FedPAL: Adaptive Head Aggregation for Personalized Federated Learning Guided by Generative Feature Distributions

Heng Zhang[1,†], Ping Zhang[1,2,*], Defeng Wang[1,†], Wenhui Huang[1], Cankun Yue[1]

[1]School of Mathematics and Statistics, Henan Univ. of Sci. & Tech, Luoyang , China
[2]Intelligent System Science and Technology Innovation Center, Longmen Laboratory, Luoyang , China

***Abstract*—Federated learning, as an emerging distributed machine learning paradigm, offers a key advantage in effectively mitigating data silos while enhancing data privacy protection. However, heterogeneity in client data distributions often leads to insufficient generalization of the global model, thereby weakening personalized adaptation across clients. To address this issue, we propose a novel personalized federated learning method, FedPAL. On the server side, a class-conditional generator is constructed and trained using a classification loss and a supervised contrastive loss to generate feature distributions corresponding to class labels, so that the feature distributions exhibit intra-class compactness and inter-class separability. After downloading the generator, the client leverages the generated feature distributions to learn fusion weights between the local model head parameters and the global model head parameters. This process enables adaptive aggregation, which balances the absorption of global knowledge with the preservation of critical local information. As a result, an effective local model is initialized for each client at every communication round. To evaluate the effectiveness of FedPAL, extensive experiments are conducted on four benchmark datasets in the computer vision domain. The results demonstrate that the proposed method significantly outperforms ten state-of-the-art federated learning baseline methods.**

***Index Terms*—federated learning, generator, feature distributions, data heterogeneity, adaptively aggregate**

I. Introduction

Federated learning (FL) is a machine learning paradigm that balances data privacy protection with distributed collaboration. By combining decentralized local training with centralized model aggregation, clients share only model parameters rather than raw data, enabling collaborative training over multiple data sources while preserving privacy [1]. Owing to its data-locality property, FL has been widely applied in data-sensitive scenarios such as medical diagnosis [2] and financial risk control [3].

However, due to differences in client environments, data sources, and user behaviors, client data typically exhibit significant heterogeneity [4], such as regional traffic flow variations in intelligent transportation systems, differences in accents and environmental noise in speech recognition [5], and regional disparities in consumption and credit behaviors in financial scenarios [6]. Such data heterogeneity weakens the generalization ability and fairness of federated learning models, constituting one of the key challenges that limit their performance improvement [7].

To address data heterogeneity, personalized federated learning (PFL) has been proposed, which learns client-specific models tailored to local data distributions. Unlike traditional federated learning (TFL) approaches [8]–[10], which construct a single global model, PFL seeks to share global knowledge to maintain basic generalization, while guiding clients to perform personalized updates based on their local data. Existing PFL methods can be broadly categorized into three groups. (1) Model decomposition methods: split the model into shared and personalized components, enabling independent optimization of local parameters on top of a shared representation, such as FedPer [11], FedRep [12], and FedBABU [13]. (2) Regularization-based methods: incorporate global-model-related regularization terms into local objectives to constrain model drift and achieve stable personalization, including pFedMe [14], APFL [15], and Ditto [16]. (3) Personalized aggregation methods: design differentiated aggregation strategies, such as clustering or adaptive weight assignment, to improve model adaptation to heterogeneous client data distributions, with representative methods including FedALA [17], APPLE [18], and FedFomo [19].

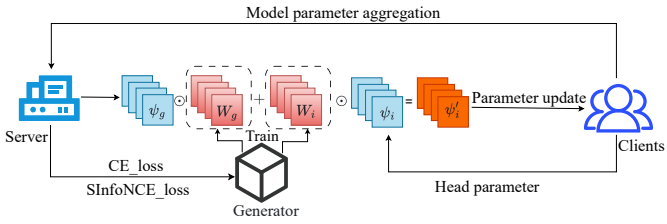

Fig. 1. Overall workflow of the FedPAL framework. The server trains a generator with CE_loss and SInfoNCE_loss using client-uploaded model head parameters to learn intra-class compact and inter-class separable feature distributions; clients adaptively aggregate global and local model head parameters based on these distributions, return updated parameters to the server, and achieve efficient fusion of global knowledge and local information.

[†]These authors contributed equally.
[*]Corresponding author. Email: zping@haust.edu.cn

This research was supported by the Joint Fund of Science and Technology Research and Development Program of Henan Province (No. 242103810054), the Major Science and Technology Projects of Longmen Laboratory (No. 231100220300).

In recent years, thanks to their ability to accurately model data distributions [20], generators have played a key role in areas such as image generation [21], data augmentation, text creation, and dialogue system construction. Inspired by this, we propose FedPAL, an adaptive model-head aggregation method guided by generative feature distributions, with the overall workflow illustrated in Figure 1. On the server side, a class-conditional generator is built to learn feature distributions for each class. Without access to raw data, the quality of the generated feature distributions is indirectly evaluated using client model head parameters through a cross-entropy loss, while a supervised contrastive loss, Supervised Information Noise-Contrastive Estimation (SInfoNCE) [22], [23], is introduced to enhance intra-class compactness and inter-class separability. At the client side, the generated feature distributions are used to adaptively aggregate global and local model head parameters, enabling the updated local model to retain critical local information while absorbing global knowledge. Experiments are conducted on four benchmark datasets under both pathological and practical heterogeneity scenarios, showing that FedPAL outperforms ten state-of-the-art federated learning methods in overall performance.

Our contributions are summarized as follows:

1) Proposed a generative feature distributions guidance mechanism: We use the feature distributions generated by the generator as global signals to guide client model aggregation, while ensuring data privacy.
2) Designed an adaptive aggregation strategy for client model heads: Clients dynamically adjust the fusion weights between global and local model head parameters based on the generator-produced feature distributions, balancing global stability with local specificity.
3) Significant performance improvement: FedPAL achieves notably higher classification accuracy and training stability compared with existing methods on four computer vision benchmark datasets under two heterogeneity scenarios, demonstrating its effectiveness and generalization capability.

II. Related Work

A. Traditional Federated Learning

TFL methods construct a single global model through local training and server-side parameter aggregation, where FedAvg [8] adopts a sample-size–weighted parameter averaging strategy. However, under non-iid data conditions [5], a single global model often suffers from performance degradation and unstable convergence. To mitigate this, FedProx [9] introduces a proximal regularization term to alleviate local model drift, while FedGen [10] leverages generative models and knowledge distillation to improve overall performance in heterogeneous settings. Nevertheless, since these methods still optimize a single global model, they struggle to adapt to client diversity and exhibit limitations in personalized performance and generalization ability.

B. Personalized Federated Learning

PFL arises to address the limitation that a single global model cannot adapt to the highly heterogeneous data distributions across clients. Its goal is to learn a model with client-specific characteristics while still leveraging shared global knowledge. Existing PFL methods can be roughly divided into three types: Model decomposition-based methods split the model into a shared part and a personalized part, aggregating only the shared parameters while the personalized part is trained independently by each client; FedPer [11] and FedRep [12] aggregate only feature extraction or backbone layers, leaving the classification heads for local optimization. Regularization-based methods constrain the difference between the local personalized model and the global model during training; pFedMe [14] uses a regularization term to guide local models toward personalized solutions. Personalized aggregation-based methods introduce differentiated strategies during model aggregation, allowing clients to receive distinct updates; FedALA [17] adaptively fuses global and local model parameters for personalization.

Our method belongs to the model decomposition-based PFL category, dividing the model into a feature extractor and a classification head. It introduces a generator, which clients use to produce feature distributions and adaptively aggregate local and global model head parameters. This allows the local models to absorb global knowledge while preserving critical local information, effectively enhancing model performance.

III. Preliminaries

In the FL framework, the system consists of a central server and N clients, where each client i uploads only model parameters rather than raw data to preserve data privacy [8]. TFL methods aim to learn a single unified global model. The server aggregates the local models from all clients to minimize the global objective function $H(w)$:

$$\min_{w \in \mathbb{R}^d} H(w) = \min_{w} \sum_{i=1}^{N} \alpha_i F_i(w) \tag{1}$$

Here, the model parameters are denoted by $w \in \mathbb{R}^d$, and D_i represents the local training dataset of a client, α_i denotes the aggregation weight of client i, which is typically defined as $\alpha_i = \frac{|D_i|}{\sum_{j=1}^{N} |D_j|}$, and satisfies $\sum_{i=1}^{N} \alpha_i = 1$. $|D_i|$ represents the number of local data samples held by client i, while $\sum_{j=1}^{N} |D_j|$ denotes the total number of samples across all clients, $F_i(w)$ represents the loss function of the client i on its local data D_i. The local objective function of a client can be expressed in terms of the empirical risk as:

$$F_i(w) = \frac{1}{|D_i|} \sum_{(x,y) \in D_i} \mathcal{L}(w, (x_i, y_i)) \tag{2}$$

Each sample pair (x_i, y_i) consists of an input feature $x_i \in X$ and its corresponding ground-truth label $y_i \in Y$. X and Y denote the features and labels in the feature distribution of dataset D_i, respectively. $\mathcal{L}(w, (x_i, y_i))$ represents the error loss between the model prediction and the true label. In practice,

heterogeneity in client data distributions leads to inconsistent local optima, making a single global model insufficient to satisfy all clients. To address this, personalized federated learning (PFL) is proposed, aiming to learn a dedicated model for each client. Let w_i denote the personalized model parameters of client i, then the optimization objective of PFL can be expressed as:

$$\{w_1, w_2, \ldots, w_N\} = \arg\min \sum_{i=1}^{N} \alpha_i \mathcal{L}(w_i, (x_i, y_i)) \quad (3)$$

w_i denotes the model parameters of client i. PFL learns a dedicated model for each client, alleviating the adaptation issues of a single global model under heterogeneous data, better aligning with local data distributions and improving performance. However, without effective global information sharing, personalized models tend to overfit to local data, limiting their generalization ability. To address this, our method enhances model generalization by fully leveraging both global and local information.

IV. Method

To fully leverage global information and enhance the performance of personalized models, this paper proposes a novel PFL method from two perspectives: model architecture and knowledge-sharing mechanism. Specifically, each client model is decoupled into a feature extractor and a classification head. On the server side, a generator produces feature distributions corresponding to the true labels, and clients use these generated distributions to adaptively aggregate global and local classification head parameters, enabling indirect knowledge sharing across clients. Guided by the global feature knowledge, each client further constructs a personalized model adapted to its local data distribution. Based on this, the overall model parameters are decoupled as follows:

$$\Theta_i = h\left(\theta_i, \psi_i\right) \quad (4)$$

Accordingly, the optimization objective of this method can be defined as:

$$\{\Theta_1, \Theta_2, \ldots, \Theta_N\} = \arg\min \sum_{i=1}^{N} \alpha_i \mathcal{L}(h(\theta_i, \psi_i), (x_i, y_i)) \quad (5)$$

Here, Θ_i denotes the overall model parameters of the i-th client, and θ_i represents the feature extractor parameters of the i-th client, where $\theta_i : \mathbb{R}^{b\times d} \to \mathbb{R}^{b\times m}$. b is the batch size, d is the input data dimension, and m is the feature representation dimension; ψ_i denotes the head parameters of the i-th client (where $\mathbb{R}^{b\times m} \to \mathbb{R}^{b\times t}$, correspond to the number of classes, matching the number of neurons in the output layer of θ_i). The function $h(\cdot)$ implements the integration of θ_i and ψ_i, such that $\Theta_i = h\left(\theta_i, \psi_i\right)$. The detailed algorithm flow is provided in **Algorithm 1**.

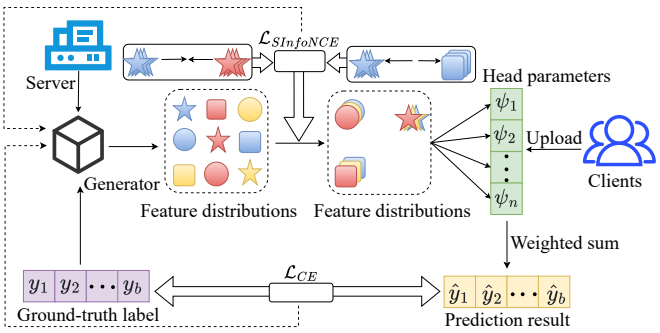

Fig. 2. Generator training pipeline. Different shapes represent the feature distributions of different labels; different colors for the same shape represent different feature distributions under the same label. The generator learns class-wise feature distributions under the joint supervision of the $\mathcal{L}_{CE}$ and the $\mathcal{L}_{SInfoNCE}$, with the corresponding loss gradients updating the generator parameters via backpropagation.

A. Training of the Server-Side Generator

During FL, the server cannot directly access clients' raw data and thus cannot explicitly construct the global data distribution; however, the uploaded model parameters implicitly encode discriminative information from distributed data. Based on this observation, we construct a class-conditional generator on the server to indirectly learn feature distributions constrained by global semantics at the model level, providing a unified shared reference for personalized model learning across clients. Specifically, we define the generator on the server as $G(\cdot)$:

$$t = G(\varepsilon, y, \theta^*, \psi_i) \quad (6)$$

Here, $\varepsilon \sim N(0, 1)$ denotes the random noise input, y denotes a randomly selected class label, $y \in \mathbb{R}^{b\times l}$, θ^* denotes the generator parameters, and ψ_i represents the head parameters of the i-th client. The generated feature distribution $t \in \mathbb{R}^{b\times m}$ is aligned with the output space of the client's feature extractor, where t_i denotes the feature distribution corresponding to label y_i. The generator learns label-specific feature distributions via conditional constraints. To ensure correct class semantics, the server supervises the generated feature distributions. Due to privacy constraints, it cannot access raw data, so following the idea of perceptual loss [24], the server indirectly constrains the feature distributions through model outputs, producing the global classification output:

$$\hat{y} = \sum_{k=1}^{n} \alpha_k \psi_k\left(t\right) \quad (7)$$

$\psi_k(\cdot)$ denotes the classification head of the k-th client, $\psi_k(t)$ represents the class predictions of the k-th client based on the feature distribution t, and α_k denotes the data-volume weight of the k-th client. $\hat{y}$ represents the aggregated model head prediction obtained by performing a data-volume-weighted average across all clients. Based on this aggregated prediction, the corresponding empirical classification cross-entropy loss is defined as:

$$CE\_loss = CE(\hat{y}, y) \quad (8)$$

$CE(\cdot)$ denotes the cross-entropy loss. The loss encourages the generator to produce feature distributions that can be correctly recognized by the global classifier, thereby indirectly aggregating discriminative information from client models into the generator. However, relying solely on cross-entropy tends to bias the generated feature distributions toward the classification head, which may cause feature collapse and weaken discriminability. To address this issue, the SInfoNCE loss is introduced to explicitly constrain feature similarity, enforcing intra-class compactness and inter-class separability in the feature space, and thus improving both representation discriminability and training stability.

We assume that the generator outputs a set of feature distributions $\{t_1, t_2, ..., t_b\}$ within a batch (b denotes the batch size and $t_i \in \mathbb{R}^{b\times m}$, m denotes the feature dimension). These feature distributions $t_1, t_2, ..., t_b$ correspond to labels $y_1, y_2, ..., y_b$. Based on the correspondence between feature distributions and labels, contrastive sample pairs are constructed as follows: if $y_i = y_j$ and $i \neq j$, then (t_i, t_j) forms a positive pair, representing different feature distributions generated under the same label, with the goal of maximizing their similarity to enhance intra-class compactness; if $y_i \neq y_j$ and $i \neq j$, then (t_i, t_j) forms a negative pair, representing feature distributions from different labels, with the goal of minimizing their similarity to enhance inter-class separability. To eliminate the interference of feature scale differences on similarity computation, each feature distribution t_i is first subjected to L_2 normalization:

$$\hat{t}_i = \frac{t_i}{\|t_i\|_2 + \alpha} \tag{9}$$

$\alpha = 10^{-8}$ is a small constant added to prevent the denominator from being zero. The similarity between sample pairs can be measured via the dot product $s(\hat{t}_i, \hat{t}_j) = \hat{t}_i \cdot \hat{t}_j$. To accommodate the server's lack of access to raw data, an intra-batch contrastive strategy is adopted: for each $\hat{t}_i$, samples with the same label within the batch form the positive set S, while samples with different labels form the negative set P. The loss function is then constructed based on these sets:

$$\mathcal{L}_{SInfoNCE}(\hat{t}_i) = -\log\left(\frac{A_i}{A_i + B_i}\right) \tag{10}$$

$$A_i = \sum_{\hat{t}_{i+}\in S} \exp\big(s(\hat{t}_i, \hat{t}_{i+})/\tau\big) \tag{11}$$

$$B_i = \sum_{\hat{t}_{i-}\in P} \exp\big(s(\hat{t}_i, \hat{t}_{i-})/\tau\big) \tag{12}$$

$\hat{t}_{i+}$ and $\hat{t}_{i-}$ denote the positive and negative samples of $\hat{t}_i$, respectively, and $s(\cdot)$ represents the similarity function. The temperature parameter τ controls the sensitivity of similarity differentiation—smaller values make the model more sensitive to similarity differences. The objective of this loss function is to maximize the similarity among positive samples with the same label while minimizing the similarity among negative samples with different labels. The loss is averaged over all feature distributions within the batch to obtain the final batch loss function:

$$SInfoNCE\_Loss = \frac{1}{b}\sum_{i=1}^{b} \mathcal{L}_{SInfoNCE}(\hat{t}_i) \tag{13}$$

As the loss decreases, the model is encouraged to learn feature distributions that are compact within the same class and well separated across different classes. The total training loss of the generator is the sum of these two training losses:

$$loss = CE\_loss + SInfoNCE\_loss \tag{14}$$

The total loss optimizes the generator parameters via back-propagation, enhancing feature discriminability while avoiding distribution aliasing.

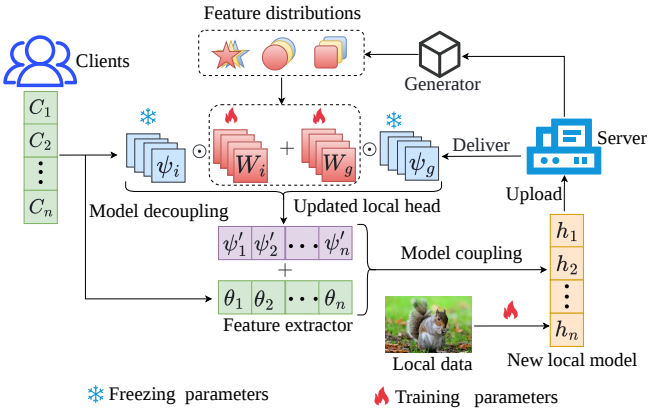

Fig. 3. The model is decoupled, and the feature distributions generated by the generator are used to train the aggregation weights W_1 and W_g (satisfying W_1 + W_g = 1). These weights are applied to adaptively fuse the head parameters of the local and global models, producing the updated local head parameters, which are subsequently re-coupled with the frozen feature extractor to train a new local model on the local data.

B. Adaptive Aggregation of Model Head

In TFL's distributed training, full-parameter exchanges incur high communication and energy costs, causing delays on resource-limited clients, and poorly adapt to data heterogeneity, limiting personalization. To address this, we propose an adaptive aggregation strategy. Figure 3 illustrates this aggregation process. Freezing the large feature extractor parameters reduces communication overhead, and aggregating only lightweight head parameters enhances personalized adaptation. Accordingly, after aggregating local and global heads, the overall model parameters of the i-th client are expressed as:

$$\Theta_i^{r+1} := h(\theta_g^r, \psi_i^r + (\psi_g^r - \psi_i^r) \odot \ W_i^r) \tag{15}$$

Here, $W_i^r \in \mathbb{R}^{m\times t}$ denotes the aggregation weight of the i-th client at the r-th round, ψ_g^r represents the global head parameters, θ_g^r represents the feature extractor parameters, ψ_i^r denotes the local head parameters of the client, and $\odot$ indicates the element-wise Hadamard product. Since the data distributions and training states vary across clients, the dependency of head parameter aggregation on global information

differs for each client. Therefore, it is necessary to train a dedicated aggregation weight W_i^r for each client. To prevent leakage of local client data, we construct representative feature distributions in the feature space using the generator, providing a stable and controllable training signal for weight learning. Let $\hat{\psi}_i$ denote the aggregated head parameters. The optimization objective of the i-th client can be uniformly expressed as the loss function $\mathcal{L}(\hat{\psi}_i, \sigma)$, where σ represents the feature distribution t and its corresponding label y. During training, the matrix W_i^r is initialized as an all-ones matrix, and W_i^r is updated via gradient descent:

$$W_i^r \leftarrow W_i^r - \eta \nabla_{W_i} \mathcal{L}(\hat{\psi}_i, \sigma) \tag{16}$$

η denotes the learning rate for training W_i^r. Based on the functional relationship between $\hat{\psi}_i$ and W_i^r, the gradient of the loss function with respect to W_i^r can be expanded using the chain rule:

$$\nabla_{W_i^r} \mathcal{L} = \nabla_{\hat{\psi}_i} \mathcal{L} \odot \nabla_{W_i^r} \hat{\psi}_i = \left(\psi_{\mathrm{g}}^{\mathrm{r}} - \psi_i^{\mathrm{r}}\right) \odot \nabla_{\hat{\psi}_i} \mathcal{L} \tag{17}$$

This gradient characterizes the influence of the difference between the global and local head parameters on the update direction of the aggregation weights. Furthermore, the update rule form of the aggregated head parameters $\hat{\psi}_i$ can be expressed as:

$$\hat{\psi}_i \leftarrow \hat{\psi}_i - \eta \left(\psi_{\mathrm{g}}^{\mathrm{r}} - \psi_i^{\mathrm{r}}\right) \odot \left(\psi_{\mathrm{g}}^{\mathrm{r}} - \psi_i^{\mathrm{r}}\right) \odot \nabla_{\hat{\psi}_i} \mathcal{L} \tag{18}$$

Finally, the client updates its local head parameters with the adaptively aggregated parameters, recombines the updated head with the frozen feature extractor to form the local model, and completes training using local data. The updated model parameters are then uploaded to the server for use in the next aggregation round.

Algorithm 1 FedPAL: Adaptive Head Aggregation for Personalized Federated Learning Guided by Generative Feature Distributions

Input: The number of clients N, client participation rate ρ, local loss function $\mathcal{L}_i$ of client i, number of training iterations for the aggregation weights j, local dataset $\mathcal{D}_i$ of client i, initial global model Θ^0, and local learning rate η.
Output: Personalized models: $\hat{\Theta}_1, \ldots, \hat{\Theta}_N$.

```
Server sends Θ_0 to all clients to initialize their local models
for each round n = 1, ..., T do
    Server samples a subset of clients S^t based on ρ
    Train G(y, z, θ*, ψ_i) by CE_loss and SInfoNCE_loss
    Send Θ_0 together with generator to all clients in S^t
    for each client i ∈ S^t in parallel do
        for j from 1 to j do
            Train W_i with generator by Equation (16)
        end for
        Aggregation head: ψ̂_i = (ψ_i^{r-1} + W_i ⊙ (ψ_g^{r-1} − ψ_i^{r-1}))
        Θ_i ← Θ_i − η∇L_i(Θ_i, D_i)
    end for
    Clients upload updated local parameters Θ_i to the server
    Θ ← Σ_{i∈S^t} k_i Θ_i
end for
return Θ̂_1, ..., Θ̂_N
```

V. Experiments

In this section, we evaluate the effectiveness of FedPAL by answering the following four research questions:

RQ1: Does FedPAL achieve superior classification performance compared with mainstream federated learning baselines under two heterogeneous scenarios?

RQ2: How robust is FedPAL when dataset heterogeneity and the number of clients vary?

RQ3: Does FedPAL incur lower computational and communication overhead compared with mainstream baselines?

RQ4: How stable is FedPAL's performance in scenarios with dynamic client participation?

A. Experimental Setup

Datasets and Non-iid scenarios: We conducted extensive experiments in computer vision using four widely adopted image classification datasets: MNIST [25], Cifar10 [26], Cifar100 [26], and Tiny-Imagenet [27], covering varying levels of complexity and diversity to comprehensively evaluate model performance. To simulate data heterogeneity, two scenarios were considered: pathological heterogeneity, where each client randomly samples 2/2/10/20 classes from the 10/10/100/200 classes of MNIST/Cifar10/Cifar100/Tiny-Imagenet with no overlap between clients, and practical heterogeneity [28], where client data distributions are controlled by a Dirichlet distribution $Dir(\beta)$, with smaller β indicating stronger heterogeneity (default $\beta = 0.1$).

Baselines: We select 10 representative FL methods as baselines, covering five core paradigms to ensure a comprehensive comparison. These include traditional FL methods: FedAvg [8] and FedProx [9]; generative FL methods: FedGen [10]; model-decomposition-based PFL methods: FedPer [11], FedRep [12], and FedBABU [13]; regularization-based PFL methods: pFedMe [14] and Ditto [16]; and personalized-aggregation-based PFL methods: APPLE [18] and FedFomo [19].

Implementation: The experiments use a 4-layer CNN [8] as the base network, with ResNet-18 [29] additionally employed on Tiny-Imagenet to validate the method's effectiveness on more complex models. Local learning rates are set to 0.005 and 0.01. The batch size is 10, with 1 local epoch per global iteration. The generator is built with two fully connected layers as its core architecture and uses ReLU as the activation function. For generator training, the SInfoNCE temperature is 0.1, and the learning rate is 0.005. All experiments run for 500 global iterations with 20 clients and a participation rate of $\rho = 1$. Evaluation metrics follow pFedMe [14]: TFL methods report the global model's best test accuracy, while PFL methods report the average best local model accuracy across clients. Each client uses 25% of its data for testing and the remainder for training.

All experiments are conducted on an NVIDIA GeForce RTX 4090 GPU, repeated 5 times, with results reported as mean ± standard deviation.

B. Performance Comparison and Analysis (RQ1)

Performance Comparison and Analysis: As shown in Table I, FedPAL outperforms the ten selected mainstream methods across two heterogeneity scenarios on four benchmark datasets (Tiny and Tiny* denote Tiny-ImageNet evaluated with a 4-layer CNN and ResNet-18, respectively). Its key advantage lies in the adaptive aggregation strategy guided by generative feature distributions, which preserves global knowledge consistency while accurately adapting to local data characteristics. Ablation results further indicate that FedPAL*, which omits the SInfoNCE loss, exhibits reduced feature discriminability on complex datasets, highlighting the SInfoNCE loss as crucial for maintaining the strategy's performance advantage.

TFL methods have limitations in handling heterogeneity. FedAvg aggregates models weighted by sample size; while simple, it ignores data heterogeneity and performs the worst. FedProx improves robustness via a regularization term but cannot fundamentally address heterogeneity. FedGen leverages a generator to alleviate data scarcity, performing well on simple datasets, yet it lacks feature-level constraints, leading to degraded performance on complex datasets.

Compared with traditional federated learning, personalized federated learning better adapts to local data distributions under heterogeneous scenarios, yet existing approaches have limitations. Model decomposition methods like FedRep, FedPer, and FedBABU only share the feature extractor, while keeping the classification head fully local, causing personalized heads to rely solely on local training and yielding limited performance on complex datasets. Regularization-based methods such as pFedMe and Ditto balance global and local objectives via fixed regularization, performing well under pathological heterogeneity but struggling under Dirichlet heterogeneity. Personalized aggregation methods like APPLE and FedFomo can introduce redundant information when distribution gaps are large, thereby reducing discriminative accuracy. In contrast, FedPAL combines model decomposition with generative feature distributions and adaptively aggregates global and local classification heads, integrating global knowledge with local information and achieving strong performance across heterogeneous scenarios.

C. Robustness of the Proposed Method (RQ2)

Robustness: To evaluate FedPAL's robustness and scalability, we conducted experiments on Cifar100 and Tiny-Imagenet. Data heterogeneity was controlled via Dir(β), with smaller β indicating stronger heterogeneity. As shown in Table II, FedPAL consistently achieves higher accuracy across varying heterogeneity levels. While other PFL methods perform well under high heterogeneity, their accuracy drops noticeably under weaker heterogeneity; only APPLE and FedPAL maintain an advantage. FedPAL, in particular, benefits from its generator-guided adaptive aggregation strategy, which balances global knowledge with local personalization. In scalability experiments with $\beta = 0.1$ and client numbers N=10, 30, 50, 100, and 150, most methods' performance declined as N increased, whereas FedPAL showed only minor fluctuations, demonstrating robustness under data scarcity and intensified heterogeneity.

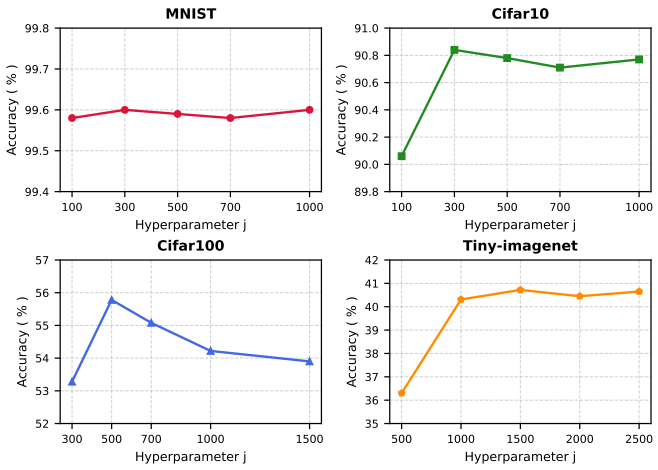

Fig. 4. Multi-dataset-j sensitivity analysis of FedPAL

Hyper-parameter analysis: The number of training iterations for the aggregation weights j is a key hyperparameter affecting the performance of FedPAL. On relatively simple datasets such as MNIST and Cifar10, a small j is sufficient for the aggregation weights to converge adequately, and the model accuracy is close to its peak. Therefore, setting $j = 300$ achieves a good trade-off between performance and computational cost. For the more complex Cifar100 dataset, $j = 300$ is insufficient for full convergence; increasing it to $j = 500$ yields optimal performance, whereas excessively large j may lead to overfitting. For the most complex Tiny-Imagenet dataset, $j = 500$ results in clearly suboptimal accuracy, while increasing it to $j = 1500$ allows the model to nearly converge; further increasing j causes a slight performance degradation. Overall, j exhibits a regular influence on FedPAL performance: values that are too small prevent sufficient convergence of aggregation weights, while overly large values increase computational overhead and may induce overfitting. Hence, an optimal value that ensures sufficient convergence with reasonable computation should be selected according to the dataset complexity.

D. Computation and Communication Overhead (RQ3)

Computation and Communication Cost Analysis: On the CIFAR-100 dataset, we compare algorithms in terms of total training time, communication rounds, per-round time, and communication overhead. As shown in Table III, FedGen requires 64.98 seconds per round with a total training time of 120 minutes; APPLE exceeds 150 seconds per round and incurs rapidly increasing communication overhead as the number of clients grows; pFedMe and Ditto both take over 40 seconds per round, resulting in longer total training time. Although FedRep and FedPer reduce total training time, they only train

TABLE I
THE TEST ACCURACY(%) UNDER TWO TYPES OF HETEROGENEITY SETTING

Settings	Pathological heterogeneous setting			Practical heterogeneous setting				
Methods	MNIST	Cifar10	Cifar100	MNIST	Cifar10	Cifar100	TINY	TINY*
FedAvg	95.42±0.06	55.81±0.81	26.86±0.24	97.77±0.04	53.63±0.43	32.56±0.39	19.46±0.20	19.45±0.13
FedProx	95.49±0.13	55.72±0.79	26.80±0.15	97.71±0.03	53.66±0.60	32.85±0.51	19.37±0.22	19.27±0.23
FedGen	99.69±0.10	56.21±0.58	23.90±0.54	98.46±0.07	57.07±0.47	30.97±0.62	19.39±0.18	18.53±0.32
FedRep	99.81±0.02	91.02±0.17	66.09±0.31	99.48±0.04	90.64±0.17	51.33±0.40	38.68±0.16	39.95±0.61
FedPer	99.81±0.03	90.71±0.29	62.63±0.44	99.48±0.05	90.29±0.27	49.68±0.38	39.24±0.19	39.65±0.31
FedBABU	99.77±0.02	88.66±0.07	65.21±0.12	99.52±0.04	89.87±0.11	54.66±0.15	39.27±0.14	41.89±0.36
pFedMe	99.75±0.02	90.11±0.10	58.20±0.14	99.52±0.02	88.09±0.32	47.34±0.46	26.93±0.19	33.44±0.33
Ditto	99.75±0.00	91.02±0.16	66.24±0.37	99.20±0.00	90.59±0.01	52.87±0.64	32.15±0.04	35.92±0.43
APPLE	99.73±0.03	90.78±0.03	65.81±0.11	99.24±0.02	89.27±0.07	53.18±0.22	39.74±0.44	40.34±0.33
FedFomo	99.78±0.01	90.63±0.05	64.17±0.30	99.13±0.02	88.26±0.03	45.19±0.39	31.44±0.17	33.65±0.54
FedPAL*	99.82±0.02	91.10±0.05	50.51±0.08	99.60±0.01	90.64±0.11	46.91±0.20	34.44±0.47	43.80±0.52
FedPAL	**99.85±0.02**	**91.10±0.06**	**66.88±0.12**	**99.60±0.01**	**90.84±0.08**	**55.78±0.20**	**40.72±0.18**	**44.31±0.16**
Δ SOTA	↑ 0.04	↑ 0.08	↑ 0.64	↑ 0.08	↑ 0.20	↑ 1.12	↑ 0.98	↑ 2.42

FedPAL* refers to training using only the CE loss, without the SInfoNCE loss.

TABLE II
THE TEST ACCURACY(%) UNDER DIFFERENT HETEROGENEITIES AND VARYING CLIENT NUMBERS

	different heterogeneous setting				different client counts setting				
Dataset	Cifar100		Tiny-net		Cifar100				
Setting	Dir(0.3)	Dir(0.5)	Dir(0.05)	Dir(0.5)	10 clients	30 clients	50 clients	100 clients	150 clients
FedAvg	34.90±0.27	36.41±0.19	16.05±0.51	21.14±0.47	31.47±0.01	31.15±0.05	31.90±0.37	31.95±0.37	26.40±0.72
FedProx	35.11±0.33	36.93±0.17	16.00±0.45	21.22±0.47	31.24±0.08	31.21±0.08	31.94±0.30	31.97±0.24	26.51±0.52
FedGen	34.00±0.17	35.15±0.22	13.99±0.37	17.29±0.43	30.12±0.05	31.49±0.14	32.04±0.16	27.07±0.20	14.62±0.88
FedRep	37.45±0.23	30.10±0.17	**50.12±0.15**	20.15±0.09	52.89±0.10	50.24±0.01	47.41±0.18	44.61±0.20	38.57±0.43
FedPer	37.00±0.08	30.86±0.13	47.02±0.34	20.38±0.44	50.31±0.19	49.05±0.20	46.85±0.18	42.37±0.41	40.26±0.57
FedBABU	44.22±0.21	39.75±0.29	46.71±0.37	20.42±0.42	53.55±0.10	53.73±0.15	56.22±0.28	50.68±0.27	40.02±0.48
pFedMe	34.55±0.13	29.96±0.32	34.72±0.14	17.48±0.61	44.06±0.29	47.04±0.28	48.36±0.64	46.45±0.18	41.55±0.61
Ditto	37.79±0.09	31.05±0.15	46.01±0.21	18.98±0.05	52.32±0.19	52.53±0.42	54.22±0.04	52.89±0.22	40.18±0.53
APPLE	38.11±0.11	31.47±0.17	40.17±0.15	24.15±0.06	49.44±0.15	48.12±0.21	55.03±0.18	52.80±0.15	39.42±0.33
FedFomo	31.60±0.34	25.77±0.23	39.53±0.60	15.13±0.15	46.71±0.23	43.20±0.05	42.56±0.33	38.91±0.08	35.85±0.42
FedPAL	**46.05±0.02**	**43.84±0.18**	47.63±0.09	**24.19±0.11**	**55.42±0.02**	**55.74±0.03**	**56.85±0.17**	**55.37±0.20**	**55.32±0.32**
Δ SOTA	↑ 1.83	↑ 4.09	↑ 0	↑ 0.04	↑ 1.87	↑ 2.01	↑ 0.63	↑ 2.48	↑ 13.77

TABLE III
COMPARATIVE ANALYSIS OF TIME REQUIRED FOR CONVERGENCE OF DIFFERENT ALGORITHMS ON CIFAR100

Method	Total Time	Iterations	Time per Iteration	Communication
FedAvg	58.03 min	187	18.62 s	$2 * \Sigma$
FedProx	70.81 min	190	22.32 s	$2 * \Sigma$
FedGen	121.56 min	112	64.98 s	$(2 + \beta) * \Sigma$
FedRep	26.53 min	65	24.49 s	$2 * \alpha * \Sigma$
FedPer	22.79 min	71	19.34 s	$2 * \alpha * \Sigma$
FedBABU	63.24 min	186	20.38 s	$2 * \Sigma$
pFedMe	144.9 min	207	41.80 s	$2 * \Sigma$
Ditto	39.61 min	55	43.21 s	$2 * \Sigma$
APPLE	244.59 min	93	157.59 s	$(1 + n) * \Sigma$
FedFomo	65.04 min	154	25.21 s	$(1 + n) * \Sigma$
FedPAL	40.3 min	136	17.59 s	$2 * \Sigma$

Σ denotes the parameter volume transmitted per communication round. β represents the generator parameter size in FedGen. α specifies the proportion of classifier parameters relative to the total model parameters. n indicates the number of models downloaded by both APPLE and FedFomo.

local modules at the expense of generalization ability. In contrast, FedPAL requires only 17.59 seconds per round with a total training time of 40.3 minutes, while maintaining low communication overhead. Its generator performs lightweight fitting of global feature distributions on the server, without client involvement in generator training or additional parameter transmission. Thus, FedPAL avoids redundant computation and communication, effectively integrates global and local information, and achieves a favorable balance between performance and training efficiency.

TABLE IV
ACCURACY(%) WITH DIFFERENT CLIENT PARTICIPATION RATES ON CIFAR100

Ratios	$\rho = [0.8, 1]$	$\rho \in [0.5, 1]$	$\rho \in [0.2, 1]$	$\rho \in [0.1, 1]$
FedAvg	33.53±0.31	33.20±0.51	33.35±0.48	33.28±0.51
FedProx	33.45±0.34	33.24±0.46	33.28±0.49	33.15±0.57
FedGen	32.01±0.23	31.87±0.54	30.98±0.92	30.15±0.95
FedRep	47.56±0.20	47.01±0.23	47.56±0.22	46.97±0.22
FedPer	45.73±0.24	45.32±0.27	45.07±0.27	44.07±0.27
FedBABU	46.14±0.53	46.02±0.74	45.85±1.02	45.65±1.15
pFedMe	46.35±0.55	43.28±0.85	42.17±0.93	41.71±1.02
Ditto	50.15±0.15	49.88±0.36	49.15±0.45	48.35±3.22
APPLE	51.16±0.12	49.85±0.24	49.65±0.34	47.65±0.74
FeFomo	42.51±0.34	41.13±0.07	40.98±0.11	40.55±0.07
FedPAL	**54.69±0.15**	**54.41±0.21**	**54.16±0.25**	**53.47±0.30**
Δ SOTA	↑ 3.53	↑ 4.53	↑ 4.51	↑ 5.12

E. Stability of Client Dynamic Participation (RQ4)

Client Participation Rate Variations: In practical federated learning scenarios, clients may randomly drop out or rejoin due to network fluctuations, device resource limitations, or

other factors. To evaluate the stability of different PFL methods under such dynamic environments, we simulate this situation on Cifar100 by adjusting the per-round client participation rate ρ, where a wider ρ sampling range indicates greater participation variability. In Table IV, experimental results show that when ρ expands from [0.8, 1] to [0.1, 1], FedGen's accuracy drops by over 2%, with its standard deviation increasing from 0.23 to 0.95; FedBABU's accuracy decreases by nearly 4.5%, with standard deviation exceeding 1.15; pFedMe drops over 4.6%, and APPLE declines by 3.7%, all exhibiting poor stability. In contrast, FedPAL consistently outperforms other methods across the entire ρ range, with accuracy dropping by only about 1% and standard deviation remaining below 0.30. This robustness arises from its generator-guided adaptive aggregation strategy, which captures global information while quickly adapting to the local characteristics of rejoining clients, thereby maintaining both high performance and stability in dynamic participation scenarios.

VI. Conclusion

This work proposes a personalized federated learning method, FedPAL, to address insufficient model generalization and poor personalization caused by data heterogeneity. FedPAL constructs a class-conditional generator on the server and jointly optimizes a classification loss and a supervised contrastive loss to produce intra-class compact and inter-class separable feature distributions, which guide cross-client knowledge sharing. Based on these distributions, clients perform adaptive, parameter-wise aggregation of global and local classification heads, effectively absorbing global knowledge while preserving local data characteristics. Extensive experiments on multiple computer vision benchmarks under heterogeneous settings demonstrate that FedPAL consistently outperforms mainstream methods and maintains strong stability and scalability under varying client scales and random dropouts.

References

[1] J. Konečný, H. B. McMahan, F. X. Yu, P. Richtarik, A. T. Suresh, and D. Bacon, "Federated learning: Strategies for improving communication efficiency," arXiv preprint arXiv:1610.05492, 2016.

[2] N. T. Madathil, F. K. Dankar, M. Gergely, et al., "Revolutionizing healthcare data analytics with federated learning: a comprehensive survey of applications, systems, and future directions," Computational and Structural Biotechnology Journal, 2025.

[3] P. K. R. Gouni and M. A. Faheem, "A scalable federated learning architecture for privacy-preserving financial data processing," Int. Adv. Res. J. Sci. Eng. Technol. (IARJSET), vol. 12, no. 12, pp. 400–411, Dec. 2025.

[4] P. Kairouz, H. B. McMahan, B. Avent, A. Bellet, M. Bennis, A. N. Bhagoji, K. Bonawitz, Z. Charles, G. Cormode, R. Cummings et al., "Advances and open problems in federated learning," Foundations and Trends® in Machine Learning, vol. 14, no. 1–2, pp. 1–210, 2021.

[5] J. Pei, M. Omar, M. M. Al Dabel, et al., "Federated few-shot learning with intelligent transportation cross-regional adaptation," IEEE Trans. Intell. Transp. Syst., 2025.

[6] R. Li, Y. Cao, Y. Shu, J. Guo, B. Shi, J. Yu, Y. Di, Q. Zuo, and H. Tian, "A dynamic receptive field and improved feature fusion approach for federated learning in financial credit risk assessment," Scientific Reports, vol. 14, no. 1, p. 26515, 2024.

[7] E. Collins and M. Wang, "Federated Learning: A Survey on Privacy-Preserving Collaborative Intelligence," arXiv preprint arXiv:2504.17703, 2025.

[8] B. McMahan, E. Moore, D. Ramage, et al., "Communication-efficient learning of deep networks from decentralized data," in Proc. 20th Int. Conf. Artif. Intell. Stat. (AISTATS), PMLR, pp. 1273–1282, Apr. 2017.

[9] T. Li, A. K. Sahu, M. Zaheer, M. Sanjabi, A. Talwalkar, and V. Smith, "Federated optimization in heterogeneous networks," arXiv preprint arXiv:1902.01046, 2020.

[10] Z. Zhu, J. Hong, and J. Zhou, "Data-free knowledge distillation for heterogeneous federated learning," in Proc. 38th Int. Conf. Mach. Learn., vol. 139, pp. 12878–12889, Jul. 2021.

[11] M. G. Arivazhagan, V. Aggarwal, A. K. Singh, and S. Choudhary, "Federated learning with personalization layers," arXiv preprint arXiv:1912.00818, 2019.

[12] L. Collins, H. Hassani, A. Mokhtari, and S. Shakkottai, "Exploiting shared representations for personalized federated learning," in Proc. Int. Conf. Mach. Learn. (ICML), PMLR, pp. 2089–2099, Jul. 2021.

[13] J. Oh, S. Kim, and S.-Y. Yun, "FedBABU: Towards enhanced representation for federated image classification," arXiv preprint arXiv:2106.06042, Mar. 16, 2022.

[14] C. T. Dinh, N. Tran, and J. Nguyen, "Personalized federated learning with Moreau Envelopes," Adv. Neural Inf. Process. Syst., vol. 33, pp. 21394–21405, 2020.

[15] Y. Deng, M. M. Kamani, and M. Mahdavi, "Adaptive personalized federated learning," in Proc. Int. Conf. Learn. Represent. (ICLR), 2021.

[16] T. Li, S. Hu, A. Beirami, and V. Smith, "Ditto: Fair and Robust Federated Learning Through Personalization," in Proc. Int. Conf. Mach. Learn. (ICML), PMLR, pp. 6357–6368, Jul. 2021.

[17] J. Zhang, Y. Hua, H. Wang, T. Song, Z. Xue, R. Ma, and H. Guan, "Fedala: Adaptive local aggregation for personalized federated learning," in Proc. AAAI Conf. Artif. Intell., vol. 37, no. 9, pp. 11237–11244, 2023.

[18] J. Luo, S. Wu, "Adapt to adaptation: Learning personalization for Cross-Silo Federated Learning," in Proc. Int. Joint Conf. Artif. Intell. (IJCAI), NIH Public Access, p. 2166, 2022.

[19] M. Zhang, K. Sapra, S. Fidler, S. Yeung, and J. M. Alvarez, "Personalized federated learning with first order model optimization," arXiv preprint arXiv:2012.08565, Mar. 26, 2021.

[20] I. Goodfellow, J. Pouget-Abadie, M. Mirza, B. Xu, D. Warde-Farley, S. Ozair, A. Courville, and Y. Bengio, "Generative Adversarial Nets," Adv. Neural Inf. Process. Syst., vol. 27, pp. 2672–2680, 2014.

[21] Z. Liu, Y. Li, and J. Wang, "OmniGen2: Exploration to advanced multimodal generation," arXiv preprint arXiv:2506.18871, 2025.

[22] C. A. Barbano, B. Dufumier, E. Tartaglione, M. Grangetto, and P. Gori, "Unbiased Supervised Contrastive Learning," in Proc. Int. Conf. Learn. Represent. (ICLR), 2023.

[23] P. Khosla, P. Teterwak, C. Wang, A. Sarna, Y. Tian, P. Isola, A. Maschinot, E. Liu, and D. Krishnan, "Supervised Contrastive Learning," in Adv. Neural Inf. Process. Syst., vol. 33, pp. 18661–18673, 2020.

[24] J. Johnson, A. Alahi, and L. Fei-Fei, "Perceptual losses for real-time style transfer and super-resolution," in Proc. Eur. Conf. Comput. Vis. (ECCV), Cham, Switzerland: Springer, pp. 694–711, 2016.

[25] Y. LeCun, L. Bottou, Y. Bengio, and P. Haffner, "Gradient-based Learning Applied to Document Recognition," Proc. IEEE, vol. 86, no. 11, pp. 2278–2324, 1998.

[26] A. Krizhevsky, G. Hinton, "Learning multiple layers of features from tiny images," Tech. Rep., 2009.

[27] P. Chrabaszcz, I. Loshchilov, and F. Hutter, "A downsampled variant of ImageNet as an alternative to the CIFAR datasets," arXiv preprint arXiv:1707.08819, 2017.

[28] T. Lin, L. Kong, S. U. Stich, and M. Jaggi, "Ensemble distillation for robust model fusion in federated learning," Adv. Neural Inf. Process. Syst., vol. 33, pp. 2351–2363, 2020.

[29] K. He, X. Zhang, S. Ren, and J. Sun, "Deep residual learning for image recognition," in Proc. IEEE Conf. Comput. Vis. Pattern Recognit., pp. 770–778, 2016.