FedGroup: Accurate Federated Learning via Decomposed Similarity-Based Clustering

Moming Duan*, Duo Liu*, Xinyuan Ji†, Renping Liu*, Liang Liang*, Xianzhang Chen*, Yujuan Tan*
*College of Computer Science, Chongqing University, Chongqing, China
†School of Computer Science and Technology, Xi'an Jiaotong University, Xi'an, China

arXiv:2010.06870v3 [cs.LG] 8 Jan 2021

***Abstract*—Federated Learning (FL) enables the multiple participating devices to collaboratively contribute to a global neural network model while keeping the training data locally. Unlike the centralized training setting, the non-IID and imbalanced (statistical heterogeneity) training data of FL is distributed in the federated network, which will increase the divergences between the local models and global model, further degrading performance. In this paper, we propose a novel clustered federated learning (CFL) framework FedGroup based on a similarity-based client clustering strategy, in which we 1) group the training of clients based on the similarities between the clients' optimize directions for high training performance; 2) reduce the complexity of client clustering algorithm by decomposing the high-dimension low-sample size (HDLSS) direction vectors. 3) implement a newcomer device cold start mechanism based on the auxiliary global model for framework scalability and practicality.**

FedGroup can achieve improvements by dividing joint optimization into groups of sub-optimization, and can be combined with *FedProx*, the state-of-the-art federated optimization algorithm. We evaluate FedGroup and FedGrouProx (combined with *FedProx*) on several open datasets. The experimental results show that our proposed frameworks significantly improving absolute test accuracy by $+14.7\%$ on FEMNIST compared to *FedAvg*, $+5.4\%$ on Sentiment140 compared to *FedProx*.

I. Introduction

Federated Learning (FL) [1]–[5] is a promising distributed neural network training approach, which enables multiple end-users to collaboratively train a shared neural network model while keeping the training data decentralized. In practice, a FL server first distributes the global model to a random subset of participating clients (e.g. mobile and IoT devices). Then each client optimizes its local model by gradient descent based on its local data in parallel. Finally, the FL server averages all local models' updates or parameters and aggregates them into a new global model. Unlike the traditional cloud-centric learning paradigm and the distributed machine learning frameworks based on Parameter Server [6], there is no need to transfer private data over the communication network during the FL training procedure. With the advantage of privacy-preserving, Federated Learning is currently the most attractive distributed machine learning framework for collaborative training based on large-scale mobile devices or IoT devices.

Nevertheless, due to the FL server does not have the authority to access the user data or collect statistical information, some data preparation operations such as balancing and outlier detection will be restricted. Therefore, the high statistical heterogeneity is a key challenge of federated learning [5].

To tackle heterogeneity in federated learning, several efforts have been made. McMahan *et al.* propose the vanilla FL framework *Federated Averaging (FedAvg)* [2] and experimentally demonstrate that *FedAvg* is communication-efficient and can converge under statistical heterogeneity setting (non-IID). However, Zhao *et al.* [7] show that the accuracy of FL reduces $\sim 55\%$ for CNN trained for highly skewed non-IID CIFAR-10 [8] dataset. The experiments based on VGG11 [9] by Sattler *et al.* [10] show that non-IID data not only leads to accuracy degradation, but also reduces the convergence speed. Li *et al.* [11] theoretically analyze the convergence of *FedAvg* on non-IID data and indicates that heterogeneity of data slows down the convergence for the strongly convex and smooth problem. In addition, Duan *et al.* [12] demonstrate that global imbalanced data has a negative impact on mobile federated training applications. Unfortunately, the retrieval of model accuracy decreases as the local model diverges in [7], [10], [12]. Recently, Sattler *et al.* [13], [14] propose a novel Federated Multi-Task Learning framework Cluster Federated Learning (CFL), which exploits geometric properties of FL loss surface to cluster the learning processes of clients based on their optimization direction, provides a new way of thinking about the statistical heterogeneity challenge. Many researchers follow up on CFL-based framework [15]–[17] and confirm CFL is more accurate than traditional FL with the consensus global model. However, above CFL-based frameworks are inefficient in the large-scale federated training systems or ignore the presence of newcomer devices. (see Section II for additional details)

In this paper, we propose an efficient and accurate clustered federated learning framework FedGroup, which clusters clients into multiple groups based on the decomposed cosine similarities between their parameter updates. In each communication round, each active client only contributes its local optimization result to the group model it belongs to. The FL server still maintains an auxiliary global average model to address the cold start issues of newcomer devices. To reduce the clustering complexity of high-dimension low-sample size (HDLSS) parameter updates, we decompose the directions of clients' updates in three componential direction vectors to derive the ternary cosine similarities, then group these clients based on the distances of their similarities. Furthermore, we combine FedGroup with the state-of-the-art federated optimizer *FedProx* [18] and we name it FedGrouProx, which shows its advancement in the experiment.

With the above methods, FedGroup can significantly improve test accuracy by $+5.2\%$ on MNIST [19], $+14.7\%$ on FEMNIST [20], $+5.4\%$ on Sentiment140 [21] compared to the best of FedAvg and FedProx.

The main contributions of this paper are summarized as follows.

- We propose two novel similarity-based clustered federated learning frameworks, FedGroup and FedGrouProx, and show its superiority on four open datasets (with statistical heterogeneity) compared to FedAvg and FedProx.
- We propose a decomposed cosine similarity-based client clustering algorithm to improve the effectiveness of CFL framework, which alleviates the high computational complexity caused by directly clustering HDLSS direction vectors.
- Our framework presents an efficient cold start strategy for the groups and the newcomers, which provide a new approach to improve the scalability and practicality of the traditional CFL frameworks. We demonstrate its efficiency experimentally. In addition, we open source the code of FedGroup to contribute to the community.

II. BACKGROUND

A. Federated Learning

In this section, we introduce the most widely adopted federated learning algorithm FedAvg and briefly explain the SOTA federated optimization method FedProx.

McMahan *et al.* first introduce federated learning [2] and the vanilla FL optimization method FedAvg, which is designed to provide privacy-preserving support for distributed machine learning model training. Different from the previous distributed machine learning framework, the FL is a more challenging task because the server has limited or no control over the clients. For example, the server cannot access the users' data and each client can freely participate in the training network or disconnect from the network. The distributed objective of FL is:

$$\min_{\boldsymbol{w}} \Big\{ f(\boldsymbol{w}) \triangleq \sum_{k=1}^{N} p_k F_k(\boldsymbol{w}) \Big\}, \tag{1}$$

Where N is the number of clients, p_k is the weight of the k-th device, $p_k \geqslant 0$, $\sum_k p_k = 1$. In statistical heterogeneity setting, the local objectives $F_k(\boldsymbol{w})$ measure the local expirical risk over possibly differing data distributions $p_{data}^{(k)}$.

For a machine learning problem, we can set $F_k(\boldsymbol{w})$ to the user-specified loss function $L(\cdot;\cdot)$ of the predictions on examples $(\boldsymbol{x}, \boldsymbol{y})$ made with model parameters w. Hence, the local objective is defined by

$$F_k(\boldsymbol{w}) \triangleq \mathbb{E}_{(\boldsymbol{x},\boldsymbol{y})\sim p_{data}^{(k)}} L(\boldsymbol{x}, \boldsymbol{y}; \boldsymbol{w}). \tag{2}$$

The global objective $f(\boldsymbol{w})$ can be regarded as a joint objective function for multiple clients, and FL tries to minimize it by optimizing all local optimization objectives.

In practice, a typical implementation of FedAvg-based large-scale federated training system [3] includes a cloud server that maintains a global model and multiple participating devices, which communicate with the server through network. At each communication round t, server selects a random subset K of the active devices (i.e. clients) to participate in this round of training. The server then broadcasts the latest global model w^t to these clients for further local optimization. Then each participating client optimizes the local objective function based on the device's data by its local solvers (e.g. SGD) with several local epochs E in parallel. Finally, the FL server collects and aggregates the locally-computed parameter updates $\Delta \boldsymbol{w}_i^t$ from these clients that completed the local training within the time budget. The FL server will update the global to w^{t+1} and finish the current communication round. In general, a federated learning training task requires hundreds of communication rounds to reach target accuracy.

Algorithm 1 Federated Averaging

```
procedure FL SERVER TRAINING
    Initialize global model w_0, w_1 ← w_0.
    for each communication round t = 1, 2, ..., T do
        S_t ← Server selects random subset of K clients.
        Server broadcasts w_t to all selected clients.
        for each activate client i ∈ S_t parallelly do
            Δw^i_{t+1} ← ClientUpdate(i, w_t).
        w_{t+1} ← w_t + Σ_{i∈S_t} (n_i/n) Δw^i_{t+1}

function CLIENTUPDATE(i, w)
    ŵ ← w
    for each local epoch e = 1, 2, ..., E do
        for each local batch b ∈ B do
            w ← w − η∇L(b; w)
        Δw ← w − ŵ
    return Δw
```

The details of $FedAvg$ are shown in Algorithm 1. Here n_i denotes the training data size of client i, the total training data size $n = \sum n_i$. The $\mathcal{B}$ is the batches of training data, η is the learning rate of the local solver. There have two key hyperparameters in $FedAvg$, the first is the number of participating clients K in each round, or the participation rate $\frac{K}{N}$. For IID setting, a higher participation rate can improve the convergence rate, but for non-IID setting, a small participation rate is recommended to alleviate the straggler's effect [11]. The second is the local epoch E, an appropriately large E can increase the convergence speed of the global optimization and reduce the communication requirement. However, an excessively large E will increase the discrepancy between the local optimization solutions and the global model, which will lead to the federated training procedure be volatile and yields suboptimal results.

As mentioned above, the key to increasing the convergence speed and robustness of the federated training system is to bound the discrepancies between the local optimization objectives and the global objective. To alleviate the divergences, Li *et al.* [18] introduce a penalty (aka proximal term) to the local objective function. The modified local objective function is:

$$F_k(\boldsymbol{w}) + \frac{\mu}{2}\|\boldsymbol{w} - \boldsymbol{w}^t\|^2. \tag{3}$$

Where the hyperparameter μ controls how far the updated model can be from the starting model. Obviously that FedAvg is a special case of FedProx with $\mu = 0$.

B. Clustered Federated Learning

One of the main challenges in the design of large-scale FL applications is statistical heterogeneity (e.g. non-IID, size imbalanced, class imbalanced) [3], [5], [10], [17], [22]. The conventional FL try to train a consensus global model upon multiple incongruent data distributions, which yields unsatisfactory model performance in highly heterogeneity setting [7].

Instead of optimizing a consensus global model, CFL divides the optimization goal to several sub-objective and follows a *Pluralistic Group* architecture. CFL maintains multiple groups (or clusters) models, which are more specialized than the consensus model and achieve high accuracy. (We recommend readers refer to the taxonomy proposed in [27])

Sattler *et al.* propose the first CFL-based framework [13], which recursively separate the two groups of clients with incongruent descent directions. The authors further propose [14] to improve the robustness of CFL-based framework in byzantine setting. However, the recursive bi-partitioning algorithm is computational inefficiency and requires multiple communication rounds to completely separate all incongruent clients. Furthermore, since the participating client in each round is random, this may cause the recursive bi-partitioning algorithm to fail. To improve the efficient of CFL, Ghosh *et al.* propose IFCA [16], which randomly generates cluster centers and divides clients into clusters that will minimize their loss values. Although the clustering algorithm of ICFA is efficient, its communication is inefficient because the center server needs to broadcast all groups' parameters at each round. FeSEM [15] is similar to ICFA, the main difference is the clustering metrics of FeSEM is based on the L2 distance between clients' weights. Similarity to the cosine similarity-based clustering method [13], Briggs *et al.* propose an agglomerative hierarchical clustering method named FL+HC [17]. However, FL+HC relies on iterative calculating the pairwise distance between all clusters, which is computationally complex. Note that, all the above CFL frameworks assume that all clients will participate the clustering process and no newcomer devices, which is unpractical in a large-scale FL system.

III. FedGroup and FedGrouProx

A. Motivation

Before introducing our proposed FedGroup and FedGrouProx, we show a toy example to illustrate the motivation of our work. To study the impacts of statistical heterogeneity, model accuracy, and discrepancy, we implement a convex multinomial logistic regression task train with FedAvg method on MNIST [19] dataset following the instructions of [18]. We manipulate the statistical heterogeneity of training set by forcing each client to have only a limited number of classes of data. All distributed client data is randomly sub-sampled from the original MNIST dataset without replacement, and the number of samples per client follows a power law.

TABLE I
Quantitative results of FedAvg training based on non-IID MNIST with different number of classes of data per client.

# Classes	Discrepancy		Accuracy		# Round to Reach
	Mean	Variance	Max	Median	Target Acc-85%
1	15.33	46.42	87.2%	79.35%	36
2	15.96	43.63	89.3%	84.75%	29
3	15.70	31.73	90.4%	86.30%	14
5	13.98	9.02	91.2%	89.90%	10

We build four non-IID MNIST datasets with a different number of classes limitation. The total number of clients $N = 1000$, the number of clients selected in each round $K = 20$, the local mini-batch size is 10, the local epoch $E = 20$, the learning rate $\eta = 0.03$. We run $T = 200$ communication rounds of $FedAvg$ training and evaluated the global model on test set per round. Note that although we use the same test data in each round, we did not readjust the model structure based on the testing results, so we ignore the possibility of test data leakage (e.g. overfitting to the test data) [23].

The experimental results are shown in Fig. 1. Where the left y-axis is the testing top-1 classification accuracy, and the right y-axis is the sum of norm difference between the client models weights and the latest global model weights. Specifically, the discrepancy in communication round t is defined as:

$$Discrepancy(t) \triangleq \sum_{i \in S_t} \|\boldsymbol{w}_i - \boldsymbol{w}_t\|. \tag{4}$$

As shown by the blue lines in Fig. 1, the discrepancy is relaxed as the classes number limitation of each client's data increases, which means the high heterogeneity will make the federated training models prone to diverge. Moreover, high data heterogeneity also hurt the convergence rate of training, and the model accuracy, which is shown by the black lines in Fig 1, becomes more fluctuant with the increase of data heterogeneity.

The quantitative results which are shown in TABLE I support our observations. As the data heterogeneity increases, the variance of the discrepancy significantly decreases by 80.5% (from 46.42 to 9.02), and the max of absolute testing accuracy is increased +4.0% (from 87.2% to 91.2%). The number of the required communication rounds to reach 85% testing accuracy is also significantly reduced in the low heterogeneous setting, which means faster convergence and less communication consumption.

B. Framework Overview

To tackle the statistical heterogeneity challenge in FL, we propose two novel FL frameworks, FedGroup and FedGrouProx, which leverage a similarity-based clustering

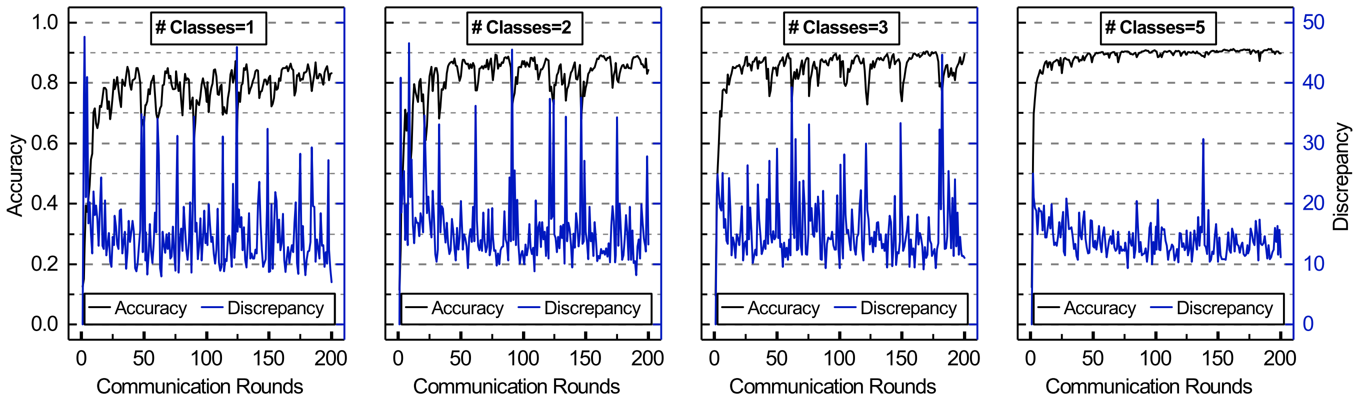


Fig. 1. A FedAvg training procedure on non-IID MNIST to illustrate the effects of statistical heterogeneity on model accuracy and discrepancy. From left to right, the number of classes of data per client increase, which means the degree of data heterogeneity decreases. The discrepancy is defined in Equation 4.

strategy to group the training of clients based on the similarities between the client's local optimization solutions. Our propose frameworks are inspired by CFL [13], which clusters clients by a recursive bi-partitioning algorithm. CFL assumes the clients have incongruent risk functions (e.g. randomly swapping out the labels of training data), and the clustering propose of CFL is to separate clients with incongruent risk functions. Our proposed frameworks borrow the client clustering idea of CFL. However, unlike CFL, our grouping strategy is static, which avoids re-clustering clients every round. More significantly, we define the distance used in clustering as the ternary cosine similarities, which makes our algorithm more efficient when the number of clients is large.

Before we go into more detail, we first show the general overview of the FedGroup framework. The overview of the federated training procedure of FedGroup is shown in Fig. 2. Note that Fig. 2 is a schematic diagram of FedGroup training process, it does not represent the physical deployment of connection and device. In practice, the clients can be connected mobiles or IoT devices, the global model is maintained by a cloud server (FL server), and the group can be deployed on the FL server or the mobile edge computing (MEC) server. To ease the discussion, we assume that all groups are deployed on the same cloud server, and all clients are connected mobile devices.

FedGroup contains one global model, a certain amount of groups, and multiple clients, each of them maintains one latest model and the latest update for this model. Each client and group have a one-to-one correspondence, but there are also possible to have a group without clients in a communication round or a client is not in any groups (e.g. a newcomer joins the training network). The newcomer device uses a pre-training based cold start algorithm to select the group it should join (we call the client that is not in any group is a cold client), we will explain the details of the cold start algorithm later. For the federated data, each client i has a training set and a test set split from the distribution $p_{data}^{(i)}$ of user data.

As shown in the Fig. 2, there are three model aggregation

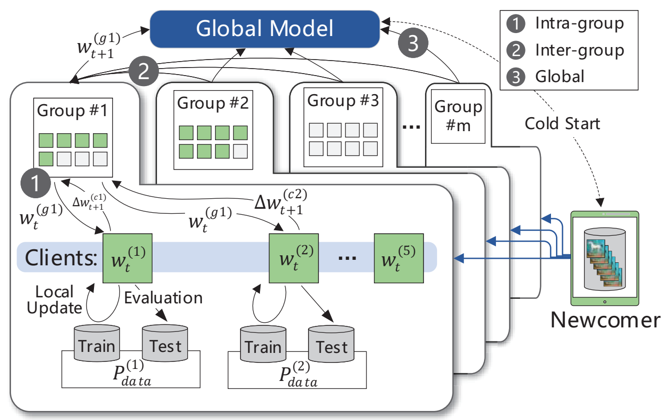


Fig. 2. An overview of FedGroup.

processes in our framework, including intra-group aggregation (①), inter-group aggregation (②), and global aggregation (③). In each communication round, all the latest models and updates will be refreshed by their local optimizers or the three model aggregation processes mentioned above.

In our design, each group federally trains on a certain set of clients based on their training data to optimize its group model, and evaluate the group model on the same set of clients based on their test data. Specifically, each group broadcasts its model parameters to the clients in this group and then aggregates the updates from these clients using *FedAvg* in parallel, we name this aggregation procedure as the intra-group aggregation. After all federated training complete, the group models are aggregate according to a certain weight and refreshed, which we named inter-group aggregation. Then, we refresh the global model by simply averaging all group models. Note that, unlike the previous FL framework, the global model in FedGroup is not broadcast to clients or groups, we only maintain this model for the cold start of newcomer.

Training: The details of the training procedure of FedGroup and FedGrouProx are shown in Algorithm 2. The FedGroup training has two key hyperparameters, m controls the total number of groups and η_g is the learning rate of inter-group aggregation. Note that, FedGroup can be regarded as a special case of FedGrouProx with $\mu = 0$. Before training, the FL server first initializes the global model and group models to the same initial weights $\boldsymbol{w}_0$. At the beginning of each round, a random subset of clients is selected to participate in this round of training (line 4). Each group trains the model in the federated learning approach and to get a temporary group model $\tilde{\boldsymbol{w}}_{t+1}^{(g_j)}$ (line 7), the group parameters are refreshed after the inter-group aggregation (line 10). At the end of each round, the FL server refreshes the global model by averaging all group models (line 11). Some details not shown in Algorithm 2 are that each object in FedGroup calculates and stores its latest model update before refreshing the local model, and the groups and clients need to tackle cold start issues before training. We will discuss our strategies for cold start issues in the next three sections.

Algorithm 2 FedGroup and FedGrouProx

Input: Set of clients $\mathcal{C} \leftarrow \{c_1, c_2, ..., c_n\}$,
set of groups $\mathcal{G} \leftarrow \{g_1, g_2, ..., g_m\}$,
set of group parameters $\mathcal{W}_0^{(G)} \leftarrow \{\boldsymbol{w}_0^{(g_1)}, \boldsymbol{w}_0^{(g_2)}, ..., \boldsymbol{w}_0^{(g_m)}\}$,
$g_j.clients \leftarrow \{c_i | c_i$ is in group $g_j, \forall c_i \in \mathcal{C}\}$,
initial model parameters $\boldsymbol{w}_0$, number of epochs T, number of selected clients per round K, inter-group learning rate η_g, proximal hyperparmeter μ.
Output: updated group model parameters $\mathcal{W}_T^{(G)}$.

1: **procedure** FedGrouProx Training
2: Init: $\boldsymbol{w}_1^{(g_j)} \leftarrow \boldsymbol{w}_0$, for all $j \in [1, m]$, initialize the group parameters $\mathcal{W}_0^{(G)} \leftarrow \{\boldsymbol{w}_1^{(g_1)}, \boldsymbol{w}_1^{(g_2)}, ..., \boldsymbol{w}_1^{(g_m)}\}$.
3: **for** each communication round $t = 1, 2, ..., T$ **do**
4: $S_t \leftarrow$ Server selects random subset of K clients.
5: **for** each group g_j in $\mathcal{G}$ parallelly **do**
6: $S_t^{(g_j)} \leftarrow \{c_i | c_i \in g_j.clients, \forall c_i \in S_t\}$.
7: $\tilde{\boldsymbol{w}}_{t+1}^{(g_j)} \leftarrow$ **IntraGroupUpdate(**$S_t^{(g_j)}$**,** $\boldsymbol{w}_t^{(g_j)}$**)**.
8: $\tilde{\mathcal{W}}_{t+1}^{(G)} \leftarrow \{\tilde{\boldsymbol{w}}_{t+1}^{(g_1)}, \tilde{\boldsymbol{w}}_{t+1}^{(g_2)}, ..., \tilde{\boldsymbol{w}}_{t+1}^{(g_m)}\}$.
9: **for** each group g_j in $\mathcal{G}$ parallelly **do**
10: $\mathcal{W}_{t+1}^{(G)} \leftarrow$ **InterGroupAggregation(**$\tilde{\mathcal{W}}_{t+1}^{(G)}$**,** η_g**)**.
11: $\boldsymbol{w}_{t+1} \leftarrow$ **GlobalAggregation(**$\mathcal{W}_{t+1}^{(G)}$**)**. //Average of group parameters without weights.

12: **function** IntraGroupUpdate($S_t, \boldsymbol{w}_t$)
13: **if** S_t is $\emptyset$ **then**
14: **return** $\boldsymbol{w}_t$
15: For **FedGroup**:
16: $\boldsymbol{w}_{t+1} \leftarrow FedAvg(S_t, \boldsymbol{w}_t)$. //Ref. Algorithm 1.
17: For **FedGrouProx**:
18: $\boldsymbol{w}_{t+1} \leftarrow FedProx(S_t, \boldsymbol{w}_t, \mu)$. //Ref. [18].
19: **return** $\boldsymbol{w}_{t+1}$

20: **function** InterGroupAggregation($\tilde{\mathcal{W}}_{t+1}, \eta_g$)
21: **for** each group parameter $\tilde{\boldsymbol{w}}_{t+1}^{(g_j)}$ in $\tilde{\mathcal{W}}_{t+1}$ parallelly **do**
22: $\Delta\tilde{\boldsymbol{w}}_{t+1}^{(g_j)} \leftarrow \eta_g \sum_{l \neq j} \frac{\tilde{\boldsymbol{w}}_{t+1}^{(g_l)}}{\|\tilde{\boldsymbol{w}}_{t+1}^{(g_l)}\|}$
23: $\boldsymbol{w}_{t+1}^{(g_j)} \leftarrow \tilde{\boldsymbol{w}}_{t+1}^{(g_j)} + \Delta\tilde{\boldsymbol{w}}_{t+1}^{(g_j)}$
24: $\mathcal{W}_{t+1} \leftarrow \{\boldsymbol{w}_{t+1}^{(g_1)}, \boldsymbol{w}_{t+1}^{(g_2)}, ..., \boldsymbol{w}_{t+1}^{(g_m)}\}$
25: **return** $\mathcal{W}_{t+1}$

C. Group Cold Start: Ternary Cosine Similarity

In the general setting of federated learning, there is only one global optimization goal, which is to minimize the joint loss function of all learning tasks of all clients. However, optimizing a global objective becomes difficult when the distributed source data for the joint distribution is non-IID or called statistical heterogeneity. Li *et al.* [11] theoretically prove that heterogeneity of training data slows down the convergence of FL. Instead of optimizing a complex global goal, why not divide it into several sub-optimization goals? Based on this idea, we group the training of clients based on the similarities between their local optimization directions. But there are two following questions: (1) how to measure the similarity, and (2) how to determine the optimization goals of each group before training.

We will discuss the second question in the next section. For the first question, a heuristic way is to use the cosines between the gradients calculated by backpropagation or the updates of model parameters to define the similarity. The cosine similarity $Cosim(i,j)$ between the parameters updates of any two clients c_i and c_j is defined by:

$$Cosim(i,j) \triangleq \frac{< \Delta\boldsymbol{w}_t^{(c_i)}, \Delta\boldsymbol{w}_t^{(c_j)} >}{\|\Delta\boldsymbol{w}_t^{(c_i)}\| \; \|\Delta\boldsymbol{w}_t^{(c_j)}\|}. \tag{5}$$

And the similarity matrix $\mathcal{M}' \subset \mathbb{R}^{n \times n}$ (all similarities between any two clients) is given by:

$$\mathcal{M}'_{ij} = Cosim(i,j), i,j \in n. \tag{6}$$

The computational complexity of calculating the similarity matrix $\mathcal{M}'$ is $O\big(n^2 d_{\boldsymbol{w}}^2\big)$, n and $d_{\boldsymbol{w}}$ are the total number of clients and the total number fo parameters, respectively. We assume the all parameters updates $\Delta\boldsymbol{w}_t$ are flattened row vectors, so $\Delta\boldsymbol{w}_t \subset \mathbb{R}^{1 \times d_w}$. Therefore, the shortcomings of the above definition are that the computational overhead of calculating the similarity matrix increases with the square of the training scale and the model size.

To reduce the computational cost, we propose a new metric named *ternary cosine similarity* to measure the local optimization direction of the update. The ternary cosine similarity $\varsigma'(i, \boldsymbol{V})$ of client i is defined by:

$$\begin{aligned} \varsigma'(i, \boldsymbol{V}) \;&\triangleq (\Delta\boldsymbol{w}_t^{(c_i)} \cdot \boldsymbol{V}) \circ \boldsymbol{v}_{scale}, i \in n \\ &\boldsymbol{V} = [\boldsymbol{v}_1 \;\, \boldsymbol{v}_2 \;\, \boldsymbol{v}_3], \;\boldsymbol{V} \subset \mathbb{R}^{d_{\boldsymbol{w}} \times 3}, \\ &\boldsymbol{v}_{scale} = (\|\Delta\boldsymbol{w}_t^{(c_i)}\| \; [\|\boldsymbol{v}_1\|, \|\boldsymbol{v}_2\|, \|\boldsymbol{v}_3\|]^T)^{\circ -1}. \end{aligned} \tag{7}$$

We normalize $\varsigma'(i, \boldsymbol{V})$ to $[0, 1]$ by:

$$\varsigma(i, \boldsymbol{V}) = \frac{-\varsigma'(i, \boldsymbol{V}) + 1}{2}, \tag{8}$$

Algorithm 3 Ternary Cosine Similarity Matrix Calculation

Input: Set of clients $\mathcal{C}$, number of group m, global initial model $\boldsymbol{w}_0$, pre-training scale hyperparameter α.
Output: Ternary cosine similarity matrix $\mathcal{M}$.

1: **procedure** CALCULATE $\mathcal{M}$
2: $S \leftarrow$ Server selects random subset of αm clients.
3: Server broadcasts $\boldsymbol{w}_0$ to all selected clients.
4: **for** each client c_i in S parallelly **do**
5: $\Delta \boldsymbol{w}_0^{(c_i)} \leftarrow$ **PreTrainClient**($c_i, \boldsymbol{w}_0$).
6: $\Delta \boldsymbol{W} \leftarrow [\Delta \boldsymbol{w}_0^{(c_1)}, \Delta \boldsymbol{w}_0^{(c_2)}, \ldots, \Delta \boldsymbol{w}_0^{(c_{\alpha m})}]$.
7: $\boldsymbol{V} = [\boldsymbol{v}_1, \boldsymbol{v}_2, \boldsymbol{v}_3] \leftarrow SVD(\Delta \boldsymbol{W}^T, 3)$ // Ref. [24]
8: $\mathcal{M}_i \leftarrow \varsigma(i, \boldsymbol{V}), \forall i \in n$ // Ref. Eq. (7) (8)
9: $\mathcal{M} \leftarrow [\mathcal{M}_1, \mathcal{M}_2, \ldots, \mathcal{M}_n]$
10: **return** $\mathcal{M}$

11: **function** PRETRAINCLIENT($c, \boldsymbol{w}$)
12: $\hat{\boldsymbol{w}} \leftarrow \boldsymbol{w}$
13: **for** each local batch $b \in \mathcal{B}$ **do**
14: $\boldsymbol{w} \leftarrow \boldsymbol{w} - \eta \nabla L(b; \boldsymbol{w})$
15: $\Delta \boldsymbol{w} \leftarrow \boldsymbol{w} - \hat{\boldsymbol{w}}$
16: **return** $flatten(\Delta \boldsymbol{w})$

and the ternary cosine similarity matrix $\mathcal{M} \subset \mathbb{R}^{n \times 3}$ is given by:

$$\mathcal{M}_i = \varsigma(i, \boldsymbol{V}), i \in n. \tag{9}$$

The $\circ$ operator is the Hadamard (element-wise) product between two matrices and the $^{\circ -1}$ is the Hadamard inverse. The $\boldsymbol{V}$ is a matrix of three column vector $\boldsymbol{v}_1, \boldsymbol{v}_2, \boldsymbol{v}_3$, we call these vectors the base direction vector, which are calculate by using Singular Value Decomposition (SVD) algorithm [24] to decompose the transposed local updates matrix $\Delta \boldsymbol{W}^T$. We show the implementation details of the ternary cosine similarity in Algorithm 3.

Ternary cosine similarity: The main idea of our Algorithm 3 is to first pre-train the clients for one epoch base on the global initial model $\boldsymbol{w}_0$ to get the local updates matrix $\Delta \boldsymbol{W}$ (line 5). And then we decompose $\Delta \boldsymbol{W}^T$ in three main componential local update vectors (not componential model parameters) using SVD (line 7), which we call base direction vectors $\boldsymbol{v}_1, \boldsymbol{v}_2, \boldsymbol{v}_3$. Then we can measure the local optimization direction of client c_i by the normalized ternary cosine similarity ς for its pre-training update $\Delta \boldsymbol{w}_0^{(c_i)}$ and base direction $\boldsymbol{V}$ (line 8). Another improvement of our algorithm is that we only select a subset of clients to participate in the pre-training and decomposition process. We control the scale of pre-training by hyperparameter α and the number of pre-training clients is set to αm. Our implementation is more practical because it is difficult to satisfy that all clients are active until they complete the training in the large scale FL system. The drop-out may occur due to network jitter. We will show the efficiency of our ternary cosine similarity metrics in Section V.

D. Group Cold Start: A Clustering Approach

To determine the optimization goals of each group, we propose a clustering approach, which clusters the parameter updates directions (that is ς in FedGroup) of clients into m groups using K-Means++ [25] algorithm. The main advantage of the clustering approach is that it is unsupervised, and we can divide the global optimization function into m sub-optimization functions regardless of whether there have incongruent optimization goals. However, in the general setting of FL, $n \ll d_{\boldsymbol{w}}$, it means that the parameter updates data $\Delta \boldsymbol{W}$ is high-dimension low-sample size (HDLSS), which has adverse effects on the performance of pairwise distance-based clustering algorithms such as K-Means, k-Medoids and hierarchical clustering [26]. Moreover, training all clients to cluster their updates directions is not communication friendly and practically achievable. Hence, our clustering procedure considers the ternary similarity matrix $\mathcal{M}$ in algorithm 3 as the distance for clustering. In other words, FedGroup performs a low-dimensional embedding of the local updates matrix $\Delta \boldsymbol{W}$, following by K-Means++ clustering. In addition, our calculation of client clustering is following the calculation of the similarity matrix, which means the above calculations only requires one round of communication. We call the combination of the above two processes as group cold start, and the details are shown in Algorithm 4. The $g_j.cm$ is the centric mean of the ternary similarity of all clients in the group j. After the server performs the group cold start, the optimization direction of each group, which is measured by $g_j.cm$, is determined and αm clients participating in this process are assigned. We leverage the centric means of groups to measure the clustering validity index like within-cluster sum-of-squares criterion. But once the group cold starting is completed, we will use the latest group parameters updates $\Delta \mathcal{W}^{(G)}$ to represent the optimization direction of the group to cluster new clients, which will be explained later.

Algorithm 4 Group Cold Start

Input: (Same as Algorithm 3).
Output: set of groups $\mathcal{G}$, set of group parameters $\mathcal{W}_0^{(G)}$, set of group parameters updates $\Delta \mathcal{W}_0^{(G)}$.

procedure GROUP COLD START
$\mathcal{M}, \Delta \boldsymbol{W} \leftarrow$ Calculate $\mathcal{M}$. // Ref. Algorithm 3
$[g_1.clients, \ldots, g_m.clients] \leftarrow$ K-Means++($\mathcal{M}$).
$\mathcal{G} \leftarrow [g_1, g_2, \ldots, g_m]$.
for g_j in $\mathcal{G}$ **do**
$g_j.cm \leftarrow Average(\mathcal{M}[i], \forall c_i \in g_j.clients)$.
$\boldsymbol{w}_0^{(g_j)} \leftarrow Average([\Delta \boldsymbol{w}_0^{(c_i)}, \forall c_i \in g_j.clients])$.
$\Delta \boldsymbol{w}_0^{(g_j)} \leftarrow \boldsymbol{w}_0^{(g_j)} - \boldsymbol{w}_0$.
$\mathcal{W}_0^{(G)} \leftarrow [\boldsymbol{w}_0^{(g_1)}, \boldsymbol{w}_0^{(g_2)}, \ldots, \boldsymbol{w}_0^{(g_m)}]$.
$\Delta \mathcal{W}_0^{(G)} \leftarrow [\Delta \boldsymbol{w}_0^{(g_1)}, \Delta \boldsymbol{w}_0^{(g_2)}, \ldots, \Delta \boldsymbol{w}_0^{(g_m)}]$.
return $\mathcal{G}, \mathcal{W}_0^{(G)}, \Delta \mathcal{W}_0^{(G)}$

E. Client Cold Start

As described before, the group cold start algorithm selects a random subset of the clients for pre-training, so the remaining clients are cold clients and are not in any groups. Since the federated training network is dynamic, the new device can join the training at any time, so we need to classify newcomers according to the similarity between their optimization goals and groups'. Our client cold start strategy is to assign clients to the groups that are most closely related to their optimization direction, as shown below:

$$\begin{aligned} g^* &= \underset{j}{\operatorname{argmin}} \frac{-\cos(\sphericalangle(\Delta \boldsymbol{w}_t^{(g_j)}, \Delta \boldsymbol{w}_{pre}^{(i)})) + 1}{2}, \\ g^*.clients &= g^*.clients \cup \{c_i\}. \end{aligned} \tag{10}$$

Suppose the newcomer joins the training network in round t, then the $\Delta \boldsymbol{w}_{pre}^{(i)}$ is the pre-training update base on the global model $\boldsymbol{w}_t$. We assign the newcomer i to group g^* to minimize the normalized cosine similarity reference to Eq. 8.

In summary, the key features of our FedGroup framework are as follows:

- FedGroup reduces the discrepancy between the joint optimization objective and sub-optimization objectives, which will help improve the convergence speed and performance of the model.
- The proposed framework determines the optimization objectives of groups by a clustering approach, which is unsupervised and can disengage from the incongruent risk functions assumption.
- FedGroup adopts a novel metrics named ternary cosine similarity, a low-dimension embedding of clients' local updates, for efficient HDLSS direction vectors clustering.
- The clustering mechanism of FedGroup considers the joining of newcomer devices.

Compared with vanilla FL, FedGroup requires additional communication to transmit pre-training parameters updates. However, we emphasize that the pre-training procedure does not occupy a whole communication round, the client continues to train $E-1$ epochs and uploads the parameters updates for the intra-group aggregation.

IV. Convergence Analysis

We analyze convergence for FedGroup in this section. First, we analyze the convergence of our proposed framework without inter-group aggregation (e.g. $\eta_g = 0$ in Algorithm 2 line 10), and then extent it to the case with inter-group aggregation.

In FedGroup, the group membership for each group is static, so we can assume that any client k is allocated to group g. We make the following assumptions on the local loss function $F_{k,g}(\cdot)$ for any client k.

Assumption 1. *For any client k, $F_{k,g}(\cdot)$ is convex.*

Assumption 2. *$F_{k,g}$ is M-Lipschitz continuous: for all $\boldsymbol{w}$ and $\boldsymbol{v}$, $\|F_{k,g}(\boldsymbol{w}) - F_{k,g}(\boldsymbol{v})\| \le M\|\boldsymbol{w} - \boldsymbol{v}\|$.*

Assumption 3. *$F_{k,g}$ is L-Lipschitz smooth: for all $\boldsymbol{w}$ and $\boldsymbol{v}$, $\|\nabla F_{k,g}(\boldsymbol{w}) - \nabla F_{k,g}(\boldsymbol{v})\| \le L\|\boldsymbol{w} - \boldsymbol{v}\|$.*

Above assumptions have been make by many relevant works [27]–[29].

Definition 1 (Group Loss Function)**.** *For any group $g \in \mathcal{G}$, the group loss function is $F_g(\cdot) \triangleq \sum_k p_k F_{k,g}(\cdot)$, and $\sum_k p_k = 1$.*

Lemma 1. *Under Assumptions 1 to 3, the group loss function F_g are convex, M-Lipschitz continuous, L-Lipschitz smooth for any g.*

Proof. This simply follows by Definition 1, given $F_g(\cdot)$ is a linear combination of the local loss function $F_{f,g}(\cdot)$.

Let w_t^g be the model parameter maintained in the group g and at the t-th step. Let e be the current local epoch number, $e \in [0, E]$. We assume that the 0-th local epoch as the synchronization step, so e will be reset before the start of the communication round t. Then the update of FedGroup without inter-group aggregation can be described as:

$$\begin{aligned} \boldsymbol{w}_{t,e}^{k,g} &= \begin{cases} \boldsymbol{w}_t^g, & e = 0 \\ \boldsymbol{w}_{t,e-1}^{k,g} - \eta \nabla F_{k,g}(\boldsymbol{w}_{t,e-1}^{k,g}), & e \in [1, E] \end{cases} \\ \boldsymbol{w}_t^g &\triangleq \sum_k p_k \boldsymbol{w}_{t,E}^{k,g} \end{aligned} \tag{11}$$

Here we introduce an additional notation *virtual group model* $\boldsymbol{v}_{t,e}^g$ to measure the divergence between federated training and SGD-based centralized training, which is motivated by [11], [27], [30]. We assume that there is a virtual group model $\boldsymbol{v}_{t,e}^g$ that is centralized trained on the collection of members' data, and is synchronized with the federated model in each communication round. We introduce this notion formally below.

$$\boldsymbol{v}_{t,e}^g = \begin{cases} \boldsymbol{w}_t^g, & e = 0 \\ \boldsymbol{v}_{t,e-1}^g - \eta \nabla F_g(\boldsymbol{v}_{t,e-1}^g), & e \in [1, E] \end{cases} \tag{12}$$

Definition 2 (Intra-Group Gradient Divergence)**.** *Given a certain group membership, for any g and k, $\delta_{k,g}$ represents the gradient difference between the loss functions of client k and group p, as expressed below:*

$$\delta_{k,g} \triangleq \max_{\boldsymbol{w}} \|\nabla F_{k,g}(\boldsymbol{w}) - \nabla F_g(\boldsymbol{w})\| \tag{13}$$

And the intra-group gradient divergence is defined as Eq. (14),

$$\delta \triangleq \sum_{g \in \mathcal{G}} \sum_{k \in g.clients} p_g p_k \delta_{k,g} \tag{14}$$

Lemma 2 (Upper bound of the divergence of $\boldsymbol{w}_{t,e}^{k,g}$)**.** *Let Assumptions 1 to 3 hold, the upper bound of divergence between the FedGroup model and the virtual group model for any t, e is given by*

$$\|\boldsymbol{w}_{t,e}^{k,g} - \boldsymbol{v}_{t,e}^g\| \le \frac{\delta_{k,g}}{L}((\eta L + 1)^e - 1) \tag{15}$$

Proof. By the smoothness of $F_{k,g}(\cdot)$ and the Definition 2, we have

$$
\begin{aligned}
&\|\boldsymbol{w}_{t,e}^{k,g} - \boldsymbol{v}_{t,e}^{g}\| \\
&= \|\boldsymbol{w}_{t,e-1}^{k,g} - \eta\nabla F_{k,g}(\boldsymbol{w}_{t,e-1}^{k,g}) - \boldsymbol{v}_{t,e-1}^{g} + \eta\nabla F_g(\boldsymbol{v}_{t,e-1}^{g})\| \\
&\leq \|\boldsymbol{w}_{t,e-1}^{k,g} - \boldsymbol{v}_{t,e-1}^{g}\| + \eta\|\nabla F_{k,g}(\boldsymbol{w}_{t,e-1}^{k,g}) - \nabla F_g(\boldsymbol{v}_{t,e-1}^{g})\| \\
&\leq \|\boldsymbol{w}_{t,e-1}^{k,g} - \boldsymbol{v}_{t,e-1}^{g}\| + \eta\|\nabla F_{k,g}(\boldsymbol{w}_{t,e-1}^{k,g}) - \nabla F_{k,g}(\boldsymbol{v}_{t,e-1}^{g})\| \\
&\quad + \eta\|\nabla F_{k,g}(\boldsymbol{v}_{t,e-1}^{g}) - \nabla F_g(\boldsymbol{v}_{t,e-1}^{g})\| \\
&\leq (\eta L + 1)\|\boldsymbol{w}_{t,e-1}^{k,g} - \boldsymbol{v}_{t,e-1}^{g}\| + \eta\delta_{k,g}
\end{aligned} \tag{16}
$$

Let $h(e) = \|\boldsymbol{w}_{t,e}^{k,g} - \boldsymbol{v}_{t,e}^{g}\|$, then we can rewrite Eq. (16) as

$$
\begin{aligned}
&h(e) \leq (\eta L + 1)h(e-1) + \eta\delta_{k,i} \\
&\frac{h(e) + \delta_{k,g}/L}{h(e-1) + \delta_{k,g}/L} \leq \eta L + 1
\end{aligned} \tag{17}
$$

Given that $h(0) = \|\boldsymbol{w}_{t,0}^{k,g} - \boldsymbol{v}_{t,0}^{g}\| = 0$, by induction, we have

$$
g(e) + \frac{\delta_{k,g}}{L} \leq \frac{\delta_{k,g}}{L}(\eta L + 1)^e \tag{18}
$$

Therefore, Lemma 2 is proved. □

Combing Lemma 2 and Eq. (11) and using Jensen's inequality we get

$$
\begin{aligned}
\|\boldsymbol{w}_t^g - \boldsymbol{v}_{t,e}^g\| &\leq \sum_k p_k \|\boldsymbol{w}_{t,E}^{k,g} - \boldsymbol{v}_{t,e}^g\| \\
&\leq \frac{\delta}{L}((\eta L + 1)^E - 1)
\end{aligned} \tag{19}
$$

Consider the continuous of $F_g(\cdot)$ we have

$$
\|F_g(\boldsymbol{w}_t^g) - F_g(\boldsymbol{v}_{t,e}^g)\| \leq \frac{\delta M}{L}((\eta L + 1)^E - 1) \tag{20}
$$

Theorem 1 (Convergence Bound of FedGroup without inter-group aggregation)**.** *Let Assumption 1 to 3 hold and* $g, t, E, \boldsymbol{w}_t^g, \boldsymbol{v}_{t,e}^g$ *be defined therein. Then the convergence bound between the federated group model and the virtual group model is* $\frac{\delta M}{L}((\eta L + 1)^E - 1)$.

Then we extent Theorem 1 to the case where $\eta_g > 0$. First we introduce $\tilde{\boldsymbol{w}}_t^g$ to represent the model parameter of group g after inter-group aggregation. Then the update of FedGroup with inter-group aggregation can be described as Eq. (11) and

$$
\tilde{\boldsymbol{w}}_t^g = \boldsymbol{w}_t^g + \eta_g \sum_{l \in \mathcal{G}, l \neq g} \frac{\boldsymbol{w}_t^l}{\|\boldsymbol{w}_t^l\|} \tag{21}
$$

We replace $\boldsymbol{w}_t^g$ in Eq. (19) with $\tilde{\boldsymbol{w}}_t^g$ and derive

$$
\begin{aligned}
\|\tilde{\boldsymbol{w}}_t^g - \boldsymbol{v}_{t,e}^g\| &= \|\boldsymbol{w}_t^g - \boldsymbol{v}_{t,e}^g + \eta_g \sum_{l \in \mathcal{G}, l \neq g} \frac{\boldsymbol{w}_t^l}{\|\boldsymbol{w}_t^l\|}\| \\
&\leq \|\boldsymbol{w}_t^g - \boldsymbol{v}_{t,e}^g\| + \eta_g \sum_{l \in \mathcal{G}, l \neq g} \|\frac{\boldsymbol{w}_t^l}{\|\boldsymbol{w}_t^l\|}\| \\
&\leq \frac{\delta}{L}((\eta L + 1)^E - 1) + \eta_g(|\mathcal{G}| - 1)
\end{aligned} \tag{22}
$$

Then by the M-Lipschitz continuous of $F_g(\cdot)$ we get the convergence bound of FedGroup with inter-group aggregation:

$$
\|F_g(\tilde{\boldsymbol{w}}_t^g) - F_g(\boldsymbol{v}_{t,e}^g)\| \leq \frac{\delta M}{L}((\eta L + 1)^E - 1) + \eta_g(|\mathcal{G}| - 1) \tag{23}
$$

Note that, Eq. (23) degrades to Eq. (20) when $\eta_g = 0$ or $|\mathcal{G}| = 1$, which means the learning rate of inter-group aggregation is 0 (disabled) or there is only one group in FedGroup (be degraded to FedAvg framework).

V. Evaluation

In this section, we present the experimental results for FedGroup and FedGrouProx frameworks. We show the performance improvement of our frameworks on four open datasets. Then we demonstrate the effectiveness of our grouping strategy, which includes the clustering algorithm (group cold start) and the newcomer assignment algorithm (client cold start). We further study the relationship between the ternary cosine similarity and $\mathcal{M}'$. Our implementation is based on Tensorflow [31], and all code and data are publicly available at *github.com/morningD/GrouProx*. To ensure reproducibility, we fix the random seeds for all experiments in practice.

A. Experimental Setup

We evaluate FedGroup and FedGrouProx on four federated datasets, which including two image classification tasks, a synthetic dataset, a sentiment analysis task. In this section, we adopt the same notation for federated learning settings as Section III and as [18]: the local epoch $E = 20$, the number of selected clients per round $K = 20$, the pre-training scale $\alpha = 20$, the inter-group learning rate $\eta_g = 0.01$. The local solver is mini-batch SGD with $B = 10$. Besides, the learning rate η and FedProx hyperparameter μ in our experiments are consistent with the recommended settings of [18].

Datasets and Models

- EMNIST [19]. A 10-class handwritten digits image classification task, which is divided into 1,000 clients, each with only two classes of digits. We train a convex multinomial logistic regression (MCLR) model and a non-convex multilayer perceptron (MLP) model based on it. The MLP has one hidden layer with 128 hidden units.
- Federated Extended MNIST (FEMNIST) [20]. A 62-class handwritten digits and characters image classification task, which is built by resampling the EMNIST [20] according to the writer. Similar to the experiment of EMNIST, we train a MCLR model and a MLP model (one hidden layer with 512 hidden units) based on it.
- Synthetic. It's a synthetic federated dataset proposed by *Shamir et. al* [32]. Our hyperparameter settings of this data-generated algorithm are $\alpha = 1$ and $\beta = 1$, which

TABLE II
Statistics of Federated Datasets and Models.

Dataset	Devices	Samples	Model	# of Parameters
MNIST	1,000	69,035	MCLR	7,850
			MLP	101,770
FEMNIST	200	18,345	MCLR	20,410
			MLP	415,258
Synthetic(1,1)	100	75,349	MCLR	610
Sent140	772	40,783	LSTM	243,861

TABLE III
COMPARISONS WITH FEDAVG AND FEDPROX ON EMNIST, FEMNIST, SYNTHETIC, SENT140. THE ACCURACY IMPROVEMENT (↑) IS CALCULATED RELATIVE TO THE BEST SCORE OF FEDAVG AND FEDPROX. RCC: RANDOM CLUSTER CENTERS, RAC: RANDOMLY ASSIGN CLIENTS.

Dataset-Model	FedAvg [2]	FedProx [18]	FedGroup-RCC	FedGroup-RAC	FedGroup (Ours)	FedGrouProx (Ours)
MNIST-MCLR	89.3	90.5	93.5	88.1	95.7(↑ 5.2)	**95.8**(↑ 5.3)
MNIST-MLP	94.4	93.4	95.7	92.9	**97.6**(↑ 3.2)	97.1(↑ 2.7)
FEMNIST-MCLR	76.8	76.2	87.0	70.3	**87.8**(↑ 11.0)	87.1(↑ 10.3)
FEMNIST-MLP	79.8	78.5	91.5	75.5	**94.5**(↑ 14.7)	93.5(↑ 13.7)
Synthetic(1,1)-MCLR	72.6	80.4	92.3	—	**92.1**(↑ 11.7)	84.6(↑ 4.2)
Sent140-LSTM	71.1	71.1	76.3	70.0	**76.5**(↑ 5.4)	76.1(↑ 5.0)

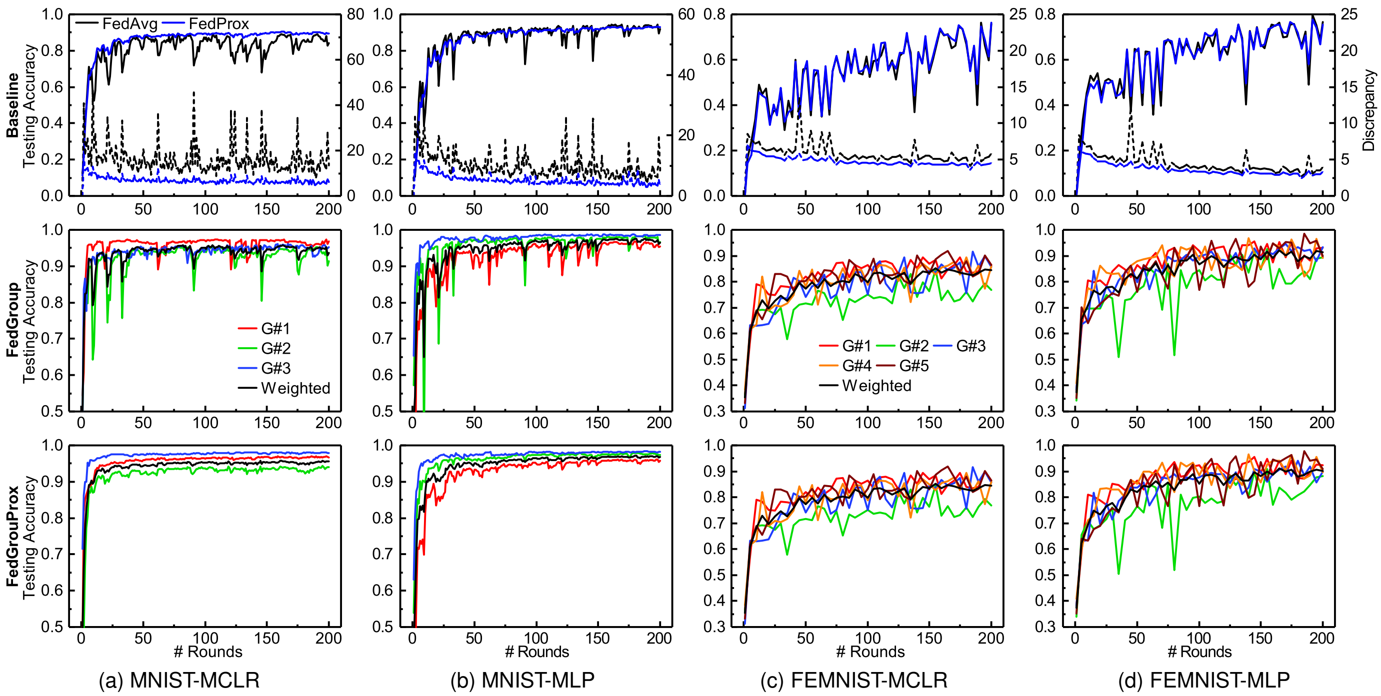


Fig. 3. Classfication accuracy evaluated on MNIST ($m = 3$) and FEMNIST($m = 5$). Top: test accuracy (straight lines) and discrepancy (dot lines) of FedAvg and FedProx; Middle: test accuracy of FedGroup; Bottom: test accuracy of FedGrouProx.

control the statistical heterogeneity among clients. We study a MCLR model based on it.

- Sentiment140 (Sent140) [21]. A tweets sentiment analysis task, which contains 772 clients, each client is a different twitter account. We explore a LSTM classifier based on it.

The statistics of our experimental datasets and models are summarized in TABLE II.

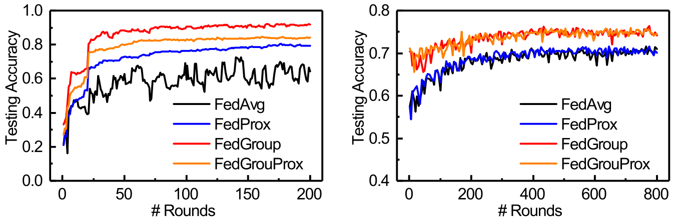
Baselines

- FedAvg [2]: the vanilla federated learning framework.
- FedProx [18]: the SOTA federated learning optimizer toward heterogeneous networks.

Metrics

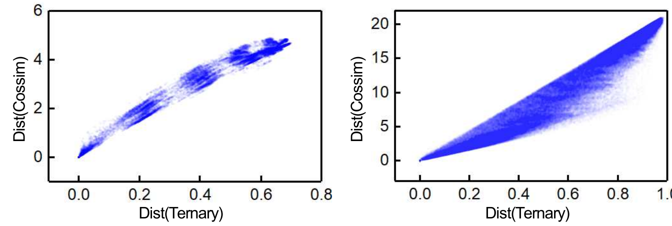
Since each client has a local test data set in our experimental setting, we evaluate its corresponding model based on these data. For example, in FedAvg and FedProx we evaluate the global model based on the test set for all clients. And in FedGroup and FedGrouProx we evaluate the group model based on the test set for the clients in this group. We leverage top-1 classification accuracy to measure the performance of the classifiers. For FedGroup and FedGrouProx, where there are multiple accuracies of groups with different sizes, we use a "weighted" accuracy to measure the overall performance and the weight is proportional to the test data size of each group. In fact, the "weighted" accuracy is equivalent to the sum of the misclassified sample count in all groups divided by the total test data size.

To make our results more comparable, the test clients of the group model is all the clients historically assigned to this group. As the number of training rounds increases, all clients will be included in the test. Note that the heterogeneity of client data will affect the convergence of the federated training, resulting in greater fluctuations in model accuracy during the training process. Therefore, we assume the early stopping [33] strategy is applied and we regard the maximum accuracy during training as the final score.



(a) Synthetic(1,1)-MCLR (b) Sent140-LSTM

Fig. 4. Calssification accuracy evaluated on Synthetic(1,1) ($m = 5$) and Sent140 ($m = 5$).



(a) FEMNIST-MLP (b) Sent140-LSTM

Fig. 5. The mapping from the pair-wise clustering (euclidean) distance in the similarity matrix $\mathcal{M}'$ to the pair-wise distance in the ternary cosine similarity matrix M, the pre-training scale $\alpha = 20$.

B. Effects of Proposed Framework

In TABLE III, we compare the evaluated results of our proposed FedGroup and FedGrouProx with baselines. Sine FedProx is not completely superior to FedAvg, we calculate the accuracy improvement (↑) related to the best score between them. The results show that FedGroup and FedGrouProx significantly improve the accuracy of models compared to the baselines. In particular, FedGroup improves absolute test accuracy by $+14.7\%$ on FEMNIST, which is highly statistical heterogeneity. Our evaluation shows that FedGroup is superior to FedGrouProx in almost all configurations, except for the Synthetic(1,1) experiment. To explore the effects of our proposed strategies, we further compare our clustering algorithm (Algorithm 4) and client cold start strategy (Eq. 10) with two random strategies: RCC (random cluster centers) and RAC (randomly assign cold clients, but the pre-training clients are retained in their groups). In the RCC setting, the accuracy is slightly degraded (expect Synthetic) but still surpasses the baselines. However, the RAC strategy leads to a significant decrease in accuracy (-11.06% on average) and the final scores are worse than the baselines. Therefore, the combination of our clustering algorithm and client cold start strategy is efficient and can achieve more performance improvements.

The details of the test accuracy in specific rounds are shown in Fig. 3 and Fig. 4. The empirical results illustrate that adding the proximal term can reduce the divergence caused by the heterogeneous data and make the training more stable. However, there is not enough evidence to suggest that adding the proximal term is significantly helpful in improving accuracy. Although the training procedure of FedGrouProx is more stable than that of FedGroup, the model accuracy of FedGroup is higher than FedGrouProx in most experiments.

C. Ternary Cosine Similarity

Finally, in Fig. 5, we study the difference between the similarity matrix $\mathcal{M}'$ and our ternary cosine similarity matrix $\mathcal{M}$ as clustering distance. The experiments show that the mapping from the pair-wise distance in $\mathcal{M}'$ to that in $\mathcal{M}$ is approximately linear, which means that the ternary similarity can be well adapted to the distance-based clustering algorithm like K-Means. In addition, the low-dimensional embedding method only needs to pre-train $\alpha * m$ clients, which is more computationally efficient and implementable than calculating $\mathcal{M}'$.

VI. Conclusion

In this work, we have presented two similarity-based clustered federated learning frameworks FedGroup and FedGrouProx, which can improve the model performance of federated training by dividing global optimization into groups of sub-optimization. We evaluated the proposed frameworks on four open datasets and shown the superiority of FedGroup compared to FedAvg and FedProx. FedGroup significantly improved 14.7% top-1 test accuracy on FEMNIST compared to FedAvg. Besides, our experiments on FedGrouProx have found that adding the proximal term can make the federated training more stable, but it does not significantly help to improve classification accuracy. The experiments of our novel clustering metrics, named ternary cosine similarity, shown that it is an appropriate linear approximation of the traditional cosine similarity. Our evaluation results on 7 models (expect Synthetic-MCLR) shown that FedGroup achieved higher classification accuracy compared to FedAvg, FedProx, and two random strategies.

References

[1] J. Konečný, H. B. McMahan, F. X. Yu, P. Richtárik, A. T. Suresh, and D. Bacon, "Federated learning: Strategies for improving communication efficiency," *arXiv preprint arXiv:1610.05492*, 2016.

[2] B. McMahan, E. Moore, D. Ramage, S. Hampson, and B. A. y Arcas, "Communication-efficient learning of deep networks from decentralized data," in *Proceedings of the 20th International Conference on Artificial Intelligence and Statistics (AISTATS)*, 2017, pp. 1273–1282.

[3] K. Bonawitz, H. Eichner, W. Grieskamp, D. Huba, A. Ingerman, V. Ivanov, C. Kiddon, J. Konecny, S. Mazzocchi, H. B. McMahan *et al.*, "Towards federated learning at scale: System design," in *Proceedings of the 2nd SysML Conference*, 2019.

[4] Q. Yang, Y. Liu, T. Chen, and Y. Tong, "Federated machine learning: Concept and applications," *ACM Transactions on Intelligent Systems and Technology (TIST)*, vol. 10, no. 2, pp. 1–19, 2019.

[5] T. Li, A. K. Sahu, A. Talwalkar, and V. Smith, "Federated learning: Challenges, methods, and future directions," *IEEE Signal Processing Magazine*, vol. 37, no. 3, pp. 50–60, 2020.

[6] M. Li, D. G. Andersen, J. W. Park, A. J. Smola, A. Ahmed, V. Josifovski, J. Long, E. J. Shekita, and B.-Y. Su, "Scaling distributed machine learning with the parameter server," in *Proceedings of the 11th USENIX Symposium on Operating Systems Design and Implementation (OSDI)*, 2014, pp. 583–598.

[7] 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.

[8] A. Krizhevsky and G. Hinton, "Learning multiple layers of features from tiny images," Citeseer, Tech. Rep., 2009.

[9] K. Simonyan and A. Zisserman, "Very deep convolutional networks for large-scale image recognition," in *Proceedings of the 3rd International Conference on Learning Representations (ICLR)*. IEEE, 2015.

[10] F. Sattler, S. Wiedemann, K.-R. Müller, and W. Samek, "Robust and communication-efficient federated learning from non-i.i.d. data," *IEEE Transactions on Neural Networks and Learning Systems (TNNLS)*, pp. 1–14, 2019.

[11] X. Li, K. Huang, W. Yang, S. Wang, and Z. Zhang, "On the convergence of fedavg on non-iid data," in *Proceedings of the 8th International Conference on Learning Representations (ICLR)*, 2020.

[12] M. Duan, D. Liu, X. Chen, Y. Tan, J. Ren, L. Qiao, and L. Liang, "Astraea: Self-balancing federated learning for improving classification accuracy of mobile deep learning applications," in *Proceedings of the IEEE 37th International Conference on Computer Design (ICCD)*. IEEE, 2019, pp. 246–254.

[13] F. Sattler, K.-R. Müller, and W. Samek, "Clustered federated learning: Model-agnostic distributed multitask optimization under privacy constraints," *IEEE Transactions on Neural Networks and Learning Systems (TNNLS)*, pp. 1–13, 2020.

[14] F. Sattler, K.-R. Müller, T. Wiegand, and W. Samek, "On the byzantine robustness of clustered federated learning," in *Proceedings of the IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP)*. IEEE, 2020, pp. 8861–8865.

[15] M. Xie, G. Long, T. Shen, T. Zhou, X. Wang, and J. Jiang, "Multi-center federated learning," *arXiv preprint arXiv:2005.01026*, 2020.

[16] A. Ghosh, J. Chung, D. Yin, and K. Ramchandran, "An efficient framework for clustered federated learning," *arXiv preprint arXiv:2006.04088*, 2020.

[17] C. Briggs, Z. Fan, and P. Andras, "Federated learning with hierarchical clustering of local updates to improve training on non-IID data," in *Proceedings of the IEEE International Joint Conference on Neural Networks (IJCNN)*, 2020, pp. 1–9.

[18] T. Li, A. K. Sahu, M. Zaheer, M. Sanjabi, A. Talwalkar, and V. Smith, "Federated optimization in heterogeneous networks," in *Proceedings of the 3rd SysML Conference*, 2020.

[19] Y. LeCun, L. Bottou, Y. Bengio, and P. Haffner, "Gradient-based learning applied to document recognition," *Proceedings of the IEEE*, vol. 86, no. 11, pp. 2278–2324, 1998.

[20] G. Cohen, S. Afshar, J. Tapson, and A. van Schaik, "EMNIST: Extending mnist to handwritten letters," in *Proceedings of the 2017 International Joint Conference on Neural Networks (IJCNN)*. IEEE, 2017, pp. 2921–2926.

[21] A. Go, R. Bhayani, and L. Huang, "Twitter sentiment classification using distant supervision," *CS224N project report, Stanford*, 2009.

[22] M. Duan, D. Liu, X. Chen, R. Liu, Y. Tan, and L. Liang, "Self-balancing federated learning with global imbalanced data in mobile systems," *IEEE Transactions on Parallel and Distributed Systems (TPDS)*, vol. 32, no. 1, pp. 59–71, 2020.

[23] C. Dwork, V. Feldman, M. Hardt, T. Pitassi, O. Reingold, and A. Roth, "The reusable holdout: Preserving validity in adaptive data analysis," *Science*, vol. 349, no. 6248, pp. 636–638, 2015.

[24] G. H. Golub and C. Reinsch, "Singular value decomposition and least squares solutions," in *Linear Algebra*. Springer, 1971, pp. 134–151.

[25] D. Arthur and S. Vassilvitskii, "k-means++: The advantages of careful seeding," Stanford, Tech. Rep., 2006.

[26] S. Sarkar and A. K. Ghosh, "On perfect clustering of high dimension, low sample size data," *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 2019.

[27] J.-w. Lee, J. Oh, S. Lim, S.-Y. Yun, and J.-G. Lee, "Tornadoaggregate: Accurate and scalable federated learning via the ring-based architecture," *arXiv preprint arXiv:2012.03214*, 2020.

[28] L. Liu, J. Zhang, S. Song, and K. B. Letaief, "Client-edge-cloud hierarchical federated learning," in *Proceedings of the IEEE International Conference on Communications (ICC)*. IEEE, 2020, pp. 1–6.

[29] S. Wang, T. Tuor, T. Salonidis, K. K. Leung, C. Makaya, T. He, and K. Chan, "Adaptive federated learning in resource constrained edge computing systems," *IEEE Journal on Selected Areas in Communications*, vol. 37, no. 6, pp. 1205–1221, 2019.

[30] S. U. Stich, "Local SGD converges fast and communicates little," in *Proceedings of the 7th International Conference on Learning Representations (ICLR)*, 2019.

[31] M. Abadi, P. Barham, J. Chen, Z. Chen, A. Davis, J. Dean, M. Devin, S. Ghemawat, G. Irving, M. Isard *et al.*, "Tensorflow: A system for large-scale machine learning," in *Proceedings of the 12th USENIX Symposium on Operating Systems Design and Implementation (OSDI)*, 2016, pp. 265–283.

[32] O. Shamir, N. Srebro, and T. Zhang, "Communication-efficient distributed optimization using an approximate newton-type method," in *Proceedings of the 31th International conference on machine learning (ICML)*, 2014, pp. 1000–1008.

[33] Y. Yao, L. Rosasco, and A. Caponnetto, "On early stopping in gradient descent learning," *Constructive Approximation*, vol. 26, no. 2, pp. 289–315, 2007.