FedAPA: Server-side Gradient-Based Adaptive Personalized Aggregation for Federated Learning on Heterogeneous Data

Yuxia Sun[1] , **Aoxiang Sun**[2] , **Siyi Pan**[1] , **Zhixiao Fu**[3] and **Jingcai Guo**[*4]
[1]College of Information Science and Technology, Jinan University
[2]College of Information and Computational Science, Jilin University
[3]Department of Computer Science, University of Toronto
[4]Department of Computing & LSGI, The Hong Kong Polytechnic University
tyxsun@email.jnu.edu.cn, 2835604014@qq.com, StuPsy@stu2022.jnu.edu.cn,
zx.fu@mail.utoronto.ca, jc-jingcai.guo@polyu.edu.hk

Abstract

Personalized federated learning (PFL) tailors models to clients' unique data distributions while preserving privacy. However, existing aggregation-weight-based PFL methods often struggle with heterogeneous data, facing challenges in accuracy, computational efficiency, and communication overhead. We propose FedAPA, a novel PFL method featuring a server-side, gradient-based adaptive aggregation strategy to generate personalized models, by updating aggregation weights based on gradients of client-parameter changes with respect to the aggregation weights in a centralized manner. FedAPA guarantees theoretical convergence and achieves superior accuracy and computational efficiency compared to 10 PFL competitors across three datasets, with competitive communication overhead. The code and full proofs are available at: https://github.com/Yuxia-Sun/FL_FedAPA.

1 Introduction

As a prominent distributed machine learning paradigm, Federated Learning (FL) allows clients to collaboratively train models while keeping data local [Yang *et al.*, 2019]. In cross-silo FL, client data is often non-IID (non-Independent and Identically Distributed), exhibiting statistical heterogeneity that leads to client drift and reduces the effectiveness of methods like FedAvg [McMahan *et al.*, 2017].

To address the challenges posed by heterogeneous data distributions, Personalized Federated Learning (PFL) has been developed to tailor models to the unique data characteristics of individual clients[Wu *et al.*, 2020]. PFL can be broadly categorized into three main approaches: (1) Personalized model-based approaches, which directly create a personalized local model for each client, typically without relying on a global model [Li *et al.*, 2021b; Huang *et al.*, 2021; Luo and Wu, 2022; Shamsian *et al.*, 2021]; (2) Global model-based approaches, where a global model is first constructed and then personalized through techniques like client-specific fine-tuning [Xu *et al.*, 2023; Zhang *et al.*, 2024]; and (3) Other PFL approaches include those that leverage information beyond model parameters, such as prototypes, to optimize local models [Tan *et al.*, 2022]. Among these, personalized model-based approaches, the primary focus of this paper, have gained significant attention for their ability to directly tailor local models to each client's unique data distribution.

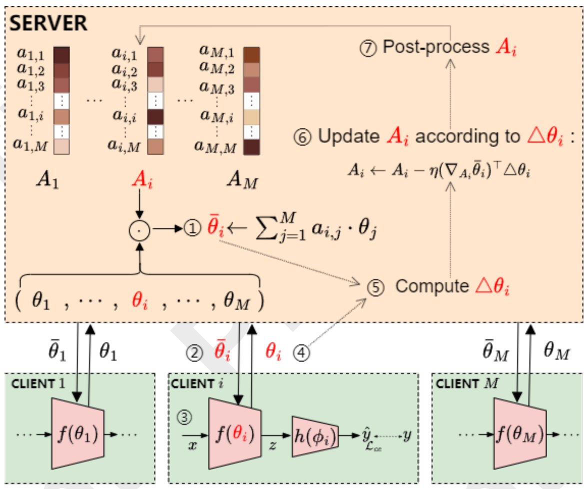
SERVER
$a_{1,1}$ $a_{1,2}$ $a_{1,3}$ $a_{1,i}$ $a_{1,M}$
$a_{i,1}$ $a_{i,2}$ $a_{i,3}$ $a_{i,i}$ $a_{i,M}$
$a_{M,1}$ $a_{M,2}$ $a_{M,3}$ $a_{M,i}$ $a_{M,M}$
A_1
A_i
A_M
⑦ Post-process A_i
⑥ Update A_i according to $\triangle\theta_i$:
$A_i \leftarrow A_i - \eta(\nabla_{A_i}\bar{\theta}_i)^\top \triangle\theta_i$
① $\bar{\theta}_i \leftarrow \sum_{j=1}^{M} a_{i,j} \cdot \theta_j$
$(\theta_1, \cdots, \theta_i, \cdots, \theta_M)$
⑤ Compute $\triangle\theta_i$
$\bar{\theta}_1$ θ_1
② $\bar{\theta}_i$ θ_i ④
$\bar{\theta}_M$ θ_M
CLIENT 1
CLIENT i
CLIENT M
③
$f(\theta_1)$
$f(\theta_i)$
$h(\phi_i)$
$f(\theta_M)$
x z $\hat{y}$ $\mathcal{L}_{ce}$ y

Figure 1: Framework of FedAPA. ① The server generates personalized model $\bar{\theta}_i$ via aggregation according to weight vector A_i; ② Each client downloads its model $\bar{\theta}_i$; ③ Local training on private data; ④ Each client upload its model θ_i; ⑤ The server computes the update of model parameters $\triangle\theta_i$; ⑥ Updates A_i via gradient descent according to $\triangle\theta_i$; ⑦ Post-processes A_i via clipping, self-weight setting, and normalization.

We categorize existing personalized model-based PFL approaches into three groups based on their aggregation strategies for creating each personalized model: (1) Static metric-based methods: Approaches like FedAMP[Huang *et al.*, 2021] and FedPHP[Li *et al.*, 2021b] define aggregation weights using predefined metrics derived from parameter similarity or training progress. However, these predefined metrics do not directly reflect client-specific optimization objectives (e.g., local losses or gradients). (2) Local gradient-based methods: Approaches such as APPLE[Luo and Wu,

[*]Corresponding Author

2022] and FedALA[Zhang *et al.*, 2023] calculate aggregation weights locally via gradient descent. While these methods share model parameters between clients or use element-wise model combination, they incur high computational complexity and potential communication overhead. Additionally, their focus on individual client objectives limits effective cross-client coordination. (3) Auxiliary network-based methods: Approaches like pFedHN[Shamsian *et al.*, 2021] utilize Hypernetworks[Ha *et al.*, 2016] to derive personalized models or aggregation weights. However, the need to train auxiliary models introduces significant computational overhead. In summary, these limitations highlight the need for new model-aggregation techniques that not only enable adaptive personalization in non-IID scenarios but also reduce computational and communication costs.

To address these limitations, we propose a novel personaliezed model based PFL approach, **FedAPA** (Adaptive Personalized Aggregation for Federated Learning), as illustrated in Fig. 1. FedAPA introduces a server-side aggregation strategy that adaptively learns aggregation weights using gradients of client parameter changes with respect to the aggregation weights, generating aggregated personalized models for each client. Specifically, the server: (1) maintains the client parameters and an **aggregation-weight vector**, where each element quantifies the relevance of other clients' model knowledge during aggregation; (2) constructs personalized client parameters through weighted aggregation of all collaborative clients' parameters using the learned weights; and (3) updates the weight vector based on the **client-parameter delta**—the change in the client's local model parameters after multiple local update steps. As an additional feature, FedAPA adopts a partial model sharing strategy, transferring only the client's feature extractor to reduce irrelevant information during aggregation.

The key innovation of FedAPA lies in its aggregation strategy, which updates aggregation weights on the server side using the gradient of the client-parameter delta with respect to the aggregation weights, optimizing the personalized aggregation process. Half of the squared norm of the client-parameter delta serves as a proxy for the local loss, effectively capturing each client's optimization state. By dynamically adjusting aggregation weights to reduce this proxy loss, FedAPA generates an aggregated personalized model for each client, better aligning the model with the client's local objectives. Compared to existing model-aggregation approaches, FedAPA's strategy offers the following advantages: (1) Adaptability: FedAPA dynamically learns aggregation weights based on gradients derived from the proxy loss, enabling the aggregation weights to adapt effectively to each client's unique optimization trajectory. (2) Centralized Efficiency: By centralizing weight updates on the server, FedAPA minimizes communication and computational overhead while enhancing cross-client coordination. (3) Simplicity: FedAPA avoids the need for auxiliary networks, eliminating the computational complexity associated with training additional models. In summary, FedAPA's aggregation strategy effectively balances personalization and collaboration in non-IID scenarios, enhancing computational and communication efficiency.

We summarize our contributions as follows:

- We propose FedAPA, a novel model-aggregation PFL framework for heterogeneous data scenarios, featuring a server-side gradient-based model-aggregation strategy that enhances FL performance while maintaining low computational and communication overhead. while maintaining low computational and communication overhead.
- To the best of our knowledge, FedAPA is the first to adaptively update aggregation weights on the server side to generate aggregated personalized models, using gradients derived from client-parameter changes.
- We provide a theoretical convergence guarantee for the FedAPA algorithm.
- We experimentally demonstrate that our approach outperforms 10 PFL competitors across three benchmarks under two non-IID settings, achieving top accuracy, optimal computational efficiency, and low communication overhead. Ablation studies highlight the critical contribution of the aggregation strategy and the complementary benefits of the partial model sharing strategy.

2 Related Work

2.1 Personalized Model-based PFL Approaches

Personalized model-based PFL approaches, also known as PFL approaches via learning personalized local models, directly create personalized models for individual clients, typically without the need for a global model. These approaches employ various aggregation strategies to construct each personalized model, which can be broadly categorized as follows: (1) Static metric-based methods: These methods define aggregation weights using predefined, static metrics. For example, FedAMP [Huang *et al.*, 2021] calculates similarity as weights based on the Euclidean distance of model parameters, while FedPHP [Li *et al.*, 2021b] measures relevance using training progress. Although straightforward, these metrics do not directly align with client-specific optimization objectives, limiting their adaptability to non-IID data distributions. (2) Local gradient-based methods: They learn aggregation weights via gradient descent on the client side. For example, APPLE [Luo and Wu, 2022] learns inter-client relations by sharing model parameters between clients, leading to high communication costs and privacy concerns. FedALA [Zhang *et al.*, 2023] computes element-wise combination weights between client and global models, resulting in high computational complexity. While these methods utilize non-local information, their focus on local objectives limits cross-client coordination and adaptability to data heterogeneity. (3) Hypernetwork-based methods: Hypernetworks [Ha *et al.*, 2016] have been utilized as the auxilliary networks to enhance personalization. For example, pFedHN [Shamsian *et al.*, 2021] employs Hypernetworks to generate personalized models However, these methods incur significant computational overhead due to the complexity of training Hypernetworks

2.2 Global Model-based PFL Approaches

Global model-based PFL approaches, also referred to as PFL approaches via global model personalization, involve training a global model first and then personalizing it to derive local models for individual clients. For example, FedPAC [Xu *et al.*, 2023] leverages a probabilistic framework for personalization using Bayesian neural networks and a global model, allowing personalized models to adaptively incorporate uncertainty during client-specific training. DBE [Zhang *et al.*, 2024] enhances the generalization of the global feature extractor through bi-directional knowledge transfer, reducing representation bias and fine-tuning client models for improved personalization.

2.3 Other PFL Approaches

PFL can also leverage non-model information, such as prototypes, to achieve personalization. For example, FedProto [Tan *et al.*, 2022] leverages local prototypes to construct global prototypes, which are used to improve feature extraction and classification. As a combined method, FedGH[Yi *et al.*, 2023] uses client-uploaded prototypes (i.e., class-averaged representations) to train a shared global head on the server, which is then downloaded to replace local heads.

3 Method

3.1 Problem Formulation

Personalized Federated Learning (PFL) aims to collaboratively train personalized models for a set of clients, each with its own private data. PFL addresses the challenge of data heterogeneity across clients, ensuring that each client's model is tailored to its specific data while maintaining data privacy and improving individual model performance.

Let M be the total number of clients participating in the federated learning process. Each client $i \in \{1, \cdots, M\}$ has a local dataset D_i; with its own data distribution P_i on $X \times Y$. In PFL, each client i seeks to learn a personalized model parameterized by ω_i. The objective function for PFL can be formulated as:

$$\arg \min_{\omega_1, \cdots, \omega_M} \sum_{i=1}^{M} \frac{|D_i|}{N} L_i \left(F\left(\omega_i; x\right), y\right) \tag{1}$$

where $|D_i|$ is the size of the local dataset Di; N is the total number of data points across all clients; $F(\omega_i; x)$ is the model parameterized by ω_i evaluated on input x; y is the true label for input x; and L_i is the local loss function for client i, such as the following cross-entropy loss function:

$$\begin{aligned} &L_{CE,i}\left(F\left(\omega_i; x\right), y\right) \\ &\quad = -\frac{1}{|D_i|} \sum_{(x,y) \in D_i} \sum_{c=1}^{C} y_c \log \left(F\left(\omega_i; x\right)_c\right) \end{aligned} \tag{2}$$

where C is the number of classes; y_c is the binary indicator (0 or 1) if class label c is the correct classification for input x; and $F(\omega_i; x)_c$ is the predicted probability of input x belonging to class c under the model parameterized by ω_i.

3.2 FedAPA Algorithm

The framework of FedAPA is demonstrated in Fig1. Each FedAPA client uploads its feature extractor, receives a personalized feature extractor after server-side aggregation, and then updates the entire client model locally, including both the feature extractor and the classifier. On the server side, the model parameters of each client's feature extractor and the corresponding aggregation-weight vector are maintained. The core of FedAPA lies in the server-side aggregation, where the server constructs a personalized feature extractor for each client by adaptively aggregating all clients' model parameters using weights updated via gradient descent.

FedAPA adjusts each aggregation weight vector on the server using the gradient calculated from the client parameter delta, which represents the change in local model parameters after multiple local updates within a communication round. At the end of each round, the updated client parameters can be considered as an approximation of the local task's optimal solution, with the client-parameter delta approximately reflecting the local optimization direction (as detailed in subsubsection 3.2). For each client, the server iteratively updates the client's aggregation-weight vector based on the gradient of the client-parameter delta with respect to the vector. The learned weights represent adaptively acquired knowledge contributions from other clients, which may potentially reflect the data distribution correlation among clients. By aggregating beneficial model knowledge from collaborative clients, FedAPA addresses non-IID data challenges for individual clients effectively.

Local Training

As shown in Fig1, for each client i, the local model $F(\omega_i)$ is partitioned into two parts: the feature extractor $f(\theta_i)$, which takes the input sample x and produces the representation $z = f(\theta_i; x)$, and the decision layer $h(\phi_i)$, which processes the representation z and outputs the predicted label $\hat{y} = h(\phi_i; z)$. The shared parameter θ_i is uploaded to the server, while the private parameter ϕ_i is kept locally. Thus, the local objective function in PFL can be expressed as

$$\arg \min_{\omega_i} \mathcal{L}_i \left(F\left(\omega_i; x\right), y\right) = \arg \min_{\theta_i, \phi_i} \mathcal{L}_i(h(\phi_i; f(\theta_i; x)), y) \tag{3}$$

Upon downloading the parameters $\bar{\theta}_i$ as the current feature extractor, client i obtains its current entire model $\omega_i := [\theta_i; \phi_i]$. Each client conducts several local update steps on the local data using the following update rule:

$$\omega_i^{(t+1)} \leftarrow \omega_i^{(t)} - \alpha_i \nabla_{\omega_i} L_i \left(F(\omega_i^{(t)}; x), y\right) \tag{4}$$

where α_i denotes the learning rate of the local model parameters ω_i. After the local updates, client i separates θ_i from the updated ω_i, and uploads the updated θ_i as the shared client parameters to the server.

Server-Side Aggregation of Client-Parameters

For each client i, FedAPA server manages a trainable aggregation-weight vector $A_i = (a_{i,1}, \ldots, a_{i,j}, \ldots, a_{i,M})^T$ where $i, j \in \{1, \cdots, M\}$, and M indicates the total number of clients in the federated learning system. Here, $a_{i,j}$ denotes the weight of client j during the weighted aggregation used to form the target parameters $\bar{\theta}_i$ for client i. During the initialization of the learning process, we set the self-weight $a_{i,i}$ to 1, while the other weights $a_{i,j}$ to 0 for $i \neq j$. For the t-th training round, the server computes the personalized parameters $\bar{\theta}_i$ for client i by performing a weighted aggregation of all clients' shared parameters according to the following formula:

$$\bar{\theta}_i = \sum_{j=1}^{M} a_{i,j} \cdot \theta_j \tag{5}$$

Server-Side Updating of Weight-Vectors

Upon receiving the client-parameters θ_i from each client i, the FedAPA server calculates the difference between the current θ_i and the previous parameter versions $\bar{\theta}_i$ stored on the server, obtaining the client-parameter delta as follows:

$$\Delta\theta_i = \theta_i - \bar{\theta}_i \tag{6}$$

Subsequently, the server leverages this client-parameter delta to update the aggregation-weight vector A_i for client i according to the following rule:

$$A_i \leftarrow A_i - \eta(\nabla_{A_i}\bar{\theta}_i)^T \Delta\theta_i \tag{7}$$

where η is the learning rate for updating the aggregation-weight vector on the server.

Equation (7) is pivotal to the FedAPA algorithm. To provide intuition behind this formula, consider the optimal solution for the local task's model parameters, $\theta_i^* = \arg\min_{\theta_i} L_i$. After multiple local updates within a communication round, θ_i can be regarded as an approximation of θ_i^*. Consequently, $\frac{1}{2}\|\theta_i^* - \bar{\theta}_i\|_2^2$ approximates $\frac{1}{2}\|\theta_i - \bar{\theta}_i\|_2^2$, which is equivalent to half of the squared norm of $\Delta\theta_i$. This measure, $\frac{1}{2}\|\theta_i^* - \bar{\theta}_i\|_2^2 = \frac{1}{2}\|\bar{\theta}_i - \theta_i^*\|_2^2$, can be interpreted as a surrogate for the local loss. Thus, half of the squared norm of $\Delta\theta_i$ serves as a proxy for the local loss L_i, where $\Delta\theta_i$ can be interpreted as $-\nabla_{\bar{\theta}_i} L_i$. According to the chain rule, $(\nabla_{A_i}\bar{\theta}_i)^T \Delta\theta_i$ represents $-\nabla_{A_i} L_i$. Consequently, Equation (7) updates the aggregation weights A_i on the server in a way that minimizes the local loss L_i for each client, ensuring that the personalized aggregation aligns with the client-specific optimization trajectory.

Before sending the updated aggregation-weight vector A_i to client i , the server further optimizes the weights using following three post-processing steps: (1) Clipping: To improve training stability and convergence, we control the update range of each weight element in the vector A_i by clipping each element value $a_{i,j}$ to be within $[0, 1]$, where $j \in \{1, \cdots, M\}$. (2) Self-weight adjusting: Each client should prioritize its own local knowledge over that of other clients. Therefore, for the clipped aggregation-weight vector of each client i, we set the weight of the element $a_{i,i}$ (also known as the self-weight μ) to 0.5, ensuring it is not less than the weights of the other elements in the vector. (3) Normalization: To balance client contributions and minimize variance during aggregation, thus improving training generalization, we normalize each aggregation-weight vector from the previous step to ensure that the total sum of the weights equals 1.

Algorithm 1 FedAPA

Input: dataset $\{\mathcal{D}_1, \ldots, \mathcal{D}_M\}$, learning rates η and α_i, total communication rounds T, local rounds E, client-parameter set $\{\theta_1, \ldots, \theta_M\}$, aggregation-weight vector set $\{A_1, \ldots, A_M\}$.
Output: Trained personalized model set $\{\bar{\theta}_1, \bar{\theta}_2, \ldots, \bar{\theta}_M\}$.

```
 1: ∀$i \in \{1, \cdots, M\}$, initialize $\theta_i$ on server
 2: ∀$i \in \{1, \cdots, M\}$, initialize $A_i = (a_{i,1}, \ldots, a_{i,M})^T$ on
    server
 3: Function Server Executes
 4: for each communication round $t = 1, \ldots, T$ do
 5:     Sample a subset of clients $S^t$
 6:     for each client $i \in S^t$ in parallel do
 7:         $\bar{\theta}_i \leftarrow \sum_{j=1}^{M} a_{i,j}\theta_j$ {Weighted aggregation}
 8:         $\theta_i \leftarrow ClientUpdate(\bar{\theta}_i)$
 9:         $\Delta\theta_i \leftarrow \theta_i - \bar{\theta}_i$ {Computing $\Delta\theta_i$}
10:         Update $A_i$ via Eq 7 {Updating based on $\Delta\theta_i$}
11:         $a_{i,j} \leftarrow min(max(a_{i,j}, 0), 1)$ , $j = 1, \ldots, M$
            {Clipping}
12:         $a_{i,i} \leftarrow 0.5$
13:         Normalize $A_i$
14:     end for
15: end for
16: End Function
17: Function Clientupdate($\bar{\theta}_i$)
18: Download $\bar{\theta}_i$
19: $\theta_i \leftarrow \bar{\theta}_i$.
20: $\omega_i \leftarrow [\theta_i; \phi_i]$
21: for each local epoch $e = 1, \ldots, E$ do
22:     for mini-batch $B \subseteq \mathcal{D}_i$ do
23:         $\omega_i \leftarrow \omega_i - \alpha_i \nabla_{\omega_i} L_i(B)$ {Local training}
24:     end for
25: end for
26: $[\theta_i; \phi_i] \leftarrow \omega_i$
27: return $\theta_i$ {Uploading $\theta_i$}
28: End Function
```

In summary, during a communication round for each client i, the FedAPA server first constructs the personalized client parameters $\bar{\theta}_i$ by performing a weighted aggregation of the collaborative clients' parameters. Next, the client downloads $\bar{\theta}_i$ and uploads the optimized client parameters θ_i after local training. Subsequently, the server computes the client-parameters change $\Delta\theta_i$, and uses it to update the aggregation-weight vector A_i, adaptively learning beneficial knowledge from collaborative clients. Algorithm 1 outlines the overall learning process in FedAPA.

4 Convergence Analysis

In this section, we analyze the convergence of FedAPA under suitable conditions. Additional notations used in this section are detailed in Appendix.

Assumption 1. (*Lipschitz* Smooth). Each local objective function is L_1-*Lipschitz* smooth, which also means the gradient of local objective function $\mathcal{L}$ is L_1-*Lipschitz* continuous. This assumption is reasonable as it encapsulates the differentiability and smoothness properties inherent to many common loss functions in deep learning, aligning with the mathematical prerequisites for the convergence of gradient-based optimization methods fundamental to neural network training,

$$||\nabla\mathcal{L}_{t_1} - \nabla\mathcal{L}_{t_2}|| \leq L_1||\omega_i^{(t_1)} - \omega_i^{(t_2)}||_2, \quad \forall t_1, t_2 > 0, i \in \{1, 2, \ldots, M\}. \tag{8}$$

which implies the following quadratic bound,

$$\mathcal{L}_{t_1} - \mathcal{L}_{t_2} \leq \langle \nabla\mathcal{L}_{t_2}, (\omega_i^{(t_1)} - \omega_i^{(t_2)})\rangle + \frac{L_1}{2}||\omega_i^{(t_1)} - \omega_i^{(t_2)}||_2^2, \forall t_1, t_2 > 0, i \in \{1, 2, \ldots, M\}. \tag{9}$$

Assumption 2. (Unbiased Gradient and Bounded Variance). The stochastic gradient $gr_i^{(t)} = \nabla\mathcal{L}(\omega_i^{(t)}, \xi_i)$ is an unbiased estimator of the local gradient for each client, where the random variable ξ_i follows the distribution $\mathcal{D}_i$. This foundational assumption is pivotal, as it ensures that the expectation of the stochastic gradient, when averaged over the local dataset, corresponds to the true gradient of the local loss function. Despite the inherent variability in data distribution across different clients, this property remains intact, underscoring its relevance in Federated Learning where data heterogeneity is often the rule rather than the exception,

$$\mathbb{E}_{\xi_i \sim \mathcal{D}_i}[gr_i^{(t)}] = \nabla\mathcal{L}(\omega_i^{(t)}) = \nabla\mathcal{L}_t, \forall i \in \{1, 2, \ldots, M\}, \tag{10}$$

and its variance is bounded by the constant σ^2:

$$\mathbb{E}[||gr_i^{(t)} - \nabla\mathcal{L}(\omega_i^{(t)})||_2^2] \leq \sigma^2, \quad \forall i \in \{1, 2, \ldots, M\}, \sigma^2 \geq 0. \tag{11}$$

Assumption 3. (*Lipschitz* Continuity). Each local loss function $\mathcal{L}$ is L_2-*Lipschitz* continuous with respect to θ. This assumption is deemed reasonable, akin to assumption 1, as it reflects a common property of loss functions, ensuring the boundedness of the gradient's impact on parameter updates, which is a prerequisite for the stability and convergence of optimization algorithms,

$$||\mathcal{L}(\theta_i^{(t_1)}) - \mathcal{L}(\theta_i^{(t_2)})|| \leq L_2||\theta_i^{(t_1)} - \theta_i^{(t_2)}||_2, \quad \forall t_1, t_2 > 0, i \in \{1, 2, \ldots, M\}. \tag{12}$$

Theorem 1. (One-round Deviation Bound). Let Assumptions 1 to 3 hold. For an arbitrary client, after every communication round, we have,

$$\mathbb{E}[\mathcal{L}_{(t+1)E+\frac{1}{2}}] \leq \mathbb{E}[\mathcal{L}_{tE+\frac{1}{2}}] - (\alpha - \frac{L_1\alpha^2}{2})\sum_{e=\frac{1}{2}}^{E-1}||\nabla\mathcal{L}_{tE+e}||_2^2 + \frac{L_1E\alpha^2}{2}\sigma^2 + 2L_2\eta(t+1)Max^3 \tag{13}$$

where $Max = \underset{\substack{i=1,2,\ldots,M \\ t=1,2,\ldots}}{max}||\theta_i^{(t)}||$.

This theorem indicates the deviation bound of the local objective function for an arbitrary client after each communication round. Convergence of each client's local objective function can be guaranteed when there is a certain expected one-round decrease, which can be achieved by choosing appropriate η and α. **The proof of Theorem 1 is in Appendix.**

5 Experiments

5.1 Experimental Setup

Datasets. We employ three public image-classification datasets of varying complexities: FMNIST (simple), CIFAR-10 (medium), and CIFAR-100 (complex). Experiments are conducted with 20 and 50 clients, respectively. For each dataset, we construct heterogeneous scenarios following the two widely-used settings: the pathological and the practical settings[McMahan *et al.*, 2017; Li *et al.*, 2021a]. For the pathological setting, we allocate disjoint and unbalanced data of 2/2/10 classes to each client out of a total of 10/10/100 classes on FMNIST/CIFAR-10/CIFAR-100 datasets. For the practical setting, we use a Dirichlet distribution[Yurochkin *et al.*, 2019], denoted as $Dir(\alpha)$, to sample data points from each class in a given dataset to each client, where a smaller value of the concentration parameter α reflects stronger data heterogeneity. We set α to 0.1/0.1/0.01 for FMNIST/CIFAR-10/CIFAR-100 datasets.

Competitors. We evaluated FedAPA against FedAvg, the standard non-PFL baseline, and 10 state-of-the-art PFL algorithms. The comparison includes six personalized model-based methods (FedPHP, FedAMP, APPLE, FedALA, pFedHN), two global model-based approaches (FedPAC and DBE), and two other PFL approaches (FedProto, and FedGH). The dashed lines in Table 1 and Table 2 sequentially divide the competitor methods into these three groups.

Model and Training. In all experiments, each client uses the same LeNet-5 architecture, with the first four layers serving as shared feature extraction layers with the server, and the final fully connected (FC) layer acting as a private decision layer. To accommodate the size and color channels of the images in each dataset, we set the input channels/FC layer input dimension/output channels of each client model to 1/256/10 for FMNIST, 3/400/10 for CIFAR-10, and 3/400/100 for CIFAR-100. We utilize the default training-to-test set ratio of each dataset, namely 6:1 for FMNIST and 5:1 for CIFAR-10 and CIFAR-100. Unless otherwise specified, we use the following training settings. we use SGD with a momentum of 0.9 as the client optimizer. We run two local training epochs in each iteration with a batch size of 64 and a learning rate α = 0.01 for 50 communication rounds with the server. Each round involves participation from at least 60% of all users. For our FedAPA algorithm, we initialize the self-weight μ in aggregation-weight vectors to 0.5 and set the learning rate η for updating weight vectors to 0.01.

Implementation. The experiments were conducted on a workstation running Ubuntu 22.04 LTS. The implementation utilized Python 3.8.10, CUDA version 11.3, and the PyTorch 1.11.0 deep learning framework. The workstation features a 12 vCPU Intel Xeon Platinum 8255C @ 2.50GHz CPU and an NVIDIA GeForce RTX 2080 Ti GPU.

	Approach	Pathological Non-IID			Practical Non-IID		
		FMNIST (Simple)	CIFAR-10 (Medium)	CIFAR-100 (Complex)	FMNIST (Simple)	CIFAR-10 (Medium)	CIFAR-100 (Complex)
		$c = 2$	$c = 2$	$c = 10$	$\alpha = 0.1$	$\alpha = 0.1$	$\alpha = 0.01$
#Client = 20	FedAvg [McMahan *et al.*, 2017]	75.08	48.48	18.49	86.96	56.68	20.39
	FedPHP [Li *et al.*, 2021b]	99.17	84.40	52.02	95.33	82.52	62.09
	FedAMP [Huang *et al.*, 2021]	99.26	86.31	56.94	96.44	82.38	66.43
	APPLE [Luo and Wu, 2022]	99.06	84.69	54.67	95.42	79.79	61.83
	FedALA [Zhang *et al.*, 2023]	99.33	87.52	**59.75**	96.19	82.53	67.24
	pFedHN [Shamsian *et al.*, 2021]	99.29	86.12	57.43	95.95	80.71	64.55
	pFedLA [Ma *et al.*, 2022]	98.85	78.85	41.96	95.30	82.86	50.02
	FedPAC [Xu *et al.*, 2023]	**99.41**	86.18	56.06	96.36	84.93	61.89
	DBE [Zhang *et al.*, 2024]	99.20	86.39	57.46	95.97	83.45	63.00
	FedProto [Tan *et al.*, 2022]	99.31	86.39	57.20	95.52	82.71	66.81
	FedGH [Yi *et al.*, 2023]	99.32	86.50	57.34	95.22	80.75	66.59
	FedAPA *(ours)*	**99.35**	**87.69**	**59.24**	**96.71**	**85.85**	**68.11**
#Client = 50	FedAvg [McMahan *et al.*, 2017]	72.54	38.69	13.19	82.98	50.77	11.84
	FedPHP [Li *et al.*, 2021b]	98.90	82.85	46.20	95.27	81.73	69.33
	FedAMP [Huang *et al.*, 2021]	99.02	83.56	49.61	94.40	82.18	73.67
	APPLE [Luo and Wu, 2022]	98.71	80.97	42.51	95.21	79.44	68.10
	FedALA [Zhang *et al.*, 2023]	99.11	84.50	52.44	95.81	82.84	74.12
	pFedHN [Shamsian *et al.*, 2021]	98.96	84.47	52.50	95.75	82.76	74.50
	pFedLA [Ma *et al.*, 2022]	98.26	71.22	31.22	92.65	77.31	56.09
	FedPAC [Xu *et al.*, 2023]	97.83	85.27	51.40	95.65	81.69	71.79
	DBE [Zhang *et al.*, 2024]	99.04	84.26	51.43	95.51	82.81	72.24
	FedProto [Tan *et al.*, 2022]	99.14	83.28	50.16	95.45	81.73	74.86
	FedGH [Yi *et al.*, 2023]	99.07	83.37	49.84	94.87	79.61	73.04
	FedAPA *(ours)*	**99.15**	**85.33**	**52.62**	**95.95**	**84.14**	**75.09**

Table 1: Test accuracy (%) of FedAPA and 11 competitors on Fashion MNIST, CIFAR-10, and CIFAR-100 in pathological and practical non-IID settings for 20 and 50 collaborative clients. c denotes the number of classes held by each client in pathological settings. α is the concentration parameter of the Dirichlet distribution used in practical settings.

5.2 Accuracy Evaluation

To evaluate and compare the performance of FedAPA with the competitors, we ran each FL algorithm and computed the average accuracy from three random data allocations. Accuracy was calculated as the ratio of correct to total predictions. Table 1 shows the test accuracy of all methods across 12 scenarios, covering three datasets, two client numbers, and two non-IID settings. It also includes the number of classes c per client for the pathological settings, and the concentration parameter α of the Dirichlet distribution for the practical settings. As shown in Table 1, FedAPA outperforms all competitors in test accuracy across 10 out of 12 scenarios. The only exceptions are the pathological non-IID settings on the FMNIST and CIFAR-100 datasets with 20 clients, where FedAPA trails the top-performing methods by just 0.06% (vs. FedPAC) and 0.51% (vs. FedALA), respectively. FedAPA demonstrates a more pronounced performance advantage in practical non-IID settings, which better reflect realistic heterogeneous data distributions. Overall, FedAPA consistently exhibits superior performance compared to all competing methods.

5.3 Computation and Communication Overhead

The computation times for each FL algorithm are summarized in the computation column of Table 2, showing total time (in minutes) and time per iteration (in seconds) for 20 clients using the medium CIFAR-10 dataset in the practical non-IID setting with 2 training epochs. As shown, FedAPA requires 17.66 (approximately 18) seconds per iteration, comparable to the baseline FedAvg. Thus, FedAPA incurs only an additional 0.34 seconds per iteration while achieving significant accuracy improvements. It offers state-of-the-art performance with minimal computation overhead among all compared PFL algorithms.

The communication costs per iteration are listed in the communication column of Table 2. Prototype-based methods, such as FedProto and FedGH, incur the lowest communication overhead by transmitting prototypes instead of model parameters. FedAPA, while achieving higher accuracy, maintains communication efficiency by transmitting only partial model parameters, resulting in the least communication overhead among all non-prototype FL algorithms. In summary, FedAPA achieves superior accuracy with the lowest communication cost among non-prototype methods.

5.4 Ablation Study

(1) FedAPA incorporates the **Adaptive Personalized Aggregation (APA)** strategy and the **Partial Model Sharing (PMS)** strategy. To assess the contributions of these two strategies, we conduct an ablation study, as shown in the

Approach	**Acc.**	**Comput.**		**Commun.**
		Total time	Time per iter.	(Practice) per iter.
	%	*min.*	*sec.*	*KB*
FedAvg [McMahan *et al.*, 2017]	56.68	***42***	***17***	484
FedPHP [Li *et al.*, 2021b]	82.52	50	20	484
FedAMP [Huang *et al.*, 2021]	82.38	46	19	484
APPLE [Luo and Wu, 2022]	79.79	50	20	5087
FedALA[Zhang *et al.*, 2023]	82.53	67	27	484
pFedHN [Shamsian *et al.*, 2021]	80.71	48	19	478
pFedLA [Ma *et al.*, 2022]	82.86	89	35	484
FedPAC [Xu *et al.*, 2023]	84.93	161	64	553
DBE [Zhang *et al.*, 2024]	83.45	44	18	484
FedProto [Tan *et al.*, 2022]	82.71	66	26	**5**
FedGH [Yi *et al.*, 2023]	80.75	63	25	**5**
FedAPA *(ours)*	**85.85**	**44**	**18**	**478**

Table 2: Comparison of test accuracy (%), computation time (minutes/seconds), and communication overhead (parameters per iteration per client) on CIFAR-10 (20 clients, practical Non-IID setting, $\alpha = 0.1$).

Method	**ΔAcc.(%)**	**Acc.(%)**
Baseline *(FedAvg)*	–	56.68
Baseline **+ APA**	↑ **28.25**	84.93
Baseline + APA **+ PMS** *(FedAPA)*	↑ 28.25 **+ 0.92**	85.85
FedAPA *(ours)*	–	85.85
w/o **Clipping**	↓ **2.58**	83.27
w/o **Self-weight**	↓ **1.44**	84.41
w/o **Normalization**	↓ **5.51**	80.34
w/o **All**	↓ **26.14**	59.71

Table 3: Ablation results of 2 strategies and 3 post-processing steps to FedAPA's accuracy (Acc.) on CIFAR-10 (20 clients, practical Non-IID setting).

top half of Table 3. Starting with FedAvg as the baseline, which uses average aggregation and entire-model sharing, we achieve an accuracy of 56.68%. Introducing the APA strategy significantly improves accuracy by 28.25%, reaching 84.93%. Adding the PMS strategy further enhances accuracy by 0.92%, resulting in a final accuracy of 85.85%. These results underscore the pivotal contribution of the APA strategy as the primary driver of performance improvement, while the PMS strategy offers a smaller, supplementary boost. (2) Following the server-side gradient update of the weight vectors, FedAPA applies three post-processing steps to the weights: clipping, self-weight adjustment, and normalization. Ablation studies in the lower half of Table 3 show that removing each step individually reduces accuracy by 2.58%, 1.44%, and 5.51%, respectively, with a significant 26.14% drop when all are omitted. These results highlight the importance of each step in improving model accuracy.

5.5 Convergence Curve Analysis

The convergence curves in Fig2 show the performance of the 12 methods over training rounds. FedAPA reaches convergence within approximately 10 rounds, demonstrating its rapid convergence rate compared to the competing methods. The curve for FedAPA exhibits a smooth progress towards convergence, indicating stable training and consistent perfor-

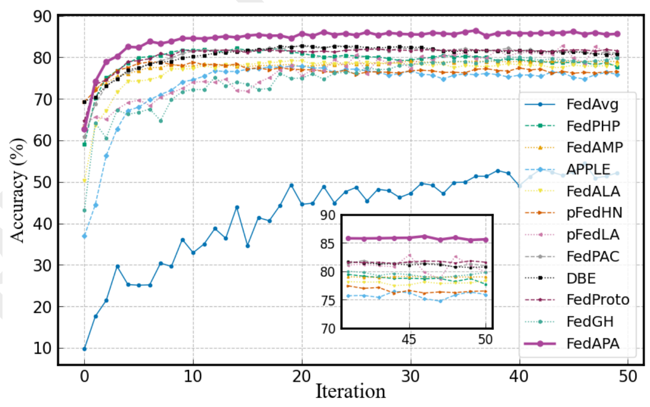

Figure 2: Convergence curves of FedAPA and 11 compared methods on CIFAR-10 (20 clients, practical Non-IID setting).

μ	0	0.3	**0.5**	0.7	1
Accuracy	84.41	84.76	**85.85**	84.02	82.95
drop	1.44	1.09	**-**	1.83	2.90

Table 4: Accuracy of FedAPA with varying self-weight μ on CIFAR-10 (20 clients, practical Non-IID setting).

mance improvement. Upon convergence, FedAPA achieves an accuracy of around 86%, which is higher than all other competitors. These results confirm that FedAPA is one of the fastest-converging algorithms, achieving the highest accuracy with stable convergence behavior.

5.6 Effect of Self-weight μ

FedAPA adjusts the self-weight of each aggregation-weight vector using the hyperparameter μ. We evaluate the effect of μ on accuracy by varying it from 0 to 1, using 20 clients on CIFAR-10 in a Practical Non-IID setting. As shown in Table 4, the best accuracy is achieved when $\mu = 0.5$. Lower values underweight local knowledge, while higher values overly emphasize local tasks, reducing the benefits of shared knowledge. Thus, $\mu = 0.5$ balances local and shared contributions effectively.

6 Conclusion

In this paper, we introduce a new PFL framework, FedAPA, designed to automatically consolidate desired knowledge from collaborative clients to form personalized local models. FedAPA adaptively learns aggregation weights on the server by utilizing gradients derived from changes in client parameters. We provide a theoretical proof for the convergence of our FedAPA algorithm. Experimental results under heterogeneous data scenarios demonstrate that FedAPA achieves higher accuracy and lower computational overhead compared to the competing PFL methods, while maintaining communication efficiency. These contributions highlight the effectiveness and efficiency of FedAPA in improving PFL performance, suggesting its potential suitability for handling large-scale heterogeneous data in practical applications.

Acknowledgements

This research was partially supported by funding from Guangdong Basic and Applied Basic Research Foundation (No. 2021A1515012297), the Hong Kong RGC General Research Fund (No. 152211/23E, 15216424/24E, and 152115/25E), the PolyU Internal Fund (No. P0056171), and the Huawei Gifted Fund.

References

[Ha *et al.*, 2016] David Ha, Andrew M. Dai, and Quoc V. Le. Hypernetworks. *CoRR*, abs/1609.09106, 2016.

[Huang *et al.*, 2021] Yutao Huang, Lingyang Chu, Zirui Zhou, Lanjun Wang, Jiangchuan Liu, Jian Pei, and Yong Zhang. Personalized cross-silo federated learning on non-iid data. *Proceedings of the AAAI Conference on Artificial Intelligence*, 35(9):7865–7873, May 2021.

[Li *et al.*, 2021a] Qinbin Li, Bingsheng He, and Dawn Song. Model-contrastive federated learning. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*, pages 10713–10722, June 2021.

[Li *et al.*, 2021b] Xin-Chun Li, De-Chuan Zhan, Yunfeng Shao, Bingshuai Li, and Shaoming Song. Fedphp: Federated personalization with inherited private models. In Nuria Oliver, Fernando Pérez-Cruz, Stefan Kramer, Jesse Read, and Jose A. Lozano, editors, *Proceedings of the Joint European Conference on Machine Learning and Knowledge Discovery in Databases. Research Track*, pages 587–602, Cham, 2021. Springer International Publishing.

[Luo and Wu, 2022] Jun Luo and Shandong Wu. Adapt to adaptation: Learning personalization for cross-silo federated learning. In Lud De Raedt, editor, *Proceedings of the Thirty-First International Joint Conference on Artificial Intelligence, IJCAI-22*, pages 2166–2173. International Joint Conferences on Artificial Intelligence Organization, 7 2022. Main Track.

[Ma *et al.*, 2022] Xiaosong Ma, Jie Zhang, Song Guo, and Wenchao Xu. Layer-wised model aggregation for personalized federated learning. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*, pages 10092–10101, June 2022.

[McMahan *et al.*, 2017] Brendan McMahan, Eider Moore, Daniel Ramage, Seth Hampson, and Blaise Aguera y Arcas. Communication-Efficient Learning of Deep Networks from Decentralized Data. In Aarti Singh and Jerry Zhu, editors, *Proceedings of the 20th International Conference on Artificial Intelligence and Statistics*, volume 54 of *Proceedings of Machine Learning Research*, pages 1273–1282. PMLR, 20–22 Apr 2017.

[Shamsian *et al.*, 2021] Aviv Shamsian, Aviv Navon, Ethan Fetaya, and Gal Chechik. Personalized federated learning using hypernetworks. In Marina Meila and Tong Zhang, editors, *Proceedings of the 38th International Conference on Machine Learning*, volume 139 of *Proceedings of Machine Learning Research*, pages 9489–9502. PMLR, 18–24 Jul 2021.

[Tan *et al.*, 2022] Yue Tan, Guodong Long, LU LIU, Tianyi Zhou, Qinghua Lu, Jing Jiang, and Chengqi Zhang. Fedproto: Federated prototype learning across heterogeneous clients. *Proceedings of the AAAI Conference on Artificial Intelligence*, 36(8):8432–8440, Jun. 2022.

[Wu *et al.*, 2020] Qiong Wu, Kaiwen He, and Xu Chen. Personalized federated learning for intelligent iot applications: A cloud-edge based framework. *IEEE Open Journal of the Computer Society*, 1:35–44, 2020.

[Xu *et al.*, 2023] Jian Xu, Xinyi Tong, and Shao-Lun Huang. Personalized federated learning with feature alignment and classifier collaboration. *arXiv preprint arXiv:2306.11867*, 2023.

[Yang *et al.*, 2019] Qiang Yang, Yang Liu, Tianjian Chen, and Yongxin Tong. Federated machine learning: Concept and applications. *ACM Trans. Intell. Syst. Technol.*, 10(2), jan 2019.

[Yi *et al.*, 2023] Liping Yi, Gang Wang, Xiaoguang Liu, Zhuan Shi, and Han Yu. Fedgh: Heterogeneous federated learning with generalized global header. In *Proceedings of the 31st ACM International Conference on Multimedia*, MM '23, page 8686–8696, New York, NY, USA, 2023. Association for Computing Machinery.

[Yurochkin *et al.*, 2019] Mikhail Yurochkin, Mayank Agarwal, Soumya Ghosh, Kristjan Greenewald, Nghia Hoang, and Yasaman Khazaeni. Bayesian nonparametric federated learning of neural networks. In Kamalika Chaudhuri and Ruslan Salakhutdinov, editors, *Proceedings of the 36th International Conference on Machine Learning*, volume 97 of *Proceedings of Machine Learning Research*, pages 7252–7261. PMLR, 09–15 Jun 2019.

[Zhang *et al.*, 2023] Jianqing Zhang, Yang Hua, Hao Wang, Tao Song, Zhengui Xue, Ruhui Ma, and Haibing Guan. Fedala: Adaptive local aggregation for personalized federated learning. *Proceedings of the AAAI Conference on Artificial Intelligence*, 37(9):11237–11244, Jun. 2023.

[Zhang *et al.*, 2024] Jianqing Zhang, Yang Hua, Jian Cao, Hao Wang, Tao Song, Zhengui XUE, Ruhui Ma, and Haibing Guan. Eliminating domain bias for federated learning in representation space. In A. Oh, T. Naumann, A. Globerson, K. Saenko, M. Hardt, and S. Levine, editors, *Advances in Neural Information Processing Systems*, volume 36, pages 14204–14227. Curran Associates, Inc., 2024. [Online]. Available: https://dl.acm.org/doi/10.5555/3666122.3666747.