A Remedy for Heterogeneous Data: Clustered Federated Learning with Gradient Trajectory

***Abstract*—Federated Learning (FL) has recently attracted a lot of attention due to its ability to train a machine learning model using data from multiple clients without divulging their privacy. However, the training data across clients can be very heterogeneous in terms of quality, amount, occurrences of specific features, etc. In this paper, we demonstrate how the server can observe data heterogeneity by mining gradient trajectories that the clients compute from a two-dimensional mapping of high-dimensional gradients computed by each client from its bottom layer. Based on these ideas, we propose a new Clustered Federated Learning method called CFLGT, which dynamically clusters clients together based on the gradient trajectories. We analyze CFLGT both theoretically and experimentally to show that it overcomes several drawbacks of mainstream Clustered Federated Learning methods and outperforms other baselines.**

***Index Terms*—Federated Learning, Clustering, Heterogeneous Data, Distributed System**

I. Introduction

The key driver of sophisticated deep learning is the availability of vast amounts of data; however, the data is often produced at many different locations and typically owned by different parties. It is generally not possible to collect all the required data in a central place for deep learning both because of the difficulty of sending and centrally storing the data and because the parties that generate them are unlikely to share it freely [1]. Federated Learning (FL) was proposed to solve this problem [2], where clients in FL keep their data locally. In the typical federated learning framework (FedAvg) [2], the server broadcasts a global model to several clients, and each client trains it with their data and uploads the model to the server. After receiving all updated models, the server aggregates them into a new global model for the next round of training. The server can thus utilize the data owned by different clients for training without the clients having to disclose the data.

But FL still faces a host of challenges. One is training with heterogeneous data [3] since the data from different clients could have varying quality, size, features, etc. This can cause several problems in FL, such as bad model performance [4], [5], slow or unstable convergence in training [6], and so on. Several methods have been proposed to reduce the harmful effects from heterogeneous data. References [7]–[9] optimize FL with heterogeneous data via meta-learning, which empowers the client model to learn new data quickly. [10]–[13] propose personalized federated learning, where clients adjust the global model to get a unique model suitable for local data. The contribution of different level layers has also been analyzed recently [12], [14]–[16]. In addition, clustering-based federated learning (CFL) is implemented to mitigate the negative effects of heterogeneous data. The server collects information from all active clients in the FL system and iteratively (or recursively) assigns clients into clusters, where every client in the same cluster has similar data distribution. Thus the non-IID data among all clients can be transformed into IID data in every client cluster, which diminishes the negative impacts from the data distribution. Despite the benefit, several drawbacks are associated with the CFL: 1) how to estimate the number of clusters, 2) how do we ensure that the data transfer bandwidth between clients and the server is well constrained, 3) how do we ensure that the computing requirements of the client-server coordination are low, and 4) how do we ensure that the additional data exchange does not leak client's data.

To address these challenges in CFL, our exploration focused on extracting information from high-level layers on the client side and applying it as the input of the server for client clustering. Recent research in FL with non-IID data has highlighted the significance of high-level layers. It explicitly or implicitly emphasizes leveraging bottom-layer characteristics to enhance FL performance with heterogeneous data. Inspired by this research, our approach involves extracting information from high-level layers, transmitting it to the server, and optimizing the CFL from an optimal global perspective.

In our work, the gradient information of the bottom layer in the client-side model is the input of clustered-based Federated Learning. We call it a gradient trajectory, an innovative two-dimensional vector based on gradient decomposition. The gradient trajectory contains client data distribution information with no privacy concerns and huge computation costs. Thus the server can achieve accurate client clustering efficiently. Specifically, focused on a classification task with softmax layer and cross-entropy loss function, inspired by [15], we decompose the bottom layer gradient into two vectors, pulling force and pushing force. We show that the pulling forces and pushing forces are directly related to dataset distribution. Then we extract distribution information in these two forces to get the gradient trajectory of clients. The trajectories are composed of points connected in two dimensions, produced by two forces [15]. With clients' privacy guaranteed, we explain why the server can mitigate heterogeneity in the system with gradient trajectories.

We thus propose a novel Clustered Federated Learning framework called Clustered Federated Learning with Gradient Trajectory (CFLGT) to solve the heterogeneous data problem in FL, which clusters clients accurately and swiftly. The main contributions can be summarized as follows:

- We deep dive into the underlying gradients and present the first gradient trajectory for representing data heterogeneity.
- We theoretically prove the mathematical meaning of the gradient trajectory and explore its potential as a clustering object.

- Based on the study of gradient trajectories, we propose the new FL algorithm, and the following experiments show that our method mitigates the negative effects of the above drawbacks achieves excellent performance with a small cost, and is practical in real-world tasks.

The rest of the paper is organized as follows. Section II introduces recent research on FL with heterogeneous data and clustered federated learning. Section III goes through basic concepts in Federated Learning, illustrates how the gradient trajectory is created, and then observes the data heterogeneity through the gradient trajectories of clients. Section IV describes the proposed algorithm framework, a new CFL based on gradient trajectories, and the mathematical proof of its enhancement of the new framework. In Section V, the baselines and experimental results are presented. Finally, Section VI concludes the whole paper and proposes future development direction.

II. RELATED WORK

A. FL with heterogeneous Data

Researchers have proposed several methods to handle the heterogeneity problem in FL. Li *et al.* proposed the FedProx [17], which adds a regularization term to the loss function to prevent the locally trained model from deviating too much from the global model. The studies in [7]–[9] optimize the performance of FL with heterogeneous data by leveraging the fast learning properties of meta-learning for new tasks. Researchers in [10]–[12] proposed personalized federated learning to optimize the performance of federated learning by training personalized models for different clients. Also, several researchers have recently investigated the impacts of high-level layers on FL with heterogeneous data. Sun *et al.* [12] demonstrated that aggregating too many high-level layers will hurt FL performance with heterogeneous data. The experiments in [14] exhibit that the highest disparity occurs between bottom layers of neural networks in FL with heterogeneous data. Li *et al.* [15] accounted for the influence on the bottom layer in a label-shift condition while clients in FedRep [16] train and keep their high-level layers locally, thus protecting their model from the pollution of heterogeneous data. These works show that high-level layers of a neural network are more sensitive to heterogeneous data in FL than other layers and affect greater model performance because of the closer contact with common features [18].

B. Clustered Federated Learning

Clustered Federated Learning (CFL) [19] techniques can control the aggregation of clients' models by clustering clients into groups. However, the existing works still face quite a few problems.

1) **Data privacy issue**: In the CFL algorithms [19]–[21], clients need to upload the gradient or model to the server. If attackers capture the information, they can complete an inference attack. Zhu *et al.* [22] demonstrated the possibility of obtaining raw data by intercepting the uploaded gradients and successfully implementing the data-privacy-level attack. In addition, Melis *et al.* [23] proposed an attack on the client's data privacy by obtaining the model update. This attribute inference attack uses the updated model to infer sensitive attributes of the training data. As a result, uploading a model or gradient for clustering is dangerous to clients in the FL system.
2) **The number of clusters needs to be determined in advance**: The popular clustered federated learning algorithms need the number of clusters in advance as the algorithm's input. However, it is difficult to determine a reasonable number of clusters on the server side in a real scenario, considering the huge scale of participating clients and data privacy constraints. Additionally, a wrong cluster number may damage the FL performance, which is shown in our experiments.
3) **Ignoring the limitation of bandwidth and client's computing power**: When the number of clusters is large and the model structure is complex, IFCA [24] needs to broadcast all clusters' global models to clients in each round, which requires considerable pressure on the bandwidth and client's computing resources. Moreover, Gholizadeh *et al.* [25] proposed an algorithm that uses model hyper-parameters for clustering. The client uses grid search to find the optimal hyper-parameters (including the number of neurons and epochs) and uploads them to the server side. When using large models, grid search will consume computational resources greatly, even exceeding the client's limitation.

III. FORMULATION OF GRADIENT TRAJECTORY

In this section, we show how a server explores the bottom layer's gradient information. First, we introduce the basic concepts of Federated Learning (FL) and properties of the bottom layer with softmax (classifier). Then we transform the gradient into gradient trajectories in subsection III-B. Thus, the server can receive the trajectories from clients with privacy guarantees and observe the data heterogeneity in the FL system by mining trajectories simultaneously.

A. Basic concepts

Basic Concepts of Federated Learning: In a typical FL system with a classification task, there is one machine learning agent in the "server" and M machine learning agents in clients. The server broadcasts global model h_0 to the participating clients and receives clients' model $h_m (m \in [1, M])$ from the clients in each round. Each client owns a dataset D_m containing N_m samples for local training, denoted as $(\boldsymbol{x}_{m,i}, y_{m,i})_{i=1}^{N_m}$, where i indicates a sample in D_m. In the t-th round of learning, the server randomly selects a subset of clients U^t among M clients and broadcasts the current global model h_g^{t-1} to them. Once receiving the h_g^{t-1}, each client m in U^t updates it by performing local training with data D_m with the objective:

$$\min_{h_m^t} \mathbb{E}_{(\boldsymbol{x},y)\sim D_m} \left[F\left(h; \boldsymbol{x}, y\right) \right], \tag{1}$$

where F indicates the loss function. After local training, the client uploads its new model h_m^t to the server. When all models

are received, the server aggregates them together as a new global model h_g^t as follows,

$$h_g^t = \frac{1}{\sum_{m \epsilon U^t} |D_m|} \sum_{(m \epsilon U^t)} |D_m| \, h_m^t. \tag{2}$$

Properties of Softmax Layer: Like [14], [15], we decompose a neural network into a feature extractor and classifier. The classifier represents the bottom layers with weights in a network, and the rest of the layers are feature extractors. After the i-th sample enters the neural network, the feature extractor outputs the extracted feature vector, denoted as $\boldsymbol{v}_i \in \mathbb{R}^d$. For a C-classes classification task, the weights in the classifier can be written as $\boldsymbol{W} = \{\boldsymbol{w}_c\}_{c=1}^{C} \in \mathbb{R}^{d \times C}$ (the bias in the classifier is omitted). Then the classifier receives the $\boldsymbol{v}_i$ and outputs the probability vector of the i-th sample by $\boldsymbol{P} = \mathrm{Softmax}(\boldsymbol{W}^T \boldsymbol{v}_i)$. Specifically, the probability that the i-th sample belongs to the c-th class can be written as:

$$P_{i,c} = \frac{exp(\boldsymbol{w}_c^T \boldsymbol{v}_i)}{\sum_{j=1}^{C} exp(\boldsymbol{w}_j^T \boldsymbol{v}_i)}. \tag{3}$$

As cross-entropy loss is implemented, the loss of client m is denoted as

$$\mathcal{L}_m = \sum_{i=1}^{|D_m|} \sum_{c=1}^{C} \mathcal{I}\{y_i = c\} \log(P_{i,c}), \tag{4}$$

where $\mathcal{I}\{\cdot\}$ is the indicator function. The gradient of c-th weights is calculated as

$$\boldsymbol{g}_{\boldsymbol{w}_c} = -\sum_{i=1}^{|D_m|} (\mathcal{I}\{y_i = c\} - P_{i,c}) \boldsymbol{v}_i, \tag{5}$$

which equals to

$$\boldsymbol{g}_{\boldsymbol{w}_c} = -\sum_{i=1, y_i = c}^{|D_m|} (1 - P_{i,c}) \boldsymbol{v}_i + \sum_{i=1, y_i \neq c}^{|D_m|} (P_{i,c}) \boldsymbol{v}_i. \tag{6}$$

Borrowing from previous works [15], [26]–[28], we call the first term as pulling force $\boldsymbol{Z}_1 = \sum_{i=1, y_i = c}^{|D_m|} (1 - P_{i,c}) \boldsymbol{v}_i$, consisting of extracted feature from positive samples ($y_i = c$). For the second term, which consists of extracted features from negative samples ($y_i \neq c$), we call it pushing force $\boldsymbol{Z}_2 = \sum_{i=1, y_i \neq c}^{|D_m|} (P_{i,c}) \boldsymbol{v}_i$. An example of this decomposition is shown in Fig. 1. In summary, the gradient of the softmax layer can be divided into pulling force and pushing force, which has a practical meaning in mathematics.

B. From Gradient to Trajectory

In a FL system with heterogeneous data, obtaining the distribution of clients' datasets is beneficial to optimize the FL system. However, uploading class statistics of the client dataset may cause client-level label leakage [29], thus it is not wise for the client to send dataset statistics to the server. Thanks to the mechanism of FL, every client who receives the same global model in each round can compute its pulling force and pushing force by Eq. (6) on local data and then upload them. In this way, the server can be aware of the system's heterogeneity by analyzing the values of pulling force and pushing force.

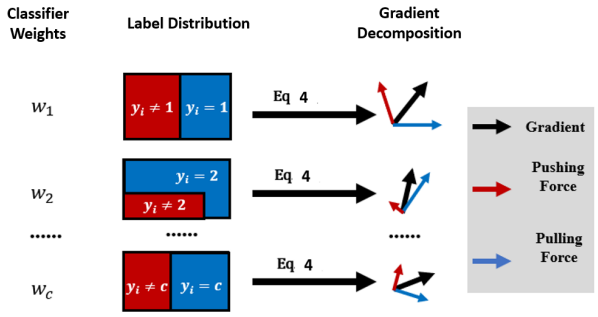

Fig. 1. Classifier Weights Gradient Composition. In a c-th classes classification task, the gradient of each weight in the classifier can be divided into pulling force and pushing force by data labeled as c and labeled not as c.

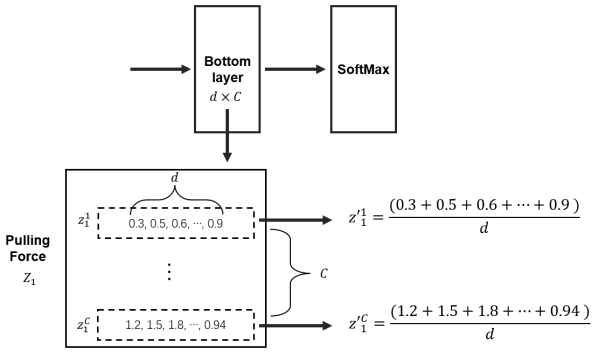

Fig. 2. The computation of $Z_i'^j$. The figure shows a toy example of $Z_1'^j$ formation. The bottom layer is shaped as $d \times C$, thus pushing force $\boldsymbol{Z}_1$ has the same shape. $\boldsymbol{Z}_1$ has C vectors as z_i^j, which contains d values. For every z_i^j, we average it to get $Z_i'^j$.

Yet the strategy poses a huge risk of a data privacy breach. Note that an attacker can restore the gradient of the bottom layer by simply combining pulling force with pushing force. As a result, attacks on data privacy in [30], [31] can be implemented by capturing the pulling force and pushing force.

Therefore, we propose a method to transform the pulling force and pushing force into a gradient trajectory, uploaded to a server with privacy protection. Gradient trajectory is a computed projection of pulling force and pushing force into a two-dimensional space.

To be specific, after receiving the global model, the client computes pushing forces $\boldsymbol{Z}_1$ and pulling forces $\boldsymbol{Z}_2$ by Eq. (7).

$$(\boldsymbol{Z}_1, \boldsymbol{Z}_2) = \left(\left[z_1^1, z_1^2, \cdots, z_1^C \right], \left[z_2^1, z_2^2, \cdots, z_2^C \right] \right), \tag{7}$$

where C represents the number of classes in the task, and $[z_i^j], i \in [1, 2], j \in [1, \cdots, C]$ is a vector shaped as $[d, 1]$, where d is the shape of classifier input (also is the output of feature extractor). Then we compute $[\boldsymbol{Z}_1'^j]$ and $[\boldsymbol{Z}_2'^j], j \in [1, \cdots, C]$ by averaging corresponding vector (z_1^j or z_2^j) in the force. Fig. 2 displays an example. With this method, every vector in $\boldsymbol{Z}_1$ and $\boldsymbol{Z}_2$ is averaged. Lastly, $[\boldsymbol{Z}_1'^j]$ and $[\boldsymbol{Z}_2'^j]$ are sequentially grouped as $(\boldsymbol{Z}_1', \boldsymbol{Z}_2')$, which constitutes the gradient trajectory by Eq. (8).

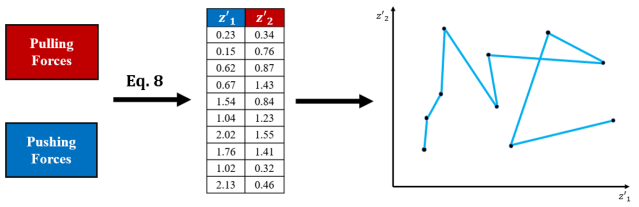

Fig. 3. The formation of gradient trajectory. In a 10-th classes classification task, a client calculates its pulling force and pushing force, gets $\boldsymbol{Z}'_1$ and $\boldsymbol{Z}'_2$, and plots the trajectory in a two-dimensional space, where x-axis is for $\boldsymbol{Z}'_1$ and y-axis is for $\boldsymbol{Z}'_2$.

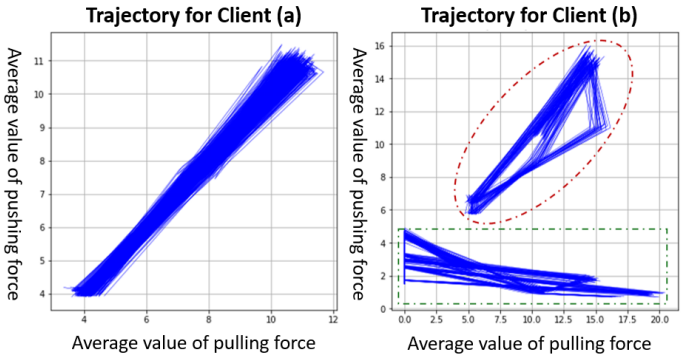

Fig. 4. Gradient Trajectories of all Clients in an FL System. The trajectories in (a) are from IID data and trajectories in (b) are from heterogeneous data.

$$\begin{aligned}\boldsymbol{Z}'_1 &= \left[Z'^1_1, Z'^2_1, \cdots, Z'^C_1\right]^T = \left[avg(\boldsymbol{z}^1_1), \cdots, avg(\boldsymbol{z}^C_1)\right]^T, \\ \boldsymbol{Z}'_2 &= \left[Z'^1_2, Z'^2_2, \cdots, Z'^C_2\right]^T = \left[avg(\boldsymbol{z}^1_2), \cdots, avg(\boldsymbol{z}^C_2)\right]^T.\end{aligned} \tag{8}$$

Because of the above transformation, $\boldsymbol{Z'}_1$ and $\boldsymbol{Z'}_2$ inherit mathematical meaning from $\boldsymbol{Z}_1$ and $\boldsymbol{Z}_2$, that is the norm of these two high dimensional vectors. Note that $[\boldsymbol{z}^j_i] \geq 0$ because $P_{i,c}, (1 - P_{i,c}) \geq 0$ and $\boldsymbol{v}_i$ is the output of a activation (e.g. sigmoid, relu), which cannot be negative value. Therefore, $[Z'^j_i]$ equal to ratio between L1 norm of $[\boldsymbol{z}^j_i]$ and their size. Because the size is constant, decided by the model in the FL system. We can see that $\boldsymbol{Z}'_1$ and $\boldsymbol{Z}'_2$ composed of $[Z'^j_i]$, are considered as the deformation of L1 paradigm, which represents the values of $\boldsymbol{Z}_1$ and $\boldsymbol{Z}_2$. In this way, our transform method passes the mathematical significance of pulling force and pushing force to $\boldsymbol{Z}'_1$ and $\boldsymbol{Z}'_2$.

To display the gradient trajectory in the plane, a client puts $\boldsymbol{Z}'_1$ and $\boldsymbol{Z}'_2$ into the two-dimensional space, gets C points, and then gets its gradient trajectory by connecting C points in a certain order, which given by the server. The order is randomly computed by the server and applied to all clients' $\boldsymbol{Z}'_1$ and $\boldsymbol{Z}'_2$ for trajectory visualization. Fig. 3 displays a gradient trajectory of a client in the FL system.

C. Data Heterogeneity Observation by Gradient Trajectory

In this part, we set up an FL system based on FedAvg [1] to illustrate the analysis of data heterogeneity with gradient trajectories from clients. During training rounds, all clients in the system receive a new global model, compute the pulling force and pushing force, and upload their own gradient trajectories. From the server's perspective, the data heterogeneity in the FL system has become gradient trajectories from all clients. To gain a deep insight into data heterogeneity, the server can observe the gradient trajectories by qualitative and quantitative analysis. For better comparison, we set up two data distributions in the system, IID data and heterogeneous data.

Qualitative Analysis: The server collects all gradient trajectories and visualizes them to observe data heterogeneity. As the left chart in Fig. 4 shows, all trajectories in the system with IID data have the same shape, meaning that data distribution across clients is the same too. In the right chart, there are clear distinctions between many trajectories. The trajectories in the red circle have a clear difference from the trajectories in the green square, reflecting the diversity across the clients' dataset. Therefore, the trajectory visualization provides a clear picture of the degree of heterogeneity for the server.

Quantitative analysis: In addition to observing heterogeneity at the visualization level, the server also needs specific values to evaluate data heterogeneity. Here server computes the coefficient of variation c_v [32] of $\boldsymbol{Z}'_1$ and $\boldsymbol{Z}'_2$ to evaluate heterogeneity in a FL system. The c_v is defined as the ratio of the standard deviation σ to the mean μ. The larger it is, the deviation of data is. It shows that for trajectories in Fig. 4(a), the c_v of the x-axis and y-axis are 0.407 and 0.408. For trajectories in Fig. 4(b), the c_v of the x-axis and y-axis are 0.881 and 1.397. The result implies that the coordinates in the right plot in Fig. 4 are more discrete and the points of the trajectory are more sparsely distributed. Therefore, c_v could assist the server in the FL system in calculating the degree of heterogeneity in the FL system.

IV. METHOD

A. Motivation

Though the server can observe heterogeneity with gradient trajectories, it still needs a method that utilizes the information in gradient trajectories to optimize the FL system. Suppose that a server in an FL system with label distribution shift [33] knows roughly the dataset distribution among clients, so it can aggregate clients with similar dataset distribution to improve FL performance. Clustered Federated Learning [19] is an ideal framework where the server clusters clients into groups and executes model aggregation in every cluster. It is possible for servers who obtain all clients' gradient trajectories to cluster clients accurately in the FL system, as the gradient trajectory contains important information, which is illustrated in Section IV-C.

B. Cluster clients in FL

We introduce our method, Clustered Federated Learning by Gradient Trajectory (CFLGT), where the server collects gradient trajectories of all clients and clusters them into groups, so each group can have its global model achieving a better performance. The algorithm is presented in Algorithm

1. The algorithm first runs for K rounds of FedAvg as pre-training. After finishing pre-training, the server broadcasts the current global model h_g to every client in the FL system. Once receiving h_g, the client calculates the pulling force $\boldsymbol{Z}_1$ and pushing force $\boldsymbol{Z}_2$ on local data by Eq. (6) and then gets average values of pulling force and pushing force separately $\boldsymbol{Z}_1', \boldsymbol{Z}_2'$ by Eq. (8). Note that $\boldsymbol{Z}_1', \boldsymbol{Z}_2'$ are the x- and y-axis coordinates of gradient trajectory in the two- dimensional space. $\boldsymbol{Z}_1', \boldsymbol{Z}_2'$ of every client will be uploaded to the server, which uses gradient trajectories ($\boldsymbol{Z}_1', \boldsymbol{Z}_2'$ of clients) as input of Affinity Propagation to cluster clients into groups. Specifically, the input of Affinity Propagation is the distance between two trajectories, calculated by averaging the L2 distance between corresponding points, such as $(Z_1'^C, Z_2'^C)$ of two trajectories.

Once client clusters are defined, the algorithm runs T rounds of Clustered Federated Learning. In every round, the server randomly selects a subset of clients U^t. The client m chooses its group's model $h_{g,\overline{c}}$ as initialization, updates it to $h_{c,\overline{c}}^m$ where c is short for client, and sends it to the server. The server aggregates clients in the same group to update a new $h_{g,\overline{c}}$ for the next round of training.

C. Information in Gradient Trajectory

Why is gradient trajectory the key in our method? The answer lies inside the information in the gradient trajectory. Since the gradient trajectory represents the pulling force $\boldsymbol{Z}_1$ and pushing force $\boldsymbol{Z}_2$, and then figuring out what information the $\boldsymbol{Z}_1$ and $\boldsymbol{Z}_2$ carry can help us understand the information gradient trajectory carries.

Information of dataset: According to Eq. (6), it is clear that the value of $\boldsymbol{Z}_1$ and $\boldsymbol{Z}_2$ is directly correlated to the label distribution of the dataset. As a result, reconstructed by $\boldsymbol{Z}_1$ and $\boldsymbol{Z}_2$, the gradient trajectory also has information on dataset distribution.

Information of model update: In this part, we theoretically prove that the gradient norm is greatly affected by $\boldsymbol{Z}_1$ and $\boldsymbol{Z}_2$. According to [34], the $L2$ norm of gradient in l-th layer for i-th sample $(\boldsymbol{x}_i, y_i)$ can be written as:

$$\left\|\nabla_{\boldsymbol{w}_l} F(h; \boldsymbol{x}_i, y_i)\right\|_2 \leq \rho \left\|\varpi' diag((\boldsymbol{w}_1^T \boldsymbol{v}_i), \cdots, (\boldsymbol{w}_C^T \boldsymbol{v}_i)) \nabla_{\boldsymbol{P}_i} F\right\|_2, \tag{9}$$

where F is loss function, h is model, ϖ is the softmax function, $\{\boldsymbol{w}_j\}_{j=1}^C$ is the weights of bottom layer (classifier), C is the classes of task, $\boldsymbol{v}_i$ is the output of feature extractor of i-th sample, ρ is the coefficient determined by model structure, and $\boldsymbol{P}_i$ is the probability vector of i-th sample (the output of softmax layer).

Since the loss function of i-th sample$(\boldsymbol{x}_i, y_i)$ is represented as

$$F = -\sum_{j=1}^{C} y_{i,j} log P_{i,j}, \tag{10}$$

where $y_{i,j} \in \{0, 1\}$ and $P_{i,j}$ is a non-zero probability value, we can get

$$\left\|\nabla F(h; \boldsymbol{x}_i, y_i)\right\|_2 \leq \rho \left\|\varpi' diag((\boldsymbol{w}_1^T \boldsymbol{v}_i), \cdots, (\boldsymbol{w}_C^T \boldsymbol{v}_i))(-\frac{\lg e}{P_{i,k}})\right\|_2. \tag{11}$$

Algorithm 1 CFLGT

Require: learning rate η, initial model h, set of clients M, pre-training rounds K, FL rounds T, local training epoch r.

Server: $h_g \leftarrow h$
for $t = 0, 1, 2, ..., K-1$ **do**
  $h_g \leftarrow FedAvg(h_g, M)$
end for
for $m \in M$ in parallel **do**
  get ${}^m\boldsymbol{Z}_1$, ${}^m\boldsymbol{Z}_2$ by Eq. (6)
  get ${}^m\boldsymbol{Z}_1'$, ${}^m\boldsymbol{Z}_2'$ by Eq. (8)
  send ${}^m\boldsymbol{Z}_1'$, ${}^m\boldsymbol{Z}_2'$ to server
end for
In server:
$C, [h_{g,c}]_{c=1}^C \leftarrow Affinity Propagation([{}^i\boldsymbol{Z}_1', {}^i\boldsymbol{Z}_2']_{i=1}^M)$
for $t = 0, 1, 2, ..., T-1$ **do**
  $U^t \leftarrow$ random subset of clients M
  for $\overline{c} \in C$ **do**
    for $c \in U^t$ in parallel **do**
      if $c \in \overline{c}$ **then**
        $h_c \leftarrow h_{g,\overline{c}}$
        $h_c \leftarrow$ LocalUpdate(h_c)
        client sends $\boldsymbol{h}_c$ to server
      end if
    end for
    $\boldsymbol{h}_{g,\overline{c}} \leftarrow \frac{1}{\sum_{m \in U^t \& m \in \overline{c}} N_m} \sum_{(m \in U^t \& m \in \overline{c})} h_{\overline{c},m}$.
  end for
end for
return $[h_{g,\overline{c}}]_{\overline{c}=1}^C$

LocalUpdate(h_c, r, η) for m-th client
for $q = 0, ..., r-1$ **do**
  gradient descent $h_{c,m}' = h_{c,m} - \eta \hat{\nabla} F(h_{c,m}, D_m)$
end for
return $h_{c,m}'$

Note that k is the only class satisfying $y_{i,k} = 1$. The right term of Eq. (11) can be written as

$$\begin{aligned} &\rho^0 \left\|diag(\frac{1}{P_{i,k}}\varpi'(\boldsymbol{w}_1^T \boldsymbol{v}_i), \cdots, \frac{1}{P_{i,k}}\varpi'(\boldsymbol{w}_C^T \boldsymbol{v}_i))\right\|_2 \\ &= \rho^0 \left\|diag(\beta_1^1, \cdots, \beta_k^2, \cdots, \beta_C^1)\right\|_2. \end{aligned} \tag{12}$$

For $\beta_j^1 (j \neq k)$, we get

$$\begin{aligned} \beta^1 &= \frac{1}{P_{i,k}(\sum_j^C \exp(\boldsymbol{w}_j^T \boldsymbol{v}_i))} \frac{\exp(\boldsymbol{w}_j^T \boldsymbol{v}_i) \exp(\boldsymbol{w}_j^T \boldsymbol{v}_i)}{\sum_j^C \exp(\boldsymbol{w}_j^T \boldsymbol{v}_i)} \\ &= -\frac{1}{P_{i,k}(\sum_j^C \exp(\boldsymbol{w}_j^T \boldsymbol{v}_i))} \sigma_0 \\ &\leq -\frac{\sigma_0}{\sum_j^C [P_{i,k}(1 + (\boldsymbol{w}_j^T \boldsymbol{v}_i))]} \\ &\leq -\frac{\sigma_0}{\sum_j^C [P_{i,k} + (\boldsymbol{w}_j^T P_{i,k} \boldsymbol{v}_i))]}. \end{aligned} \tag{13}$$

Recall that the pulling force $\boldsymbol{Z}_{1,i}$ of a sample can be

represented as $P_{i,k}\boldsymbol{v}_i$, so we get

$$\beta^1 \leq B^1 = -\frac{\sigma}{\sum_j^C [P_{i,k} + (\boldsymbol{w}_j^T \boldsymbol{Z}_{1,i}))]}. \tag{14}$$

For $\beta_j^2 (j = k)$, we get

$$\begin{aligned}
\beta^2 &= \frac{exp(\boldsymbol{w}_j^T \boldsymbol{v}_i)(\sum_j^C exp(\boldsymbol{w}_j^T \boldsymbol{v}_i) - exp(\boldsymbol{w}_j^T \boldsymbol{v}_i))}{\sum_j^C exp(\boldsymbol{w}_j^T \boldsymbol{v}_i) P_{i,k} \sum_j^C exp(\boldsymbol{w}_j^T \boldsymbol{v}_i)} \\
&= \sigma_1 \frac{1}{P_{i,k} \sum_j^C exp(\boldsymbol{w}_j^T \boldsymbol{v}_i)} \\
&\leq \sigma_1 \frac{1}{P_{i,k} \sum_j^C exp(\boldsymbol{w}_j^T \boldsymbol{v}_i) - \sum_j \boldsymbol{w}_j^T \boldsymbol{v}_i} \\
&\leq \sigma_1 \frac{1}{P_{i,k} \sum_j^C [1 + (\boldsymbol{w}_j^T \boldsymbol{v}_i)] - \sum_j \boldsymbol{w}_j^T \boldsymbol{v}_i} \\
&\leq \frac{\sigma_1}{\sum_j [P_{i,k} - (1 - P_{i,k}) \boldsymbol{w}_j^T \boldsymbol{v}_i]}.
\end{aligned} \tag{15}$$

Recall that the pushing force $\boldsymbol{Z}_{2,i}$ of a sample can be represented as $(1 - P_{i,k})\boldsymbol{v}_i$, so we get

$$\beta^2 \leq B^2 = \frac{\sigma_1}{\sum_j [P_{i,k} - \boldsymbol{w}_j^T \boldsymbol{Z}_{2,i}]}$$

As a result, the L2 norm of gradient in l-th layer for i-th sample $(\boldsymbol{x}_i, y_i)$ can be written as:

$$\begin{aligned}
&\|\nabla_{\boldsymbol{w}_l} F(h; x_i, y_i)\|_2 \\
&\leq \rho^0 \left\| diag(\beta_1^1, ..., \beta_k^2, ..., \beta_C^1) \right\|_2 \\
&\leq \rho^0 \left\| diag(B_1^1, ..., B_k^2, ..., B_C^1) \right\|_2 .
\end{aligned} \tag{16}$$

As shown in [35], the $L2$ norm of the gradient reflects the model update characteristics well. The larger the gradient norm, the bigger the contribution to the model update of the sample. The Eq. (16) shows that pulling force $\boldsymbol{Z}_1$ and pushing force $\boldsymbol{Z}_2$ affect the variation of gradient norm; thus, we can show that $\boldsymbol{Z}_1$ and $\boldsymbol{Z}_2$ can show vital information of model update.

In summary, since the pulling and pushing forces carry information on model update and dataset distribution, the gradient trajectory is a good choice for client clustering. In an FL system, a trajectory represents the dataset and model update of the correspondent client. By clustering the trajectories, we can group clients who are similar at the model update and dataset level into the same category, thus alleviating the impact of heterogeneous data.

D. Analysis of CFLGT

Firstly, the point order of trajectory cannot influence the result of client clustering. Though it is the order of the points that defines the shape of the trajectory, our clustering input is a distance matrix, computed by the distance between corresponding points of the trajectory. Thus, whatever the order is, the distance matrix is fixed and so is the clustering result.

For clients, calculating pulling force $\boldsymbol{Z}_1$ and pushing force $\boldsymbol{Z}_2$ does not consume much time. As defined in Eq. (6), the output of feature extractor v and the probability of correct label p are all $\boldsymbol{Z}_1$ or $\boldsymbol{Z}_2$ need, which can be obtained in forward propagation. It means a client only needs to complete forward propagation to get its $\boldsymbol{Z}_1$ and $\boldsymbol{Z}_2$. Considering that [34] proves that the backward propagation requires about twice the amount of time as the forward propagation since it needs to compute full gradients, calculating $\boldsymbol{Z}_1$ and $\boldsymbol{Z}_2$ is not a time-consuming task for a client.

The Affinity Propagation algorithm [36] helps achieve clients to be clustered without a pre-defined number. The success of Affinity Propagation is the apparent clustering trend in the gradient trajectory, which carries client information (proved in the section above). After rounds of pre-training, the gradient trajectories of clients often exhibit great potential for clustering and distinctly different distributions (An example is shown in Fig. 5). Besides, we introduce a concept named Hopkins Statistic H [37] to assess the clustering trend of the data. If the data points are uniformly distributed in the space, H is approximately 0.5. If the clustering situation exists in the data set, H will be close to 1. The case that H is higher than 0.75 indicates a clustering trend in the data set at a 90% confidence level. The H of gradient trajectories are usually higher than 0.75, reflecting the clustering trend. As a result, the clustering trend of gradient trajectories ensures that the server can utilize gradient trajectories for client clustering by Affinity Propagation.

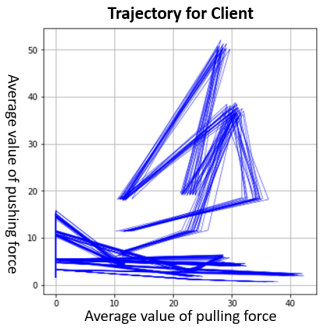

Fig. 5. Gradient Trajectories of 100 clients in an FL System. The trajectories of the 100 clients form several distributions, showing a clear trend of clustering.

One of the concerns about privacy in FL is unintentional data leakage & reconstruction through inference [38], [39], that attackers may obtain the information uploaded by clients to the server and then utilize the information to reconstruct or attack raw data in clients. For privacy protection, our method only requires the client to upload its average values of pulling force $\boldsymbol{Z}_1'$ and pushing force $\boldsymbol{Z}_2'$, rather than $\boldsymbol{Z}_1$ and $\boldsymbol{Z}_2$. In this way, even though an attacker successfully captures the $\boldsymbol{Z}_1'$ and $\boldsymbol{Z}_2'$, it still cannot restore the clients' raw data by the attacking method in [31]. The details of the privacy protection of our method are shown in Section V.

V. EXPERIMENTS

This section presents the experiments we have performed to evaluate CFLGT. First, we compare CFLGT with other baselines on three scenarios, where heterogeneity is different. Secondly, we experimentally show that CFLGT does not need the number of clusters as input while other Clustered

TABLE I
PERFORMANCE COMPARISONS WITH OTHER BASELINES ON THREE SCENARIOS.
THE AVERAGE ACCURACY OF THE LAST 20 ROUNDS IS REPORTED. THE μ IN FEDPROX IS 0.001 [15].

Methods	Cifar			Fmnist			SVHN		
	scenario1	scenario2	scenario3	scenario1	scenario2	scenario3	scenario1	scenario2	scenario3
Fedavg	46.44%	50.51%	36.23%	79.09%	83.21%	74.45%	25.23%	75.94%	69.89%
Fedavg+FT	78.05%	64.87%	69.03%	97.58%	90.45%	91.14%	57.28%	82.53%	71.07%
Fedprox	40.64%	48.57%	32.02%	78.45%	83.04%	74.72%	25.23%	71.69%	56.86%
Fedsem	73.97%	64.49%	69.10%	99.63%	91.37%	84.86%	74.47%	81.54%	81.12%
FL+HC	81.40%	65.37%	72.70%	99.61%	**92.37%**	84.58%	91.36%	84.69%	85.74%
IFCA	72.80%	60.20%	54.66%	98.87%	91.01%	84.89%	51.03%	80.84%	30.47%
CFLGT	**83.17%**	**65.72%**	**73.49%**	**99.64%**	92.34%	**93.09%**	**92.61%**	**84.91%**	**86.33%**

Federated Learning (CFL) methods may suffer from a wrong pre-defined number of clusters. Then a comparative experiment demonstrates that our method is more efficient and can be implemented without significant communication or computation expense. Finally, we show why our method can protect the data privacy of clients.

A. Performance Comparison

Experimental Setting: We design three scenarios in experiments, which are done in related works [19], [40], [41]. In the first heterogeneous scenario (scenario 1), we consider that each dataset consists of two types of labels. Take Cifar10 as an example. One dataset can have labels 0 and 1, whereas the other has labels 1 and 3. We form five types of label combinations randomly from these to indicate different data distributions. After that, each client randomly chooses one label combination and is assigned 50 samples for each type of label. Note that the selected sample will not be selected again. As a result, the FL system finally has 100 clients, each with 50×2 training samples. Since clusters should be generated based on types of data distribution, the clients can naturally form 5 clusters.

The second heterogeneous scenario (scenario 2) is similar to the first one, except that we consider five labels in each combination. Thus, in scenario 2, a client has 50×5 training samples.

The third heterogeneous scenario (scenario 3) also follows scenario 1, but there are twenty combinations in all, and every combination contains three types of labels. Considering that every client selects one from 20 twenty combinations and the server does not exactly know the combination of every client, it is possible that the FL system contains less than 20 data distributions, making scenario 3 a more challenging task. For other CFL methods, we suppose that the clients form twenty clusters based on the data distribution, and the FL system in scenario 3 has 100 clients with 50×3 samples each.

The three scenarios are implemented with Cifar10, FashionMNIST, and SVHN. Lenet (A CNN in Python) is chosen in Cifar10 and Street View House Numbers (SVHN) Dataset and a 2-layer DNN is for FashionMNIST. We take 200 global rounds (100 global rounds in FashionMNIST) and an extra 25 pre-training rounds for Clustered Federated Learning baselines with a batch size of 32, SGD with momentum 0.9 as the optimizer, and a learning rate of 0.001. In each training round, 20% of clients are chosen to participate in training, and accuracy is calculated by averaging accuracy on the test dataset across all clients.

Baselines: We compare CFGLT with other CFL algorithms, including FL+HC based on model parameters, FedSem based on model L2-distance, and IFCA based on models' local performance. In FL+HC, the server receives the update of the client model and uses Hierarchical Clustering [42] to assign clients into groups. In Fedsem, after obtaining updated models from clients, the server assigns the client to the corresponding group with the L2 distance of models. In IFCA, the server only stores global models of all clusters while every client chooses its group based on the global models' performance on the local dataset. Some popular FL algorithms are also involved: FedAvg, FedProx based on regularization in the loss function, and FedAvg+FT, where after receiving the global model, the client will update the bottom layer of the model on its data. FedAvg+FT is proved as a powerful and straightforward Personalized Federated Learning method in [16].

For a fair comparison, all clustering methods are completed in the pre-training round, where all clients are involved. After entering the global training, each client will not be moved to another group. This setting can help us know the efficiency and accuracy of clustering methods since a bad clustering of clients causes a bad model performance in FL.

Results and Analysis: The results compared with baselines are listed in Table I. Note that we take five local training epochs and calculate the average accuracy of the last 20 rounds. Among all algorithms, our method achieves almost all the best outcomes. Some methods perform almost as well as CFLGT in several situations, because they get the right pre-defined number of clusters, which is impractical in real tasks (like scenario 3). With a wrong pre-defined cluster number, their performance decreases sharply while our method will not be influenced, which will be displayed in the following part.

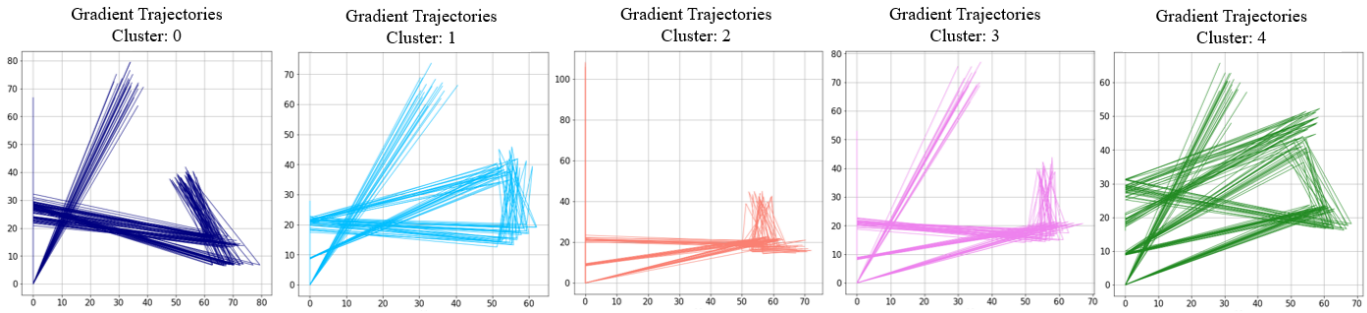

Fig. 6. Trajectories for clusters in scenario 1 based on Cifar10. Each one plots the trajectories of clients in one cluster, the output of the clustering algorithm. It is clear that clients in the same cluster have similar trajectories. Some clusters of trajectories may have similar shapes but have different coordinate values. (e.g. cluster 0 and cluster 1) These images prove that the clustering of trajectories in CFLGT is reasonable.

In summary, The results echo the values of gradient trajectory about the model update and dataset distribution, which are utilized in client clustering.

To evaluate the clustering of CFLGT, we visualize the trajectories of clusters in the experiment (scenario 1 based on Cifar10) as shown in Fig. 6. Note that we set 5 dataset distributions among clients in scenario 1, and our method produces five groups of clients, which totally matches our setting. It shows that our method works correctly. Meanwhile, from Fig. 6 we can see that there is apparent diversity between trajectories of clusters. Because a trajectory represents a client in the FL system, the diversity displays that our method indeed groups clients well. Additionally, we can see that clients in the same cluster have similar shapes of trajectories. It means that similar clients are clustered into the same group.

B. No Need of Pre-defined Number of Cluster

In a real-world task, it is hard for the server to get information about clients' data distribution, resulting in a wrong pre-defined number of clusters. But our method relies on defining the number of clusters, avoiding potential damage by the wrong number of clusters. In this part, our experiments show the negative impact of the wrong number of clusters on the FL system and the benefits of our method. Considering the high performance in the previous experiments, we choose FL+HC to testify to the damage from the wrong cluster number for a better comparison. First, we add two extra baselines, FL+HC(10) and FL+HC(30). The first one represents 10 clusters as the input of FL+HC and another one represents 30 clusters as the input. The experiments are implemented in scenario 3 based on Cifar10 (we set 20 distributions across clients). As a result, we compare our method with three baselines including FL+HC(20) (right number of cluster), FL+HC(10), and FL+HC(30).

The results are shown in Table II, from which we can see that CFLGT can achieve good performance in scenario 3 as it does not need anything about dataset information. However, the FL+HC algorithm with the right number is almost as good as CFLGT. But FL+HC(10) and FL+HC(30) lead to reduced performance. The source of the problem is obvious. As shown in Fig. 7, FL+HC(10) has fewer clusters than FL+HC(20), so the server arranges unique clients with other dissimilar clients into the same group, hurting the model aggregation

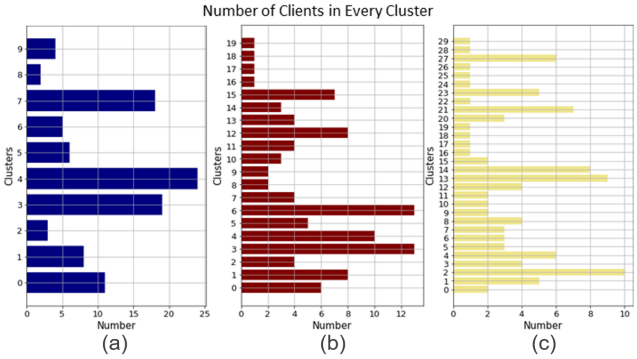

Fig. 7. Number of clients in each cluster for three baselines. The Y-axis represents the generated clusters and the X-axis shows the number of clients in each corresponding cluster. The results (a)(b)(c) are from FL+HC(10), FL+HC(20), and FL+HC(30) respectively.

TABLE II
PERFORMANCE COMPARISONS WITH OTHER BASELINES ON SCENARIO 3. THE FL+HC HAS THE CORRECT NUMBER OF CLUSTERS AS INPUT. THE AVERAGE ACCURACY OF THE LAST 20 ROUNDS IS REPORTED AND THE FIGURE REPRESENTS THE PRE-DEFINED NUMBER OF CLUSTERS.

	FL+HC(10)	FL+HC(20)	FL+HC(30)	CFLGT
scenario3	67.90%	72.70%	70.51%	**73.49%**

in the group. FL+HC(30) has more clusters, so the server splits similar clients into different groups, hurting the model aggregation as well. Because in several groups, there is only one client, resulting in the aggregation as a failure.

C. Efficiency Comparison

We analyze the time cost and volume of uploaded data of CFL methods among the baselines. In the experiment, we set up a scenario closer to a complex environment in the real world, where the trained model is AlexNet [43] without batch norm layers and dropout layers. Every client's dataset contains 1500 images and all clients are involved during client clustering. We compare our method (CFLGT) with the CFL methods above (FL+HC, FedSem, and IFCA). Note that we set the number of clusters as 15 since there are 15 data distributions among clients. In the experiment, we focus on the consumption of computing resources and the needed bandwidth of these methods.

Consumption of computing resources: For the consumption

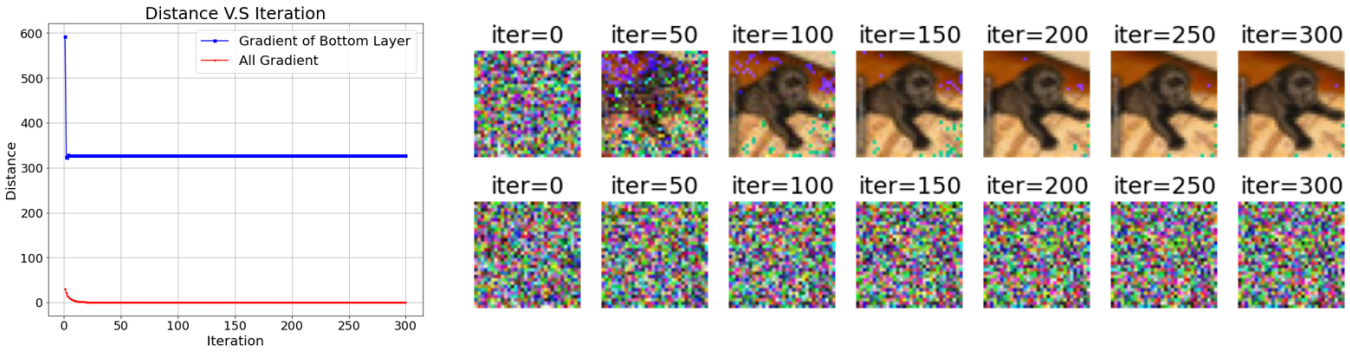

Fig. 8. Results in Gradient Leakage. The two sets of images on the right display the update of the fake image to the original image as iteration grows. The upper one is based on the gradient of the whole model and another one is based on the gradient of the bottom layer of the model. The figure on the left shows the change in gradient distance as the iteration increases.

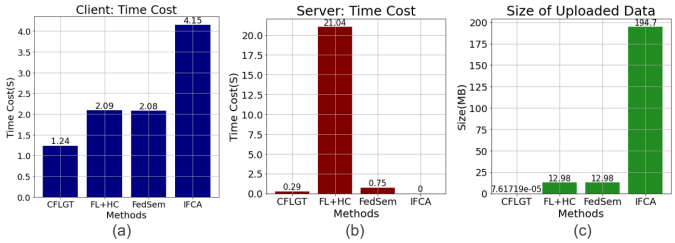

Fig. 9. Efficiency Comparison. The result (a) represents the time cost of the client. The result (b) represents the time cost of the server and the result (c) represents the size of uploaded data from client to server.

of computing resources, the time cost of CFLGT is the best. Since the clustering is composed of computation in client and server, we analyze the time-cost on the client and the server separately. For time-cost on the client side, Left plots in Figure 9 show that our method uses the least computing resources. The FL+HC and FedSem need to update the model, including complex backpropagation. The IFCA requires every client to test all 15 global models on their dataset, causing a huge consumption of computing resources. For the time cost on the server side, the middle plot of Figure 9 shows that our method does not require the server to complete extensive calculations while the server in FL+HC is burdened with heavy work.

Bandwidth: We evaluate the required bandwidth by estimating the size of the uploaded data from the client. The results in Figure 9 (c) clarify that only a very low bandwidth is required in our method even though every client is involved in client clustering. The client just uploads its gradient trajectories, i.e., 0.078KB estimated in the experiment. In contrast, the other CFL methods need to upload the whole model parameters, which increases the burden of communication when a huge model is implemented in the FL system. For example, BERT [44], a popular NLP model, contains over 300 million parameters sizing in 340 MB. If BERT is chosen as the training model, IFCA even needs a server to broadcast BERT models of all clusters to every client.

The above results in consumption of resources and bandwidth show that our method can be used in a poor communication environment and resource-constrained devices, and thus can be more practical in real-world tasks.

D. Privacy Protection

We set up a comparison experiment to testify to the privacy protection of CFLGT. Note that we only discuss the privacy protection of information exchange between the client and the server and the rest of FL privacy issues are not included. Suppose there is an attacker disguised as a client in the FL system. The FL system is vulnerable to inference attacks, as every client contains one image as a dataset and updates only once time in local training in each round. If the attacker captures the update uploaded to the server by one of the clients, it can restore the raw image by the method in [22]. Specifically, the attacker generates a noisy image of the same size as the raw image, called a fake image. Then it shortens the distance between the captured gradient and the gradient from the fake image in multiple iterations. The attacker utilizes the gradient descent method to update its fake image during the iterations until the distance is close to zero. Finally, the attacker can obtain an image very close to the raw image. The top right images in Fig. 8 show a successful inference attack by the captured gradient of the whole model.

For CFLGT, we set up the worst-case situation, where the attacker obtains the pulling force and pushing force and then combines them into the gradient of the bottom layer. As a result, the attack can utilize the same inference method with the captured gradient of the bottom layer to restore raw data. However, the bottom right images in Fig. 8 show that the attacker fails to obtain the raw image from the fake image since it still gets an image composed of noise after 300 iterations. Meanwhile, the left chart in Fig. 8 exhibits that the gradient distance in the scenario remains much higher than the attack based on the gradient of the whole model. It means that the attacker cannot complete an inference attack only with the gradient of the bottom layer in a model.

Note that in a real scenario of CFLGT, the attacker cannot even restore the pulling force and pushing force because it only captures the average values of pulling force and pushing force. Thus it is much harder to compute the gradient of the bottom layer. Therefore, our method, where clients only upload average values of pulling force and pushing force, is more robust on inference attacks.

VI. Conclusions and Future Work

In this paper, we utilize the gradient of the bottom layer of the model in Federated Learning and transform it into a gradient trajectory. We show that with the trajectories from all the clients, the server can effectively observe the heterogeneity of the data. Based on this, we proposed a Clustered Federated Learning method (CFLGT) where the server utilizes the gradient trajectory to cluster clients into groups and then aggregates models in every group. We have shown our method to be beneficial in theory and comprehensive experiments demonstrate its advantages in real-world scenarios.

For future research, we plan to improve the client clustering mechanisms of the CFLGT so that it can perform accurate clustering when new clients join or clients' data changes. We also hope to expand our research to regression tasks and unsupervised tasks. Plus, as a variant of federated learning, a still faces the problem of Communication and Synchronization Overhead. We will try to optimize the communication protocols to overcome the difficulties and further improve the performance of the CFLGT.

Acknowledgement

References

[1] A. Mehmood, I. Natgunanathan, Y. Xiang, G. Hua, and S. Guo, "Protection of big data privacy," *IEEE access*, vol. 4, pp. 1821–1834, 2016.

[2] B. McMahan, E. Moore, D. Ramage, S. Hampson, and B. A. y Arcas, "Communication-efficient learning of deep networks from decentralized data," in *Artificial intelligence and statistics*. PMLR, 2017, pp. 1273–1282.

[3] P. Kairouz, H. B. McMahan, B. Avent, A. Bellet, M. Bennis, A. N. Bhagoji, K. Bonawitz, Z. Charles, G. Cormode, R. Cummings *et al.*, "Advances and open problems in federated learning," *arXiv preprint arXiv:1912.04977*, 2019.

[4] Q. Li, Y. Diao, Q. Chen, and B. He, "Federated learning on non-iid data silos: An experimental study," 2021.

[5] A. Nilsson, S. Smith, G. Ulm, E. Gustavsson, and M. Jirstrand, "A performance evaluation of federated learning algorithms," in *Proceedings of the second workshop on distributed infrastructures for deep learning*, 2018, pp. 1–8.

[6] X. Li, K. Huang, W. Yang, S. Wang, and Z. Zhang, "On the convergence of fedavg on non-iid data," *arXiv preprint arXiv:1907.02189*, 2019.

[7] M. Al-Shedivat, L. Li, E. Xing, and A. Talwalkar, "On data efficiency of meta-learning," in *International Conference on Artificial Intelligence and Statistics*. PMLR, 2021, pp. 1369–1377.

[8] A. Fallah, A. Mokhtari, and A. Ozdaglar, "Personalized federated learning with theoretical guarantees: A model-agnostic meta-learning approach," *Advances in Neural Information Processing Systems*, vol. 33, pp. 3557–3568, 2020.

[9] Z. Charles and J. Konečnỳ, "Convergence and accuracy trade-offs in federated learning and meta-learning," in *International Conference on Artificial Intelligence and Statistics*. PMLR, 2021, pp. 2575–2583.

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

[11] T. Li, S. Hu, A. Beirami, and V. Smith, "Ditto: Fair and robust federated learning through personalization," in *International Conference on Machine Learning*. PMLR, 2021, pp. 6357–6368.

[12] B. Sun, H. Huo, Y. Yang, and B. Bai, "Partialfed: Cross-domain personalized federated learning via partial initialization," *Advances in Neural Information Processing Systems*, vol. 34, 2021.

[13] V. Kulkarni, M. Kulkarni, and A. Pant, "Survey of personalization techniques for federated learning," in *2020 Fourth World Conference on Smart Trends in Systems, Security and Sustainability (WorldS4)*. IEEE, 2020, pp. 794–797.

[14] 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," *arXiv preprint arXiv:2106.05001*, 2021.

[15] X.-C. Li and D.-C. Zhan, "Fedrs: Federated learning with restricted softmax for label distribution non-iid data," in *Proceedings of the 27th ACM SIGKDD Conference on Knowledge Discovery & Data Mining*, 2021, pp. 995–1005.

[16] L. Collins, H. Hassani, A. Mokhtari, and S. Shakkottai, "Exploiting shared representations for personalized federated learning," *arXiv preprint arXiv:2102.07078*, 2021.

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

[18] J. Yosinski, J. Clune, Y. Bengio, and H. Lipson, "How transferable are features in deep neural networks?" *arXiv preprint arXiv:1411.1792*, 2014.

[19] F. Sattler, K.-R. Müller, and W. Samek, "Clustered federated learning: Model-agnostic distributed multitask optimization under privacy constraints," *IEEE transactions on neural networks and learning systems*, 2020.

[20] M. Duan, D. Liu, X. Ji, R. Liu, L. Liang, X. Chen, and Y. Tan, "Fedgroup: Efficient federated learning via decomposed similarity-based clustering," in *2021 IEEE Intl Conf on Parallel & Distributed Processing with Applications, Big Data & Cloud Computing, Sustainable Computing & Communications, Social Computing & Networking (ISPA/BDCloud/SocialCom/SustainCom)*. IEEE, 2021, pp. 228–237.

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

[22] L. Zhu and S. Han, "Deep leakage from gradients," in *Federated learning*. Springer, 2020, pp. 17–31.

[23] L. Melis, C. Song, E. De Cristofaro, and V. Shmatikov, "Exploiting unintended feature leakage in collaborative learning," in *2019 IEEE Symposium on Security and Privacy (SP)*. IEEE, 2019, pp. 691–706.

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

[25] N. Gholizadeh and P. Musilek, "Federated learning with hyperparameter-based clustering for electrical load forecasting," *Internet of Things*, vol. 17, p. 100470, 2022.

[26] Y. Movshovitz-Attias, A. Toshev, T. K. Leung, S. Ioffe, and S. Singh, "No fuss distance metric learning using proxies," in *Proceedings of the IEEE International Conference on Computer Vision*, 2017, pp. 360–368.

[27] J. Snell, K. Swersky, and R. S. Zemel, "Prototypical networks for few-shot learning," *arXiv preprint arXiv:1703.05175*, 2017.

[28] A. Zhai and H.-Y. Wu, "Making classification competitive for deep metric learning," *arXiv preprint arXiv:1811.12649*, vol. 11, 2018.

[29] A. Wainakh, F. Ventola, T. Müßig, J. Keim, C. G. Cordero, E. Zimmer, T. Grube, K. Kersting, and M. Mühlhäuser, "User label leakage from gradients in federated learning," *arXiv preprint arXiv:2105.09369*, 2021.

[30] J. Geiping, H. Bauermeister, H. Dröge, and M. Moeller, "Inverting gradients–how easy is it to break privacy in federated learning?" *arXiv preprint arXiv:2003.14053*, 2020.

[31] B. Zhao, K. R. Mopuri, and H. Bilen, "idlg: Improved deep leakage from gradients," *arXiv preprint arXiv:2001.02610*, 2020.

[32] C. E. Brown, "Coefficient of variation," in *Applied multivariate statistics in geohydrology and related sciences*. Springer, 1998, pp. 155–157.

[33] X. Li, M. Jiang, X. Zhang, M. Kamp, and Q. Dou, "Fedbn: Federated learning on non-iid features via local batch normalization," *arXiv preprint arXiv:2102.07623*, 2021.

[34] A. Katharopoulos and F. Fleuret, "Not all samples are created equal: Deep learning with importance sampling," in *International conference on machine learning*. PMLR, 2018, pp. 2525–2534.

[35] Y. He, J. Ren, G. Yu, and J. Yuan, "Importance-aware data selection and resource allocation in federated edge learning system," *IEEE Transactions on Vehicular Technology*, vol. 69, no. 11, pp. 13 593–13 605, 2020.

[36] D. Dueck, *Affinity propagation: clustering data by passing messages*. Citeseer, 2009.

[37] A. Banerjee and R. N. Dave, "Validating clusters using the hopkins statistic," in *2004 IEEE International conference on fuzzy systems (IEEE Cat. No. 04CH37542)*, vol. 1. IEEE, 2004, pp. 149–153.

[38] V. Mothukuri, R. M. Parizi, S. Pouriyeh, Y. Huang, A. Dehghantanha, and G. Srivastava, "A survey on security and privacy of federated learning," *Future Generation Computer Systems*, vol. 115, pp. 619–640, 2021.

[39] B. Hitaj, G. Ateniese, and F. Perez-Cruz, "Deep models under the gan: information leakage from collaborative deep learning," in *Proceedings of the 2017 ACM SIGSAC Conference on Computer and Communications Security*, 2017, pp. 603–618.

[40] D. Li and J. Wang, "Fedmd: Heterogenous federated learning via model distillation," *arXiv preprint arXiv:1910.03581*, 2019.

[41] F. Yu, A. S. Rawat, A. Menon, and S. Kumar, "Federated learning with only positive labels," in *International Conference on Machine Learning*. PMLR, 2020, pp. 10 946–10 956.

[42] F. Murtagh and P. Contreras, "Algorithms for hierarchical clustering: an overview," *Wiley Interdisciplinary Reviews: Data Mining and Knowledge Discovery*, vol. 2, no. 1, pp. 86–97, 2012.

[43] A. Krizhevsky, I. Sutskever, and G. E. Hinton, "Imagenet classification with deep convolutional neural networks," *Advances in neural information processing systems*, vol. 25, 2012.

[44] I. Tenney, D. Das, and E. Pavlick, "Bert rediscovers the classical nlp pipeline," *arXiv preprint arXiv:1905.05950*, 2019.