using (9) as {0.235, 0.184, 0.156, 0.033, 0.054, 0.102, 0.235}.

3) *Selecting clients.*

According to the client selection weights, the server selects a subset of clients $S_k^r = \{\text{client}_{33}, \text{client}_{27}\}$ to participate in training at this iteration.

4.4 DYNAMIC-SELECT Algorithm

4.4.1 Algorithm Design

The DYNAMIC-SELECT algorithm performs data representation evaluation and client selection for client group in turn. Each client is measured for the balance degree of local data based on the similarity between local data distribution and global uniform distribution. Evaluate data representativity according to the ascending sequence of balance degree $\{b_{i,k}^r \mid i = 1, ..., |g_k^r|\}$ and construct the client selection weights V_k^r within the group. And the hyperparameters to be specified in the DYNAMIC-SELECT algorithm include the number of clients n, the number of client groups m, and the group update period r_u. The DYNAMIC-SELECT algorithm is detailed in Algorithm 3.

Therfore, we get a subset of clients sampled from each group and merge the results to complete client selection. When the selected clients end the current iteration, their local computational efficiency requires being updated by the UPDATE algorithm.

Algorithm 3 DYNAMIC-SELECT Algorithm

```
Input:
    $n$: number of clients
    $m$: number of groups
    $r_u$: iteration round
    $\{g^r_k \,|\, k = 1, ..., m\}$: scheduled for updating client grouping results
    $\{c^r_{i,k} \,|\, i = 1, ..., |g^r_k|\}$: the clients in each group
    $\{A^r_{i,k} \,|\, i = 1, ..., |g^r_k| \,; k = 1, ..., m\}$: local data distributions
Output:
    $S^r = \{S^r_k \,|\, k = 1, ..., m\}$: client selection results
1:  define $S^r = \emptyset$
2:  for $k = 1$ to $m$ do
3:      define $S^r_k = \emptyset$
4:      define $V^r_k = \emptyset$
5:      if $r\%r_u == 0$ then
6:          for $i = 1$ to $|g^r_k|$ do
7:              $b^r_{i,k} = e^{-\mathrm{D_{KL}}(A^r_{i,k}, U)}$
8:          end for
9:          ascending $\{b^r_{i,k} \,|\, i = 1, ..., |g^r_k|\}$
10:         define $sum = 0$
11:         for $i = 1$ to $|g^r_k|$ do
12:             $o^r_{i,k} = (b^r_{i',k} - \frac{b^r_{1,k} + b^r_{|g^r_k|,k}}{2})^2 + \varepsilon$
13:             $sum = o^r_{i,k} + sum$
14:         end for
15:         for $i = 1$ to $|g^r_k|$ do
16:             $v^r_{i,k} = o^r_{i,k} / sum$
17:             add $v^r_{i,k}$ to $V^r_k$
18:         end for
19:     end if
20: else
21:     $V^r_k = V^{r-1}_k$
22:     end else
23:     $S^r_k = CHOICE(V^r_k)$
24:     $S^r = S^r \cup S^r_k$
25: end for
26: return $S^r$
```

4.4.2 Complexity Analysis

For the DYNAMIC-SELECT algorithm, calculating the balance degree of local data requires traversing every client, with a complexity of $O(n)$. The key step to evaluate the data representativity is to sort the data representativity of all clients in ascending order, with the complexity of $O(n \log(n))$. And the complexity is $O(n \log(n))$ in the data representativity evaluation phase. Then, a number of clients from each client group will be selected to participate in the local training for that iteration round, i.e., the number of clients selected exceeds the number of client groups $|S^r| > m$. The complexity of the selection phase is $O(m) < O(|S^r|) < O(n)$. Therefore, the overall complexity of the DYNAMIC-SELECT algorithm is $O(n \log(n))$ which is the same as that of FedMS, and significantly lower than FedGS's complexity of $O(2^n!)$.

5 Performance Evaluation

5.1 Experiment System

5.1.1 Federated Learning System

We create a federated learning framework with one central server and $n = 100$ clients to simulate the real mobile edge environment. The federated learning framework, in particular, uses Docker containers to expand the range of client types by virtualizing server resources. We present and publish Appendix publicly as a supplementary material online[1]. The exact parameters are listed in Appendix A1.

In this framework, the central server is a workstation with a GPU, and the client work nodes, while the client work nodes are comprised of 5 Raspberry Pi 3B+, 10 Nvidia Jetson Nano, 5 Nvidia Jetson TX2, as well as 80 Docker containers. The computational power of the Docker containers ranges from 0.8 to 5.6 CPU cores. In addition, the server and clients are situated on the same local area network (LAN) and use the WebSocket protocol provided by PySyft[2] to communicate with each other. Since our work is focused on the difference in computational power of clients, the clients are trained in a serial manner in order to shield the effect of communication conditions.

[1]https://github.com/Porcucu/Appendix, Jun. 2023.
[2]https://github.com/OpenMined/PySyft/tree/syft_0.2.x, Dec. 2020.

5.1.2 Federated Datasets

We use MNIST-Fed and CIFAR-10-Fed as the local training datasets, respectively. MNIST-Fed and CIFAR-10-Fed used for experiments in this paper are generated from the public datasets MNIST and CIFAR-10.

MNIST-Fed. MNIST-Fed is composed of the images of handwritten numbers from 0 to 9. 100 clients are divided into ten groups, and each client has the same class of data with 500 training samples and 100 test samples. There are 10 classes in total and each class has the same amount of samples.

CIFAR-10-Fed. A cross-category CIFAR-10-Fed is generated for 100 clients using the Dirichlet distribution $Dir(\alpha)$, where the parameter $\alpha \in [0, +\infty)$ monitors the heterogeneity degree of the generated federated dataset. When α has a large value, data distributions of clients are more consistent. $\alpha = 0$ indicates that only a single client is assigned to one class of data. When $m = 5$, 100 clients are divided into five groups and every group contains 10, 30, 30, 20, and 10 clients, respectively. The client of the five groups holds 100, 250, 500, 750, and 1000 training samples and 20, 50, 100, 150, and 200 test samples, respectively.

The data distributions in MNIST-Fed and CIFAR-10-Fed datasets are shown as the Appendix A2 and A3 of a supplementary material online[3].

5.1.3 Model Settings

The experiments of our work focus on the impact of the local test accuracy variance on performance of the global model, therefore the complexity of the model does not affect the experimental results and subsequent analysis.

For MNIST-Fed, an empirical CNN model with two convolutional layers, two maximum pooling layers, and two fully connected layers is created. Another CNN model consisting of three convolutional layers with dropout, two maximum pooling layers, and two fully connected layers, is also utilized as the experimental model for CIFAR-10-Fed. The architecture of CNN for experiments on MNIST-Fed and CIFAR-10-Fed is shown in Table 3 and Table 4, respectively.

Table 3. Model Architecture on the Client with MNIST-Fed

Layer	I_channel	O_channel	C_kernel	Step	A_function
Conv2d	1	20	5×5	1	ReLU
MaxPool2d	-	-	2×2	2	-
Conv2d	20	50	5×5	1	ReLU
MaxPool2d	-	-	2×2	2	-
Full Connect	800	500	-	-	ReLU
Full Connect	500	10	-	-	Softmax

Note: I_channel is the number of input channels. O_channel is the number of output channels. C_kernel is the number of convolution kernels. A_function is the activation function.

Table 4. Model Architecture on the Client with CIFAR-10-Fed

Layer	I_channel	O_channel	C_kernel	Step	A_function
Conv2d	3	32	3×3	1	ReLU
MaxPool2d	-	-	2×2	2	-
Conv2d	32	64	3×3	1	ReLU
MaxPool2d	-	-	2×2	2	-
Conv2d	64	64	3×	1	-
Full Connect	1024	64	-	-	ReLU
Full Connect	64	10	-	-	-

5.2 Experiment Settings

5.2.1 Baseline

As the most general client selection strategy in federated learning, FedAvg [1] is chosen as the experimental baseline for FedSDR. And FedSDR is compared with FedGS [36] and FedMS [37].

FedGS selects clients directly based on local distribution and FedMS is based on the similarity of data distribution. The two methods are similar to FedSDR in that they select clients based on client groups using local data distribution.

[3]https://github.com/Porcucu/Appendix, Jun. 2023.

5.2.2 Parameter Settings

Table 5 shows the parameters of models for the experiments on different datasets.

Table 5. Parameter Settings of Models with Different Datasets

Dataset	R	*batch_size*	lr	n	m
MNIST-Fed	200	20	0.1	100	10
CIFAR-10-Fed	300	10	0.05	100	10

Note: R is the number of iterations. *batch_size* is the batch size. lr is the learning rate. n is the number of clients. m is the number of client groups.

5.2.3 Evaluation Metrics

We evaluate the FedSDR method in terms of local test accuracy variance, client participation, local test accuracy, the training loss, the global test accurac, and total training time.

Client Participation $Var(Fre)$. $\overline{Fre}$ denotes the average times of clients participating in local training, whereas Fre_i is the client c_i's participation in local training. More concisely, the difference in participation among clients can be expressed as a variance. The lower the variance $Var(Fre)$, the better the method is at balancing the client participation.

Local Test Accuracy Variance $Var(Acc)$. The variance $Var(Acc) = \frac{1}{n}\sum_{i=1}^{n}\left(Acc_i - \overline{Acc}\right)^2$ equals the difference between each client's test accuracy and the average test accuracy. When $Var(Acc)$ is smaller, the method is more directly helpful in improving federated learning fairness.

Minimum Test Accuracy $\min(Acc_i)$. $\min(Acc_i)$ means minimum local test accuracy of 100 local models. If the minimum local test accuracy is acceptable, the method is effective in improving the local performance of the global model across clients indirectly. Thus, $\min(Acc_i)$ is an indirect metric in improving the fairness of federated learning.

The Training Loss L_g *and the Global Test Accuracy* Acc_g. L_g is the training loss of the global model. The training loss assesses the prediction error of all samples for the model on the training set. Acc_g is the test accuracy of the global model. Both L_g and Acc_g are adopted as the performance criterions of the global model in federated learning. The lower L_g and higher Acc_g, the better the global model is trained.

Total training time $Time_{\text{total}}$. The total training time consumed by each method is counted as $Time_{\text{total}}$. Since each approach has the same number of iteration rounds on the same federated dataset, smaller $Time_{\text{total}}$ indicates more efficient federated learning training.

5.3 Experiment Results

5.3.1 Analysis of Hyperparameter Selection

Table 6. Experimental Results of Different Group Update Periods (MNIST-Fed, 5 Groups)

r_u	$Var(Fre)$	$Var(Acc)$	$Acc_g(\%)$
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 7. Experimental Results of Different Group Update Periods (MNIST-Fed, 10 Groups)

r_u	$Var(Fre)$	$Var(Acc)$	$Acc_g(\%)$
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

FedSDR's hyperparameter r_u indicates that the results of client grouping are 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, and 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. r_u is selected as four different values for comparison experiments on the number of groups $m = 5$ and $m = 10$. The experimental outcomes are presented in Tables 6 to 9.

Table 8. Experimental Results of Different Group Update Periods (CIFAR-10-Fed, 5 Groups)

Group update	CIFAR-10-Fed($\alpha = 10$)			CIFAR-10-Fed($\alpha = 0.1$)			CIFAR-10-Fed($\alpha = 0.01$)		
Period r_u	$Var(Fre)$	$Var(Acc)$	$Acc_g(\%)$	$Var(Fre)$	$Var(Acc)$	$Acc_g(\%)$	$Var(Fre)$	$Var(Acc)$	$Acc_g(\%)$
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 9. Experimental Results of Different Group Update Periods (CIFAR-10-Fed, 10 Groups)

Group update	CIFAR-10-Fed($\alpha = 10$)			CIFAR-10-Fed($\alpha = 0.1$)			CIFAR-10-Fed($\alpha = 0.01$)		
Period r_u	$Var(Fre)$	$Var(Acc)$	$Acc_g(\%)$	$Var(Fre)$	$Var(Acc)$	$Acc_g(\%)$	$Var(Fre)$	$Var(Acc)$	$Acc_g(\%)$
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	165.22	41.05	70.94	**1115.62**	**113.21**	65.10	**1094.64**	**353.47**	53.77

FedSDR is compared with FedAvg [1], TiFL [20], and FedSS [7] to client participation.

The following analysis can be made by comparing the experimental results in Tables 6 to 9,

- From Table 6 and Table 7, larger r_u has good participation on IID dataset (MNIST-Fed). For MNIST-Fed, the variance of client participation tends to increase with larger r_u. For non-IID dataset (CIFAR-10-Fed), $Var(Fre)$, $Var(Acc)$ and Acc_g perform better for smaller r_u, with low heterogeneity ($\alpha = 10$), mainly because CIFAR-10-Fed is more complex than MNIST-Fed. For high data heterogeneity ($\alpha = 0.1$,$\alpha = 0.01$), $Var(Fre)$ and $Var(Acc)$ have similar trends to MNIST-Fed experiments, with better results at higher r_u. The global test accuracy slightly improves with increasing r_u. Specifically, a larger r_u should be taken under higher data heterogeneity conditions while a smaller r_u should be taken under lower data heterogeneity conditions.

- As shown in Table 6 and Table 7, the overall of the test accuracy variance is decreasing with increasing r_u. In terms of the global test accuracy, it can be seen that a larger r_u corresponds to higher test accuracy. Different r_u values of FedSDR impact greatly on $Var(Fre)$ and $Var(Acc)$. It can be noticed that r_u does affect FedSDR in balancing the differences of client participation and local performance of the global model. However, r_u has little impact on global test accuracy.

- $Var(Fre)$, $Var(Acc)$ and Acc_g are all affected by the number of groups m used in FedSDR in Table 8 and Table 9. Although numerically $Var(Fre)$ is much higher with $m = 10$ than that with $m = 5$, the number of clients participated in training with $m = 10$ is twice as many as that with $m = 5$ for selecting two clients per group. It can be observed that the square root of $Var(Fre)$ with $m = 10$ is also about twice as large as that with $m = 5$. Therefore, when it comes to the variance of participation, m fundamentally fails to affect the number of clients who attend local training when the number of clients is constant. 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 realistic distribution of client computational capacity. When the data heterogeneity is high, the settings of $m = 5$ and $m = 10$ can effectively improve the global test accuracy. However, the global test accuracy is not statistically different from varied r_u when the data heterogeneity is low. This is mostly due to FedSDR's ability to select more 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 groups. These settings applied in the following comparison experiments.

5.3.2 *Client Participation*

As can be seen in Table 10 on the MNIST-Fed dataset, FedSDR is slightly worse than TiFL in terms of client participation, but surpasses FedAvg, TiFL, and FedSS by roughly 27.4%, 37.9%, and 23.3%. Because MNIST-Fed is uniformly distributed, with each class being assigned to the same number of samples. TiFL based on training time grouping can characterize the systematic heterogeneity of clients better.

Table 10. Experiment Results of Client Participation

Method	$Var(Fre)$			
	MNIST-Fed	CIFAR-10-Fed		
		Dir(10)	Dir(0.1)	Dir(0.01)
FedAvg	64.30	1323.4	1326.86	1322.5
TiFL	**17.02**	1537.14	1527.5	1547.14
FedSS	78.36	1258.06	1259.86	1252.56
FedSDR	48.37	**1032.98**	**984.57**	**960.25**

When the degree of statistical heterogeneity varies, the variance of client participation increases significantly on CIFAR-10-Fed. In the high heterogeneity setting ($\alpha = 10$, $\alpha = 0.1$, $\alpha = 0.01$), FedSDR has the best client participation among all four approaches. Therefore, FedSDR demonstrates the ability to balance the participation of clients with different computing capacity, further reducing the performance variance of the global model across clients.

5.3.3 *Local Test Accuracy Variance*

Table 11 shows the results of the local test accuracy variance for the four approaches FedAvg, FedGS, FedMS, and FedSDR on MNIST-Fed as well as CIFAR-10-Fed. For the simple MNIST-Fed, FedSDR has the smallest local test accuracy variance and still surpasses all the other approaches.

Table 11. Experiment Results of Local Test Accuracy Variance

Method	$Var(Acc)$			
	MNIST-Fed	CIFAR-10-Fed		
		Dir(10)	Dir(0.1)	Dir(0.01)
FedAvg	24.02	34.29	116.28	406.91
FedGS	22.17	32.81	95.35	402.30
FedMS	18.89	**32.05**	93.34	343.91
FedSDR	**18.34**	32.46	**91.20**	**320.17**

On the CIFAR-10-Fed, the advantage of FedSDR gradually emerges with higher statistical heterogeneity. FedSDR reaches the optimal results in the local test accuracy variance. Under the three data heterogeneity settings, $Var(Acc)$ of FedSDR is 32.46, 91.20, and 320.17, respectively. FedSDR exceeds all the other methods and is optimal in the high data heterogeneity settings ($\alpha = 0.1$, $\alpha = 0.01$).

For $\alpha = 10$, despite the increased complexity of CIFAR-10, the local test accuracy variances of the four approaches are relatively close due to low data heterogeneity and low risk of fairness. Selecting the unbalanced data distribution involved in training has limited effect on balancing the local performance bias of the global model. For $\alpha = 0.1$, FedGS which is more similar to FedAvg, provides a smaller variance in the local test accuracy due to the data heterogeneity. Specifically, FedGS uniformizes the data distribution through dividing clients with complementary data distribution into the same group. FedMS and FedSDR both achieve a better equalization effect by selecting unbalanced data. With the further increase in data heterogeneity, FedGS cannot provide balanced local performance for the complementary distributions in the same group. Instead, FedMS and FedSDR are effective at improving accuracy performance on clients. In particular, FedSDR not only distinguishes between clients better, but also takes into account the peculiar distribution in each client group, and thus it can offer better local test accuracy variance than FedMS in a high statistical heterogeneity system.

5.3.4 Minimum Test Accuracy

Table 12 shows the minimum test accuracy results of the global model across 100 clients obtained by FedAvg, FedGS, FedMS, and FedSDR on MNIST-Fed and CIFAR-10-Fed.

Table 12. Experiment Results of Minimum Local Test Accuracy

Method	$\min(Acc_i)$ (%)			
	MNIST-Fed	CIFAR-10-Fed		
		Dir(10)	Dir(0.1)	Dir(0.01)
FedAvg	66.3	37.4	17.9	1.1
FedGS	68.7	41.0	21.0	3.33
FedMS	**78.5**	**50.0**	25.1	**12.5**
FedSDR	76.0	48.1	**25.3**	9.0

FedGS, FedMS, and FedSDR exceed FedAvg in terms of local test accuracy. With the continuous enhance of statistical heterogeneity, $Var(Acc)$ of FedSDR is sub-optimal or optimal in all the approaches. Although the statistical results of FedSDR and FedMS are similar, FedSDR's overall complexity is much lower than that of FedMS, consuming less local computational resources.

It is noticeable that FedSDR has an improved effect on the local test accuracy compared with FedAvg, FedGS, and FedMS. Considering local data distribution for client selection indeed helps to improve the performance of the global model across clients.

5.3.5 Training Loss and Test Accuracy of the Global Model

As shown in Fig.5, on the simple MNIST-Fed, the training loss of FedSDR arrives at the lowest and FedSDR is sub-optimal in the test accuracy.

On the CIFAR-10-Fed ($\alpha = 10$, $\alpha = 0.1$, $\alpha = 0.01$), although FedMS achieves the lowest training loss and the highest test accuracy of global model, FedSDR is able to achieve similar experimental results as FedMS. Besides, FedSDR accomplishes a significant improvement in terms of Acc_g, compared with FedGS and FedAvg as Fig.6.

The above experimental results demonstrate FedSDR's high adaptability to statistical heterogeneity. FedSDR boosts the test accuracy of the global model and local test accuracy variance by selecting clients with unbalanced data. The boosting effect is similar to the client selection method regarding the model similarity. But measuring the similarity between clients occupies great computational consumption. In general, FedSDR balances the performance differences of the global model across clients.

5.3.6 Total Training Time

Table 13 shows the experimental results of the total federated learning training time for the four approaches FedAvg, FedGS, FedMS, and FedSDR on MNIST-Fed and CIFAR-10-Fed with three different heterogeneity ($\alpha = 10$, $\alpha = 0.1$, $\alpha = 0.01$).

Table 13. Experiment Results of Total Training Time

Method	$Time_{\text{total}}$ (s)			
	MNIST-Fed	CIFAR-10-Fed		
		Dir(10)	Dir(0.1)	Dir(0.01)
FedAvg	51843	183572	185067	185058
FedGS	56827	229363	221896	226819
FedMS	69630	275156	271162	272965
FedSDR	**42794**	**165669**	**168176**	**168342**

As can be seen from Table 14, the overall distribution of the total training time for the four approaches

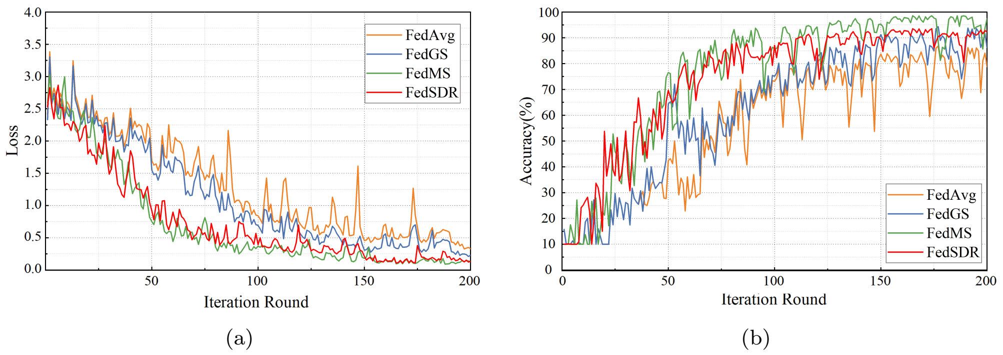


Fig.5. Training loss and test accuracy of the global model on MNIST-Fed. (a) Training loss L_g. (b) Test accuracy Acc_g.

remains the same on different datasets and data heterogeneity settings. Specifically, FedSDR presents the minimum training time mainly because it takes clients with varying computing efficiency into account and has low computational complexity of client selection based on data representativity within the group.

In the total training time, FedSDR spends less than all the other approaches. Besides, both FedGS and FedMS have increased the total training time compared to FedAvg. FedGS utilizes the local model output to evaluate the similarity among data distributions and obtains the unbalanced data distribution by grouping clients with complementary data into the same group. FedMS necessitates scheduling all the clients to periodically participate in local training, which significantly increases the consumption of computational resources and the training time. The above experimental results also show that FedSDR is more suitable for resource-constrained edge system.

5.3.7 *Experiment Conclusions*

In the heterogeneous experiments of this paper, the Federated Dynamic Client Selection Method based on Data Representativity (FedSDR) is compared with FedAvg, FedGS, and FedMS on the federated datasets MNIST-Fed and CIFAR-10-Fed ($\alpha = 10$, $\alpha = 0.1$, $\alpha = 0.01$).

The experimental results manifest that FedSDR can balance the client participation and relieve the performance biases of the global model across clients from the local data perspective. Compared with FedAvg, TiFL, and FedSS, FedSDR invokes client participation by 27.4%, 37.9%, and 23.3%, respectively. And FedSDR surpasses FedAvg, FedGS, and FedMS by 1.21 times, 1.20 times, and 1.07 times in the local test accuracy variance, respectively. Furthermore, FedSDR creates a global model with good performance in terms of training loss and training time, compared with other client selection approaches.

6 Conclusions

In this paper, we addressed the problem of unfair federated client selection in edge computing, which leads to a biased performance of the global model across clients. Facing the absence of some stragglers participating in training and unbalanced data distribution, we proposed the Federated Dynamic Client Selection Method based on Data Representativity (FedSDR). FedSDR groups clients for allowing clients with different computing capacity to participate in training, guaranteeing a fair chance to engage in training for all clients. Furthermore, FedSDR selects federated clients from each group based on data representativity for global aggregation. The selection strategy of FedSDR balances the performance of the global model across all

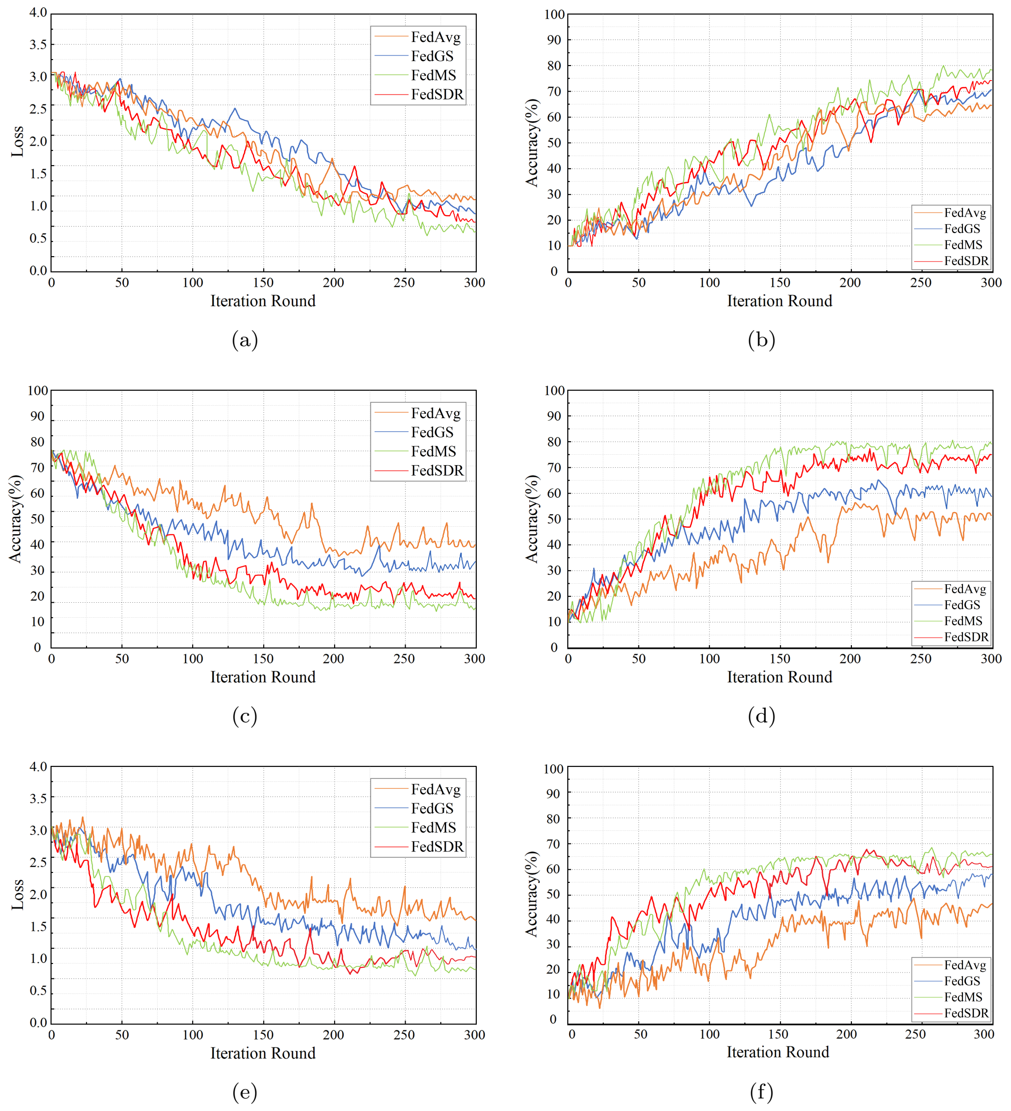


Fig.6. Training loss and test accuracy of the global model on CIFAR-10-Fed. (a) Training loss L_g when $\alpha = 10$. (b) Test accuracy Acc_g when $\alpha = 10$. (c) Training loss L_g when $\alpha = 0.1$. (d) Test accuracy Acc_g when $\alpha = 0.1$. (e) Training loss L_g when $\alpha = 0.01$. (f) Test accuracy Acc_g when $\alpha = 0.01$.

clients. We assessed our approach on several popular machine learning tasks which are performed with clusters of different sizes. The experimental results manifest that when the overall performance of the global model is guaranteed, FedSDR offers a balanced local test accuracy distribution. More interestingly, FedSDR ensures the fairness of the federated learning and enables all clients to obtain model parameters with great performance, and thus mitigating the performance bias of the global model among clients.

References

[1] McMahan B, Moore E, Ramage D, et al. Communication-efficient learning of deep networks from decentralized data. In*Proceedings of the 20th International Conference on Artificial Intelligence and Statistics*, Apr. 2017, pp.1273-1282. DOI; 10.48550/arXiv.1602.05629.

[2] Xu J, Wang H. Client Selection and Bandwidth Allocation in Wireless Federated Learning Networks: A long-term perspective. *IEEE Transactions on Wireless Communications*, 2021, 20(2): 1188-1200. DOI: 10.1109/TWC.2020.3031503.

[3] Wei K, Li J, Ding M, et al. Federated learning with differential privacy: Algorithms and performance analysis. *IEEE*

Transactions on Information Forensics and Security, 2020, 15: 3454-3469. DOI: 10.1109/TIFS.2020.2988575.

[4] Chu X, Jiang H, Li B, et al. Advances in mobile, edge and cloud computing. *Mobile Networks and Applications*, 2022, 27(1): 219-221. DOI: 10.1007/s11036-020-01654-9.

[5] Zhang F, Kuang K, Liu Y, et al. Unified group fairness on federated learning. arXiv:2111.04986, 2021. https://arxiv.org/abs/2111.04986, Febr. 2022.

[6] Hu C, Lu R, Wang D. FEVA: A federated video analytics architecture for networked smart cameras. *IEEE Network*, 2021, 35(6): 163-170. DOI: 10.1109/MNET.001.2100261.

[7] Fraboni Y, Vidal R, Kameni L, et al. Clustered sampling: Low-variance and improved representativity for clients selection in federated learning. In *International Conference on Machine Learning*, Jul. 2021, pp.3407-3416. DOI: 10.48550/arXiv.2105.05883.

[8] Li T, Sahu A K, Zaheer M, et al. Federated optimization in heterogeneous networks. *Proceedings of Machine Learning and Systems*, 2020, 2: 429-450. DOI: 10.48550/arXiv.1812.06127.

[9] Bonawitz K, Eichner H, Grieskamp W, et al. Towards federated learning at scale: System design. *Proceedings of Machine Learning and Systems*, 2019, 1: 374-388. DOI: 10.48550/arXiv.1902.01046.

[10] Gao H, Thai M T, Wu J. When Decentralized Optimization Meets Federated Learning. *IEEE Network*, 2023, 1-7. DOI: 10.1109/MNET.132.2200530.

[11] Hong J, Zhu Z, Yu S, et al. Federated adversarial debiasing for fair and transferable representations. In *Proceedings of the 27th ACM SIGKDD Conference on Knowledge Discovery Data Mining*, Aug. 2021, pp.617-627. DOI: 10.1145/3447548.3467281.

[12] Livni R, Shalev-Shwartz S, Shamir O. On the computational efficiency of training neural networks. *Advances in neural information processing systems*, 2014, 27. DOI: 10.48550/arXiv.1410.1141.

[13] Hao X, Ren W, Xiong R, et al. Fair and autonomous sharing of federate learning models in mobile internet of things. arXiv:2007.10650, 2020. https://arxiv.org/abs/2007.10650, Sept. 2021.

[14] Horvath S, Laskaridis S, Almeida M, et al. Fjord: Fair and accurate federated learning under heterogeneous targets with ordered dropout. *Advances in Neural Information Processing Systems*, 2021, 34: 12876-12889. DOI: 10.48550/arXiv.2102.13451.

[15] Xin F, Zhang J, Luo J, et al. Federated learning client selection mechanism under system and data heterogeneity. In *2022 IEEE 25th International Conference on Computer Supported Cooperative Work in Design (CSCWD)*, May. 2022, pp.1239-1244. DOI: 10.1109/CSCWD54268.2022.9776061.

[16] Zhao F, Huang Y, Sai A M V V, et al. A cluster-based solution to achieve fairness in federated learning. In *2020 IEEE Intl Conf on Parallel & Distributed Processing with Applications, Big Data & Cloud Computing, Sustainable Computing & Communications, Social Computing Networking (ISPA/BDCloud/SocialCom/SustainCom)*, Dec. 2020, pp.875-882. DOI: 10.1109/ISPA-BDCloud-SocialCom-SustainCom51426.2020.00135.

[17] Zhao Z, Joshi G. A dynamic reweighting strategy for fair federated learning. In *ICASSP 2022 - 2022 IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP)*, Dec. 2022, pp.8772-8776. DOI: 10.1109/ICASSP43922.2022.9746300.

[18] Amiri S, Belloum A, Nalisnick E, et al. On the impact of non-IID data on the performance and fairness of differentially private federated learning. In *2022 52nd Annual IEEE/IFIP International Conference on Dependable Systems and Networks Workshops (DSN-W)*, Jun. 2022, pp.52-58. DOI: 10.1109/DSN-W54100.2022.00018.

[19] Zhang W, Weiss J C. Fair decision-making under uncertainty. In *2021 IEEE International Conference on Data Mining (ICDM)*, Dec. 2021, pp.886-895. DOI: 10.1109/ICDM51629.2021.00100.

[20] Chai Z, Ali A, Zawad S, et al. Tifl: A tier-based federated learning system. In *Proceedings of the 29th International Symposium on High-Performance Parallel and Distributed Computing*, Jun. 2020, pp.125-136. DOI: 10.1145/3369583.3392686.

[21] Ek S, Lalanda P, Portet F. Federated learning within pervasive heterogeneous environments. In *2022 IEEE International Conference on Pervasive Computing and Communications Workshops and other Affiliated Events (PerCom Workshops)*, Mar. 2022, pp.134-135. DOI: 10.1109/PerComWorkshops53856.2022.9767288.

[22] Zhou Y, Fu Y, Luo Z, et al. The role of communication time in the convergence of federated edge learning. *IEEE Transactions on Vehicular Technology*, 2022, 71(3): 3241-3254. DOI: 10.1109/TVT.2022.3144099.

[23] Karimireddy S P, Kale S, Mohri M, et al. Scaffold: Stochastic controlled averaging for federated learning. In *Proceedings of the 37th International Conference*

on Machine Learning, Nov. 2020, pp.5132-5143. DOI: 10.48550/arXiv.1910.06378.

[24] Rong Y. Staged text clustering algorithm based on K-means and hierarchical agglomeration clustering. In *2020 IEEE International Conference on Artificial Intelligence and Computer Applications (ICAICA)*, Jun. 2020, pp.124-127. DOI: 10.1109/ICAICA50127.2020.9182394.

[25] Nadiger C, Kumar A, Abdelhak S. Federated reinforcement learning for fast personalization. In *IEEE Second International Conference on Artificial Intelligence and Knowledge Engineering (AIKE)*, Jun. 2019, pp.123-127. DOI: 10.1109/AIKE.2019.00031.

[26] Du X, He Y, Huang J Z. Random sample partition-based clustering ensemble algorithm for big data. In *2021 IEEE International Conference on Big Data (Big Data)*, Dec. 2021, pp.5885-5887. DOI: 10.1109/BigData52589.2021.9671297.

[27] Ji S, Xing R. Clustering ensemble of massive data based on trusted region. In *2021 3rd International Conference on Machine Learning, Big Data and Business Intelligence (MLBDBI)*, Dec. 2021, pp.337-340. DOI: 10.1109/MLBDBI54094.2021.00070.

[28] Rizk E, Vlaski S, Sayed A H. Optimal importance sampling for federated learning. In *ICASSP 2021-2021 IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP)*, Jun. 2021, pp.3095-3099. DOI: 10.1109/ICASSP39728.2021.9413655.

[29] Gong B, Xing T, Liu Z, et al. Adaptive client clustering for efficient federated learning over Non-IID and imbalanced data. *IEEE Transactions on Big Data*, 2022. DOI: 10.1109/TBDATA.2022.3167994.

[30] Baghersalimi S, Teijeiro T, Atienza D, et al. Personalized real-time federated learning for epileptic seizure detection. *IEEE Journal of Biomedical and Health Informatics*, 2021, 26(2): 898-909. DOI: 10.1109/JBHI.2021.3096127.

[31] Mohammed I, Tabatabai S, Al-Fuqaha A, et al. Budgeted online selection of candidate iot clients to participate in federated learning. *IEEE Internet of Things Journal*, 2021, 8(7): 5938-5952. DOI: 10.1109/JIOT.2020.3036157.

[32] Correa J, Cristi A, Feuilloley L, et al. The secretary problem with independent sampling. In *Proceedings of the 2021 ACM-SIAM Symposium on Discrete Algorithms (SODA)*, 2021, pp.2047-2058. DOI: 10.1137/1.9781611976465.122.

[33] Nishio T, Yonetani R. Client selection for federated learning with heterogeneous resources in mobile edge. In *ICC 2019-2019 IEEE international conference on communications (ICC)*, May. 2019, pp.1-7. DOI: 10.1109/ICC.2019.8761315.

[34] Yoshida N, Nishio T, Morikura M, et al. Mab-based client selection for federated learning with uncertain resources in mobile networks. In *2020 IEEE Globecom Workshops (GC Wkshps*, Dec. 2020, pp.1-6. DOI: 10.1109/GCWkshps50303.2020.9367421.

[35] Yin T, Li L, Lin W, et al. Grouped Federated Learning: A decentralized learning framework with low latency for heterogeneous devices. In *2022 IEEE International Conference on Communications Workshops (ICC Workshops)*, May. 2022, pp.55-60. DOI: 10.1109/ICCWorkshops53468.2022.9814558.

[36] Ma J, Sun X, Xia W, et al. Client Selection based on label quantity information for federated learning. In *2021 IEEE 32nd Annual International Symposium on Personal, Indoor and Mobile Radio Communications (PIMRC)*, Sept. 2021, pp.1-6. DOI:10.1109/PIMRC50174.2021.9569487.

[37] Xu X, Duan S, Zhang J, et al. Optimizing federated learning on device heterogeneity with a sampling strategy. In *2021 IEEE/ACM 29th International Symposium on Quality of Service (IWQOS)*, Jun. 2021, pp.1-10. DOI:10.1109/IWQOS52092.2021.9521361.

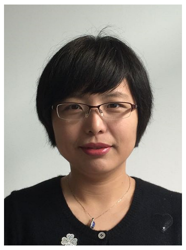
[38] Paszke A, Gross S, Massa F, et al. Pytorch: An imperative style, high-performance deep learning library. *Advances in neural information processing systems*, 2019, 32: 8026. DOI: 10.48550/arXiv.1912.01703.

[39] Giulini M, Menichetti R, Shell M S, et al. An information-theory-based approach for optimal model reduction of biomolecules. *Journal of chemical theory and computation*, 2020, 16(11): 6795-6813. DOI: 10.1021/acs.jctc.0c00676.

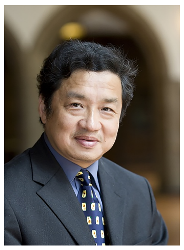
Ying-Chi Mao received her B.E. and M.S. degree in computer application technology from Hohai University, Nanjing, in 1999 and 2003 respectively, and her Ph.D. degree in computer software and theory from Nanjing University, Nanjing, in 2007. She is currently a professor in the College of Computer and Information at Hohai University, Nanjing. She is a senior member of CCF. Her research interests include edge intelligent computing, Internet of Things data analysis, and mobile sensing system.

Li-Juan Shen received her B.E. degree in computer and technology from Hohai University, Nanjing, in 2021. She is currently a Master student in the College of Computer and Information at Hohai University, Nanjing. Her research interests include distributed learning, edge computing, and federated learning.

Jun Wu received his B.E. degree in computer and technology from Hohai University, Nanjing, in 2020. He is currently a Master student in the College of Computer and Information at Hohai University, Nanjing. He is a graduate student member of IEEE. His research interests include distributed learning, edge computing, and federated learning.



Ping Ping received her Ph.D. degree in computer science and engineering from Nanjing University of Science and Technology, Nanjing, in 2009. She is currently an associate professor in the College of Computer and Information at Hohai University, Nanjing. Her research interests include network and information security, cloud computing and big data security, and image hiding encryption.



Jie Wu is the Director of the Center for Networked Computing and Laura H. Carnell professor at Temple University. He also serves as the Director of International Affairs at College of Science and Technology. He served as Chair of Department of Computer and Information Sciences from the summer of 2009 to the summer of 2016 and Associate Vice Provost for International Affairs from the fall of 2015 to the summer of 2017. Prior to joining Temple University, he was a program director at the National Science Foundation and was a distinguished professor at Florida Atlantic University. Dr. Wu is a Fellow of the AAAS and a Fellow of the IEEE. He is the recipient of the 2011 China Computer Federation (CCF) Overseas Outstanding Achievement Award. His current research fields include mobile computing and wireless networks, routing protocols, network trust and security, distributed algorithms, and cloud computing.