Group-based Hierarchical Federated Learning: Convergence, Group Formation, and Sampling

Jiyao Liu, Xinliang Wei, Xuanzhang Liu, Hongchang Gao, Yu Wang
Department of Computer and Information Sciences, Temple University
Philadelphia, Pennsylvania, USA
{jiyao.liu,xinliang.wei,xzliu,hongchang.gao,wangyu}@temple.edu

ABSTRACT

Hierarchical federated learning has been studied as a more practical approach to federated learning in terms of scalability, robustness, and privacy protection, particularly in edge computing. To achieve these advantages, operations are typically conducted in a grouped manner at the edge, which means that the formation of client groups can affect the learning performance, such as the benefits gained and costs incurred by group operations. This is especially true for edge and mobile devices, which are more sensitive to computation and communication overheads. The formation of groups is critical for group-based federated edge learning but has not been studied in detail, and even been overlooked by researchers. In this paper, we consider a group-based federated edge learning framework that leverages the hierarchical cloud-edge-client architecture and probabilistic group sampling. We first theoretically analyze the convergence rate with respect to the characteristics of the client groups, and find that group heterogeneity plays an important role in the convergence. Then, on the basis of this key observation, we propose new group formation and group sampling methods to reduce data heterogeneity within groups and to boost the convergence and performance of federated learning. Finally, our extensive experiments show that our methods outperform current algorithms in terms of prediction accuracy and training cost.

CCS CONCEPTS

• **Computing methodologies** → Machine learning; Distributed computing methodologies; • **Networks** → Network architectures.

KEYWORDS

Hierarchical federated learning, non-IID, group formation, group sampling, distributed learning, edge computing

ACM Reference Format:
Jiyao Liu, Xinliang Wei, Xuanzhang Liu, Hongchang Gao, Yu Wang. 2023. Group-based Hierarchical Federated Learning: Convergence, Group Formation, and Sampling. In *52nd International Conference on Parallel Processing (ICPP 2023), August 7–10, 2023, Salt Lake City, UT, USA.* ACM, New York, NY, USA, 10 pages. https://doi.org/10.1145/3605573.3605584

The work is partially supported by the US National Science Foundation under Grant No. CCF-1908843 and CNS-2006604.

Permission to make digital or hard copies of all or part of this work for personal or classroom use is granted without fee provided that copies are not made or distributed for profit or commercial advantage and that copies bear this notice and the full citation on the first page. Copyrights for components of this work owned by others than the author(s) must be honored. Abstracting with credit is permitted. To copy otherwise, or republish, to post on servers or to redistribute to lists, requires prior specific permission and/or a fee. Request permissions from permissions@acm.org.
ICPP 2023, August 7–10, 2023, Salt Lake City, UT, USA
© 2023 Copyright held by the owner/author(s). Publication rights licensed to ACM.
ACM ISBN 979-8-4007-0843-5/23/08…$15.00
https://doi.org/10.1145/3605573.3605584

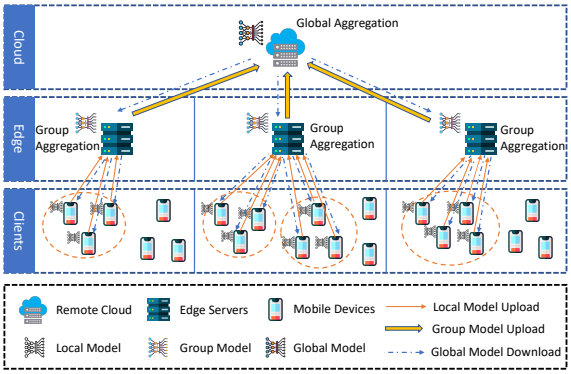
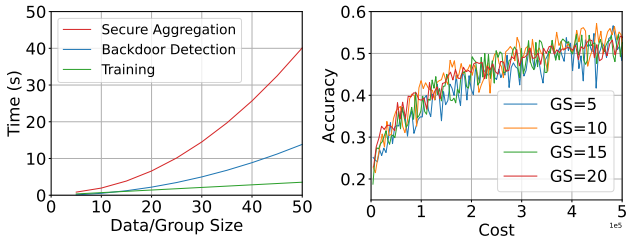


Figure 1: Group-based federated edge learning (Group-FEL) via the client-edge-cloud architecture.

1 INTRODUCTION

Hierarchical federated learning (HFL) [1, 2] has been studied as a more practical federated learning (FL) [3] paradigm in terms of scalability, efficiency, robustness, privacy, etc. Federated learning, whose fundamental purpose protecting users' data privacy, is actually born hierarchical [2]: clients are generally divided into groups to reduce the communication and computation costs incurred by secure aggregation [4]. As edge servers can greatly improve the scalability, connection stability, and system robustness, it is natural to deploy the HFL framework in the client-edge-cloud architecture, as shown in Fig. 1. In this paper, we consider group-based federated edge learning (Group-FEL) over such an HFL framework. First, each edge server manages a set of clients and groups them based on a specific policy. Such group information is sent to the cloud. The cloud adopts a probabilistic group sampling from all groups to select a few client groups to perform HFL at each global round. The clients within a selected group download the global model and train it with their own datasets, and then send local updates to the edge server for the group aggregation. Group operations, such as secure aggregation and backdoor detection, occur during group aggregation. Each group performs this in-group workflow in multiple rounds before sending the group updates to the cloud aggregator. The cloud aggregator performs the final global aggregation and then sends the latest global model back to the edge servers and mobile clients. Note that when there is only one group on each edge server, the system degrades to a typical client-edge-cloud HFL.

Group-FEL can gain benefits from the edge network due to the low communication cost and more stable connection. However, group operations may still incur high costs, which have not yet been considered by previous works. Fig. 2(a) shows the overheads



(a) Group Overheads (b) Group Size (GS)

Figure 2: Overhead of Group-FEL: (a) different group overheads: for training cost, x-axis is training data number, for secure aggregation and backdoor detection, x-axis is group size; (b) accuracy over cost with different group sizes.

of a client via a real-world measurement of a group-based FEL over Raspberry PI based edge systems. It plots three types of overheads: training cost, secure aggregation cost, and backdoor detection cost. It is obvious that the overhead of these group operations can be comparable to or even significantly exceeds the training cost. Considering the amount of data owned by clients is typically small, when the group size is large, the overheads of group operations overwhelmingly dominate the costs. This is critical for HFL using mobile or IoT devices as clients, due to their limited resources. Simply reducing the group size, however, may not work. As shown in Fig. 2(b), even we reduce the group size from 20 to 5, the total cost (defined as Equ. (5)) needed to achieve a certain accuracy stays similar. Here, the plotted cost is the total cost during the whole FL training period including training cost and group operation cost.

The reason why a smaller group size does not always lead to total cost reduction is that the data within smaller client groups are usually more skewed, which hinders the convergence, thus causing high costs. As shown by many FL works [5–12], the non-IID (non-Independent and Identically Distributed) issue hurts the FL convergence. For Group-FEL, we first theoretically derive its convergence rate and show that the data distribution within groups (i.e., the IID degree of the group data) indeed affects the training convergence in theory. Based on such an observation, we introduce a new grouping method (i.e., CoV-Grouping), which leverages the coefficient of variation (CoV) to form client groups. Instead of simply playing trade-offs between the group size and the IID degree (or between cost and accuracy), our group formation method generates smaller groups with less skewed data. In that way, it is possible to form groups that are beneficial to convergence and are less costly. In addition, we further propose different CoV-based sampling methods to calculate group sampling probabilities, so that priority is given to groups with better CoV values. Our results also show that CoV is effective in judging the quality of a group. This observation is also useful for other HFL-based methods. Finally, we compare our proposed method with existing non-IID countering methods via extensive experiments, including the training method based approaches (FedProx [6] and SCAFFOLD [7]), client assignment based approaches (OUEA [13] and SHARE [14]), and a personalized FL approach (FedCLAR [12]). Our results confirm the advances of our proposed method over these existing methods in a group-based HFL setting. To the best of our knowledge, this is the first work that considers the impact of group overhead in FL, and offers a pioneer yet comprehensive exploration, including theoretical analysis, group formation, and a new sampling strategy specifically designed for Group-FEL. We hope this works inspires more future investigations into this important but ignored problem.

In short, our contributions are summarized as follows:

- We introduce a general group-based hierarchical federated edge learning framework (Group-FEL) where edge servers perform client grouping and the cloud performs probabilistic group sampling. (Section 3)
- We provide a theoretical convergence analysis of Group-FEL with emphasis on the quality of group data distribution, and discovery that the group heterogeneity plays an important role in the convergence. This general result applies to all generic HFL systems[1]. (Section 4)
- We design a new group formation algorithm based on the group's coefficient of variation (CoV), to generate groups with better data distribution, thus speeding up convergence and reducing costs. (Section 5)
- We also propose several group sampling strategies to sample groups with better distribution. The results also shed light on other group/cluster sampling methods. (Section 6)
- Extensive experiments are conducted to demonstrate the effectiveness of the proposed group formation algorithm, sampling strategies, as well as the training result of the whole system. (Section 7)

2 RELATED WORKS

2.1 Non-IID in Federated Learning

Recently, the Non-IID problem in FL has been well-studied in prior research such as [5–10, 12, 15]. These works can be categorized into two types based on how they address statistical heterogeneity and their corresponding objectives. The first type of work, such as [5–7], aims to reduce the negative impact of objective inconsistency caused by skewed data and obtain a global model that performs well in the overall task (where there is no data distribution skew). The other type of work recognizes that heterogeneity reflects the different natures and demands of different clients, thus they focus on training multiple personalized models designed for different users, instead of training a good global model. This kind of method is also called *personalized FL*. In general, if the goal is to extend the available data to generate a general model, the first approach may be used. If the goal is to improve the immediate user experience, the personalized approach may be more suitable.

To train a global model that works well for the overall task, we need to strive to remedy the negative impact of inconsistent local objectives. The pioneer work mainly uses training-based methods, such as augmenting the data, modifying the loss function, and changing the descent direction. Zhao *et al.*[5] utilize a globally shared dataset to bootstrap and reduce the label distribution skew among clients. FedProx [6] limits the divergence of local training from the last global model to mitigate inaccurate updates. SCAFFOLD [7] records the direction of local and global gradient to re-direct updates to an estimated correct direction. CCVR [11] calibrates the classifier layer using sampled virtual representations,

[1]Our theoretical analysis does not rely on a specific grouping method, thus is general enough to cover any generic group-based HFL. Furthermore, since we consider both the quality of the group data distribution and the probability of sampling the group, our result is different from the existing HFL convergence analysis (e.g., [1] and [13]).

while DFL [16] tries to separate global and local-only features to perform alternative local-global optimization for both generalized adaption and personalized performance. For personalized FL, as suggested by [17], there are many methods, including but not limited to transfer learning [12], meta-learning [18], knowledge distillation[19], and clustering [12].

This paper focuses on the first type of approach to train a global model and our proposed method outperforms existing methods in the HFL environment. We choose the two most popular methods FedProx and SCAFFOLD as part of the baselines in experiments. Although personalized FL methods do not align well with our target scenario, we also include FedCLAR [12] in experiments to confirm that this type of solution is not suitable in our considered scenario.

2.2 Hierarchical Federated Learning (HFL)

HFL has been studied recently due to advances in the scalability and privacy protection of FL. A cloud-edge-client HFL framework has been studied in [1] and it gives the convergence analysis of HFL. Wang *et al.* [20] also considered cluster structure formation in HFL where edge servers are grouped in different clusters for model aggregation. While both [1] and [20] provide the HFL convergence analysis, their analysis is different from ours, since they do not consider (1) group sampling (i.e., the sampling fraction for each group) and (2) group characters (such as the variance of data number among clients and groups).

To handle the non-IID problem in HFL, OUEA [13] and SHARE [14] consider the client assignment problem: they try to assign clients to edge servers such that the clients associated with each edge server have a balanced data distribution when virtually combined. OUEA [13] first clusters similar clients together and then distributes them to different groups so that the data distribution inside each group tends to be IID. OUEA also offers a convergence analysis, but similar to [1], they do not consider group sampling and some group-specific characters. Furthermore, OUEA requires the loss functions to be convex (which is not true for neural networks) while our analysis does not. SHARE [14] also considers data distribution among edge aggregators and optimizes communication cost during the client-edge assignment. It uses Kullback–Leibler divergence (KLD) to measure the IID degree among edge aggregators. Both OUEA and SHARE consider each edge server as one single aggregator (i.e., aggregating one client group) and do not control the group size. In the experiments, we port their assignment policy to the group formation and re-implement their algorithms for comparison with our grouping method.

All of the works above consider the hierarchical structure and/or the communication cost, but none of them takes into account the **overhead incurred by group operations** (especially those for security and privacy protection operations). As a result, they do not limit the number of clients associated with an edge aggregator (within a group), which may not be cost-efficient, as shown in Fig. 2. Additionally, some efforts focused on privacy protection and participant selection in HFL. Wainakh *et al.* [21] pointed out the gain of HFL on privacy enhancement. Yang *et al.* [22] proposed a compression mechanism and gradient clipping method in an HFL architecture to reduce communication overhead and protect privacy. Wei *et al.* [23] studied the participant selection problem of a multi-model HFL. Our work instead focuses on group formation

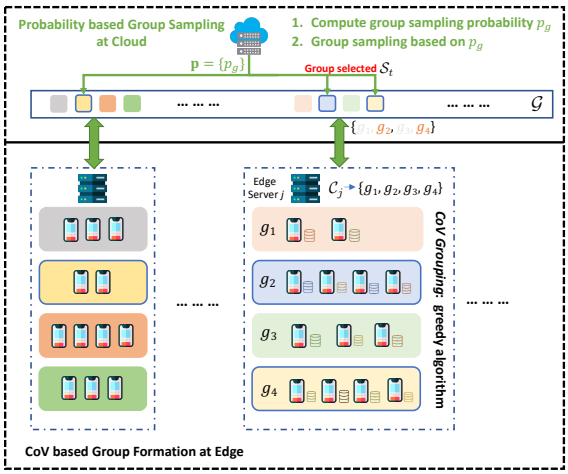


Figure 3: The overall grouping framework in Group-FEL: (lower) *CoV-based group formation* at the edge, and (upper) *probability based group sampling* at the cloud.

and sampling in a group-based HFL to reduce the total learning cost (including overhead caused by group operations).

2.3 Performance Measurement

Most existing works measure convergence by iteration, which is not effective in many cases. For example, some algorithms [6, 7] require more computation and/or communication in each round to achieve faster convergence regarding global iterations but may be slower when measured by wall clock time and/or resource cost. Therefore, more realistic measurements for FL systems have been investigated to satisfy different application requirements. Luo *et al.* [24] seek to reduce the training wall clock time. They propose a new convergence upper bound for arbitrary client selection probabilities and generate a non-convex training time minimization problem. Their approach significantly reduces the convergence time to achieve the same target loss compared to several baselines regarding the wall-clock time. Yang *et al.*. [25] study the energy consumption optimization problem for battery-sensitive devices and the proposed method can save up to 59.5% energy. Some works [26, 27] notice that communication traffic may also be a potential bottleneck for cross-device FL systems, and hence seek to reduce the bandwidth requirement by gradient/model compression. They compare the convergence rates by loss over total network traffic. In this paper, we consider all costs incurred by training and group operations and unify the latter into one quadratic function. We will then evaluate the performance of FL systems in terms of accuracy over this generally defined cost.

3 GROUP-BASED FEL (GROUP-FEL)

3.1 System Model and Learning Procedures

In this paper, we consider a group-based hierarchical federated learning over edge computing, as shown in Fig. 1. We assume that multiple mobile clients (let C be the client set) are connected to cloud via edge servers. Each edge server will divide its clients (C_j, the client set of j-th edge server) into multiple mutually exclusive client groups. Let $\mathcal{G}$ and $\mathcal{G}_j$ be the set of all groups and the set of groups of j-th edge server, respectively. Then the federated learning is performed with selected client groups (denoted by $\mathcal{S}_t$) based on

Algorithm 1 Group-based Federated Edge Learning

Input: Client sets C_j of each server, number of sampled groups in each round $S = |\mathcal{S}_t|$, initial global model x_0, global round T, group round K, local round E, learning rate η.
Output: Final global model x_{T-1}.

1: $\mathcal{G} = \emptyset$
2: **for** each client set C_j **do** ▷ *in parallel*
3: $\mathcal{G} = \mathcal{G} \cup$ CoV-Grouping(C_j) ▷ *group formation*
4: $\mathbf{p}$ =Sampling-Prob$(\mathcal{G})$ ▷ *group sampling prob*
5: **for** t from 0 to $T-1$ **do**
6: Sample $\mathcal{S}_t \subseteq \mathcal{G}$ according to $\mathbf{p}$ ▷ *group sampling*
7: **for** group $\mathbf{g}$ in $\mathcal{S}_t$ **do** ▷ *in parallel*
8: $x^g_{t,0} = x_t$ ▷ *initialize group model*
9: **for** k from 0 to $K-1$ **do**
10: **for** client c_i in group g **do** ▷ *in parallel*
11: $x^i_{t,k,0} = x^g_{t,k}$ ▷ *initialize client model*
12: **for** e from 0 to $E-1$ **do**
13: $x^i_{t,k,e+1} = x^i_{t,k,e} - \eta \nabla f_i(x^i_{t,k,e}; \xi^i_{t,k,e})$ ▷ *local update*
14: $x^g_{t,k+1} = \sum_{i \in g} \frac{n_i}{n_g} x^i_{t,k,E-1}$ ▷ *group aggregation*
15: $x_{t+1} = \sum_{g \in \mathcal{S}_t} \frac{n_g}{n_t} x^g_{t,K-1}$ ▷ *global aggregation*
16: **return** x_{T-1}

a certain selection mechanism (i.e., via group sampling with a probability vector $\mathbf{p}$) at each global round t. Each client c_i in a selected group performs local training and sends its local model updates to its edge server for group aggregation. Then edge servers will perform group aggregation and submit the group model updates to the cloud server (i.e., parameter server) for global aggregation. The overall training algorithm of group-based FEL is shown in Algorithm 1. Fig. 3 illustrates the overall grouping framework of Group-FEL, where the grouping is performed at the edge server for its clients, and the group sampling is done at the cloud with the sampling probability vector.

In Algorithm 1, Lines 2-3 are for group formation at each edge server, and Line 4 is for the computation of sampling probability vector $\mathbf{p}$ of all groups at the cloud. These are important steps for Group-FEL, thus we will present our detailed design of them in Sections 5 and 6, respectively. Lines 5-15 are the group-based federated learning steps, which include group sampling (Line 6), local update (Line 13), group aggregation (Line 14), and global aggregation (Line 15). Here, x_t, $x^g_{t,k}$, $x^i_{t,k,e}$ represent the global model at t-th global round, the group model at k-th group round within t-th global round, the local model of client c_i at e-th local round within k-th group round and t-th global round, respectively.

In classic federated learning, given the client set C with N clients, and the loss function f_i of the client c_i, we have the global loss function

$$f(x) = \sum_{c_i \in C} \frac{n_i}{n} f_i(x), \tag{1}$$

where n_i is the data entry number on i-th client, and $n = \sum_{i=0}^{N-1} n_i$.

When we divide clients into a set of groups $\mathcal{G}$, then for each group $\mathbf{g}$, its loss function is

$$f_g(x) = \sum_{c_i \in \mathbf{g}} \frac{n_i}{n_g} f_i(x), \tag{2}$$

where n_g is the number of data on all clients inside the group $\mathbf{g}$. Hence, the global loss function can be rewritten as

$$f(x) = \sum_{\mathbf{g} \in \mathcal{G}} \frac{n_g}{n} f_g(x). \tag{3}$$

At Line 15 of Algorithm 1, the global aggregation may lead to the learned model biased since some groups have higher probability to be sampled during the group sampling. This is true for our design since we always give higher priority to groups with better distribution to boost convergence. Therefore, we will discuss this in Section 6. If the model is required to be unbiased, a correction factor $\frac{1}{p_g S}$ can be introduced and Line 15 is then replaced by

$$x_{t+1} = \sum_{g \in \mathcal{S}_t} \frac{1}{p_g S} \cdot \frac{n_g}{n} x^g_{t,K-1}, \tag{4}$$

where p_g is the probability to sample the group $\mathbf{g}$ during the group sampling and S is the number of sampled groups in each round $S = |\mathcal{S}_t|$. Note that n_t in Line 15 of Algorithm 1 in the number of data entries on the t-th global round.

3.2 Cost Model

To measure the Group-FEL learning cost, we focus on both computation and communication loads on all clients. Each client has two types of costs: training cost and group operation cost. The built-in training cost $\mathcal{H}_i(n_i)$ of client c_i measures the time needed to iterate through its trainset once. Given the hardware, model, and training hyperparameters are fixed, this cost is proportional to the data sample number n_i owned by this client. Overheads incurred by group operations (both for secure or privacy-preserving computation and communication) are quadratic to the group size $|\mathbf{g}|$ [4, 28]. These two assumptions can be further confirmed by our experiments shown in Fig. 8. Hereafter, we use $O_g(|\mathbf{g}|)$ to denote the group overhead of each client in group $\mathbf{g}$. Note that this group overhead cost is often ignored in the analysis of existing works. By adding training costs and group operation costs of all clients in each group, the total learning costs in the whole training process can be measured by

$$O = \sum_{t=0}^{T-1} \left(\sum_{\mathbf{g} \in \mathcal{S}_t} K \sum_{c_i \in \mathbf{g}} (O_g(|\mathbf{g}|) + E\mathcal{H}_i(n_i)) \right). \tag{5}$$

In our evaluations, we will measure the performance of all algorithms using the achieved accuracy by certain learning costs instead of the accuracy by the global round.

4 CONVERGENCE ANALYSIS

In this section, we will present our main theorem on the convergence of Group-FEL with an emphasis on the group characters. This result is an important theoretical contribution and also inspires the design of our grouping formation and sampling schemes. The result applies to all HFL structures where an intermediary aggregation layer is used. The result is also general enough to cover existing convergence results[2].

[2]When $|\mathcal{S}_t| = |\mathcal{G}|$, it degrades to HFL without group sampling. When there is only one group on each edge server, it degrades to the classic HFL.

4.1 Assumptions

To boost theoretical analysis of convergence, previous works [29–31] have made common assumptions about the local and global loss functions. We simply borrow and list them here.

Assumption 1. ***Bounded variance of local gradient:*** *for any local loss function f_i,*

$$\|\nabla f_i(x;\xi^i) - \nabla f_i(x)\|^2 \le \sigma^2. \tag{6}$$

Here ξ^i denotes the data used to compute the gradient at client c_i in a certain round. Clients may not always use all data to calculate the gradient, thus there is a variance compared to the full gradient.

Assumption 2. ***L-smoothness:*** *for any local loss function f_i (also global loss function f),*

$$\|\nabla f_i(x) - \nabla f_i(y)\| \le L\|x-y\|. \tag{7}$$

Assumption 3. ***Bounded local heterogeneity:*** *for any local loss function f_i,*

$$\|\nabla f_i(x) - \nabla f(x)\|^2 \le \zeta^2. \tag{8}$$

Assumption 4. ***Bounded group heterogeneity:*** *the heterogeneity between any group loss function f_g and global loss function f satisfies*

$$\|\nabla f_g(x) - \nabla f(x)\|^2 \le \zeta_g^2. \tag{9}$$

Here ζ_g is a constant that measures the heterogeneity between any f_g and f. There is no practical way to compute ζ_g and L but it is generally believed that ζ_g relies on the difference between the global and group data distributions, i.e., the more similar the two distributions are, the smaller ζ_g is. Note that although ζ_g and ζ are both on heterogeneity they are quite different in our design. ζ reflects the heterogeneity caused by individual clients which cannot be controlled, while ζ_g is the heterogeneity of the formed groups. In Group-FEL, if we can **control the group formation smartly to reduce** ζ_g, then the selected groups will be more IID, thus leading to better performance.

4.2 Main Convergence Result

Theorem 1. *The convergence rate of Group-FEL is bounded as follows,*

$$\frac{1}{T}\sum_{t=0}^{T-1}\|\nabla f(x_t)\|^2 \le \frac{f(x_0)-\mathbb{E}[f(x_T)]}{\lambda_1\eta TKE} + \frac{\lambda_s\cdot\frac{\Gamma_p}{|\mathcal{S}_t|}}{\lambda_1 TKE} + \frac{\gamma\Gamma(\lambda_2\sigma^2+\lambda_3\zeta^2+\lambda_4\zeta_g^2)}{\lambda_1 T}. \tag{10}$$

Here γ, Γ, Γ_p and constants $\lambda_1, \lambda_2, \lambda_3, \lambda_4, \lambda_s, \lambda_\sigma, \lambda_f$ are defined or constrained by following (to simplify the expression of the final result in the proofs)

$$\gamma = |\mathbf{g}|^2\left[\frac{1}{|\mathbf{g}|^2} + Var\left(\frac{n_i}{n_g}\right)\right], \tag{11}$$

$$\Gamma = |\mathcal{G}|^2\left[\frac{1}{|\mathcal{G}|^2} + Var\left(\frac{n_g}{n}\right)\right],\ \ \Gamma_p \ge \sum_{\mathbf{g}\in\mathcal{G}}\frac{1}{p_g}, \tag{12}$$

$$\lambda_s = \eta\gamma\Gamma K^2(1+10\eta^2E^2L^2\sigma^2), \tag{13}$$

$$0 < \lambda_1 \le \frac{1}{2} - 3\lambda_f\eta\gamma\Gamma KEL^2, \tag{14}$$

$$\lambda_2 = 3\lambda_\sigma\gamma L^2 + 5\eta^2E^2L^2,\ \ \lambda_3 = 2700\eta^4\gamma K^2E^4L^2, \tag{15}$$

$$\lambda_4 = 90\eta^2K^2E^2L^2,\ \ \lambda_f = 30\eta^2K^2(1+90\gamma\eta^2E^2L^2), \tag{16}$$

$$\lambda_\sigma = 5K\eta^2E^2\left[1+((1+6K)E+9K)10\eta^2EL^2 + \frac{18K}{|\mathbf{g}|E}\right], \tag{17}$$

$$0 \ge \eta^2 - \frac{\eta}{2KE}. \tag{18}$$

Proof. Due to the space limit, we cloud not include all proof details. Here we only present a brief proof skeleton, and more details will be provided as a technical report. First, we consider another form of Assumption 2, as

$$f(y) \le f(x) + \langle\nabla f(x), (y-x)\rangle + \frac{L}{2}\|x-y\|^2. \tag{19}$$

Based on this smoothness assumption, we can have

$$\mathbb{E}\left[f(x_{t+1})\right] \le f(x_t) - \eta KE\|\nabla f(x_t)\|^2 + \frac{L}{2}\mathbb{E}_t\|\Delta_t\|^2 + \langle\nabla f(x_t), \mathbb{E}_t\left[\Delta_t + \eta KE\nabla f(x_t)\right]\rangle, \tag{20}$$

where $\Delta_t = x_{t+1} - x_t = \sum_{\mathbf{g}\in\mathcal{S}_t}\frac{n_g}{n_t}\sum_{k=0}^{K-1}\eta\nabla F_g(x_{t,k})$ is the global update at the round t and $F_g(x) = \sum_{i\in\mathbf{g}}\frac{n_i}{n_g}\sum_{e=0}^{E-1}f_i(x_e)$ is single round group update.

By defining $A_1 = \langle\nabla f(x_t), \mathbb{E}_t\left[\Delta_t + \eta KE\nabla f(x_t)\right]\rangle$ and $A_2 = \mathbb{E}_t\|\Delta_t\|^2$, we can rewrite Equ. (20) as

$$\mathbb{E}\left[f(x_{t+1})\right] \le f(x_t) - \eta KE\|\nabla f(x_t)\|^2 + A_1 + \frac{L}{2}A_2. \tag{21}$$

Then we bound A_1 and A_2 by proving Lemma **??** and Lemma 2, respectively. Bringing such bounds into Equ. (21), we then have

$$\mathbb{E}\left[f(x_{t+1})\right] \le f(x_t) - \eta KE(\frac{1}{2} - 3\lambda_f\eta\gamma\Gamma KEL^2)\|\nabla f(x_t)\|^2 + \lambda_2\eta KE\gamma\Gamma\sigma^2 + \lambda_3\eta KE\gamma\Gamma\zeta^2 + \lambda_4\eta KE\gamma\Gamma\zeta_g^2 + \frac{\eta^2\gamma\Gamma\Gamma_pK^2}{|\mathcal{S}_t|}(1+10\eta^2E^2L^2\sigma^2) + (\eta^2 - \frac{n}{2KE})\mathbb{E}_t\left[\|\sum_{g\in\mathcal{G}}\frac{n_g}{n}\sum_{k=0}^{K-1}\nabla F_g(x_{t,k}^g)\|^2\right]. \tag{22}$$

With conditions of Equ. (13), (14), and (18), we can show

$$\mathbb{E}\left[f(x_{t+1})\right] \le f(x_t) - \lambda_1\eta KE\|\nabla f(x_t)\|^2 + \eta\gamma\Gamma KE(\lambda_2\sigma^2+\lambda_3\zeta^2+\lambda_4\zeta_g^2) + \eta\cdot\lambda_s\cdot\frac{\Gamma_p}{|\mathcal{S}_t|}. \tag{23}$$

Finally, by rearranging and telescoping, we have

$$\frac{1}{T}\sum_{t=0}^{T-1}\|\nabla f(x_t)\|^2 \le \frac{f(x_0)-\mathbb{E}[f(x_T)]}{\lambda_1\eta TKE} + \frac{\lambda_s\cdot\frac{\Gamma_p}{|\mathcal{S}_t|}}{\lambda_1 TKE} + \frac{\gamma\Gamma(\lambda_2\sigma^2+\lambda_3\zeta^2+\lambda_4\zeta_g^2)}{\lambda_1 T}.$$

□

Lemma 2. *Under the assumptions and conditions in Theorem 1, A_1 is bounded as follows*

$$A_1 \le \eta KE(\frac{1}{2} + 90\eta^3\gamma\Gamma K^3E^3L^2)\|\nabla f(x_t)\|^2 + \lambda_2\eta KE\gamma\Gamma\sigma^2 + \lambda_3\eta KE\gamma\Gamma\zeta^2 + \lambda_4\eta KE\gamma\Gamma\zeta_g^2 - \frac{\eta}{2KE}\mathbb{E}_t\left[\|\sum_{g\in\mathcal{G}}\frac{n_g}{n}\sum_{k=0}^{K-1}\nabla F_g(x_{t,k}^g)\|^2\right]. \tag{24}$$

Lemma 3. *Under the assumptions and conditions in Theorem 1, A_2 is bounded as follows*

$$A_2 \le \eta^2\mathbb{E}_t\left[\|\sum_{g\in\mathcal{G}}\frac{n_g}{n}\sum_{k=0}^{K-1}\nabla F_g(x_{t,k}^g)\|^2\right] + \frac{\eta^2\gamma\Gamma\Gamma_pK^2}{|\mathcal{S}_t|}(1+10\eta^2E^2L^2\sigma^2). \tag{25}$$

Proofs of these two lemmas are ignored due to the space limit.

Recall that T, K, E, and η are the number of global rounds, the number of group rounds in each global round, the number of local rounds in each group round, and the learning rate in local updating,

respectively, as defined in Algorithm 1. The inequality (Equ. 10) tells us that when the right-hand side is diminishing as T increases (more global rounds), the gradient norm $\|\nabla f(x_t)\|$ tends to be zero, which means the model converges to a local minimum.

4.3 Key Observations

From the main theorem, we have the following observations.

First, the heterogeneity ζ_g between the combined group loss function f_g and the global loss functions f plays a role in convergence. With a larger heterogeneity, the convergence will be slower. Therefore, in our proposed group formation and group sampling schemes, we aim to reduce this heterogeneity. Unfortunately, the definition of ζ_g is not straightforward and we cannot directly compute it to quantify the heterogeneity. Therefore, instead, we use the difference between data distributions to measure how analogous two loss functions are. Concretely, we use the *Coefficient of Variance* (CoV) of the labels in a group, which will be discussed in the next section. The CoV-based grouping algorithm bridges this observation to our system design.

Second, the larger variance of sampling vector $\mathbf{p}$ (thus larger Γ_p) may also delay the convergence of unbiased sampling. Due to the unbiasedness factor $\frac{1}{p_g|\mathcal{S}_t|}$, any $\frac{1}{p_g}$ should not be too large, otherwise the aggregation is numerically unstable: $\frac{1}{p_g}$ extremely amplifies the gradient and ruins all previous training results. On the other hand, we want to be able to set arbitrary $\mathbf{p}$ to prioritize good groups with smaller ζ_g. To handle this, when we adopt unbiased aggregation together with prioritized sampling, we will use a normalization method (see Section 6). Note that $\mathbf{p}$ may also entangle with ζ_g, γ, and other group-specific characters as it decides which groups participate in the training more, and therefore their characters affect the FL system more.

Last, to boost convergence, we need a smaller γ. We find that $\gamma - 1 = |\mathbf{g}|^2 Var(\frac{n_i}{n_g}) = (\frac{\sigma_c}{\mu_c})^2$, where σ_c and μ_c are the standard deviation and mean of the total data sample number among clients within the same group. Interestingly, $\gamma - 1$ is also the square of CoV of data sample number within the group. Γ has a similar property. Reducing γ also helps to converge faster and smoother. We leave further considering this in our design as one of the future works.

5 GROUP FORMATION

In Group-FEL, how to perform group formation is critical since ζ_g plays an important role in convergence (as suggested by the first key observations). Again, note that ζ_g depends on the difference between the in-group data distributions and the global data distribution, and a good group formation method helps to reduce ζ_g. Therefore, in this section, we first discuss the possible grouping criteria and then present our proposed grouping method.

5.1 Grouping Criteria: CoV

Based on the key observation from our convergence analysis, the principle of grouping criteria should be to make the group loss functions as similar to the global loss function as possible (i.e., similar data distribution and smaller ζ_g). In general, the properties of a loss function are closely related to its data distributions. Therefore, our grouping criteria aims to measure the similarity of data distributions between group and global. We further assume the global data are evenly distributed, thus we can just focus on the data distribution within each local group. To do so, we introduce the *coefficient of variation* (**CoV**) of the labels in a group.

Here we focus on the grouping of a client set $\mathcal{K}$, whose i-th client is c_i. The data label set $\mathcal{Y}$ contains m kinds of labels. We define a label matrix $\mathcal{L}$, where $\mathcal{L}_{i,j}$ is the number of j-th category of data samples on i-th client. Then, a grouping $\mathcal{G}$ of $\mathcal{K}$ is a partition of $\mathcal{K}$. Let $\mathcal{G}_l$ be the l-th group in $\mathcal{G}$, which contains all clients in this group. To compute the CoV of a group, we only need to know the data label distributions from users in that group, without any information of their local data, model, nor gradient.

Ideally, we would like the distribution of every group $\mathcal{G}_l$ is identical to the global distribution, i.e.,

$$\frac{\sum_{c_i \in \mathcal{G}_l} \mathcal{L}_{i,j}}{\sum_{c_i \in \mathcal{G}_l} \sum_{k \in \mathcal{Y}} \mathcal{L}_{i,k}} = \frac{\sum_{c_i \in \mathcal{K}} \mathcal{L}_{i,j}}{\sum_{c_i \in \mathcal{K}} \sum_{k \in \mathcal{Y}} \mathcal{L}_{i,k}}, \quad \forall j, \forall l. \tag{26}$$

However, such restricted criteria might lead to infeasible grouping. Thus, instead, we use the coefficient of variation as the grouping criterion. For a given group $\mathbf{g}$, we calculate its coefficient of variation (CoV) in the following way

$$CoV(\mathbf{g}) = \frac{\sigma(\mathbf{g})}{\mu(\mathbf{g})} = \frac{\sqrt{\sum_{j \in \mathcal{Y}} \left(\frac{n_g}{m} - \sum_{c_i \in \mathbf{g}} \mathcal{L}_{i,j}\right)^2}}{n_g}. \tag{27}$$

$$\sigma(\mathbf{g}) = \frac{\sqrt{\sum_{j \in \mathcal{Y}} \left(\frac{n_g}{m} - \sum_{c_i \in \mathbf{g}} \mathcal{L}_{i,j}\right)^2}}{m}. \tag{28}$$

Recall that n_g is the number of data samples in the group and m is the number of labels (data samples types). The reason why the variance (i.e., $\sigma^2(\mathbf{g})$) is not suitable as the criterion is that it is susceptible to the scale of data number. For example, a group with a smaller total data number but larger data distribution skew may have a smaller variance than a group with more data but smaller distribution skew. We may prefer the latter group but, on the contrary, the smaller variance criterion prefers the first one. Note that neither variance nor CoV has been considered in previous works on client grouping at edge learning.

5.2 Group Formation Problem

The group formation problem aims to divide all clients associated with an edge server into multiple client groups such that the summation of CoVs of all groups is minimized. We can use matrices to more succinctly express this grouping problem. Suppose we have three matrices A, X, and B, where A_{ji} is the number of label type j that client c_i possesses; X_{il} is the grouping decision indicator where $X_{il} = 1$ if the client c_i is in the group $\mathcal{G}_l$, otherwise 0; B_{jl} is the number of data type j in group l. Then the group formation problem can be formulated as the following optimization problem:

$$\min_X \sum_l \frac{\sqrt{\sum_j \left(\frac{\sum_j B_{jl}}{m} - B_{jl}\right)^2}}{\sum_j B_{jl}} \tag{29}$$

$$\textbf{s.t.} \quad AX = B, \tag{30}$$

$$\sum_i X_{il} \geq MinGS, \quad \forall l, \tag{31}$$

$$\sum_l X_{il} = 1, \quad \forall i, \tag{32}$$

$$X_{il} \in \{0, 1\}, \quad \forall i, \forall l. \tag{33}$$

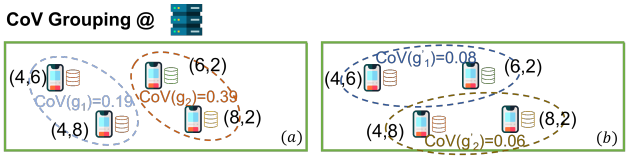


Figure 4: Example of CoV-Grouping at edge.

Constraint (31) is an anonymity constraint to make sure that each group at least has *MinGS* clients[3], while Constraint (32) ensures that a client will be grouped into one and only one group.

Fig. 4 illustrates a simple toy example. This edge server has four clients and all wants to form a group with at least two clients. As shown in the figure, though two different grouping methods can both formulate two groups each with two clients, the total CoVs are quite different. In our design, we would prefer the one with lower CoVs (Fig. 4(b)).

5.3 Grouping Algorithm: CoV Grouping

Directly solving the above grouping optimization is not easy. Note that the grouping problem with a fixed number of groups is a variation of the k-mean clustering problem, which is known as NP-hard [32, 33]. Therefore, we design a greedy algorithm (CoV-Grouping) to generate an approximating solution. It generates groups one by one until no more groups are possible. For each group, it first randomly picks a client, then greedily adds clients one by one. When adding a new client to the current group, it tries every possible client and adds the one that reduces the group CoV the most (Line 5). If no one meets this criterion and the group size is large enough (reach *MinGS*), then this group is finalized and the algorithm starts the next group. In addition, besides checking the group size constraint *MinGS*, we also add a maximum CoV requirement (*MaxCoV*) which makes sure the resulting group CoV is smaller than *MaxCoV*[4]. Algorithm 2 shows the details.

Algorithm 2 CoV-Grouping

Input: Client set $\mathcal{K}$, min group size *MinGS*, and *MaxCoV*.
Output: Group set $\mathcal{G}$.

1: $\mathcal{G} = \emptyset$
2: **while** $\mathcal{K} \neq \emptyset$ **do**
3: Find a random $c \in \mathcal{K}$, $\mathbf{g} = \{c\}$, $\mathcal{K} = \mathcal{K} \backslash c$. ▷ *a new group*
4: **while** (CoV(g) > *MaxCoV* or |g| < *MinGS*) and $\mathcal{K} \neq \emptyset$ **do** ▷ *not meet the group requirement yet*
5: Find $c \in \mathcal{K}$ that minimizes CoV($\mathbf{g} \cup c$)
6: **if** CoV($\mathbf{g} \cup c$) < CoV(g) or |g| < *MinGS* **then**
7: $\mathbf{g} = \mathbf{g} \cup c$, $\mathcal{K} = \mathcal{K} \backslash c$ ▷ *add c to current group*
8: **else** ▷ *no suitable c and enough group size*
9: **break** ▷ *finalize current group*
10: $\mathcal{G} = \mathcal{G} \cup \mathbf{g}$ ▷ *add finalized group*
11: **return** $\mathcal{G}$

[3]The minimum group size *MinGS* can make sure that the secure group operation can protect the model/data privacy of its clients. Here we assume the requirement is a controllable constant for our system, but this can be easily extended to the case where each client has its own group size requirement.
[4]This is not a hard constraint, i.e., the algorithm only tries to adding clients until the CoV is satisfied, but sometimes it might be infeasible to reach lower than *MaxCoV*, then it just gives up adding more clients and finalize this group (Line 9 of Algorithm 2).

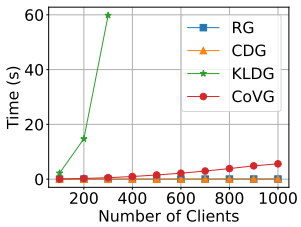


Figure 5: Running time of grouping methods.

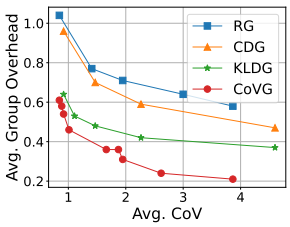


Figure 6: CoV v.s. average group overhead.

The time complexity of Algorithm 2 is $O(|\mathcal{K}|^3|\mathcal{Y}|)$ where $|\mathcal{K}|$ and $|\mathcal{Y}|$ are the number of clients and the number of label types, respectively. First, Line 5 is executed at most $O(|\mathcal{K}|)$ times. Then, in each execution, it implicitly calls $CoV()$ at most $O(|\mathcal{K}|)$ times to try every possible client. Last, the complexity of function $CoV()$ is $O(|\mathbf{g}||\mathcal{Y}|) = O(|\mathcal{K}||\mathcal{Y}|)$. Thus, the total time complexity of CoV-Grouping is $O(|\mathcal{K}|^3|\mathcal{Y}|)$. Given $|\mathcal{Y}|$ is usually a fixed small number for a given task (e.g., 10 for CIFAR-10, 35 for SpeechCommand), the time complexity becomes $O(|\mathcal{K}|^3)$. This is cubic to the client number associated with the edge server but irrelevant to the data amount owned by clients.

5.4 Compared with Other Grouping Algorithms

We now compare our algorithm with two existing solutions: clustering then distribution grouping (CDG, used by OUEA [13]) and KLD grouping (KLDG, used by SHARE [14]), as well as random grouping (RG). Fig. 5 shows the time consumed by each algorithm to group different numbers of clients. We can see that RG can group 1,000 clients at almost no cost (less than 0.3 seconds). CDG has similar efficiency to RG (around 1 second). KLDG is inefficient because i) its time complexity is $O(|\mathcal{K}|^4|\mathcal{Y}|)$; ii) it frequently calculates the KLD, which needs the expensive operation floating-point $log()$. On the contrary, calculating CoV only involves addition and multiplication, which are much cheaper than $log()$. That is why CoVG is able to group 1,000 clients in 6 seconds.

We then compare the quality of the grouping results of these four algorithms. Fig. 6 shows a result more directly related to our concerned question: how do different grouping algorithms affect the learning cost and accuracy? With the same cost (i.e., group overhead), CoVG always gives us the best groups with lower CoV (i.e., higher IID degree), which implies better training accuracy. Similarly, to achieve the same level of CoV (i.e., accuracy), CoVG saves us costs. We will present more details about this implication by experiments in Section 7.

6 GROUP SAMPLING

We now discuss our group sampling method, deployed in the cloud as in Fig. 3, which is a probability-based sampling. Each group $\mathbf{g}$ is sampled based on a probability $\mathbf{p}_g$. The key problem is the sampling criteria, i.e., how to compute the sampling probability, which groups should be sampled more frequently, and how frequent it should be. Existing works in FL already considered many sampling criteria to improve the system performance in specific aspects. For example, [24] considers both training time and gradient in sampling to speed up the training. Since now we have the group CoV, it is reasonable to design new sampling methods based on CoV to improve the system performance. In this section, we discuss

the possible sampling methods and how to better utilize them. As group heterogeneity widely exists in HFL systems, our designs and observations here are also applicable to other HFL systems.

6.1 Sampling Criteria

Obviously, the probability vector $\mathbf{p}$ should satisfy $\sum_g \mathbf{p}_g = 1$. Based on the sampling probability, our system select groups in each global iteration. Let $\mathcal{S}_t$ be the selected group set in the t-th global iteration.

Similar to our group formation method, our grouping sampling method tends to select those groups whose data are combined IID. Recall that $CoV(\mathbf{g})$ is the CoV of the group $\mathbf{g}$, and a larger $CoV(\mathbf{g})$ means a more biased data distribution in this group. Then, we can compute $\mathbf{p}$ in the following way:

$$p_g = \frac{w(\frac{1}{CoV(\mathbf{g})})}{\sum_{\mathbf{g}\in\mathcal{G}} w(\frac{1}{CoV(\mathbf{g})})}, \tag{34}$$

where $w()$ can be a non-decreasing function. The rationale behind this formula is i) $w()$ reflects the importance of each group but CoV means how bad a group is so we should inverse it in $w()$; p_g is the probability so it is in the format of $w()/\sum w()$ (so all of them sum to 1). We find that the choice of $w()$ also has an impact on the result. We consider three choices[5]: $w(x) = x$, x^2, and e^{x^2}, and use RCoV, SRCoV and ESRCoV to denote them. Such sampling methods will be used as Sampling-Prob($\mathcal{G}$) in Algorithm 1 to generate $\mathbf{p}$.

Fig. 7 shows the accuracy achieved by these three sampling methods. Overall, the more we emphasize CoV in sampling, the smoother and faster the convergence is. In this set of experiments, we select 12 groups among 60 client groups based on their CoV values in each round. In general, the more we emphasize CoV, the less frequently those groups with larger CoV are sampled, hence actually less diverse the sampled groups are during the whole training process. It seems that this contradicts better performance with more data. But the key reason is that those groups with larger CoV values can have a smaller or even negative impact on convergence as shown in our theoretical analysis. Such results show a similar implication as the second key observation, i.e., more frequently sampled groups tend to dominate the characters of the whole HFL system. Though we yet can not, and it is hard to rigorously confirm this conjecture. This also confirms that the CoV is capable of properly capturing the distribution skew. In our experiments in Section 7, we use ESRCoV sampling as our default CoV sampling method for our methods, since it has the best performance.

If we would like to utilize the remaining data in those client groups with larger CoV values, one possible solution is regrouping clients (rerunning the group formation algorithm) after a certain number of global iterations. In that case, our design of randomly selecting the first client for each group in CoV-Grouping becomes critical and useful.

6.2 Handling the Unbiased Factor

As aforementioned in Section 4, Γ_p or $(\frac{1}{p_g})$ can be infinitely large especially when $w()$ amplifies the impact of CoV on sampling and the unbiasedness factor is introduced. Meanwhile, to prioritize the good groups, we hope to assign them a much higher probability (as shown in the comparison of different $w()$). Therefore, when we need to adopt the two mechanisms at the same time, the model is likely to diverge. To avoid catastrophic numerical instability, we will use stabilized aggregation, by normalizing the weights in the following way

$$weight(\mathbf{g}) = \frac{\frac{1}{p_g|\mathcal{S}_t|}\frac{n_g}{n}}{\sum_{g\in\mathcal{S}_t}\frac{1}{p_g|\mathcal{S}_t|}\frac{n_g}{n}}. \tag{35}$$

Such weights will replace $\frac{n_g}{n_t}$ at Line 15 of Algorithm 1. Note that we cannot guarantee that the aggregation is still unbiased after using this normalization. Thus, there is always a trade-off. In addition, when the number of selected groups $|\mathcal{S}_t|$ is close to or even larger than the number of good client groups we have, we will have to select some groups with small p_g, then they will dominate the aggregation because they have large $\frac{1}{p_g}$. Therefore, the selection of $|\mathcal{S}_t|$ needs to be carefully set in practice. This can be done before starting the training, by peeking at the grouping result (the sampling probability), as we always have it prior to training.

7 PERFORMANCE EVALUATIONS

In this section, we report detailed performance evaluations of our proposed Group-FEL method via experiments.

7.1 Experiments Setup

Baselines: Upon the selection of baselines, we consider the following three types of related methods: training-based methods (FedProx [6] and SCAFFOLD [7]), client-edge(aggregator) association based methods (CDG from OUEA [13] and KLDG from SHARE [14]), and a clustering method for personalized FL (FedCLAR [12]). Classical FedAvg [3] is also included for reference. Note that the reason why we include FedCLAR is to show that personalized FL is not suitable for training a good global model. CDG and KLDG are originally designed for edge association in HFL, we adopt their basic ideas and port them to group formation algorithms. For a fair comparison, we tune all grouping algorithms so that they tend to generate similar group sizes.

Datasets and ML Models/Tasks: We use both CIFAR-10 [35] and Speech Commands (SC) [34] as our training datasets. CIFAR-10 is a popular image classification dataset containing 10 types of pictures. For this dataset, a 3-block ResNet is used to represent relatively heavy load tasks. The Speech Commands dataset contains 35 types of audio commands and is used for command recognition. For this task, we adopt a 5-layer convolutional neural network (CNN) that is easy to train on RPi to represent lightweight tasks. (as shown in Fig. 8). Both image classification and audio recognition are typical edge AI applications.

Total Cost Emulation: As mentioned in Section 3.B, we evaluate the total learning costs based on Equ. (5). To describe the cost of group operations more accurately, we conduct group-based FEL experiments on RPi 4 devices with both CIFAR10 and SC to extract $\mathcal{O}_g()$ and $\mathcal{H}_i()$ according to the collected measurements. Here, all costs are measured by time. As shown in Fig. 8, the original version of secure aggregation (SecAgg) and SCAFFOLD (+SecAgg) [7] are the most costly operation. We use them to estimate different quadratic cost functions for each method and then emulate the costs

[5]We choose these three functions, since they amplify the impact of CoV from less to more: the first differs but not much from random selection; the last is close to always selecting the groups with the top CoVs; the middle one is between them. Although one may more elaborately select the function, we simply choose these to show how the learning result varies w.r.t. the sampling function, and our results next confirm that they are at least sufficient for our purpose.

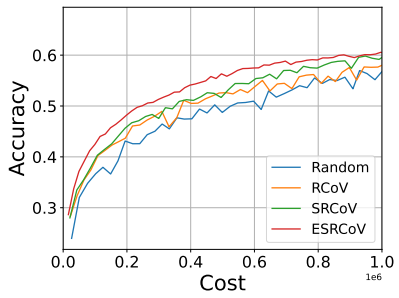


Figure 7: Diff. sampling methods: RCoV, SRCoV, ESRCoV & Random.

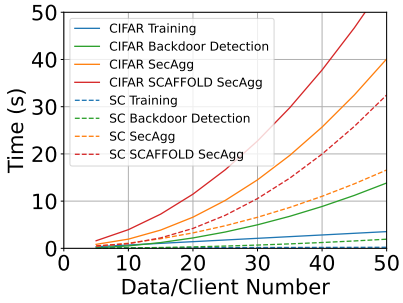


Figure 8: Overhead measurement over Raspberry PI.

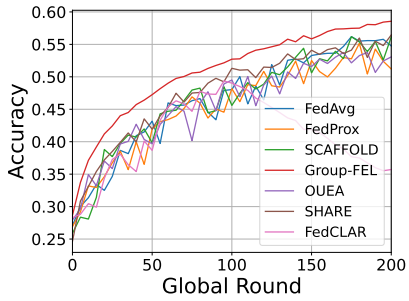


Figure 9: Accuracy vs iteration - all methods over CIFAR-10.

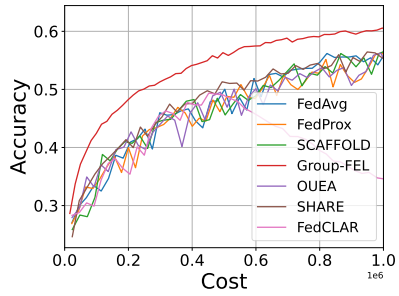


Figure 10: Accuracy vs cost - CIFAR10.

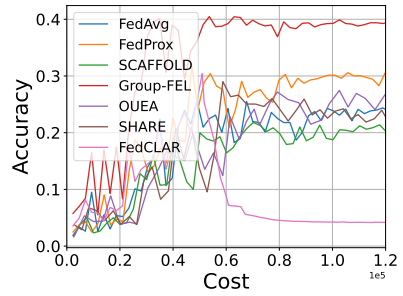


Figure 11: Accuracy vs cost - SC.

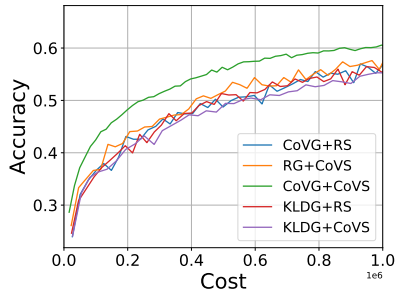


Figure 12: Diff. grouping & sampling.

of group operations in our experiments. Note that other types of group operations can be easily integrated in the future.

HFL Environment: All training and testing of FL models are performed in a virtual environment developed by our group on a server with 40 cores, 512 GB RAM, 8 × NVIDIA Tesla V100. The total costs are computed using $O_g()$ and $\mathcal{H}_i()$ according to the grouping/sampling methods and generated results.

7.2 Performance of Group-FEL

We first test the performance of our design (Group-FEL) with different values of the key hyperparameter *MaxCoV* and different data heterogeneity (i.e., α). We split CIFAR-10 data to 300 clients with 20 to 200 (normal distribution, restricted by the available data of CIFAR-10) data entries each. On each client, the labels follow the Dirichlet distribution with parameter α[6]. We use three edge servers and each of them has 100 clients.

The budget is set as 10^6 unit, which is sufficient for the model to converge. Table 1 shows the detailed performances, including the range and average of group sizes generated, the average group *CoV*, and the achieved accuracy, when $K = 5$, $E = 2$, and $MinGS = 5$. Clearly, with larger *MaxCoV* (that allows more skewed distributions), our method generates smaller groups with larger group *CoV*. Note that smaller group *CoV* (more balanced data) does not necessarily lead to higher accuracy as it may require larger group sizes and hence higher overhead. When the data is more IID (larger α), smaller *MaxCoV* leads to better accuracy because we can now have more IID groups with small sizes. However, when the data is skewed, larger *MaxCoV* may be better. Overall, with less skewed data (larger α), our method can achieve better accuracy.

[6]This is adopted by many previous works on non-IID FL, such as [36]. In general, smaller α means more skewed data.

α	MaxCoV	GS [min,max](avg)	Avg. CoV	Accu
0.1	0.1	[6, 19](10.96)	0.28	56.68%
	0.5	[5, 11](6.13)	0.43	59.80%
	1.0	[5, 6](5.03)	0.54	**60.56%**
0.5	0.1	[5, 11](7.66)	0.19	64.11%
	0.5	[5, 9](5.23)	0.25	63.40%
	1.0	[5,5](5.00)	0.29	**65.02%**
1.0	0.1	[5, 19](6.95)	0.15	**65.08%**
	0.5	[5, 6](5.02)	0.20	64.85%
	1.0	[5, 5](5.00)	0.20	64.45%

Table 1: Performance of Group-FEL: Group Size (GS), Group CoV, and Accuracy for different α and *MaxCoV*.

7.3 Comparison with Existing Methods

7.3.1 Against Baselines over CIFAR-10. Next, we compare our method with the selected baselines, classical FedAvg [3], FedProx [6], SCAFFOLD [7], OUEA[13], SHARE[14], and FedCLAR[12]. For fairness, they are all modified to a hierarchical version (if not originally) with uniform group sampling. FedAvg, FedProx, and SCAFFOLD use random grouping, while FedCLAR uses random grouping at the beginning and then performs its clustering method at a specific round. OUEA uses its CDG algorithm (Algorithm 1 in [13]) and SHARE uses its KLD-based grouping. Fig. 9 shows the results of accuracy over global iterations on CIFAR-10. We can see that our method outperforms all baselines while the baselines do not differ much from each other. Note that the accuracy of FedCLAR drops after clustering since it is designed for personalized FL and is not suitable for training the global model. Fig. 10 shows the same results over the corresponding training cost. Clearly, our method advances even more in this measurement. With the same training cost, our

proposed method can achieve significantly higher accuracy. The reason is that FedProx and SCAFFOLD demand more computation (both) and communication (SCAFFOLD) in each round; OUEA and SHARE, even though we tune their group size, still generate some costly groups as they do not control the group size. Compared with Fig 9, Fig. 10 can illustrate the critical advance of our proposed method over existing FL solutions more clearly.

7.3.2 Performances over Speech Command. We also conduct similar experiments over the Speech Command (SC) dataset, in which there are 35 types of commands. We set $\alpha = 0.01$, which means the data on each client is extremely skewed: the data on each client are mainly dominated by less than 5 types of data. We set $MinGS = 15$ for all and no $MaxCoV$ constraint. Fig. 11 shows the results. Clearly, the convergence is unstable due to the serious inconsistency (large ζ). In general, we can observe similar results as those on CIFAR-10 (our method is the best).

7.3.3 Impacts of Group Formation and Group Sampling. Finally, we investigate the impacts of group formation and group sampling in our proposal methods. Fig. 12 shows different combinations of group formation methods (random grouping (RG), KLD-based grouping (KLDG), and our proposed CoV grouping (CoVG)) and group sampling methods (random sampling (RS) and our proposed CoV sampling (CoVS)). CDG is omitted as it does not show a significant difference from RG. The key observation from this result is the advantage of the proposed methods is more clear when both CoVG and CoVS are used together. When only CoVG is adopted and random sampling is used, the good groups are not prioritized so they do not have much impact on the learning result. Compared with the performance of our method (CoVG+CoVS), we can see CoVS indeed adding a significant advantage over CoVG. When CoVS is adopted alone, the quality of prioritized groups is not fundamentally better than others due to poor grouping. We repeat this set of experiments on the SC dataset and observe similar results. Therefore, our recommendation is to use both CoVG and CoVS as we did in our Group-FEL.

8 CONCLUSION

In this paper, we first address the issue of group formation in group-based HFL, which is critical due to the widely adopted group operations (for privacy and security) and remains unsolved. We demonstrate through both theoretical and empirical results that the group size and group data distribution are key factors for group formation in group-based HFL and have a significant impact on its convergence and total cost. To address this, we design a greedy grouping algorithm based on group CoV to reduce group size while maintaining the group data relatively IID. Group sampling methods for CoV-aware groups are also proposed and analyzed. Through extensive experiments, we show that current popular FL algorithms do not perform well in HFL, and our methods outperform them in all cases. We believe that our proposed Group-FEL can support more intelligent applications via edge computing and mobile AI. In future work, we plan to explore ways to enhance the proposed grouping formation and sampling strategies by considering γ, related to the CoV of data sample amount among clients, and to maintain client/data fairness in the proposed group-based HFL.

REFERENCES

[1] L. Liu, J. Zhang, S. Song, and K. B. Letaief, "Client-edge-cloud hierarchical federated learning," in *Proc. of IEEE ICC*, 2020.
[2] K. Bonawitz, H. Eichner, *et al.*, "Towards federated learning at scale: System design," *Proc. of MLSys*, vol. 1, pp. 374–388, 2019.
[3] B. McMahan, E. Moore, *et al.*, "Communication-efficient learning of deep networks from decentralized data," in *Proc. of AIStats*, 2017.
[4] K. Bonawitz, V. Ivanov, *et al.*, "Practical secure aggregation for privacy-preserving machine learning," in *Proc. of ACM CCS*, 2017.
[5] Y. Zhao, M. Li, L. Lai, N. Suda, D. Civin, and V. Chandra, "Federated learning with non-IID data," *arXiv preprint arXiv:1806.00582*, 2018.
[6] T. Li, A. K. Sahu, *et al.*, "Federated optimization in heterogeneous networks," *Proc. of MLSys*, vol. 2, pp. 429–450, 2020.
[7] S. P. Karimireddy, S. Kale, *et al.*, "Scaffold: Stochastic controlled averaging for federated learning," in *Proc. of ICML*, 2020.
[8] T. Yu, E. Bagdasaryan, and V. Shmatikov, "Salvaging federated learning by local adaptation," *arXiv preprint arXiv:2002.04758*, 2020.
[9] F. Chen, M. Luo, Z. Dong, Z. Li, and X. He, "Federated meta-learning with fast convergence and efficient communication," *arXiv preprint arXiv:1802.07876*, 2018.
[10] A. Ghosh, J. Hong, D. Yin, and K. Ramchandran, "Robust federated learning in a heterogeneous environment," *arXiv preprint arXiv:1906.06629*, 2019.
[11] M. Luo, F. Chen, *et al.*, "No fear of heterogeneity: Classifier calibration for federated learning with non-IID data," in *Proc. of NeurIPS*, 2021.
[12] R. Presotto, *et al.*, "Fedclar: Federated clustering for personalized sensor-based human activity recognition," in *Proc. of IEEE PerCom*, 2022.
[13] N. Mhaisen, *et al.*, "Optimal user-edge assignment in hierarchical federated learning based on statistical properties and network topology constraints," *IEEE Trans. on Network Science and Engineering*, vol. 9, no. 1, pp. 55–66, 2021.
[14] Y. Deng, *et al.*, "SHARE: Shaping data distribution at edge for communication-efficient hierarchical federated learning," in *Proc. of IEEE ICDCS*, 2021.
[15] J. Wang, Q. Liu, *et al.*, "Tackling the objective inconsistency problem in heterogeneous federated optimization," in *Proc. of NeurIPS*, 2020.
[16] Z. Luo, *et al.*, "Disentangled federated learning for tackling attributes skew via invariant aggregation and diversity transferring," *preprint arXiv:2206.06818*, 2022.
[17] A. Z. Tan, H. Yu, L. Cui, and Q. Yang, "Towards personalized federated learning," *IEEE Trans. on Neural Networks and Learning Systems*, 2022.
[18] A. Fallah, A. Mokhtari, and A. Ozdaglar, "Personalized federated learning: A meta-learning approach," *arXiv preprint arXiv:2002.07948*, 2020.
[19] J. Zhang, S. Guo, *et al.*, "Parameterized knowledge transfer for personalized federated learning," in *Proc. of NeurIPS*, 2021.
[20] Z. Wang, H. Xu, *et al.*, "Resource-efficient federated learning with hierarchical aggregation in edge computing," in *Proc. of IEEE INFOCOM*, 2021.
[21] A. Wainakh, A. S. Guinea, T. Grube, and M. Mühlhäuser, "Enhancing privacy via hierarchical federated learning," in *Proc. of IEEE EuroS&PW*, 2020.
[22] H. Yang, "H-FL: A hierarchical communication-efficient and privacy-protected architecture for federated learning," *arXiv preprint arXiv:2106.00275*, 2021.
[23] X. Wei, J. Liu, X. Shi, and Y. Wang, "Participant selection for hierarchical federated learning in edge clouds," in *Proc. of IEEE NAS*, 2022.
[24] B. Luo, W. Xiao, *et al.*, "Tackling system and statistical heterogeneity for federated learning with adaptive client sampling," in *Proc. of IEEE INFOCOM*, 2022.
[25] Z. Yang, *et al.*, "Energy efficient federated learning over wireless communication networks," *IEEE Trans. on Wireless Commu.*, vol.20, no.3, pp.1935–1949, 2020.
[26] J. Hamer, M. Mohri, and A. T. Suresh, "FedBoost: A communication-efficient algorithm for federated learning," in *Proc. of ICML*, 2020.
[27] H. Gao, A. Xu, and H. Huang, "On the convergence of communication-efficient local sgd for federated learning," in *Proc. of AAAI*, 2021, pp. 7510–7518.
[28] T. D. Nguyen, P. Rieger, *et al.*, "Flame: Taming backdoors in federated learning," *Cryptology ePrint Archive*, 2021.
[29] H. Yang, M. Fang, and J. Liu, "Achieving linear speedup with partial worker participation in non-IID federated learning," *Proc. of ICLR*, 2021.
[30] L. Wang, Y. Guo, T. Lin, and X. Tang, "Client selection in nonconvex federated learning: Improved convergence analysis for optimal unbiased sampling strategy," *arXiv preprint arXiv:2205.13925*, 2022.
[31] X. Li, K. Huang, W. Yang, S. Wang, and Z. Zhang, "On the convergence of FedAvg on non-IID data," in *Proc. of ICLR*, 2019.
[32] M. Mahajan, P. Nimbhorkar, and K. Varadarajan, "The planar k-means problem is NP-hard.," *Theoretical Computer Science*, 442, 13-21, 2012.
[33] D. Aloise, A. Deshpande, *et al.*, "NP-hardness of Euclidean sum-of-squares clustering," *Machine learning*, 75, 245-248, 2009.
[34] P. Warden, "Speech Commands: A Dataset for Limited-Vocabulary Speech Recognition," *arXiv preprint arXiv:1804.03209*, 2018.
[35] A. Krizhevsky, G. Hinton, *et al.*, "Learning multiple layers of features from tiny images," Technical Report, University of Toronto, 2009.
[36] T.-M. H. Hsu, *et al.*, "Measuring the effects of non-identical data distribution for federated visual classification," *arXiv preprint arXiv:1909.06335*, 2019.