FedPA: Generator-Based Heterogeneous Federated Prototype Adversarial Learning

Lei Jiang, Xiaoding Wang, Xu Yang, Jiwu Shu, *Fellow, IEEE*, Hui Lin, and Xun Yi

***Abstract*—Federated Learning is an emerging distributed algorithm that is designed to collaboratively train the global model without accessing clients' private data. However, heterogeneity of data among clients leads to significant degradation in model performance. Some studies suggest adopting model regularization and using generators to enrich datasets with diverse features can effectively enhance model performance. But current research focuses on regularizing specific modules of the model, failing to achieve regularization across the entire model, and offering limited mitigation of bias from heterogeneous data. Moreover, few methods consider that generators often produce samples with simple features, and the direct use for generating raw data can raise privacy concerns. To solve these challenges, we propose a generator-based heterogeneous federated prototype adversarial learning framework, named FedPA, which combines prototype learning and lightweight generators to achieve regularization of the entire model. Our generators are designed to generate features rather than raw data, and use prototype learning to find the hard features in an adversarial learning manner, thereby improving model performance. Experimental results show that FedPA improves test accuracy by 3.7% compared to state-of-the-art methods, validating that FedPA can effectively mitigate model bias.**

***Index Terms*—Federated learning, Prototype learning, Model regularization, Feature mining, Privacy protection.**

I. Introduction

Federated Learning [1] has been widely studied in many application scenarios due to its privacy-preserving nature. It involves two main steps: (1) The training of local models occurs on the client-side, during which clients upload model updates to the server instead of sending their private data. (2) These updates are then aggregated on the server to generate a global model, which is then sent back to the clients as the initial model for subsequent training. The idea behind this learning paradigm is to keep client-sensitive data safe without transmitting it to the server or other clients.

Note that there is highly heterogeneous data in federated learning, exhibiting non-identically and independently distributed client datasets. Due to significant differences in data distributions among clients, model parameters trained on a single client might not perform well on others, resulting in model bias caused by data heterogeneity. This can lead to unstable convergence and performance degradation. If full-model regularization is not performed and biased local model parameters are directly aggregated into the global model, these issues remain unsolved [2], [3], [4], [5].

To solve these issues, some researchers train generators to learn the data distribution of clients. For example, the classic FedGan [6] directly generates raw data to mitigate the impact of non-identically and independently distributed datasets. However, since it directly generates raw data, it raises some privacy concerns. Similarly, FedDTG [7] adds a classifier on each client to accelerate the learning process of data distribution, but it still does not address privacy concerns. FedGen [8] considers the data privacy issue and suggests using a lightweight generator to produce feature representations instead of raw data. However, local models might struggle to extract effective feature representations for certain classes without any efficient feature extraction guidance that makes full-model regularization difficult. Moreover, since the generator does not mine hard sample features, it only produces low-quality features.

In heterogeneous scenarios, where the global model has difficulty adapting to local data, Personalized Federated Learning (PFL) has gained much attention. PFL tailors the model for each client, as seen in representative works such as [9], [10], [11]. However, a big challenge with PFL is that it requires

Lei Jiang, Xiaoding Wang, and Hui Lin are with the College of Computer and Cyber Security, Fujian Normal University, Fuzhou 350117, China. E-mails: wyqtjl0322@163.com, wangdin1982@fjnu.edu.cn, and linhui@fjnu.edu.cn.

Xu Yang and Jiwu Shu are with the College of Computer and Data Science, Minjiang University, Fuzhou 350108, China, and Jiwu Shu is also with Department of Computer Science and Technology, Tsinghua University, Beijing 100084, China. E-mail: xu.yang@mju.edu.cn, shujw@tsinghua.edu.cn.

Xun Yi is with the School of Computing Technologies, RMIT University, Melbourne 3000, Australia. E-mail: xun.yi@rmit.edu.au.

local models to have the same structures, however model heterogeneity is common in real application scenarios. To address this, researchers have turned to knowledge distillation, originally introduced by Hinton et al. [12], in federated learning to mitigate model heterogeneity [13]. However, it's important to note that knowledge distillation poses another challenge, as it requires an additional proxy dataset.

Notably, FedProto [14] introduces prototype learning to federated learning, the advantages of which is that this method does not require model homogeneity or extra datasets. However, FedProto has limited impact on the classifier component, as it does not provide comprehensive model regularization and does not find hard features. Inspired by FedProto, FedHKD [15] is designed to achieve classifier regularization by aggregating soft labels. However, it does not solve the problem of finding hard sample features as well.

Due to local model bias caused by data heterogeneity, the global model performance deteriorates severely. Our inspiration comes from the idea that model regularization and generating diverse features by a generator can improve model performance. Therefore, we propose to perform full-model regularization under privacy preservation conditions, and meanwhile to explore hard features. Based on the above analysis, we summarize the issues addressed in this paper as follows: ***Full-model regularization***, ***Privacy protection***, and ***Mining hard features***.

After reviewing the aforementioned works, it is evident that most algorithms that use generators [6], [7] mainly focus on directly generating raw data without considering privacy concerns. Although [8] deals with feature generation by the generator, it still lacks full-model regularization. While [14] addresses the issue of the proxy dataset in [13], it also does not achieve full-model regularization. Inspired by [14], [15] considers using soft labels to achieve full-model regularization, however it does not explore hard features.

In this paper, we propose a generator-based heterogeneous federated prototype adversarial learning framework, named FedPA. In FedPA, each client calculates the mean of the features for each class during feature extraction, creating a local prototype representation. These representations are uploaded to the server to create a global prototype representation. Next, a lightweight feature generator is trained on the server. The generator uses the global prototype representation in an adversarial way to help it find hard features. Then, the server sends both the global prototype representation and the generator to the clients. This allows the client to regularize local feature extractor and local classifier, thereby achieving full-model regularization. Compared to the aforementioned methods, FedPA successfully solves three issues simultaneously.

We summarize our contributions of this paper as follows:

- We propose FedPA, a methods that aggregates and creates global prototype representations on the server. This ensures alignment along feature dimensions during local feature extraction on each client, allowing local models to effectively extract features, even for classes with fewer samples.
- We propose adversarial mining of hard sample features. When training the generator on the server, we use the idea of "moving away" from global prototype representations to create hard sample features. Simultaneously, we utilize the classifier of the local model to learn the distribution of model features, ensuring the authenticity of these features. Through this adversarial learning method, we significantly improve the quality of features produced by the generator and the model's robustness as well. To the best of our knowledge, we are the first to use prototype representations in extracting hard sample features among all research works.
- To rigorously demonstrate the effectiveness of FedPA, we have provided a thorough mathematical proof of its convergence and generalization bounds.
- We use the Dirichlet distribution Dir(α) to model data heterogeneity among clients. Experimental results show that FedPA improves test accuracy by 3.7% compared to state-of-the-art methods, thereby proving its effectiveness in reducing model bias caused by data heterogeneity.

The rest of the paper is organized as follows: In Section II, we conduct the literature review. Section III elaborates the FedPA framework, including the design of feature alignment, generator training, and hard feature mining. Section IV provides theoretical analysis of the convergence and generalization bounds of FedPA. We then present experimental

results and discussions in Section V. Finally, we conclude this paper in Section VI.

II. Related Work

In this section, we present related work on mitigating data heterogeneity in federated learning.

A. Heterogeneity Federated Learning

The issue of data heterogeneity in federated learning significantly affects its performance, which has drawn much attention from researchers. One of the solutions is to correct the bias of local models, as seen in FedProx [16], which achieves this by introducing an L2 approximate regularization term. FedNova [17] introduces a normalized averaging method to eliminate objective inconsistencies and achieve fast convergence. FedEnsemble [18] uses ensemble training in federated learning by collecting predictions from all clients. FedNTD [19] effectively regularizes local models by using global model predictions on locally held-out data for non-true classes. FedCAD [20], inspired by FedNTD, uses auxiliary data to assess the confidence of the global model on each class, enabling dynamic, adaptive adjustments of the global model's impact on local models. Unlike previous approaches, FedFTG [21] allows local models to use free-data knowledge distillation to adjust the global model, thus avoiding direct aggregation impact.

In recent years, Personalized Federated Learning (pFL) methods have gained more attention due to their ability to address statistical data heterogeneity [22]. Some examples of pFL methods include LG-Fed [9], FedPer [10], and CD2-pFed [11]. The first two algorithms personalize the neural network at the bottom and top layers, respectively, while CD2-pFed achieves personalization at each layer of the neural network by decoupling channels and personalizing them proportionally. FedNH [23] shares only the feature extraction part and overcomes data heterogeneity by imposing semantic adjustment of prototype relationships. Other research aimed at overcoming data heterogeneity includes methods such as Scaffold [24], MOON [25], and FedMix [26].

B. Generator in FL

Generative Adversarial Networks (GANs) [27] are a class of deep learning models comprising two main neural networks: the Generator and the Discriminator. Their primary objective is to generate data that closely matches the distribution of real data. Researchers also tackle data heterogeneity using GANs. For instance, FedGan [6] utilizes GANs within the federated learning framework to address data heterogeneity. FedDTG [7] deploys generators and discriminators on the client side and adds a classifier to assist the generator to learn the data distribution rapidly. FedGen [8] addresses some privacy concerns by training the generator on feature representations for each class instead of using raw data. The generator is then deployed to each client to aid local models in feature classification, thus enhancing privacy protection to some extent.

Note that generator-based solutions [6], [7] primarily address the impact of data heterogeneity by directly generating raw data. However, this type of method often raises privacy concerns. In [8], although privacy protection is addressed, the absence of hard feature mining leads to the the quality degradation of the generated features.

C. Prototype Learning in FL

Prototype learning that calculates the mean feature representation for each class is very good in tasks like few-shot learning and one-shot learning because it efficiently use limited sample information for classification. This indicates that prototype learning can also be used to reduce data heterogeneity, where some classes may have very few or no samples on certain clients. FedProto [14] proposes that clients not upload model parameters or gradients but only send prototype representations to regulate client training so as to reduce classification errors. FedHKD [15] uses prototype learning's mean feature, together with soft predictions from knowledge distillation, to form super-knowledge. This super-knowledge is sent to each client, effectively reducing performance degradation.

In the prototype-based learning methods mentioned above, FedProto [14] only uses prototype alignment for feature extraction without performing **full-model regularization**. FedHKD [15] introduces soft labels to get full-model regularization, but it not **mine hard features**.

In summary, none of the methods discussed above manage to achieve ***privacy protection, full-model regularization, mining hard features*** at the same time.

Notation	Description
θ	The generator model parameters
ω_g	The global model parameters
ω_i	Local model parameters
Π	Feature extractor parameters
F_Π	The embedding function of the Feature extractor
Φ	Classifier parameters
R_Φ	The embedding function of the Classifier
n^m	The total number of samples of class m in all clients
n_i^m	The number of samples of class m in client i
c_i	Label counter for client i
$\tilde{p}(y)$	The label distribution
z	The noise vector
h	The features after feature extraction or features generated by the generator
$\varphi_i^{t,m}$	The local prototype representation obtained by client i using local data aggregation for class m in round t
φ^{t+1}	Global prototype representations for round $t+1$ (including all classes)

TABLE I: **Notation table** of FedPA Algorithm.

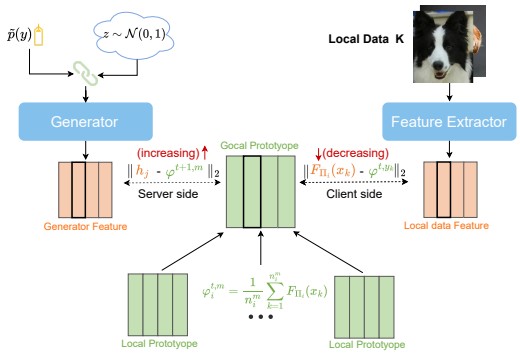


Fig. 1: The global prototype representation can be used to guide feature extraction of the client and mining of hard features, respectively. The bold box in the figure indicates that the prototype representation and feature correspond to the same class.

III. Methodology

We will elaborate the implementation details of the proposed FedPA model in this section, as shown in Figure 2 and Algorithm 1. The algorithm primarily emphasizes the use of global prototype representations to guide both feature extraction on the client-side and feature classification learning with the assistance of the generator. During the training process of the generator, the global prototype representations are also utilized for hard feature mining, as illustrated in Figure 1. We present the symbols used in this paper and their explanations in Table I.

A. Problem Definition

We consider an architecture comprising N clients and one server, where all clients engage in collaborative model training through communication with the server, without the access to their private data. Let each client have its own independent data distributions, denoted by $D_1, D_2, ..., D_N$.

The implementation of the proposed FedPA relies on two following components: **the feature extractor** and **the classifier** (constructed using fully connected layers). The former is responsible for extracting features from the data, while the latter classifies the features from the former output. We parameterize extractor Π_i and classifier Φ_i by $\omega_i = (\Pi_i, \Phi_i)$. Let the embedding functions for these two components be denoted by $F_{\Pi_i}(\cdot)$ and $R_{\Phi_i}(\cdot)$, respectively. Thus, the federated learning process can be expressed as the following optimization problem:

$$\min_{\omega} \mathcal{P}(\omega) = \sum_{i=1}^{N} q_i \, \mathbb{E}_{(x_j,y_j)\sim D_i} L\left(R_{\Phi_i}(F_{\Pi_i}(x_j)), y_j\right) \tag{1}$$

where ω is the global model parameter; L is the loss function, $q_i = {|D_i|}/{\sum_{k\in N}|D_k|}$.

B. Aggregation of Global Prototype Representations

In this subsection, we will introduce the aggregation of local prototype representations and global prototype representations, respectively. First, the data is mapped through the feature extraction layer of the model to a q dimension latent feature space, represented as $(\mathcal{X} \to \mathcal{Z} \subset \mathbb{R}^q)$. Then, the mean of feature vectors of the same class is computed to obtain the prototype representation of this class. Thus, the formula for computing the local prototype representation of client i for class m in the tth global rounds is then given by:

$$\varphi_i^{t,m} = \frac{1}{n_i^m} \sum_{k=1}^{n_i^m} F_{\Pi_i}(x_k) \tag{2}$$

where $\varphi_i^{t,m}$ denotes the local prototype representation of class m computed by client i; x_k represents samples of class m; n_i^m indicates the number of samples belonging to class m on client i.

This article has been accepted for publication in IEEE Transactions on Dependable and Secure Computing. This is the author's version which has not been fully edited and content may change prior to final publication. Citation information: DOI 10.1109/TDSC.2024.3419211

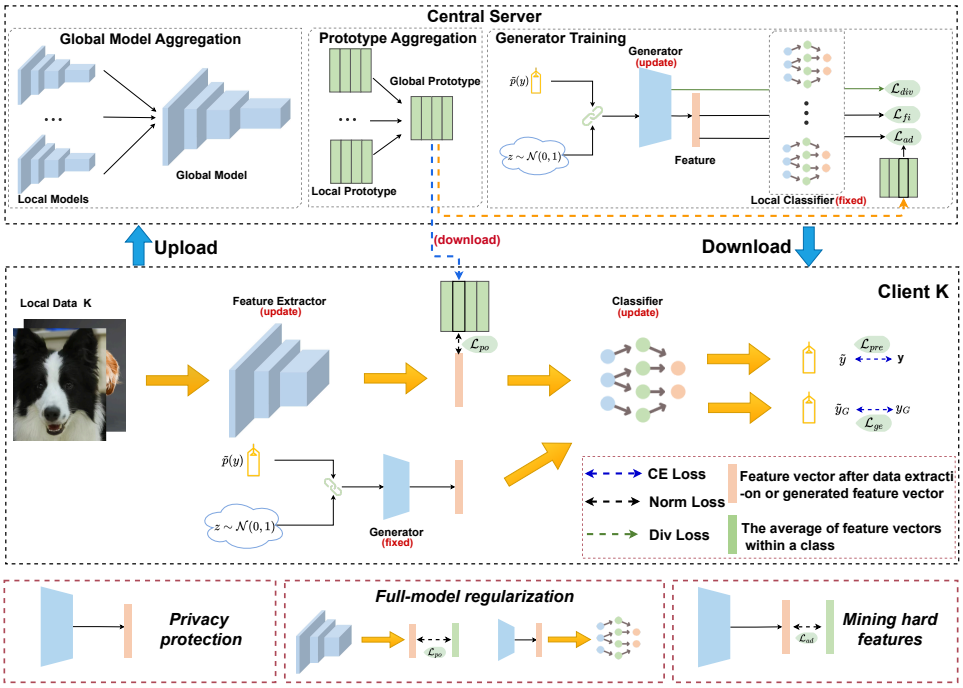


Fig. 2: The server aggregates local models and local prototype representations submitted by clients to create a global model and a global prototype representation. Subsequently, the server utilizes the local model classifiers uploaded by clients to train a generator. When this process is completed, the server delivers global prototype representations, the global model, and the generator to the clients. Clients use the global model to initialize their local models and align the extracted feature with the global prototype representations. Moreover, clients use the generator to enrich the feature space by generating features and feeding them into the classifier, enabling the classifier to learn more features.

After client i computes the prototype representations for all classes, it obtains the local prototype representations, denoted by φ_i^{t+1}. At the tth round, when the local training is completed, the selected clients upload their local models and the locally computed prototype representations φ_i^{t+1} to the sever, who constructs global prototype representations and the global model for the next round. The aggregation of the prototype representation for class m is implemented using the following formula:

$$\varphi^{t+1,m} = \sum_{i \in S_t} \frac{n_i^m}{n^m} \varphi_i^{t+1,m} \tag{3}$$

where n_i^m represents the number of samples from class j of client i; $n^m = \sum_{i \in S_t} n_i^m$ represents the total number of class m samples from selected clients.

Then the global prototype representations φ^{t+1} is obtained by aggregating the prototype representations of all classes.

C. Prototype-Adversarial Based Generator Training

Inspired by [8], we train a lightweight generator on the server side, which generates feature vectors rather than raw data using input label and Gaussian noise only, e.g., $\mathcal{Y} \rightarrow \mathcal{Z}$. Since the feature vectors generated by each client cannot be directly accessed by other clients, it prevents information leakage, achieving a certain degree of **privacy protection**. Moreover, because it does not directly generate high-dimensional data, e.g., images, the model complexity, storage overhead, and computational costs are significantly reduced.

$$h = G_\theta(z, y) \tag{4}$$

where h represents the features generated by the generator for label y; $z \sim \mathcal{N}(0, 1)$ corresponds to standard Gaussian noise.

In terms of generator training, we focus on optimizing the generator's performance through three

critical dimensions: *fidelity, difficulty keeping, and diversity*.

Regarding fidelity, generator training requires the classifier of a local model. The server feeds labels and noise into the generator to produce feature representations, which are then fed into the classifier of uploaded local model to evaluate the feature quality. Thereby, to ensure fidelity, we consider the following loss function:

$$L_{fid} = \frac{1}{B \cdot |S_t|} \sum_{k=1}^{|S_t|} \sum_{j=1}^{B} p_j L_{CE}(R_{\Phi_k}(h_j), y_j) \quad (5)$$

where B represents the batch size for generating features on the generator; $|S_t|$ is the number of selected clients; p_j is a weighting factor that represents the proportion of the number of classes y_j in client k to the total number of classes y_j in all selected clients, that is $p_j = n_k^{y_j} / n_{i \in S_t}^{y_j}$.

Regarding difficulty keeping, we adopt an adversarial method to **mine hard feature** representations. Since L_{fid} encourages the generator to generate samples that perform well on the local model, it tends to generate relatively straightforward features, which often result in lower feature quality. To address this, we design the generator to produce feature representations "moving away" from the global prototype representations. Under such condition, while still maintaining fidelity, the generator will seek hard features in the feature space, thereby enhancing the generalization capability of local model. Thus, to ensure difficulty keeping, the following loss function is considered:

$$L_{ad} = \frac{1}{B} \sum_{j=1}^{B} ||h_j - \varphi^{t+1,m}||_2 \quad (6)$$

where h_j denotes the feature representations produced by the generator for class m; $\varphi^{t+1,m}$ is the global prototype representations for class m at round t.

By reducing the L_{fid} and increasing the L_{ad} through adversarial training, the generator gradually focuses on ensuring the generation of high-quality features.

Regarding diversity, we borrow the idea of applying [28] to FedGen [8]. However, we make some adjustments by paying more attention to the diversity within classes. Thereby, to ensure diversity, we use the following loss function:

$$L_{div} = e^{\frac{1}{B \cdot B} \sum_{p,q \in \{1,\ldots,B\}} [1_{y_p = y_q}](-||h_p - h_q|| \cdot ||z_p - z_q||)} \quad (7)$$

where $1_{\text{condition}}$ is the indicator function.

The generator ultimately utilizes three loss functions as training objective as follows:

$$\min_{\theta} \mathbb{E}_{z \sim \mathcal{N}(\mathbf{0},\mathbf{1}), y \sim \tilde{p}(y)} L = \gamma_{fid} L_{fid} + \gamma_{div} L_{div} - \gamma_{ad} L_{ad} \quad (8)$$

where γ_{fid}, γ_{div} and γ_{ad} are both hyperparameters that can be adjusted.

D. Local Training and Feature Extractor Regularization Guided by Global Prototype Representations

In FedPA, during each global round, the selected clients receive the global model and global prototype representations from the previous round, and they also obtain a lightweight generator trained by the server. Then, they initialize the local models with the global one, and train them with the local dataset. The objective of client i can be expressed by:

$$L_{pre} = \frac{1}{B_i} \sum_{k=1}^{B_i} L_{CE}(R_{\Phi_i}(F_{\Pi_i}(x_k)), y_k) \quad (9)$$

where L_{CE} is the cross-entropy loss; $(x_k, y_k) \sim D_i$ represents the sample data and labels from the local dataset D_i.

However, during local model training, each client should ensure that the feature extractor possesses a global prototype perspective for each class. In other words, our goal is to guide local models in feature extraction for each class utilizing global prototype representations so as to achieve the extractor regularization. Thereby, we need to minimize the norm distance between the feature representation extracted from local data and the global prototype representation of the corresponding class to achieve feature alignment for each class. To this end, we design the following loss function to measure the feature alignment as:

$$L_{po} = \frac{1}{B_i} \sum_{k=1}^{B_i} \left\| F_{\Pi_i}(x_k) - \varphi^{t,y_k} \right\|_2 \quad (10)$$

where B_i represents the batch size for client i on their local dataset; $(x_k, y_k) \sim D_i$ represents the

Algorithm 1 FedPA Algorithm

Input: Total communication rounds T, global model parameters ω_g, local model parameters ω_i, generator parameters θ, local epochs E, the total number of samples of class m in all clients n^m, learning rate η, prototype average feature φ, clients sample ratio C.

Output: The final global model ω_g

Server executes:

Initialize global model ω_g^0 on server.
for $t = 0, \ldots, T-1$ **do**
  $S_t \leftarrow$ Randomly sample a set of $C \cdot N$ clients.
  broadcast generator θ^t, global parameters ω_g^t, global prototype φ^t to $i \in S_t$ clients.
  for $i \in S_t$ in parallel **do**
    $\omega_i^t, \varphi_i^{t+1}, c_i \leftarrow \text{ClientUpdate}(\omega_g^t, \theta^t, \varphi^t)$
  end for
  for m class $\varphi^{t+1,m} = \sum_{i \in S_t} \frac{n_i^m}{n^m} \varphi_i^{t+1,m}$ traverse all classes to get φ^{t+1}, and update label counter c, $\tilde{p}(y)$ based on c_i.
  $\omega^{t+1} \leftarrow \sum_{i \in S_t} {|D_i|}/{|D_{S_t}|} \omega_i^t$
  $\theta^t = \theta^t - \eta_2 \triangle L(\theta^t)$ **[Eq.8]**
end for

each i **ClientUpdate:** $(\omega_g^t, \theta^t, \varphi^t)$

$\omega_i^t \leftarrow \omega_g^t$
for $e = 1, 2, \ldots, E$ **do**
  from server get φ^t and $\tilde{p}(y)$, input noise vector z, and $y \sim \tilde{p}(y)$ to generator θ^t to get $\{h_j\}_{j=1}^B$.
  $\omega_i^t = \omega_i^t - \eta_1 \triangle L(\omega_i^t)$ **[Eq.12]**
  for m class $\varphi_i^{t,m} = \sum_{k=1}^{n_i^m} F_{\Pi_i}(x_k^m)/n_i^m$ traverse all classes to get φ_i^{t+1}.
  update label counter c_i.
end for
return $\omega_i^t, \varphi_i^{t+1}, c_i$ to the server

samples and labels in the dataset of client i; φ^{t,y_j} represents the global prototype representations for class y_j in the tth round; $||\cdot||$ denotes the Euclidean norm.

It is essential to emphasize that alignment and local training use the same samples. In other words, the very same samples are used for both regular classification training and feature alignment simultaneously.

E. Classifier Regularization based on Feature Generator

Due to the extreme imbalance in client data distribution, it's possible for some clients have very few samples or even none for certain classes. We are aware that the regularization of the feature extractor only is insufficient. This is because the classifier performs unsatisfactorily for classes with few or no samples. To address this, we introduce a lightweight feature generator to enhance the classification ability of local models.

In local training, each client utilizes the generator given by the server to generate features. These features are then fed to the local model's classifier to achieve the classifier regularization according to the following loss function:

$$L_{ge} = \frac{1}{B_i} \sum_{j=1}^{B_i} L_{CE}(R_{\Phi_i}(h_j), y_j) \tag{11}$$

Where $y_j \sim \tilde{p}(y)$ represents the label of the jth class; $\tilde{p}(y)$ is the global label distribution, which is formed by each client uploading its local label distribution to the server, and subsequently the server aggregates these distributions to form the global label distribution; $h_j \sim G_\theta(z_j, y_j)$ signifies the features generated as the result of inputting the label y_j into $G_\theta(z_j, y_j)$.

For each client, its local learning objective consists of the three loss functions mentioned above to achieve **full-model regularization**. The total loss function is as follows:

$$\min_{\omega_i} \mathbb{E}_{(x_k, y_k) \sim D_i, h \sim G_\theta(z,y)} L = L_{pre} + \lambda_{ge} L_{ge} + \lambda_{po} L_{po} \tag{12}$$

where λ_{ge} and λ_{po} are both hyperparameters that can be adjusted.

IV. Convergence and Generalization Bounds Analysis

A. Convergence Analysis

In this subsection, we present the convergence analysis for FedPA and make the following assumptions to derive the proof of FedPA's convergence, following a framework similar to that in [14] and [15]. The specific details pertaining to the assumptions and the proof can be found in Appendix A.

Assumption 1. *We define that each loss function $\mathcal{L}(\boldsymbol{\omega})$ satisfies L1-Lipschitz smoothness, meaning that its gradient is L1-Lipschitz continuous, and the embedding function of the feature extractor $F_{\Pi}(\cdot)$ satisfies L2-Lipschitz continuous.*

$$\left\|\nabla\mathcal{L}(\boldsymbol{\omega}^{t_1}) - \nabla\mathcal{L}(\boldsymbol{\omega}^{t_2})\right\|_2 \leq L_1 \left\|\boldsymbol{\omega}^{t_1} - \boldsymbol{\omega}^{t_2}\right\|_2, \quad \forall t_1, t_2 > 0 \tag{13}$$

$$\left\|F_{\boldsymbol{\Pi}^{t_1}}(\cdot) - F_{\boldsymbol{\Pi}^{t_2}}(\cdot)\right\| \leq L_2 \left\|\boldsymbol{\Pi}^{t_1} - \boldsymbol{\Pi}^{t_2}\right\|_2, \quad \forall t_1, t_2 > 0, \tag{14}$$

Assumption 2. *For each client i, the stochastic gradient $\boldsymbol{g}_i^t = \nabla\mathcal{L}\left(\boldsymbol{\omega}_i^t, \xi_i^t\right)$ used to update its model is an unbiased estimate.*

$$\mathbb{E}_{x_i \sim D_i}\left[g_i^t\right] = \nabla\mathcal{L}\left(\boldsymbol{\omega}_i^t\right) \quad \forall i\{\in 1, 2, 3, \ldots, N\}, \tag{15}$$

the variance of the gradient is bounded by σ^2:

$$\mathbb{E}\left[\left\|g_i^t - \nabla\mathcal{L}\left(\omega^t\right)\right\|_2^2\right] \leq \sigma^2, \forall i \in \{1, 2, 3, \ldots, N\}. \tag{16}$$

Assumption 3. *The expected value of the Euclidean norm of the stochastic gradient is bounded by V,*

$$\mathbb{E}\left[\left\|g_i^t\right\|_2^2\right] \leq V^2, \quad \forall i \in \{1, 2, 3, \ldots, N\} \tag{17}$$

Theorem 1. *By the above assumptions, it can be derived that in a communication round, the upper bound on the loss function for any client is satisfied as:*

$$\mathbb{E}\left[\mathcal{L}_i^{t+1,\frac{1}{2}}\right] - \mathcal{L}_i^{t,\frac{1}{2}} \leq E\eta_0 V^2 + L_1 E^2 \eta_0^2 V^2 + 2\lambda_{po} L_2 E\eta_0 V - \sum_{e=\frac{1}{2}}^{E-1}\left(\eta_e - \frac{\eta_e^2 L_1}{2}\right)\left\|\nabla\mathcal{L}_i^{t,e}\right\|_2^2 + \frac{\eta_0^2 L_1 E}{2}\sigma^2 \tag{18}$$

Theorem 1 shows that for any client, adjusting the appropriate hyperparameters ensures convergence of the upper bound on the deviation of the loss function between two consecutive communication rounds when this bound becomes less than 0.

Theorem 2. *(FedPA convergence) Let $\beta\eta_0 < \eta_e < \eta_0$, $e \in \{1/2, 1, 2, \ldots, E\}$, where β represents the decay factor for the learning rate.*

$$\beta\eta_0 < \eta_e < \frac{2\left\|\nabla\mathcal{L}_i^{t,e}\right\|_2^2 - 4\lambda_{po} L_2 V - 2V^2}{\left(2L_1 E V^2 + L_1\left\|\nabla\mathcal{L}_i^{t,e}\right\|_2^2 + L_1\sigma^2\right)} \tag{19}$$

Theorem 2 states that when the learning rate satisfies the conditions mentioned above, the loss function of any client decreases monotonically between two communication rounds, ensuring the convergence of the algorithm.

B. Generalization Bounds Analysis

In this subsection, we provide an analysis of the generalization bounds for FedPA. The specific details pertaining to the assumptions and the proof can be found in Appendix B.

Theorem 3. *(FedPA Generalization Bounds) Symbols with the same form as Lemma 4 share identical definitions. $\tilde{\mathcal{D}}_k'$, $\tilde{\mathcal{D}}_k$, and $\tilde{\mathcal{D}}_A$, after being influenced by global prototype representations, respectively become $\tilde{\mathcal{D}}_k^{\prime,p}$, $\tilde{\mathcal{D}}_k^p$, and $\tilde{\mathcal{D}}_A^p$, with probability of $1-\delta$:*

$$\mathcal{L}_{\mathcal{T}}(h) \leq \frac{1}{K}\sum_k \mathcal{L}_{\mathcal{T}_k^{\prime,p}}(h_k) + \frac{1}{K}\sum_k (d_{\mathcal{H}\triangle\mathcal{H}}(\tilde{\mathcal{D}}_k^{\prime,p}, \tilde{\mathcal{D}})) + \frac{1}{K}\sum_k \lambda_k^{\prime,p} + \sqrt{\frac{4}{N'}\left(d\log\frac{2eN'}{d} + \log\frac{4K}{\delta}\right)} \tag{20}$$

where $\mathcal{T}_k^{\prime,p} = \{\mathcal{D}_k^{\prime,p}, c^*\}$ represent updated local domain, $\lambda_k^{\prime,p}$ refers to the risk of the optimal hypothesis on both $\mathcal{T}$ and $\mathcal{T}_k^{\prime,p}$.

FedPA introduces global prototype representations, which further makes the local feature distribution closer to the global feature distribution, and by mining difficult samples, the augmented data distribution is closer to the global feature distribution, thus making the algorithm have better generalization.

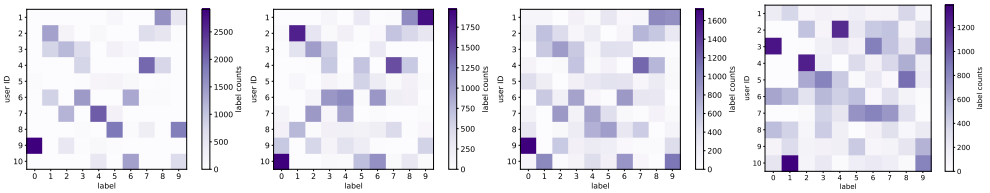
V. Experience

In this section, we compare the performance of FedPA with that of FedAvg, FedProx, FedEnsemble, FedGen, and FedHKD on four realistic datasets. All the mentioned algorithms are implemented using the PyTorch framework. Our code is available at https://github.com/threeStoneLei1/FedPA.

A. Experiment Detail

1) Datasets: We will evaluate the effectiveness of FedPA on four different datasets: MNIST, EMNIST, Fashion MNIST, and CelebA. MNIST is a

This article has been accepted for publication in IEEE Transactions on Dependable and Secure Computing. This is the author's version which has not been fully edited and content may change prior to final publication. Citation information: DOI 10.1109/TDSC.2024.3419211



(a) $\alpha = 0.1$ (a) $\alpha = 0.4$ (b) $\alpha = 0.7$ (c) $\alpha = 1.0$

Fig. 3: Visualization of varying degrees of heterogeneity in the Mnist dataset, where the depth of color represents the quantity of samples.

classic handwritten digit recognition dataset containing a series of handwritten digit images. The EMNIST dataset is an extension of the classic MNIST dataset, encompassing various character classes. Fashion MNIST is analogous to MNIST, encompassing grayscale images of 10 distinct categories of fashionable apparel and accessories. CelebA is a dataset widely used for research in facial recognition and facial attribute analysis.

2) Model: The generator consists of two fully connected layers, with the dimensions of each layer varying across different datasets. We denote the structural dimensions adopted for each dataset as $[d_i, d_h, d_o]$. For the MNIST and Fashion MNIST datasets, the generator structure is $[42, 256, 32]$. For the EMNIST dataset, the generator structure is $[58, 256, 32]$. For the CelebA dataset, the generator structure is $[32, 256, 32]$. We adopt CNN and fully connected layers as the model architecture for each client. For the MNIST and Fashion MNIST datasets, we designed the model structure as follows: $[6, 16, 32, 784, 10]$. For the EMnist dataset, our model structure is: $[6, 16, 32, 784, 26]$. In the case of the CelebA dataset, we employ the model structure: $[16, M, 32, M, 64, M, 32, 64, 2]$, where M represents the maximum pooling layer.

3) Hyperparameters: For all algorithms across all datasets, we have set 200 communication rounds. During each communication round, the local training consists of $E = 20$ epochs with a batch size of 32. The number of clients is fixed at 20 and the activation ratio C is 0.5, so that the number of clients S_t selected at each time is 10. In the local training loss function for clients, λ_{ge} starts at 25 with a decay rate of 0.98 per communication round. λ_{po} begins at 5 with a decay rate of 0.98 per communication round, reaching a minimum value of 0.15. As for the generator's hyperparameters, γ_{fid} starts at 25 and decays by 0.98 each communication round. γ_{div} is set to 1, and γ_{ad} is set to 0.15. We adopted a consistent random seed, set to 3. For training both the generator model and the client models, we employed the Adam optimizer, with a learning rate of 0.0001 for the EMNIST dataset, while other datasets were trained with a learning rate of 0.0003.

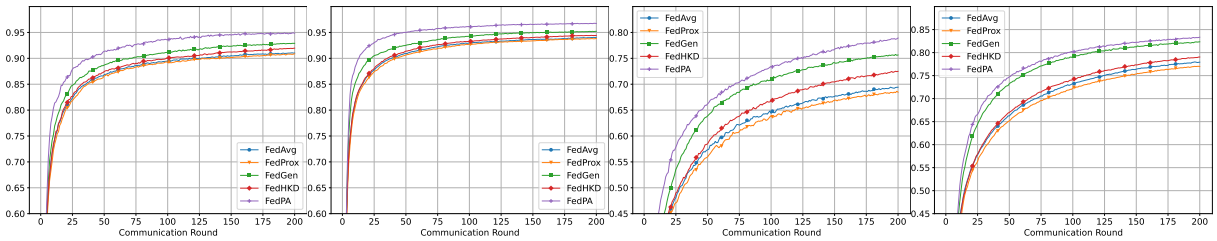
4) Data heterogeneity: We use the Dirichlet distribution Dir(α) to model data heterogeneity among clients, where smaller values of α indicate higher data heterogeneity. In the MNIST and EMNIST datasets, we set α to 0.1 and 1, respectively. For the Fashion MNIST dataset, we set α to 0.3 and 1. In Figure 3, a visualization of the heterogeneity of the MNIST dataset under different levels is presented. In these images, darker colors represent a higher number of samples for a specific client and label, while lighter colors indicate fewer samples.

5) Communication Overhead Analysis: In terms of communication overhead in the FedPA algorithm, we conducted two aspects of analysis: the upload phase and the download phase. In the upload phase, compared to conventional federated learning, we introduce local prototype representations and local label distributions. These two are essentially two-dimensional vectors, resulting in minimal communication overhead. In the download phase, we further introduce global prototype representations, label distributions, and a lightweight generator. Firstly, the communication overhead between global prototype representations and local prototype representations is roughly equivalent. Secondly, the lightweight generator consists of two fully connected layers. Compared to the neural networks required for training, its communication overhead is significantly reduced.

B. Performance Comparison

1) Test Accuracy: To validate the effectiveness of FedPA, we compared the accuracy of various federated learning algorithms on three datasets under

This article has been accepted for publication in IEEE Transactions on Dependable and Secure Computing. This is the author's version which has not been fully edited and content may change prior to final publication. Citation information: DOI 10.1109/TDSC.2024.3419211



MNIST Accuracy $\alpha = 0.1$ MNIST Accuracy $\alpha = 1.0$ EMNIST Accuracy $\alpha = 0.1$EMNIST Accuracy $\alpha = 1.0$

Fig. 4: Test accuracy curves across MNIST and EMNIST datasets under two levels of data heterogeneity

	MNIST		EMNIST		FashionMNIST		CelebA
	$\alpha = 0.1$	$\alpha = 1$	$\alpha = 0.1$	$\alpha = 1$	$\alpha = 0.3$	$\alpha = 1$	iid
FedAvg	88.87±2.19	94.58±0.56	70.09±0.81	78.41±0.55	84.36±0.26	86.31±0.18	85.53±0.34
FedProx	88.18±2.69	94.19±0.43	69.27±0.93	77.60±0.59	84.58±0.10	86.19±0.17	85.85±0.44
FedEnsemble	91.09±0.19	94.71±0.64	70.33±0.80	78.51±0.49	85.44±0.13	86.49±0.18	**87.64±0.69**
FedGen	90.49±2.40	95.34±0.18	76.34±0.78	82.60±0.33	85.26±0.18	86.90±0.14	86.57±0.68
FedHKD	90.15±2.53	94.76±0.35	72.39±0.23	79.27±0.36	84.67±0.19	86.44±0.05	85.66±0.83
FedPA(ours)	**94.79±0.18**	**96.77±0.05**	**78.25±0.67**	**83.30±0.09**	**85.52±0.50**	**87.41±0.23**	87.32±0.53

TABLE II: Test Accuracy of different FL algorithms on MNIST, EMNIST, FashionMNIST, and CelebA.

two different levels of data heterogeneity, as shown in Table II. In order to better visualize the learning processes of each method, in Figure 4, we plotted the test accuracy curves of various algorithms on different datasets. It's evident that FedPA demonstrates outstanding performance on the vast majority of datasets.

Results Analysis. Furthermore, as α decreases, indicating deeper data heterogeneity, the performance gap between FedPA and other algorithms widens. For example, on the EMNIST dataset with $\alpha = 0.1$ and $\alpha = 1.0$, FedPA outperforms FedAvg by 8 and 5 percentage points, respectively. This trend is also observed when comparing FedPA to other algorithms, indicating that FedPA effectively mitigates model bias in scenarios with high data heterogeneity through global prototype representations, generator-learned feature distribution, and the exploration of hard features.

To further validate that FedPA maintains an advantage under different client participation rates, we considered three values for the client participation rate C: 0.1, 0.5, and 0.8. We calculated the accuracy on the EMNIST dataset with $\alpha = 0.1$ as shown in Table III.

Results Analysis. FedPA consistently delivers the best performance across various client participation rates. It is noteworthy that, with the decrease in participation rate, our algorithm exhibits a more pronounced performance difference compared to the state-of-the-art algorithm. This phenomenon further underscores the exceptional robustness of our algorithm when dealing with heterogeneous data environments, specifically situations where lower client participation rates lead to a more uneven distribution of the global dataset.

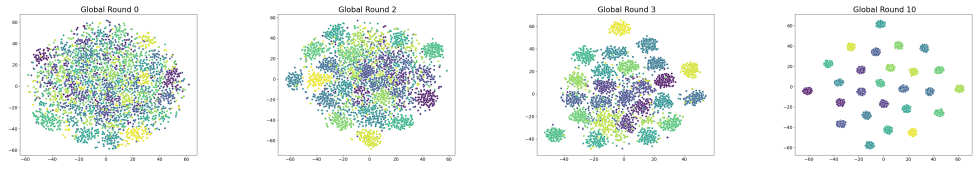
Algorithm	$\mathcal{C} = 0.1$	$\mathcal{C} = 0.5$	$\mathcal{C} = 0.8$
FedAvg	66.39	70.13	70.87
FedGen	69.46	75.59	76.21
FedHKD	70.58	71.83	73.41
FedPA(ours)	**74.00**	**77.14**	**78.35**

TABLE III: The performance influence of varying client participation rates on the EMNIST dataset.

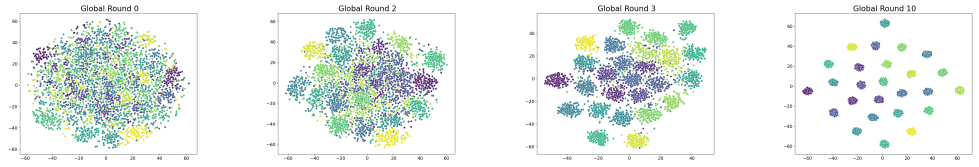
	$\mathcal{L}_{ad}$	$\mathcal{L}_{ge}$	$\mathcal{L}_{po}$	Accuracy
Module	×	×	×	70.94
	×	×	✓	75.21
	×	✓	×	77.39
	✓	✓	×	77.42
	×	✓	✓	78.85
	✓	✓	✓	**78.93**

TABLE IV: The impact of various components of FedPA.

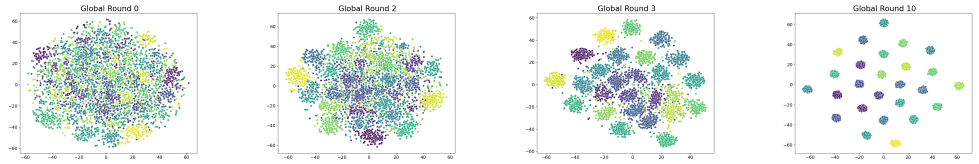
2) Ablation Study: To validate the effectiveness of various components of FedPA, we conducted ablation experiments on the EMNIST dataset with $\alpha = 0.1$. The experiments aim to verify the contributions of feature alignment, rich feature generation



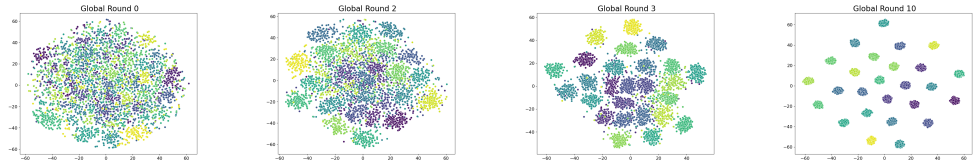
(a) $\alpha = 0.1$.



(b) $\alpha = 0.4$.



(c) $\alpha = 0.7$.



(d) $\alpha = 1.0$.

Fig. 5: t-SNE visualization of generator output features trained using clients trained on the EMNIST dataset with different data heterogeneity.

	$\alpha = 0.1$		$\alpha = 0.4$		$\alpha = 0.7$		$\alpha = 1.0$	
	intra-class	inter-class	intra-class	inter-class	intra-class	inter-class	intra-class	inter-class
round 0	27.898	27.351	27.898	27.351	27.898	27.351	27.898	27.351
round 2	16.673	40.106	17.227	39.785	16.328	40.558	14.244	40.731
round 3	7.626	44.693	6.987	44.571	7.215	45.525	6.431	43.939
round 10	**2.262**	**53.056**	**2.241**	**52.454**	**2.305**	**54.133**	**2.280**	**52.811**

TABLE V: The intra-class and inter-class distances for generated samples in different data heterogeneity.

by the generator, and exploration of challenging features. Accuracy comparisons can be observed in Table IV, demonstrating the effectiveness of different components of FedPA.

Results Analysis. In the absence of all three components, the algorithm is equivalent to the vanilla federated learning, exhibiting the poorest results. With the introduction of prototype learning or a generator, the performance has improved by 5% and 7%, respectively (see the second and third lines), indicating that aligning features enables local models to have a global perspective. Even for sparsely represented classes, the feature extractor demonstrates outstanding feature extraction capabilities. As the generator produces features, the classifier can comprehensively classify features, effectively alleviating overfitting. Furthermore, by incorporating challenging feature mining (see the fourth row), the generator gradually learns to generate complex features, thereby enhancing the quality of feature generation. Ultimately, in the presence of all three components, FedPA achieves optimal performance, conclusively confirming the effectiveness of these three components.

C. Generator's Learning Ability Analysis

To evaluate the generator's ability to learn data distributions under different levels of heterogeneity, we conducted visualization experiments using t-SNE. We set up 20 clients, each trained on the EMNIST dataset, with a client participation rate of 0.5. We conducted four experiments based on four different levels of data heterogeneity in the EMNIST dataset, with α set to 0.1, 0.4, 0.7, and 1.0, respectively. We mapped the prototype representations generated by the generator trained in each global round to data points on a two-dimensional plane using t-SNE (we only selected 0, 2, 3, and 10 global rounds for display, where the generator for global round 0 was not trained). These data points were represented in different colors to denote different categories, with data points of the same color representing the same category.

Results Analysis. By observing Figure 5 and Table V, we can clearly see that as the global rounds increase, the generator gradually learns the characteristics of each category, reducing the intra-class distance and increasing the inter-class distance.

VI. Conclusion

In this paper, to tackle the challenge of reduced model performance in federated learning stemming from high data heterogeneity, we introduce the Federated Prototype Adversarial (FedPA) framework. We successfully combine prototype learning and lightweight generators to achieve full-model regularization, and utilize prototype learning to mine hard features while preserving privacy. We have provided mathematical proofs for FedPA convergence analysis and generalization bounds, and extensive experiments have validated its superiority.

References

[1] Brendan McMahan, Eider Moore, Daniel Ramage, Seth Hampson, and Blaise Aguera y Arcas. Communication-efficient learning of deep networks from decentralized data. In *Artificial intelligence and statistics*, pages 1273–1282. PMLR, 2017.

[2] Yue Zhao, Meng Li, Liangzhen Lai, Naveen Suda, Damon Civin, and Vikas Chandra. Federated learning with non-iid data. *arXiv preprint arXiv:1806.00582*, 2018.

[3] Tzu-Ming Harry Hsu, Hang Qi, and Matthew Brown. Measuring the effects of non-identical data distribution for federated visual classification. *arXiv preprint arXiv:1909.06335*, 2019.

[4] Tian Li, Anit Kumar Sahu, Ameet Talwalkar, and Virginia Smith. Federated learning: Challenges, methods, and future directions. *IEEE signal processing magazine*, 37(3):50–60, 2020.

[5] Ahmed Khaled, Konstantin Mishchenko, and Peter Richtárik. Tighter theory for local sgd on identical and heterogeneous data. In *International Conference on Artificial Intelligence and Statistics*, pages 4519–4529. PMLR, 2020.

[6] Mohammad Rasouli, Tao Sun, and Ram Rajagopal. Fedgan: Federated generative adversarial networks for distributed data. *arXiv preprint arXiv:2006.07228*, 2020.

[7] Zhenyuan Zhang, Tao Shen, Jie Zhang, and Chao Wu. Feddtg: Federated data-free knowledge distillation via three-player generative adversarial networks. *arXiv preprint arXiv:2201.03169*, 2022.

[8] Zhuangdi Zhu, Junyuan Hong, and Jiayu Zhou. Data-free knowledge distillation for heterogeneous federated learning. In *International conference on machine learning*, pages 12878–12889. PMLR, 2021.

[9] Paul Pu Liang, Terrance Liu, Liu Ziyin, Nicholas B Allen, Randy P Auerbach, David Brent, Ruslan Salakhutdinov, and Louis-Philippe Morency. Think locally, act globally: Federated learning with local and global representations. *arXiv preprint arXiv:2001.01523*, 2020.

[10] Manoj Ghuhan Arivazhagan, Vinay Aggarwal, Aaditya Kumar Singh, and Sunav Choudhary. Federated learning with personalization layers. *arXiv preprint arXiv:1912.00818*, 2019.

[11] Yiqing Shen, Yuyin Zhou, and Lequan Yu. Cd2-pfed: Cyclic distillation-guided channel decoupling for model personalization in federated learning. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pages 10041–10050, 2022.

[12] Geoffrey Hinton, Oriol Vinyals, and Jeffrey Dean. Distilling the knowledge in a neural network. In *NIPS Deep Learning and Representation Learning Workshop*, 2015.

[13] Tao Lin, Lingjing Kong, Sebastian U Stich,

and Martin Jaggi. Ensemble distillation for robust model fusion in federated learning. *Advances in Neural Information Processing Systems*, 33:2351–2363, 2020.

[14] Yue Tan, Guodong Long, Lu Liu, Tianyi Zhou, Qinghua Lu, Jing Jiang, and Chengqi Zhang. Fedproto: Federated prototype learning across heterogeneous clients. In *Proceedings of the AAAI Conference on Artificial Intelligence*, volume 36, pages 8432–8440, 2022.

[15] Huancheng Chen, Chaining Wang, and Haris Vikalo. The best of both worlds: Accurate global and personalized models through federated learning with data-free hyper-knowledge distillation. In *The Eleventh International Conference on Learning Representations*, 2023.

[16] Tian Li, Anit Kumar Sahu, Manzil Zaheer, Maziar Sanjabi, Ameet Talwalkar, and Virginia Smith. Federated optimization in heterogeneous networks. *Proceedings of Machine learning and systems*, 2:429–450, 2020.

[17] Jianyu Wang, Qinghua Liu, Hao Liang, Gauri Joshi, and H Vincent Poor. Tackling the objective inconsistency problem in heterogeneous federated optimization. *Advances in neural information processing systems*, 33:7611–7623, 2020.

[18] Naichen Shi, Fan Lai, Raed Al Kontar, and Mosharaf Chowdhury. Fed-ensemble: Improving generalization through model ensembling in federated learning. *arXiv preprint arXiv:2107.10663*, 2021.

[19] Gihun Lee, Minchan Jeong, Yongjin Shin, Sangmin Bae, and Se-Young Yun. Preservation of the global knowledge by not-true distillation in federated learning. *Advances in Neural Information Processing Systems*, 35:38461–38474, 2022.

[20] Yuting He, Yiqiang Chen, Xiaodong Yang, Yingwei Zhang, and Bixiao Zeng. Class-wise adaptive self distillation for heterogeneous federated learning. In *Proceedings of the 36th AAAI Conference on Artificial Intelligence, Virtual*, volume 22, 2022.

[21] Lin Zhang, Li Shen, Liang Ding, Dacheng Tao, and Ling-Yu Duan. Fine-tuning global model via data-free knowledge distillation for non-iid federated learning. In *Proceedings of the IEEE/CVF conference on computer vision and pattern recognition*, pages 10174–10183, 2022.

[22] Alysa Ziying Tan, Han Yu, Lizhen Cui, and Qiang Yang. Towards personalized federated learning. *IEEE Transactions on Neural Networks and Learning Systems*, 2022.

[23] Yutong Dai, Zeyuan Chen, Junnan Li, Shelby Heinecke, Lichao Sun, and Ran Xu. Tackling data heterogeneity in federated learning with class prototypes. In *Proceedings of the AAAI Conference on Artificial Intelligence*, volume 37, pages 7314–7322, 2023.

[24] Sai Praneeth Karimireddy, Satyen Kale, Mehryar Mohri, Sashank Reddi, Sebastian Stich, and Ananda Theertha Suresh. Scaffold: Stochastic controlled averaging for federated learning. In *International conference on machine learning*, pages 5132–5143. PMLR, 2020.

[25] Qinbin Li, Bingsheng He, and Dawn Song. Model-contrastive federated learning. In *Proceedings of the IEEE/CVF conference on computer vision and pattern recognition*, pages 10713–10722, 2021.

[26] Tehrim Yoon, Sumin Shin, Sung Ju Hwang, and Eunho Yang. Fedmix: Approximation of mixup under mean augmented federated learning. In *International Conference on Learning Representations*, 2020.

[27] Ian Goodfellow, Jean Pouget-Abadie, Mehdi Mirza, Bing Xu, David Warde-Farley, Sherjil Ozair, Aaron Courville, and Yoshua Bengio. Generative adversarial nets. *Advances in neural information processing systems*, 27, 2014.

[28] Qi Mao, Hsin-Ying Lee, Hung-Yu Tseng, Siwei Ma, and Ming-Hsuan Yang. Mode seeking generative adversarial networks for diverse image synthesis. In *Proceedings of the IEEE/CVF conference on computer vision and pattern recognition*, pages 1429–1437, 2019.

[29] Xingchao Peng, Zijun Huang, Yizhe Zhu, and Kate Saenko. Federated adversarial domain adaptation. In *International Conference on Learning Representations*, 2020.

[30] Shai Ben-David, John Blitzer, Koby Crammer, and Fernando Pereira. Analysis of representations for domain adaptation. *Advances in neural information processing systems*, 19, 2006.

[31] Shai Ben-David, John Blitzer, Koby Crammer, Alex Kulesza, Fernando Pereira, and Jen-

nifer Wortman Vaughan. A theory of learning from different domains. *Machine learning*, 79:151–175, 2010.

Lei Jiang received the bachelor's degree from the School of Science and Technology at Nanchang University in 2022 and is currently pursuing a master's degree at the School of Computer and Cyberspace Security at Fujian Normal University. His research interests include federated learning and edge computing.

Xiaoding Wang received the Ph.D. degree in 2016 from the College of Mathematics and Informatics, Fujian Normal University, Fuzhou, China, where he is currently an Associate Professor. His main research interests include network optimization and fault tolerance.

Xu Yang received the Ph.D. degree in 2021 from the School of Science, RMIT University, Melbourne, Australia, in collaboration with Data61 CSIRO, Australia. He is currently an Associate Professor with the College of Computer and Data Science, Minjiang University, Fuzhou, China. He has published papers in major conferences/journals, such as IEEE TDSC, IEEE TSC, IEEE TSUSC, IEEE IoTJ, VEH COMMUN, FGCS, etc. His research interests include cryptography and information security.

Jiwu Shu ***(IEEE, Fellow)*** received the PhD degree in computer science from Nanjing University, in 1998, and finished the postdoctoral position research with Tsinghua University, in 2000. Since then, he has been teaching with Tsinghua University, and currently he is also the president of Minjiang university. His current research interests include storage security and reliability, non-volatile memory based storage systems, and parallel and distributed computing.

Hui Lin received the Ph.D. degree in computing system architecture from the College of Computer Science, Xidian University, Xi'an, China, in 2013. He is currently a Professor with the College of Computer and Cyber Security, Fujian Normal University, Fuzhou, China. His research interests include mobile cloud computing systems, blockchain, and network security.

Xun Yi received the Ph.D. degree in electronic engineering from Xidian University, Xi'an, China, in 1995. He is currently a Professor with the School of Computing Technologies, RMIT University, Australia. His research interests include data privacy protection, Cloud and IoT security, Blockchain, network security and applied cryptography. He has published over 300 research papers in international journals and conference proceedings. Currently, he is an Associate Editor for IEEE Transactions on Dependable and Secure Computing, IEEE Transactions on Knowledge and Data Engineering, ACM Computing Survey, Information Science (Elsevier) and Journal of Information Security and Application (Elsevier).