Optimizing Personalized Federated Learning Through Adaptive Layer-Wise Learning

Weihang Chen[1] , **Cheng Yang**[1] , **Jie Ren**[1*] , **Zhiqiang Li**[1] , **Zheng Wang**[2]
[1]Shaanxi Normal University, China
[2]University of Leeds, United Kingdom
{cwh,yangcheng,renjie,lizq}@snnu.edu.cn
z.wang5@leeds.ac.uk

Abstract

Real-life deployment of federated Learning (FL) often faces non-IID data, which leads to poor accuracy and slow convergence. Personalized FL (pFL) tackles these issues by tailoring local models to individual data sources and using weighted aggregation methods for client-specific learning. However, existing pFL methods often fail to provide each local model with global knowledge on demand while maintaining low computational overhead. Additionally, local models tend to over-personalize their data during the training process, potentially dropping previously acquired global information. We propose FLAYER, a novel layer-wise learning method for pFL that optimizes local model personalization performance. FLAYER considers the different roles and learning abilities of neural network layers of individual local models. It incorporates global information for each local model as needed to initialize the local model cost-effectively. It then dynamically adjusts learning rates for each layer during local training, optimizing the personalized learning process for each local model while preserving global knowledge. Additionally, to enhance global representation in pFL, FLAYER selectively uploads parameters for global aggregation in a layer-wise manner. We evaluate FLAYER on four representative datasets in computer vision and natural language processing domains. Compared to eight state-of-the-art pFL methods, FLAYER improves the inference accuracy, on average, by 5.20% (up to 14.29%). The code is available at https://github.com/lancasterJie/FLAYER/.

1 Introduction

Federated learning (FL) enables collaborative model training across diverse, decentralized data sources while preserving the confidentiality and integrity of each dataset. It is widely used in mobile applications like private face recognition [Niu and Deng, 2022], predictive text, speech recognition, and image annotation [Song *et al.*, 2022]. However, data from mobile devices frequently exhibits non-independent and identically distributed (non-IID) characteristics due to variations in user behavior, device types, or regional differences [Zhu *et al.*, 2021a]. This data heterogeneity poses significant challenges for typical FL algorithms, as the trained global model may struggle to adapt to the specific needs of individual clients, resulting in poor inference performance and slow convergence.

Personalized Federated Learning (pFL) tackles this issue by tailoring learning to each client [Tan *et al.*, 2022]. Instead of producing a single global model, pFL generates client-specific models to better capture local data nuances. Customization can be implemented via diverse strategies, including model-wise [Luo and Wu, 2022], layer-wise [Ma *et al.*, 2022], or element-wise [Zhang *et al.*, 2023b] aggregation, each offering varying degrees of granularity and control.

Model-wise aggregation methods like APPLE [Luo and Wu, 2022], FedAMP [Huang *et al.*, 2021] and Ditto [Li *et al.*, 2021b] aggregate entire client models using learned weights, enabling broad global knowledge integration but lacking granularity for data-specific variations. This approach also incurs high computational overhead for model selection, hindering scalability. Layer-wise methods address some of these challenges by allowing local aggregation in layer units. Here, a “layer unit” can be a single neural network layer or a block comprising multiple layers. Representative approaches include FedPer [Arivazhagan *et al.*, 2019], FedRep [Collins *et al.*, 2021] and pFedLA [Ma *et al.*, 2022]. These methods allow for more targeted adaptation of different parts of the network. For example, FedRep uses the global model to construct the lower layers (i.e., layers toward the input layer, also termed as base layers) of each local model and the higher layers (i.e., layers toward the model’s output layer, also termed as head layers) are built solely on local data for personalization. This design strikes a balance between global knowledge sharing and local personalization. Element-wise aggregation, such as FedALA [Zhang *et al.*, 2023b], assigns per-parameter weights for finer control, but introduces prohibitive computational/memory costs in large models. Its pre-defined static weights limit adaptability to dynamic data environments. Moreover, all above pFL methods use a constant learning rate across all clients during training, without considering the differing learning needs of various layers for each client. This may lead to over-personalization for each client, resulting in the loss of previously aggregated global information.

To address these gaps, we introduce FLAYER, a novel layer-wise optimization framework for pFL that effectively balances global representation learning with local customization. Figure 1 outlines FLAYER 's local learning process, comprising three key stages: local model initialization (incorporating both local and global insights into the head layers), adaptive layer-wise training (mitigating gradient vanishing and facilitating faster convergence), and layer-specific parameter uploading via masking (ensuring that only essential updates are shared to preserve global features). Through this layered, adaptive approach, FLAYER effectively balances personalized learning with shared global knowledge while reducing computational overhead. We evaluate FLAYER by applying it to both image [Krizhevsky *et al.*, 2009; Chrabaszcz *et al.*, 2017] and text [Zhang *et al.*, 2015] classification tasks using four widely adopted benchmarks. The results show that FLAYER outperforms eight other pFL methods in inference accuracy and computational cost. This paper makes the following contributions:

- A new performance-guided layer-wise aggregation method allows clients to dynamically incorporate both local and global information in a cost-effective manner;
- A new layer-specific adaptive learning rate scheme for pFL to steer the personalization and speed up convergence;
- A new layer-wise masking technique for selectively uploading essential parameters to the central server to improve global representation.

2 Background and Overview

2.1 Problem Definition

pFL learns personalized models cooperatively among clients. Consider a scenario where we have n clients, and each client processes their distinct private training data denoted as $D_1, D_2, \ldots, D_N$, which has different data classes and sizes. These datasets exhibit heterogeneity, characterized by non-IID [Zhao *et al.*, 2018]. The goal of pFL can be defined as:

$$\{\theta_1, \theta_2, ..., \theta_n\} = \arg\min_{\theta} \sum_{k=1}^{n} \frac{m_k}{M} L_k(\theta_k) \tag{1}$$

where θ_k is the model parameters of client k, m_k is the data size of D_k, M is the whole data size of all clients, $L_k(\theta_k)$ is the loss function of client k.

2.2 Overview of FLAYER

Figure 1 illustrates FLAYER workflow for the k-th client in iteration t. Each client begins by downloading the global model and applying an adaptive aggregation strategy that dynamically weights global and local models based on local inference accuracy, allowing stronger performers to guide head-layer initialization. During local training, a layer-wise adaptive learning rate optimizes updates for each layer according to its position and gradient, improving personalization and convergence. Finally, a layer-wise masking strategy selectively uploads only essential parameters to the server, preserving critical local insights and enhancing the global representation. Algorithm 1 details the entire FL process.

3 Methodology

3.1 Performance-guided Layer-wise Local Aggregation

Building on previous findings [Yosinski *et al.*, 2014; Zhu *et al.*, 2021b] that base layers of deep neural network (DNN) capture generalized features while head layers encode task-specific features, we introduce a differentiated update strategy for these layers. In our approach, during the local model initialization stage, the base layers of each client's model are directly updated using parameters from the global model. This ensures the consistent refinement of generalized features across all clients. For the head layers, we use a dynamic aggregation method to integrate both global and local parameters. This integration is tailored based on the performance of the k-th client's model on its local dataset D_k from the previous iteration. The process is defined as follows:

$$\tilde{\theta}_k^t := [\theta_g^{(1:L-s,t-1)}, A_{k,l}^{t-1} \odot \theta_k^{(L-s+1:L,t-1)} + A_{k,g}^{t-1} \odot \theta_g^{(L-s+1:L,t-1)}]$$
$$\text{s.t.} \quad A_{k,l} + A_{k,g} = 1 \tag{2}$$

Here, θ_k is the local model parameter matrix of the k-th client, $\tilde{\theta}_k$ denotes the local model parameter matrix of the k-th client after initialization, L is the total number of layers, s is the number of layers in the head for personalization, and $\theta_g^{(1:L-s,t-1)}$ represents the lower $L - s$ layers in the base part of the global model at iteration $t - 1$, which are used to update the base layers in the local model, and all the clients share the same base layers. $\theta_k^{(L-s+1:L,t-1)}$ denotes the head layers of the k-th local model. We aggregate the head layers from the local and global models to initialize the head layers for the local model. The aggregation weights $A_{k,g}$ and $A_{k,l}$ control the influence of global and local parameters, respectively. At iteration $t - 1$, the local model inference accuracy on dataset D_k sets the local weight $A_{k,l}^{t-1}$, with $(1 - A_{k,l})$ adjusting for global influence. The dynamic weighting for head layers helps each client gradually "specialize" its head layers, without getting stuck in unproductive local optima. When local performance is low, leaning on the global head acts like a safety net, pulling the client's parameters closer to the collective knowledge. As local performance improves, the client can afford to rely more on its own specialized parameters. FLAYER ensures convergence by aligning foundational features (base layers) and employing performance-driven head adjustments (head layers), the method fosters stable updates and mitigates divergence, enabling steady convergence.

3.2 Adaptive Layer-specific Learning Rate

After the model initialization, the local model trains on its local dataset D_k:

$$\hat{\theta}_k^t := \tilde{\theta}_k^t - \eta \nabla_{\tilde{\theta}_k} \mathcal{L}_k(\tilde{\theta}_k^t, D_k) \tag{3}$$

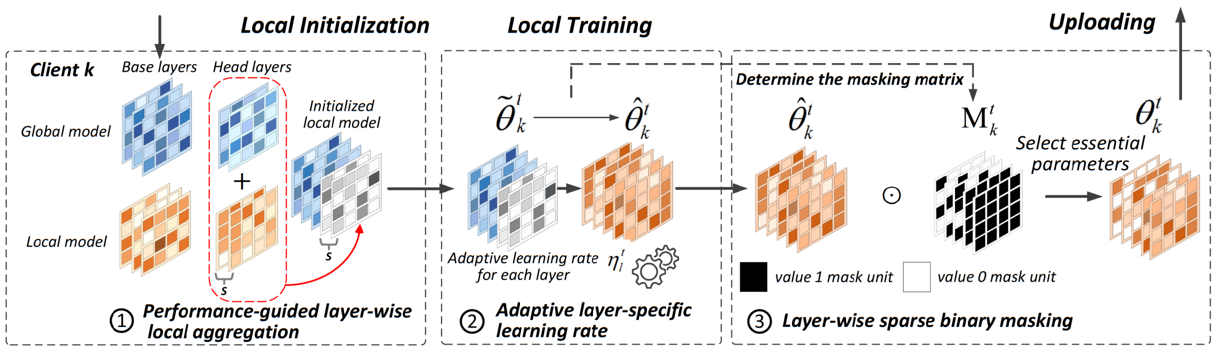

Local Initialization
Local Training
Uploading
Client k
Base layers
Head layers
Global model
Local model
Initialized local model
s
Adaptive learning rate for each layer
Determine the masking matrix
Select essential parameters
value 1 mask unit
value 0 mask unit
① Performance-guided layer-wise local aggregation
② Adaptive layer-specific learning rate
③ Layer-wise sparse binary masking

Figure 1: The local learning process of FLAYER on k-th client during the t-th iteration. In the local initialization stage, FLAYER aggregates both local and global head layers based on the local model's performance from the previous iteration. The initialized local model is then trained on the local data, using an adaptive learning rate for each layer. Based on the parameter changes before and after local training, FLAYER constructs a masking matrix to identify and select essential parameters, with different proportions per layer, for updating the global model.

η represents the learning rate, $\nabla_{\tilde{\theta}_k}\mathcal{L}_k(\tilde{\theta}_k^t, D_k)$ is the gradient of the loss function $\mathcal{L}_k$ with respect to the parameters $\tilde{\theta}_k$ evaluated using local dataset D_k.

The existing pFL methods [Luo and Wu, 2022; Li *et al.*, 2021b; Huang *et al.*, 2021; Arivazhagan *et al.*, 2019; Collins *et al.*, 2021; Zhang *et al.*, 2023b] typically employ a fixed learning rate. However, in the context of FL with non-IID data, we observe that the learning rate is a critical hyperparameter that significantly impacts both the performance and convergence speed of local models (see Section *Ablation*). Previous work [Singh *et al.*, 2015] pioneered layer-wise learning rate adjustments, primarily to mitigate the vanishing gradient issue in the lower layers of DNNs within a non-distributed training context. However, this approach is not well-suited for the pFL context. Inspired by [Luo *et al.*, 2021] and our observation (see Section *Layer Similarity*), we note that the first layer among all local models, showing the highest similarity, captures universal features and thus requires a smaller learning rate with more gradual adjustments. In contrast, deeper layers, exhibiting greater divergence, deal with more complex, client-specific features and benefit from larger learning steps, which aids local model personalization. Building on these insights, we implement an adaptive learning rate scheme for pFL that integrates layer positional information with the corresponding gradient:

$$\eta^{(i,t)} = \eta\left(1 + \log\left(1 + \frac{1}{\|g^{(i,t)}\|_2}\right) \times \frac{i}{L}\right) \tag{4}$$

where:

- η is the base learning rate.
- $g^{(i,t)}$ represents the gradient vector of the i-th layer at iteration t.
- $\|g^{(i,t)}\|_2$ is the L2 norm (Euclidean norm) of the gradient.
- L is the total number of layers.

This approach applies lower learning rates to the initial layers, which capture universal features, and larger rates to deeper layers, which handle client-specific features. Empirical results (see Section *Ablation*) demonstrate that the adaptive learning rate scheme significantly improves convergence speed and model accuracy compared to fixed learning rate strategies.

3.3 Layer-wise Sparse Binary Masking

In non-IID settings, naively averaging updated client parameters on the server can overshadow essential local information. To mitigate this issue, we propose a layer-wise binary masking scheme that selectively uploads parameters to preserve critical knowledge and maintain a robust global representation. Our approach leverages the observation that early layers capture broadly shared features across clients, whereas deeper layers learn more specialized, client-specific representations [Luo *et al.*, 2021]. Specifically, we prioritize parameters in the initial layers that undergo larger changes, as they likely embody fundamental patterns vital for model generalization. In contrast, we upload a more extensive set of parameters for deeper layers to retain a diverse range of complex, client-specific features necessary for personalized performance. Formally, we denote the upload proportion for the i-th layer as UP^i, computing it based on the layer's position within the network:

$$UP^i := \min(\max(\frac{i}{L}, 0.1), 1) \tag{5}$$

where UP^i is constrained to be at least 0.1 to ensure that every layer contributes to the global model aggregation, but not more than 1, reflecting a full update contribution.

To identify and select significant weights for sharing, we calculate the absolute weight fluctuation value of the local model within each layer after local training:

$$\Delta\theta_k^{(i,t)} := |\hat{\theta}_k^{(i,t)} - \tilde{\theta}_k^{(i,t)}| \tag{6}$$

Then, to focus on parameters that have undergone notable changes, we identify the top UP^i parameters from each layer i in k-th client model based on their fluctuation values after the t-th training round:

Algorithm 1 FLAYER

Input: N clients, ρ: client joining ratio, L: loss function, Θ_g^0: initial global model, η: base local learning rate, s: the hyperparameter of FLAYER.

Output: Well-performing local models $\tilde{\Theta}_1, \ldots, \tilde{\Theta}_N$

```
1:  Server sends $\Theta_g^0$ to all clients to initialize local models.
2:  for iteration $t = 1, \ldots, T$ do
3:      Server samples a subset $C^t$ of clients according to $\rho$.
4:      Server sends $\Theta_g^{t-1}$ to $|C^t|$ clients.
5:      for Client $k \in C^t$ in parallel do
6:          Client $k$ initializes local model $\tilde{\Theta}_k^t$ by Equation (2).
7:          Client $k$ obtains $\hat{\Theta}_k^t$ by Equation (3) - (4).
8:                                              ▷ Local model training
9:          Client $k$ obtains masked $\Theta_k^t$ by Equation (5) - (9).
10:         Client $k$ sends $\Theta_k^t$ to the server.     ▷ Uploading
11:     end for
12:     Server-side Aggregation:
13:     Server obtains $\Theta_g^t$ by $\Theta_g^t \leftarrow \sum_{k \in C^t} \frac{n_k}{\sum_{j \in C^t} n_j} \Theta_k^t$ by
        Equation (10).
14: end for
15: return $\tilde{\Theta}_1, \ldots, \tilde{\Theta}_N$
```

$$S_k^{(i,t)} := \mathit{top\_percent}\left(\Delta\theta_k^{(i,t)}, UP^i\right) \tag{7}$$

To manage which parameters are uploaded from each layer i of a local model on client k, we use a binary mask matrix, M_k^t, which has the same dimensions as the parameter matrix $\hat{\theta}_k^t$. Initially, all elements of this matrix are initialized to one. Each item $m_{j,k}^{(i,t)}$ in $M_k^{(i,t)}$ is determined by whether the corresponding parameter $\theta_{j,k}^{(i,t)}$ in $\Delta\theta_k^{(i,t)}$ belongs to the subset of parameters with the highest weight changes, $S_k^{(i,t)}$. This setup uses the following rule:

$$m_{j,k}^{(i,t)} = \begin{cases} 1, & \text{if } \theta_{j,k}^{(i,t)} \in S_k^{(i,t)} \\ 0, & \text{otherwise} \end{cases} \tag{8}$$

Finally, we obtain the essential parameter θ_k^t required for uploading by multiplying the $\hat{\theta}_k^t$ with the binary mask M_k^t.

$$\theta_k^t := \hat{\theta}_k^t \odot M_k^t \tag{9}$$

During the server aggregation in round t, each client $k \in C^t$ provides updated parameters θ_k^t and has a local dataset of size n_k. Let Θ_g^{t-1} denote the global parameters from the previous round and $\mathbb{I}$ be the indicator function. The condition $\sum_{k \in C^t} n_k \mathbb{I}(\theta_k^t[i] \neq 0) > 0$ checks if at least one client provides a non-zero update for parameter i, and ensures that we only average over those clients that have a non-zero parameter, preventing division by the total number of clients when many of them do not contribute to that parameter. The aggregation process is defined as:

$$\Theta_g^t[i] = \begin{cases} \dfrac{\sum_{k \in C^t} n_k \, \theta_k^t[i]}{\sum_{k \in C^t} n_k \, \mathbb{I}(\theta_k^t[i] \neq 0)}, & \text{if } \sum_{k \in C^t} n_k \mathbb{I}(\theta_k^t[i] \neq 0) > 0, \\ \Theta_g^{t-1}[i], & \text{otherwise.} \end{cases} \tag{10}$$

FLAYER employs selective weight sharing to balance global knowledge with client-specific adaptation, distinguishing it from FedMask [Li *et al.*, 2021a] and FedSelect [Tamirisa *et al.*, 2024]. While FedMask, which also achieves personalization using a heterogeneous binary mask with a small overhead. However, FedMask does not consider the unique characteristics of different layers, failing to capture layer-specific information. Moreover, FedMask's binary parameter aggregation is insufficient for complex tasks, such as CIFAR-100. FLAYER applies a layer-wise masking strategy informed by parameter significance and layer position. This preserves universal features in early layers and ensures rich adaptation in deeper layers. Moreover, unlike FedSelect, which uses the Lottery Ticket Hypothesis to incrementally identify and expand "winning" subnetworks, which adds complexity due to evolving subnetworks, FLAYER retains a stable global structure, thus reducing computational overhead while maintaining fine-grained personalization. This approach effectively integrates fundamental representations and diverse local insights, resulting in higher accuracy and more stable convergence in non-IID settings.

4 Evaluation Setup

4.1 Platforms and Workloads

To evaluate the performance of FLAYER, we use a four-layer CNN [McMahan *et al.*, 2017] and ResNet-18 [He *et al.*, 2016] for CV tasks, training them on three benchmark datasets: CIFAR-10, CIFAR-100 [Krizhevsky *et al.*, 2009], and Tiny-ImageNet [Chrabaszcz *et al.*, 2017]. For the NLP task, we train fastText [Joulin *et al.*, 2017] on the AG News dataset [Zhang *et al.*, 2015]. We use the Dirichlet distribution $Dir(\beta)$ with $\beta = 0.1$ [Lin *et al.*, 2020; Wang *et al.*, 2020] to model a high level of heterogeneity across client data. Following FedAvg, we use a batch size of 10 and a single epoch of local model training per iteration. We execute the training process five times for each task and calculate the geometric mean of training latency and inference accuracy until convergence. Our experiments consider 20 clients. The number of layers in the head for CNN, ResNet-18, and fastText is 1, 2 and 1, respectively. Following FedALA, we set a base learning rate of 0.1 for ResNet-18 and fastText and 0.005 for CNN during local training. All experiments were conducted on a multi-core server with a 24-core 5.7GHz Intel i9-12900K CPU and an NVIDIA RTX A5000 GPU with 24GB of GPU memory.

4.2 Competitive Baselines

We compare FLAYER with eight other pFL methods alongside FedAvg, including model-wise aggregation methods APPLE, Ditto and FedAMP, layer-wise aggregation methods FedPer, FedRep, GPFL [Zhang *et al.*, 2023a] and FedCP [Zhang *et al.*, 2023c], and element-wise FedALA

	CNN			ResNet-18			fastText
Method	CIFAR-10	CIFAR-100	Tiny-ImageNet	CIFAR-10	CIFAR-100	Tiny-ImageNet	AG News
FedAvg	59.16±0.56	33.08±0.61	18.86±0.29	86.95±0.39	37.08±0.43	20.32±0.20	80.12±0.31
APPLE (model)	89.60±0.16	54.45±0.24	39.42±0.49	89.78±0.19	57.29±0.30	43.26±0.55	95.37±0.23
Ditto (model)	89.48±0.04	47.68±0.59	33.89±0.08	88.70±0.18	48.46±0.89	36.37±0.52	94.66±0.18
FedAMP (model)	89.31±0.17	47.77±0.46	33.82±0.33	88.52±0.22	48.75±0.49	35.83±0.25	94.02±0.11
GPFL (layer)	88.05±0.19	59.38±0.42	45.05±0.39	88.86±0.20	51.06±0.39	42.28±0.47	93.78±0.12
FedPer (layer)	89.55±0.28	49.15±0.57	39.61±0.24	89.20±0.21	54.26±0.43	42.38±0.55	95.07±0.16
FedRep (layer)	90.62±0.18	51.45±0.31	41.79±0.52	90.29±0.29	53.94±0.40	45.98±0.72	96.47±0.15
FedCP (layer)	91.23±0.15	58.27±0.28	45.22±0.30	86.65±0.24	46.72±0.73	41.69±0.41	96.04±0.14
FedALA (element)	90.84±0.09	56.98±0.18	45.10±0.25	91.30±0.35	58.65±0.26	49.09±0.89	96.58±0.10
FLAYER	**91.66±0.05**	**60.50±0.33**	**45.88±0.29**	**91.68±0.21**	**60.68±0.42**	**50.12±0.36**	**98.27±0.22**

Table 1: The average inference accuracy (%) across all clients on CIFAR-10, CIFAR-100, Tiny-ImageNet and AG News.

	Model	CNN		ResNet-18	
Dataset	Method	#Iter.	Total time (s)	#Iter.	Total time (s)
CIFAR-10	FedAvg	157	1256	179	5191
	APPLE	190	6650	130	31070
	Ditto	51	1071	172	11696
	FedAMP	47	#**517**	191	7067
	GPFL	37	592	210	7980
	FedPer	156	1248	183	5307
	FedRep	169	2028	185	6845
	FedCP	231	3003	127	4064
	FedALA	152	1520	133	5187
	FLAYER	78	858	53	#**2067**
CIFAR-100	FedAvg	180	1620	181	5430
	APPLE	195	6825	25	6000
	Ditto	57	1254	101	6868
	FedAMP	61	671	173	6401
	GPFL	450	7200	442	17238
	FedPer	101	909	184	5704
	FedRep	69	828	179	6802
	FedCP	200	2600	120	3960
	FedALA	120	1200	76	2660
	FLAYER	27	#**324**	58	#**2378**
Tiny-ImageNet	FedAvg	48	2016	74	5920
	APPLE	69	9867	37	17427
	Ditto	35	3150	174	29754
	FedAMP	28	1316	84	7392
	GPFL	72	4032	73	7081
	FedPer	31	1302	78	6240
	FedRep	39	1794	115	10350
	FedCP	118	4484	75	5775
	FedALA	64	2944	48	4368
	FLAYER	16	#**896**	18	#**1782**

Table 2: The average computation cost for CV tasks.

	CNN			ResNet-18		
Hyperparameter (s)	3	2	1	3	2	1
Accuracy (%)	53.58	54.42	#**60.50**	59.80	#**60.68**	60.16

Table 3: The inference accuracy (%) of FLAYER on CIFAR-100 by using CNN and ResNet-18 with various s.

on four popular benchmark datasets in inference accuracy. In addition, we also evaluate the performance of FLAYER in terms of the computation cost, hyperparameter, layer similarity, data heterogeneity, scalability, and applicability.

5 Experimental Results

5.1 Overall Performance

Inference accuracy. Table 1 compares the inference accuracy of FLAYER with eight other pFL methods in CV and NLP domains with $Dir(0.1)$. APPLE gives the highest accuracy in the model-wise category, but with a high computation cost. FedPer uses a simple local aggregation strategy, utilizing global base layers and local head layers to initialize the local model, improving accuracy by an average of 17.66% over FedAvg. Building on this, GPFL and FedCP seek to integrate global and personalized features: GPFL adjusts base layers, while FedCP focuses on head layers. Although both outperform FedRep and FedPer with a 4-layer CNN, they fail on deeper models like ResNet-18. In contrast, FedRep extends FedPer by alternating head and base layer training, boosting accuracy by 19.28% over FedAvg. FedALA enhances FedPer by incorporating global information into local head initialization, achieving a 21.85% improvement. Previous pFL methods acknowledge the distinct roles of base and head layers in non-IID settings and integrate global and local information only at initialization. However, they fail to adapt these layers effectively during local training, limiting the capture of on-demand local information and slowing convergence. In contrast, FLAYER integrates global and local knowledge in a layer-wise manner throughout initialization, local training, and model updating, achieving the highest test accuracy among all pFL methods and improving over FedAvg by 23.31%.

Computation cost. Table 2 compares the computation cost of FLAYER with other pFL methods and FedAvg, measured by the training time required for convergence. Except for CIFAR-10 with CNN, where FedAMP and GPFL bring the lower training cost (but with poor inference accuracy) than FLAYER, our approach gives the lowest computation cost across all other tasks, reducing total training cost by an average of 58.9% (up to 80.1%) compared to FedAvg. Specifically, model-wise methods like APPLE and Ditto involve complex calculations leading to high overhead. FedRep trains the base and head layers separately, which incurs significant training costs. FLAYER effectively incorporates both local and global information across all layers, resulting in fewer rounds needed for convergence compared to FedALA, with an average reduction of 52.7% in total training time.

	Heterogeneity			Scalability			Applicability	
Methods	**Dir (0.5)**	**Dir (0.1)**	**Dir (0.01)**	**20 clients**	**50 clients**	**100 clients**	**Acc.**	**Imps.**
FedAvg	40.27±0.28	37.08±0.43	43.74±0.38	37.08±0.43	34.56±0.25	33.08±0.41	60.68±0.42	24.77
APPLE	46.22±0.23	57.29±0.30	74.52±0.19	57.29±0.30	58.09±0.24	48.46±0.32	-	-
Ditto	28.98±0.21	48.46±0.89	72.94±0.22	48.46±0.89	46.08±0.19	43.42±0.37	58.49±0.21	10.03
FedAMP	29.01±0.28	48.75±0.49	73.12±0.17	48.75±0.49	46.49±0.44	43.74±0.20	60.72±0.27	11.97
GPFL	44.32±0.30	51.06±0.42	74.59±0.21	51.06±0.42	48.30±0.29	44.61±0.32	-	-
FedPer	36.89±0.38	54.26±0.43	73.52±0.15	54.26±0.43	51.24±0.39	47.67±0.36	63.13±0.23	8.87
FedRep	36.10±0.33	53.94±0.40	75.08±0.18	53.94±0.40	50.10±0.30	45.80±0.27	61.33±0.17	7.39
FedCP	45.70±0.46	46.72±0.38	69.42±0.32	46.72±0.38	42.86±0.33	40.19±0.24	-	-
FedALA	47.11±0.22	58.65±0.26	75.24±0.11	58.65±0.26	59.46±0.23	58.80±0.41	63.55±0.58	4.90
FLAYER	#**47.47±0.34**	#**60.68±0.42**	#**77.39±0.24**	#**60.68±0.42**	#**61.70±0.30**	#**59.96±0.39**	-	-

Table 4: The inference accuracy (%) of eight FL methods across varying levels of statistical heterogeneity and scalability, and the performance improvement (%) when applying our approach to them using ResNet-18 on CIFAR-100.

5.2 Evaluation on Personalization Layers

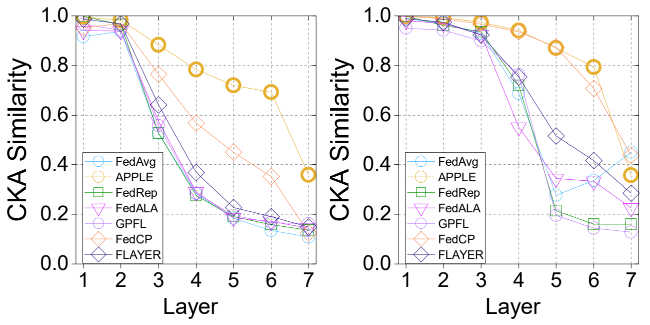
Table 3 shows inference accuracy for a 4-layer CNN and ResNet-18 with varying sizes (termed as s) of the head layers. For ResNet-18, the highest inference accuracy is achieved with s set to 2, focusing personalization on the final two layers. For the 4-layer CNN, s is set to 1, with the remaining layers updated using the global model.

5.3 Layer Similarity

We measure Centered Kernel Alignment (CKA) [Kornblith *et al.*, 2019] similarity across clients' model layers to understand how pFL methods balance personalization and generalization on non-IID data. Figure 2 compares FedAvg, APPLE, FedRep, FedALA, GPFL, FedCP, and FLAYER on CIFAR-10. After training, base-layer (layer 1, 2, 3) similarity increases for all methods, indicating effective global feature extraction. In contrast, the head layers remain more distinct, focusing on client-specific patterns. FLAYER achieves moderate similarity in both base and head layers, suggesting a balanced integration of global and local features. Additionally, under FedAvg, the last layer is a little higher than that of the second-to-last layer due to FedAvg aggregating local models by averaging their parameters. The last layer, having fewer parameters (especially in classification tasks with a moderate number of classes), experiences less variability and is more directly influenced by the averaging process.

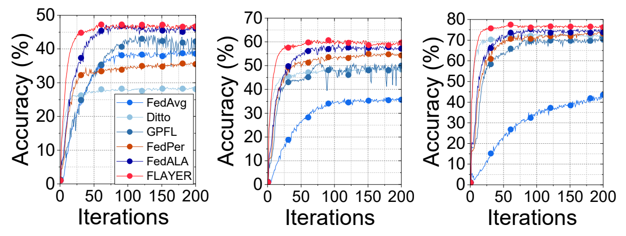
5.4 Evaluation on Data Heterogeneity

We examine the impact of statistical heterogeneity on FLAYER and other eight pFL methods, using 20 clients under three settings: $\beta = 0.5$, $\beta = 0.1$, and $\beta = 0.01$. Here, smaller β values indicate higher data heterogeneity, with $\beta = 0.1$ serving as our baseline. As summarized in Table 4, FLAYER and FedALA deliver comparable performance when heterogeneity is low ($\beta = 0.5$). However, under more challenging scenarios ($\beta = 0.01$, and $\beta = 0.1$), FLAYER consistently outperforms all competing methods, achieving the highest accuracy. Additionally, we observe that some pFL methods, such as Ditto and FedAMP, exhibit lower accuracy than FedAvg under $\beta = 0.5$. This outcome may occur because they rely too heavily on local information, overshadowing the benefits of aggregated global knowledge and thus leading to suboptimal generalization. Figure 3 presents the learning curves of FedAvg, Ditto, GPFL, FedPer, FedALA, and our approach. FLAYER converges more rapidly and achieves higher accuracy than all baselines. Moreover, as the degree of non-IID data distribution increases, all pFL methods tend to improve in accuracy, although some, such as FedRep and GFPL, require more communication rounds to converge. In contrast, FLAYER 's dynamic learning capability enables it to reach convergence even under these challenging conditions rapidly.



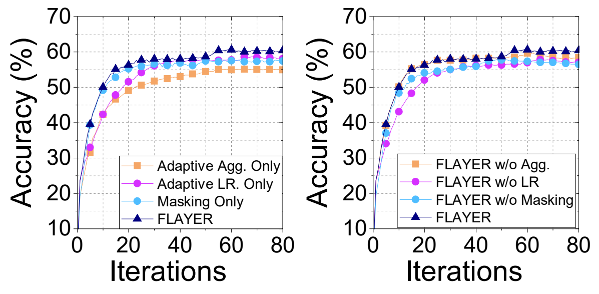
(a) ResNet-18 at round 1 (b) ResNet-18 until convergence

Figure 2: The average CKA similarities of the same layers in different local models with CIFAR-10 under $Dir(0.1)$.



(a) Dir (0.5) (b) Dir (0.1) (c) Dir (0.01)

Figure 3: Learning curves of FLAYER and five baselines on CIFAR-100 using ResNet-18 model.



(a) Single scheme (b) Removing one scheme

Figure 4: The ablation study using Resnet-18 on CIFAR-100, conducted under a $Dir(0.1)$ distribution.

5.5 Scalability

To evaluate the scalability, we vary the number of clients from 20 to 100 on CIFAR-100 with ResNet-18 under the $Dir(0.1)$ heterogeneity setting. Table 4 compares the average inference accuracy between FLAYER and other pFL methods. We can see that FLAYER consistently outperforms others across various scales of client quantity. However, we notice that the accuracy is higher with 50 clients compared to 20 clients for both FedALA and FLAYER. With 50 clients, the learning algorithm has access to more diverse training data than with 20 clients, improving the generalization ability and overall accuracy of the learned model. While a decrease in accuracy across all methods is observed as the client count increases from 50 to 100, FLAYER experiences a decline of less than 2%. In contrast, the APPLE method shows a significant drop in performance, with a 9.6% decrease in inference accuracy in the same scenario. This underlines the efficiency of FLAYER in managing larger numbers of clients, particularly in scenarios characterized by increased scalability demands.

5.6 Applicability

Our evaluation so far applied FLAYER to FedAvg. We now apply FLAYER to other FL methods to evaluate the generalization ability of our approach. Note that FLAYER does not replace the foundational architectures of an FL method. Table 4 reports the inference accuracy and improvements achieved after applying our approach to an underlying FL method. FLAYER improves the accuracy of all pFL methods, boosting the accuracy by 4.90% to 11.97%.

5.7 Ablation Study

Figure 4 illustrates how each component impacts both convergence speed and final accuracy. Figure 4(a) shows that *Agg. Only* contributes the least to convergence speed yet still improves accuracy over standard methods. For example, FedPer requires 184 iterations to converge and achieves an accuracy of 54.26%, *Agg. Only* reaches 55.11% accuracy in just 63 iterations, underscoring the benefit of even partial global aggregation over purely local strategies. Meanwhile, *LR. Only* delivers the highest overall accuracy, whereas *Masking Only* excels in accelerating convergence. Figure 4(b) indicates that combining the LR and Masking schemes yields only about a 1% drop in accuracy compared to using all three components. Notably, FLAYER achieves better accuracy at 53 iterations than its counterpart without Aggregation, suggesting that *Agg.Only* can be introduced later in training to boost performance further. These findings also highlight that *LR. Only* and *Masking Only* schemes are highly effective in non-IID settings.

6 Related Work

Previous pFL methods for managing non-IID data issues typically fall into two categories: personalizing the global model and customizing individual models for each client. Our work focuses on learning personalized models, where pFL tailors individual models to each client's data, primarily via weighted aggregation for local model adaptation.

Model-wise aggregation. Train personalized models for each client by combining clients' models using weighted aggregation. For example, FedFomo [Zhang *et al.*, 2020] employs a distance metric for weighted aggregation, while APPLE [Luo and Wu, 2022] introduces an adaptive mechanism to balance global and local objectives. FedAMP [Huang *et al.*, 2021] uses attention functions for client-specific models, and Ditto [Li *et al.*, 2021b] incorporates a proximal term for personalized models. However, existing model aggregation methods may overlook complex variations and unique characteristics in client data, leading to suboptimal personalization.

Layer-wise aggregation. Customizing different layers has been explored by FedPer [Arivazhagan *et al.*, 2019] and FedRep [Collins *et al.*, 2021], and extended by approaches like GPFL [Zhang *et al.*, 2023a] and FedCP [Zhang *et al.*, 2023c], which aim to incorporate global and personalized information within the base or head layers. pFedLA [Ma *et al.*, 2022] further uses hypernetworks to learn layer-wise aggregation weights at a high computational cost. However, all these methods overlook the influence of diverse local data on both base and head layers, constraining their potential to improve accuracy.

Element-wise aggregation. This is the most fine-grained local aggregation approach, aggregating at the parameter level. FedALA [Zhang *et al.*, 2023b] introduces an element-level aggregation weight matrix in the head layers, enhancing accuracy across various tasks. However, extra computation is required for weight calculation and does not account for the distinct roles and learning abilities of different layers during training.

7 Conclusion

We have presented FLAYER, a new layer-wise pFL approach to optimize FL in the face of non-IID data. FLAYER adaptively adjusts the aggregation weights and learning rate and selects layer-wise masking to effectively incorporate local and global information throughout all network layers. Experimental results show that FLAYER achieves the best inference accuracy and significantly reduces computational overheads compared to existing pFL methods.

Acknowledgements

This work was supported in part by the National Natural Science Foundation of China (No.62372281); Shaanxi Province 2024 Innovation Capacity Support Program (2024ZC-KJXX-014). For any correspondence, please contact Jie Ren (E-mail: renjie@snnu.edu.cn).

References

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

[Chrabaszcz *et al.*, 2017] Patryk Chrabaszcz, Ilya Loshchilov, and Frank Hutter. A downsampled variant of imagenet as an alternative to the cifar datasets. *arXiv preprint arXiv:1707.08819*, 2017.

[Collins *et al.*, 2021] Liam Collins, Hamed Hassani, Aryan Mokhtari, and Sanjay Shakkottai. Exploiting shared representations for personalized federated learning. In *International conference on machine learning*, pages 2089–2099. PMLR, 2021.

[He *et al.*, 2016] Kaiming He, Xiangyu Zhang, Shaoqing Ren, and Jian Sun. Deep residual learning for image recognition. In *Proceedings of the IEEE conference on computer vision and pattern recognition*, pages 770–778, 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. In *Proceedings of the AAAI conference on artificial intelligence*, volume 35, pages 7865–7873, 2021.

[Joulin *et al.*, 2017] Armand Joulin, Édouard Grave, Piotr Bojanowski, and Tomáš Mikolov. Bag of tricks for efficient text classification. In *Proceedings of the 15th Conference of the European Chapter of the Association for Computational Linguistics: Volume 2, Short Papers*, pages 427–431, 2017.

[Kornblith *et al.*, 2019] Simon Kornblith, Mohammad Norouzi, Honglak Lee, and Geoffrey Hinton. Similarity of neural network representations revisited. In *International conference on machine learning*, pages 3519–3529. PMLR, 2019.

[Krizhevsky *et al.*, 2009] Alex Krizhevsky, Geoffrey Hinton, et al. Learning multiple layers of features from tiny images. 2009.

[Li *et al.*, 2021a] Ang Li, Jingwei Sun, Xiao Zeng, Mi Zhang, Hai Li, and Yiran Chen. Fedmask: Joint computation and communication-efficient personalized federated learning via heterogeneous masking. In *Proceedings of the 19th ACM Conference on Embedded Networked Sensor Systems*, pages 42–55, 2021.

[Li *et al.*, 2021b] Tian Li, Shengyuan Hu, Ahmad Beirami, and Virginia Smith. Ditto: Fair and robust federated learning through personalization. In *International conference on machine learning*, pages 6357–6368. PMLR, 2021.

[Lin *et al.*, 2020] 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.

[Luo and Wu, 2022] Jun Luo and Shandong Wu. Adapt to adaptation: Learning personalization for cross-silo federated learning. In *IJCAI: proceedings of the conference*, volume 2022, page 2166. NIH Public Access, 2022.

[Luo *et al.*, 2021] Mi Luo, Fei Chen, Dapeng Hu, Yifan Zhang, Jian Liang, and Jiashi Feng. No fear of heterogeneity: Classifier calibration for federated learning with non-iid data. *Advances in Neural Information Processing Systems*, 34:5972–5984, 2021.

[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*, pages 10092–10101, 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 *Artificial intelligence and statistics*, pages 1273–1282. PMLR, 2017.

[Niu and Deng, 2022] Yifan Niu and Weihong Deng. Federated learning for face recognition with gradient correction. In *Proceedings of the AAAI Conference on Artificial Intelligence*, volume 36, pages 1999–2007, 2022.

[Singh *et al.*, 2015] Bharat Singh, Soham De, Yangmuzi Zhang, Thomas Goldstein, and Gavin Taylor. Layer-specific adaptive learning rates for deep networks. In *2015 IEEE 14th International Conference on Machine Learning and Applications (ICMLA)*, pages 364–368. IEEE, 2015.

[Song *et al.*, 2022] Congzheng Song, Filip Granqvist, and Kunal Talwar. Flair: Federated learning annotated image repository. *Advances in Neural Information Processing Systems*, 35:37792–37805, 2022.

[Tamirisa *et al.*, 2024] Rishub Tamirisa, Chulin Xie, Wenxuan Bao, Andy Zhou, Ron Arel, and Aviv Shamsian. Fedselect: Personalized federated learning with customized selection of parameters for fine-tuning. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pages 23985–23994, 2024.

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

[Wang *et al.*, 2020] Jianyu Wang, Qinghua Liu, Hao Liang, Gauri Joshi, and H Vincent Poor. Tackling the objective inconsistency problem in heterogeneous federated optimization. In *Proceedings of the 34th International*

Conference on Neural Information Processing Systems, pages 7611–7623, 2020.

[Yosinski *et al.*, 2014] Jason Yosinski, Jeff Clune, Yoshua Bengio, and Hod Lipson. How transferable are features in deep neural networks? *Advances in neural information processing systems*, 27, 2014.

[Zhang *et al.*, 2015] Xiang Zhang, Junbo Zhao, and Yann LeCun. Character-level convolutional networks for text classification. *Advances in neural information processing systems*, 28, 2015.

[Zhang *et al.*, 2020] Michael Zhang, Karan Sapra, Sanja Fidler, Serena Yeung, and Jose M Alvarez. Personalized federated learning with first order model optimization. *arXiv preprint arXiv:2012.08565*, 2020.

[Zhang *et al.*, 2023a] Jianqing Zhang, Yang Hua, Hao Wang, Tao Song, Zhengui Xue, Ruhui Ma, Jian Cao, and Haibing Guan. Gpfl: Simultaneously learning global and personalized feature information for personalized federated learning. In *Proceedings of the IEEE/CVF International Conference on Computer Vision*, pages 5041–5051, 2023.

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

[Zhang *et al.*, 2023c] Jianqing Zhang, Yang Hua, Hao Wang, Tao Song, Zhengui Xue, Ruhui Ma, and Haibing Guan. Fedcp: Separating feature information for personalized federated learning via conditional policy. In *Proceedings of the 29th ACM SIGKDD Conference on Knowledge Discovery and Data Mining*, pages 3249–3261, 2023.

[Zhao *et al.*, 2018] 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.

[Zhu *et al.*, 2021a] Hangyu Zhu, Jinjin Xu, Shiqing Liu, and Yaochu Jin. Federated learning on non-iid data: A survey. *Neurocomputing*, 465:371–390, 2021.

[Zhu *et al.*, 2021b] 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.