





Towards Hierarchical Clustered Federated Learning With Model Stability on Mobile Devices

Biyao Gong , Tianzhang Xing , Zhidan Liu , *Member, IEEE*, Wei Xi , *Member, IEEE*, and Xiaojiang Chen 

Abstract—Clustered federated learning (CFL) has proved to be an effective way to alleviate the non-IID (not independently and identically distributed) data challenge, which severely restricts the wider application of federated learning. However, existing approaches either lack adaptability, i.e., they require an additional number of clusters as a guide when clustering, or lack effectiveness in terms of communication. In this paper, we explore the differences in the ability of different layers in a model to represent non-IID data, and propose a hierarchical CFL approach, named *HiCFL*, which considers both adaptivity and communication efficiency. The improvement of communication efficiency is due to our proposed novel concept of model stability, which characterizes the variation of model weights during training. Based on model stability, *HiCFL* can find the proper time to bi-partition the clusters of mobile devices in a hierarchical manner more quickly. We conduct extensive experiments based on popular datasets with various non-IID data settings. The results show that *HiCFL* achieves excellent performance effectiveness and efficiency. Compared to state-of-the-art approaches, *HiCFL* can improve the model accuracy by 2.0%~9.0%, while reducing the communication overheads by 27.3%~80.6%.

Index Terms—Federated learning, clustered federated learning, communication efficiency, hierarchical clustering, model stability.

I. INTRODUCTION

TO LEVERAGE the massive amount of data generated on mobile devices, e.g., smartphones, to train machine learning models while protecting data privacy, federated learning (FL) has emerged as a promising distributed machine learning paradigm [1]. In the FL setting, mobile devices as the clients

only need to upload local models or just the model updates, rather than the raw data, to a central server to train a globally shared model. Recently, FL has been applied to a wide range of domains that raise imperative concerns on data privacy, such as recommender systems [2], [3], finance [4], [5], health care [6], [7], [8], and vehicle networks [9], [10].

A typical FL approach, e.g., the most famous *FedAvg* [1], aggregates model weights from all mobile clients iteratively until converging to a stationary model. However, such a single model learning paradigm suffers poor performance in practical applications, e.g., image recognition [11], due to unbalanced and non-IID (not independently and identically distributed) data distributions among mobile clients. Heterogeneous data commonly exists in many application scenarios because the users, who produce the data, may have different physical environments and diverse usage habits [11], [12], [13]. The non-IID data generally requires more communication overheads between the server and mobile clients to reach the converged model, which largely increases the burden on mobile clients [14], [15], while the accuracy of the learned model may be severely degraded [16], [17]. As a result, the non-IID data issue greatly hinders the uses of FL in real-world applications.

In the literature, many efforts have been devoted to mitigating the impact of non-IID data. Previous works primarily train one single global model from some well selected participating clients (e.g., [18], [19]) or even partial high-quality data of each client (e.g., [16], [20], [21]). These approaches, however, are still ineffective because a single model cannot well reflect the underlying data distributions, resulting in poor model accuracy. Recently, a novel framework, as known as *clustered federation learning* (CFL) [22], is proposed to attack the non-IID challenge. The key idea of CFL is that mobile clients can be classified into different clusters according to the similarity of their data distributions, and clients belonging to the same cluster collaborate to train a shared model. As a result, multiple global models exist and the impact of data heterogeneity will be largely reduced. CFL has inspired a number of follow-up works [22], [23], [24], [25], [26], [27]. Most existing CFL approaches [23], [24], [25], [26] require inputting the number κ of clusters in advance, while it is difficult or even impossible to determine the optimal κ with no prior knowledge of clients' data distributions in the FL setting. A few works [22] implement CFL without pre-specifying κ by iteratively separating clients into clusters. However, they separate a cluster only when all local models have been converged, leading to huge communication costs, since convergence of all

Manuscript received 20 January 2023; revised 17 September 2023; accepted 3 November 2023. Date of publication 14 November 2023; date of current version 7 May 2024. This work was supported in part by China NSFC under Grants 62172284 and 61972316, and in part by Guangdong Basic and Applied Basic Research Foundation under Grant 2022A1515010155. Recommended for acceptance by M. Zhang. (*Corresponding authors: Tianzhang Xing; Zhidan Liu.*)

Biyao Gong is with the School of Information Science and Technology, Northwest University, Xi'an, Shaanxi 710069, China (e-mail: gby@stumail.nwu.edu.cn).

Tianzhang Xing and Xiaojiang Chen are with the School of Information Science and Technology, Shaanxi International Joint Research Centre for the Battery-Free Internet of Things, Northwest University, Xi'an, Shaanxi 710069, China (e-mail: txz@nwu.edu.cn; xjchen@nwu.edu.cn).

Zhidan Liu is with the College of Computer Science and Software Engineering, Shenzhen University, Shenzhen 518060, China (e-mail: liuzhidan@szu.edu.cn).

Wei Xi is with the School of Computer Science and Technology, Xi'an Jiaotong University, Xi'an, Shaanxi 710049, China (e-mail: weixi.cs@gmail.com).

This article has supplementary downloadable material available at <https://doi.org/10.1109/TMC.2023.3332637>, provided by the authors.

Digital Object Identifier 10.1109/TMC.2023.3332637

local models requires a large number of communication rounds between the server and clients. In addition, when there are a large number of clients involved in FL, these CFL approaches will also incur non-negligible computation costs for client clustering [28].

In this paper, we present *HiCFL*, a Hierarchical Clustered Federated Learning approach, to advance the existing CFL approaches. Without pre-specifying the cluster number κ , *HiCFL* can still effectively and efficiently classify all clients into a suitable number of clusters. The key idea of *HiCFL* derives from one important observation. Existing literatures have investigated the differences between the different layers in the model [29], [30], [31], e.g., the higher level weights are more task-related compared to the lower level weights. We further experimentally observe that different layers of a local model show diverse capabilities for describing the data distribution in FL, and the cosine similarity among layer-wise model updates of different clients demonstrates a clear clustering effect. In particular, we find that some layers may possess such a capability much earlier and meanwhile stronger than others. Inspired by this observation, we propose the concept of *model stability* to measure the convergence state for the weights of each model layer, and exploit model stability to guide the process of client clustering. Specifically, *HiCFL* continuously calculates the model stability for each client using a sliding window. Once a stable state is reached for an identical layer in all local models, *HiCFL* bi-partitions clients of a cluster into two sub-clusters according to the similarity values of the layers reaching stability. *HiCFL* separates clients into proper clusters in a hierarchical manner, and terminates the clustering until no cluster separation. As a result, *HiCFL* can intelligently group clients, even without the input of cluster number κ . Moreover, *HiCFL* incorporates a model weight selection mechanism to retain only updates of unstable model weights for the model stability calculations.

The main contributions of this paper are as follows:

- We observe that different layers of a client's model have diverse capabilities on describing the underlying data distribution, and for the first time propose a novel indicator named model stability to measure the state of model updates at the layer level.
- We present *HiCFL* that exploits model stability to effectively guide the client clustering in FL. Specifically, we propose a reference-based bi-partitioning strategy to separate mobile clients into proper sub-clusters in a hierarchical manner, and devise a model weight selection mechanism to further optimize the calculations of model stability.
- We conduct extensive experiments with three popular datasets under various non-IID data settings. The experimental results demonstrate the effectiveness and efficiency of *HiCFL*. Compared to state-of-the-art approaches, *HiCFL* can reduce communication costs by 27.3% ~ 80.6%, while improving model accuracy by 2.0% ~ 9.0%.

The rest of this paper is organized as follows. Section II presents the preliminary. Section III introduces the concept of model stability. *HiCFL* is elaborated and evaluated in

Sections IV and V, respectively. The related works are reviewed in Section VI. Section VII concludes this paper.

II. PRELIMINARIES

A. Federated Learning

Mobile devices generate massive amounts of data at the edge of the network, and centralizing this data in a single server to train deep learning models is often impractical due to data privacy concerns. Federated learning (FL) enables distributed mobile devices to collaboratively train a shared model without exposing their raw data. *FedAvg* [1] is so far the most commonly used algorithm for implementing the idea of FL. Specifically, *FedAvg* trains a globally shared model through a plenty of rounds of communication between the central server and distributed clients. At the beginning of each communication round, the server distributes the current global model to the clients, who will proceed to train this model with their own data. Once the local training is completed, the model updates are synchronously uploaded to the server for aggregating a new global model. These operations are repeated until the global model has been converged.

We take the multi-classification problem as a vehicle to explain *FedAvg* in detail. In this problem, the feature space is \mathcal{X} and the label space is \mathcal{Y} . Without loss of generality, we assume that there are C classes in total, i.e., $|\mathcal{Y}| = C$. Let $(x, y | x \in \mathcal{X}, y \in \mathcal{Y})$ denotes a labeled sample, and $f(\cdot)$ is defined as the prediction function. For the multi-classification task, loss function $\mathcal{L}(\cdot)$ is usually defined as the cross entropy loss. Thus, the learning objective is

$$\min_{\omega} \left\{ \mathcal{L}(\omega) \triangleq - \sum_{a=1}^C p(y=a) \mathbb{E}_{x|y=a} [\log(f_a(x, \omega))] \right\}, \quad (1)$$

where ω is the weight vector, and f_a represents the probability of predicting sample x as the class a .

In the FL setting, assume that there are m clients, denoted by $\mathbb{C} = \{c_1, c_2, \dots, c_m\}$, and a central server. Each client c_i locally stores n_i samples that obey a data distribution of P_{c_i} , i.e., client c_i 's local dataset $\mathcal{D}_{c_i} \sim P_{c_i}$. The objective of *FedAvg* is thus defined as

$$\min_{\omega} \left\{ F(\omega) \triangleq \sum_{i=1}^m \frac{n_i}{N} F_{c_i}(\omega) \right\}, \quad (2)$$

where $F_{c_i}(\omega)$ denotes the loss of client c_i , i.e., $\mathcal{L}_{c_i}(\omega)$, and $\sum_{i=1}^m n_i = N$ is the total number of samples from m clients.

In each communication round t of *FedAvg*, the clients download current global model ω_{t-1} from the server and conduct stochastic gradient descent (SGD) [32] locally

$$\begin{aligned} \omega_t^{(c_i)} &= \omega_{t-1} - \eta \nabla \mathcal{L}_{c_i}(\omega_{t-1}) \\ &= \omega_{t-1} - \eta \sum_{a=1}^C p(y=a) \nabla_{\omega} \mathbb{E}_{x|y=a} [\log(f_a(x, \omega_{t-1}))], \end{aligned} \quad (3)$$

where η is the learning rate when client c_i performs SGD locally. After each client c_i has obtained $\omega_t^{(c_i)}$, the clients synchronously upload model updates, i.e., the difference between local model weights $\Delta_t^{(c_i)} \triangleq \omega_t^{(c_i)} - \omega_{t-1}^{(c_i)}$, to the server. Once the server receives all model updates, it performs aggregation to update the global model

$$\Delta_t = \sum_{i=1}^m \frac{n_i \Delta_t^{(c_i)}}{N},$$

$$\omega_t \leftarrow \omega_{t-1} + \Delta_t. \quad (4)$$

FedAvg repeats above operations until the training process is converged (e.g., Δ_t is sufficiently small), and the final global model will be transmitted to all clients for use.

B. FL Challenges on Mobile Devices

In reality, the data of different users often exhibit a high level of heterogeneity, resulting in non-IID training data for FL. As an example, users may active in different physical environments and own diverse biological features, thus their data for human activity recognition will exhibit high heterogeneity, which leads to poor model accuracy [11]. The shared model learned via *FedAvg* has demonstrated good performance when the training data from different clients are IID. However, more and more studies report that *FedAvg* may be unstable or even ineffective when the data of different clients is non-IID, i.e., $P_{c_i} \not\sim P_{c_j}$ ($i \neq j$), even with severe model accuracy drops as large as 51% [16].

The objective of each client c_i 's local execution of SGD is to minimize the empirical loss on local dataset \mathcal{D}_{c_i} , and thus different data distributions of clients make their objectives diverge. As a result, local objectives cannot reliably approximate the global objective from the perspective of expectation, i.e.,

$$\mathbb{E} \left[F_{c_i} \left(\omega^{(c_i)} \right) \right] \neq F(\omega). \quad (5)$$

If we keep on training local models on such non-IID data, the divergences between local model weights $\omega^{(c_i)}$ accumulate and the conflicts at model aggregation on the server will increase accordingly, resulting in degraded training performance or even failing to derive a converged model.

Therefore, many research efforts [16], [18], [20], [21], [33] have been made to mitigate the negative effects of non-IID data for FL. For example, Zhao et al. [16] propose to use a globally shared dataset among clients to reduce the differences between local models. FedProx [21] improves *FedAvg* by allowing model aggregation based on partial information. Li et al. [20] propose a data selection strategy at the sample level, which only utilizes high-quality training samples for model training. Moreover, Wang et al. [18] exploit reinforcement learning to select clients for each round of model training, so as to balance the biases caused by non-IID data. These works primarily train a single global model from some well selected participating clients or even partial high-quality data of each client. The trained single model, however, is usually inadequate to capture the heterogeneous data of all clients, and thus cannot achieve high accuracy in practical applications [11].

In addition to the challenge of non-IID data, mobile devices participating in FL encounter various obstacles, including limited computational power, communication capacity, storage capacity, and device heterogeneity. Among these challenges, communication resource limitations represent the primary barrier preventing mobile devices from participating effectively in FL, since communication in the network can be many orders of magnitude slower than local computation [34]. Therefore, reducing communication costs is a crucial aspect of realizing FL on mobile devices.

C. Clustered Federated Learning

Recently, a promising framework named *clustered federated learning* (CFL) [22] has been proposed to attack the non-IID challenge. In general, CFL divides all clients into a number κ of clusters, i.e., $\mathbb{G} = \{G_1, G_2, \dots, G_\kappa\}$, according to their data distributions, and trains a global model for clients of each cluster, respectively. To indirectly measure the data distribution similarity among clients, existing CFL works calculate the model similarity between clients based on their model updates or gradients. Specifically, each client sends its model updates or gradients to the server, and the server computes the cosine similarity of model updates or gradients for any two clients. Based on the model similarity results, the server divides the clients into clusters. It is worthy to noting that CFL groups the clients according to their model similarity, while the data among clients of the same cluster may not be strict IID. However, the impact of non-IID data for FL-based model training is largely reduced.

Compared to training one single model for all clients with heterogeneous data, multiple models may be easily negotiated once the clients with similar data distributions have been correctly distinguished. Such a strategy is equivalent to decomposing the global objective of the former FL problem into multiple (i.e., κ) sub-objectives, such that these sub-objectives can be well approximated by the local objectives of clients from the perspective of expectation

$$\mathbb{E} \left[F_{c_i}^k \left(\omega^{(c_i)} \right) \right] = F^k(\omega_k), \quad k = 1, 2, \dots, \kappa, \quad (6)$$

where F^k is the global sub-objective for cluster G_k , $F_{c_i}^k$ is the objective of client c_i in G_k , and ω_k is the model of G_k .

Next we will briefly analyze the feasibility of clustering clients. For simplicity, assume that all m clients are grouped into two clusters, i.e., G_1 and G_2 , and their local data are sampled from two different distributions P_{G_1} and P_{G_2} , i.e., $\mathcal{D}_{G_1} \sim P_{G_1}$ and $\mathcal{D}_{G_2} \sim P_{G_2}$. Each client c_i (from either G_1 or G_2) has an empirical risk loss function defined as

$$F_{c_i}^k(\omega^{c_i}) \triangleq \frac{1}{|\mathcal{D}_{G_k}^i|} \sum_{(x,y) \sim \mathcal{D}_{G_k}} \mathcal{L}(y; x, \omega^{c_i}), \quad k = 1 \text{ or } 2. \quad (7)$$

Then (2) can be rewritten as

$$F(\omega) \triangleq \sum_{i=1}^m \frac{n_i}{N} F_{c_i}(\omega)$$

$$= \sum_{a=1}^{|G_1|} \frac{n_a}{N_{G_1}} F_{c_a}^1(\omega) + \sum_{b=1}^{|G_2|} \frac{n_b}{N_{G_2}} F_{c_b}^2(\omega), \quad (8)$$

where $N_{G_1} = \sum_{i=1}^{|G_1|} n_i$ and $N_{G_2} = \sum_{i=1}^{|G_2|} n_i$. If the FL training converges to a stationary point ω^* of FL's objective, i.e., $F(\omega^*) = \min(F(\omega))$, then we have

$$0 = \nabla F(\omega^*) = \sum_{a=1}^{|G_1|} \frac{n_a}{N_{G_1}} \nabla F_{c_a}^1(\omega^*) + \sum_{b=1}^{|G_2|} \frac{n_b}{N_{G_2}} \nabla F_{c_b}^2(\omega^*). \quad (9)$$

To simplify above equation, we assume that the empirical losses of clients in the same cluster to be the same, i.e., $F_{G_1}(\omega) = F_{c_a \in [G_1]}^1(\omega)$ and $F_{G_2}(\omega) = F_{c_b \in [G_2]}^2(\omega)$. Let $\rho_1 = \sum_{a=1}^{|G_1|} \frac{n_a}{N_{G_1}}$ and $\rho_2 = \sum_{b=1}^{|G_2|} \frac{n_b}{N_{G_2}}$. To satisfy (9), there exists two such cases, one is

$$\nabla F_{G_1}(\omega^*) = \nabla F_{G_2}(\omega^*) = 0, \quad (10)$$

while the other is

$$\nabla F_{G_1}(\omega^*) = -\frac{\rho_2}{\rho_1} \nabla F_{G_2}(\omega^*) \neq 0. \quad (11)$$

For the latter case, in practice, we often have no control over ρ_1 and ρ_2 . For the former case, it usually does not hold because the data distributions P_{G_1} and P_{G_2} are different. As a result, we cannot find one unified model ω^* for all clients. In order to minimize the objective of FL, clustering clients with similar data distributions is thus a necessary and effective way to alleviate the impact of non-IID data.

There exist several remarkable works that have practiced the idea of CFL [22], [23], [24], [25], [26], and they have demonstrated that CFL is effective on alleviating the impact of non-IID data. In general, existing CFL approaches can be classified into two categories, according to whether the number κ of clusters needs to be specified in advance. For example, these works in [23], [24], [25], [26] require knowing κ before clustering clients. However, it is difficult or even impossible to determine the optimal κ with no prior knowledge of the clients' data distributions. Although Sattler et al. [22] implement CFL through multiple rounds of bipartite separation, without inputting κ , their approach separates a cluster only when the local models of all clients have been converged, resulting in great latency and huge communication costs.

Therefore, we are motivated to further advance the CFL idea by devising an approach that can effectively group all clients into proper clusters without specifying κ in a communication-efficient manner. However, it is non-trivial to achieve this goal due to the following major challenges:

- *Without knowledge about clients' data:* FL is initially proposed for privacy-preserved machine learning, and thus it is prohibited for the server to access to clients' raw data. As a result, we should implicitly measure the similarity of clients' data distributions, without compromising the data privacy.
- *Hard to determine the optimal κ :* Previous studies show that the number κ of clusters is highly relative to the

final model accuracy [24], [25], while without knowledge about the exact data distributions of all clients, it is quite difficult to determine the optimal κ , which can best reflect underlying data distributions and lead to accurate models.

- *Huge communication overheads:* FL normally involves many communications between the server and clients to train a global model, while CFL may need more extra communication costs to negotiate the clustering of clients. In particular for mobile clients, communications will not only prolong the whole training process, but also largely consume precious battery energy and network traffics. Thus, an efficient communication mechanism is desired for the new CFL approach.

III. MODEL STABILITY

In this section, we experimentally study the characteristics of FL-based model training over non-IID data, and introduce the concept of model stability to motivate our design.

A. FL-Based Model Training Over Non-IID Data

To better understand the impact of non-IID data on FL, we conduct a simple motivating experiment by training a CNN model for $m = 10$ clients, i.e., $\mathbb{C} = \{c_1, c_2, \dots, c_{10}\}$, with FL over the CIFAR10 dataset [35]. Specifically, the CNN model consists of two convolutional layers (denoted by *Conv1* and *Conv2*) and a fully connected layer (denoted by *FC*) (Please see details about the datasets and models in Section V-A). To simulate the non-IID data distributions among clients, we artificially divide all ten clients into two clusters, where client $c_1 \sim c_5$ form a cluster and the rest clients form the other cluster. We use two different methods to generate non-IID data. The first one assigns samples with the same feature space but different label space for the clients, while the second one assigns samples with the same label space but different feature space for the clients. In addition, the local data for clients of each cluster are sampled from the CIFAR10 dataset according to the Dirichlet distribution with its scaling parameter $\alpha = 1$. We execute *FedAvg* [1] to learn the global model, and constantly observe the local model updating on each client. For any two clients, we use the *cosine similarity* to measure the similarity between their layer-wise model updates.

For the first non-IID data setting, Fig. 1 plots the layer-wise similarity for all clients after round 1 and round 10, where the darker is the more similar between the model updates of clients would be. After the first round, the similarity distributions for layer *Conv1* and *Conv2* are random and cannot well reflect the cluster-relation between clients, while we see the clear gathering phenomenon in Fig. 1(c), which implies that model updates of layer *FC* are capable of distinguishing clients with similar data distributions. After the tenth round, we find that model updates of each layer have better capability on distinguishing whether any two clients own the similar data distributions or not. We see two distinct clusters from Fig. 1(e) and (f).

For the second non-IID data setting, the results as shown in Fig. 2 are different from Fig. 1. We see that after the first round, only layer *Conv2* can show the gathering phenomenon as shown in Fig. 2(b). After the tenth round, two convolutional layers,

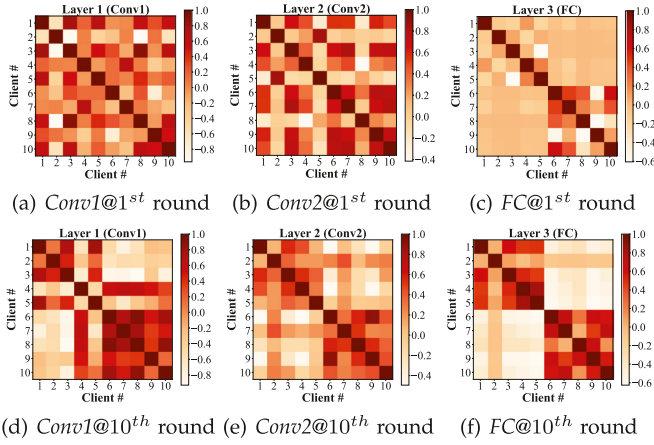


Fig. 1. Cosine similarity between model updates of different model layers for all clients at the 1st and 10th communication rounds in the first non-IID data setting.

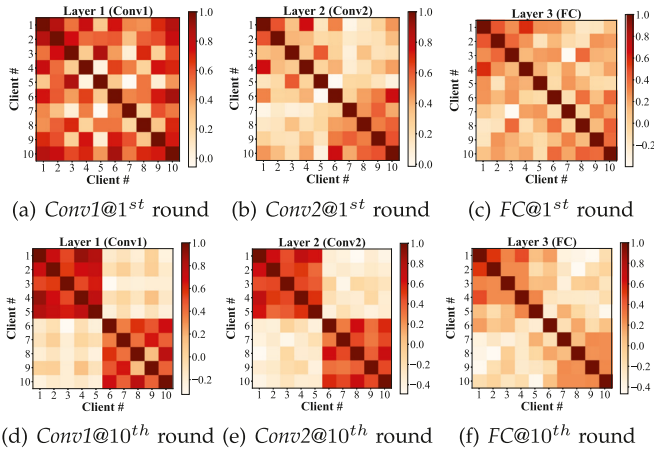


Fig. 2. Cosine similarity between model updates of different model layers for all clients at the 1st and 10th communication rounds in the second non-IID data setting.

i.e., *Conv1* and *Conv2*, can show the gathering phenomenon as shown in Fig. 2(d) and (e), while layer *FC* still cannot.

In summary, we have the following two key observations based on the above experiments.

- *Observation 1: different layers in a model have varied capability on describing the underlying data distribution.* We find that the *FC* layer can well distinguish clients of different clusters when the clients' data differ in the label space, while the convolutional layers are better at capturing data distribution difference in the feature space. This is because from a functional point of view, the convolutional layer is used to extract local features of the data, and its output is a feature map containing lots of local features. While the *FC* layer aims to globally perceive and transform the feature map to the label space for classifying the samples.
- *Observation 2: Each layer's capability on describing the underlying data distribution becomes stronger along with more rounds.* With more rounds of FL-based training, a

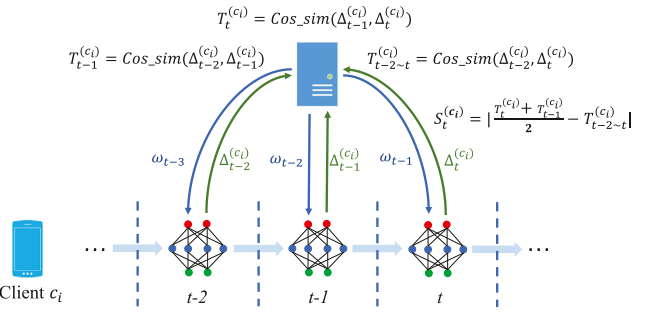


Fig. 3. Illustration of the model stability computation process.

client's local model becomes more stable and the updates of each layer fit the underlying data better. By comparing Fig. 1(c) and (f) (or Fig. 2(a) and (d)), we see that the model similarity among clients in the early stage e.g., the first round, is not stable and the gathering phenomenon is unclear, while their model similarity becomes more stable and the clustering effect is more obvious in the tenth round.

Previous works [18], [22] demonstrate that the weights of a local model can indirectly reflect a client's data distribution, while we further observe that the model updates of different layers and at different training phases have the unequal capability on describing a client's data distribution. Therefore, we could calculate the model similarity between clients by exploiting only partial "valuable" model updates (e.g., the model updates of the *FC* layer as shown in Fig. 1(f)), rather than all model weights/updates and start client clustering at some proper time. These insights motivate us to devise a brand-new CFL solution to attack the non-IID challenge.

B. Model Stability

Existing works [22] mainly exploit model updates of clients' converged models to calculate their model similarity. In FL, it usually requires many communication rounds between the server and clients to make the model converge due to non-IID data. According to the experiment results in Figs. 1 and 2, we find that after certain rounds, although the models are not converged yet, model updates of some layers can be used to well distinguish clients. In fact, model convergence indicates the state in which the training process should be terminated, while the model updates may be in a stable status much earlier than the convergence. Therefore, we should have a better view of the model updating trends for accurate and steady client clustering earlier. To this end, we propose the concept of *model stability* that measures how much the local model changes over several consecutive communication rounds, so as to represent the dynamic state of a local model updating.

We explain model stability using Fig. 3 that illustrates three rounds of communication between client c_i and the server. As shown in the figure, there are three rounds of model training, which generates three consecutive model updates accordingly, i.e., $\Delta_{t-2}^{(c_i)}, \Delta_{t-1}^{(c_i)}, \Delta_t^{(c_i)}$, $t \geq 3$. For any two consecutive model

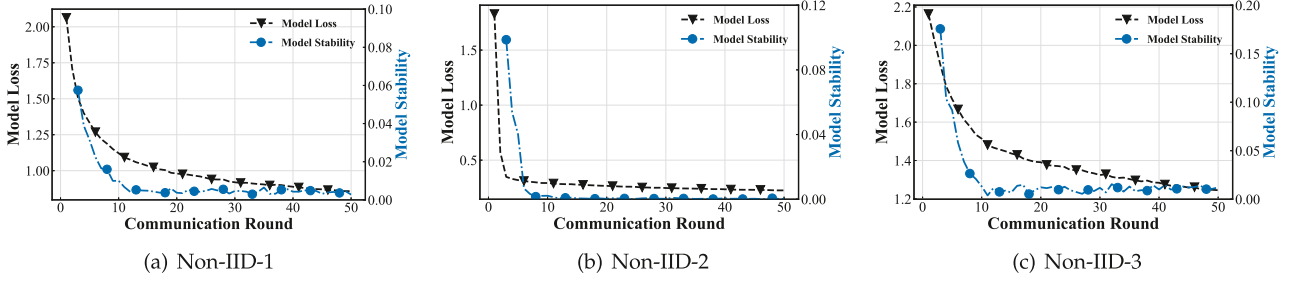


Fig. 4. Comparisons of model loss and model stability during the FL-based model training.

updates, we calculate model update trend $T_t^{(c_i)}$ at round t as

$$T_t^{(c_i)} = \frac{\langle \Delta_{t-1}^{(c_i)}, \Delta_t^{(c_i)} \rangle}{\|\Delta_{t-1}^{(c_i)}\| \cdot \|\Delta_t^{(c_i)}\|}. \quad (12)$$

In addition, for the first and last model updates, we calculate model update trend $T_{t-2 \sim t}^{(c_i)}$ at round t as

$$T_{t-2 \sim t}^{(c_i)} = \frac{\langle \Delta_{t-2}^{(c_i)}, \Delta_t^{(c_i)} \rangle}{\|\Delta_{t-2}^{(c_i)}\| \cdot \|\Delta_t^{(c_i)}\|}. \quad (13)$$

We thus define model stability for client c_i at round t as

$$S_t^{(c_i)} \triangleq \left| \frac{T_t^{(c_i)} + T_{t-1}^{(c_i)}}{2} - T_{t-2 \sim t}^{(c_i)} \right|. \quad (14)$$

The model update trends $T_t^{(c_i)}$ and $T_{t-2 \sim t}^{(c_i)}$ are calculated as the cosine similarity of two model updates. In general, a smaller $S_t^{(c_i)}$ indicates the higher stability of client c_i 's local model, which implies that the local model tends to be converged. It is worthy to noting that the computations of both $T_t^{(c_i)}$, $T_{t-2 \sim t}^{(c_i)}$ and $S_t^{(c_i)}$ are performed at the server side, and thus no additional computation cost will be introduced to the clients.

From (14), we see that model stability measures the changing trend of the client's local model during the training process. If model stability reaches a stable state, it means the model updates tend to be stable only with slight changes. In this case, if the model stability values of clients to be clustered become stable, their model updates are sufficiently stable and less influenced by other models, which can accurately demonstrate the cluster structure. Furthermore, compared to the converged state, model stability can quickly reflect whether the client's model is in a stable state or not. When the model stability is small, it implies that the local model becomes more stable and may be converged later. If the model stability is large, it means the client's model is still dramatically varying on its model weights.

To investigate the advantage of model stability, we conduct experiments to compare the changes of model loss and model stability when training the clients' local models with FL. In practice, when the model loss does not change or is smaller than a predefined threshold, we say the model has converged. In the experiments, we perform the clients' model training over the CIFAR10 dataset in the Non-IID-1, Non-IID-2 and Non-IID-3

data settings and record the average model loss and average model stability during the model training (Please find more details about the data settings in Section V-A.). The results in Fig. 4 show that model stability can reach a steady state much earlier than the model loss. For example, model stability becomes stable after 12 rounds, while model loss becomes stable (i.e., model convergence) around 40th rounds for all data settings.

IV. DESIGN OF *HiCFL*

In this section, we present the workflow of *HiCFL*, and then introduce the key designs involved in the client clustering.

A. Workflow

Similar to existing CFL methods [22], [24], we also consider a typical FL setting, where the clients are geographically distributed with good network connections and assume all clients will join the FL-based model training process to achieve better modeling results. Fig. 5 illustrates how *HiCFL* performs FL by separating clients into clusters based on their model stability. In general, *HiCFL* works like existing FL algorithms that use synchronous model aggregation strategy, and has the major steps as follows:

- Step 1:* All m eligible clients check in with the FL server, which initially treats all clients as one cluster and broadcasts current global model to all clients.
- Step 2:* Each client c_i performs local SGD training on the downloaded model using its own data. The training times may vary among clients, depending on the data amount on each client.
- Step 3:* Each client c_i calculates the model updates that are then transferred to the server.
- Step 4:* The server receives model updates from all clients, and then updates the global model for each cluster G_k by aggregating with model updates of clients belonging to cluster G_k . Later, the server broadcasts the newly updated global models to the clients of each cluster, respectively.
- Step 5:* For every W consecutive rounds of communication, e.g., at round t , the server calculates model stability $S_t^{(c_i)}$ for each client c_i using received model updates.
- Step 6:* If possible, the server will separate the clients of a cluster into two smaller clusters.

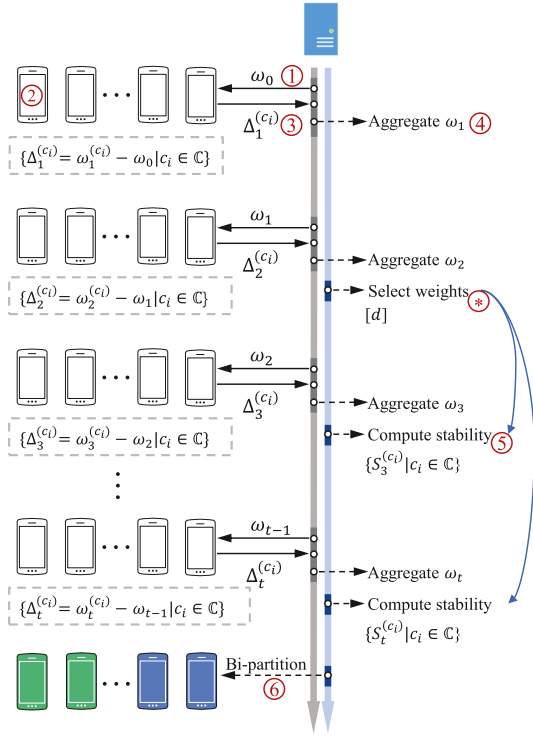


Fig. 5. Workflow of *HiCFL*.

Similar as *FedAvg*, each client in *HiCFL* needs to download a global model from the server, trains the model using its own data, and then transfers the model updates to the server for aggregating a new global model. Different from *FedAvg*, *HiCFL* will bi-partition a cluster G_k into two clusters according to the model stability of clients in G_k . Instead of training only one single global model for all clients, *HiCFL* will train one model for clients of each cluster respectively. *HiCFL* repeats Step 2–6 until no cluster will be separated. By constantly detecting the model stability of clients, *HiCFL* bi-partitions clusters and groups all clients into proper clusters in a hierarchical manner, which gets rid of the requirement of pre-specifying the number of clusters. Finally, κ clusters are formed, and clients of each cluster will collaboratively train their “private” global model.

Similar to existing FL works [1], [11], we adopt the synchronous learning strategy. However, *HiCFL* can speedup the whole FL training process. First, *HiCFL* exploits model stability to find the right timing of client clustering, which is earlier than previous CFL works and can reduce the number of communication rounds required for model similarity calculations. Second, compared to the conventional FL that involves many clients to train one single model, different client clusters train their respective shared models in an asynchronous manner that is quicker. Although the clients in each cluster still train their model in a synchronous manner, the number of clients in each cluster is much smaller, and thus the total model training time can be reduced.

Next we detail two key designs of *HiCFL*, namely *client clustering guided by model stability* and *weight selection for model stability calculation*.

B. Client Clustering Guided by Model Stability

At the beginning of FL, *HiCFL* treats all m clients as one cluster, and then hierarchically separates the cluster into more sub-clusters. Therefore, it is important to know *when HiCFL* should bi-partition a cluster and *how HiCFL* separates the clients of a cluster into two groups.

Timing of Bi-Partitioning a Cluster: Intuitively, when the local model of client c_i becomes converged, it means that the local data of c_i has been well exploited and the resultant model can well represent c_i 's underlying data. Thus previous works [22] use the weights of converged local models for client clustering. However, it may take many rounds for a local model to be converged, and thus greatly prolong the clustering process. As discussed in Section III-A, we observe that some layers of a local model may become stable (or converged) much earlier than the whole model. Therefore, *HiCFL* utilizes the model stability of clients to detect whether it is an opportune time to bi-partition a cluster. Specifically, *HiCFL* adopts a sliding window with size W to measure the model stability of each client's local model. The sliding window size W is the number of communication rounds between the clients and the server. Instead of relying on one single model stability value, we will calculate the average model stability within the sliding window, and thus can find a more accurate time to bi-partition the clients. In general, the larger W can reduce the randomness of model stability calculations, and thus derive more stable and accurate client clustering results.

Assuming the model to be trained has a total of L layers, at round t , if the following condition expressed in (15) is satisfied for any client $c_i \in G_k$, then cluster G_k can be bi-partitioned into two smaller clusters.

$$S_t^{(c_i, l)} < \epsilon, \quad \forall c_i \in G_k \ \& \ \exists l \in \{0, 1, \dots, L-1\}. \quad (15)$$

In (15), we can properly set ϵ given the learning rate η in local SGD, since the model stability of a client is only related with η as well, which is proved as the following.

For the l th layer of client c_i 's local model, its model stability at t th round is expressed as

$$\begin{aligned} S_t^{(c_i, l)} &= \left| \frac{T_t^{(c_i)} + T_{t-1}^{(c_i)}}{2} - T_{t-2 \sim t}^{(c_i)} \right| \\ &= \left| \frac{\langle \Delta_t^{(c_i, l)}, \Delta_{t-1}^{(c_i, l)} \rangle}{\|\Delta_t^{(c_i, l)}\| \cdot \|\Delta_{t-1}^{(c_i, l)}\|} + \frac{\langle \Delta_{t-1}^{(c_i, l)}, \Delta_{t-2}^{(c_i, l)} \rangle}{\|\Delta_{t-1}^{(c_i, l)}\| \cdot \|\Delta_{t-2}^{(c_i, l)}\|} \right. \\ &\quad \left. - \frac{\langle \Delta_t^{(c_i, l)}, \Delta_{t-2}^{(c_i, l)} \rangle}{\|\Delta_t^{(c_i, l)}\| \cdot \|\Delta_{t-2}^{(c_i, l)}\|} \right|. \end{aligned} \quad (16)$$

Interestingly, we find that the model stability of a client is only determined by the learning rate η when the other hyperparameters e.g., batchsize, and training optimization methods have been fixed. Please see Appendix A for a detailed discussion about this property, available online.

Separation of a Cluster: Once the time to separate a cluster G_k is determined, *HiCFL* needs to bi-partition the clients of G_k into

two groups. Given a cluster of clients, existing CFL approaches [22], [23], [25], [26] extensively calculate the similarity between any two clients' model weights, and separate them into two clusters according to their model similarity values. Such a strategy will introduce huge computations, since numerous clients may be involved in FL.

Therefore, we propose a reference based cluster bi-partitioning strategy. For a cluster G_k to be bi-partitioned, we select the client $c_r \in G_k$, which has the smallest model stability in cluster G_k , as the reference.

Assume that in cluster G_k , the l^{th} layer of all local models reaches stable first at the t^{th} round, i.e., satisfying the condition in (15), then the reference c_r of cluster G_k is selected as

$$c_r = \arg \min_{c_i \in G_k} \left(S_t^{(c_i, l)} \right), \quad (17)$$

Then, we calculate the cosine similarity of model updates between any other client $c_j \in G_k$ and the reference client c_r . The clients with positive similarity values are classified into one sub-cluster, while the rest are gathered into another sub-cluster. Specifically, the cosine similarity of model updates between c_j and c_r is calculated as

$$Sim(c_j, c_r) = \frac{\langle \Delta_t^{(c_r, l)}, \Delta_t^{(c_j, l)} \rangle}{\|\Delta_t^{(c_r, l)}\| \cdot \|\Delta_t^{(c_j, l)}\|}. \quad (18)$$

HiCFL will bi-partition a cluster G_k once the member clients of G_k meet the condition expressed in (15). With such a strategy, the server computes a $1 \times m$ similarity vector instead of an $m \times m$ similarity matrix in bi-partitioning, i.e., reducing the computational cost of similarity between models from $\mathcal{O}(m^2|\omega|^2)$ to $\mathcal{O}(m|\omega|^2)$, which greatly reduces the computation overheads. However, the cluster bi-partitioning process should be terminated when clients with similar data distributions have been properly grouped. Similar as [22], for all clients we calculate the average model update norm $\Delta_{ave} = \|\sum_{i=1}^m \frac{n_i}{N} \Delta^{(c_i)}\|$ and the maximum model update norm $\Delta_{max} = \max_{i=1, \dots, m} \|\Delta^{(c_i)}\|$. *HiCFL* will terminate the cluster bi-partitioning process if both Δ_{ave} and Δ_{max} are consistent.

C. Weight Selection for Model Stability Calculation

The server has to periodically calculate each client's model stability every W rounds for possibly bi-partitioning clients into smaller clusters. Due to the large number of parameters involved in machine learning (especially deep learning) models, however, the model stability calculations will introduce huge computation overheads.

In the *HiCFL* design, we thus adopt a simple yet effective weight selection mechanism to reduce the number of model updates used for model stability calculations. We observe that local model weights affect model stability with varying degrees. Specifically, the unstable model weights are relatively fixed between different FL training rounds. As shown in Fig. 6, we plot two consecutive rounds of weights' magnitude variations for a client's local model, which is a CNN model trained on

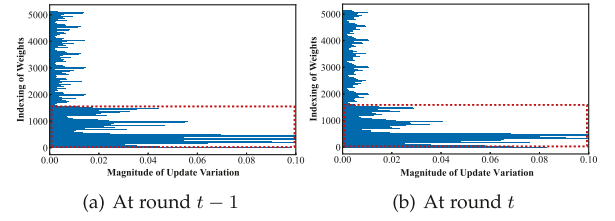


Fig. 6. Unstable model weights in each layer of the local model are relatively fixed between communication rounds.

FashionMNIST dataset under Non-IID-2 data setting, as detailed in Section V-A. By comparing the results in Fig. 6(a) and (b), we see that the model weights with indexes ranging from 0 to 1,200 are relatively unstable, while other weights seem to be more stable with small changes on their magnitudes. Intuitively, these stable model weights will have a negligible impact on the model stability calculation. Therefore, we can only use the model updates, whose model weights are unstable, for model stability calculation, so as to reduce the computation costs. For each layer l of client c_i 's local model, we select model weights as

$$\arg \min_{[d]} \left(\Delta_{new}^{(c_i, l, d)} - \Delta_{old}^{(c_i, l, d)} \right) > \xi, \quad (19)$$

where d is the index of a model weight in the l^{th} layer of c_i 's local model, and ξ is a pre-defined threshold. After obtaining the indexes of model weights, which satisfy the above condition, for each client c_i , the server will only use model updates indicated by indexes ds for calculating c_i 's model stability using (12) and (14). For each client c_i , the weight selection process is only performed at the FL initialization phase or after c_i 's cluster is bi-partitioned.

D. Theoretical Analysis

Computational Complexity Analysis: About the computational complexity of *HiCFL*, we have the following theorem.

Theorem 1: The computational complexity of *HiCFL* is $\mathcal{O}(mT|\omega'|^2 + mT + \kappa|G_*||\omega|^2)$.

Proof: *HiCFL* mainly involves two parts of calculations, namely model stability calculation for each client and model similarity calculation between clients for clustering.

For the first part, we select a small number of key parameters ω' to calculate model stability for each client. To calculate the model stability of a client at the given round, *HiCFL* needs to calculate model update trend between any two rounds (i.e., (12) and (13)), which involves $\mathcal{O}(|\omega'|^2)$ computations, and then computes the model stability (i.e., (14)) that involves only $\mathcal{O}(1)$ computation. Assume that the clients and the server communicate for T rounds, the computational complexity for model stability calculations is $\mathcal{O}(mT|\omega'|^2 + mT)$, where m is the number of clients in the FL training process.

For the second part, for each cluster G_k , we select a client with the least model stability as the reference, and then compute model similarity between the reference client and any other client in the cluster. Based on their model similarity values, we

bi-partition this cluster into two smaller clusters. Specifically, we calculate the cosine similarity of model updates between two clients as their model similarity, which involves $\mathcal{O}(|\omega|^2)$ computations, where $|\omega|$ is the number of a client's model parameters. The computational complexity for bi-partitioning cluster G_k is $\mathcal{O}(|G_k||\omega|^2)$. Assume there are κ clusters, and the total computational complexity for client clustering will be $\mathcal{O}(\kappa|G_*||\omega|^2)$, where $|G_*|$ is the average number of clients in a cluster.

Therefore, the overall computation complexity of *HiCFL* is $\mathcal{O}(mT|\omega'|^2 + mT + \kappa|G_k||\omega|^2)$. \square

Convergence Analysis: Rather than training one single global model for all clients, *HiCFL* divides all clients into different clusters and individually trains a shared model for the clients of each cluster. Therefore, we use cluster G_k as an example to analyze the convergence of *HiCFL*. We assume that there exists an optimal solution ω_k^* for cluster G_k , i.e., $F_{G_k}(\omega_k^*) = \min(F_{G_k}(\omega))$ and $\nabla F_{G_k}(\omega_k^*) = 0$.

Before analyzing the convergence bound of *HiCFL*, we first state some assumptions by referring to [17].

Assumption 1: The loss function F_{c_i} of client c_i is μ -strongly convex where $\mu \geq 0$, for $\forall x, y$

$$F_{c_i}(y) - F_{c_i}(x) \geq \langle \nabla F_{c_i}(x), y - x \rangle + \frac{\mu}{2} \|y - x\|^2.$$

Assumption 2: The loss function F_{c_i} of client c_i is L -smooth where $L > 0$, for $\forall x, y$

$$F_{c_i}(y) - F_{c_i}(x) \leq \langle \nabla F_{c_i}(x), y - x \rangle + \frac{L}{2} \|y - x\|^2.$$

Assumption 3: Assuming the variance of the stochastic gradient of each client $c_i \in G_k$ is bounded, i.e.,

$$\mathbb{E} \|\nabla F_{c_i}(y_t^{(c_i)}; x_t^{(c_i)}, \omega_t^{(c_i)}) - \nabla F_{c_i}(\omega_t^{(c_i)})\|^2 \leq \sigma^2.$$

Assumption 4: Assuming the expected squared norm of stochastic gradients of each client c_i is uniformly bounded, i.e.,

$$\mathbb{E} \|\nabla F_{c_i}(y_t^{(c_i)}; x_t^{(c_i)}, \omega_t^{(c_i)})\|^2 \leq Q^2,$$

where $(x_t^{(c_i)}, y_t^{(c_i)}) \in D_{c_i}$. Then we have the convergence bound of *HiCFL* as the following theorem.

Theorem 2: The convergence bound of *HiCFL* is

$$\begin{aligned} & \mathbb{E}[F_{G_k}(\omega_T) - F_{G_k}(\omega_k^*)] \\ & \leq \frac{\tau}{\gamma + t} \left(\frac{2B}{\mu} + \frac{\mu\gamma + \mu}{2} \mathbb{E} \|\omega_1 - \omega_k^*\|^2 \right), \end{aligned}$$

where $\tau = \frac{L}{\mu}$, $\gamma = \max(8\tau, E)$, and $B = \sum_{i=1}^{|G_k|} \left(\frac{n_i}{N_{D_k}} \sigma \right)^2 + 6L\Gamma_{G_k} + 8(E-1)^2Q^2$. In addition, E is the number of local iterations as defined in [1], Γ is used to measure the non-IID degree of the client's data as defined in [17], Γ_{G_k} denotes the non-IID degree of the data in cluster G_k , and $\Gamma_{G_k} = F_{G_k}(\omega_k^*) - \sum_{i=1}^{|G_k|} \frac{n_i}{N_{D_k}} F_{c_i}(\omega_k^*)$.

According to Theorem 2, we find that the convergence for cluster G_k 's model is bounded. We can accelerate the convergence speed by reducing the non-IID degree of cluster G_k

when compared to *FedAvg* [1] that trains only one single model, i.e., $\mathbb{E}[\Gamma_{G_k}] \leq \Gamma, k = 1, \dots, \kappa$. Therefore, by properly dividing clients with similar data distribution, i.e., reducing the non-IID degree of clients' data, into the same cluster, the model of each cluster derived by *HiCFL* can be converged quickly. Please see Appendix B for the detailed proof of Theorem 2, available online.

E. Discussions

In this subsection, we discuss some issues about *HiCFL*'s robustness and design choices.

Selection of an Improper Reference Model for Cluster Bi-Partitioning: In *HiCFL*, we select a client as the reference to separate the clients of a cluster into two sub-clusters. If a client, whose model temporarily gets stuck in a locally optimal solution, is selected as the reference, it will lead to incorrect client clustering results because the reference's model does not reflect the data distribution correctly. Such a case happens only when the following conditions are satisfied: (i) The model of a client c_i temporarily gets stuck in a locally optimal solution, and is mistakenly considered to have reached a stable state, i.e., its model stability is sufficiently small; (ii) The cluster G_k where client c_i belongs to should be bi-partitioned; and (iii) The client c_i is selected as the reference for bi-partitioning cluster G_k .

Due to the unique design of *HiCFL*, this case can be avoided with a high probability, due to the following reasons. First, we calculate the model stability of a client using the model updates of multiple consecutive communication rounds, and we calculate the average model stability within a large window W to determine the right timing of client clustering. Second, bi-partitioning one cluster requires that all local models of clients in the same cluster have reached a stable state. Therefore, the clustering mechanism of *HiCFL* can avoid the influence of a client's model that gets stuck in locally optimal solution as much as possible. Even though the special case happens, the cluster formed by the improper reference would be bi-partitioned later since clients of that cluster essentially have different data distribution. As a result, the influence can be further eliminated.

Re-Organizing Clusters versus Fine-Grained Clustering: Re-organizing the clusters after certain communication rounds seems to be an effective solution to utilize the information from clients with similar data distribution. However, this design choice has two major limitations. First, the model training among different clusters is asynchronous in the CFL setting, and thus how to re-organize the clusters that are in different training phases is challenging, which inevitably introduces extra communication and computation overheads. Second, from the perspective of model accuracy, we find that re-organizing clusters cannot derive higher model accuracy when compared to fine-grained clustering, i.e., bi-partitioning clients into smaller sub-clusters as our current design. We experimentally compare the two design choices in Section V-C. Therefore, we find that fine-grained clustering is a better design choice than re-organizing clusters.

TABLE I
STATISTICS OF THE CNN MODEL WEIGHTS FOR THREE DATASETS

Model	# of total weights	# of weights per layer
MNIST-CNN	33500	500 / 25000 / 8000
FmnistMNIST-CNN	18320	400 / 12800 / 5120
CIFAR10-CNN	6850	450 / 2400 / 4000

V. PERFORMANCE EVALUATION

A. Experimental Setup

We compare the performance of *HiCFL* with four baseline approaches by training CNN models on three popular publicly available datasets under different non-IID data settings.

Baseline Approaches: We compare *HiCFL* with the following four baselines: (1) *Centralized* collects raw data from all clients to the server for centrally training a global model, which works without FL. (2) *FedAvg* [1], the most commonly used approach in FL, coordinates all clients to collaboratively train a global shared model. (3) *MTCFL* (Multitask Clustered Federated Learning) [22], a state-of-the-art CFL approach, exploits multi-task learning to group clients into clusters by exploiting the geometric properties of clients' FL loss surface. (4) *IFCA* (Iterative Federated Clustering Algorithm) [24], another state-of-the-art CFL approach, alternatively estimates the cluster identities of clients and optimizes the weights of each cluster's global model in an iterative manner.

Noting that *Centralized* and *FedAvg* only train one model, while *MTCFL* and *IFCA* will train multiple, i.e., as the number κ of clusters, models. Besides, *MTCFL* groups clients without knowing κ , but *IFCA* needs to know κ in advance. Therefore, we set the optimal κ as an input for *IFCA* in the experiments.

Datasets and Models: We exploit FL to train various CNN models over different datasets for totally $m = 20$ clients. The datasets and models are described as follows.

- *MNIST* [36] contains 10 classes of handwritten digits, where the size of each sample is 28×28 . Models related to MNIST consist of two 5×5 convolutional layers and one fully connected layer. The first convolutional layer has 20 output channels and the second has 50, with each layer followed by a 2×2 max pooling layer.
- *FashionMNIST* [37] contains 10 classes of images, where the size of each sample is 28×28 . Models related to FashionMNIST consist of two 5×5 convolutional layers and one fully connected layer. The first convolutional layer has 16 output channels and the second has 32, with each layer followed by a 2×2 max pooling layer.
- *CIFAR10* [35] contains 10 classes of RGB images, where the size of each sample is 28×28 . Models related to CIFAR10 consist of two 5×5 convolutional layers and one fully connected layer. The first convolutional layer has 6 output channels and the second has 16, with each layer followed by a 2×2 max pooling layer.

Table I shows the statistics on the three models' weights. In addition, we chose Resnet18 [38] and VGG11 [39] to evaluate *HiCFL* in handling complex models.

To further evaluate the performance of different methods on more realistic datasets, we conduct experiments using the USC-HAD dataset [40]. USC-HAD is a dataset for well-defined low-level daily activity recognition, which contains IMU data for 12 simple activities (e.g., walking, running, jumping, and so on) from 14 volunteers. Each volunteer repeats each action five times, with each movement lasting about 24 seconds. The readings of the 3-axis accelerometer and 3-axis gyroscope of a MotionNode instrument placed around the volunteers' waists are collected under a sampling frequency of 100Hz. The HAR model contains three convolutional layers and two fully connected layers. The batch size is 16, and the learning rate is 0.001.

Non-IID Data Settings: Considering the joint distribution of data x and label y , i.e., $p(x, y) = p(y)p(x|y)$, which is jointly determined by $p(y)$ and $p(x|y)$. We thus generate non-IID data from two different aspects of label space and feature space. For $p(x)$ we set label distribution on clients to be different. For $p(x|y)$ we set label distribution on clients to be the same, while changing the original samples.

- *Non-IID in label space:* We set up two types of non-IID label distributions. (1) *Non-IID-1:* All clients are equally divided into four clusters, and the clients of each cluster are assigned with samples of the same labels. Specifically, label indexes for the four clusters are 0–3, 3–6, 4–9, and 0–9, respectively. (2) *Non-IID-2:* Similar to the data setting in [18], we use parameter β to indicate the non-IID level, e.g., $\beta = 0.7$ means that 70% of the samples in each client belong to the same label, while the remaining samples belong to other labels. All clients are grouped into four clusters as well.
- *Non-IID in feature space:* We let each client own the same labels, and then generate non-IID data by randomly selecting a fraction of clients and rotating their samples. For MNIST and FashionMNIST, we rotate half of the clients' samples by 180° , and for CIFAR10 we equally divide all clients into four clusters, and rotate the samples of each cluster with angles of 0° , 90° , 180° , and 270° , respectively. We denote this data setting as *Non-IID-3*.

To make non-IID data generations more realistic, each client samples data from a dataset following Dirichlet distribution with $\alpha = 1.0$, and the number of samples on each client is different. For different non-IID data settings, the learning rate η and batch size (BS) are set accordingly, i.e., 1) Non-IID-1: $\eta = 0.1$, $BS = 128$; 2) Non-IID-2: $\eta = 0.01$ for MNIST and FashionMNIST, $\eta = 0.001$ for CIFAR10, and $BS = 256$; 3) Non-IID-3: $\eta = 0.1$, $BS = 128$. Moreover, to generate natural data distribution, we assign samples to clients using the Latent Dirichlet allocation method [33], [41], which is commonly used for simulating non-IID data. Specifically, we set parameter α of Dirichlet distribution to 0.3 and 0.5, respectively, where a smaller value indicates a higher degree of non-IID. Besides, the batch size for VGG11 and Resnet18 is 16 and 32, respectively, and the learning rate is 0.01.

Implementation: We have implemented *HiCFL* using PyTorch and properly set the thresholds. Threshold ϵ for cluster bi-partitioning in (14) is set as 0.5η , and threshold ξ for weight selection in (19) is set as 0.05η . In addition, we carefully tune the parameters for the baselines to achieve their best performances.

TABLE II
COMPARISONS ON THE AVERAGE MODEL ACCURACY IN DIFFERENT NON-IID SETTINGS

I	Dataset	Centralized	FedAvg	MTCFL	IFCA	HiCFL
Non-IID-1	MNIST	97.01%	94.23%	96.62%	96.22%	97.46%
	F-MNIST	93.64%	84.94%	90.16%	91.97%	<u>93.34%</u>
	CIFAR10	57.16%	44.03%	<u>62.84%</u>	58.13%	64.56%
Non-IID-2	MNIST	99.81%	97.50%	97.77%	97.74%	<u>98.86%</u>
	F-MNIST	97.77%	90.67%	93.97%	94.81%	<u>95.97%</u>
	CIFAR10	74.65%	49.86%	71.79%	<u>75.42%</u>	76.54%
Non-IID-3	MNIST	<u>93.11%</u>	85.58%	91.06%	92.23%	93.41%
	F-MNIST	84.60%	85.76%	82.89%	84.44%	87.53%
	CIFAR10	38.44%	34.62%	<u>40.47%</u>	39.33%	40.92%

For each experiment setting, the best result is marked in bold, and the second best result is marked with underline. Besides, F-MNIST denotes the FashionMNIST dataset.

We adopt the *number of communication rounds* and *model accuracy* as the performance metrics to evaluate all approaches. Each experiment setting is repeated five times, and the average results are reported. In particular, the model accuracy shown in the experimental results represents the average of model accuracy of all clusters. All experiments are performed on a server equipped with an RTX 3090 GPU and an AMD 3800X CPU.

B. Performance Comparison

We run different approaches on the three datasets to train CNN models under various non-IID data settings for 50 rounds of communication, and then compare their average model accuracy results at the 50th round in Table II.

Overall, *HiCFL* performs much better than the baselines, with six best results out of the nine experiment settings. Even for the cases where *HiCFL* does not take the first place, it still achieves the second highest model accuracy, which is quite close to the best one. Both *Centralized* and *FedAvg* train only one single global model, and we see that *Centralized* outperforms *FedAvg* by deriving higher model accuracy in most cases. This is because *Centralized* gathers all clients' data for model training, however, it violates data privacy and may not be applicable in practice. Even so, *Centralized* cannot always achieve the best model accuracy. This is because the clients' data do not exactly match the FL's IID data assumption. As a result, one single model, even trained on the raw data in a centralized manner, is inadequate to well model the heterogeneous data that are too complex for a given target model structures, e.g., the CNN model in our experiments.

On the other hand, the CFL approaches, i.e., *MTCFL*, *IFCA*, and *HiCFL*, generally have higher model accuracy than *FedAvg*, which implies that multiple models are better than one single model on describing the non-IID data. Compared to the frequently used *FedAvg* and the two state-of-the-art *MTCFL* and *IFCA*, on average *HiCFL* improves the model accuracy by 9.0%, 2.3% and 2.0%, respectively.

Since Non-IID-3 data setting produces more heterogeneous data for clients in the CIFAR10 dataset, all approaches cannot well handle the non-IID data, and as a result obtain low accuracy, i.e., about 40%.

TABLE III
COMPARISONS ON THE NUMBER OF COMMUNICATION ROUNDS REQUIRED TO COMPLETE CLIENT CLUSTERING BETWEEN *MTCFL* AND *HiCFL*

I	Approach	MNIST	CIFAR10	FashionMNIST
Non-IID-1	<i>HiCFL</i>	9	14	13
	<i>MTCFL</i>	36	37	22
Non-IID-2	<i>HiCFL</i>	9	7	11
	<i>MTCFL</i>	24	36	33
Non-IID-3	<i>HiCFL</i>	12	16	15
	<i>MTCFL</i>	28	22	24

These results reflect the phenomenon that *HiCFL* improves Non-IID data due to differences in label space \mathcal{Y} more significantly than Non-IID data due to differences in feature space \mathcal{X} . This is because *HiCFL* bi-partitions the client clusters by the similarity between the client's model updates, which are more sensitive to data with different label spaces than data with different feature spaces.

Among all approaches, only *MTCFL* and *HiCFL* can group clients without knowing the cluster number κ . However, they pay different communication costs to achieve the clustering results. For each experiment setting, we record the time when client clustering is converged, i.e., the clustering results are the same as the final clusters in the 50th round. Table III shows that *HiCFL* requires much fewer rounds than *MTCFL*, i.e., reducing the communication costs by 27.3%~80.6%. Therefore, *HiCFL* is more communication-efficient than *MTCFL* on clustering clients, especially in the scenarios where mobile clients are constrained by energy and bandwidth.

C. Detailed Evaluation

In this subsection, we conduct some benchmark experiments to evaluate the performance of *HiCFL*.

Effect of Learning Rate η : In Section IV-B, we find that the threshold ϵ on determining the timing of cluster separation is only related to learning rate η . We conduct additional experiments to examine whether η really affects the performance of *HiCFL*. Fig. 7 shows the training curves of CNN models on different datasets under Non-IID-2 data setting when η takes different values. For the MNIST dataset, *HiCFL* works well and the model training can converge properly in different settings of η , as shown in Fig. 7(a). For the FashionMNIST dataset, as shown in Fig. 7(b), *HiCFL* can make the training converge only when $\eta = 0.01$, while when $\eta = 0.1$ the derived models have poor accuracy. The reason may be that a larger $\eta = 0.1$ impacts the training process and degrades the model accuracy. We observe similar results on the CIFAR10 dataset. As shown in Fig. 7(c), *HiCFL* works well when $\eta = 0.001$, but fails on a larger $\eta = 0.01$. However, when we increase the batch size from 256 to 512, *HiCFL* becomes feasible again. According to Fig. 7, we conclude that a proper η is important for *HiCFL* to achieve good clustering results, and sometimes we could increase the batch size of SGD to help *HiCFL* work well.

Effect of Different Non-IID Levels: We study the performance of *HiCFL* under different non-IID levels by varying parameter β in Non-IID-2 data setting. As shown in Fig. 8, *HiCFL* can always

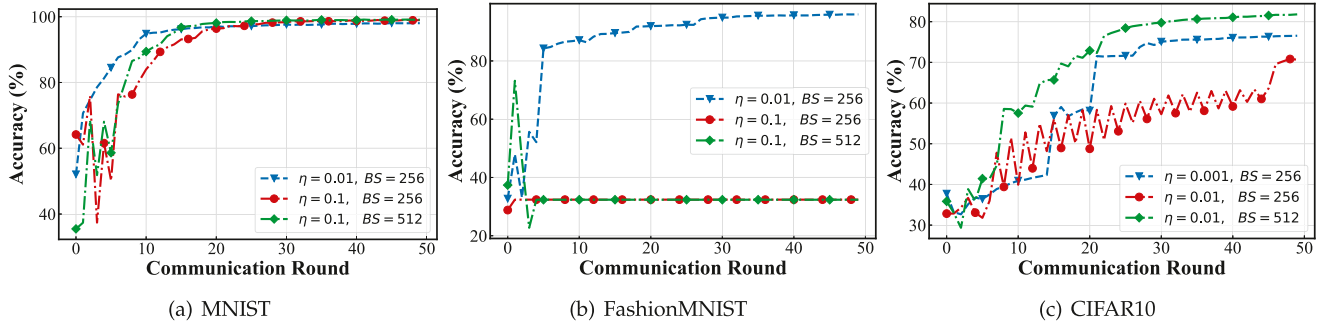


Fig. 7. Training curves of CNN models under the Non-IID-2 data setting with different η and batch size (BS) on different datasets.

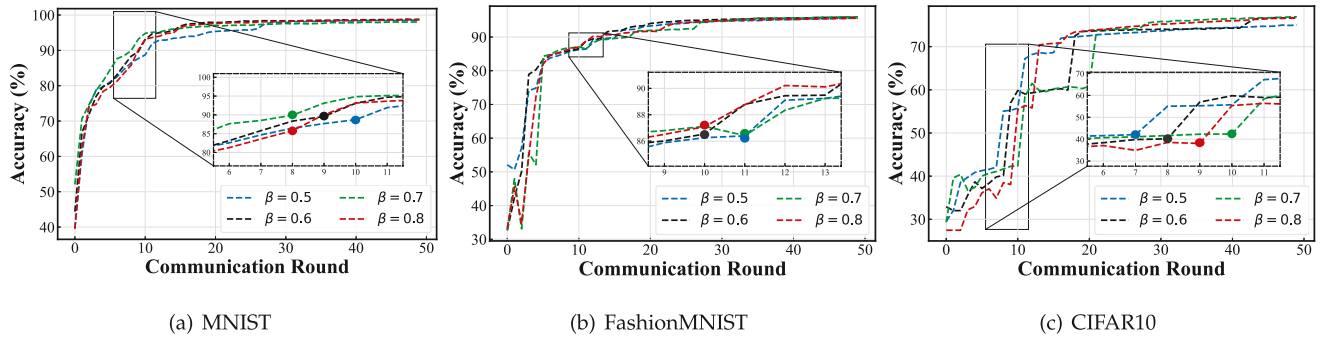


Fig. 8. Effect of non-IID levels on different datasets under the Non-IID-2 data setting.

properly complete client clustering within 10 communication rounds. Across different non-IID levels, *HiCFL* achieves stable clustering results with similar rounds, with the largest difference as 4 rounds that is observed on CIFAR10. Experiment results in Fig. 8 imply that *HiCFL* is robust to different non-IID data distributions.

Effect of Threshold ξ : We introduce threshold ξ in (19) to retain only important weight updates for computing clients' model stability. We thus perform experiments to study the impact of ξ on *HiCFL*, and plot the results in Fig. 9, where the values of different colored bars indicate the multiplicative relationship between ξ and learning rate η , e.g., “0.1” means $\xi = 0.1\eta$. In addition, “W.C.” means “without compression”, since all weight updates are retained.

Intuitively, a larger ξ leads to much fewer selected model weight updates, and thus saves more computation costs. However, *HiCFL* may not well bi-partition a cluster since the model stability of each client is only calculated based on partial model updates. Experimental results in Fig. 9 are in accordance with our analysis. Furthermore, Fig. 9 suggests that $\xi = 0.05\eta$ can make a good tradeoff between the number of communication rounds and the amount of used updates. This setting requires almost the same number of rounds as “W.C.”, while only using 36%~55% of model weights for the model stability calculations.

Validity of Model Stability: We experimentally compare the average accuracy of bi-partitioning client clusters before and after the timing when model stability becomes stable. In this experiment, we put 20 clients that should belong to two clusters into the same cluster. We perform *HiCFL* over CIFAR10 dataset

TABLE IV
AVERAGE MODEL ACCURACY OF BI-PARTITIONING CLUSTER BEFORE AND AFTER STABLE MODEL STABILITY

Setting	Before	After
Non-IID-1	87.14%	100.00%
Non-IID-2 $\beta = 0.5$	78.19%	98.89%
Non-IID-2 $\beta = 0.7$	76.00%	98.00%
Non-IID-3	56.67%	100.00%

under four different data settings, i.e., Non-IID-1, Non-IID-2 $\beta = 0.5$, Non-IID-2 $\beta = 0.7$ and Non-IID-3.

In each data setting, we train the model with FL for totally $T = 20$ communication rounds, and for each round we calculate the average model accuracy. We record the client clustering timing as t_s , and calculate the model accuracy results for the “before” period (from the first round to the $t_s - 1$ th rounds) and the “after” period (from the t_s round to the T th rounds), respectively Table IV shows the accuracy comparisons for the two clustering scenarios. We find that we can get much higher model accuracy for the clients that are clustered after model stability becomes stable.

In addition, we conduct experiments to compare the average model accuracy of client clustering guided by model stability or the converged models. We apply the model stability to an existing method, i.e., *MTCFL* [22]. Table V shows the model accuracy of *MTCFL* and *MTCFL* with model stability in different non-IID settings over the CIFAR10 dataset. The results show that

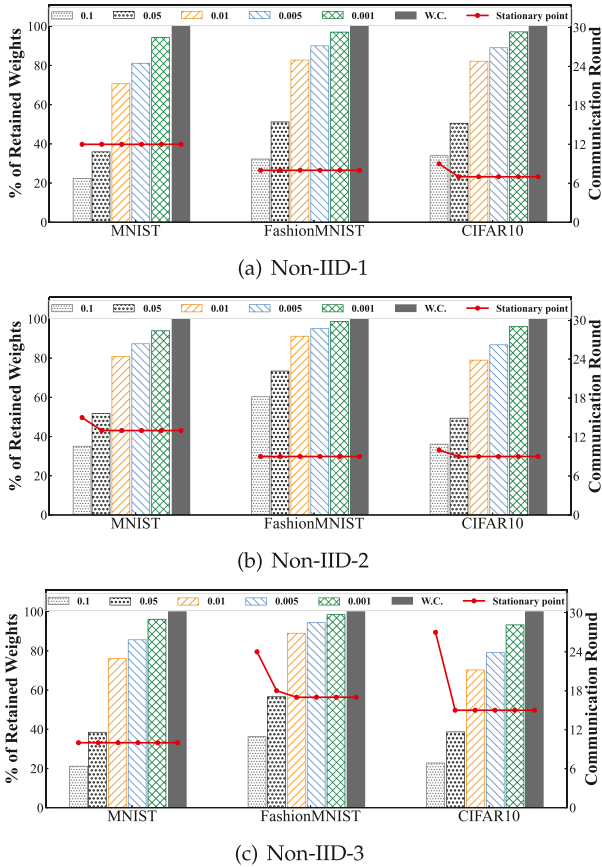


Fig. 9. Impact of ξ on different datasets under various non-IID settings.

TABLE V
COMPARISONS ON MODEL ACCURACY AND REQUIRED COMMUNICATION ROUNDS OF CLIENT CLUSTERING FOR *MTCFL* WITH CONVERGED MODELS OR MODEL STABILITY

Setting	<i>MTCFL</i>		<i>MTCFL</i> with model stability	
	Accuracy	Comm. rounds	Accuracy	Comm. rounds
Non-IID-1	62.84%	37	61.63%	18
Non-IID-2	71.79%	36	72.06%	11
Non-IID-3	40.47%	22	41.18%	19

the model stability-guided clients clustering achieves compared model accuracy as *MTCFL* which groups clients when their local models have been converged. Besides, Table V shows that *MTCFL* with model stability achieves such a model accuracy with much fewer communication rounds.

The experimental results in Fig. 4, Tables IV and V demonstrate that model stability is a better indicator than model loss by earlier reflecting the state of model training. Moreover, model stability is an effective timing indicator to bi-partition clients into suitable clusters and derive more accurate local models for clients with Non-IID data.

Validity of the Reference Client: To validate the effectiveness of our model stability based reference selection method, we conduct an experiment using the CIFAR10 dataset in the Non-IID-2 $\beta = 0.7$ data setting. The experiment results are shown

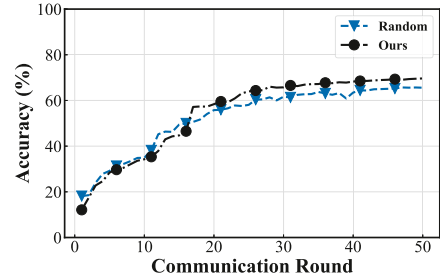


Fig. 10. Model accuracy comparisons with random reference and the reference with the least model stability over CIFAR10 dataset.

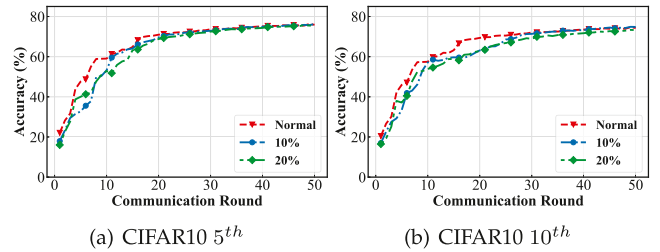


Fig. 11. Auto-correction for improper clustering over CIFAR10 dataset.

in Fig. 10. Compared to client bi-partitioning with the random reference, we see that the reference with the least model stability can achieve higher model accuracy with improvement by 4.1%, which implies that model stability based reference selection can produce much better client clusters than the random reference.

Auto-Correction for Improper Clustering: We conduct an experiment in Non-IID-2 data setting over CIFAR10 dataset to investigate the auto-correction capability of *HiCFL* for improper clustering at the initial stage. To simulate the case where some clients are mistakenly clustered, we artificially divide different proportions, e.g., 10% and 20%, of clients into incorrect clusters in the early stage, e.g., at the 5th and 10th communication round of FL training. We compare the average model accuracy of different cases and the normal execution where clients are clustered without intervention, and report the results in Fig. 11.

Fig. 11 shows that if a fraction of clients is improperly clustered in the early stage, the average model accuracy will be affected, with lower accuracy than the normal execution. However, with continuous model training, the model accuracy will tend to be consistent, in all cases, with the normal execution after some communication rounds. The results demonstrate that *HiCFL* can correct improper client clustering and finally achieves high model accuracy.

Re-Organizing Clusters versus Fine-Grained Clustering: We conduct experiments to compare the performance of the two design choices using CIFAR10 dataset. For the design of re-organizing clusters, we re-organize the clusters using their global models after client clustering is completed. The cosine similarity between global model updates of any two clusters is calculated, and whether two clusters should be merged or not is determined based on the cosine similarity. Clusters are merged according to the following rules: 1) a smaller cluster is merged into the

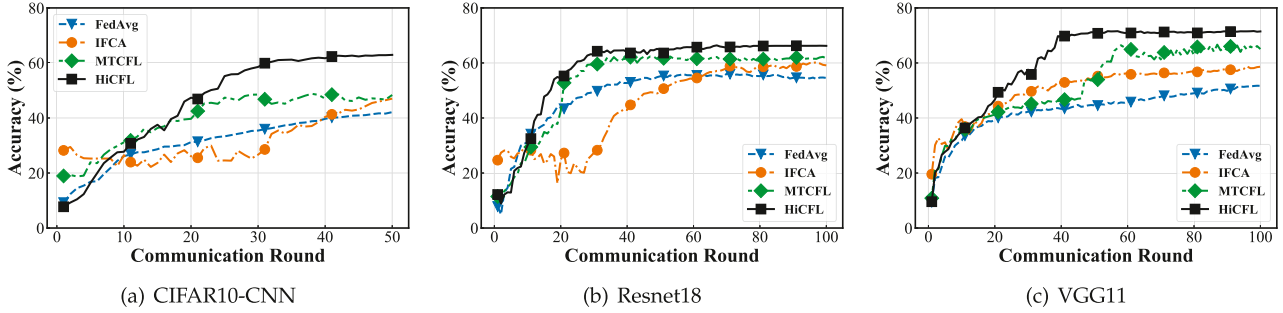
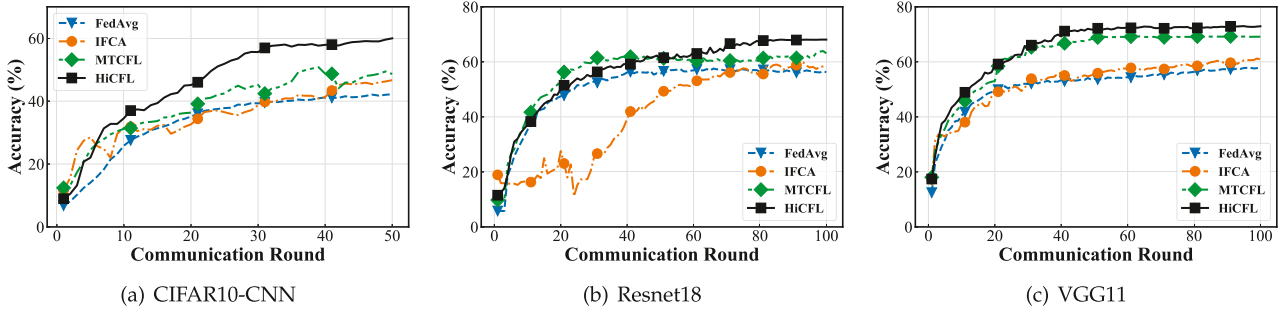
Fig. 12. “Time-accuracy” training curves on CIFAR10-CNN model, Resnet18, and VGG11, respectively, with $\alpha = 0.3$.Fig. 13. “Time-accuracy” training curves on CIFAR10-CNN model, Resnet18, and VGG11, respectively, with $\alpha = 0.5$.

TABLE VI
COMPARISONS ON THE AVERAGE MODEL ACCURACY OF RE-ORGANIZING CLUSTERS, FINE-GRAINED CLUSTERING AND PERSONALIZED MODELS

Setting	Re-organizing	Fine-grained clustering	Personalized
Non-IID-1	65.09%	65.24%	65.83%
Non-IID-2	76.66%	78.09%	79.17%
Non-IID-3	42.27%	44.08%	45.18%

larger cluster; 2) cluster G_1 will be merged with the cluster that has the highest cosine similarity with G_1 ; 3) the two clusters that should be merged don't have a third cluster with a conflict. There is a conflict if two clusters are dissimilar, i.e., with a negative cosine similarity. For example, cluster G_1 is similar to cluster G_2 , and cluster G_1 is similar to cluster G_3 , but cluster G_2 conflicts with cluster G_3 , then cluster G_1 and G_2 will not be merged. If condition 3) is not satisfied, then we will consider to merge cluster G_1 with the next most similar cluster.

For the design choice of fine-grained clustering, after the clustering is complete, we will divide the clients of a cluster into two sub-clusters following the current design of *HiCFL*. Besides, we continue to bi-partition the clusters until each cluster contains one client only, i.e., generating personalized model for each client.

Table VI shows the average model accuracy for the three design choices. In general, fine-grained clustering can derive better model accuracy than the design of re-organizing clusters, while the design of personalized models achieves the highest model

accuracy. Compared to re-organizing clusters, an additional benefit of fine-grained clustering is model personalization. In the early stage, *HiCFL* groups clients with similar data distribution to train a global model, while in the later stage, *HiCFL* can refine clients' model of each cluster to learn more personal information from their own data. Therefore, fine-grained clustering could provide more personalized services for the users.

D. Evaluation With Complex Data and Models

Complex Models and Natural Data Distribution: In the experiments, we train three models, namely CIFAR10-CNN as described in Section V-A, Resnet18, and VGG11, by using different FL methods on the CIFAR10 dataset. Figs. 12 and 13 show the “time-accuracy training” curves of different methods under the Latent Dirichlet allocation with $\alpha = 0.3$ and $\alpha = 0.5$, respectively. From the results, we have the following observations. First, *HiCFL* achieves the best model accuracy during the training processes of the three models. It demonstrates that *HiCFL* can work well with complex models. In most of the cases, *FedAvg* has the lowest model accuracy than CFL-like methods. Second, *HiCFL* has a more stable training curve than other methods. This is because *HiCFL* can find the proper clustering timing for bi-partitioning clusters and can accurately group the clients with similar data distribution. As a result, model accuracy can be stably improved by *HiCFL*.

The results of the experiment demonstrate the phenomenon that the performance improvement of *HiCFL* with complex data is not as significant as that with simple data. For simple

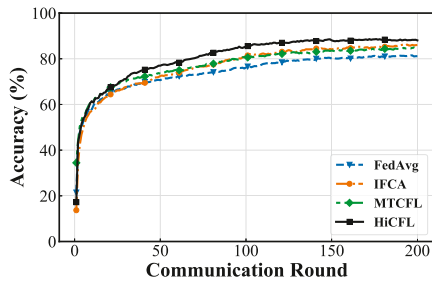


Fig. 14. Training curves of HAR models with different methods on the USC-HAD dataset.

data, the features can be extracted easily, and there are significant differences between the features of different classes. As a result, clients with Non-IID simple data can be effectively separated into distinct clusters. However, complex data possess more intricate features, and it is not easy to discern the features of different classes. Additionally, *HiCFL* assesses the similarity between client data by calculating the cosine similarity of model updates. In the case of complex data, the disparities between the model updates may not be adequately reflected in the model similarity. This can lead to improper clustering and subsequently a less notable improvement in accuracy.

Experiment on Realistic Dataset: In the experiment, we assign IMU data of 14 volunteers to each of the 14 clients to simulate the natural non-IID data setting. We use different FL methods to train convolutional models for the clients to realize human activity recognition (HAR). There are 200 communication rounds between the server and the clients.

Fig. 14 shown the results of *HiCFL* and other methods on the USC-HAD dataset. The results show that *HiCFL* outperforms existing methods in “time-accuracy” under the natural non-IID data setting. Compared to the existing methods, *HiCFL* can achieve higher accuracy at an earlier stage. It implies that *HiCFL* can work well with the natural non-IID data. Thanks to the use of model stability for finding the proper timing of client clustering, *HiCFL* can complete the client clustering process more quickly and derive stable models much earlier.

Experiment on an FL Testbed: We first use a device configured with an Intel 5300 NIC to collect WiFi Channel State Information (CSI) data from 10 volunteers aged 21-26 years old. Six of the ten volunteers are male and four are female. Each volunteer was required to complete six specified gestures (e.g., clapping, pushing, pulling, and so on), each repeated 20 times. To understand the performance of all methods on mobile devices, we randomly assign the CSI data from each of the volunteers to 6 Raspberry Pi 3B+ and 4 laptops, which forms a simple FL testbed. We implement all methods using PySyft. Besides, we process CSI data and build a CNN model for gesture recognition following the operations proposed in [42]. To train the CNN model, we set the batch size as 8 and the learning rate as 0.01. The model training process lasts about 50 minutes.

We train the gesture recognition models using different models on our FL testbed, and their “time-accuracy” curves are shown in Fig. 15. Compared to *FedAvg*, CFL-like methods can

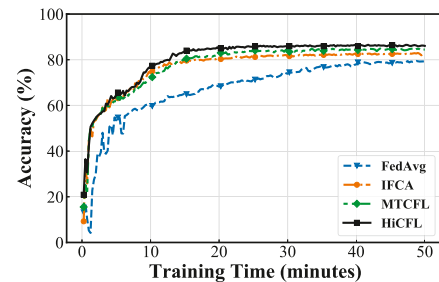


Fig. 15. Training curves of gesture recognition models with different methods on the CSI dataset.

achieve higher accuracy with a stable model training process. *HiCFL* performs better than other CFL-like methods on the FL testbed.

VI. RELATED WORK

Federated learning (FL) is an emerging distributed machine learning paradigm that enables training on large amounts of data that reside on distributed clients without compromising data privacy. Statistical heterogeneity challenge, in particular non-IID data, is the focus of research in FL, and has attracted many efforts [1], [16], [18], [20], [21], [43], [44], [45], [46] in the literature. For example, McMahan et al. [1] attempt to overcome the non-IID issue by averaging clients’ local models. Zhao et al. [16] assume that only a subset of data are shared among clients for model training with FL. Li et al. [43] reduce feature shift with batch normalization before averaging the models. Wang et al. [18] and Li et al. [20] mitigate the impact of non-IID data from client selections and sample selections, respectively. However, these approaches train only one single global model for all clients, which is usually of limited generality and insufficient to effectively eliminate the impact of non-IID data. For example, Ouyang et al. [11] show that one single model is not applicable to human activity recognition.

A promising way to address the non-IID challenge is to train multiple global models or personalized models for all clients according to their data distributions [47], [48], [49]. For example, Smith et al. [47] exploit multi-task learning for building multiple models in FL. Feng et al. [49] use transfer learning to improve the accuracy of FL trained model by personalizing it with local data. Tu et al. [46] propose a novel federated learning system, named FedDL, for human activity recognition based on the bottom-up layer-wise dynamic layer sharing scheme. Our work differs from FedDL in both design motivation and operations. We propose model stability as a means to determine the proper clustering time, whereas FedDL implicitly groups clients based on layer similarity in a bottom-up fashion.

Recently, the framework of clustered federated learning (CFL) [22] inspires some novel approaches [23], [24], [25], [26], [50], which partition clients with similar data distributions into clusters and train a global model for clients of each cluster. Saputra et al. [9] propose a novel economic-efficiency framework for the electric vehicle network to maximize the profits of charging stations, which exploit FL to train an energy demand prediction for each station. To reduce the bias in energy demand

prediction, they employ a constrained K-means algorithm to divide charging stations into a predefined number of clusters based on their deployment locations, since nearby stations may have similar profiles. However, existing CFL approaches either require to input the number κ of clusters, e.g., [23], [24], [25], [26], or are inefficient in terms of communications, e.g., [22]. In general, we cannot determine the optimal κ with no knowledge on the clients' data distributions [51]. Besides, communication efficiency is essentially important for many FL applications scenarios, where mobile clients are usually limited by battery energy and network traffics. Different from previous works, *HiCFL* exploits the novel concept of model stability to intelligently bi-partition clients in a hierarchical manner, which is communication-efficient and requires no knowledge of κ .

VII. CONCLUSION

In this paper, we present *HiCFL*, an efficient CFL approach to alleviate the impact of non-IID data. Built on the novel concept of model stability, *HiCFL* can intelligently determine the opportune time to separate a cluster and properly group all clients into clusters. We evaluate *HiCFL* with three popular datasets under various non-IID settings. Experimental results demonstrate that *HiCFL* significantly outperforms state-of-the-art approaches, e.g., improving model accuracy by 2.0%~9.0% and reducing communication costs by 27.3%~80.6%.

In the future, we will continue to explore more designs that facilitate the FL on mobile devices, such as effective FL over heterogeneous mobile devices with varying computational and communication resources.

REFERENCES

- [1] B. McMahan, E. Moore, D. Ramage, S. Hampson, and B. Aguera y Arcas, "Communication-efficient learning of deep networks from decentralized data," in *Proc. 20th Int. Conf. Artif. Intell. Statist.*, 2017, pp. 1273–1282.
- [2] A. Jalalirad, M. Scavuzzo, C. Capota, and M. Sprague, "A simple and efficient federated recommender system," in *Proc. IEEE/ACM 6th Int. Conf. Big Data Comput. Appl. Technol.*, 2019, pp. 53–58.
- [3] K. Maeng, H. Lu, L. Melis, J. Nguyen, M. Rabbat, and C.-J. Wu, "Towards fair federated recommendation learning: Characterizing the inter-dependence of system and data heterogeneity," in *Proc. ACM Conf. Recommender Syst.*, 2022, pp. 156–167.
- [4] W. Zheng, L. Yan, C. Gou, and F.-Y. Wang, "Federated meta-learning for fraudulent credit card detection," in *Proc. 29th Int. Joint Conf. Artif. Intell.*, 2020, Art. no. 642.
- [5] Y. Cheng, Y. Liu, T. Chen, and Q. Yang, "Federated learning for privacy-preserving AI," *Commun. ACM*, vol. 63, no. 12, pp. 33–36, 2020.
- [6] N. Rieke et al., "The future of digital health with federated learning," *NPJ Digit. Med.*, vol. 3, no. 1, pp. 1–7, 2020.
- [7] D. Zhang, Z. Kou, and D. Wang, "FedSens: A federated learning approach for smart health sensing with class imbalance in resource constrained edge computing," in *Proc. IEEE Conf. Comput. Commun.*, 2021, pp. 1–10.
- [8] Q. Wu, X. Chen, Z. Zhou, and J. Zhang, "FedHome: Cloud-edge based personalized federated learning for in-home health monitoring," *IEEE Trans. Mobile Comput.*, vol. 21, no. 8, pp. 2818–2832, Aug. 2022.
- [9] Y. M. Saputra et al., "Federated learning meets contract theory: Economic-efficiency framework for electric vehicle networks," *IEEE Trans. Mobile Comput.*, vol. 21, no. 8, pp. 2803–2817, Aug. 2022.
- [10] Y. M. Saputra, D. T. Hoang, D. N. Nguyen, L.-N. Tran, S. Gong, and E. Dutkiewicz, "Dynamic federated learning-based economic framework for Internet-of-Vehicles," *IEEE Trans. Mobile Comput.*, vol. 22, no. 4, pp. 2100–2115, Apr. 2023, doi: [10.1109/TMC.2021.3122436](https://doi.org/10.1109/TMC.2021.3122436).
- [11] X. Ouyang, Z. Xie, J. Zhou, J. Huang, and G. Xing, "ClusterFL: A similarity-aware federated learning system for human activity recognition," in *Proc. 19th Annu. Int. Conf. Mobile Syst. Appl. Serv.*, 2021, pp. 54–66.
- [12] X. Cui, S. Lu, and B. Kingsbury, "Federated acoustic modeling for automatic speech recognition," in *Proc. IEEE Int. Conf. Acoust. Speech Signal Process.*, 2021, pp. 6748–6752.
- [13] K. Yue, R. Jin, R. Pilgrim, C.-W. Wong, D. Baron, and H. Dai, "Neural tangent kernel empowered federated learning," in *Proc. Int. Conf. Mach. Learn.*, K. Chaudhuri, S. Jegelka, L. Song, C. Szepesvári, G. Niu, and S. Sabato, Eds., 2022, pp. 25783–25803.
- [14] F. Lai, X. Zhu, H. V. Madhyastha, and M. Chowdhury, "Oort: Efficient federated learning via guided participant selection," in *Proc. USENIX Symp. Operating Syst. Des. Implementation*, 2021, pp. 19–35.
- [15] M. H. Shullar, A. A. Abdellatif, and Y. Massoud, "Energy-efficient active federated learning on non-IID data," in *Proc. IEEE Int. Midwest Symp. Circuits Syst.*, 2022, pp. 1–4.
- [16] Y. Zhao, M. Li, L. Lai, N. Suda, D. Cavin, and V. Chandra, "Federated learning with non-IID data," 2018, *arXiv:1806.00582*.
- [17] X. Li, K. Huang, W. Yang, S. Wang, and Z. Zhang, "On the convergence of FedAvg on non-IID data," in *Proc. Int. Conf. Learn. Representations*, 2020, pp. 1–26.
- [18] H. Wang, Z. Kaplan, D. Niu, and B. Li, "Optimizing federated learning on non-IID data with reinforcement learning," in *Proc. IEEE Conf. Comput. Commun.*, 2020, pp. 1698–1707.
- [19] T. Huang, W. Lin, W. Wu, L. He, K. Li, and A. Y. Zomaya, "An efficiency-boosting client selection scheme for federated learning with fairness guarantee," *IEEE Trans. Parallel Distrib. Syst.*, vol. 32, no. 7, pp. 1552–1564, Jul. 2021.
- [20] L. Anran, Z. Lan, T. Juntao, Q. Yaxuan, W. Junhao, and L. Xiang-Yang, "Sample-level data selection for federated learning," in *Proc. IEEE Conf. Comput. Commun.*, 2021, pp. 1–10.
- [21] T. Li, A. K. Sahu, M. Zaheer, M. Sanjabi, A. Talwalkar, and V. Smith, "Federated optimization in heterogeneous networks," in *Proc. Mach. Learn. Syst.*, 2020, pp. 429–450.
- [22] F. Sattler, K.-R. Müller, and W. Samek, "Clustered federated learning: Model-agnostic distributed multitask optimization under privacy constraints," *IEEE Trans. Neural Netw. Learn. Syst.*, vol. 32, no. 8, pp. 3710–3722, Aug. 2021.
- [23] C. Briggs, Z. Fan, and P. Andras, "Federated learning with hierarchical clustering of local updates to improve training on non-IID data," in *Proc. IEEE Int. Joint Conf. Neural Netw.*, 2020, pp. 1–9.
- [24] A. Ghosh, J. Chung, D. Yin, and K. Ramchandran, "An efficient framework for clustered federated learning," in *Proc. Annu. Conf. Neural Inf. Process. Syst.*, 2020, Art. no. 1643.
- [25] G. Long, M. Xie, T. Shen, T. Zhou, X. Wang, and J. Jiang, "Multi-center federated learning: Clients clustering for better personalization," *World Wide Web*, vol. 26, pp. 481–500, 2023.
- [26] A. Ghosh, J. Hong, D. Yin, and K. Ramchandran, "Robust federated learning in a heterogeneous environment," 2019, *arXiv:1906.06629*.
- [27] N. Wang, R. Zhou, L. Su, G. Fang, and Z. Li, "Adaptive clustered federated learning for clients with time-varying interests," in *Proc. IEEE/ACM 30th Int. Symp. Qual. Service*, 2022, pp. 1–10.
- [28] A. Z. Tan, H. Yu, L. Cui, and Q. Yang, "Towards personalized federated learning," *IEEE Trans. Neural Netw. Learn. Syst.*, early access, Mar. 28, 2022, doi: [10.1109/TNNLS.2022.3160699](https://doi.org/10.1109/TNNLS.2022.3160699).
- [29] M. Long, Y. Cao, Z. Cao, J. Wang, and M. I. Jordan, "Transferable representation learning with deep adaptation networks," *IEEE Trans. Pattern Anal. Mach. Intell.*, vol. 41, no. 12, pp. 3071–3085, Dec. 2019.
- [30] L. Mou et al., "How transferable are neural networks in NLP applications?," in *Proc. Conf. Empir. Methods Natural Lang. Process.*, 2016, pp. 479–489.
- [31] J. Yosinski, J. Clune, Y. Bengio, and H. Lipson, "How transferable are features in deep neural networks?," in *Proc. Adv. Neural Inf. Process. Syst.*, 2014, pp. 3320–3328.
- [32] O. Dekel, R. Gilad-Bachrach, O. Shamir, and L. Xiao, "Optimal distributed online prediction using mini-batches," *J. Mach. Learn. Res.*, vol. 13, no. 1, pp. 165–202, 2012.
- [33] H. Wang, M. Yurochkin, Y. Sun, D. S. Papailiopoulos, and Y. Khazaeni, "Federated learning with matched averaging," in *Proc. Int. Conf. Learn. Representations*, 2020, pp. 1–16.

- [34] T. Li, A. K. Sahu, A. Talwalkar, and V. Smith, "Federated learning: Challenges, methods, and future directions," *IEEE Signal Process. Mag.*, vol. 37, no. 3, pp. 50–60, May 2020.
- [35] A. Krizhevsky et al., "Learning multiple layers of features from tiny images," 2009.
- [36] Y. LeCun, L. Bottou, Y. Bengio, and P. Haffner, "Gradient-based learning applied to document recognition," *Proc. IEEE*, vol. 86, no. 11, pp. 2278–2324, Nov. 1998.
- [37] H. Xiao, K. Rasul, and R. Vollgraf, "Fashion-MNIST: A novel image dataset for benchmarking machine learning algorithms," 2017, *arXiv:1708.07747*.
- [38] K. He, X. Zhang, S. Ren, and J. Sun, "Deep residual learning for image recognition," in *Proc. IEEE Conf. Comput. Vis. Pattern Recognit.*, 2016, pp. 770–778.
- [39] K. Simonyan and A. Zisserman, "Very deep convolutional networks for large-scale image recognition," in *Proc. Int. Conf. Learn. Representations*, 2015, pp. 1–14.
- [40] M. Zhang and A. A. Sawchuk, "USC-HAD: A daily activity dataset for ubiquitous activity recognition using wearable sensors," in *Proc. ACM Conf. Ubiquitous Comput.*, 2012, pp. 1036–1043.
- [41] J. Zhang et al., "DENSE: Data-free one-shot federated learning," in *Proc. Adv. Neural Inf. Process. Syst.*, 2022, pp. 21414–21428.
- [42] T. Xing et al., "WiFine: Real-time gesture recognition using Wi-Fi with edge intelligence," *ACM Trans. Sensor Netw.*, vol. 19, no. 1, pp. 1–24, 2022.
- [43] X. Li, J. Meirui, X. Zhang, M. Kamp, and Q. Dou, "FedBN: Federated learning on non-IID features via local batch normalization," in *Proc. Int. Conf. Learn. Representations*, 2021, pp. 1–27.
- [44] D. Yongheng et al., "FAIR: Quality-aware federated learning with precise user incentive and model aggregation," in *Proc. IEEE Conf. Comput. Commun.*, 2021, pp. 1–10.
- [45] Z. Zhou, Y. Li, X. Ren, and S. Yang, "Towards efficient and stable k -asynchronous federated learning with unbounded stale gradients on non-IID data," *IEEE Trans. Parallel Distrib. Syst.*, vol. 33, no. 12, pp. 3291–3305, Dec. 2022.
- [46] L. Tu, X. Ouyang, J. Zhou, Y. He, and G. Xing, "FedDL: Federated learning via dynamic layer sharing for human activity recognition," in *Proc. 19th ACM Conf. Embedded Netw. Sensor Syst.*, 2021, pp. 15–28.
- [47] V. Smith, C.-K. Chiang, M. Sanjabi, and A. Talwalkar, "Federated multi-task learning," in *Proc. 31st Int. Conf. Neural Inf. Process. Syst.*, 2017, pp. 4427–4437.
- [48] O. Marfoq, G. Neglia, A. Bellet, L. Kameni, and R. Vidal, "Federated multi-task learning under a mixture of distributions," in *Proc. Int. Conf. Neural Inf. Process. Syst.*, M. Ranzato, A. Beygelzimer, Y. N. Dauphin, P. Liang, and J. W. Vaughan, Eds., 2021, pp. 15434–15447.
- [49] J. Feng, C. Rong, F. Sun, D. Guo, and Y. Li, "PMF: A privacy-preserving human mobility prediction framework via federated learning," in *Proc. ACM Interactive Mobile Wearable Ubiquitous Technol.*, vol. 4, no. 1, pp. 1–21, 2020.
- [50] A. Krizhevsky et al., "Learning multiple layers of features from tiny images," Univ. Toronto, Tech. Rep., pp. 1–60, 2009.
- [51] Z. Wang, H. Xu, J. Liu, H. Huang, C. Qiao, and Y. Zhao, "Resource-efficient federated learning with hierarchical aggregation in edge computing," in *Proc. IEEE Conf. Comput. Commun.*, 2021, pp. 1–10.



Biyao Gong received the BE degree in computer science and technology from Northwest University, Xi'an, China, in 2020. He is currently working toward the master's degree in computer science and technology with Northwest University. His research interests include machine learning and federated learning.



Tianzhang Xing received the BE degree in telecommunications engineering from Xidian University, Xi'an, China, in 2004, the MPhil and PhD degrees in computer science and technology from the Northwest University, Xi'an, in 2009 and 2014. He is currently an associate professor with the School of Information and Technology, Northwest University. His research interests include mobile computing, pervasive computing and wireless networks.



Zhidan Liu (Member, IEEE) received the PhD degree in computer science and technology from Zhejiang University, Hangzhou, China, in 2014. After that, he worked as a research fellow with Nanyang Technological University, Singapore. He is currently an associate professor with the College of Computer Science and Software Engineering, Shenzhen University, Shenzhen, China. His research interests include mobile computing, Big Data analytics, Internet of Things, and urban computing. He is a member of ACM and CCF.



Wei Xi (Member, IEEE) received the PhD degree in computer science from Xi'an Jiaotong University, in 2014. He is currently an associate professor with Xi'an Jiaotong University. His main research interests include wireless networks, mobile computing, and artificial intelligence. He is a member of CCF, ACM.



Xiaojiang Chen received the PhD degree in computer software and theory from Northwest University, Xi'an, China, in 2010. He is currently a professor with the School of Information Science and Technology, Northwest University. His current research interests include localization and performance issues in wireless ad hoc, mesh, and sensor networks and named data networks.