World Wide Web (2023) 26:2175–2200
https://doi.org/10.1007/s11280-022-01132-0

Personalized federated learning with model interpolation among client clusters and its application in smart home

Zhikai Yang[1] · Yaping Liu[1] · Shuo Zhang[1] · Keshen Zhou[2]

Received: 17 October 2022 / Revised: 27 November 2022 / Accepted: 8 December 2022 /
Published online: 16 January 2023

Abstract
The proliferation of high-performance personal devices and the widespread deployment of machine learning (ML) applications have led to two consequences: the volume of private data from individuals or groups has exploded over the past few years; and the traditional central servers for training ML models have experienced communication and performance bottlenecks in the face of massive amounts of data. However, this reality also provides the possibility of keeping data local for ML training and fusing models on a broader scale. As a new branch of ML application, Federated Learning (FL) aims to solve the problem of multi-party joint learning on the premise of protecting personal data privacy. However, due to the heterogeneity of devices, including network connection, network bandwidth, computing resources, etc., it is unrealistic to train, update and aggregate models in all devices in parallel, while personal data is often not independent and identically distributed (Non-IID) due to multiple reasons. This reality poses a challenge to the speed and convergence of FL. In this paper, we propose the pFedCAM algorithm, which aims to improve the robustness of the FL system to device heterogeneity and Non-IID data, while achieving some degree of federation model personalization. pFedCAM is based on the idea of clustering and model interpolation by classifying heterogeneous clients and performing FedAvg algorithm in parallel, and then combining them into personalized federated global models by inter-cluster model interpolation. Experiments show that the accuracy of pFedCAM improves 10.3% on Fashion-MNIST and 11.3% on CIFAR-10 compared to the benchmark in the case of Non-IID data. In the end, we applied pFedCAM in HomeProtect, a smart home privacy protection framework we designed, and achieved good practical results in the case of flame recognition.

✉ Yaping Liu
ypliu@gzhu.edu.cn

✉ Shuo Zhang
szhang18@gzhu.edu.cn

Zhikai Yang
yangzhikai96@foxmail.com

Keshen Zhou
kzho6770@uni.sydney.edu.au

[1] Cyberspace Institute of Advanced Technology, Guangzhou University, Guangzhou, China

[2] School of Computer Science, The University of Sydney, Sydney, Australia

Keywords Federated Learning · Personalization · Clustering · Model interpolation

1 Introduction

In the past few years, a large number of devices with different computing capabilities have been put into the market, such as mobile devices like smartphones, Internet of Things (IOT) devices, and smart cars. These devices have generated a large amount of data due to their extensive and long-term use. These data are very attractive for data-driven machine learning (ML), and will contribute to the training of ML models. However, the traditional way of centralized ML is to upload personal data to a central server for model training, which will compromise the privacy of individuals. The EU introduced the General Data Protection Regulation (GDPR) in 2018, which is a privacy protection regulation designed to set out the rules that companies should follow when collecting, processing and using users' data. With the gradual implementation of privacy protection policies in various countries and the gradual awakening of people's awareness of privacy protection, the method of collecting data, uploading it to servers and training it no longer applies. Google provides us with an effective distributed ML paradigm. In 2016, Google [1] proposed the concept of Federated Learning (FL) and successfully applied it to Google keyboard [2], providing a powerful tool to break the barrier of data silos. With FL, instead of uploading data, the data owner will upload the ML models obtained using local computing resources to the server, which will aggregate the models. Because of the privacy-sensitive data protection feature, FL is widely used in the field of privacy-preserved ML, such as financial lending, medical diagnosis [3, 4], etc. If it is based on existing blockchain technologies and applications, such as data auditing [5] and energy dispatching [6], FL will have a broader application prospect and its privacy protection features will be strengthened.

However, unlike distributed ML based on server environment, FL is built on a more complex device environment and it faces some fundamental challenges. Since the devices of individuals or groups participating in the FL system have different computing resources, network bandwidth and network connectivity, and the availability of these devices are not stable at all times due to non-hardware factors such as usage habits, it is very difficult to design synchronous or semi-synchronous protocols as in the case of traditional distributed ML. For these reasons, it is a common strategy to select some clients but not all to participate in training in order to avoid the FL system from getting into long-time waiting due to Some devices being offline or unstable network conditions. McMahan et al. [1] proposed FedAvg algorithm, which randomly selects a certain proportion of clients to upload model weights at the end of each round of local training, and then the server averages the weights. FedCS [7] selects the appropriate clients by measuring the client resources, and accommodates as many clients as possible to participate in the aggregation without entering a long wait.

In addition to the device heterogeneity challenge, FL also faces the statistical heterogeneity challenge. Most of the existing FL algorithms do not consider the statistical challenges posed by heterogeneous local datasets in a global sense. Due to the heterogeneity of devices and different user usage patterns, individual data may have attribute skew or label skew, that is, data from different clients may not come from the same global distribution, and the model trained by selecting from part of clients may not reflect the overall data distribution, leading to the introduction of unavoidable bias in the update of the global model. Device heterogeneity and statistical heterogeneity cause the problem of not independent and identically distributed (Non-IID) data in FL. Several studies [8–10] have shown that in

the case of Non-IID data, there is a significant decrease in model convergence speed and accuracy with the resulting increase in the number of communication rounds in FL. Considering the FedAvg algorithm under Non-IID data, since clients use Non-IID data in local training, the variation between models trained by different clients is too large, resulting in slowing down the global model convergence speed and significantly reducing the model accuracy in the model aggregation phase.

The generation of device heterogeneity and Non-IID data problems on the other hand reflects the need for personalized FL on the client side. Personalization of the global model to adapt it to the local data distribution of the client is one of the ways to solve the various heterogeneity problems. The personalized FL also has a real demand. As an information filtering model, recommendation system uses user portraits and habits of the whole community to recommend content that may be of interest or high relevance to specific users. However, the premise to enjoy personalized recommendation service is to hand over personal privacy data, such as historical browsing records, to content providers, which undoubtedly damages personal privacy. The combination of FL and recommendation system can provide users with accurate and personalized services on the premise of protecting users' privacy and trade secrets. FL also shines in areas such as finance and medicine.

In this paper, we design a personalized Federated Learning algorithm with clustering and model interpolation (pFedCAM) for mitigating the impact of device heterogeneity and Non-IID data on the FL system, while adopting a personalization mechanism to ensure that the algorithm performs well on local Non-IID datasets. For the first time, we extend the model interpolation [11] from the global model and the local model to the client clusters, and construct a personalized model for each cluster by setting parameters. The accuracy, stability and robustness of our algorithm on IID data and Non-IID data are better than FedAvg and FAVOR [12]. Due to the privacy protection principle, we cannot directly access the data on the clients located at the edge or analyze the data distribution directly, but we can analyze the potential connection between different clients from a more coarse-grained perspective. Based on intuition and experience, the target data distribution learned by clients with similar computing resources and similar data ownership has some similarity. Therefore, we first measure the computational resources of the clients as well as the data resources held, and use the clustering algorithm to divide the clients into several different clusters, and execute the original FedAvg algorithm within each client cluster as a way to enhance the prediction capability for similar targets and tasks. Then we combine the cluster-averaged models of different clusters in the central server using model interpolation to form a globally integrated model, and the generalization capability of the model can be enhanced by this step. Finally, by assigning different weights to the base learners in the globally integrated model, we can obtain a personalized FL model that enhances the performance of the client model in the local environment. In order to prove the usability of pFedCAM in the real environment, we designed a smart home privacy protection framework based on FL, HomeProtect [13], in which the FL algorithm uses pFedCAM and has achieved good performance in fire identification and early warning cases.

The rest of this paper is organized as follows. In the second part, we will specifically introduce the FL background, the specific impact of heterogeneous devices and Non-IID data on the FL system, as well as the related concepts of model interpolation. In the third part, we will explain the mechanism of pFedCAM. In the fourth part, we use PyTorch to conduct a number of experiments to illustrate the effectiveness of the work. In the fifth part, we will introduce HomeProtect and how pFedCAM is applied in it. In the end, we will briefly introduce the relevant work and make a summary of our work.

2 Background and motivation

In this section we introduce how FL works in general, and describe how heterogeneous devices and Non-IID data pose challenges to existing FL systems, and how model interpolation ideas and Bagging algorithms can be used for the purpose of mitigating heterogeneity and personalizing FL.

2.1 Federated learning

FL obtains the global ML model by aggregating the gradients or weights from the model locally trained by the client. FL can be divided into horizontal FL (HFL), vertical FL (VFL) and Federated Transfer Learning (FTL) according to the distribution of data in sample space and attribute space [14]. In HFL, the data owned by each participant has the same attribute space but different sample space, which is the most common FL method. In VFL, the data owned by each participant has overlapping sample space but the attribute space is not completely consistent. In FTL, the sample space and the attribute space overlap but are not completely consistent. According to different communication architectures, FL can be divided into centralized FL with central parameter server and P2P architecture [15, 16] without central server. In this paper, we will focus on the centralized HFL.

The general process of FL is as follows: the clients check in with the FL server, and the server issues the initial model. After the client receives the model, it uses the local data for training and uploads the model parameters or gradients to the server, which will update the global model and issue the updated model to the clients again. After several rounds of communication, the global model will converge to the predetermined target.

As part of our proposed approach, we will formally describe the FL process as well as the FedAvg algorithm. Assuming that there are K participants, with k representing the kth client, $k \in [1, K]$. D_k represents the dataset of the client k. M_v is the ML model obtained by single machine algorithm, P_v is its effectiveness, M_f is the model obtained by FL, P_f is its effectiveness, if

$$\left|P_f - P_v\right| < \delta, \delta > 0 \tag{1}$$

FL is said to be valid. Assume that the target function of the client k is

$$F_k(w) = \frac{1}{n_k} \sum_{i \in D_k} f_i(w) = \frac{1}{n_k} \sum_{i \in D_k} l\left(x_i, y_i; w\right) \tag{2}$$

where w is the model parameter, $n_k := \left|D_k\right|$ is the number of samples owned by client k, $n = \sum_{i=1}^{K} n_i$ is the total number of samples of all clients, $l(\cdot)$ is the loss function, and (x_i, y_i) is the sample. The optimization objective is

$$minimize \ F_k(w) \tag{3}$$

Using the stochastic gradient descent algorithm (SGD), assuming that the initial parameter of the client k is w_t, there is local training

$$\begin{aligned} w_{t+1}^k &\leftarrow w_t \\ g^k &= \nabla F\left(w_{t+1}^k\right) \\ w_{t+1}^k &= w_{t+1}^k - \eta g^k \end{aligned} \tag{4}$$

where η is learning rate. After uploading the updated weight w_{t+1}^k to the server, the server will average the weight of each client

$$w_{t+1} = \sum_{k=1}^{K} \frac{n_k}{n} w_{t+1}^k \tag{5}$$

and the global average model is obtained.

The FedAvg algorithm based on the above idea is as follows, which we will use later.

Algorithm 1. FedAvg

Input: The K clients are indexed by k; B is the local minibatch size, E is the number of local epochs, and η is the learning rate.
Server:
Initialize w_0
1. **for** each round $t = 1, 2, \ldots$ **do**
2. $m \leftarrow max(C \cdot K, 1)$
3. $S_t \leftarrow$(random set of m clients)
4. **for** each client $k \in S_t$ **in parallel do**
5. $w_{t+1}^k \leftarrow \text{ClientUpdate}(k, w_t)$
6. **end for**
7. $w_{t+1} \leftarrow \sum_{k=1}^{K} \frac{n_k}{n} w_{t+1}^k$
8. **end for**

ClientUpdate(k,w): //*Run on client k*
9. $\mathcal{B} \leftarrow$(split $\mathcal{P}_k$ into batches of size B)
10. **for each** local epoch i from 1 to E **do**
11. **for** batch $b \in \mathcal{B}$ **do**
12. $w \leftarrow w - \eta \nabla l(w; b)$
13. **end for**
16. **end for**
17. return w to server

2.2 Heterogeneous clients and non-IID data

Device heterogeneity is one of the inevitable challenges for FL. The FL system is built on a large number of mobile devices, IOT devices, IOV devices, etc. These devices have differences in hardware (CPU, memory, etc.), network conditions (4G, 5G, WiFi or Ethernet, etc.), power supply and available idle time, which will result in different computing, storage and communication capabilities. There is a problem that the training speed and training time between heterogeneous devices are too different. At the same time, due to unstable network conditions, some devices will often be offline. The FL system cannot apply a reasonable synchronous training protocol, and device heterogeneity simultaneously causes the vulnerability of the whole system, and the overall efficiency is lower compared to distributed ML.

The usage habits of individuals or organizations are as important as device heterogeneity in causing Non-IID data to arise. Although FedAvg's performance on independent and identically distributed (IID) data has been proved to be similar to centralized model training, the data owned by different devices are Non-IID and a large number of studies have shown

that FedAvg's performance will inevitably decline [8] for Non-IID data. The main reason for the decline is that the local models trained by different clients on Non-IID data will converge to different models, and the difference between the global model and the local model becomes larger by uploading the models and averaging them. And when the clients receive the global model, they will start retraining again from an unsatisfactory model, which leads to the decrease of the convergence speed of the entire FL system and the deterioration of the global model performance.

Consider supervised learning, for client k, remember that its local data distribution is $p_k(x, y)$, x is the sample, y is the label, and Non-IID means that p_k is different for different clients. Non-IID data classification [17] has attribute skew (referring to the partial overlap or non-overlap between data attributes on different clients), label skew (referring to the different label distributions on different clients, with both label distribution bias and label preference bias), time skew and quantity skew, etc. Under FedAvg, the possible consequences of client drift [18] on model training are shown in Fig. 1a. When the data is IID, the average model w_{t+1} is close to the global optimal solution w^* because it is equidistant from the local optimal solutions w_1^* and w_2^*. However, when the data is Non-IID, the global optimal solution w^* may be closer to a local optimal solution, resulting in a deviation between the average model w_{t+1} and the global optimal solution w^*.

We have also proved through experiments that in the FedAvg algorithm, Non-IID data will slow down the convergence speed of FL. We used PyTorch to train a shallow convolutional neural network (CNN) model on the Fashion-MNIST dataset. The experimental results are as shown in Fig. 1b. It can be clearly seen that compared with the IID setting, the accuracy under Non-IID data is lower. And the curve of global model accuracy with the gradual improvement of communication rounds is more unstable, there are violent fluctuations. We will discuss the impact of Non-IID data in more detail in Sect. 4.

2.3 Model interpolation

Model interpolation [11] is a very effective personalized FL method. It balances generalization and personalization by combining the client local model M_l and the global model

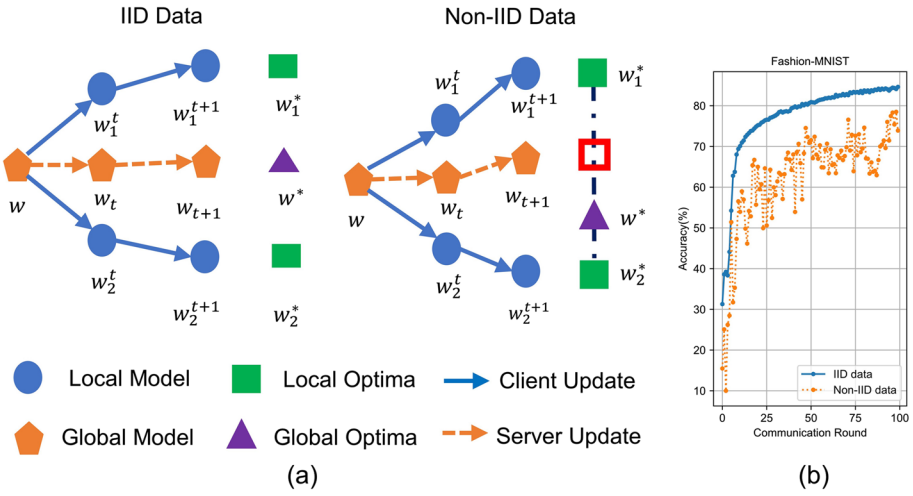

Fig. 1 Effects of IID and Non-IID data on FL

M_c. As shown in the following formula, the degree of personalization can be determined by adjusting the parameter λ. When $\lambda \to 1$, the combined model is more biased towards the local model, and when $\lambda \to 0$, the model is more biased towards the global model.

$$\lambda \cdot M_l + (1 - \lambda) \cdot M_c \tag{6}$$

Model interpolation is proved [11] to have the same communication cost and security as training a single model.

We use the Bagging to realize the model interpolation in this paper. Bagging [19] (bootstrap aggregating) algorithm is a parallel integrated learning method. Bagging reduces generalization error by combining several models. The main idea is to sample several subdatasets from the data set through bootstrap sampling, then train a basic learner on each subdataset, and finally combine these basic learners. When predicting output, Bagging usually uses simple voting method for classification task and simple average method for regression task. Bagging has been proved to be effective in reducing variance [20], so it is more effective in neural networks and other learners that are easily disturbed by samples.

We will use the shallow neural network as our basic learner. The shallow neural network does not need huge computing resources as the deep neural network. The deep neural network training is slow and bulky, while the shallow neural network training is fast and lightweight. It is more suitable for small devices such as mobile devices and IOT devices, and does not need to occupy a large amount of communication bandwidth.

Combined with Bagging algorithm, we extend model interpolation to basic learners, so as to realize potential information transfer between different client clusters, and realize personalized FL by setting parameters λ.

2.4 Risk of information leakage in smart home

In smart homes, almost all intelligent IoT devices are monitored and used through mobile phones [21] or web pages. Therefore, smart home systems generally include intelligent IOT devices, servers designated by manufacturers for cloud services, mobile phones and other terminal devices. All devices follow this working mode, the IOT device collects information (such as images, audio, etc.) from the home, uploads it to the manufacturer's server, performs intelligent learning in the server, or transmits the data to the user's terminal device for monitoring and use by the user [22]. However, the process of uploading data from smart home devices to the server is transparent to users, which may lead to privacy disclosure risk [23].

FL is one of the solutions to this kind of information leakage problem, but it also needs to face the problem of device heterogeneity of smart home and the problem of statistical heterogeneity of data generated by different families.

3 pFedCAM based on clustering and model interpolation

We design a server-client FL framework to execute pFedCAM. First, we will introduce the components of the framework, which is the basis to ensure the good operation of the algorithm. According to the above description, clustering clients is good for improving the efficiency of FL system. We will give a simple client clustering protocol in the second part of this section to ensure that clients can be grouped by similarity. In the third part, we will put forward the key ideas of our personalized FL mechanism. In the end, we will elaborate on the whole workflow.

3.1 Architecture design

We focus on the central HFL, so the framework includes a large number of clients (smartphones, IOT devices, etc.) and a server used to coordinate the training process of various participants. As shown in Fig. 2, on the left are the clients to participate in FL, and some clients have potential similarities. The right is the server responsible for managing the clients and model processing. We purpose two strategies: Simple Clustering Protocol and Integration Model Generation Strategy. They are respectively responsible for the clustering of clients and the generation of personalized models.

The client is an intelligent device, that is, there is a micro computing system, which needs hardware resources such as CPU, RAM, storage, network, and a software environment that can perform ML training and detecting.

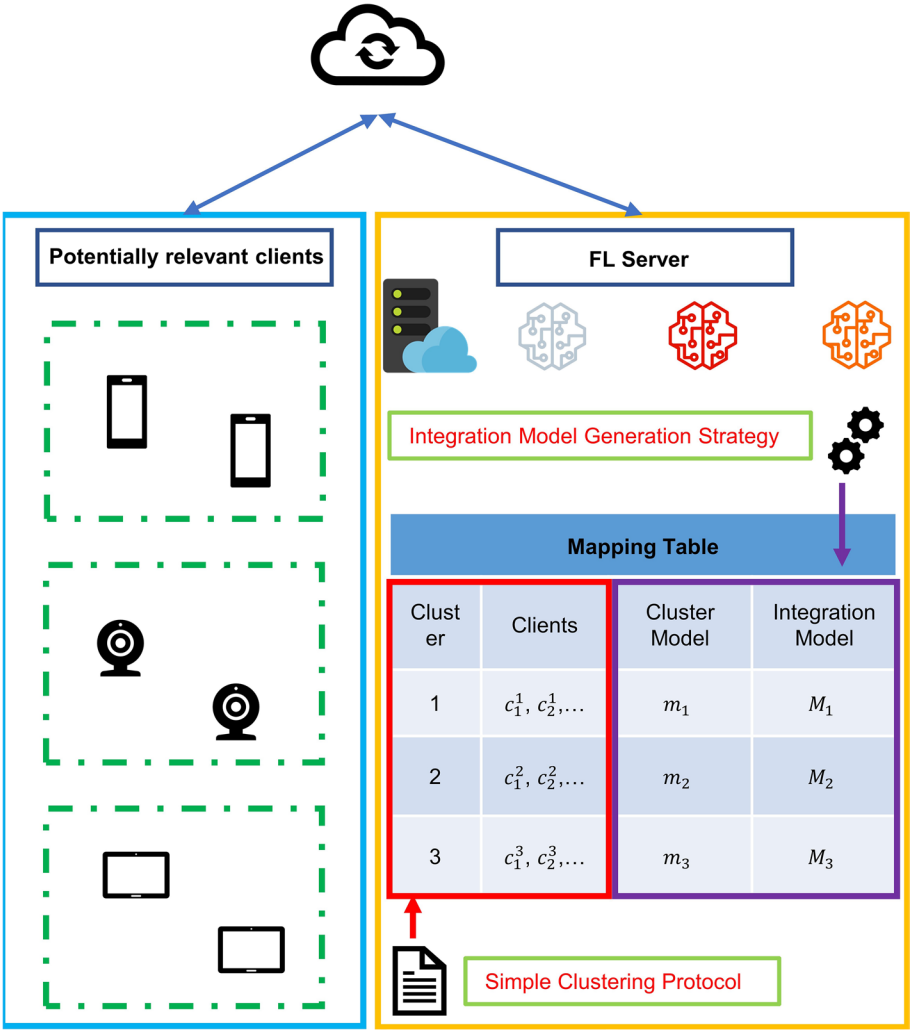

Fig. 2 The Components of pFedCAM

The server is the core of FL system. In our framework, the server executes two strategies: Simple Clustering Protocol and Integration Model Generation Strategy. In addition, a mapping table is saved and maintained. Simple Clustering Protocol is a basic protocol for server and clients to interact and schedule the clients. It clusters the clients reasonably according to the hardware resources of the clients and the data distribution information that does not involve privacy. The Integration Model Generation Strategy responsible for model personalization, it integrates different models generated by various clusters into global models according to the personalization mechanism below. We do not only generate one global model here, but generate global models with the same number of client clusters. The Integration Model Generation Strategy is also responsible for sending these models to the corresponding clients, and the clients will use these models to complete their own reasoning and prediction tasks. A Cluster-Clients-Cluster Model-Integration Model Mapping table is also saved and maintained in the server. It records the corresponding relationship between the clusters and the clients, and saves the model generated by each client cluster and the integration model processed by the personalized mechanism. Among them, the Simple Clustering Protocol is responsible for generating and maintaining the Cluster-Clients relationship in the table, and the Integration Model Generation Strategy is responsible for the Cluster Model-Integration Model part.

3.2 Simple clustering protocol

According to the above discussion, we will cluster the clients before training. We can think that the clients in the cluster are similar in data distribution and hardware resources, while the clients between clusters are more different. Clients in the same cluster can enhance each other's processing ability for similar target tasks, which can be regarded as FL under IID data settings. The differences between different clusters can strengthen the generalization ability. For example, the client of cluster A can learn other data distribution information from the client of cluster B. This is why we cluster clients.

Protocol. Simple Clustering Protocol

Initialization: Clients check in with the FL server and send the resource information S_i to the server.

Server: Initialize $t_{reset} = 0$, number of clusters $N_{cluster}$, clients redistribution time T_{reset}, $G = \{G_1,...,G_g,...,G_{N_{cluster}}\}$ with KMeans and S_i, where $G_g = \{c_i^g\}$.

1. **while 1:**
2. **while** $t_{reset} < T_{reset}$:
3. FL's procession
4. **if** a new client c_{new} check in with the server **do**:
5. $c_{new} = c_{new}^g$ where the g is the index of G_g, $s.t.$ minimize $d(S_{new},G)$
6. **if** $\exists G' \in G, G' = \emptyset$ **do**:
7. $t_{reset} = T_{reset}$
8. **break**
9. $t_{reset}++$
10. $t_{reset} = 0$
11. make the clients to upload resource information and reclustering.

We have designed a Simple Clustering Protocol to ensure that clients are grouped by similarity. The pseudocode of the protocol is given below. First, clients who intend to participate in FL will register with the FL server. The server will send a message to the clients asking the clients to upload resource information. After receiving the message, the client will send the statistical resource information to the server. The resource information to be uploaded by the client c_i, $i \in [1, K]$ includes the computing resource information and data distribution information. Our method is very flexible. The clients can only upload information about distribution (e. g. the proportion of the largest class in the local training data) to protect privacy, or upload sufficient hardware information and additional data information to make clustering more accurate. In this paper, we use the following information. The computing resource information includes the hardware information, like the CPU frequency, the estimated available CPU time, and the estimated available RAM, etc. The data distribution information includes the total amount of data D_i available for training, and the proportion of the class with the largest proportion in the total quantity of samples, B_i. Finally, the server will calculate the time from sending the message to receiving the client information as the client response time T_i^{Re}, which represents the network status of the client. The server calculates the score sequence S_i of the client through the maximum and minimum standardization method. S_i is used as the feature sequence for KMeans clustering, and the cluster number $N_{cluster}$ is determined by the server.

Since the client resources change dynamically, we set a client redistribution time T_{reset}, which is a fixed value, or arrive immediately after the number of clients in a client cluster is zero. After T_{reset} passed, the server will perform the query-clustering process again. The server will save the information of the cluster to which each client belongs $c_i^g, i \in [1, K]$, $g \in [1, N_{cluster}]$, that is, generate and maintain the Cluster-Clients Mapping table. If a new client c_{new} wants to join FL, the server divides it into the nearest cluster according to the Euclidean distance $d(S_{new}, G_{\sim})$ between the client's resource information S_{new} and each cluster center, and normally participates in FL until the next client redistribution time.

3.3 Integration model generation strategy

Considering the personalization mechanism, if customize the model for each client, we need to meticulously design the model structure and the cooperation mechanism between clients. The method is lack of scalability. Our idea is to generate a personalization model for each client cluster, and the clients in each client cluster will use the personalization model of the cluster to which they belong. We use Bagging to implement the concept of model interpolation. pFedCAM extends the model interpolation method from the interpolation between the global model and the local model to the client clusters, and generates a personalized model by combining $N_{cluster}$ cluster models with different weights into integrated models.

$G_g, g \in [1, N_{cluster}]$ are client clusters generated by the Simple Clustering Protocol. At the beginning, the server sends the basic ML model m_{base} to the clients. Here, we use the shallow neural network as the basic model of pFedCAM to better adapt to different software and hardware environments. Then execute the FedAvg algorithm in the cluster G_g to generate the cluster average model m_g. In other words, in cluster G_g, after the clients are trained with m_{base}, the server randomly selects at least one client to participate

in cluster model aggregation. At this time, the server contains $N_{cluster}$ cluster average models.

Then we generate a personalization model for each cluster, which will be applied to each client in the cluster. Next, we explain the personalization mechanism by establishing the relationship between Cluster Model and Integration Model in the mapping table in the FL server. For cluster G_g, the personalization integrated model M_g which really used by clients is constructed by model interpolation and Bagging,

$$M_g = \sum_g \lambda_g m_g, g \in [1, N_{cluster}] \tag{7}$$

where λ_g is the weight of sub-model m_g. Taking $N_{cluster} = 3$ as an example, as shown in Fig. 3a, each personalized integration model consists of all cluster models. The server will maintain a $N_{cluster} \times N_{cluster}$ weight matrix W like Fig. 3b. The row vector is the weight sequence of each sub-model of m_g, and the column vector is the weights of sub-model m_g in different M_g. We define $W = [w_{ii}]$ as a symmetric matrix with.

$$w_{ii} \geq \sum_{j=1, j\neq i}^{N_{cluster}} w_{ij}, w_{ii} > 0, i = 1, 2, 3, \ldots, N_{cluster} \tag{8}$$

That is, in the personalized model m_g of cluster G_g, the cluster average model m_g of this cluster accounts for the largest weight. Here, we set w_{ii} to 0.5 according to experience to ensure that m_g can play a leading role in M_g. Even if the clients are only divided into two clusters, it can be ensured that when the integrated model M_g is used, the reasoning result of m_g will occupy a dominant position in the final result. For the weights of other sub-models, for example, m_k, in integrated model M_g, we use a distance dependent function to define the weight of m_k in M_g:

Cluster Model	Integration Model
m_1	M_1
m_2	M_2
m_3	M_3

(a)

$$\begin{cases} M_1 = \lambda_{11} m_1 + \lambda_{12} m_2 + \lambda_{13} m_3 \\ M_2 = \lambda_{21} m_1 + \lambda_{22} m_2 + \lambda_{23} m_3 \\ M_3 = \lambda_{31} m_1 + \lambda_{32} m_2 + \lambda_{33} m_3 \end{cases}$$

(b)

$$W = \begin{pmatrix} \lambda_{11} & \lambda_{12} & \lambda_{13} \\ \lambda_{21} & \lambda_{22} & \lambda_{23} \\ \lambda_{31} & \lambda_{32} & \lambda_{33} \end{pmatrix}$$

(c)

$$W = \begin{bmatrix} 0.5 & 0.22 & 0.28 \\ 0.28 & 0.5 & 0.22 \\ 0.28 & 0.22 & 0.5 \end{bmatrix}$$

Fig. 3 Example of weight matrix

$$w_{k,g,k\neq g} = 0.5 \times \frac{1/d_{k,g}}{\sum_{i=1,i\neq g}^{N} \frac{1}{d_{k,i}}} = \frac{\sum_{i=1,i\neq g}^{N} \frac{1}{d_{k,i}}}{2 \cdot d_{k,g}} \tag{9}$$

where $d_{k,i}$ is the Euclidean distance between cluster i and cluster k computed by the information using in clustering steps. That means, the client clusters closer to this cluster has a higher weight in the model M_g. Taking $N_{cluster} = 3$ as an example, Fig. 3c shows the specific value of the weight matrix settings in pFedCAM. Except for the models of this cluster, the average models of other clusters share a weight of 0.5.

The clients in cluster G_g will use the integrated model M_g for prediction or classification tasks. We borrow the idea of Bagging algorithm, but we do not use the simple voting method, but use the weighted probability addition. As shown in Fig. 4, taking the n-Class classification task as an example, the clients in the cluster G_g use the integrated model M_g, assuming that the sample to be classified is x, the result predicted by the cluster model m_g (sub-model ofM_g) is$P_g = [p_{g,1}(x), p_{g,2}(x), ..., p_{g,n}(x)]$, where $p_{g,i}(x)$ is the probability that the predicted category is$i$, then the final classification result is

$$M_g(x) = \sum_{i}^{N_{cluster}} \lambda_i P_i(x) \tag{10}$$

We just need to choose the one with the highest probability.

The model used by the client for training is different from that used for tasks. The high proportion of the average model of this cluster in the integrated model ensures that the prediction with similar data distribution is strengthened, while the results of adding other clusters can strengthen its generalization ability. The reason why the cluster models are not merged into one model here is that we need additional information in the reasoning results of other cluster models.

In order to avoid the network pressure caused by the server pushing the integrated models M_g to all clients in each round of communication, we push the model only when we reach the client redistribution time T_{reset} in the Simple Clustering Protocol.

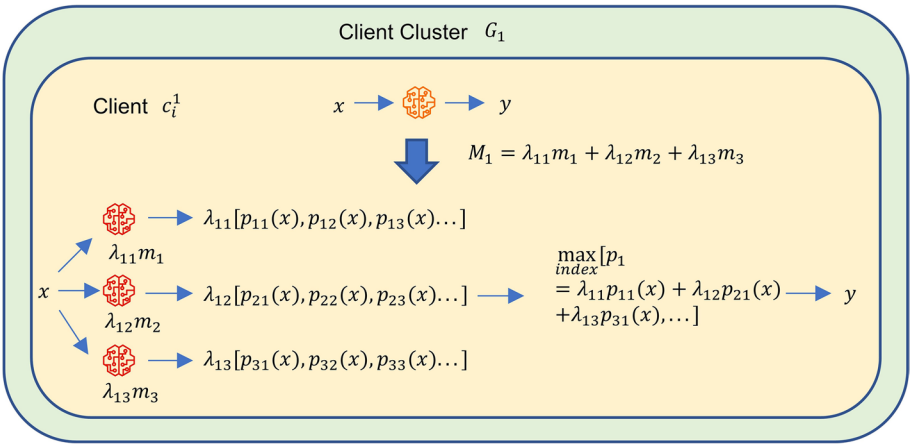

Fig. 4 The details of the personalized integration model used by the client

3.4 Workflow of pFedCAM

Figure 5 shows the process of pFedCAM in a FL training round. It follows the following steps:

Step 1: Clients interested in participating in FL check in with the FL server.

Step 2: The Simple Clustering Protocol in the FL server starts to work. The server and the clients follow the Simple Clustering Protocol for message interaction. The server divides the clients into $N_{cluster}$ clusters $G_1, G_2, ..., G_g, ..., G_{N_{cluster}}$ and marks each client as c_i^g. The FL server will generate an maintain the Cluster-Clients mapping relationship.

Step 3: The FedAvg algorithm is executed in each client cluster. After receiving the basic model m_{base} from the server, the clients use local data for model training. When the client model is generated, at least one client is randomly selected from each client cluster according to the set proportion to upload the model parameters to the server. The server will aggregate the models and obtains the cluster average model m_g after receives the model from the client. If the time not achieve theT_{reset}, the server will push the cluster average model to the clients in the corresponding client clusters and turn to *Step 5*. If time comes T_{reset}, turn to *Step 4*.

Step 4: In this step, pFedCAM uses Integration Model Generation Strategy to generate the personalized global models. According to the personalization mechanism described above, the personalized model M_g is generated and distributed to the clients in the corresponding cluster before the next clustering.

Step 5: The clients will use m_g as the basic model m_{base} for the next round of training. The clients in cluster G_g receives the model M_g and use it to perform classification or reasoning tasks after the *Step 4*.

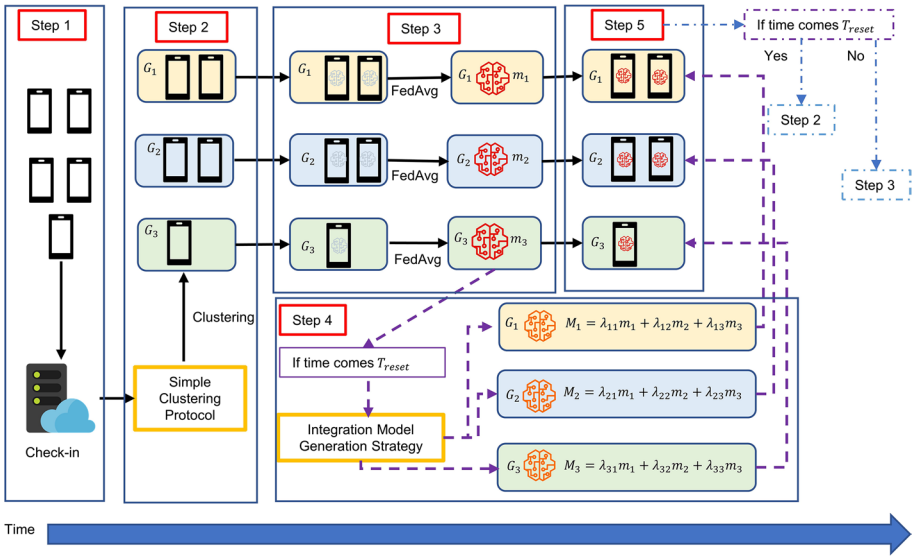

Fig. 5 Workflow of pFedCAM

Steps 3–5 will repeat until the preset target accuracy or communication rounds are reached. After the client redistribution time T_{reset} is reached, *Step 2* will be executed again. When a new client joins the FL system, it will continue to participate in *Steps 3–5* after executing the Simple Clustering Protocol. Finally, we give the pseudo code of pFedCAM algorithm. Our algorithm is proved to be efficient and feasible, especially in the case of Non-IID data in the following Sect. 4.

3.5 Analysis of generalization bound

Here we will give the generalization bound of the pFedCAM by using the same method in Reference [11]. First, we will give the notions and definitions to be used. Let's take a multi-class classification as an example. Let $\mathcal{X}$ denote the input space and $\mathcal{Y}$ the output space. The hypotheses of the form $h \in \mathcal{H} : \mathcal{X} \to \mathcal{Y}$, where $\mathcal{H}$ is a family of such hypotheses. We also denote the l as the loss function over$\mathcal{X} \times \mathcal{Y}$, so the loss of $h \in \mathcal{H}$ for a labeled sample $(x, y) \in (\mathcal{X}, \mathcal{Y})$ is given by$l(h(x), y)$. We will denote by $\mathcal{L}_D(h)$ the expected loss of a hypotheses h with respect to a distribution D over$(\mathcal{X}, \mathcal{Y})$:

$$\mathcal{L}_D(h) = \mathop{\mathbb{E}}_{(x,y)\sim D}\left[l(h(x), y)\right] \tag{11}$$

and by h_D its minimizer $h_D = argmin_{h\in\mathcal{H}}\mathcal{L}_D(h)$. Denote the Rademacher complexity of class $\mathcal{H}$ over the distribution D with m samples $\mathcal{R}_{D,m}(\mathcal{H})$. Let p be the number of clients and q be the number client clusters, client $k \in [1, p]$ has m_k samples, so the number of total samples is $m = \sum_{k=1}^{p} m_k$.

For clients clustering steps, the optimization is

$$\min_{h_1,\ldots,h_q} \sum_{k=1}^{p} \lambda_k \bullet \min_{i\in[q]} \mathcal{L}_{D_k}\left(h_i\right) \tag{12}$$

where $h_i, i \in [q]$ is a particular hypothesis associated with a client cluster. To simplify the analysis and explain the scalability of our algorithm, we use the fraction of samples from each user m_k/m as λ_k instead of distance related parameters mentioned above. The conclusion can be extended to other forms of parameter settings. We use the empirical distributions $\widehat{D}_k$ replace the true distributionsD_k. Let $C_1, ..., C_q$ be the clusters and let m_{C_i} be the number of samples from cluster$i$. Let $\mathcal{C}_i$ and $\widehat{\mathcal{C}}_i$ be the empirical and true distributions of cluster$\mathcal{C}_i$. Let d be the pseudo-dimension of $\mathcal{H}$, then with probability at least$1 - \delta$, the generalization error of the client clustering steps has the bound

$$\max_{h_1,\ldots,h_q}\left|\sum_{k=1}^{p}\frac{m_k}{m} \bullet \left(\min_{i\in[q]}\mathcal{L}_{D_k}\left(h_i\right) - \min_{i\in[q]}\mathcal{L}_{\widehat{D}_k}\left(h_i\right)\right)\right| \le \sqrt{\frac{4plog\frac{2q}{\delta}}{m}} + \sqrt{\frac{dq}{m}log\frac{em}{d}} \tag{13}$$

Algorithm 2. pFedCAM

Input: K clients denoted by c_i, $i \in [1,K]$. The numbers of client clusters $N_{cluster}$. The weight matrix $W = (w_1,w_2,...,w_{N_{cluster}})^T$.

Server:

Initialize client clusters $G = \{G_1,G_2,...,G_{N_{cluster}}\}$ with Simple Clustering Protocol and the models $m = \{m_1,m_2,...,m_{N_{cluster}}\}$, $m_i = m_{base}$. Push the model m_{base} to all clients before training.

1. **for** each round $t = 1,2,...$ **do**
2. **for** $i = 1,2,...,N_{cluster}$ **in parallel do**
3. $m_i \leftarrow \text{FedAvg}(G_i)$
4. **end for**
5. **for** $i = 1,2,...,N_{cluster}$ **do**
6. $M_i = w_i \cdot m$
7. **end for**
8. if time comes T_{reset} do
9. Push the integrated model M_i to the clients in the cluster G_i.
10. **end for**

Client:

11. **for** $G_g \in G$ **in parallel do**
12. **for** client $c_i^g \in G_g$ **in parallel do**
13. Training with m_g
14. Using with M_g
15. **end for**
16. **end for**

FedAvg(G_i):

Using FedAvg within client cluster G_i

For the model interpolation among client clusters, the optimization is

$$\min_{h_1 \ldots, h_q} \sum_{k=1}^{p} \frac{m_k}{m} \mathcal{L}_{D_k} \left(\sum_{i=1}^{q} \mu_{k,i} \bullet h_i \right) \tag{14}$$

Same as the assumption in client clustering steps, we use the fraction of samples from each user m_k/m asλ_k, and $\mu_{k,i}$ is the weight of cluster model h_i in the final model of the client$k$. Let the loss l is L Lipschitz, $\mathcal{H}_{C_i}$ be the hypotheses class for the model of clusterC_i, and let $\widehat{\mu}_{k,i}^*$,$\widehat{h}_i^*$ be the optimal values and$\mu_{k,i}{}^*$, $h_i{}^*$ be the optimal values for the empirical estimates. then with probability at least$1 - \delta$, the generalization error of the client clustering steps has the bound

$$\sum_{k=1}^{p} \frac{m_k}{m} \mathcal{L}_{D_k}\left(\sum_{i=1}^{q} \widehat{\mu}_{k,i}^{*} \cdot \widehat{h}_i^{*}\right) - \sum_{k=1}^{p} \frac{m_k}{m} \mathcal{L}_{D_k}\left(\sum_{i=1}^{q} {\mu_{k,i}}^{*} \cdot {h_i}^{*}\right) \leq 2L\sqrt{\frac{d_q p}{m} log\frac{em}{d_q}} + 2\sqrt{\frac{log\frac{1}{\delta}}{m}} \tag{15}$$

where d_q is the max pseudo-dimension of $\mathcal{H}_{C_i}$.

Two key steps in our algorithm ensure a certain generalization bound error.

4 Evaluation of pFedCAM

We use PyTorch to simulate and implement the pFedCAM algorithm. We evaluated our algorithm on two datasets: Fashion-MNIST [24] and CIFAR-10 [25], and used shallow CNN model as the basic models of the algorithm. The Fashion-MNIST dataset contains 60,000 training samples and 10,000 test samples from 10 categories. The CIFAR-10 dataset contains 50,000 training samples and 10,000 test samples from 10 categories. The CNN model has two 5×5 convolution layers and a 2×2 max pooling layer behind each convolution layer. In order to simulate the real scenario, we set some hyperparameters in the training process as the mapping of computing resources, such as epoch, batch size, samples quantity, etc., and randomize them within a certain range to reflect the heterogeneity of the clients. At the same time, we separately set IID data and Non-IID data of different Non-IID levels for each client in order to measure the effect of the algorithm from multiple angles. We will focus on comparing the efficiency of pFedCAM and FedAvg algorithms under Non-IID. We also discussed the impact of the number of clusters and clients on pFedCAM.

4.1 FedAvg with Different Levels of Non-IID Data

We first tested our model and data settings with the FedAvg algorithm to ensure the effectiveness of the subsequent comparison tests, and showed the great impact of Non-IID data on the FL system. In this experiment, we simulated 50 clients participating in FL, each client's epoch in local training was 3, the batch size was 512, and the number of samples was 3,000. A total of 100 rounds of communication are conducted, and the probability that the client is involved in the selected aggregation in each round of communication is 40%. We simulate the scenario of Non-IID data by setting parameters μ, which indicates the proportion of the category with the largest number of samples in the dataset to the total number of samples, while the remaining samples are evenly divided under other categories. For Fashion-MNIST and CIFAR-10, IID means that the number of samples under each label is the same. $\mu = 0.4$ means that 40% of the samples belong to the same label, while the remaining samples belong to other labels uniformly. $\mu = 1$ means that all samples of this client belong to the same label. The size of μ represents the level of data Non-IID. We set four levels of Non-IID and test the accuracy of FedAvg on IID data.

The results are shown in Fig. 6, it can be seen that with the increase of μ, the accuracy of the model is declining. On Fashion-MNIST, the maximum performance decline is 17.5%, and on CIFAR-10, the maximum performance decline is 48.8%, and the training process becomes more unstable. This is one of the challenges FL faces and our algorithm will mitigate the impact of these challenges.

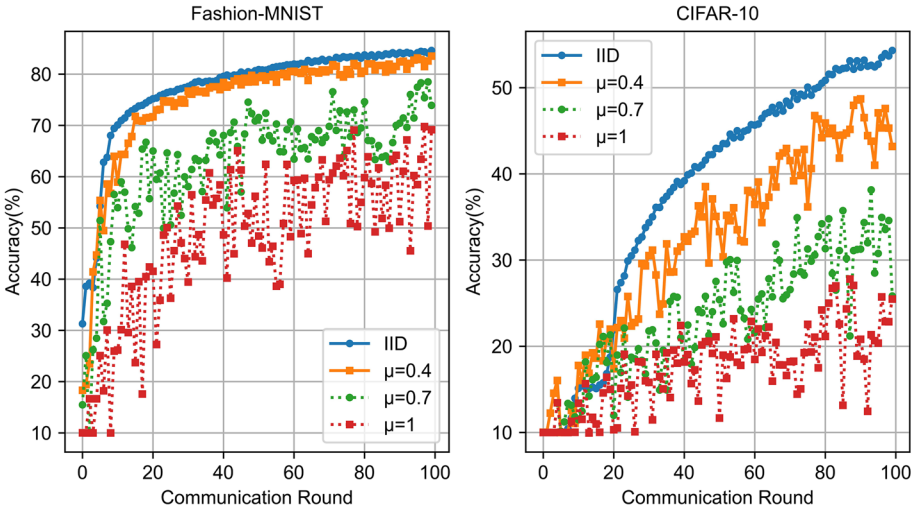

Fig. 6 The influence of different levels of Non-IID data

4.2 pFedCAM with Different Levels of Non-IID Data

Next, we consider the generalization ability and personalization ability of the generated model of pFedCAM. For the measurement of generalization ability, we will test the model accuracy on a public IID dataset. For the personalization ability, we maintain a test set with the same data distribution as the training set on each client. We will measure the average accuracy of the test set on all clients to represent the personalization ability of the model. We set four levels of Non-IID: IID, $\mu \in [0.1, 0.4]$, $\mu \in [0.4, 0.7]$,$\mu \in [0.7, 1]$. In addition to randomizing μ in the interval, we also simulate the heterogeneity of the clients by setting $epoch \in [1, 5]$, $batchsize \in [2, 1024]$ and the total amount of data $D_i \in [1000, 5000]$, which increases the challenge of the experiment. We set the number of clusters to 5 and the number of clients to 100.

On Fashion-MNIST, the experimental test results of the model on IID test dataset are shown in Fig. 7. It can be seen that under different levels of Non-IID data settings, the performance of pFedCAM is more stable, and the accuracy is higher than that of FedAvg, with an average increase of about 7.4%. The test results on Non-IID test dataset, such as Fig. 8, note the slight difference between it and Fig. 7, can improve the

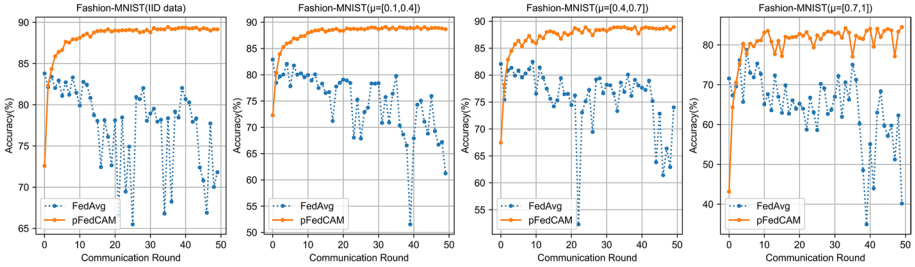

Fig. 7 The accuracy of FedAvg and pFedCAM testing on IID data

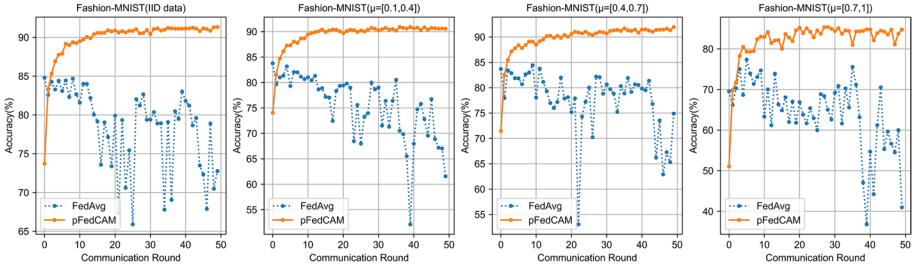

Fig. 8 The accuracy of FedAvg and pFedCAM testing on Non-IID data

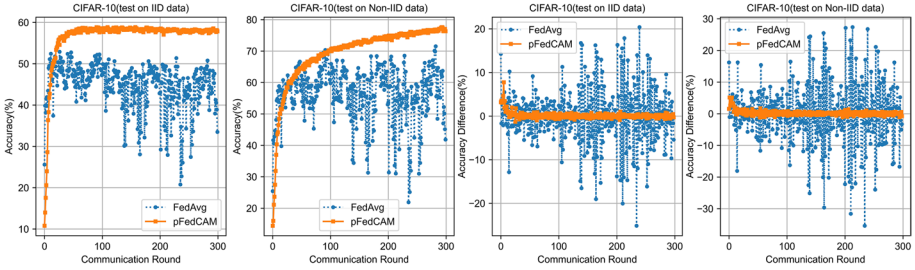

Fig. 9 The accuracy and difference sequence of accuracy of pFedCAM v.s. FedAvg

accuracy by about 10.3% compared with FedAvg. It shows that pFedCAM has better generalization ability and personalization ability than FedAvg.

We also tested the performance of pFedCAM after a long time of communication and interaction, as shown in Fig. 9. We set μ to [0.1, 1] and conducted 300 rounds of communication. Compared with FedAvg, the accuracy of pFedCAM improved about 11.3% on CIFAR-10. We think that in FL, the stability of the training process is also very important, because the client is likely to use the model for prediction or classification during training time on each round. Ensuring the stability of FL will ensure that the performance of the client will not deviate greatly within a certain period of time. In this regard, our algorithm is obviously superior to FedAvg. Two figures on the right in Fig. 9 is the difference sequence of accuracy two figures on the left, indicating the fluctuation range of accuracy with communication rounds. Compared with FedAvg, pFedCAM reduces the standard deviation from 9.89 to 0.80.

Compared with FAVOR [12], a framework for processing Non-IID data using reinforcement learning, shown in Table 1, pFedCAM has faster convergence speed and fewer communication rounds when reaching the predetermined accuracy on CIFAR-10.

Table 1 The number of communication rounds to reach a target accuracy for pFedCAM v.s. FAVOR and K-Center on CIFAR-10

	μ	Rounds	Test accuracy
FAVOR[26]	0.5	50	55%
K-Center	0.5	52	55%
pFedCAM	0.5	13	55%
FAVOR	0.8	61	55%
K-Center	0.8	74	55%
pFedCAM	0.8	27	55%

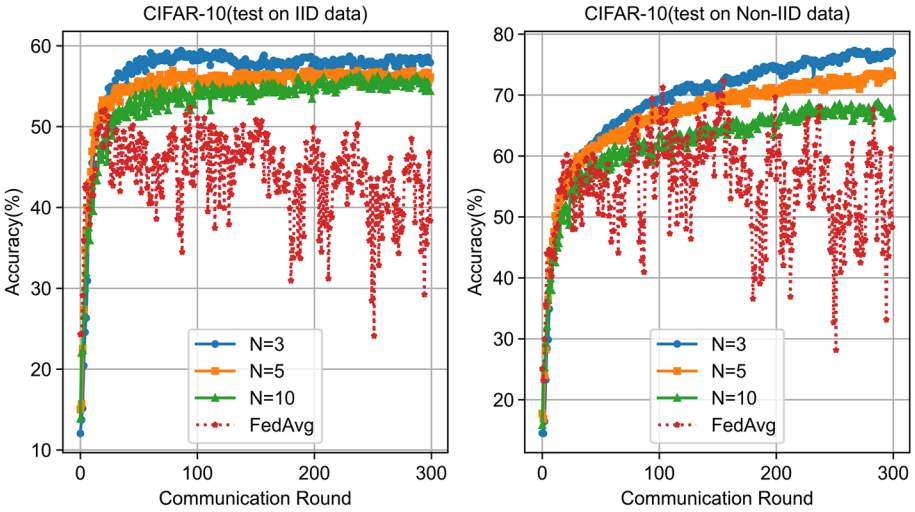

Fig. 10 The influence of different numbers of clusters

4.3 Impact of Different Quantities of Clusters

The number of client clusters also affects the effect of pFedCAM. When $N_{cluster} \rightarrow 1$, pFedCAM will degenerate into FedAvg. When $N_{cluster}$ becomes too large, the algorithm will become meaningless. At this time, the client will download a large number of models in each round of communication, which is undoubtedly a challenge to the client network performance. Therefore, the appropriate number of clusters is also very important for pFedCAM.

Figure 10 shows the impact of cluster number on pFedCAM. When the number of clusters is 3, the effect of pFedCAM is better than that of clusters of 5 and 10. On CIFAR-10, when the test set is IID data, the model accuracy when the number of clusters is 3 is 4.1% higher than that when the number of clusters is 5, and 5.1% higher than that when the number of clusters is 10; when the test set is Non-IID data, the model accuracy when the number of clusters is 3 is 4.4% higher than that when the number of clusters is 5, and 11.0% higher than that when the number of clusters is 10.

4.4 Different Numbers of Clients

We also explored the impact of the number of clients K on pFedCAM. For example, in Fig. 11, the more the number of clients, the higher the accuracy of the algorithm. On CIFAR-10, the accuracy can be improved by 33.4% at most, and the learning process is more stable. We guess that more clients will weaken the impact of Non-IID data globally, that is, the more clients, the more the overall data distribution of all clients will favor IID.

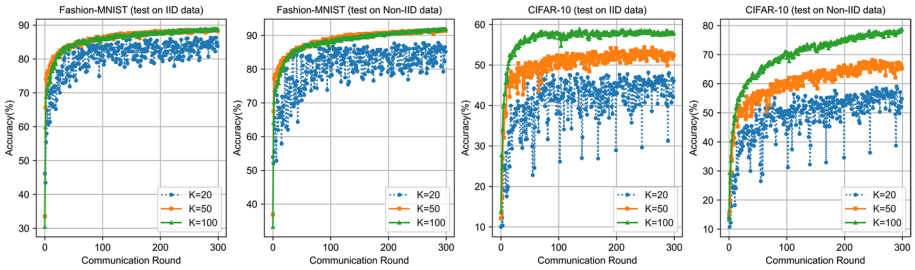

Fig. 11 The influence of different numbers of clients

5 Application in HomeProtect

FL has a very broad application scenario, especially in the face of ML in urgent need of large amounts of data, but the samples are scattered in different organizations with strict privacy protection regulations. We investigated the possibility of FL applying our algorithm in the field of smart home, and elaborated the application prospect of pFedCAM through real environment.

5.1 The architecture of HomeProtect

HomeProtect [13] is a privacy protection framework for smart home proposed by us, mainly based on two methods: FL and PPTrans. The latter is a privacy protection communication protocol designed by us. Traditional smart home devices interact directly with the manufacturer's cloud service. The PPTrans protocol is taken over by the smart gateway to interact with the manufacturer's cloud, and the interactive data is privacy protected through the gateway. Figure 12 shows the architecture of HomeProtect.

HomeProtect is mainly composed of four types of physical components: smart home devices, smart gateways, manufacturer cloud services and personal cloud services. Smart home devices, including cameras and speakers, are responsible for capturing private environmental information data. In addition to the function of managing the incoming and outgoing traffic of the general gateway, the intelligent gateway also has the edge computing capability, which can perform ML model training and detecting without leaking the original data. The smart gateway can also monitor and manage the behavior of smart home devices. The manufacturer cloud service is responsible for aggregation and distribution of ML models uploaded by different intelligent gateways, while maintaining different types of intelligent services. The manufacturer's cloud service and smart gateways constitute the FL system. Personal cloud services store users' data, which users can access and manage remotely. The separate storage and management of the model and the original data provide a certain degree of privacy protection capability.

PPTrans communication protocol distinguishes whether the device has intelligent service capability and whether it has the request to participate in FL by modifying specific fields in DHCP message. For devices willing to participate in FL, the smart gateway will start a Docker containing FL training and reasoning services, accept the data collected by the device for training, and upload the model to the server. The server will execute the pFedCAM algorithm with the model uploaded by smart gateways from other smart homes, and finally send the generated personalized model to the smart gateway for detecting.

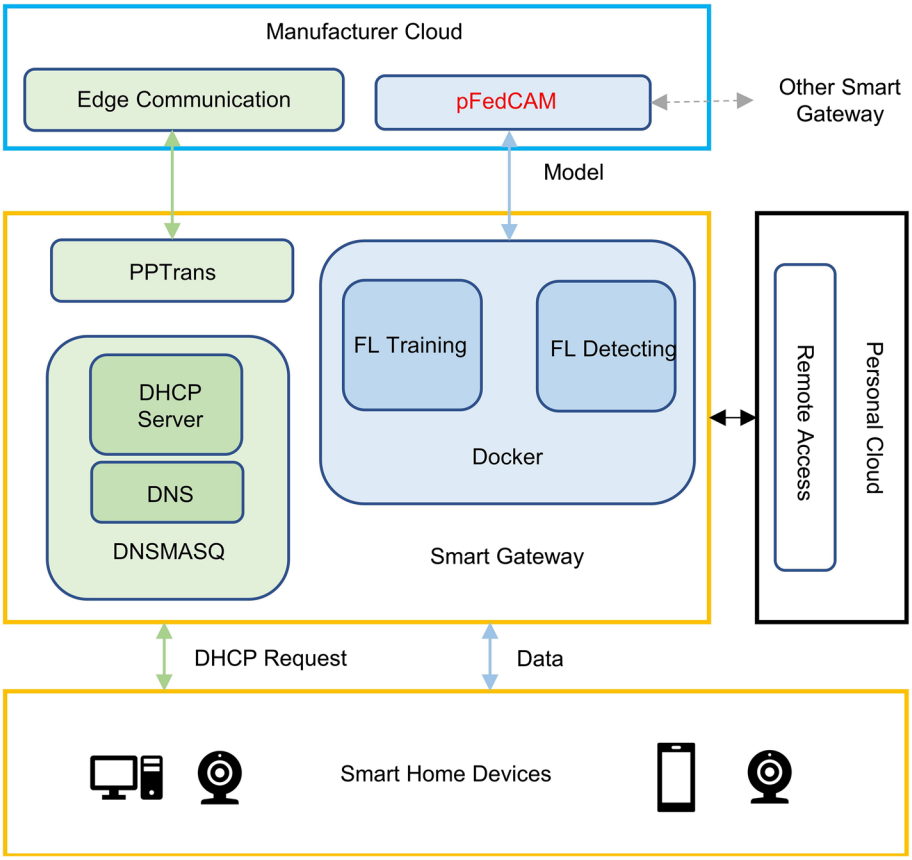

Fig. 12 The architecture of HomeProtect

The framework has strong compatibility and can accommodate various types of devices and perform different FL tasks at the same time. At the same time, it also has scalability, and can run different types and strengths of privacy protection technologies on the smart gateway.

5.2 The application of pFedCAM in HomeProtect

We designed a flame recognition case to verify the functionality of HomeProtect and pFedCAM. The main content of this case is that the smart gateway can collect video from cameras in homes or factories, and then identify whether there is flame in the environment through ML detecting. If there is a fire hazard, it will warn users through cloud services to avoid serious consequences.

Our experimental devices, as shown in Fig. 13a, includes several intelligent gateways with the aforementioned functions and cameras belonging to them. These devices are not from the same manufacturer, so they are heterogeneous devices. The dataset we use is divided into two parts. One is the open-source flame dataset, which is used to initialize the model; the other part comes from the flame images obtained by the local smart gateway

Fig. 13 The deployment of HomeProtect in the real environment in the case of flame recognition

using the initialization model for real-time detecting, and the data has been confirmed by the user. The open-source dataset (https://github.com/gengyanlei/fire-smoke-detect-yolov4) contains 2,059 pictures of flame, we choose 1800 pictures as the training data and use another 200 pictures to test the model. The addition of users' local data will lead to the statistical heterogeneity of data from different smart homes, but it will also help to personalize the model, thus improving the reasoning accuracy of the local model. We use YOLOv5 (https://github.com/ultralytics/yolov5) as the basic model, which has few parameters, but has a fast detection speed and high detection accuracy in the target detection task.

The smart gateways and the manufacturer cloud service together used pFedCAM to execute the FL process. After the initial training, we created different types of flames in the front of the camera, and then added the screenshots contain flame with personal privacy to the training data set after the user's confirmation. When we created an artificial flame in front of the camera again, the remote monitoring website marked the fire source and gave confidence, as shown in Fig. 13b, which proves that our HomeProtect privacy protection framework and pFedCAM algorithm have passed the functional test and performed well.

6 Related work

In this section, we will introduce some recent research progress of FL and personalized FL on Non-IID data and heterogeneous clients.

For Non-IID data and heterogeneous clients, we can generally deal with them from three aspects: data-based method, algorithm-based method and system design. The research on personalized FL is generally inseparable from the processing of Non-IID data. The method of personalized FL is generally based on global model personalization or direct learning of personalized FL model.

Data-based methods are generally realized through data sharing or data expansion. Zhao et al. [8] improves the training of Non-IID data by creating a data subset of all classes shared by all clients and distributing it to clients, and forming a training set together with local data. Experiments show that if 5% of the shared data is distributed, the accuracy on CIFAR-10 dataset can be improved by 30%. Reference [26, 27] the same idea is to change the data distribution by sharing data. The disadvantage of data sharing is that it violates

the principle of data privacy protection to a certain extent. Data augmentation is a technology to increase the diversity of training samples through random sample transformation or knowledge transfer, which can be used to alleviate the Non-IID data problem. Astraea [28] is a self-balance FL framework. The server collects the label sample quantity information of the clients before training, and generates samples for the clients based on this information to alleviate the local Non-IID situation. There are also some methods to generate data or transfer knowledge by Generative Adversarial Networks (GAN) [29] and knowledge distillation [30].

Algorithm or model-based methods generally include local fine-tuning, personalization layer, multi task learning, etc. Wang et al. [31] adopted the strategy of local fine-tuning. After receiving the global model from the server, the client uses local data to make personalized fine-tuning of the model, thus alleviating the impact of Non-IID data and realizing a certain degree of personalization. It is also another feasible method to combine the global model with the local model. The gap between the local model and the global model can be reduced to the greatest extent through the regularization method. Hanzely et al. [32] designed a new cost function with regularization term to balance the global model and the local model. The personalization layer, as the name suggests, is that the model of each client consists of two parts, the basic layer participating in the FL process and the personalization layer used for personalization. Only the basic layer will be uploaded to the server for aggregation of the global model. FedPer [33] is a typical FL algorithm added with personalization layer. Experiments show that the accuracy of FedPer on Non-IID data is higher than FedAvg, and even the accuracy on Non-IID data is higher than IID data. HeteroFL [34] generates models with different structures for heterogeneous clients, challenges the underlying assumption of existing work that local models have to share the same architecture as the global model and can enable the training of heterogeneous local models with varying computation complexities and still produce a single global inference model. MOCHA [35] is a federal multi task learning framework. It considers the communication cost, dropped line and fault tolerance in FL for the first time, and generates an independent but relevant model for each client. The results show that the FL process is significantly accelerated.

The system design-based method can mitigate the impact of Non-IID data and realize personalized FL by elaborately designing the overall FL architecture or customizing the model for each client. When it is known that there are significant differences in hardware or tasks between different clients, it is not the best practice to use the server client FL architecture to train the global model. It is natural to cluster clients and train models for homogeneous clients. FL + HC [36] algorithm clusters the clients after the first few rounds of FL, and the clustered client clusters are trained independently. This not only takes advantage of the rapid convergence of the global model due to the joint learning of a large number of clients at the beginning of FL, but also ensures the personalization of the internal model of each cluster in the later stage. FedAMP and HeurFedAMP [37] are novel attentive message passing mechanism to significantly facilitate the collaboration effectiveness between clients without infringing their data privacy which enables similar clients to have stronger collaboration than clients with dissimilar models, and this mechanism significantly improves the learning performance. In framework FedProto [38], the clients and server communicate the abstract class prototypes instead of the gradients, and aggregates the local prototypes collected from different clients, and then sends the global prototypes back to all clients to regularize the training of local models. The training on each client aims to minimize the classification error on the local data while keeping the resulting local prototypes sufficiently close to the corresponding global ones. FAVOR [12] is an experience driven control

framework, which implements a reward mechanism based on deep Q-learning, selects a subset of devices that can maximize the reward in each communication round, encourages the improvement of accuracy, offsets the deviation caused by Non-IID data, and accelerates the training speed.

7 Conclusion

In this paper, we introduce a personalized FL algorithm pFedCAM based on client clustering and model interpolation. pFedCAM extends the method of model interpolation in personalized FL from global model and local model to interpolation between client clusters, and realizes personalized FL to a certain extent by generating different models for clients in different clusters. Our experiments show that pFedCAM is better than FedAvg and FAVOR [12] in generalization ability and personalization ability, and has better stability and higher convergence rate in the training process. By using the pFedCAM algorithm in the case of flame recognition in HomeProtect, our smart home privacy protection framework, we have verified its practicality in the real environment.

Our future work will focus on reducing the traffic of FL using pFedCAM algorithm, and give a more flexible way to generate cluster weight matrix and cluster with as little clients' information as possible.

Acknowledgements This work is supported by Key-Area Research and Development Program of Guangdong Province (No.2019B010137005).

Author contributions Yang and Liu provided the research ideas of the manuscript. Yang completed the investigation, the design and validation of experimental methods, and the writing of the original draft. Liu is responsible for project supervision and guidance. Zhang and Zhou completed the review and editing of the manuscript.

Data Availability The data that support the findings of this study are available from the corresponding author upon reasonable request.

Declarations

Competing interests The authors declare no competing interests.

References

1. McMahan, B., Moore, E., Ramage, D., Hampson, S., & Arcas, B. A.: Communication-efficient learning of deep networks from decentralized data. In Artificial intelligence and statistics. PMLR. pp. 1273–1282 (2017)
2. Hard, A., Rao, K., Mathews, R., Ramaswamy, S., Beaufays, F., Augenstein, S., ... & Ramage, D.: Federated learning for mobile keyboard prediction. arXiv preprint arXiv:1811.03604 (2018)

3. Kaissis, G.A., Makowski, M.R., Rückert, D., Braren, R.F.: Secure, privacy-preserving and federated machine learning in medical imaging. Nat. Mach. Intell. **2**(6), 305–311 (2020)
4. Warnat-Herresthal, S., Schultze, H., Shastry, K. L., Manamohan, S., Mukherjee, S., Garg, V., ... & Schultze, J. L.: Swarm learning for decentralized and confidential clinical machine learning. Nature, **594**(7862), 265–270 (2021)
5. Xu, Y., Ren, J., Zhang, Y., Zhang, C., Shen, B., Zhang, Y.: Blockchain empowered arbitrable data auditing scheme for network storage as a service. IEEE Trans. Serv. Comput. **13**(2), 289–300 (2019)
6. Xu, Y., Liu, Z., Zhang, C., Ren, J., Zhang, Y., & Shen, X.: Blockchain-based trustworthy energy dispatching approach for high renewable energy penetrated power systems. IEEE Internet Things J **9**(12):10036–10047 (2021). https://doi.org/10.1109/JIOT.2021.3117924
7. Nishio, T., & Yonetani, R.: Client selection for federated learning with heterogeneous resources in mobile edge. In ICC 2019–2019 IEEE international conference on communications (ICC). pp. 1–7 (2019). https://doi.org/10.1109/ICC.2019.8761315
8. Zhao, Y., Li, M., Lai, L., Suda, N., Civin, D., & Chandra, V.: Federated learning with non-iid data. arXiv preprint arXiv:1806.00582 (2018)
9. Sattler, F., Wiedemann, S., Müller, K.R., Samek, W.: Robust and communication-efficient federated learning from non-iid data. IEEE Trans. Neural Netw. Learn. Syst. **31**(9), 3400–3413 (2019)
10. Mohri, M., Sivek, G., & Suresh, A. T.: Agnostic federated learning. In International Conference on Machine Learning, PMLR. vol. 97, pp. 4615–4625 (2019)
11. Mansour, Y., Mohri, M., Ro, J., & Suresh, A. T.: Three approaches for personalization with applications to federated learning. arXiv preprint arXiv:2002.10619 (2020)
12. Wang, H., Kaplan, Z., Niu, D., & Li, B.: Optimizing federated learning on non-iid data with reinforcement learning. In IEEE INFOCOM 2020-IEEE Conference on Computer Communications. pp. 1698–1707 (2020). https://doi.org/10.1109/INFOCOM41043.2020.9155494
13. Chen, B., Liu, Y., Zhang, S., Chen, J., & Han, Z.: FLYFDetect: A Smart Home Privacy Protection Framework via Federated Learning. In 2022 7th IEEE International Conference on Data Science in Cyberspace (DSC). pp. 432–439 (2022). https://doi.org/10.1109/DSC55868.2022.00066
14. Yang, Q., Liu, Y., Chen, T., Tong, Y.: Federated machine learning: Concept and applications. ACM Trans. Intell. Syst. Technol. **10**(2), 1–19 (2019)
15. Roy, A. G., Siddiqui, S., Pölsterl, S., Navab, N., & Wachinger, C.: Braintorrent: A peer-to-peer environment for decentralized federated learning. arXiv preprint arXiv:1905.06731 (2019)
16. Shayan, M., Fung, C., Yoon, C.J., Beschastnikh, I.: Biscotti: A blockchain system for private and secure federated learning. IEEE Trans. Parallel Distrib. Syst. **32**(7), 1513–1525 (2020)
17. Zhu, H., Xu, J., Liu, S., Jin, Y.: Federated learning on non-IID data: A survey. Neurocomputing **465**, 371–390 (2021)
18. Karimireddy, S. P., Kale, S., Mohri, M., Reddi, S., Stich, S., & Suresh, A. T.: Scaffold: Stochastic controlled averaging for federated learning. In International Conference on Machine Learning. PMLR. pp. 5132–5143 (2020)
19. Breiman, L.: Bagging predictors. Mach. Learn. **24**(2), 123–140 (1996)
20. Bühlmann, P., Yu, B.: Analyzing bagging. Ann. Stat. **30**(4), 927–961 (2002)
21. Adiono, T., Marthensa, R., Muttaqin, R., Fuada, S., Harimurti, S., & Adijarto, W.: Design of database and secure communication protocols for internet-of-things-based smart home system. TENCON. pp. 1273–1278 (2017). https://doi.org/10.1109/TENCON.2017.8228053
22. Zhou, W., Jia, Y., Yao, Y., Zhu, L., Guan, L., Mao, Y., ... & Zhang, Y.: Discovering and understanding the security hazards in the interactions between {IoT} devices, mobile apps, and clouds on smart home platforms. USENIX security symposium. pp. 1133–1150 (2019)
23. Plachkinova, M., Vo, A., & Alluhaidan, A.: Emerging trends in smart home security, privacy, and digital forensics. (2016)
24. Xiao, H., Rasul, K., & Vollgraf, R.: Fashion-mnist: a novel image dataset for benchmarking machine learning algorithms. arXiv preprint arXiv:1708.07747 (2017). https://doi.org/10.48550/arXiv.1708.07747
25. Krizhevsky, A., & Hinton, G.: Learning multiple layers of features from tiny images. (2009)
26. Tuor, T., Wang, S., Ko, B. J., Liu, C., & Leung, K. K.: Overcoming noisy and irrelevant data in federated learning. International Conference on Pattern Recognition (ICPR). pp. 5020–5027 (2021). https://doi.org/10.1109/ICPR48806.2021.9412599
27. Yoshida, N., Nishio, T., Morikura, M., Yamamoto, K., & Yonetani, R.: Hybrid-FL for wireless networks: Cooperative learning mechanism using non-IID data. IEEE International Conference on Communications (ICC). pp. 1–7 (2020). https://doi.org/10.1109/ICC40277.2020.9149323
28. Duan, M., Liu, D., Chen, X., Tan, Y., Ren, J., Qiao, L., & Liang, L.: Astraea: Self-balancing federated learning for improving classification accuracy of mobile deep learning applications. IEEE 37th

international conference on computer design (ICCD). pp. 246–254 (2019). http://arxiv.org/abs/1910.10252
29. Goodfellow, I., Pouget-Abadie, J., Mirza, M., Xu, B., Warde-Farley, D., Ozair, S., ... & Bengio, Y.: Generative adversarial networks. Commun. ACM, **63**(11), 139–144 (2020)
30. Wu, C., Wu, F., Lyu, L., Huang, Y., Xie, X.: Communication-efficient federated learning via knowledge distillation. Nat. Commun. **13**(1), 1–8 (2022)
31. Wang, K., Mathews, R., Kiddon, C., Eichner, H., Beaufays, F., & Ramage, D.: Federated evaluation of on-device personalization. arXiv preprint arXiv:1910.10252 (2019). https://doi.org/10.1109/ICCD46524.2019.00038
32. Hanzely, F., & Richtárik, P.: Federated learning of a mixture of global and local models. arXiv preprint arXiv:2002.05516 (2020)
33. Arivazhagan, M. G., Aggarwal, V., Singh, A. K., & Choudhary, S.: Federated learning with personalization layers. arXiv preprint arXiv:1912.00818 (2019)
34. Diao, E., Ding, J., & Tarokh, V.: HeteroFL: Computation and communication efficient federated learning for heterogeneous clients. arXiv preprint arXiv:2010.01264 (2020)
35. Smith, V., Chiang, C. K., Sanjabi, M., & Talwalkar, A. S.: Federated multi-task learning. Adv. Neural Inf. Process. Sys. pp. 4427–4437 (2017)
36. Briggs, C., Fan, Z., & Andras, P.: Federated learning with hierarchical clustering of local updates to improve training on non-IID data. International Joint Conference on Neural Networks. pp. 1–9 (2020). https://doi.org/10.1109/IJCNN48605.2020.9207469
37. Huang, Y., Chu, L., Zhou, Z., Wang, L., Liu, J., Pei, J., & Zhang, Y.: Personalized Cross-Silo Federated Learning on Non-IID Data. AAAI Conference on Artificial Intelligence . 35(7865–7873)pp. 7865–7873 (2021). https://doi.org/10.1609/aaai.v35i9.16960
38. Tan, Y., Long, G., Liu, L., Zhou, T., Lu, Q., Jiang, J., & Zhang, C.: Fedproto: Federated prototype learning across heterogeneous clients. In AAAI Conference on Artificial Intelligence. **36**(8), 8432–8440 (2022). https://doi.org/10.1609/aaai.v36i8.20819