FedGroup: Efficient 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

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 the global model, further degrading performance. In this paper, we propose a novel clustered federated learning (CFL) framework FedGroup, in which we 1) group the training of clients based on the similarities between the clients' optimization directions for high training performance; 2) construct a new data-driven distance measure to improve the efficiency of the client clustering procedure. 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 FL optimizer *FedProx*. The convergence and complexity are analyzed to demonstrate the efficiency of our proposed framework. We also evaluate FedGroup and FedGrouProx (combined with *FedProx*) on several open datasets and made comparisons with related CFL frameworks. The results show that FedGroup can significantly improve absolute test accuracy by $+14.1\%$ on FEMNIST compared to *FedAvg*, $+3.4\%$ on Sentiment140 compared to *FedProx*, $+6.9\%$ on MNIST compared to *FeSEM*.

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 to construct 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. With the advantage of privacy-preserving, Federated Learning is currently the most attractive distributed machine learning framework. 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 be restricted. Therefore, the high statistical heterogeneity is a challenging problem in 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 reduces $\sim55\%$ for CNN trained on highly skewed CIFAR-10 [8]. 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* and indicates that the 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 also has adverse effects on federated training. 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]–[18] 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.

In this paper, we propose an efficient and accurate clustered federated learning framework FedGroup, which clusters clients into multiple groups based on a new decomposed data-driven measure between their parameter updates. In each communication round, each active client only contributes its local optimization result to the corresponding group model. The framework still maintains an auxiliary server to address the cold start issues of new devices. To improve the performance of high-dimension low-sample size (HDLSS) parameter updates clustering, we use a novel data-driven measure of cosine dissimilarity called Euclidean distance of Decomposed Cosine similarity (EDC), which can also avoid the concentration phenomenon of ℓ_p distances in high dimensional data clustering [19]. Furthermore, by combining FedGroup with the federated optimizer *FedProx* [20], FedGroup can be revised as FedGrouProx, which be explored in our experiments.

With the above methods, FedGroup can significantly improve test accuracy by $+6.9\%$ on MNIST [21], $+26.9\%$ on

The main contributions of this paper are summarized as follows.

- We propose two novel clustered federated learning frameworks, FedGroup and FedGrouProx, and show their superiority on four open datasets (with statistical heterogeneity) compared to several FL and CFL frameworks.
- We propose a decomposed data-driven measure named EDC to improve the effectiveness of client clustering in FedGroup, which prevents the adverse effects of concentration phenomenon caused by directly clustering HDLSS model parameters or direction vectors.
- Our framework presents an efficient cold start strategy for the groups and the newcomers, which provides a new approach to improve the scalability and practicality of the previous works. We demonstrate its efficiency experimentally. In addition, we open source the code of FedGroup to contribute to the community.

The rest of this paper is organized as follows. Section II provides the background of FL and an overview of related works. Section III shows the motivation and the design of FedGroup. The evaluation results are presented and analyzed in Section IV. Section V concludes the paper.

II. BACKGROUND

A. Federated Learning

In this section, we introduce the most widely adopted FL algorithm FedAvg and briefly explain the 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. The distributed objective of FL is:

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

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

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

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.

Algorithm 1 Federated Averaging (FedAvg)

1: **procedure** FL SERVER TRAINING
2:     Initialize global model $\boldsymbol{w}_0$, then $\boldsymbol{w}_1 \leftarrow \boldsymbol{w}_0$.
3:     **for** each communication round $t = 1, 2, ..., T$ **do**
4:         $S_t \leftarrow$ Server selects a random subset of K clients.
5:         Server broadcasts $\boldsymbol{w}_t$ to all selected clients.
6:         **for** each activate client $i \in S_t$ parallelly **do**
7:             $\Delta \boldsymbol{w}_{t+1}^i \leftarrow$ **ClientUpdate**($i, \boldsymbol{w}_t$).
8:         $\boldsymbol{w}_{t+1} \leftarrow \boldsymbol{w}_t + \sum_{i \in S_t} \frac{n_i}{n} \Delta \boldsymbol{w}_{t+1}^i$
9: **function** CLIENTUPDATE($i, \boldsymbol{w}$)
10:     $\hat{\boldsymbol{w}} \leftarrow \boldsymbol{w}$.
11:     **for** each local epoch $e = 1, 2, ..., E$ **do**
12:         $\boldsymbol{w} \leftarrow \boldsymbol{w} - \eta \nabla L(b; \boldsymbol{w})$ for local batch $b \in \mathcal{B}_i$.
13:     **return** $\Delta \boldsymbol{w} \leftarrow \boldsymbol{w} - \hat{\boldsymbol{w}}$

In practice, a typical implementation of FL system [3] includes a cloud server that maintains a global model and multiple participating devices, which communicate with the server through a network. At each communication round t, the 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. Then, the locally-computed parameter updates $\Delta \boldsymbol{w}_i^t$ from these clients that completed the training within the time budget are collected and aggregated. Finally, the server will update the global to w^{t+1} and finish the current round. In general, the FL training task requires hundreds of rounds to reach target accuracy.

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 $\mathcal{B}_i$ is the batches of training data of client i, 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 K/N. For the IID setting, a higher participation rate can improve the convergence rate, but for the 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 FL system is to bound the discrepancies between the local optimization objectives and the global objective. To alleviate the divergences, Li *et al.* [20] introduce a penalty (aka proximal term) to the local objective function:

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

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 the large-scale FL system is statistical heterogeneity (e.g. non-IID, size imbalanced, class imbalanced) [3], [5], [10], [17], [24], [25]. The conventional way is to train a consensus global model upon incongruent data, which yields unsatisfactory performance in the high heterogeneity setting [7]. Instead of optimizing a consensus global model, CFL divides the optimization goal into several sub-objectives and follows a *Pluralistic Group* architecture [26]. CFL maintains multiple groups (or clusters) models, which are more specialized than the consensus model and achieve high accuracy.

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 inefficient 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 efficiency 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 model accuracy improvements of IFCA are significant, IFCA needs to broadcast all group models in each round and is sensitive to the initialization with probability success. FeSEM [15] uses a ℓ_2 distance-based stochastic expectation maximization (EM) algorithm instead of distance-based neighborhood methods. However, ℓ_2 distance often suffer in HDLSS situation which is known as distance concentration phenomenon in high dimension [19]. The distance concentration of ℓ_p will cause the violation of neighborhood structure [27], which has adverse effects on the performance of pairwise distance-based clustering algorithms such as K-Means, k-Medoids and hierarchical clustering [19]. Similarity to the cosine similarity-based clustering method [13], Briggs *et al.* propose an agglomerative hierarchical clustering method named FL+HC [17]. It 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 in the clustering process and no newcomer devices, which is unpractical in a large-scale FL system.

III. FedGroup and FedGrouProx

A. Motivation

Before introducing 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 on MNIST [21] following

TABLE I
QUANTITATIVE RESULTS OF FEDAVG TRAINING BASED ON NON-IID AND IID MNIST WITH DIFFERENT #CLASSES/CLIENT.

# Classes	Discrepancy		Accuracy		# Round to Reach
	Mean	Variance	Max	Median	Target Acc-85%
1	0.767	0.11	87.9%	79.3%	39
3	0.767	0.073	90.9%	87.0%	14
5	0.685	0.021	91.3%	90.0%	10
10 (IID)	0.627	0.0082	92.2%	91.5%	4

the instructions of [20]. We manipulate the statistical heterogeneity by forcing each client to have only a limited number of classes of data. All distributed client data is randomly subsampled from the original MNIST dataset without replacement, and the training set size follows a power law.

We build three non-IID and one IID MNIST datasets with different class limitations. The total client size $N = 1000$, 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 $T = 300$ rounds of training and evaluated the global model on the local test set. Note that although we use the same test data in each round, we did not fine-tune the model structure based on the testing results, so we ignore the possibility of test data leakage [28].

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 arithmetic mean of norm difference between the clients' model weights and the latest global model weights. Specifically, the discrepancy in communication round t is defined as:

$$Discrepancy(t) \triangleq \frac{1}{|S_t|} \sum_{i \in S_t} \|\boldsymbol{w}_i - \boldsymbol{w}_t\|. \qquad (4)$$

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

The quantitative results which are shown in TABLE I support our observations. As the heterogeneity increases, the variance of the discrepancy significantly decreases by 92.5% (from 0.11 to 0.0082), and the max accuracy is increased +4.3% (from 87.9% to 92.2%). The number of the required communication rounds to reach 85% accuracy is also significantly reduced in the IID setting, which means faster convergence and less communication consumption.

B. Framework Overview

To tackle the statistical heterogeneity challenge in FL, we propose FedGroup, which groups the training of clients based on the similarities between the client's local optimization solutions. Our proposed framework is inspired by CFL [13], which clusters clients by a recursive bi-partitioning algorithm.
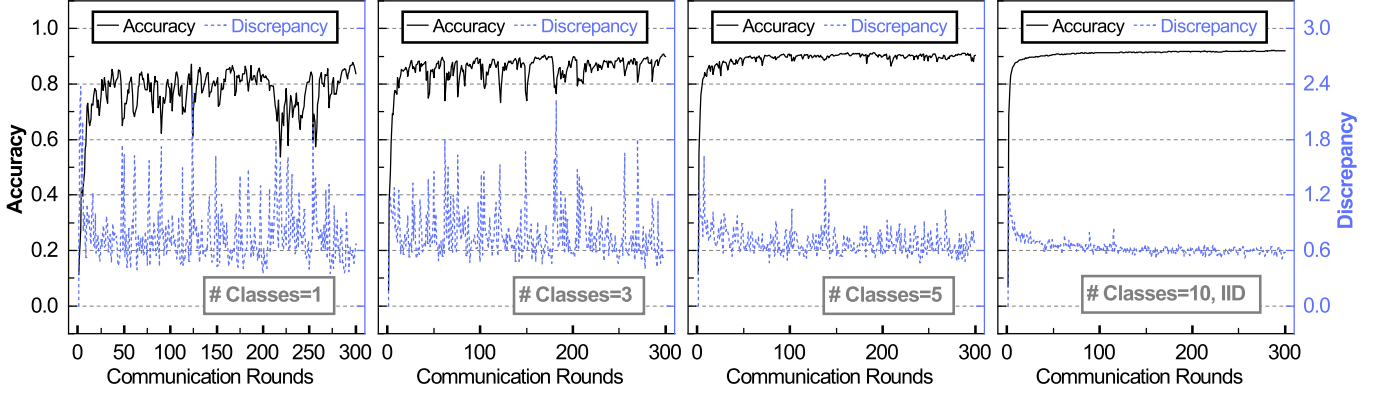
Fig. 1. A FedAvg training procedure on three non-IID MNIST datasets and one IID MNIST dataset to illustrate the effects of statistical heterogeneity on model accuracy and discrepancy. From left to right, the number of classes of training data per client increase, which means the degree of data heterogeneity decreases. The discrepancy is defined in Equation (4).

CFL assumes the clients have incongruent risk functions (e.g. randomly swapping out the labels of training data), and the goal of clustering is to separate clients with incongruent risk functions. FedGroup borrows the client clustering idea of CFL. But unlike previous CFL-based frameworks [13]–[16], [29], our grouping strategy is static, which avoids rescheduling clients for each round. More significantly, we propose the Euclidean distance of Decomposed Cosine similarity (EDC) for efficient clustering of high dimensional direction vectors, which can be regarded as a decomposed variant of MADD [19].

Before we go into more detail, we first show the general overview of the FedGroup. The federated training procedure of FedGroup is shown in Fig. 2. Note that Fig. 2 is a schematic diagram, it does not represent the physical deployment of connections or devices. In practice, the clients can be connected mobiles or IoT devices, the auxiliary server can be deployed in the cloud (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 auxiliary server, a certain amount of groups, and multiple clients, each of them maintains the 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 (empty group) or a client is not in any groups (e.g. a newcomer joins the training later). The newcomer device uses a cold start algorithm to determine the assigned group (we call the unassigned client is cold), we will explain the details of the cold start algorithm later. For the dataset, each client c_i has a training set and a test set according to its local data distribution $p_{data}^{(c_i)}$.

As shown in the Fig. 2, FedGroup has three model transmission processes, including intra-group aggregation (①), inter-group aggregation (②), and optimization gradient upload (③). First, the auxiliary server determines the initial model and
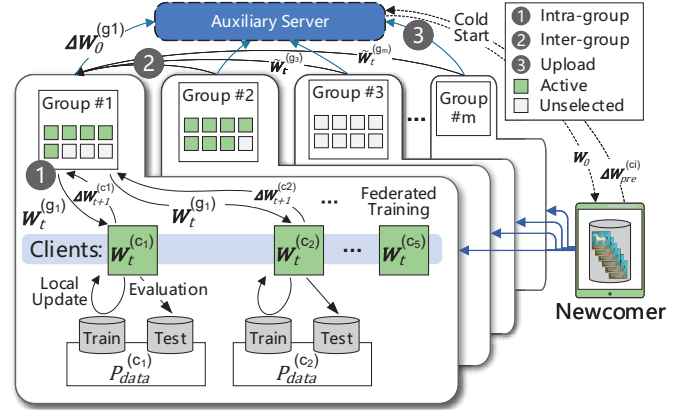

Fig. 2. An overview of FedGroup.

optimization direction of each group through clustering. Then, each group federally trains on a certain set of clients based on their training data to optimize its group model, and evaluates the group model on the same set of clients based on its test data. Specifically, each group broadcasts its model parameters to their clients and then aggregates the updates from these clients using *FedAvg* in parallel, we call this aggregation procedure the intra-group aggregation. After all federated trainings of groups complete, the models are aggregated using a certain weight, which we call inter-group aggregation. Note that, unlike the previous FedAvg-based frameworks [2], [12], [20], [30], [31], the optimization gradients in the server is not broadcast to all clients or groups, we only maintain this gradients for the cold start of newcomers.

Training: The details of the training procedure of FedGroup and FedGrouProx (revised by FedProx) are shown in Algorithm 2, where m controls the number of groups. Note that, FedGroup can be regarded as a special case of FedGrouProx with $\mu = 0$. Before training, the server first initializes the global model and group models to the same initial weights w_0. At the beginning of each round, a random subset of

Algorithm 2 FedGroup and FedGrouProx

Input: Clients set $\mathcal{C} \leftarrow \{c_1, c_2, ..., c_n\}$, groups set $\mathcal{G} \leftarrow \{g_1, g_2, ..., g_m\}$, initial group parameters set $\mathcal{W}_0^{(G)} \leftarrow \{\boldsymbol{w}_0^{(g_1)}, \boldsymbol{w}_0^{(g_2)}, ..., \boldsymbol{w}_0^{(g_m)}\}$, $g_j.clients \leftarrow \{c_i | c_i \text{ is in group } g_j, \forall c_i \in \mathcal{C}\}$, initial model parameters $\boldsymbol{w}_0$, number of communication rounds 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:    $\mathcal{W}_1^{(G)} \leftarrow \{\boldsymbol{w}_1^{(g_1)}, \boldsymbol{w}_1^{(g_2)}, ..., \boldsymbol{w}_1^{(g_m)}\} \leftarrow$ initialized $\mathcal{W}_0^{(G)}$ by $\boldsymbol{w}_0$.
3:    **for** each communication round $t = 1, 2, ..., T$ **do**
4:       $S_t \leftarrow$ Server selects a 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**$(\tilde{\mathcal{W}}_{t+1}^{(G)}, \eta_g)$.
11: **function** INTRAGROUPUPDATE$(S_t, \boldsymbol{w}_t)$
12:    **if** S_t is $\emptyset$ **then return** $\boldsymbol{w}_t$. //Empty group.
13:    IF **FedGroup**: $\boldsymbol{w}_{t+1} \leftarrow FedAvg(S_t, \boldsymbol{w}_t)$. //Ref. Algorithm 1.
14:    IF **FedGrouProx**: $\boldsymbol{w}_{t+1} \leftarrow FedProx(S_t, \boldsymbol{w}_t, \mu)$. //Ref. [20].
15:    **return** $\boldsymbol{w}_{t+1}$
16: **function** INTERGROUPAGGREGATION$(\tilde{\mathcal{W}}_{t+1}, \eta_g)$
17:    **for** each group parameter $\tilde{\boldsymbol{w}}_{t+1}^{(g_j)}$ in $\tilde{\mathcal{W}}_{t+1}$ parallelly **do**
18:       $\Delta\tilde{\boldsymbol{w}}_{t+1}^{(g_j)} \leftarrow \eta_g \sum_{l \neq j} \frac{\tilde{\boldsymbol{w}}_{t+1}^{(g_l)}}{\|\tilde{\boldsymbol{w}}_{t+1}^{(g_l)}\|}$, then $\boldsymbol{w}_{t+1}^{(g_j)} \leftarrow \tilde{\boldsymbol{w}}_{t+1}^{(g_j)} + \Delta\tilde{\boldsymbol{w}}_{t+1}^{(g_j)}$.
19:    **return** $\mathcal{W}_{t+1} \leftarrow \{\boldsymbol{w}_{t+1}^{(g_1)}, \boldsymbol{w}_{t+1}^{(g_2)}, ..., \boldsymbol{w}_{t+1}^{(g_m)}\}$

---

clients is selected to participate in this round of training (line 4). Each group trains the model using FedAvg and 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 server refreshes the global model by averaging all group models (line 11). Some details not shown in Algorithm 2 are that each agent calculates and stores its latest update after 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 in the next two sections. When $\eta_g = 0$, our framework is pluralistic [26] like the previous CFL works (e.g. [13], [16]). The inter-group aggregation ($\eta_g > 0$) constructs a novel *semi-pluralistic* architecture for CFL-based frameworks, and we will explore it experimentally in Section IV-B.

C. Group Cold Start

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 local data distributions are non-IID or called statistical heterogeneity. Li *et al.* [11] theoretically prove that the heterogeneity will also slow down the convergence rate. Instead of optimizing a complex global goal, why not divide it into several sub-optimization goals? Based on this idea, we manage to group the trainings of clients based on the proximities between their local optimizations. But there are two following questions:

(1) how to measure the proximity or distance, and (2) how to determine the optimization goals of each group before training.

For the choice of measure, a heuristic way is to use the ℓ_2 distance between models, which has miserable performance as we mentioned in II-B. The loss value can be used as a surrogate for the proximity of difference domains [32], but the huge computational overhead involved in model inferencing. The cosines similarity between the gradients calculated by backpropagation or the updates of model parameters is an alternative measure. The cosine similarity between the updates of any two clients c_i and c_j is defined by:

$$\mathcal{S}(i, j) \triangleq \frac{< \Delta\boldsymbol{w}_t^{(c_i)}, \Delta\boldsymbol{w}_t^{(c_j)} >}{\|\Delta\boldsymbol{w}_t^{(c_i)}\| \ \|\Delta\boldsymbol{w}_t^{(c_j)}\|}, \quad (5)$$

and the pairwise cosine similarity matrix $\mathcal{M} \subset \mathbb{R}^{n \times n}$ can be written with cosine similarity kernel K as follows:

$$\mathcal{M} = K(\Delta\boldsymbol{W}, \Delta\boldsymbol{W}), \ \mathcal{M}_{ij} = \mathcal{S}(i, j). \quad (6)$$

The computational complexity of calculating $\mathcal{M}$ is $O(n^2 d_{\boldsymbol{w}}^2)$, n and $d_{\boldsymbol{w}}$ are the number of clients and number of parameters, respectively. We assume the all parameters updates $\Delta\boldsymbol{w}_t$ are flattened row vectors, so $\Delta\boldsymbol{w}_t \subset \mathbb{R}^{1 \times d_{\boldsymbol{w}}}$. In general, the number of parameters $d_{\boldsymbol{w}}$ is huge and $d_{\boldsymbol{w}} \gg n$ (HDLSS), which make the pairwise cosine similarity-based clustering methods computationally inefficient. Unlike ℓ_p distance, the expectation of $\mathcal{S}$ asymptotically remains constant as dimensionality increases [27], which is friendly to the clustering in high dimensional data. Unfortunately, $\mathcal{S}$ is not suitable for low-dimensional situations because the variance of it is $O(1/d_{\boldsymbol{w}})$. Therefore, we extend the data-driven method MADD [19] to our cosine similarity-based case, so we can reduce the observation bias by using the mean of residuals of $\mathcal{S}$, fox example:

$$MADC(i, j) = \frac{1}{n-2} \sum_{z \neq i, j} |\mathcal{S}(i, z) - \mathcal{S}(j, z)|. \quad (7)$$

The above dissimilarity measure is based on the Mean of Absolute Differences of pairwise Cosine similarity, so we call it MADC. However, MADC and $\mathcal{M}$ have the same computation complexity and both are proximity measures, which means they cannot be applied for the efficient Euclidean distance-based clustering algorithms. These motivate us to develop a variant of MADC called Euclidean distance of Decomposed Cosine similarity (EDC), which is defined by:

$$EDC(i, j) = \frac{1}{m} \sqrt{\sum_{v \in \boldsymbol{V}} (\mathcal{S}(i, v) - \mathcal{S}(j, v))^2}, \ or$$

$$EDC(i, j) = \frac{1}{m} \|K(\Delta\boldsymbol{w}_t^{(c_i)}, \boldsymbol{V}^T) - K(\Delta\boldsymbol{w}_t^{(c_j)}, \boldsymbol{V}^T)\|,$$
$$\boldsymbol{V} = SVD(\Delta\boldsymbol{W}^T, m), \boldsymbol{V} \subset \mathbb{R}^{d_{\boldsymbol{w}} \times m}. \quad (8)$$

Instead of calculating the pairwise similarity, EDC first decomposes the updates of models into m directions by using truncated Singular Value Decomposition (SVD) algorithm [33], so then only the similarities between the updates

and these directions will be calculated. It is worth noting that the complexity of truncated SVD is only $O(2m^2 d_{\boldsymbol{w}})$ for $d_{\boldsymbol{w}} \gg m$ and hence the computational complexity of EDC is $O(m^2 d_{\boldsymbol{w}}^2)$. Some previous works (i.e. [13], [17]) calculate the pairwise cosine similarity based on all participants, so n is usually hundreds or thousands. Furthermore, [13], [17] leverage the hierarchical clustering strategies, which are recursively and computationally expensive.

To determine the optimization goals of each group, FedGroup clusters the parameter updates directions of clients into m groups using K-Means++ [34] algorithm based on EDC. 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. Moreover, training all clients to cluster their updated directions is not communication-friendly and practically achievable. In other words, FedGroup performs a low-dimensional embedding of the local updates matrix $\Delta \boldsymbol{W}$, following by K-Means++ clustering. Our calculation of client clustering is following the calculation of the similarity matrix, which means the above calculations only require one round of communication. We call the combination of the above two processes as group cold start, and the details are shown in Algorithm 3. For comparison, we also provide the MADC version of FedGroup, which is clustered using the hierarchical strategy with the complete linkage.

After the server performs the group cold start, the optimization direction of group j, which is measured by $\Delta \boldsymbol{w}_0^{(g_j)}$ is determined, and the 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. The optimization gradients will be uploaded to the auxiliary server for cold start of the newcomers.

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 systems. For example, the drop-out may occur due to the network jitter. We will show the efficiency of EDC in Section IV.

D. Client Cold Start

As described before, the group cold start algorithm selects a random subset (αm) of the clients for pre-training, so the remaining clients $(n - \alpha m)$ are cold clients and are not in any groups. Since the federated training network is dynamic, the new devices 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:

$$g^* = \operatorname*{argmin}_{j} \frac{-\cos(\sphericalangle(\Delta \boldsymbol{w}_0^{(g_j)}, \Delta \boldsymbol{w}_{pre}^{(i)})) + 1}{2}. \quad (9)$$

Algorithm 3 Group Cold Start

Input: Clients set $\mathcal{C}$, number of group m, global initial model $\boldsymbol{w}_0$, pre-training scale hyperparameter α.
Output: Groups set $\mathcal{G}$, set of group parameters $\mathcal{W}_0^{(G)}$, set of group updates $\Delta \mathcal{W}_0^{(G)}$.

1: **procedure** GROUP COLD START
2:     $S \leftarrow$ Server selects a random subset of $\alpha * m$ clients.
3:     Server broadcasts $\boldsymbol{w}_0$ to all selected clients.
4:     **for** each client c_i in S parallelly **do**
5:         $\Delta \boldsymbol{w}_0^{(c_i)} \leftarrow flatten(\ \textbf{ClientUpdate}(i, \boldsymbol{w})\ )$. //Ref. Algorithm 1.
6:     $\Delta \boldsymbol{W} \leftarrow [\Delta \boldsymbol{w}_0^{(c_1)}, \Delta \boldsymbol{w}_0^{(c_2)}, \dots, \Delta \boldsymbol{w}_0^{(c_{\alpha * m})}]$.
7:     **if** MADC **then**:
8:         $\mathcal{M} \leftarrow K(\Delta \boldsymbol{W}, \Delta \boldsymbol{W})$. //Ref. Eq. (6)
9:         Proximity matrix $\mathcal{M}_p \leftarrow$ **Calculate MADC**($\mathcal{M}$). //Ref. Eq. (7)
10:         $[g_1.clients, \dots, g_m.clients] \leftarrow$ Hierarchical Clustering($\mathcal{M}_p, m$).
11:     **if** EMD **then**:
12:         $V \leftarrow truncated\ SVD(\Delta \boldsymbol{W}^T, m)$.
13:         Distance matrix $\mathcal{M}_d \leftarrow K(\Delta \boldsymbol{W}, V^T)$. //Ref. Eq. (8)
14:         $[g_1.clients, \dots, g_m.clients] \leftarrow$ K-Means++($\mathcal{M}_d, m$).
15:     **for** g_j in $\mathcal{G} \leftarrow [g_1, \dots, g_m]$ **do**
16:         $\boldsymbol{w}_0^{(g_j)} \leftarrow Average([\Delta \boldsymbol{w}_0^{(c_i)}, \forall c_i \in g_j.clients])$.
17:         $\Delta \boldsymbol{w}_0^{(g_j)} \leftarrow \boldsymbol{w}_0^{(g_j)} - \boldsymbol{w}_0$.
18:     $\mathcal{W}_0^{(G)} \leftarrow [\boldsymbol{w}_0^{(g_1)}, \dots, \boldsymbol{w}_0^{(g_m)}]$, and $\Delta \mathcal{W}_0^{(G)} \leftarrow [\Delta \boldsymbol{w}_0^{(g_1)}, \dots, \Delta \boldsymbol{w}_0^{(g_m)}]$.
19:     **return** $\mathcal{G}, \mathcal{W}_0^{(G)}, \Delta \mathcal{W}_0^{(G)}$

Suppose the newcomer i joins the training network in round t, then the $\Delta \boldsymbol{w}_{pre}^{(i)}$ is the pre-training gradient of newcomer base on the global initial model $\boldsymbol{w}_0$. We schedule the newcomer i to group g^* to minimize the normalized cosine dissimilarity. With this mechanism, FedGroup does not need to broadcast all groups' models for clustering.

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 federated learning systems.
- 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 decomposed data-driven measure named EDC, 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. However, we emphasize that the pre-training procedure does not occupy a whole round, the client can continue to train $E-1$ epochs and upload the parameters updates for the intra-group aggregation.

IV. EVALUATION

In this section, we present the experimental results for FedGroup and FedGrouProx frameworks. We show the perfor-

mance improvements 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 EDC and MADC.

Our implementation is based on Tensorflow [35], and all code and data are publicly available at *github.com/morningD/GrouProx*. To ensure reproducibility, we fix the random seeds of the clients' selection and initialization.

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 [20]: the local epoch $E = 20$, the number of selected clients per round $K = 20$, the pre-training scale $\alpha = 20$, the inter-group aggregation is disable. 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 [20].

Datasets and Models

- MNIST [21]. 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) [22]. A 62-class handwritten digits and characters image classification task, which is built by resampling the EMNIST [22] 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* [36]. Our hyperparameter settings of this data-generated algorithm are $\alpha = 1, \beta = 1$, which control the statistical heterogeneity among clients. We study a MCLR model based on it.
- Sentiment140 (Sent140) [23]. 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.

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

Baselines

- FedAvg [2]: the vanilla federated learning framework.
- FedProx [20]: the popular federated learning optimizer toward heterogeneous networks.
- IFCA [16]: An CFL framework that minimizes the loss functions while estimating the cluster identities.
- FeSEM [15]: An ℓ_2 distance-based CFL framework that minimizes the expectation of discrepancies between clients and groups stochastically.

Evaluation Metrics

Since each client has a local test set in our experimental setting, we evaluate its corresponding group 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 use top-1 classification accuracy to measure the performance of the classifiers. Given that FedGroup has 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 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 would be included in the test. Note that the heterogeneity will affect the convergence, resulting in greater fluctuations in model accuracy during the training process. Therefore, we assume the early stopping [37] strategy is applied and we regard the maximum test accuracy during training as the final score. The number of groups of all CFL-based frameworks remains the same for each dataset.

B. Effects of Proposed Framework

In TABLE III, we compare the evaluated results of FedGroup (FG) and FedGrouProx (FGP) with baselines. We calculate the accuracy improvements related to the FeSEM. The results show that FG, FGP, IFCA are significantly superior to other frameworks. In particular, FG-EDC improves absolute test accuracy by $+26.9\%$ on FEMNIST, which is highly statistical heterogeneity. The FeSEM performs worst in all datasets expect Synthetic(1,1), which can be interpreted as the lower dimensionality ($d_{\boldsymbol{w}} = 610$) of model. The EDC is mostly slightly better than MADC, but this gap is shortened in the low dimensions cases, especially Synthetic(1,1). To investigate the effects of our proposed strategies, we further perform two ablation studies: 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 moderately degraded (expect Synthetic) but still surpasses the FeSEM. The RAC strategy leads to a significant decrease in accuracy (-7.86% on average) and the final scores are even worse than the FedAvg. Therefore, the combination of our clustering algorithm and client cold start strategy is efficient and can achieve more performance improvements.

TABLE III

COMPARISONS WITH FEDAVG, FEDPROX, IFCA, FESEM, FEDGROUP (FG), FEDGROUPROX (FGP) ON MNIST, FEMNIST, SYNTHETIC, SENT140. ABLATION STUDIES OF FEDGROUP: RANDOM CLUSTER CENTERS (RCC), RANDOMLY ASSIGN COLD (I.E. NEWCOMERS) CLIENTS (RAC). THE ACCURACY IMPROVEMENTS ↑ ARE CALCULATED RELATIVE TO THE ℓ_2 DISTANCE-BASED CFL FRAMEWORK FESEM.

| Dataset-Model	FedAvg [2]	FedProx [20]	IFCA [16]	FeSEM [15]	FG-RCC-EDC	FG-RAC-EDC	FG-EDC	FGP-EDC	FG-MADC	FGP-MADC
MNIST-MCLR	89.8	90.7	95.3	89.1	92.6	88.8	96.0(↑ 6.9)	**96.1**(↑ 7.0)	95.6(↑ 6.5)	95.3(↑ 6.2)
MNIST-MLP	95.3	94.6	97.8	93.9	96.5	94.3	**97.9**(↑ 4.0)	97.7(↑ 3.8)	**97.9**(↑ 4.0)	97.7(↑ 3.8)
FEMNIST-MCLR	78.8	77.9	87.9	65.6	88.8	74.7	**91.7**(↑ 26.1)	**91.7**(↑ 26.1)	91.2(↑ 25.6)	90.7(↑ 25.1)
FEMNIST-MLP	82.7	81.0	95.5	69.9	91.5	78.4	**96.8**(↑ 26.9)	95.4(↑ 25.5)	94.6(↑ 24.7)	93.6(↑ 23.7)
Synthetic(1,1)-MCLR	74.7	81.5	**95.5**	92.1	92.9	—	88.4(↓ 3.7)	87.4(↓ 4.7)	92.1(=)	88.5(↓ 3.6)
Sent140-LSTM	72.2	71.9	**76.8**	70.0	76.1	72.2	75.3(↑ 5.3)	75.4(↑ 5.4)	75.8(↑ 5.8)	75.4(↑ 5.4)
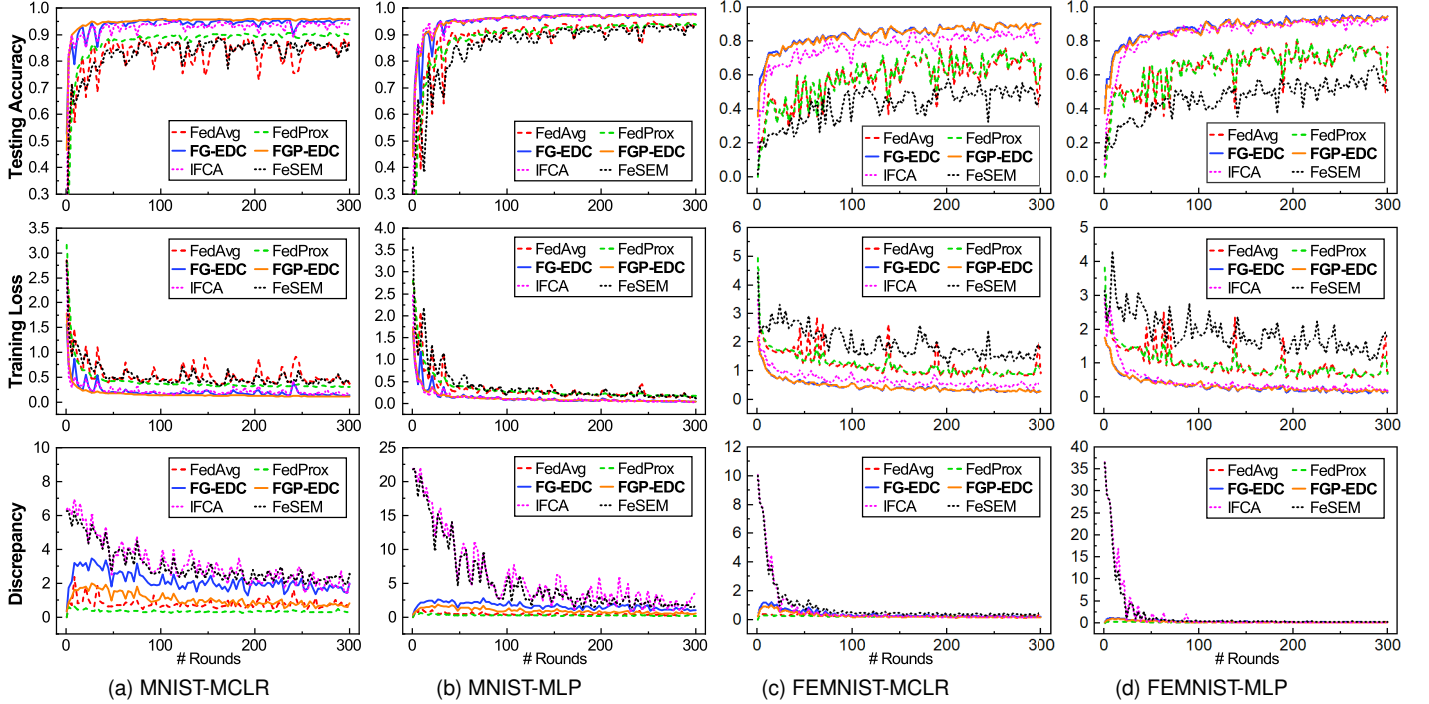
Fig. 3. Evaluation results on MNIST ($m = 3$) and FEMNIST($m = 5$). Top: test accuracy; Middle: mean training loss; Bottom: discrepancy between clients and server (FedAvg and FedProx) or weighted discrepancy between clients and groups (FedGroup, FedGrouProx, IFCA, FeSEM).
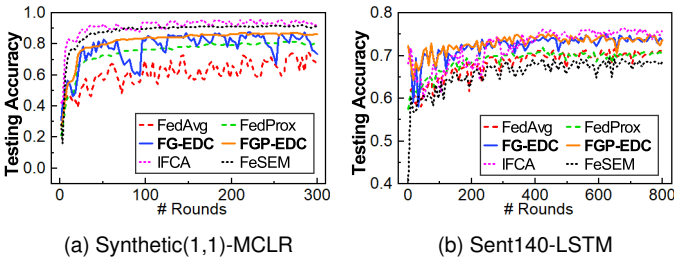

Fig. 4. Classification accuracy evaluated on Synthetic(1,1) ($m = 5$) and Sent140 ($m = 5$).
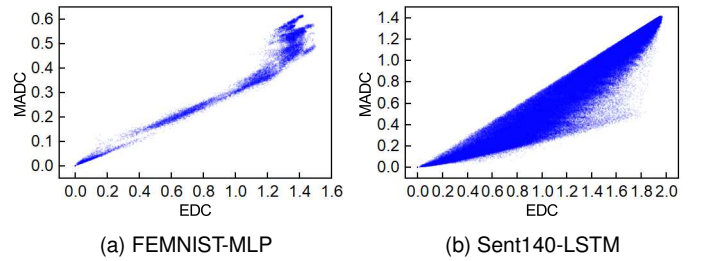

Fig. 5. The mapping from the MADC to the EDC, the pre-training scale $\alpha = 20$.

The experimental results in specific rounds are shown in Fig. 3 and Fig. 4. The accuracy and discrepancy curves illustrate that adding the proximal term can reduce the divergence caused by the heterogeneous data and make the training more stable. Although the training procedure of FedGrouProx is more stable than that of FedGroup, there

is not enough evidence to suggest that adding the proximal term is significantly helpful in improving accuracy. Compare to IFCA and FeSEM, the discrepancy of FedGroup remains at a low level in the initial round, which is attributed to the effective group cold start strategy. The FedAvg and FedProx achieve low discrepancy by maintaining the consensus global

model, but this will degrade the accuracy, which is supported by the obtained results. FedGroup also shows a significant improvement in convergence speed compared to FedAvg, which is helpful to reduce the communication consumption of the FL systems. By the way, the rescheduling methods of IFCA and FeSEM involve additional overheads in each round, such as broadcasting group models and calculating loss.

To explore the potential of semi-pluralistic architecture, we evaluate FedGroup under difference η_g. The details are not presented due to the pages-limit. Our experiments in MNIST-MCLR and Sent140-LSTM show that the inter-group aggregation mechanism can slightly improve the convergence rate of model training. Unfortunately, the convergence rates of other experimental sets do not improve as expected, which can be interpreted as each group in FedGroup is highly specialized and has few common representations. So, we will explore using a gate network to combine group models in the future.

Finally, in Fig. 5, we study the difference between MADC and our decomposed measure EDC as clustering distance. The experiments show that the mapping from the MADC to EDC is approximately linear, which means that EDC can be regarded as a low-dimensional linear approximation of MADC. In addition, calculating EDC only needs to pre-train αm clients, which is more computationally efficient and implementable than calculating pairwise proximity.

V. CONCLUSION

In this work, we have presented the decomposed data-driven measure-based clustered federated learning frameworks FedGroup, which can improve the model performance of federated training by efficient clustering and cold start strategies. We evaluated the proposed frameworks on four open datasets and shown the superiority of FedGroup compared to FedAvg, FedProx, FeSEM. FedGroup significantly improved $+26.9\%$ top-1 test accuracy on FEMNIST compared to FedSEM. 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 data-driven measure EDC, shown that it is an appropriate linear approximation of MADC. Our evaluation results on 6 models shown that FedGroup achieved higher classification accuracy compared to FedAvg, FedProx, and two random strategies RCC (except Sent140) and RAC.

REFERENCES

[1] J. Konečnỳ, 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," in *Advances in Neural Information Processing Systems*, vol. 33. Curran Associates, Inc., 2020, pp. 19 586–19 597.

[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] Y. Zhang, M. Duan, D. Liu, L. Li, A. Ren, X. Chen, Y. Tan, and C. Wang, "CSAFL: A clustered semi-asynchronous federated learning framework," *arXiv preprint arXiv:2104.08184*, 2021.

[19] S. Sarkar and A. K. Ghosh, "On perfect clustering of high dimension, low sample size data," *IEEE transactions on pattern analysis and machine intelligence (TPAMI)*, vol. 42, no. 9, pp. 2257–2272, 2019.

[20] 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.

[21] 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.

[22] 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.

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

[24] 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.

[25] L. Li, M. Duan, D. Liu, Y. Zhang, A. Ren, X. Chen, Y. Tan, and C. Wang, "FedSAE: A novel self-adaptive federated learning framework in heterogeneous systems," *arXiv preprint arXiv:2104.07515*, 2021.

[26] 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.

[27] M. Radovanović, A. Nanopoulos, and M. Ivanović, "On the existence of obstinate results in vector space models," in *Proceedings of the 33rd international ACM SIGIR conference on Research and development in information retrieval*, 2010, pp. 186–193.

[28] 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.

[29] 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.

[30] V. Smith, C.-K. Chiang, M. Sanjabi, and A. Talwalkar, "Federated multi-task learning," in *Proceedings of the 31st International Conference on Neural Information Processing Systems (NeurIPS)*, 2017, pp. 4427–4437.

[31] K. Bonawitz, V. Ivanov, B. Kreuter, A. Marcedone, H. B. McMahan, S. Patel, D. Ramage, A. Segal, and K. Seth, "Practical secure aggregation for privacy-preserving machine learning," in *Proceedings of the 2017 ACM SIGSAC Conference on Computer and Communications Security (CCS)*. ACM, 2017, pp. 1175–1191.

[32] M. Mohri, G. Sivek, and A. T. Suresh, "Agnostic federated learning," in *Proceedings of the 36th International Conference on Machine Learning (ICML)*, 2019, pp. 4615–4625.

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

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

[35] 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.

[36] 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.

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