Adaptive Client Clustering for Efficient Federated Learning Over Non-IID and Imbalanced Data

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

Abstract—Federated learning (FL) is an emerging distributed and privacy-preserving machine learning framework. However, the performance of traditional FL methods is seriously impaired by the real-world data, which appear to be non-independent and identically distributed (non-IID). The recent clustered federated learning (CFL) methods eliminate the impact of non-IID data by grouping clients with similar data distribution into the same cluster. Unfortunately, existing CFL methods heavily rely on the pre-setting of the cluster number, failing to achieve adaptive client clustering. Even worse, we experimentally observe that imbalanced data across clients largely degrade their correctness of client clustering. In this paper, we present a novel CFL method without manual intervention, named AutoCFL, which can eliminate both effects of non-IID and imbalanced data simultaneously. To deal with imbalanced data, the local training adjustment strategy adaptively adjusts the number of local training epochs for each client. To further improve the clustering correctness and adaptability, the weighted voting-based client clustering strategy automatically groups each client into an appropriate cluster. Extensive experiments are conducted to evaluate the design of *AutoCFL* with three popular datasets under various data settings. Experimental results demonstrate that *AutoCFL* outperforms the state-of-the-art methods under non-IID and imbalanced data settings, e.g., on average improving the model accuracy by 9.24% when compared to the standard FL method, i.e., *FedAvg*, while significantly reducing communication costs by $4.67\times$ in an adaptive client clustering manner.

Index Terms—Federated learning, clustered federated learning, non-IID data, imbalanced data, client clustering, weighted voting

1 Introduction

The rapid development of computing and storage capacity on edge devices is driving the shift of machine learning from the data-center mode toward the edge mode [3], [12]. Various edge devices, such as smartphones, wearables, and smart TVs, can collect, store, and manipulate user-produced data locally, which provides the data foundation for edge-assisted machine learning. To leverage the massive amount of data associated with users' personal devices, large companies like Google generally adopt *Federated Learning (FL)* [14], [19], [42], a novel distributed machine learning paradigm, to let multiple participating clients collaboratively train a globally shared model without exposing users' raw data. Due to its privacy-preserving property, FL has been widely applied in a variety of applications, e.g., computer vision [24], [25], natural language processing [9], [43], and human activity recognition [31], [37].

The lifecycle of an FL training typically involves three key steps [28]: *1)* The server broadcasts current global model to the clients; *2)* Each client updates the global model using its local data; and *3)* The server collects and aggregates all locally updated models to update the global model. The server and all clients interact with each other over the network and keep performing above three steps until a model accuracy target is achieved or the upper limit on the number of communication rounds is reached. In practice, FL usually involves thousands or even millions of participating clients, which are owned by different users with diverse user habits and usage environments. For example, different users may use their smartphones to browse different types of content with diverse frequencies. As a result, training data in FL may possess two kinds of statistical heterogeneity: non-IID (independent identically distributed) data distribution and imbalanced data samples across the clients. More specifically, non-IID data indicate that data samples from different clients do not obey the same data distribution, while imbalanced data imply that the quantity of data samples varies largely across the clients.

The primary goal of FL is to train a model with the highest possible accuracy in the shortest possible time [16], [20]. However, non-IID data across clients have been confirmed to hinder this goal by prolonging the training time while only achieving the sub-optimal accuracy [12], [22], [45]. Therefore,

- *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, Northwest University, Xi'an, Shaanxi 710069, China and also with the Internet of Things Research Center, Norhwest University, Xi'an, Shaanxi 710069, China. E-mail: {xtz, xjchen}@nwu.edu.cn.*
- *Zhidan Liu is with the College of Computer Science and Software Engineering, Shenzhen University, Shenzhen, Guangdong 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.*

Manuscript received 4 Jan. 2022; revised 9 Mar. 2022; accepted 8 Apr. 2022. Date of publication 19 Apr. 2022; date of current version 13 Nov. 2024.
This work was supported in part by NSFC under Grant 62172284, in part by the International Cooperation Project of Shaanxi Province under Grant 2020KW-004, and in part by the Natural Science Foundation of Guangdong under Grant 2020A1515011502. The work of Wei Xi was supported by the National Key R&D Program of China under Grant 2020YFB1707700 and in part by the NSFC under Grant 61832008.
(Corresponding authors: Tianzhang Xing and Zhidan Liu.)
Recommended for acceptance by Guest Editors for the Special Issue on Trustable, Verifiable, and Auditable Federated Learning.
Digital Object Identifier no. 10.1109/TBDATA.2022.3167994

in the literature there exist a plenty of research works that focus on mitigating the impact of non-IID data on FL [13], [16], [20], [35], [39]. In particular, clustered federated learning (CFL) [4], [6], [7], [31], [33], [41] seems to be a promising framework to alleviate the impact of non-IID data by dividing clients with similar data distribution into the same cluster, and training a shared model for each client cluster individually. Since either raw data or statistical features of these data cannot be directly used to guide the client clustering, existing CFL methods usually utilize three common intermediate results from collaborative model training, i.e., local model updates [33], local model weights [4], [31], [41], and empirical loss of the local model [7], to implicitly measure the similarity of clients' data distribution. Previous studies demonstrate that CFL can improve model accuracy and meanwhile reduce the required communication rounds [33], [34], showing a great advantage on confronting the non-IID challenge in FL.

Despite the promising performance, existing CFL methods, however, are still inefficient and non-adaptive on client clustering. Most of existing CFL works [4], [6], [7], [27], [41] require the specific number k of client clusters as the input, lacking of adaptability. With no prior knowledge on the clients' data distribution, it is hard to determine the optimal k. Heuristic attempts on setting k inevitably incur huge communication and computation costs, yet with no guarantee on finding the best setting. Although iterative bi-partition client clustering proposed in [33] does not require the number k as input, such a clustering process is communicationally inefficient. This is because such a method requires local models convergence before bi-partitioning clusters each time.

Moreover, existing CFL works aim at attacking non-IID data challenge, while largely ignoring the impact of imbalanced data across clients on the client clustering process. Unlike the well-known non-IID data challenge, the imbalanced data challenge has not received much attention, despite the prevalence of imbalanced data in reality. However, we experimentally observe that imbalanced data severely affect the correctness of client clustering. Especially for these local model weight based CFL methods [4], [31], [37], [41], they cannot accurately discriminate the data distribution similarity between clients with imbalanced data samples. This is because different amount of training samples leads to varied training states of local models, and the model differences caused by imbalanced data cover up the differences caused by non-IID data. This critical yet neglected issue hinders the applications of CFL, and calls for timely research efforts.

To further advance existing CFL works, in this paper, we present a novel CFL method, named *AutoCFL*, which can automatically group clients with non-IID and imbalanced data into the suitable clusters. The design of *AutoCFL* is mainly motivated from two aspects. First, imbalanced data can impair the correctness of client clustering by affecting the local model weights. The reason behind is that, according to our experiment, the local empirical losses of local models become inconsistent among the clients with similar data distribution, leading to improper client clustering. Thus, we propose the local training adjustment strategy that dynamically adjusts the number of local training epochs for each client given its data amount. This strategy can diminish the errors of local model weight distance caused by imbalanced data, and the resultant local model weight distances better reflect the similarity of all clients' data distribution.

Second, existing CFL methods fail to consider both the efficiency and the adaptability of client clustering. Instead of pre-setting or trying the number k of clusters, we propose an adaptive client clustering strategy that can extract the cluster structure of clients from the corrective model weight distances between clients. Inspired by the important criterion of evaluating clusters, i.e., minimum inter-cluster distance should be not smaller than the maximum intra-cluster distance, our strategy can automatically detect the cluster demarcation, and group clients into clusters with a well-designed weighted voting mechanism. The voting process takes imbalanced data number of clients into account, and thus enhances the correctness of client clustering further. Combining these two strategies, *AutoCFL* can achieve efficient and adaptive client clustering by addressing the challenges of non-IID and imbalanced data simultaneously.

Our main contributions are summarized as follows:

- We are the first, to the best of our knowledge, to study the impact of both non-IID and imbalanced data across clients on existing CFL methods. The imbalanced data lead to improper client clustering, and further deteriorate the performances of CFL methods along with non-IID data.
- We present *AutoCFL* to advance the CFL researches, and attack both non-IID data and imbalanced data challenges with two novel strategies, i.e., local training adjustment strategy and weighted voting based adaptive client clustering strategy.
- We conduct extensive experiments to evaluate *AutoCFL* over three popular datasets under a variety of data settings. The experimental results have demonstrated the effectiveness and efficiency of *AutoCFL*. Under the non-IID and imbalanced data settings, *AutoCFL* averagely reduces $4.67\times$ communication costs when compared to state-of-the-art methods, while improving model accuracy by 9.24% when compared to the standard FL method, i.e., *FedAvg*, in an adaptive clustering manner. *AutoCFL* outperforms state-of-the-art methods under non-IID and balanced data settings as well, e.g., averagely reducing $4.28\times$ communication costs with improved model accuracy.

The rest of the paper is organized as follows. Section 2 reviews the related works. We introduce the background and motivation in Section 3. The design of *AutoCFL* is elaborated and evaluated in Sections 4 and 5, respectively. Section 6 finally concludes this paper.

2 Related Work

We review existing research efforts on addressing the non-IID data challenge and imbalanced data challenge in federated learning (FL) in Sections 2.1 and 2.2, respectively. Since clustered federated learning (CFL) has become an important branch of FL and attracted much attention in recent years, we thus discuss these works separately in Section 2.3 to highlight our contributions in this paper.

2.1 Non-IID Data Challenge in Federated Learning

Since users may have different usage habits and patterns, the distribution of local data stored in different clients varies greatly and is not similar to each other i.e., practical data used for FL training are non-IID. The most common FL method *FedAvg* [28] does not consider such a challenge, and fails to achieve good performance in real-world applications. Many works [18], [20], [45] have extensively demonstrate that non-IID data not only affect the training convergence with increased communication overheads, but also make the model trained by native FL less accurate. In the literature, many efforts have been made to address the non-IID data challenge in FL from different aspects.

Client Drift Mitigation. The performance degradation of *FedAvg* caused by non-IID is due to client drift [22], [26]. The non-IID data make the local models differ from each other, and thus the averaged model is, as a result, far from the true global optimum. Therefore, some methods have been proposed to attack the non-IID data challenge by addressing the client drift issue. For example, *FedProx* [20] introduces a proximal term for *FedAvg* to limit the difference between the global model and local models, which makes sure that the optimization objectives similar among all local models. In addition, *SCAFFOLD* [13] corrects the bias among clients' local updates by using variance reduction.

Client Selection. Instead of training the global model with all clients, some recent methods [5], [30] turn to rigorously select a subset of clients to participate in the FL training according to some predefined policies. Specifically, *Oort* [16] proposes a participant selection strategy based on the novel concepts of client statistical utility and system utility. FL training with well-selected clients is demonstrated to have better time-to-accuracy performance. Wang *et al.* [39] exploit a reinforcement learning model to select clients with closer model weights in each communication round, which ensures that the clients selected in each round would have more similar data distribution. Besides, Cho *et al.* [5] reveal that selecting clients with higher local empirical loss for FL training will accelerate the model convergence.

Model Personalization. Differs from above approaches, some works [10], [23], [36] try to personalize the global model with each client's local data, so as to provide users with personalized service. For example, Smith *et al.* [35] extend multi-task learning to FL training to simultaneously address the challenges of non-IID data and system heterogeneity.

2.2 Imbalanced Data Challenge in Federated Learning

Most of existing FL works have realized the non-IID data challenge, while they largely neglected the imbalanced data challenge and its impacts on the FL training. In practice, the samples stored in the clients obey different distribution, and meanwhile their sample sizes may be inconsistent as well.

Imbalanced data have been examined to cause *FedAvg* to produce unfair results, i.e., accuracy of the global model is biased towards clients with more samples [21], [29], [38]. Li *et al.* [21] have considered the unfairness problem in FL, and proposed *q-FedAvg*, which encourages a more uniform accuracy distribution by reducing the variation of clients' model accuracy. Wang *et al.* [38] measure the contribution of each client in horizontal and vertical FL respectively, and propose to distribute the benefits according to clients' contributions. Furthermore, Michieli *et al.* [29] analyze the relationship between fairness, accuracy and convergence speed, and thus propose *FairAvg*, a fair model aggregation approach. In particular, they indicate that fair model aggregation can benefit both model accuracy and convergence.

2.3 Clustered Federated Learning

Clustered federated learning (CFL) [33] is an emerging FL training strategy to attack non-IID data challenge. CFL divides clients with similar data distribution into the same cluster by identifying data-related features (including local empirical loss, model weights, or update gradients), and clients of each cluster train a global model collaboratively.

Existing CFL works mainly differ in the operation of identifying clients with similar data distribution. Sattler *et al.* [33] propose to iteratively bi-partition clients into clusters by exploiting the cosine similarity between local model updates. However, their method is not communication efficient, because a cluster is bi-partitioned only after related clients' local models have been converged. Ghosh *et al.* [7] propose to exploit the local empirical loss of clients as the metric to measure clients' similarity on data distribution. The proposed method, however, requires constant communication between clients and the server to build stable clusters, which incurs high communication costs. Furthermore, its performance is heavily affected by the prior setting on the number of clusters, which is hard to decide without knowledge on clients' local data.

In addition to above works, there exist a bunch of works [4], [31], [41] that identify client clusters based on the local model distances calculated from model weights. Although these methods often require few communications between the server and clients for clustering clients, they still require to set the number of clusters in advance. Moreover, we find that imbalanced data across clients can make the local model distances not clearly and correctly, and severely affect the final client clustering results, resulting in poor model accuracy and high communication costs.

In summary, existing CFL works fail to achieve communication efficiency and clustering adaptability. To the best of our knowledge, there are no works that have investigated how to mitigate the impact of imbalanced data on the model weight distance based CFL, neither. To improve existing CFL methods, in this paper we consider both challenges of non-IID data and imbalanced data, and present a novel CFL method, i.e., *AutoCFL*, to adaptively group clients into clusters with high communication and computation efficiency.

3 Background and Motivation

3.1 Federated Learning

Recently, federated learning (FL) [14], [28], [42] has been widely used to train machine/deep learning models (e.g., image classification models) among many distributed clients, denoted by set $\mathbb{M}$, which are coordinated by a central server through T rounds of communications between the server and the clients. Each client $m \in \mathbb{M}$ owns a local dataset D_m of size $|\mathsf{D}_m|$, i.e., client m has samples $\{(x_i^m, y_i^m) \in \mathcal{X} \times \mathcal{Y} : i \in \mathsf{D}_m\}$ for the FL training, where x_i^m is the feature of the

i-th sample in feature space $\mathcal{X}$, while y_i^m is the corresponding label of the sample in label space $\mathcal{Y}$. Besides, we let $\mathbb{C}$ represent all available labels in the target task. In addition, let $\mathbb{D} = \cup_{m\in\mathbb{M}}\mathsf{D}_m$ denote all samples of the clients, and $|\mathbb{D}|$ be the total number of samples. The target model ω consists of a feature extractor and a classifier. Given an input sample (x, y), the feature extractor maps x to a feature vector, while the classifier outputs the probability distribution over all labels based on the feature vector.

Training a global model requires constant interaction between clients and the server, which involves iterative executions of two key operations, i.e., local model updating at clients and global model aggregation at server side. Without loss of generality, in t-th round of the FL training, the server broadcasts the current global model ω^{t-1} to the clients. Once received, each client updates the global model ω^{t-1} using its local data. The two key operations are detailed as follows.

Local Model Updating at Client Side. Each client $m \in \mathbb{M}$ uses its local data to update the model ω^{t-1} to ω_m^t, whose goal is to minimize the local empirical loss:

$$\min_{\omega_m^t} \mathbb{E}_{(x_i^m, y_i^m)\sim\mathsf{D}_m}\big[\mathcal{L}_m(\omega_m^t; x_i^m, y_i^m)\big], \tag{1}$$

where $\mathcal{L}_m$ is the loss function for client m. In the typical FL training, e.g., *FedAvg* [28], each client performs E_m steps of stochastic gradient descent (SGD) on the local dataset to update ω^{t-1} with cross entropy loss function defined as

$$\begin{aligned}\omega_m^t &= \omega^{t-1} - \eta\nabla\mathcal{L}_m(\omega^{t-1}) \\ &= \omega^{t-1} - \eta\sum_{c\in\mathbb{C}} p(y=c)\nabla\mathbb{E}_{x|y=c}\big[\log\left(f_c(\omega^{t-1}, x)\right)\big],\end{aligned} \tag{2}$$

where η is the learning rate, and f_c represents the probability of predicting sample x as the class c. It is worth noting that the number of local SGD execution steps E_m is set the same for each client in the standard FL training.

Global Model Aggregation at Server Side. After the local training, each client sends back the local model ω_m^t or only model updates $\Delta_m^t \triangleq \omega_m^t - \omega^{t-1}$ to the server. Once received all local models or local model updates, the server then aggregates the received parameters. As an example, we illustrate the server's model aggregation process in *FedAvg* [28] using model updates Δ_m^t, i.e.,

$$\Delta^t = \sum_{m\in\mathbb{M}} \frac{|\mathsf{D}_m|}{|\mathbb{D}|}\cdot\Delta_m^t, \omega^t \leftarrow \omega^{t-1} - \Delta^t. \tag{3}$$

In FL, the clients' local model updating and the server's global model aggregation are iteratively performed to optimize the following global objective:

$$\min_{\omega}\left\{F(\omega) \triangleq \sum_{m\in\mathbb{M}}\frac{|\mathsf{D}_m|}{|\mathbb{D}|}\cdot F_m(\omega)\right\}, \tag{4}$$

where $F_m(\omega)$ denotes the local empirical loss of client m.

3.2 Clustering Clients to Attack Non-IID Data Challenge

Among existing techniques proposed to attack non-IID data challenge in FL, clustered federated learning (CFL) [33] has been validated as one of the most promising frameworks that divides all clients into different clusters according to the similarity of their local data distribution. CFL-based approaches indirectly measure the similarity between clients' data distribution, as an example, using the distance between their local model weights [4], [31], [37], [41]. Hence, they still hold the promise of privacy preservation, while obtaining multiple models to achieve higher overall model accuracy.

We assume that $|\mathbb{M}|$ clients are divided into k clusters. i.e., $\mathbb{G} = \{\mathcal{G}_1, \mathcal{G}_2, \ldots, \mathcal{G}_k\}$, and clients belonging to the same cluster will collaboratively train a shared model. The data distribution of clients in the same cluster should be similar, while data of clients from different clusters are still non-IID. We will obtain multiple models, i.e., $\omega_{\mathbb{G}} = \{\omega_1, \omega_2, \ldots, \omega_k\}$.

Unlike traditional FL methods, e.g., *FedAvg* [28], which optimize one single global objective, CFL decomposes the global objective into multiple sub-objectives, and optimizes them simultaneously, i.e.,

$$\min_{\omega_i}\left\{F(\omega_i) \triangleq \sum_{m\in\mathcal{G}_i}\frac{|\mathsf{D}_m|}{\sum_{m\in\mathcal{G}_i}|\mathsf{D}_m|}\cdot F_m(\omega_i),\ \mathcal{G}_i \in \mathbb{G}\right\}. \tag{5}$$

The CFL idea has inspired many research works recently. Existing CFL works can be categorized into two categories, depending on whether the number k of clusters needs to be set in advance. Most of existing CFL works require the number k as an input parameter [4], [6], [7], [41]. In practice, however, we do not have access to the local data of each client, so we cannot set the optimal k without prior knowledge about all clients' data distribution. Although some of them may take several attempts on different k settings to select the most suitable one with the best clustering results, these methods inevitably introduce extra computation and communication costs. Few works [33] of the other category do not input k in advance, but they usually require more communication rounds to achieve stable client clusters.

Motivation 1: Existing CFL methods do not consider both efficiency and adaptability of the client clustering process. Therefore, it stimulates us in designing one brand-new CFL method that can adaptively cluster clients without knowing the number k of clusters in advance, yet efficient on both communications and computations.

3.3 Imbalanced Data Affects Client Clustering

Just as non-IID data is prevalent among real-world data, imbalanced data across clients i.e., varying number of samples among different clients, is quite common in FL [1], [29], [35] as well. Existing CFL works primarily focus on solving the non-IID data challenge, yet largely overlooking the impact of imbalanced data on the client clustering.

Experimental Observations. We conduct a simple experiment to understand the impact of imbalanced data on the CFL methods. Similar to the experimental settings in previous works [33], [39], we make use of FL to construct a simple three-layer convolutional neural network (CNN) for $|\mathbb{M}| = 10$ clients using the CIFAR-10 dataset [15]. Specifically, the CNN model contains two convolutional layers and a fully connected layer. To simulate non-IID data setting, we group 10 clients into two clusters, e.g., $\mathcal{G}_1 = \{1, 2, \ldots, 5\}$ and $\mathcal{G}_2 = \{6, 7, \ldots, 10\}$, according to their local label categories, i.e., $\mathcal{X}_{\mathcal{G}_1} = \mathcal{X}_{\mathcal{G}_2}$ but $\mathcal{Y}_{\mathcal{G}_1} \neq \mathcal{Y}_{\mathcal{G}_2}$. The local data for clients of each

Client #	1	2	3	4	5	6	7	8	9	10
Normal	1400	1530	1530	1400	1280	1400	1530	1530	1400	1280
Imbalanced	380	3100	1150	510	640	380	2680	2200	380	1020

Fig. 1. The number of samples stored in each client in both normal and imbalanced scenario, respectively.

cluster are sampled from CIFAR-10 dataset according to the Dirichlet distribution with its scaling parameter $\alpha = 1$. In addition, we simulate imbalanced data across clients by limiting the number of samples for each client. Fig. 1 shows the number of samples for each client in both normal and imbalanced scenario, respectively.

In both scenarios, we set up the training model with the same parameters, and let the server to perform a total of $T = 10$ global model aggregations. Since model weight distance between any two clients is usually used for guiding client clustering in CFL works [4], [31], [37], [41], we thus calculate the average euclidean distance between different client model weights, saying ω_i and ω_j, using the following equation:

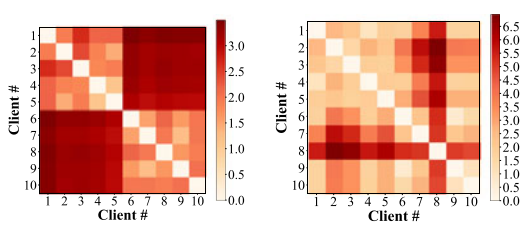
$$dist(\omega_i, \omega_j) = \frac{1}{|\omega|}\sqrt{\sum_{z=1}^{|\omega|}(\omega_{i,z} - \omega_{j,z})}, \tag{6}$$

where $|\omega|$ is the number of weights for the two models.

Figs. 2a and 2b visualize the model weight distance matrices for the normal and imbalanced scenario, respectively. As shown in Fig. 2a, when the number of samples stored in clients are balanced, a clear clustering structure can be observed from the distance matrix of local model weights, which is refereed as *model similarity matrix* in the following. In contrast, when imbalanced data across clients exists, it is difficult to observe the clustering sign from the model similarity matrix as shown in Fig. 2b.

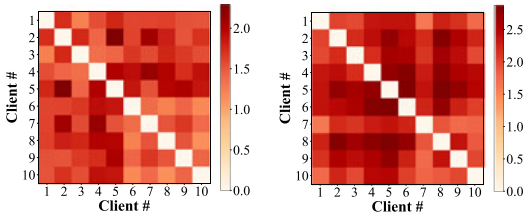
We conduct another experiment, where we divide the 10 clients into two clusters. Within each cluster, the clients share similar label distribution but different feature space, i.e., $\mathcal{X}_{\mathcal{G}_1} \neq \mathcal{X}_{\mathcal{G}_2}$ but $\mathcal{Y}_{\mathcal{G}_1} = \mathcal{Y}_{\mathcal{G}_2}$. Figs. 3a and 3b show the influences of imbalanced data on client clustering when non-IID data are caused by different input spaces of data samples. In the lower right corner of Fig. 3a, we can observe a clear clustering phenomenon, which is not captured in Fig. 3b. In addition to the influence of non-IID data, above experimental results clearly demonstrate that imbalanced data also have a great impact on the client clustering process, possibly leading to improper clusters and degrading the performance of many CFL methods.

We may have such a question: *why and how do imbalanced data influence the client clustering?* The reason may be that



(a) Normal scenario (b) Imbalanced scenario

Fig. 2. The impact of imbalanced data on client clustering under non-IID data caused by *different label spaces*.



(a) Normal scenario (b) Imbalanced scenario

Fig. 3. The impact of imbalanced data on client clustering under non-IID data caused by *different input spaces*.

imbalanced data across clients seriously affects the model weight distances between clients with varied data samples, while many existing CFL methods [4], [31], [37], [41] heavily rely on the model similarity matrix to measure data distribution similarity of clients, resulting in inappropriate client clustering results. Since model distance matrix is affected, imbalanced data would make the local empirical loss inconsistent among the clients, which have similar data distribution and formerly should belong to the same cluster. For the clustering results in Figs. 2a and 2b, we plot and compare the local empirical losses of clients under normal and imbalanced data scenario in Figs. 4a and 4b, respectively. From Fig. 4, we see that imbalanced data cause the local loss differences among clients to be much larger than the scenario when clients' data are balanced.

To further explore the relationship between number of samples and local loss, we conduct an experiment by varying the number of samples for a randomly selected client, and plot the experimental result in Fig. 5. We find that when the sample size is larger than 800, more samples lead to a lower local loss. When insufficient samples (e.g., ≤ 800) are used to train the model, however, we observe anomaly local losses. Meanwhile, when the sample size is larger than a certain number, e.g., 2800 in our experiment, we find that adding more samples leads to slower decrease of the local loss, as shown in Fig. 5. According to experimental results from Figs. 4 and 5, we find that more local samples generally yield lower empirical losses.

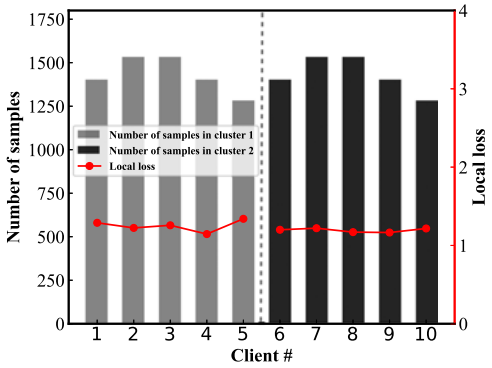
Motivation 2: Existing CFL works have neglected the impact of imbalanced data on the client clustering process, while we observe that there exists loss inconsistency among clients with imbalanced samples. Such an observation inspires us to further optimize CFL by mitigating the impact of imbalanced data through well handling different clients' local empirical losses.

4 Design of *AutoCFL*

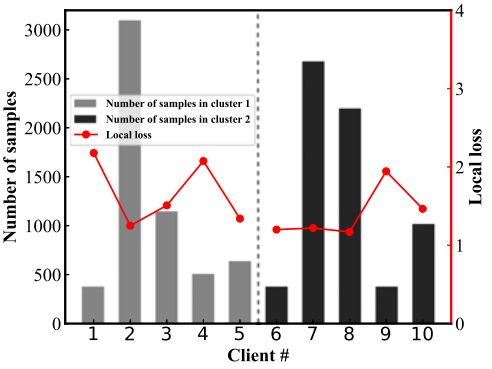
4.1 Overview

To advance existing CFL works, we present *AutoCFL* that can automatically uncover cluster structure among clients from their model weight distances. *AutoCFL* is able to mitigate the effects of both non-IID data and imbalanced data in practical FL applications, and thus is efficient and effective in both aspects of communication and computation.

Fig. 6 illustrates the workflow of *AutoCFL*, which is similar with *FedAvg* [28] and includes three main steps: *(1)* The server broadcasts the global model to the clients; *(2)* Each client trains the model using its local data; and *(3)* The server aggregates these updated local models from clients.



(a) Local losses of clients with balanced data.



(b) Local losses of clients with imbalanced data.

Fig. 4. Local losses of different clients in (a) normal scenario; and (b) imbalanced scenario.

Improved on the typical FL training, *AutoCFL* embeds two key strategies to address the inadequacy of existing CFL methods. First, we propose the *local training adjustment strategy* (in Section 4.2) that adjusts the local training epochs for each client given its data amount. Different clients may go through a varied number of local training epochs, and their resultant local models are used to calculate a model weight distance matrix (i.e., model similarity matrix). Second, an *adaptive client clustering strategy* (in Section 4.3) is proposed to uncover the cluster relationship among clients from the model similarity matrix, and adaptively partition them with similar data distribution into the same cluster even without specifying the number of clusters. A weighted voting mechanism is devised to further eliminate the effect of imbalanced data. Combining these two novel strategies, *AutoCFL* can derive stable client clusters within only a few rounds of communication between clients and the server in an efficient and adaptive manner.

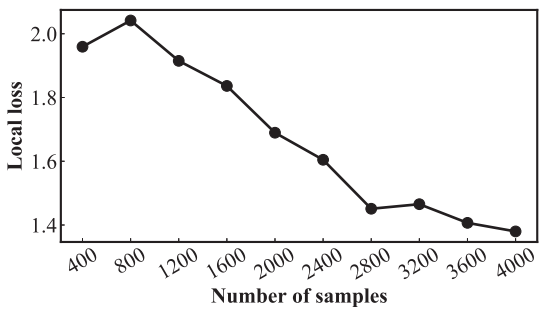


Fig. 5. Effect of the number of samples on the local loss.

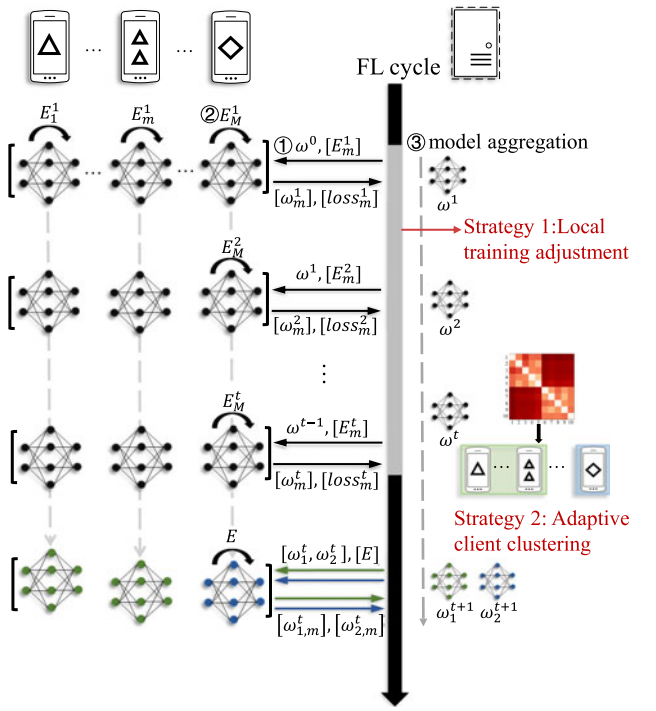


Fig. 6. The workflow of *AutoCFL* with three main steps and two key strategies: (1) The server broadcasts the global model; (2) Each client updates the model with local data samples, enhanced by the *local training adjustment strategy*; (3) The server aggregates updated local models, enhanced by the *adaptive client clustering strategy* for training and aggregating multiple global models for client clusters.

4.2 Local Training Adjustment Strategy

According to our experiments in Section 3.3, we find that there is a close relationship between the number of local samples and empirical loss of local model training. Given the same settings of model training (e.g., the learning rate) and proper training without the overfitting issue, we observe from Figs. 4 and 5 that the clients with more local samples have smaller local empirical loss, while the clients with fewer local samples have larger local empirical loss. Furthermore, the relative relationship between the number of local samples and the local empirical loss can hold during the whole FL process.

Above observation can be true from the aspect of model training. Specifically, the number of iterations N_{iter} in each epoch is an important hyper-parameter that is jointly determined by batch size BS and number of training samples, i.e., $N_{iter} = \frac{|\mathbb{D}_m|}{BS}$. In addition, the total iterations of model training is the product of N_{iter} and the total training epochs. When batch size BS is fixed, the more samples are available for training, the greater the number of iterations N_{iter} will be in a single epoch. In general, more iterations of gradient descent are performed, the closer the model is to the optimum. Thus, we propose a local training adjustment strategy, which lets clients with more samples train less epochs while clients with fewer samples train more epochs, so that to keep empirical losses among clients be consistent.

Typically, the local empirical loss reflects the state of a local model during the FL training process, and is influenced by the number of iterations. Specifically, more iterations tend to make the empirical loss of local model training smaller before model converged. In general, one epoch consists of multiple

iterations traversing through all samples in the training process of machine learning models. Therefore, unlike *FedAvg* [28] that sets the same number of local epochs for all clients, *AutoCFL* dynamically adjusts the number of epochs for each client during local model training to control the client's empirical loss. Noting that the adjusted local epochs for clients are not the same. Instead, more local epochs are set for clients with fewer samples, while a smaller number of local epochs is expected for the clients with more samples. Essentially, we hope that the empirical losses of clients that have similar data distribution should be consistent after such a local training adjustment.

However, making the local empirical loss of clients with varied number of samples be consistent in one single communication round is difficult. This is because when the number of samples stored by a client is small, increasing its local epochs may cause the local empirical loss to be significantly reduced or even over-fitted. Therefore, we turn to make the cumulative loss of a client be more consistent rather than the local empirical loss of one single communication round. A client's cumulative loss reflects the change of its local losses over multiple communication rounds, and it is easier to be controlled compared to the local empirical loss of one single communication round. The cumulative loss of client m after t communication rounds is calculated as:

$$CL_m^t = \sum_{i=1}^{t} loss_m^i, \tag{7}$$

where $loss_m^i$ indicates the local empirical loss of client m at the i-th communication round.

Based on the definition of cumulative loss CL_m^t, *AutoCFL* computes the number of local epochs E_m^t for client m at the t-th communication round as:

$$E_m^t = \begin{cases} E_m^{t-1} + (\alpha \times \frac{|\mathsf{D}_{s^*}|}{|\mathsf{D}_m|})^{\rho} & CL_m^{t-1} > CL_{s^*}^{t-1}, \\ E_m^{t-1} & CL_m^{t-1} \leq CL_{s^*}^{t-1}, \end{cases} \tag{8}$$

where s^* is the client owning the most local samples, $|\mathsf{D}_{s^*}|$ and $|\mathsf{D}_m|$ represent the number of local samples at client s^* and m, respectively. Besides, parameter α denotes the epoch growth factor that controls the epoch growth rate, and parameter ρ is defined as:

$$\rho = \min\left(1, \frac{loss_m^{t-1}}{loss_{s^*}^{t-1}}\right). \tag{9}$$

On one hand, the cumulative loss of the client with more local samples tends to be more reliable. On the other hand, increasing the number of epochs for clients will introduce excessive local computation overheads. Thus we should set the local epochs for all clients within a reasonable range. In practice, we take the cumulative loss of the client (i.e., s^*) with the most local samples as the benchmark that usually has the lowest loss, and make the cumulative loss between clients to be more consistent by controlling the cumulative loss of other clients to move closer to the benchmark. In Equation (8), *AutoCFL* calculates the number of local epochs for each client using both the number of local samples and the cumulative loss. Specifically, the number of local samples determines the maximum step size (i.e., $\frac{|\mathsf{D}_{s^*}|}{|\mathsf{D}_m|}$) of each epoch increase, while the local empirical loss determines the actual increase step size (i.e., ρ) based on the maximum step size. Our adjustment strategy is conservative, as the actual growth step of epoch is decreasing during the local training adjustment process. More local training epochs will lead to a faster decrease of local empirical loss, and thus the ratio, i.e., $\frac{loss_m^{t-1}}{loss_{s^*}^{t-1}}$, between a client's local empirical loss to the benchmark's local empirical loss will gradually decrease.

Adjustment Termination. *AutoCFL* can stop the local training adjustment naturally. After the first communication round, the client with fewer local samples usually trains the local model with more epochs, and therefore owns a faster declining of the local empirical loss. The gap between cumulative loss of such clients and the benchmark client s^* would be narrowing. But once they are smaller than CL_{s^*}, the gap between them and CL_{s^*} would be gradually increasing again because the number of epochs of the clients does not decrease. To sum up, once the variance of clients' cumulative losses is minimized, *AutoCFL* will stop the local training adjustment process, and each client will finish its local model training within current round.

4.3 Adaptive Client Clustering

Different from previous CFL works [4], [6], [7], [27] that need to input the number k of clusters in a prior, *AutoCFL* is able to accomplish the client clustering without knowing k by solely exploiting the model similarity matrix.

The adaptive client clustering design is motivated by an important criterion on evaluating the clustering results, *namely the minimum inter-cluster distance should be not smaller than the maximum intra-cluster distance [8], i.e.,*

$$\min\ dist(\mathcal{G}_{i\cdot}, \mathcal{G}_{j\cdot}) \geq \max\ dist(\mathcal{G}_{i\cdot}, \mathcal{G}_{i\cdot}) \tag{10}$$

where $dist(\mathcal{G}_{i\cdot}, \mathcal{G}_{j\cdot})$ denotes the model distance between any two clients from two different clusters (e.g., $\mathcal{G}_i$ and $\mathcal{G}_j$), and $dist(\mathcal{G}_{i\cdot}, \mathcal{G}_{i\cdot})$ denotes the model distance between any two clients of the same cluster $\mathcal{G}_i$. Such a precondition ensures that all clients can be classified into the suitable clusters. Based on this precondition, AutoCFL can simply search for cluster separability conditions rather than optimizing inter-cluster distance and intra-cluster distance.

Using all weights of a large model to calculate model similarity between clients can result in significant computational overheads. To reduce computation costs for deriving the model similarity matrix, *AutoCFL* calculates the model similarity between any two clients with partial well-selected weights. The selected weights would better reflect the differences between these two models, which is theoretically supported by some previous works. Specifically, [32] and [44] propose that higher-layer weights of the model are more task related than these lower-layer weights. Similarly, [26] reports that when non-IID data exist, the neural network model has greater model differences between weights of the classifier-layers when compared to the models trained on the IID data. Inspired by these works, we choose the weights of the layer closest to the output layer as a representative weight set of all model weights to calculate the similarity of two models. We denote ω as the whole weights of the model, and ω' as the selected partial weights.

AutoCFL groups clients using the model similarity matrix $\mathcal{M}$ only, which is computed from all clients' stable local models after the adjusted local training. Specifically, each element $\mathcal{M}[m,n] = dist(\omega'_m, \omega'_n)$ measures the model distance between any two clients, e.g., m and n, using Equation (6). The m-th row $\mathcal{M}[m,\cdot]$ of matrix $\mathcal{M}$ denotes the model distances between client m's model and all other clients' models. Specifically, $\mathcal{M}[m,m] = 0$, and $\mathcal{M}[m,n] > 0, n \neq m$. Taking $\mathcal{M}$ as the input, *AutoCFL* conducts client clustering following three key steps, i.e., *cluster demarcation detection*, *weighted voting*, and *voting based clustering*, which are detailed as follows.

(1) Cluster demarcation detection. For client m, *AutoCFL* first sorts the model distance values in $\mathcal{M}[m,\cdot]$ in an ascending order to obtain $\mathcal{M}'[m,\cdot]$, where $\mathcal{M}'[m,n] \geq \mathcal{M}'[m,z]$, $n > z$. Then, *AutoCFL* calculates the difference between any two neighboring model distance values stored in $\mathcal{M}'[m,\cdot]$, and gets the maximum distance difference τ_m. Considering the precondition of "good" clustering expressed in Equation (10), we treat τ_m as the demarcation of clustering all clients with client m as the reference. Based on τ_m, all the clients indexed by $\mathcal{M}'[m,\cdot]$ are partitioned into two groups $P_{m,1}$ and $P_{m,2}$. The difference between any two adjacent values in $P_{m,1}$ is smaller than τ_m, while the difference between any two adjacent values in $P_{m,2}$ is not smaller than τ_m.

Intuitively, clients in $P_{m,1}$ could be assigned to the same cluster, while we cannot make the clustering decisions for the clients belonging to $P_{m,2}$, since their model differences are relatively large. In addition, the maximum distance difference τ_m computed for clients with fewer samples is less stable compared to those with more samples, thus we cannot cluster clients solely relying on τ_m. To address these concerns, we propose a weighted voting mechanism to adaptively determine the final client clustering results.

(2) Weighted voting. For each client m and its corresponding $\mathcal{M}'[m,\cdot]$, we find the client $client_{mmax}$ that has the largest number of local samples among clients in $P_{m,1}$:

$$client_{mmax} = \arg\max\{|\mathsf{D}_n|,\ n \in P_{m,1}\}. \tag{11}$$

Then, *AutoCFL* lets each client n in $P_{m,1}$ to vote for $client_{mmax}$ according to Equation (12), and updates the voting score S_n^{mmax} of client n for $client_{mmax}$.

$$S_n^{mmax} = S_n^{mmax} + \frac{|\mathsf{D}_n|}{\sum_{z \in P_{m,1}} |\mathsf{D}_z|}. \tag{12}$$

Each client maintains a list to record its accumulated voting score (e.g., S_n^{mmax}) to other clients (e.g., $client_{mmax}$), which have been selected as the client with the most samples.

In general, we assign voting weights to clients according to their sample sizes. Specifically, more local samples are empowered with a larger voting weight for a given client. The weighted voting could further eliminate the instability of clusters caused by the clients with imbalanced samples, and decrease the chance of improper clustering.

(3) Voting based clustering. After running step (1) and (2) for all clients by traversing all rows of $\mathcal{M}$, we can derive the final voting score list for each client. For a given client m, we assign it to be in the same cluster $\mathcal{G}_*$ as the representative client $client_*$ that has the highest accumulated score in m's voting score list, i.e.,

$$client_* = \arg\max_*(S_m^*). \tag{13}$$

By scanning all clients and their voting score lists, *AutoCFL* automatically selects some representative clients as the cluster heads and assigns other clients to these clusters.

Algorithm 1. Adaptive Client Clustering

Input: Model similarity matrix $\mathcal{M}$
Output: Client clusters $\mathbb{G} = \{\mathcal{G}_1, \mathcal{G}_2, \ldots, \mathcal{G}_k\}$
1: **for** each client m **do**
2: $\quad\mathcal{M}'[m,\cdot] \leftarrow$ Sort $\mathcal{M}[m,\cdot]$ in an ascending order;
3: $\quad$Calculate difference between any two adjacent values in $\mathcal{M}'[m,\cdot]$;
4: $\quad\tau_m \leftarrow \max(\mathcal{M}'[m,n+1] - \mathcal{M}'[m,n])$.
5: $\quad$Partition $\mathcal{M}'[m,\cdot]$ into two groups $P_{m,1}$ and $P_{m,2}$ with τ_m as the demarcation;
6: $\quad$Find $client_{mmax}$ using Eq. (11);
7: $\quad$**for** each client n in $P_{m,1}$ **do**
8: $\quad\quad$Update voting score using Eq. (12);
9: **for** each client m **do**
10: $\quad$Assign m to cluster $\mathcal{G}_*$ based on Eq. (13);

Algorithm 1 presents the pseudocode of our adaptive client clustering. Our strategy only relies on the input of model similarity matrix $\mathcal{M}$, without requiring the input or pre-setting any parameters related to the number of clusters. The algorithm scans the matrix $\mathcal{M}$, and detects the cluster demarcation for each reference client m (line 2-4). Based on the cluster demarcation τ_m, the algorithm partitions all clients into two groups, and executes the weighted voting for clients in group $P_{m,1}$ (line 5-8). After obtaining the voting score lists for all clients, the algorithm automatically assigns clients to the clusters that are led by some representative clients with more samples and higher voting scores (line 9-10). Finally, the client clustering result $\mathbb{G}$ is outputted. It is worth noting that our client clustering strategy does not break the privacy-preserving promise. The server only needs to know the number of local samples per client, which is the same as the requirement of *FedAvg* [28].

4.4 Analysis and Discussion

We will analyze the computation and communication efficiency of *AutoCFL*, and then discuss the privacy-preserving property of *AutoCFL*.

Computation Efficiency. We analyze the time complexity of *AutoCFL*, which mainly includes two parts, i.e., model similarity calculation and client clustering process. (1) The complexity of model similarity calculation using all model weights is $\mathcal{O}(|\mathbb{M}|^2|\omega|^2)$, while complexity of model similarity calculation using partial selected weights is $\mathcal{O}(|\mathbb{M}|^2|\omega'|^2)$, where $|\mathbb{M}|$ denotes the total number of clients. Since $|\omega'|$ is much smaller than $|\omega|$, $\mathcal{O}(|\mathbb{M}|^2|\omega'|^2)$ is thus much smaller than $\mathcal{O}(|\mathbb{M}|^2|\omega|^2)$. (2) *AutoCFL* adaptively clusters clients based on their model similarity values, as shown in Algorithm 1. The time complexity of Algorithm 1 is dominated by sorting model similarity values for each row of model similarity matrix $\mathcal{M}$. For each client, the time complexity of model similarity sorting is $\mathcal{O}(|\mathbb{M}|\log|\mathbb{M}|)$, while the sorting

will be executed $|\mathbb{M}|$ times in total. Therefore, the time complexity of Algorithm 1 is $\mathcal{O}(|\mathbb{M}|^2 \log |\mathbb{M}|)$. As a comparison, the time complexity of hierarchical agglomerative clustering algorithm, which has been widely used by previous methods [4] for client clustering in FL, is $\mathcal{O}(|\mathbb{M}|^3)$. Thus, our client clustering is much efficient.

In summary, the overall time complexity of our *AutoCFL* is $\mathcal{O}(|\mathbb{M}|^2|\omega'|^2 + |\mathbb{M}|^2 \log |\mathbb{M}|)$.

Communication Efficiency. Existing CFL methods (e.g., [33]) adopt iterative clustering to continuously bi-partition client clusters. However, global convergence is required before each bi-partitioning of clusters, and thus these methods will incur a large number of communication rounds [41]. In contrary, our *AutoCFL* uses a one-pass clustering strategy, and combines with the local training adjustment strategy to derive more robust and stable local models. Therefore, *AutoCFL* can complete client clustering within a few communication rounds, without requiring global convergences. The communication efficiency of *AutoCFL* is also validated by experiments in Section 5.2.

Privacy. AutoCFL requires each client to report the number of local samples for efficient client clustering. Specifically, the number of local samples is leveraged by *AutoCFL* to calculate a specific adjusted number of local epochs for each client, and determine the weight for each client in the weighted voting process. Except for the number of client's local samples, *AutoCFL* does not require any other information. In fact, the most famous FL method, i.e., *FedAvg* [28], also holds such an assumption that the participating clients need to report the number of local samples to determine the weight of each client at the model aggregation phase. Many other existing FL methods [21], [29], [38] that are inspired by *FedAvg* also require such sample numbers from clients to support proper operations. Therefore, *AutoCFL* is similar as other FL methods, and requires the least necessary information from clients. This assumption is rational and feasible, and thus *AutoCFL* can achieve the same privacy-preserving property as *FedAvg*-like methods.

5 Performance Evaluation

5.1 Experimental Setup

Baseline Methods. We compare our proposed *AutoCFL* with the other four baseline methods:

- *FedAvg* [28] is the most common and fundamental algorithm in FL, which aggregates local models using a weighted averaging method based on the number of clients' local samples. Assuming the IID data distribution among clients, *FedAvg* coordinates all clients to train one global model.
- *IFCA* [7] assigns a client m to the cluster, where m's local model will achieve the smallest local empirical loss. For better performance, *IFCA* has to know the number k of clusters in advance, and executes the client clustering process in an iterative manner.
- *MTCFL* [33] computes the cosine similarity between updates of local models for indirectly measuring the data distribution similarity among clients, and bi-partitions the clusters, whose global models have been converged. *MTCFL* can adaptively group clients into clusters as it works without setting the number k of clusters, while it will introduce a large amount of communication costs.
- *FL+HC* [4] extracts the client cluster structure from the model similarity matrix using a hierarchical clustering algorithm. *FL+HC* does not require to know the number k of clusters in advance, but the hierarchical clustering algorithm needs to set the model distance threshold, which implicitly requires prior knowledge about clients' data distribution.

Noting that *FedAvg* will train only one global model for all clients, while our method *AutoCFL* and the other three baseline methods will train multiple models, i.e., one global model for each cluster, to attack the non-IID data challenge.

Datasets and Models. We consider three different image classification tasks that are accomplished on three popular benchmark datasets, i.e., *CIFAR-10* [15], *MNIST* [17], and *FashionMNIST* [40], respectively. For each task and dataset, we build one convolutional neural network (CNN) model, each of which consists of a feature extractor and a classifier. The three models share the basic structure that contains a total of three layers, with two 5×5 convolutional layers forming the feature extractor, and one fully connected layer acting as the classifier. Each convolutional layer is followed by a 2×2 max pooling layer. The three models differ in the number of output channels for the two convolutional layers. In our experiments, we set the numbers of output channels for MNIST model as 20 and 50, the CIFAR-10 model as 6 and 16, and the FashionMNIST model as 16 and 32.

FL Simulation Setup. To simulate the heterogeneous and imbalanced data distribution among clients, we set up four different data environments that are made of different data distribution and sample sizes: (1) IID and balanced data (*IID-Balanced*); (2) IID and imbalanced data (*IID-Imbalanced*); (3) Non-IID and balanced data (*Non-IID-Balanced*); and (4) Non-IID and Imbalanced data (*Non-IID-Imbalanced*). Specifically, we generate the data distribution and sample sizes for clients according to the following rules.

- For the IID data setting, we randomly select samples from the complete dataset for each client and the proportions of samples from different categories are consistent across all clients.
- For the non-IID setting, since the joint distribution, i.e., $p(x, y) = p(y)p(x|y)$, of $x \in \mathcal{X}$ and $y \in \mathcal{Y}$ is determined by $p(y)$ and $p(x|y)$ together, we thus adjust the local samples of the clients from either input space $\mathcal{X}$ or label space $\mathcal{Y}$. Specifically, we change the clients' samples in the input space $\mathcal{X}$ by rotating each sample (i.e., *Non-IID-Input*), while changing the label space $\mathcal{Y}$ by assigning samples with inconsistent label categories to the clients (i.e., *Non-IID-Label*).
- For the balanced data setting, we make sure that the numbers of local samples in all clients are consistent or approximate.
- For the imbalanced data setting, we randomly adjust the number of samples from each category to generate imbalanced samples among clients.

Therefore, we can obtain six combinatorial data settings, i.e., *IID-Imbalanced* (short for *IID-Im.*), *IID-Balanced* (short for *IID-B.*), *Non-IID-Input-Imbalanced* (short for *Non-IID-In.-Im.*),

TABLE 1
Model Accuracy (% $\pm$ Std) of Each Method over CIFAR10, MNIST, FashionMNIST

Dataset	Method	Data Setting					
		IID-Im.	IID-B.	Non-IID-In.-Im.	Non-IID-In.-B.	Non-IID-La.-Im.	Non-IID-La.-B.
CIFAR10	*FedAvg*	44.68 ± 0.36	48.88 ± 0.19	34.54 ± 1.51	39.11 ± 0.88	39.89 ± 1.29	43.47 ± 0.67
	IFCA	44.64 ± 0.49	49.78 ± 0.36	35.58 ± 0.73	41.47 ± 0.52	66.02 ± 0.73	67.86 ± 0.83
	MTCFL	46.83 ± 0.87	$\mathbf{50.98 \pm 0.74}$	38.74 ± 2.13	44.46 ± 1.59	57.68 ± 2.74	68.37 ± 1.42
	FL+HC	43.71 ± 1.23	48.96 ± 0.96	35.88 ± 1.76	40.02 ± 1.03	60.85 ± 0.80	74.61 ± 0.25
	AutoCFL	$\mathbf{47.23 \pm 0.35}$	50.81 ± 0.14	$\mathbf{39.21 \pm 1.32}$	$\mathbf{44.94 \pm 0.64}$	$\mathbf{69.13 \pm 0.27}$	$\mathbf{75.73 \pm 0.21}$
MNIST	*FedAvg*	95.18 ± 0.42	96.82 ± 0.27	91.49 ± 0.31	91.59 ± 0.17	96.14 ± 0.42	96.28 ± 0.28
	IFCA	$\mathbf{95.72 \pm 0.26}$	97.22 ± 0.17	95.22 ± 0.70	96.01 ± 0.54	98.35 ± 0.36	$\mathbf{98.94 \pm 0.37}$
	MTCFL	94.74 ± 0.74	96.52 ± 0.68	95.36 ± 0.97	95.57 ± 1.26	97.88 ± 1.05	98.86 ± 1.13
	FL+HC	94.75 ± 0.32	96.87 ± 0.30	91.85 ± 1.03	92.33 ± 0.87	93.42 ± 1.13	98.91 ± 0.32
	AutoCFL	95.64 ± 0.23	$\mathbf{97.50 \pm 0.19}$	$\mathbf{96.77 \pm 0.54}$	$\mathbf{96.86 \pm 0.23}$	$\mathbf{98.48 \pm 0.39}$	98.93 ± 0.17
FMNIST	*FedAvg*	84.26 ± 0.21	86.11 ± 0.12	82.52 ± 0.57	79.85 ± 0.27	82.10 ± 0.63	81.72 ± 0.59
	IFCA	84.73 ± 0.32	86.68 ± 0.23	84.91 ± 0.71	85.92 ± 0.51	90.12 ± 1.22	89.34 ± 0.93
	MTCFL	82.52 ± 0.63	84.19 ± 0.76	81.82 ± 2.49	84.61 ± 1.33	90.39 ± 0.78	91.25 ± 0.55
	FL+HC	84.49 ± 0.43	86.49 ± 0.58	83.60 ± 2.05	85.69 ± 1.48	86.35 ± 1.82	94.17 ± 0.34
	AutoCFL	$\mathbf{85.45 \pm 0.27}$	$\mathbf{86.85 \pm 0.24}$	$\mathbf{85.08 \pm 0.86}$	$\mathbf{86.49 \pm 0.69}$	$\mathbf{95.09 \pm 0.47}$	$\mathbf{94.79 \pm 0.27}$

Non-IID-Input-Balanced (short for *Non-IID-In.-B.*), *Non-IID-Label-Imbalanced* (short for *Non-IID-La.-Im.*), and *Non-IID-Label-Balanced* (short for *Non-IID-La.-B.*).

Implementation. We simulate a FL system with $|\mathbb{M}| = 20$ clients. To implement non-IID data setting with different label spaces, we divided all clients equally into four groups similar to [33], and the local label categories owned by the four groups are $0-2$, $3-6$, $4-9$, and $0-9$, respectively. Such a data division contains three relationships: non-intersection, intersection and inclusion. In addition, we let all clients have samples of the same label categories, and make the input space $\mathcal{X}$ different among them by selecting a fraction of clients and rotating their samples. For the datasets of MNIST and FashionMNIST, we rotate half of the clients' samples by $180°$, and we equally divide all clients into four clusters. For the CIFAR-10 dataset, we rotate the samples of the four clusters with angles of $0°$, $90°$, $180°$, and $270°$, respectively. For the imbalanced data setting, we randomly select 9 out of the 20 clients and adjust the sample size of each three clients to 10%, 30%, and 60% of the original sample size, which is the unified size of all clients in the balanced data setting.

We implement our *AutoCFL* and the four baseline methods using Pytorch.[1] For the three CNN models, we set the learning rate η and batch size as 0.01 and 128, respectively. For *AutoCFL*, we set $\alpha = 0.5$ by default in the local training adjustment. *AutoCFL* will stop the local training adjustment to avoid wasting communication resources when the variance of the cumulative losses of the first five rounds shows an increasing trend. For the baseline method *FL+HC*, we set it to cluster clients in the fifth communication round. For all other methods, we adopt the optimal settings specified in their corresponding papers, respectively. We conduct all experiments on a server, which is equipped with an NVIDIA 2080Ti GPU, AMD 3800X CPU and 64G RAM.

1. Pytorch: https://pytorch.org/

5.2 Performance Comparisons

In this section, we compare the performance of our *AutoCFL* with the four baseline methods on the performance metrics of *model accuracy*, *communication efficiency*, and *clustering correctness*. In addition to above experimental setup, we simulate the FL training with totally $T = 100$ communication rounds between the server and clients. For each experiment setting, we report the average results of five runs.

Model Accuracy. We compare *AutoCFL* with four baseline methods on four datasets under various data settings. Table 1 presents the overall model accuracy (with standard deviation) comparison results, where we find that *AutoCFL* achieves the highest model accuracy in most data settings, e.g., 15 out of 18 cases. Without any prior knowledge about clients' data distribution, *AutoCFL* improves the model accuracy of these state-of-the-art methods by $0.13\% - 4.70\%$. In addition, we find that *AutoCFL* owns the smallest standard deviation in the majority cases, which implies that *AutoCFL* can obtain more stable clustering results.

To verify whether *AutoCFL* can improve model accuracy of clients with fewer samples, we conduct an experiment under Non-IID-La.Im. data setting over CIFAR10. In this experiment, we compare the performances of client's local training method (*Local* for short), *FedAvg*, and *AutoCFL*. In particular, client's local training means that training the local model using client's own data separately, rather than in the FL paradigm. Table 2 shows the model accuracy results for clients that have samples fewer than the average (i.e., 1260). In general, more samples usually lead to a higher model accuracy for the three methods. The average model accuracy results of all clients are 62.40%($\pm$11.19%), 43.23%($\pm$9.39%), and 72.05%($\pm$5.86%) for *Local*, *FedAvg*, and *AutoCFL*, respectively. Table 2 demonstrates that *AutoCFL* can help clients with fewer samples achieve higher model accuracy, which is comparable with the average accuracy of all clients. In contrary, accuracy gap between clients with more samples and the ones with fewer samples is quite large for *FedAvg*.

In addition to the comprehensive model accuracy comparisons, we also plot one example running result for all

TABLE 2
The Model Accuracy Results for Clients That Have Samples Fewer than the Average of All Clients

Sample size	300	100	700	100	600	300	300	400	400	300	1260
Local	75.16	63.33	44.02	47.92	52.98	43.02	55.22	58.37	54.73	59.22	62.40 ± 11.19
FedAvg	39.49	42.1	27.78	41.91	40.93	43.42	56.13	38.52	43.94	45.23	43.23 ± 9.39
AutoCFL	77.22	77.29	65.51	65.42	62.39	74.29	69.86	77.22	74.23	74.61	72.05 ± 5.86

The number of the first row indicates the sample size of each client that has insufficient data, while the last number is the average sample size of all clients.

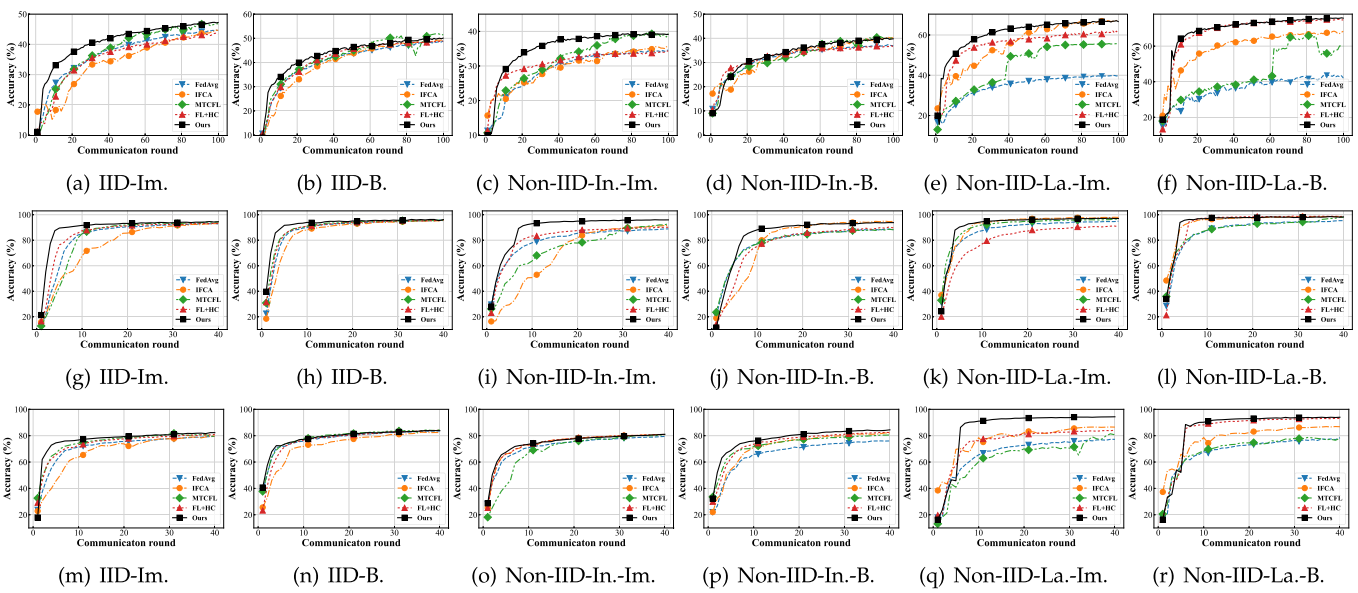
methods for each data setting in Fig. 7. This figure compares the *time-accuracy* performance of all methods. We find that *AutoCFL* can achieve the highest model accuracy within the fewest communication rounds.

Communication Efficiency and Clustering Correctness. During the FL training, we also record the number of communication rounds required by each method to obtain stable clusters, i.e., when client clusters do not change any more and the model accuracy of each cluster becomes converged. We analyze the clustering correctness of each method for IID and non-IID data settings separately. Specifically, for the non-IID data setting, we record the number of clients that have been correctly clustered. For the IID data setting, we only record the number of final clusters, because all clients under this setting should be grouped into the same cluster.

Table 3 shows the overall comparison results on performance metrics of communication efficiency and clustering correctness. Noting that we do not record the number of communication rounds for *FL+HC*, because it is configured to complete the client clustering process in the fifth communication round. In general, our method *AutoCFL* can achieve the best clustering correctness yet with the fewest communication rounds, i.e., the highest communication efficiency.

On the one hand, our method *AutoCFL* always completes the client clustering with the least number of communication rounds under various data settings (see the first number of each result in Table 3). *IFCA* can complete the client clustering with sub-optimal communication efficiency because of its extra information about the number of clusters. *MTCFL* requires much more communication rounds to achieve the final clustering results, because it cannot wisely control the clustering process due to its poor stopping condition setting. At the same time, *MTCFL* often gets the improper client clusters, when compared to the ground truth of client clusters. Furthermore, combined with the model accuracy results of *MTCFL* in Fig. 7, we see that improper clusters also lead to unstable FL training.

On the other hand, *AutoCFL* achieves the best clustering correctness in most cases (see the second number of each result in Table 3), except for the Non-IID-Input-Balanced data setting over MNIST dataset, where *AutoCFL* has assigned one client to the wrong cluster. Compared with the imbalanced data settings, the four methods have better clustering correctness on the balanced data settings in almost all cases. Such results further validate our observations in Section 3.3 that imbalanced data affect the correctness of client clustering. For the IID data settings, since all clients are supposed to be grouped into the same cluster, we record the final number of clusters for each method to compare the clustering ability. Our *AutoCFL* can always group all clients into one cluster, and the same for *IFCA* that knows the actual cluster number $k = 1$. The other two methods (i.e., *MTCFL* and *FL+HC*), however, mistakenly group clients into multiple clusters. For the non-IID data settings, *AutoCFL* also has the best clustering correctness among all methods by correctly assigning most clients into the right clusters. Even knowing the actual cluster number, *IFCA* still fails to correctly group clients into the suitable clusters in some cases, e.g., in the *Non-IID-Input-Imbalanced* data setting. In contrast, the other two methods have the worst results, as shown in Table 3.



(a) IID-Im. (b) IID-B. (c) Non-IID-In.-Im. (d) Non-IID-In.-B. (e) Non-IID-La.-Im. (f) Non-IID-La.-B.

(g) IID-Im. (h) IID-B. (i) Non-IID-In.-Im. (j) Non-IID-In.-B. (k) Non-IID-La.-Im. (l) Non-IID-La.-B.

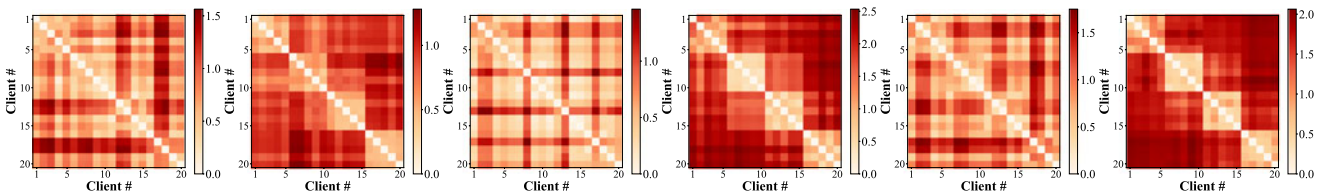
(m) IID-Im. (n) IID-B. (o) Non-IID-In.-Im. (p) Non-IID-In.-B. (q) Non-IID-La.-Im. (r) Non-IID-La.-B.

Fig. 7. Time-Accuracy performance of each method over CIFAR10, MNIST and FashionMNIST. The abbreviations of IID-Im., IID-B., Non-IID-In.-Im., Non-IID-In.-B., Non-IID-La.-Im. and Non-IID-La.-B. denote the six different data settings as specified in Section 5.1. Fig. 7(a)-(f) show the results over CIFAR10, Fig. 7(g)-7(l) show the results over MNIST, and Fig. 7(m)-(r) show the results over FashionMNIST.

TABLE 3
Comparisons on the Communication Efficiency and Clustering Correctness among Different Methods

Dataset	Method	Data Setting					
		IID-Im.	IID-B.	Non-IID-In.-Im.	Non-IID-In.-B.	Non-IID-La.-Im.	Non-IID-La.-B.
CIFAR10	*IFCA*	23 / **1**	7 / **1**	27 / 10	28 / 15	11 / **20**	6 / **20**
	MTCFL	72 / 6	73 / 4	63 / 14	55 / 15	53 / 12	27 / 15
	FL+HC	- / 2	- / **1**	- / 11	- / 12	- / 14	- / **20**
	AutoCFL	**5** / **1**	**5** / **1**	**3** / **15**	**5** / **17**	**5** / **20**	**4** / **20**
MNIST	*IFCA*	19 / **1**	6 / **1**	12 / 18	7 / **20**	7 / **20**	5 / **20**
	MTCFL	39 / 6	81 / 6	33 / 18	55 / 19	32 / **20**	47 / **20**
	FL+HC	- / 2	- / **1**	- / 11	- / 12	- / 11	- / **20**
	AutoCFL	**5** / **1**	**5** / **1**	**4** / **20**	**3** / **20**	**3** / **20**	**3** / **20**
FMNIST	*IFCA*	17 / **1**	6 / **1**	7 / **20**	**5** / **20**	10 / 15	7 / 16
	MTCFL	42 / 7	90 / 7	46 / 11	47 / 17	46 / 16	65 / 15
	FL+HC	- / **1**	- / **1**	- / 18	- / **20**	- / 11	- / **20**
	AutoCFL	**5** / **1**	**5** / **1**	**3** / **20**	**5** / **20**	**6** / **20**	**5** / **20**

For each result a/b, a indicates the number of communication rounds required by each method for clustering, while b denotes the number of correctly clustered clients or the final number of clusters. The best result is marked in bold.



(a) MNIST-W.O. (b) MNIST-W. (c) CIFAR10-W.O. (d) CIFAR10-W. (e) FMNIST-W.O. (f) FMNIST-W.

Fig. 8. Visualization of the resultant model similarity matrix of *AutoCFL* over different datasets. Here, the suffix *W.O.* and *W.* represent without and with the local training adjustment strategy, respectively.

According to the results in Table 1, Table 3 and Fig. 7, our *AutoCFL* outperforms the state-of-the-art methods on various performance metrics under non-IID and imbalanced data settings, e.g., on average reducing communication costs by $4.64\times$, while improving model accuracy by 9.24% compared to *FedAvg* [28], the standard FL method. Therefore, *AutoCFL* can eliminate the impacts of both non-IID and imbalanced data challenges, and achieves adaptive and correct client clustering with no prior knowledge about clients' raw data. Even under the non-IID and balanced data settings, *AutoCFL* still outperforms the state-of-the-art methods, e.g., averagely reducing $4.28\times$ communication costs with improved model accuracy. In summary, *AutoCFL* outperforms the state-of-the-art methods under all data settings.

5.3 Effectiveness of Local Training Adjustment

Contrast Experiments. We conduct experiments in the Non-IID-Label-Imbalanced setting to evaluate the effectiveness of the local training adjustment strategy. We further combine this strategy with *FL+HC* to explore its generality.

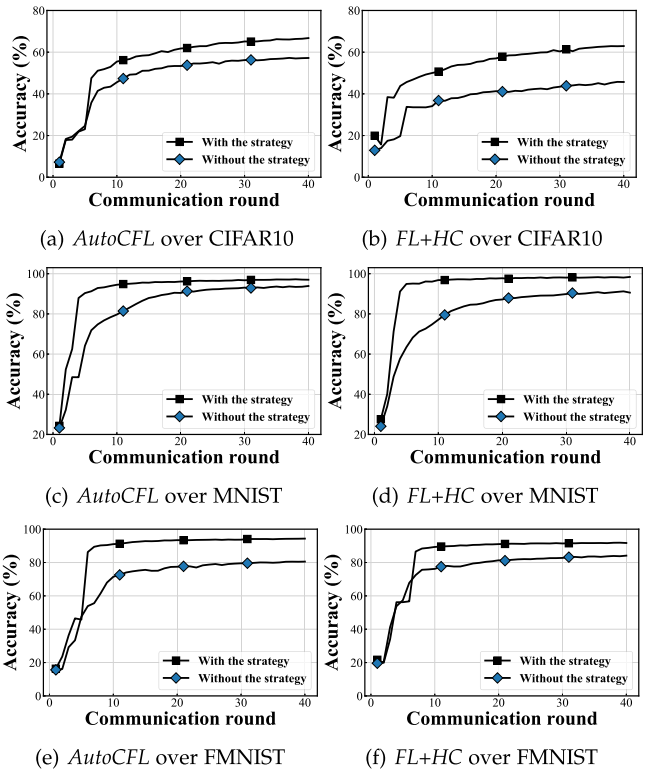
Fig. 9 shows the time-accuracy performance of our method *AutoCFL* and *FL+HC* in both with and without the dynamic adjustment strategy scenarios. When both methods enable the adjustment strategy, their final model accuracy results are much higher than that without such a strategy. This is because our local training adjustment strategy can eliminate the impact of imbalanced data and enhance the clustering signs. Fig. 8 visualizes the model similarity matrix for our method with and without the strategy on the three datasets. Obviously, we see that local training adjustment strategy is very helpful to restore the original model similarity matrix that is affected by imbalanced data. The experiment results in Figs. 8 and 9 demonstrate the effectiveness of our local training adjustment strategy, and suggest that our strategy can also improve existing CFL methods with a great generality.

Impact of Epoch Growth Factor α. We study the impact of parameter α in our local training adjustment strategy. We run *AutoCFL* for FL training with 10 communication rounds under Non-IID-Label-Imbalanced data setting over the CIFAR10 dataset by varying α. Our strategy will terminate the strategy when we observe the elbow point on the variance of cumulative loss. Fig. 10 shows that different α leads to a slight change in the optimal stopping time at which point the model similarity matrix for the best clustering results is obtained. In general, a larger α leads to an earlier stopping time, while a smaller α will postpone the timing.

5.4 Effectiveness of Adaptive Client Clustering

Contrast Experiments. To evaluate the effectiveness of our adaptive client clustering strategy, we compare the clustering results of our strategy with other classic clustering methods, i.e., K-means [8], HAC [11], and DBSCAN [2], based on the same model similarity matrix shown in Fig. 8d, which is derived by *AutoCFL* under the Non-IID-Label-Imbalanced data setting over the CIFAR10 dataset.

However, these compared methods require to set parameters related to the number of clusters, we thus configure



(a) *AutoCFL* over CIFAR10 (b) *FL+HC* over CIFAR10

(c) *AutoCFL* over MNIST (d) *FL+HC* over MNIST

(e) *AutoCFL* over FMNIST (f) *FL+HC* over FMNIST

Fig. 9. Comparison on the time-accuracy performance of *AutoCFL* and *FL+HC* over different datasets.

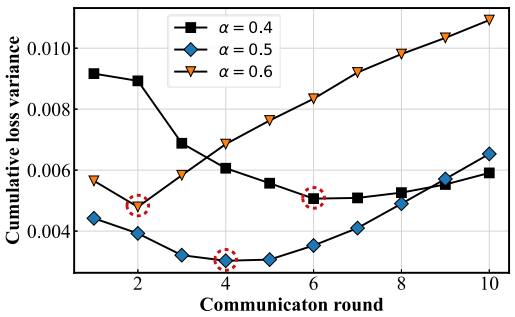


Fig. 10. The changes of cumulative loss variance under different α, where the point in the red dashed circle indicates the optimal stopping time to obtain the model similarity matrix for the best client clustering.

them with the optimal values. Table 4 compares the final clustering results of all methods. The final client clusters derived by our adaptive clustering strategy are the same as those classic clustering methods, only the order of forming clusters differs. This experiment result demonstrates that when the inputted model similarity matrix is the same, our strategy can produce the same and correct client clusters as classic methods do, even without setting any parameters.

Impact of Weighted Voting. We propose the weighted voting mechanism to further eliminate the impact of imbalanced data and improve the clustering correctness. To evaluate the effectiveness of weighted voting, we compare the clustering results with *weighted voting* and *unweighted voting*, i.e., setting the same weights for all clients, under all data settings over the three datasets.

Table 5 shows the comparison results, where we see that the clustering results with weighted voting are generally better than those with unweighted voting. In particular, the number of correctly clustered clients for unweighted clustering is almost half of that for weighted clustering under the Non-IID-Input data settings. This is because unweighted voting based clustering ignores the impact of imbalanced data, which leads to improper clustering results. The results imply that our weighted voting can overcome the problem of unstable inter-model distances caused by unbalanced local samples, and thus obtains better clustering results.

TABLE 4
Comparison on the Final Client Clusters Derived by Different Clustering Methods, Given the Input Model Similarity Matrix Shown in Fig. 8d.

Method	Cluster 1	Cluster 2	Cluster 3	Cluster 4
AutoCFL	{1, 2, 3, 4, 5}	{6, 7, 8, 9, 10}	{11, 12, 13, 14, 15}	{16, 17, 18, 19, 20}
K-means	{16, 17, 18, 19, 20}	{11, 12, 13, 14, 15}	{6, 7, 8, 9, 10}	{1, 2, 3, 4, 5}
HAC	{6, 7, 8, 9, 10}	{16, 17, 18, 19, 20}	{11, 12, 13, 14, 15}	{1, 2, 3, 4, 5}
DBSCAN	{1, 2, 3, 4, 5}	{6, 7, 8, 9, 10}	{11, 12, 13, 14, 15}	{16, 17, 18, 19, 20}

For each result a/b, a indicates the number of communication rounds required by each method for clustering, while b denotes the number of correctly clustered clients or the final number of clusters. The best result is marked in bold.

TABLE 5
Comparison on the Clustering Correctness of Weighted Voting and Unweighted Voting

Data setting	Voting policy	Dataset		
		MNIST	CIFAR10	FMNIST
IID-Im.	*Weighted*	**1**	**1**	**1**
	Unweighted	**1**	**1**	**1**
IID-B.	*Weighted*	**1**	**1**	**1**
	Unweighted	2	**1**	**1**
Non-IID-In.-Im.	*Weighted*	**20**	**15**	**20**
	Unweighted	10	10	10
Non-IID-In.-B.	*Weighted*	**19**	**10**	**20**
	Unweighted	10	**10**	10
Non-IID-La.-Im.	*Weighted*	**20**	**20**	**20**
	Unweighted	**20**	19	**20**
Non-IID-La.-B.	*Weighted*	**20**	**20**	**20**
	Unweighted	**20**	**20**	**20**

The final number of clusters for IID data settings and the number of correctly clustered clients for non-IID data settings are listed, respectively. The best result is marked in bold.

6 Conclusion

In this paper, we present *AutoCFL* to advance existing CFL researches. The design of *AutoCFL* is driven by two major observations. First, most existing CFL works heavily rely on the prior knowledge of the number of clusters, and fail to achieve efficient and adaptive client clustering. Second, existing CFL works have largely neglected the impact of imbalanced data on client clustering, while we experimentally observe that imbalanced samples across clients cause the local empirical loss inconsistent between clients even with similar data distribution. Therefore, we propose two novel strategies to attack the non-IID data and imbalanced data challenge simultaneously. Specifically, the local training adjustment strategy can dynamically adjust the local epochs for each client to eliminate the impact of imbalanced data. *AutoCFL* further exploits the weighted voting based

adaptive client clustering strategy to automatically extract the cluster structure among clients from the model similarity matrix, getting ride of pre-setting any parameters related to the cluster number. Extensive experiments over three popular datasets under various data settings have demonstrated the effectiveness and efficiency of *AutoCFL*.

REFERENCES

[1] A. A. Abdellatif *et al.*, "Communication-efficient hierarchical federated learning for IoT heterogeneous systems with imbalanced data," *Future Gener. Comput. Syst.*, vol. 128, pp. 406–419, 2022.

[2] D. Birant and A. Kut, "ST-DBSCAN: An algorithm for clustering spatial–temporal data," *Data Knowl. Eng.*, vol. 60, no. 1, pp. 208–221, 2007.

[3] K. A. Bonawitz *et al.*, "Towards federated learning at scale: System design," in *Proc. Mach. Learn. Syst. Conf.*, 2019, pp. 374–388.

[4] C. Briggs, Z. Fan, and P. Andras, "Federated learning with hierarchical clustering of local updates to improve training on non-IID data," in *Proc. Int. Joint Conf. Neural Netw.*, 2020, pp. 1–9.

[5] Y. J. Cho, J. Wang, and G. Joshi, "Client selection in federated learning: Convergence analysis and power-of-choice selection strategies," 2020, *arXiv:2010.01243*.

[6] M. Duan *et al.*, "FedGroup: Ternary cosine similarity-based clustered federated learning framework toward high accuracy in heterogeneous data," 2020, *arXiv:2010.06870*.

[7] A. Ghosh, J. Chung, D. Yin, and K. Ramchandran, "An efficient framework for clustered federated learning," in *Proc. Adv. Neural Informat. Process. Syst.*, 2020, pp. 19586–19597.

[8] J. A. Hartigan and M. A. Wong, "Algorithm as 136: A k-means clustering algorithm," *J. Roy. Statist. Soc.: Ser. C (Appl. Statist.)*, vol. 28, no. 1, pp. 100–108, 1979.

[9] D. Jiang *et al.*, "A GDPR-compliant ecosystem for speech recognition with transfer, federated, and evolutionary learning," *ACM Trans. Intell. Syst. Technol.*, vol. 12, no. 3, pp. 1–19, 2021.

[10] Y. Jiang, J. Konečnỳ, K. Rush, and S. Kannan, "Improving federated learning personalization via model agnostic meta learning," 2019, *arXiv:1909.12488*.

[11] S. C. Johnson, "Hierarchical clustering schemes," *Psychometrika*, vol. 32, no. 3, pp. 241–254, 1967.

[12] P. Kairouz *et al.*, "Advances and open problems in federated learning," *Found. Trends Mach. Learn.*, vol. 14, no. 1–2, pp. 1–210, 2021.

[13] S. P. Karimireddy, S. Kale, M. Mohri, S. Reddi, S. Stich, and A. T. Suresh, "Scaffold: Stochastic controlled averaging for federated learning," in *Proc. Int. Conf. Mach. Learn.*, 2020, pp. 5132–5143.

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

[15] A. Krizhevsky *et al.*, "Learning multiple layers of features from tiny images," Tech. Rep. TR-2009, 2009.

[16] F. Lai, X. Zhu, H. V. Madhyastha, and M. Chowdhury, "Oort: Efficient federated learning via guided participant selection," in *Proc. USENIX Symp. Oper. Syst. Des. Implementation*, 2021, pp. 19–35.

[17] 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.

[18] Q. Li, Y. Diao, Q. Chen, and B. He, "Federated learning on non-IID data silos: An experimental study," 2021, *arXiv:2102.02079*.

[19] 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.

[20] T. Li, A. K. Sahu, M. Zaheer, M. Sanjabi, A. Talwalkar, and V. Smith, "Federated optimization in heterogeneous networks," in *Proc. Mach. Learn. Syst. Conf.*, 2020, pp. 429–450.

[21] T. Li, M. Sanjabi, A. Beirami, and V. Smith, "Fair resource allocation in federated learning," in *Proc. Int. Conf. Learn. Representations*, 2020, pp. 1–25.

[22] 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.

[23] B. Liu, Y. Guo, and X. Chen, "PFA: Privacy-preserving federated adaptation for effective model personalization," in *Proc. Web Conf.*, 2021, pp. 923–934.

[24] F. Liu, X. Wu, S. Ge, W. Fan, and Y. Zou, "Federated learning for vision-and-language grounding problems," in *Proc. AAAI Conf. Artif. Intell.*, 2020, pp. 11572–11579.

[25] Y. Liu *et al.*, "Fedvision: An online visual object detection platform powered by federated learning," in *Proc. AAAI Conf. Artif. Intell.*, 2020, pp. 13172–13179.

[26] M. Luo, F. Chen, D. Hu, Y. Zhang, J. Liang, and J. Feng, "No fear of heterogeneity: Classifier calibration for federated learning with non-IID data," in *Proc. Adv. Neural Informat. Process. Syst.*, 2021, pp. 5972–5984.

[27] Y. Mansour, M. Mohri, J. Ro, and A. T. Suresh, "Three approaches for personalization with applications to federated learning," 2020, *arXiv:2002.10619*.

[28] B. McMahan, E. Moore, D. Ramage, S. Hampson, and B. A. Y Arcas, "Communication-efficient learning of deep networks from decentralized data," in *Proc. Artif. Intell. Statist.*, 2017, pp. 1273–1282.

[29] U. Michieli and M. Ozay, "Are all users treated fairly in federated learning systems?," in *Proc. IEEE/CVF Conf. Comput. Vis. Pattern Recognit.*, 2021, pp. 2318–2322.

[30] T. Nishio and R. Yonetani, "Client selection for federated learning with heterogeneous resources in mobile edge," in *Proc. IEEE Int. Conf. Commun.*, 2019, pp. 1–7.

[31] X. Ouyang, Z. Xie, J. Zhou, J. Huang, and G. Xing, "ClusterFL: A similarity-aware federated learning system for human activity recognition," in *Proc. Annu. Int. Conf. Mobile Syst. Appl. Serv.*, 2021, pp. 54–66.

[32] A. Rozantsev, M. Salzmann, and P. Fua, "Beyond sharing weights for deep domain adaptation," *IEEE Trans. Pattern Anal. Mach. Intell.*, vol. 41, no. 4, pp. 801–814, Apr. 2019.

[33] 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.

[34] F. Sattler, K.-R. Müller, T. Wiegand, and W. Samek, "On the byzantine robustness of clustered federated learning," in *Proc. IEEE Int. Conf. Acoust. Speech Signal Process.*, 2020, pp. 8861–8865.

[35] V. Smith, C.-K. Chiang, M. Sanjabi, and A. Talwalkar, "Federated multi-task learning," 2017, *arXiv:1705.10467*.

[36] A. Z. Tan, H. Yu, L. Cui, and Q. Yang, "Towards personalized federated learning," 2021, *arXiv:2103.00710*.

[37] L. Tu, X. Ouyang, J. Zhou, Y. He, and G. Xing, "FedDL: Federated learning via dynamic layer sharing for human activity recognition," in *Proc. ACM Conf. Embedded Netw.ed Sensor Syst.*, 2021, pp. 15–28.

[38] G. Wang, C. X. Dang, and Z. Zhou, "Measure contribution of participants in federated learning," in *Proc. IEEE Int. Conf. Big Data*, 2019, pp. 2597–2604.

[39] 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.

[40] H. Xiao, K. Rasul, and R. Vollgraf, "Fashion-MNIST: A novel image dataset for benchmarking machine learning algorithms," 2017, *arXiv:1708.07747*.

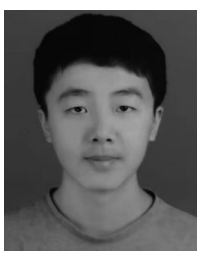
[41] M. Xie, G. Long, T. Shen, T. Zhou, X. Wang, J. Jiang, and C. Zhang, "Multi-center federated learning," 2021, *arXiv:2108.08647*.

[42] Q. Yang, Y. Liu, Y. Cheng, Y. Kang, T. Chen, and H. Yu, "Federated learning," *Synth. Lectures Artif. Intell. Mach. Learn.*, vol. 13, no. 3, pp. 1–207, 2019.

[43] T. Yang *et al.*, "Applied federated learning: improving google keyboard query suggestions," 2018, *arXiv:1812.02903*.

[44] J. Yosinski, J. Clune, Y. Bengio, and H. Lipson, "How transferable are features in deep neural networks?," in *Proc. Adv. Neural Informat. Process. Syst.*, 2014, pp. 3320–3328.

[45] Y. Zhao, M. Li, L. Lai, N. Suda, D. Civin, and V. Chandra, "Federated learning with non-IID data," 2018, *arXiv:1806.00582*.



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, Xi'an China. His research interests include machine learning and federated learning.

Tianzhang Xing (Member, IEEE) received the BE degree in telecommunications engineering from Xidian University, Xi'an, Chian, in 2004, the MPhil and PhD degrees in computer science and technology from the Northwest University, Xi'an, China, in 2009 and 2014, respectively. 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. In 2017, he joined Shenzhen University, Shenzhen, China, as an assistant professor. His research interests include distributed sensing and mobile computing, Big Data analytic, Internet of Things, and urban computing. He is a member of CCF, ACM.

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 (Member, IEEE) 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.