

pFedLoRA: Model-Heterogeneous Personalized Federated Learning with LoRA Tuning

Liping Yi

College of C.S., TMCC, SysNet,
DISSec, GTIISC
Nankai University
Tianjin, China
yiliping@nbjl.nankai.edu.cn

Han Yu*

School of Computer Science and
Engineering
Nanyang Technological University
Singapore
han.yu@ntu.edu.sg

Gang Wang*

College of C.S., TMCC, SysNet,
DISSec, GTIISC
Nankai University
Tianjin, China
wgzwp@nbjl.nankai.edu.cn

Xiaoguang Liu

College of C.S., TMCC, SysNet,
DISSec, GTIISC
Nankai University
Tianjin, China
liuxg@nbjl.nankai.edu.cn

Xiaoxiao Li

xiaoxiao.li@ece.ubc.ca
Electrical and Computer Engineering
Department, University of British
Columbia (UBC)
Vancouver, Canada

ABSTRACT

Federated learning (FL) is an emerging machine learning paradigm in which a central server coordinates multiple participants (clients) collaboratively to train on decentralized data. In practice, FL often faces statistical, system, and model heterogeneities, which inspires the field of Model-Heterogeneous Personalized Federated Learning (MHPFL). With the increased interest in adopting large language models (LLMs) in FL, the existing MHPFL methods cannot achieve acceptable computational and communication costs, while maintaining satisfactory model performance. To bridge this gap, we propose a novel and efficient model-heterogeneous personalized Federated learning framework based on LoRA tuning (FedLoRA). Inspired by the popular LoRA method for fine-tuning pre-trained LLMs with a low-rank model (*a.k.a.*, an adapter), we design a *homogeneous small adapter* to facilitate federated client’s heterogeneous local model training with our proposed *iterative training* for global-local knowledge exchange. The homogeneous small local adapters are aggregated on the FL server to generate a global adapter. We theoretically prove the convergence of FedLoRA. Extensive experiments on two benchmark datasets demonstrate that FedLoRA outperforms six state-of-the-art baselines, beating the best method by 1.35% in test accuracy, 11.81× computation overhead reduction and 7.41× communication cost saving.

1 INTRODUCTION

As data privacy laws such as GDPR [19] have been rolled out worldwide due to concerns about privacy leakage, the traditional machine learning paradigm relying on collecting data for model training faces increasing challenges. Federated learning (FL) [28] has emerged as a collaborative learning paradigm in response to such a trend. In a typical FL system, a central FL server broadcasts a global model to clients, who then train it on local data and upload the resulting model back to the server. The server aggregates the received local models to update the global model. These steps repeat until the global model converges. Only models are transmitted between the server and clients without exposing private local data.

*Corresponding authors.

The above design requires that all clients have to train models with the same structures (homogeneous), which makes the traditional FL paradigm unsuitable when facing various types of heterogeneity [42]. **Statistical (Data) Heterogeneity.** FL clients’ local data often follow non-independent and identical distributions (non-IID). A local model solely trained by the client might perform better than the global FL model trained on non-IID data. **Resource Heterogeneity.** Clients participating in FL can be mobile edge devices [18] with different hardware resources (*e.g.* computation power and bandwidth). Traditional FL requires all (resource heterogeneous) clients to train models with the same structures, leading to model performance bottleneck as low-resource clients can only support smaller models. **Model Heterogeneity.** When FL participants are enterprises, they often maintain private model repositories with heterogeneous models. Fine-tuning them during FL training not only saves training time but also protects intellectual property [41].

These challenges motivate the research field of Model Heterogeneous Personalized Federated Learning (MHPFL). Existing MHPFL methods can be divided into three categories: 1) *Knowledge distillation-based MHPFL* methods [21, 24] often rely on a public dataset with the same distribution as local data, but such suitable public datasets may not always be available. Other knowledge distillation-based MHPFL methods [17, 36] without requiring public datasets often incur high computation and/or communication costs for FL clients due to local distillation. 2) *Model mixup-based MHPFL* methods [8, 23] split each local model into a heterogeneous part for local training and a homogeneous part for model aggregation. Only parts of the entire model are aggregated by the server results heterogeneous local models often suffer from subpar performance. 3) *Mutual learning-based MHPFL* methods [35, 40] assign a large heterogeneous model and a small homogeneous model for each client. The two models are trained locally via mutual learning, and only the small homogeneous model is uploaded to the server for aggregation. Training two models locally incurs extra computation costs. Moreover, the undefined choice of the model structures also affects the resulting performance.

Low-Rank Adaptation (LoRA) [12] has recently emerged as a popular method for fine-tuning pre-trained large language models

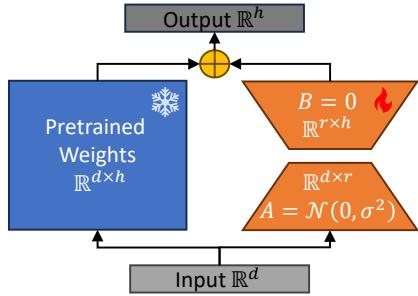


Figure 1: The working principle of LoRA.

(LLMs) to fit downstream tasks. As shown in Figure 1, it adds a branch alongside the pre-trained model, which is a low-rank adapter with the same input sample and the same output dimension as the pre-trained model. During fine-tuning, it freezes the pre-trained large model and trains the low-rank adapter. In each iteration of training, one sample is input into the frozen pre-trained model and the training adapter simultaneously, and the outputs of this sample from two models are summed as the final output. Then the hard loss between the final output and the label is calculated for updating the adapter by gradient descent. After that, the combination of the two branches is used for model inference, which may perform a similar or higher accuracy than fine-tuning the whole pre-trained model directly. As only the small adapter is trained during fine-tuning, LoRA achieves efficient computation and storage.

Inspired by LoRA, we propose an efficient model-heterogeneous personalized Federated learning framework based on LoRA tuning (FedLoRA) for supervised learning tasks. Belonging to the mutual learning category, each client holds heterogeneous data and model. FedLoRA enables a *small low-rank homogeneous adapter* to be incorporated into the large heterogeneous local model. In each communication round, 1) clients first replace their local adapters with the global homogeneous adapter received from the FL server; 2) then, they perform the proposed *iterative learning* method to train the two models alternatively for global-local knowledge transfer; and 3) finally, the updated local homogeneous adapters from the clients are aggregated by the FL server. In short, adapters are regarded as “knowledge carriers” for aggregation to support knowledge transfer among clients. Each client only additionally trains a small adapter and only communicate small homogeneous adapters with the server. Such a design ensures that FedLoRA achieves MHPFL efficiently.

As the insertion of LoRA adapters changes the process of local model training compared with traditional FL, we derive the non-convex convergence rate of FedLoRA based on local iterative training and prove that it converges over time. Extensive experiments on two benchmark datasets demonstrate significant advantages of FedLoRA in both model-homogeneous and model-heterogeneous scenarios compared to six state-of-the-art methods, beating the best of them by 1.35% in test accuracy, 11.81× computation overhead reduction and 7.41× communication cost saving.

2 RELATED WORK

Existing MHPFL methods have two branches: a) *Partially model-heterogeneous*, clients hold different subnets of the global model,

and heterogeneous subnets can be aggregated on the server, such as FedRolex [3], HeteroFL [9], FjORD [11], HFL [26], Fed2 [43], FedResCuE [49]. b) *Completely model-heterogeneous*, clients hold models with completely different model structures that can not be aggregated directly on the server. This branch can be further divided into the following categories.

Knowledge Distillation-based MHPFL. In the *public dataset-dependent* knowledge distillation-based MHPFL methods (such as Cronus [4], FedGEMS [6], Fed-ET [7], FSFL [13], FCCL [14], DS-FL [15], FedMD [21], FedKT [22], FedDF [24], FedHeNN [27], FedAUX [32], CFD [33], FedKEMF [44], KT-pFL [45]), the server aggregates the output logits of different clients’ heterogeneous models on a public dataset to construct global logits. But the public dataset is not always accessible and the algorithm performs well only if the public dataset has the same distribution as private data. Besides, transmitting logits of *each* public data sample incurs high communication costs for a large-scale public dataset. For other knowledge distillation-based MHPFL methods not dependent on a public dataset, FedZKT [46] and FedGen [48] introduce zero-shot knowledge distillation to FL, they generate a public dataset through training a generator, which is time-consuming. HFD [1, 2], FedGKT [10], FD [17], FedProto [36] allow each client to upload the local (average) logits or representations of its seen-class samples to the server for aggregation by class to generate the global class-logits or representations which are sent back to clients and used to calculate the distillation loss with local logits for each local data sample, incurring high computational overheads.

Model Mixup-based MHPFL. These methods split each client’s local model into two parts: one feature extractor and one classifier, and only one part is shared. FedMatch [5], FedRep [8], FedBABU [29] and FedAlt/FedSim [30] share homogeneous feature extractors to enhance model generalization and personalize local classifier. In contrast, FedClassAvg [16], LG-FedAvg [23] and CHFL [25] share homogeneous classifier to improve model classification and personalize local feature extractor. Since only partial parameters of the whole model are shared, the final local heterogeneous models face performance bottlenecks.

Mutual Learning-based MHPFL. FML [35] and FedKD [40] assign a small homogeneous model and a large heterogeneous model in each client, and train them in a *mutual learning* manner. The small homogeneous models after local training are aggregated on the server. In short, the small homogeneous models as information mediums implement the knowledge transfer across large heterogeneous models. However, they do not explore the relationship between the model structure and parameter capacity between the two models, which may affect the final model performance and computation costs of training an extra small homogeneous model for each client.

Our Insight. FedLoRA enables knowledge transfer across clients’ local large heterogeneous models through a *small low-rank homogeneous adapter*, which is a low-rank version of the fully connected layers of the large local heterogeneous models. It does not rely on any public dataset, and incurs low computation and communication costs as only small low-rank extra adapters are trained on clients and transmitted between the FL server and the clients.

3 PRELIMINARIES

3.1 LoRA Adapter

As shown in Figure. 1, a LoRA adapter has the same input and output dimensions as the pre-trained large model. As elaborated in [12], the structures of the current LoRA support linear, embedding and convolutional layers. Take a linear LoRA as an example. Given a linear layer of the pre-trained model with \mathbb{R}^d input and \mathbb{R}^h output (*i.e.*, with a $\mathbb{R}^{d \times h}$ parameter matrix), a linear LoRA adapter can be a combination of two small matrices $A(\mathbb{R}^{d \times r})$ and $B(\mathbb{R}^{r \times h})$ by *matrix decomposition*, where the rank r is far smaller than d and h . Before training, matrix A can be initialized with a Gaussian distribution $\mathcal{N}(0, \sigma^2)$ (0 : mean, σ^2 : variance), and matrix B can be initialized with 0 .

3.2 Overview of Federated Learning

FedAvg [28] is a typical FL algorithm, it assumes that a FL system consists of one central server and N clients. In each communication round, the server randomly selects a fraction C of clients S ($|S| = \lfloor CN \rfloor = K$) and broadcasts the global model $\mathcal{F}(\omega)$ ($\mathcal{F}(\cdot)$ is model structure, ω are model parameters) to the selected K clients. Client k trains the received global model $\mathcal{F}(\omega)$ on its local data D_k ($D_k \sim P_k$, local data D_k obeys distribution P_k , *i.e.*, local data from different clients are non-IID) to obtain updated local model $\mathcal{F}(\omega_k)$ by gradient descent, *i.e.*, $\omega_k \leftarrow \omega - \eta \nabla \ell(\mathcal{F}(x_i; \omega), y_i)$. $\ell(\mathcal{F}(x_i; \omega), y_i)$ is the loss of the global model $\mathcal{F}(\omega)$ on the sample $(x_i, y_i) \in D_k$. The updated local model $\mathcal{F}(\omega_k)$ is uploaded to the server. The server aggregates the received local models from the selected K clients by weighted averaging to update the global model, *i.e.*, $\omega = \sum_{k=0}^{K-1} \frac{n_k}{n} \omega_k$ ($n_k = |D_k|$ is data volume of client k , $n = \sum_{k=0}^{N-1} n_k$ is data volume of all clients).

In short, the typical FL algorithm requires all clients to train local models with the same structures (**homogeneous**), and its training objective is to minimize the average loss of the global model $\mathcal{F}(\omega)$ on all client data, *i.e.*,

$$\min_{\omega \in \mathbb{R}^d} \sum_{k=0}^{K-1} \frac{n_k}{n} \mathcal{L}_k(D_k; \mathcal{F}(\omega)), \quad (1)$$

where the parameters ω of the global model are d -dimensional real numbers, $\mathcal{L}_k(D_k; \mathcal{F}(\omega))$ is the average loss of the global model $\mathcal{F}(\omega)$ on client k 's local data D_k .

3.3 Problem Definition

The goal of this paper is to study model-heterogeneous personalized FL in supervised image classification tasks. We assume that all clients execute the same image classification task, and different clients may hold **heterogeneous** local models $\mathcal{F}_k(\omega_k)$ ($\mathcal{F}_k(\cdot)$ denotes different model structures, ω_k indicates personalized model parameters).

To support generalized knowledge exchanging in FL training involving heterogeneous local models, we insert a small low-rank homogeneous adapter $\mathcal{A}(\theta_k)$ ($\mathcal{A}(\cdot)$ is adapter structure, θ_k are personalized local adapter parameters) into a large local heterogeneous model $\mathcal{F}_k(\omega_k)$. Clients share the small low-rank homogeneous adapters to implement the knowledge transfer across heterogeneous models from different clients. As shown in Figure 2, the

model consisting of the small low-rank homogeneous adapter and the large heterogeneous model is denoted as $\mathcal{F}_k(\omega_k) + \mathcal{A}(\theta_k)$. The objective of FedLoRA is to minimize the sum of the loss of all clients' heterogeneous models, *i.e.*,

$$\min_{\omega_0, \dots, \omega_{K-1} \in \mathbb{R}^{d_0, \dots, d_{K-1}}} \sum_{k=0}^{K-1} \mathcal{L}_k(D_k; \mathcal{F}_k(\omega_k) + \mathcal{A}(\theta_k)), \quad (2)$$

where the parameters $\omega_0, \dots, \omega_{K-1}$ of local heterogeneous models are d_0, \dots, d_{K-1} -dimensional real numbers.

4 THE PROPOSED FEDLORA APPROACH

To reduce the computational overhead incurred by training clients' low-rank adapters, unlike LoRA which matches an adapter for the entire pre-trained model, we view each client's personalized heterogeneous model as two parts: 1) the convolutional layers $f_k(\omega_{k,conv})$, and 2) the fully-connected layers $h_k(\omega_{k,fc})$, *i.e.*, $\mathcal{F}_k(\omega_k) = f_k(\omega_{k,conv}) \circ h_k(\omega_{k,fc})$, and we only insert a low-rank adapter $\mathcal{A}(\theta_k)$ for each client's fully-connected layers $h_k(\omega_{k,fc})$, as shown in Figure 2. The workflow of FedLoRA is as follows:

- In the t -th communication round, the server broadcasts the global low-rank adapter $\mathcal{A}(\theta^{t-1})$ to randomly selected K clients. Client k replaces its local adapter $\mathcal{A}(\theta_k^{t-1})$ with the received global adapter $\mathcal{A}(\theta^{t-1})$.
- During local training, if we train the heterogeneous local model and the homogeneous adapter synchronously like LoRA, *i.e.*, summing the output of two models for loss calculation, the immature global adapter in the beginning communication rounds of FL may lead to poor model performances. To boost model accuracy, we devise a novel **iterative learning** manner to train the two models for global-local knowledge transfer.
- After local iterative learning, the updated heterogeneous local models are stored in clients and the updated homogeneous local adapters are uploaded to the server for aggregation like FedAvg to update the global adapter $\mathcal{A}(\theta^t)$, which

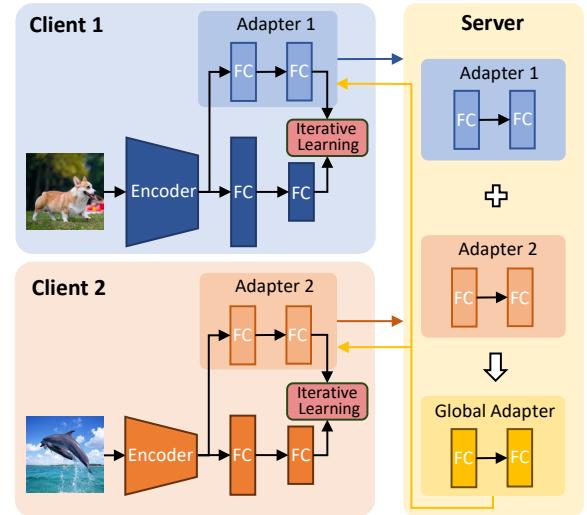


Figure 2: Workflow of FedLoRA.

fuses the knowledge from different clients' heterogeneous local models.

The above steps repeat until all personalized heterogeneous local models converge, which will be used for **inference** after federated training. More detailed description of FedLoRA is given in Algorithm 1 (Appendix A).

4.1 Iterative Learning

Treating the local heterogeneous model and the homogeneous low-rank adapter as parts of a whole local model and training them *simultaneously* is intuitive. However, training such a larger model might slow convergence and even lead to model performance degradation if local data are limited. To boost the performance of personalized heterogeneous local models, we propose an *iterative learning* method to train the heterogeneous local models and the homogeneous low-rank adapters. As illustrated in Figure 3, firstly, we freeze the global adapter received by clients and train heterogeneous local models, which transfer global knowledge to clients. Then, we freeze the updated heterogeneous local models and train homogeneous low-rank adapters which are uploaded to the server for aggregation, which transfers local knowledge to the FL server.

Freeze Adapter, Train Local Model. As Step ① shown in Figure 3, client k inputs the sample $(x, y) \in D_k$ into the encoder (convolutional layers $f_k(\omega_{k,conv}^{t-1})$) of the local heterogeneous model to obtain representation $\mathcal{R} = f_k(x; \omega_{k,conv}^{t-1})$. Then, the representation \mathcal{R} is fed into the fully-connected layers $h_k(\omega_{k,fc}^{t-1})$ of the heterogeneous local model and the low-rank adapter $\mathcal{A}(\theta^{t-1})$ to obtain

$$\hat{y}_1 = \mathcal{A}(\mathcal{R}; \theta^{t-1}), \hat{y}_2 = h_k(\mathcal{R}; \omega_{k,fc}^{t-1}). \quad (3)$$

Then, the hard loss (such as cross-entropy loss [47]) between the output prediction \hat{y}_1 of the homogeneous adapter and label y , and the hard loss between the output prediction \hat{y}_2 of the heterogeneous local model and label y can be calculated, respectively, i.e.,

$$\ell_1 = \ell(\hat{y}_1, y), \ell_2 = \ell(\hat{y}_2, y). \quad (4)$$

In the beginning communication rounds, the immature global adapter may have a negative influence on the performances of heterogeneous local models. To balance the global knowledge carried by the global adapter and the personalized local knowledge incorporated in the fully connected layers of local heterogeneous models, we take the linearly weighted sum of the hard losses from the two branches as the complete loss on the input sample, i.e.,

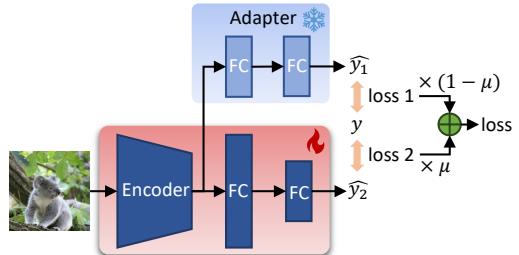
$$\ell_\omega = (1 - \mu) \cdot \ell_1 + \mu \cdot \ell_2, \mu \in [0.5, 1]. \quad (5)$$

Then, we use the complete loss to update the heterogeneous local models by gradient descent (e.g. SGD [31]),

$$\omega_k^t \leftarrow \omega_k^{t-1} - \eta_\omega \nabla \ell_\omega, \quad (6)$$

where η_ω is the learning rate of the heterogeneous local model. During this training process, the global knowledge carried by the frozen global adapter is transferred to heterogeneous local models, which promotes the generalization improvements of heterogeneous local models. Meanwhile, the personalized local knowledge involved in local data is learned by heterogeneous local models further, which facilitates the personalization of heterogeneous local models.

1 Freeze Adapter, Train Model



2 Freeze Model, Train Adapter

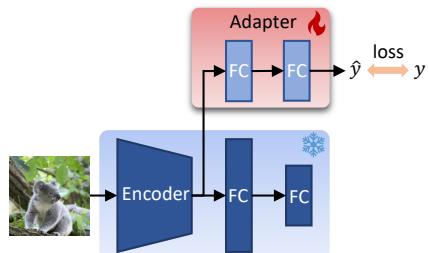


Figure 3: Iterative learning in FedLoRA.

Freeze Local Model, Train Adapter. As Step ② shown in Figure 3, client k inputs the sample $(x, y) \in D_k$ into the encoder (convolutional layers $f_k(\omega_{k,conv}^t)$) of the *updated* local heterogeneous model to obtain representation $\tilde{\mathcal{R}} = f_k(x; \omega_{k,conv}^t)$, then the representation $\tilde{\mathcal{R}}$ is input into the adapter $\mathcal{A}(\theta^{t-1})$ to obtain

$$\hat{y} = \mathcal{A}(\tilde{\mathcal{R}}; \theta^{t-1}). \quad (7)$$

The hard loss between the adapter prediction \hat{y} and y is:

$$\ell_\theta = \ell(\hat{y}, y). \quad (8)$$

The adapter parameters are updated via gradient descent:

$$\theta_k^t \leftarrow \theta^{t-1} - \eta_\theta \nabla \ell_\theta, \quad (9)$$

where η_θ is the adapter learning rate. During this process, personalized local knowledge is transferred to the updated adapter which is then uploaded to the server for aggregation.

4.2 Homogeneous Adapter Aggregation

After receiving the local homogeneous adapters, the server aggregates them like FedAvg to update the global adapter,

$$\theta^t = \sum_{k=0}^{K-1} \frac{n_k}{n} \theta_k^t. \quad (10)$$

The updated global adapter combines local knowledge across heterogeneous local models from different clients. It is then broadcast to participating clients in the next round.

4.3 Adapter Structure

To reduce extra computational overheads by training adapters, we only match adapters for the *fully connected layers* of local heterogeneous models. A low-rank adapter is an inherently “dimension-reduced” version of a local heterogeneous model, i.e., it contains

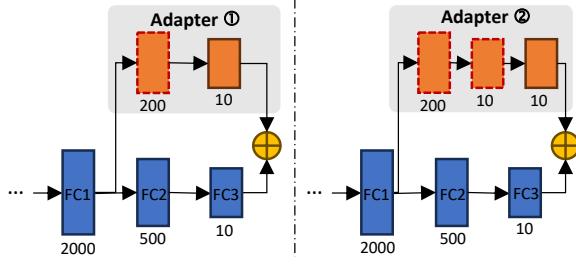


Figure 4: Two types of low-rank adapters.

far fewer parameters than the local heterogeneous model. We design two choices for constructing low-rank adapters different from typical LoRA adapters introduced in Section 3.1.

Direct Dimension Reduction. As adapter ① shown in Figure 4, we match a low-rank adapter for the last two fully connected layers of the local heterogeneous model. It consists of two linear layers: the first layer (marked with a red dashed box) is the *direct dimension-reduced* version of the FC2 in the local heterogeneous model (dimension: $500 \rightarrow 200$), and the second layer has the same dimension as the output layer FC3 in the local heterogeneous model.

Matrix Decomposition. As adapter ② shown in Figure 4, the dimension of the parameter matrix between FC1 and FC2 in the local heterogeneous model is 2000×500 . We can utilize *matrix decomposition* to transform it into two small parameter matrices: $(2000 \times 200) + (200 \times 10)$ (in a 10-class image classification task). Compared with the first adapter choice, this method reduces parameter volume while increasing network depth, which benefits from improving network learning ability. Since we only need to guarantee that the small linear LoRA adapter is a low-rank version of the fully connected layer of the large heterogeneous model, we can either manually specify the dimensions of the two decomposed matrices or leverage typical matrix decomposition approaches (e.g. SVD) in typical LoRA adapters.

4.4 Discussion

In this section, we discuss the computational overheads, communication costs and privacy protection of FedLoRA.

Computational Overhead. On top of training a local heterogeneous model, each client also trains an extra small low-rank homogeneous adapter which contains far fewer parameters than the fully connected layers of the local heterogeneous model. Thus, the extra computational overhead by training it is acceptable.

Communication Cost. Each client and the FL server only exchange a small low-rank homogeneous adapter, which incurs much lower communication costs than sending a complete local model (like in FedAvg).

Privacy Protection. Only the parameters of small low-rank homogeneous adapters are exchanged between the server and clients. Local data are always stored in clients. Hence, no private data is exposed during FedLoRA training.

5 ANALYSIS

Following Tan et al. [36], Yi et al. [42], we first declare some additional notations. We denote t as the communication round and $e \in \{0, 1, \dots, E\}$ as the iteration of local training. In each round,

each client executes E iterations during local training. $tE + e$ is the e -th iteration in the $(t+1)$ -th round; $tE + 0$ denotes that in the $(t+1)$ -th round, before local model training, clients receive the global adapter $\mathcal{A}(\theta^t)$ aggregated in the t -th round; $tE + E$ is the last iteration of local training, indicating the end of local training in the $(t+1)$ -th round. We also assume that the local heterogeneous model and local adapter have the same learning rate $\eta = \eta_\omega = \eta_{\theta}$.

ASSUMPTION 5.1. Lipschitz Smoothness. The gradients of client k 's local heterogeneous model are L_1 -Lipschitz smooth [36, 42], i.e.,

$$\|\nabla \mathcal{L}_k^{t_1}(\omega_k^{t_1}; \mathbf{x}, \mathbf{y}) - \nabla \mathcal{L}_k^{t_2}(\omega_k^{t_2}; \mathbf{x}, \mathbf{y})\| \leq L_1 \|\omega_k^{t_1} - \omega_k^{t_2}\|, \quad (11)$$

$$\forall t_1, t_2 > 0, k \in \{0, 1, \dots, N-1\}, (\mathbf{x}, \mathbf{y}) \in D_k.$$

The above formulation can be expressed as:

$$\mathcal{L}_k^{t_1} - \mathcal{L}_k^{t_2} \leq \langle \nabla \mathcal{L}_k^{t_2}(\omega_k^{t_1} - \omega_k^{t_2}) \rangle + \frac{L_1}{2} \|\omega_k^{t_1} - \omega_k^{t_2}\|_2^2. \quad (12)$$

ASSUMPTION 5.2. Unbiased Gradient and Bounded Variance. The random gradient $g_{\omega, k}^t = \nabla \mathcal{L}_k^t(\omega_k^t; \mathcal{B}_k^t)$ (\mathcal{B} is a batch of local data) of each client's local heterogeneous model is unbiased, and the random gradient $g_{\theta, k}^t = \nabla \mathcal{L}_k^t(\theta_k^t; \mathcal{B}_k^t)$ of each client's local adapter is also unbiased,

$$\mathbb{E}_{\mathcal{B}_k^t \subseteq D_k} [g_{\omega, k}^t] = \nabla \mathcal{L}_k^t(\omega_k^t), \quad (13)$$

$$\mathbb{E}_{\mathcal{B}_k^t \subseteq D_k} [g_{\theta, k}^t] = \nabla \mathcal{L}_k^t(\theta_k^t),$$

and the variance of $g_{\omega, k}^t$ and $g_{\theta, k}^t$ are bounded by:

$$\mathbb{E}_{\mathcal{B}_k^t \subseteq D_k} [\|\nabla \mathcal{L}_k^t(\omega_k^t; \mathcal{B}_k^t) - \nabla \mathcal{L}_k^t(\omega_k^t)\|_2^2] \leq \sigma^2, \quad (14)$$

$$\mathbb{E}_{\mathcal{B}_k^t \subseteq D_k} [\|\nabla \mathcal{L}_k^t(\theta_k^t; \mathcal{B}_k^t) - \nabla \mathcal{L}_k^t(\theta_k^t)\|_2^2] \leq \delta^2.$$

With these assumptions, we derive the following lemma and theorem. Their proofs can be found in Appendices B and C.

LEMMA 5.3. Based on Assumptions 5.1 and 5.2, during $\{0, 1, \dots, E\}$ local iterations of the $(t+1)$ -th FL training round, the loss of an arbitrary client's local heterogeneous model is bounded by:

$$\mathbb{E}[\mathcal{L}_{(t+1)E}] \leq \mathcal{L}_{tE+0} + (L_1 \eta^2 \mu^2 - \eta \mu) \sum_{e=0}^{E-1} \|\nabla \mathcal{L}_{tE+e}\|_2^2 + \frac{L_1 \eta^2 (\sigma^2 + \delta^2)}{2}. \quad (15)$$

THEOREM 5.4. Non-convex convergence rate of pFedLoRA. Based on the above assumptions and lemma, for an arbitrary client and any $\epsilon > 0$, the following inequality holds:

$$\frac{1}{T} \sum_{t=0}^{T-1} \sum_{e=0}^{E-1} \|\nabla \mathcal{L}_{tE+e}\|_2^2 \leq \frac{\frac{1}{T} \sum_{t=0}^{T-1} (\mathcal{L}_{tE+0} - \mathbb{E}[\mathcal{L}_{(t+1)E}])}{\eta \mu - L_1 \eta^2 \mu^2} + \frac{\frac{L_1 \eta^2 (\sigma^2 + \delta^2)}{2}}{\eta \mu - L_1 \eta^2 \mu^2} < \epsilon, \quad (16)$$

$$\text{s.t. } \eta < \frac{2\mu}{L_1(\sigma^2 + \delta^2 + 2\mu^2\epsilon)}.$$

Therefore, in FedLoRA, an arbitrary client's local heterogeneous model converges at a non-convex rate of $\epsilon \sim \mathcal{O}(\frac{1}{T})$.

6 EXPERIMENTAL EVALUATION

In this section, we compare FedLoRA against six state-of-the-art MHPFL approaches on two real-world datasets under various experiment conditions. The experiments were conducted with Pytorch on four NVIDIA GeForce RTX 3090 GPUs with 24G memory.

6.1 Experiment Setup

Datasets. We evaluate FedLoRA and baselines on two common image classification datasets: CIFAR-10 and CIFAR-100¹ [20]. They are manually divided into non-IID datasets following the method specified in Shamsian et al. [34]. For CIFAR-10, we assign only data from 2 out of the 10 classes to each client (non-IID: 2/10). For CIFAR-100, we assign only data from 10 out of the 100 classes to each client (non-IID: 10/100). Then, each client’s local data are divided into the training set, the evaluation set, and the testing set following the ratio of 8:1:1. The testing set is stored locally by each client, which follows the same distribution as the local training set.

Models. As shown in Table 3 (Appendix D), each client trains CNN models on two datasets. In model-homogeneous settings, each client has the same CNN-1 and the same adapter with two fully connected layers ($x \rightarrow Conv1 \rightarrow Conv2 \rightarrow FC1 \rightarrow [direct\ dimension-reduced\ FC2\ with\ hidden_dim = \{100, 200, 300, 400, 500\} \rightarrow FC3]$, $[\cdot]$ is the homogeneous adapter). In model-heterogeneous settings, different clients are evenly deployed with {CNN-1, ..., CNN-5} (model id is determined by client id $k\%5$) and the homogeneous adapter containing two fully connected layers ($x \rightarrow Conv1 \rightarrow Conv2 \rightarrow FC1 \rightarrow FC2 \rightarrow [matrix-decomposed\ FC2\ with\ hidden_dim = \{20, 40, 60, 80\} \rightarrow FC3]$, $[\cdot]$ is the homogeneous adapter).

Baselines. We compare FedLoRA with 6 advanced baselines from three categories of MHPFL shown in Section 2: Standalone, clients train local models solely; **Public-data independent knowledge distillation-based MHPFL:** FD [17] and FedProto [36]; **Mutual learning-based MHPFL:** FML [35] and FedKD [40]; **Model mixup-based MHPFL:** LG-FedAvg [23].

Evaluation Metrics. 1) Accuracy: we measure the *individual test accuracy (%)* of each client’s local heterogeneous model and calculate the *average test accuracy* of all clients’ local models. **2) Communication Cost:** We trace the number of transmitted parameters when the average model accuracy reaches the target accuracy.

3) Computation Cost: We track the consumed computation FLOPs when the average model accuracy reaches the target accuracy.

Training Strategy. We tune the optimal FL settings for all methods via grid search. The epochs of local training $E \in \{1, 10\}$ and the batch size of local training $B \in \{64, 128, 256, 512\}$. The optimizer for local training is SGD with learning rate $\eta = \eta_\omega = \eta_\theta = 0.01$. We also tune special hyperparameters for the baselines and report the optimal results. We also adjust the hyperparameters μ and $hidden_dim$ to achieve the best-performance FedLoRA. To compare FedLoRA with the baselines fairly, we set the total number of communication rounds $T \in \{100, 500\}$ to ensure that all algorithms converge.

6.2 Comparison Results

We compare FedLoRA with baselines under *model-homogeneous* (a special situation in model-heterogeneous scenarios) and *model-heterogeneous* settings with varied numbers of clients N and client participation fraction C . We set up three scenarios: $\{(N = 10, C = 100\%), (N = 50, C = 20\%), (N = 100, C = 10\%) \}$. For ease of comparison across the three settings, $N \times C$ is set to be the same (10 clients participate in each round of FL). For FML and FedKD under model-heterogeneous settings, we regard the smallest ‘CNN-5’ model as the small homogeneous model.

¹<https://www.cs.toronto.edu/%7Ekriz/cifar.html>

Table 1: Average accuracy for model-homogeneous FL. N is the number of clients. C is the fraction of participating clients in each round. ‘-’ denotes failure to converge.

Method	N=10, C=100%		N=50, C=20%		N=100, C=10%	
	CIFAR-10	CIFAR-100	CIFAR-10	CIFAR-100	CIFAR-10	CIFAR-100
Standalone	96.35	74.32	95.25	62.38	92.58	54.93
FML [35]	94.83	70.02	93.18	57.56	87.93	46.20
FedKD [40]	94.77	70.04	92.93	57.56	90.23	50.99
LG-FedAvg [23]	96.47	73.43	94.20	61.77	90.25	46.64
FD [17]	96.30	-	-	-	-	-
FedProto [36]	95.83	72.79	95.10	62.55	91.19	54.01
pFedLoRA	96.69	75.58	95.55	62.55	92.80	55.82

Table 2: Average accuracy for model-heterogeneous FL.

Method	N=10, C=100%		N=50, C=20%		N=100, C=10%	
	CIFAR-10	CIFAR-100	CIFAR-10	CIFAR-100	CIFAR-10	CIFAR-100
Standalone	96.53	72.53	95.14	62.71	91.97	53.04
FML [35]	30.48	16.84	-	21.96	-	15.21
FedKD [40]	80.20	53.23	77.37	44.27	73.21	37.21
LG-FedAvg [23]	96.30	72.20	94.83	60.95	91.27	45.83
FD [17]	96.21	-	-	-	-	-
FedProto [36]	96.51	72.59	95.48	62.69	92.49	53.67
pFedLoRA	96.66	73.58	95.74	64.06	92.58	53.95

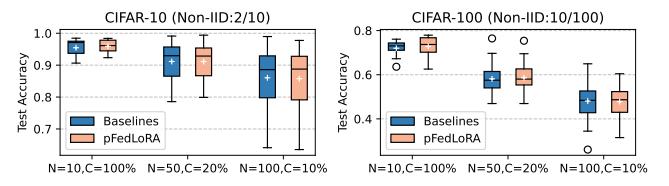


Figure 5: Accuracy distribution for individual clients.

Average Accuracy. The results in Tables 1 and 2 show that the average accuracy of all personalized heterogeneous local models in FedLoRA surpasses other baselines in both model-homogeneous and model-heterogeneous settings, and shows up to 1.26%, 1.35% accuracy improvements in model-homogeneous and model-heterogeneous settings, respectively. Figure 10 (Appendix D) shows that the average test accuracy of FedLoRA and the baselines under each $\{N, C\}$ setting specified in Table 2 varies with communication rounds. FedLoRA converges to the highest average accuracy with a lower convergence speed since an extra local adapter is required to be trained.

Individual Accuracy. We utilize *box plots* to display the distribution of individual model accuracy in model-heterogeneous settings. As shown in Figure 5, ‘+’ denotes the average accuracy of all clients for each algorithm. A small box length bounded by the upper quartile and the lower quartile indicates a more concentrated accuracy distribution across all clients with small variance. We observe that FedLoRA obtains the higher average accuracy and the lower variance than the optimal baselines (Standalone or FedProto in Table 2) at most settings.

Trade-off among Accuracy, Computation, Communication. We compare FedLoRA and the state-of-the-art baseline FedProto in model accuracy, computational overheads and communication costs. Figure 6 shows that FedLoRA always maintains the higher model accuracy and far lower computation costs than FedProto while keeping similar communication costs, indicating that FedLoRA takes the best trade-off between model accuracy, computational and communication costs. FedLoRA obtains up to 11.81× computational overhead reduction and up to 7.41× communication cost saving.

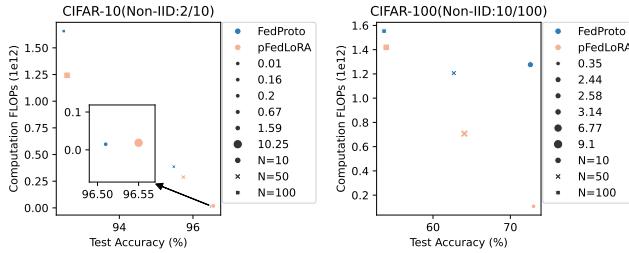


Figure 6: Trade-off among test accuracy, computational overhead and communication cost. The sizes of markers reflect the number of communicated parameters (1e6).

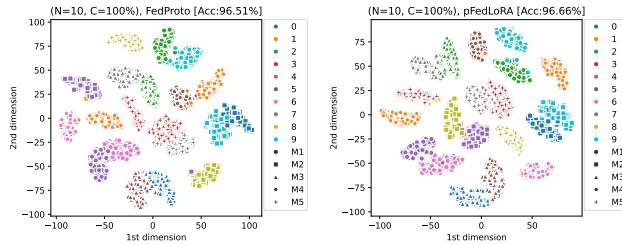


Figure 7: Representation visualization for FedProto and FedLoRA on CIFAR-10 (Non-IID: 2/10).

Visualized Personalization Analysis. In model-heterogeneous settings, we extract every sample representation from each FL client under FedLoRA and FedProto, respectively. Then, we leverage the T-SNE [37] tool to reduce the dimensionality of the extracted representations from 500 to 2, and visualize the results. Since CIFAR-100 includes 100 classes of samples, we focus on visualizing the results on CIFAR-10 (non-IID: 2/10) in Figure 7. It can be observed that most clusters in FedLoRA and FedProto consist of representations from a client’s two seen classes of samples, which indicates that each client’s local heterogeneous model has strong personalization capability. The two seen class representations within most clusters under FedLoRA and FedProto satisfy “intra-class compactness and inter-class separation”, reflecting that every client can classify its seen classes well under both algorithms. Generally, FedLoRA performs better classification boundaries than FedProto.

6.3 Case Studies

6.3.1 Robustness to Non-IIDness. We evaluate the robustness of FedLoRA and FedProto to non-IIDness with ($N = 100, C = 10\%$). We vary the number of classes seen by each client as $\{2, 4, 6, 8, 10\}$ on CIFAR-10 and $\{10, 30, 50, 70, 90, 100\}$ on CIFAR-100. Figure 8 presents that FedLoRA consistently outperforms FedProto, demonstrating its robustness to non-IIDness. As the non-IIDness decreases (the number of classes seen by each client rises), accuracy degrades since more IID local data enhances generalization and reduces personalization.

6.3.2 Robustness to Client Participant Rates. We also test the robustness of FedLoRA and FedProto to client participant rates C

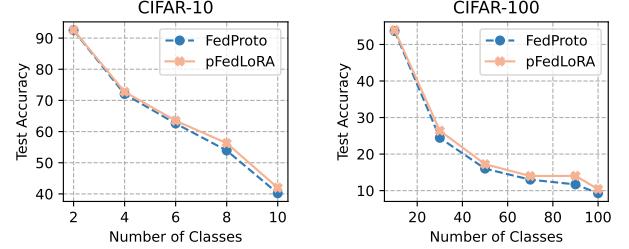


Figure 8: Robustness to Non-IIDness.

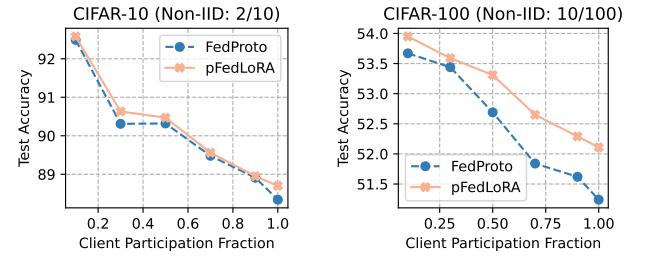


Figure 9: Robustness to client participation rates.

under ($N = 100, C = 10\%$) on CIFAR-10 (non-IID: 2/10) and CIFAR-100 (non-IID: 10/100). We vary the client participant rates as $C = \{0.1, 0.3, 0.5, 0.7, 0.9, 1\}$. Figure 9 shows that FedLoRA consistently outperforms FedProto, especially on the more complicated CIFAR-100 dataset, verifying its robustness to changes in client participant rates. Besides, as the client participant rates rise, model accuracy drops as more participating clients provide more IID local data, which also improves generalization and reduces personalization.

7 CONCLUSIONS AND FUTURE WORK

In this paper, we propose a novel computation- and communication-efficient model-heterogeneous personalized FL framework, FedLoRA, which is inspired by LoRA tuning. It assigns a homogeneous small low-rank linear adapter for each client’s local personalized heterogeneous local model. The proposed iterative learning method for training the local heterogeneous model and homogeneous adapter supports the bidirectional transfer of global knowledge and local knowledge. Aggregating the homogeneous local adapters after local iterative training on the server enables the sharing of local knowledge among FL clients. Theoretical analysis proves that FedLoRA can converge at a non-convex rate of $\mathcal{O}(\frac{1}{T})$. Extensive experiments demonstrate its superiority in model accuracy, computational overheads, and communication costs.

In future work, we plan to explore two promising improvements for FedLoRA: a) optimizing the iterative learning process to improve model accuracy, and b) exploring lighter and more effective structures of homogeneous adapters.

REFERENCES

- [1] Jin-Hyun Ahn et al. 2019. Wireless Federated Distillation for Distributed Edge Learning with Heterogeneous Data. In *Proc. PIMRC*. IEEE, Istanbul, Turkey, 1–6.
- [2] Jin-Hyun Ahn et al. 2020. Cooperative Learning VIA Federated Distillation OVER Fading Channels. In *Proc. ICASSP*. IEEE, Barcelona, Spain, 8856–8860.

- [3] Samiul Alam et al. 2022. FedRolex: Model-Heterogeneous Federated Learning with Rolling Sub-Model Extraction. In *Proc. NeurIPS*, virtual.
- [4] Hongyan Chang et al. 2021. Cronus: Robust and Heterogeneous Collaborative Learning with Black-Box Knowledge Transfer. In *Proc. NeurIPS Workshop*, virtual.
- [5] Jiangu Chen et al. 2021. FedMatch: Federated Learning Over Heterogeneous Question Answering Data. In *Proc. CIKM*. ACM, virtual, 181–190.
- [6] Sijie Cheng et al. 2021. FedGEMS: Federated Learning of Larger Server Models via Selective Knowledge Fusion. *CoRR* abs/2110.11027 (2021).
- [7] Yae Jee Cho et al. 2022. Heterogeneous Ensemble Knowledge Transfer for Training Large Models in Federated Learning. In *Proc. IJCAI*. ijcai.org, virtual, 2881–2887.
- [8] Liam Collins et al. 2021. Exploiting Shared Representations for Personalized Federated Learning. In *Proc. ICML*, Vol. 139. PMLR, virtual, 2089–2099.
- [9] Enmao Diao. 2021. HeteroFL: Computation and Communication Efficient Federated Learning for Heterogeneous Clients. In *Proc. ICLR*. OpenReview.net, Virtual Event, Austria, 1.
- [10] Chaoyang He et al. 2020. Group Knowledge Transfer: Federated Learning of Large CNNs at the Edge. In *Proc. NeurIPS*, virtual.
- [11] S. Horváth. 2021. FjORD: Fair and Accurate Federated Learning under heterogeneous targets with Ordered Dropout. In *Proc. NIPS*. OpenReview.net, Virtual, 12876–12889.
- [12] Edward J. Hu et al. 2022. LoRA: Low-Rank Adaptation of Large Language Models. In *ICLR*. OpenReview.net, Virtual, 1.
- [13] Wenke Huang et al. 2022. Few-Shot Model Agnostic Federated Learning. In *Proc. MM*. ACM, Lisboa, Portugal, 7309–7316.
- [14] Wenke Huang et al. 2022. Learn from Others and Be Yourself in Heterogeneous Federated Learning. In *Proc. CVPR*. IEEE, virtual, 10133–10143.
- [15] Sohei Itahara et al. 2023. Distillation-Based Semi-Supervised Federated Learning for Communication-Efficient Collaborative Training With Non-IID Private Data. *IEEE Trans. Mob. Comput.* 22, 1 (2023), 191–205.
- [16] Jaehye Jang et al. 2022. FedClassAvg: Local Representation Learning for Personalized Federated Learning on Heterogeneous Neural Networks. In *Proc. ICPP*. ACM, virtual, 76:1–76:10.
- [17] Eunjeong Jeong et al. 2018. Communication-Efficient On-Device Machine Learning: Federated Distillation and Augmentation under Non-IID Private Data. In *Proc. NeurIPS Workshop on Machine Learning on the Phone and other Consumer Devices*, virtual.
- [18] Yuang Jiang et al. 2022. Model Pruning Enables Efficient Federated Learning on Edge Devices. *TNNLS* 1, 1 (2022), 1.
- [19] Peter Kairouz et al. 2021. Advances and Open Problems in Federated Learning. *Foundations and Trends in Machine Learning* 14, 1–2 (2021), 1–210.
- [20] Alex Krizhevsky et al. 2009. *Learning multiple layers of features from tiny images*. Toronto, ON, Canada, .
- [21] Daliang Li and Junpu Wang. 2019. FedMD: Heterogenous Federated Learning via Model Distillation. In *Proc. NeurIPS Workshop*, virtual.
- [22] Qinbin Li et al. 2021. Practical One-Shot Federated Learning for Cross-Silo Setting. In *Proc. IJCAI*. ijcai.org, virtual, 1484–1490.
- [23] Paul Pu Liang et al. 2020. Think locally, act globally: Federated learning with local and global representations. *arXiv preprint arXiv:2001.01523* 1, 1 (2020).
- [24] Tao Lin et al. 2020. Ensemble Distillation for Robust Model Fusion in Federated Learning. In *Proc. NeurIPS*, virtual.
- [25] Chang Liu et al. 2022. Completely Heterogeneous Federated Learning. *CoRR* abs/2210.15865 (2022).
- [26] Xiaofeng Lu et al. 2022. Heterogeneous Model Fusion Federated Learning Mechanism Based on Model Mapping. *IEEE Internet Things J.* 9, 8 (2022), 6058–6068.
- [27] Disha Makhija et al. 2022. Architecture Agnostic Federated Learning for Neural Networks. In *Proc. ICML*, Vol. 162. PMLR, virtual, 14860–14870.
- [28] Brendan McMahan et al. 2017. Communication-Efficient Learning of Deep Networks from Decentralized Data. In *Proc. AISTATS*, Vol. 54. PMLR, Fort Lauderdale, FL, USA, 1273–1282.
- [29] Jaehoon Oh et al. 2022. FedBABU: Toward Enhanced Representation for Federated Image Classification. In *Proc. ICLR*. OpenReview.net, virtual.
- [30] Krishna Pillutla et al. 2022. Federated Learning with Partial Model Personalization. In *Proc. ICML*, Vol. 162. PMLR, virtual, 17716–17758.
- [31] Sebastian Ruder. 2016. An overview of gradient descent optimization algorithms. *CoRR* abs/1609.04747 (2016), 1.
- [32] Felix Sattler et al. 2021. FEDAUX: Leveraging Unlabeled Auxiliary Data in Federated Learning. *IEEE Trans. Neural Networks Learn. Syst.* 1, 1 (2021), 1–13.
- [33] Felix Sattler et al. 2022. CFD: Communication-Efficient Federated Distillation via Soft-Label Quantization and Delta Coding. *IEEE Trans. Netw. Sci. Eng.* 9, 4 (2022), 2025–2038.
- [34] Aviv Shamsian et al. 2021. Personalized Federated Learning using Hypernetworks. In *Proc. ICML*, Vol. 139. PMLR, virtual, 9489–9502.
- [35] Tao Shen et al. 2020. Federated Mutual Learning. *CoRR* abs/2006.16765 (2020).
- [36] Yue Tan et al. 2022. FedProto: Federated Prototype Learning across Heterogeneous Clients. In *Proc. AAAI*. AAAI Press, virtual, 8432–8440.
- [37] Laurens van der Maaten and Geoffrey Hinton. 2008. Visualizing Data using t-SNE. *Journal of Machine Learning Research* 9, 86 (2008), 2579–2605.
- [38] Rudin, W. 1976. *Principles of Mathematical Analysis* (3rd ed.). P McGraw-Hill., ISBN-13: 978-0070542358.
- [39] wikipedia. 2023. https://en.wikipedia.org/wiki/Dot_product.
- [40] Chuhan Wu et al. 2022. Communication-efficient federated learning via knowledge distillation. *Nature Communications* 13, 1 (2022), 2032.
- [41] Mang Ye et al. 2023. Heterogeneous Federated Learning: State-of-the-art and Research Challenges. *CoRR* abs/2307.10616 (2023), 1.
- [42] Liping Yi, Gang Wang, Xiaoguang Liu, Zhuan Shi, and Han Yu. 2023. FedGH: Heterogeneous Federated Learning with Generalized Global Header. In *Proceedings of the 31st ACM International Conference on Multimedia (ACM MM'23)*. ACM, Canada, 11.
- [43] Fuxun Yu et al. 2021. Fed2: Feature-Aligned Federated Learning. In *Proc. KDD*. ACM, virtual, 2066–2074.
- [44] Sixing Yu et al. 2022. Resource-aware Federated Learning using Knowledge Extraction and Multi-model Fusion. *CoRR* abs/2208.07978 (2022).
- [45] Jie Zhang et al. 2021. Parameterized Knowledge Transfer for Personalized Federated Learning. In *Proc. NeurIPS*. OpenReview.net, virtual, 10092–10104.
- [46] Lan Zhang et al. 2022. FedZKT: Zero-Shot Knowledge Transfer towards Resource-Constrained Federated Learning with Heterogeneous On-Device Models. In *Proc. ICDCS*. IEEE, virtual, 928–938.
- [47] Zhilu Zhang and Merv R. Sabuncu. 2018. Generalized Cross Entropy Loss for Training Deep Neural Networks with Noisy Labels. In *Proc. NeurIPS*. Curran Associates Inc., Montréal, Canada, 8792–8802.
- [48] Zhuangdi Zhu et al. 2021. Data-Free Knowledge Distillation for Heterogeneous Federated Learning. In *Proc. ICML*, Vol. 139. PMLR, virtual, 12878–12889.
- [49] Zhuangdi Zhu et al. 2022. Resilient and Communication Efficient Learning for Heterogeneous Federated Systems. In *Proc. ICML*, Vol. 162. PMLR, virtual, 27504–27526.

A ALGORITHM DESCRIPTION OF FEDLORA

Algorithm 1: FedLoRA

Input: N , total number of clients; K , number of selected clients in one round; T , total number of rounds; η_ω , learning rate of local heterogeneous models; η_θ , learning rate of local adapters; μ , weight of local heterogeneous model loss.

Randomly initialize local personalized heterogeneous models $[\mathcal{F}_0(\omega_0^0), \mathcal{F}_1(\omega_1^0), \dots, \mathcal{F}_k(\omega_k^0), \dots, \mathcal{F}_{N-1}(\omega_{N-1}^0)]$ and the global adapter $\mathcal{A}(\theta^0)$.

for each round $t=1,\dots,T-1$ **do**

- // **Server Side:**
- $S^t \leftarrow$ Randomly sample K clients from N clients;
- Broadcast the global adapter θ^{t-1} to sampled K clients;
- $\theta_k^t \leftarrow \text{ClientUpdate}(\theta^{t-1})$;
- /* Aggregate Local Adapters */

$$\theta^t = \sum_{k=0}^{K-1} \frac{n_k}{n} \theta_k^t.$$

// **ClientUpdate:**

Receive the global adapter θ^{t-1} from the server;

for $k \in S^t$ **do**

- /* Local Iterative Training */
- // Freeze Adapter, Train Model
- for** $(x, y) \in D_k$ **do**
- $\mathcal{R} = f_k(x; \omega_{k,conv}^{t-1})$;
- $\hat{y}_1 = \mathcal{A}(\mathcal{R}; \theta^{t-1})$, $\hat{y}_2 = h_k(\mathcal{R}; \omega_{k,fc}^{t-1})$;
- $\ell_1 = \ell(\hat{y}_1, y)$, $\ell_2 = \ell(\hat{y}_2, y)$;
- $\ell_\omega = (1 - \mu) \cdot \ell_1 + \mu \cdot \ell_2$;
- $\omega_k^t \leftarrow \omega_k^{t-1} - \eta_\omega \nabla \ell_\omega$;
- end**
- // Freeze Adapter, Train Model
- for** $(x, y) \in D_k$ **do**
- $\tilde{\mathcal{R}} = f_k(x; \omega_{k,conv}^t)$;
- $\hat{y} = \mathcal{A}(\tilde{\mathcal{R}}; \theta^{t-1})$;
- $\ell_\theta = \ell(\hat{y}, y)$;
- $\theta_k^t \leftarrow \theta^{t-1} - \eta_\theta \nabla \ell_\theta$;
- end**

Upload updated local adapter θ_k^t to the server.

end

end

Return personalized heterogeneous local models $[\mathcal{F}_0(\omega_0^{T-1}), \mathcal{F}_1(\omega_1^{T-1}), \dots, \mathcal{F}_k(\omega_k^{T-1}), \dots, \mathcal{F}_{N-1}(\omega_{N-1}^{T-1})]$.

B PROOF FOR LEMMA 5.3

PROOF. As formulated in Eq. (5), the local heterogeneous model of an arbitrary client k is updated by

$$\omega_{t+1} = \omega_t - \eta g_{\omega,k}^t = \omega_t - \nabla(\mu \cdot \mathcal{L}_{\omega_t} + (1-\mu) \cdot \mathcal{L}_{\theta_t}). \quad (17)$$

Based on Assumption 5.1 and Eq. (17), we can get

$$\begin{aligned} \mathcal{L}_{tE+1} &\leq \mathcal{L}_{tE+0} + \langle \nabla \mathcal{L}_{tE+0}, (\omega_{tE+1} - \omega_{tE+0}) \rangle + \frac{L_1}{2} \|\omega_{tE+1} - \omega_{tE+0}\|_2^2 \\ &= \mathcal{L}_{tE+0} - \eta \langle \nabla \mathcal{L}_{tE+0}, \nabla(\mu \cdot \mathcal{L}_{\omega_{tE+0}} + (1-\mu) \cdot \mathcal{L}_{\theta_{tE+0}}) \rangle + \frac{L_1 \eta^2}{2} \|\nabla(\mu \cdot \mathcal{L}_{\omega_{tE+0}} + (1-\mu) \cdot \mathcal{L}_{\theta_{tE+0}})\|_2^2. \end{aligned} \quad (18)$$

Take the expectations of random variable ξ_{tE+0} on both sides, we have

$$\begin{aligned} \mathbb{E}[\mathcal{L}_{tE+1}] &\leq \mathcal{L}_{tE+0} - \eta \mathbb{E}[\langle \nabla \mathcal{L}_{tE+0}, \nabla(\mu \cdot \mathcal{L}_{\omega_{tE+0}} + (1-\mu) \cdot \mathcal{L}_{\theta_{tE+0}}) \rangle] + \frac{L_1 \eta^2}{2} \mathbb{E}[\|\nabla(\mu \cdot \mathcal{L}_{\omega_{tE+0}} + (1-\mu) \cdot \mathcal{L}_{\theta_{tE+0}})\|_2^2] \\ &\stackrel{(a)}{\leq} \mathcal{L}_{tE+0} - \eta \mathbb{E}[\langle \nabla \mathcal{L}_{tE+0}, \nabla(\mu \cdot \mathcal{L}_{\omega_{tE+0}}) \rangle] + \frac{L_1 \eta^2}{2} \mathbb{E}[\|\nabla(\mu \cdot \mathcal{L}_{\omega_{tE+0}} + (1-\mu) \cdot \mathcal{L}_{\theta_{tE+0}})\|_2^2] \\ &= \mathcal{L}_{tE+0} - \eta \mu \|\nabla \mathcal{L}_{\omega_{tE+0}}\|_2^2 + \frac{L_1 \eta^2}{2} \mathbb{E}[\|\nabla(\mu \cdot \mathcal{L}_{\omega_{tE+0}} + (1-\mu) \cdot \mathcal{L}_{\theta_{tE+0}})\|_2^2] \\ &\stackrel{(b)}{=} \mathcal{L}_{tE+0} - \eta \mu \|\nabla \mathcal{L}_{\omega_{tE+0}}\|_2^2 + \frac{L_1 \eta^2}{2} (\text{Var}(\nabla(\mu \cdot \mathcal{L}_{\omega_{tE+0}} + (1-\mu) \cdot \mathcal{L}_{\theta_{tE+0}})) + \|\nabla(\mu \cdot \mathcal{L}_{\omega_{tE+0}} + (1-\mu) \cdot \mathcal{L}_{\theta_{tE+0}})\|_2^2) \\ &\stackrel{(c)}{\leq} \mathcal{L}_{tE+0} - \eta \mu \|\nabla \mathcal{L}_{\omega_{tE+0}}\|_2^2 + \frac{L_1 \eta^2}{2} ((\sigma^2 + \delta^2) + \|\nabla(\mu \cdot \mathcal{L}_{\omega_{tE+0}} + (1-\mu) \cdot \mathcal{L}_{\theta_{tE+0}})\|_2^2) \\ &\stackrel{(d)}{\leq} \mathcal{L}_{tE+0} - \eta \mu \|\nabla \mathcal{L}_{\omega_{tE+0}}\|_2^2 + \frac{L_1 \eta^2}{2} ((\sigma^2 + \delta^2) + 2\|\nabla(\mu \cdot \mathcal{L}_{\omega_{tE+0}})\|_2^2) \\ &= \mathcal{L}_{tE+0} + (L_1 \eta^2 \mu^2 - \eta \mu) \|\nabla \mathcal{L}_{\omega_{tE+0}}\|_2^2 + \frac{L_1 \eta^2 (\sigma^2 + \delta^2)}{2}, \end{aligned} \quad (19)$$

where (a): we simply denote that $\nabla \mathcal{L}_{tE+0} = A$, $\nabla(\mu \cdot \mathcal{L}_{\omega_{tE+0}}) = B$, and $\nabla((1-\mu) \cdot \mathcal{L}_{\theta_{tE+0}}) = C$. Following the additive principle of derivation, $\nabla(\mu \cdot \mathcal{L}_{\omega_{tE+0}} + (1-\mu) \cdot \mathcal{L}_{\theta_{tE+0}}) = \nabla(\mu \cdot \mathcal{L}_{\omega_{tE+0}}) + \nabla((1-\mu) \cdot \mathcal{L}_{\theta_{tE+0}}) = A + B$. So $\mathbb{E}[\langle \nabla \mathcal{L}_{tE+0}, \nabla(\mu \cdot \mathcal{L}_{\omega_{tE+0}} + (1-\mu) \cdot \mathcal{L}_{\theta_{tE+0}}) \rangle] = \mathbb{E}[\langle A, B + C \rangle]$. According to distributive over vector addition [39], $\langle A, B + C \rangle = \langle A, B \rangle + \langle A, C \rangle$. According to the geometric interpretation of the inner product, we can obtain: $\langle A \cdot C \rangle = |A| \cdot |C| \cdot \cos(\alpha)$, α is the angle between vectors A and C , $|A|$ and $|C|$ are the norm of vectors A and C . In the training process of two models on the same dataset on the same task, their gradient vectors A, C may gradually converge to similarity, with the angle α between them being less than 90 degrees and ultimately approaching 0 degrees. This is because they are both guided by similar data and task objectives, gradually adjusting parameters to make the model outputs more consistent with the training data. So we can safely consider $\cos(\alpha) \geq 0$. Since norms $|A|$ and $|C|$ are positive, $\langle A \cdot C \rangle \geq 0$. So $\langle A \cdot (B + C) \rangle - \langle A \cdot B \rangle = \langle A \cdot C \rangle \geq 0$, i.e., $\mathbb{E}[\langle \nabla \mathcal{L}_{tE+0}, \nabla(\mu \cdot \mathcal{L}_{\omega_{tE+0}} + (1-\mu) \cdot \mathcal{L}_{\theta_{tE+0}}) \rangle] - \mathbb{E}[\langle \nabla \mathcal{L}_{tE+0}, \nabla(\mu \cdot \mathcal{L}_{\omega_{tE+0}}) \rangle] \geq 0$. So $\mathbb{E}[\langle \nabla \mathcal{L}_{tE+0}, \nabla(\mu \cdot \mathcal{L}_{\omega_{tE+0}} + (1-\mu) \cdot \mathcal{L}_{\theta_{tE+0}}) \rangle] \geq \mathbb{E}[\langle \nabla \mathcal{L}_{tE+0}, \nabla(\mu \cdot \mathcal{L}_{\omega_{tE+0}}) \rangle]$, then $-\eta \mathbb{E}[\langle \nabla \mathcal{L}_{tE+0}, \nabla(\mu \cdot \mathcal{L}_{\omega_{tE+0}} + (1-\mu) \cdot \mathcal{L}_{\theta_{tE+0}}) \rangle] \leq -\eta \mathbb{E}[\langle \nabla \mathcal{L}_{tE+0}, \nabla(\mu \cdot \mathcal{L}_{\omega_{tE+0}}) \rangle]$.

(b) follows from $\text{Var}(x) = \mathbb{E}[x^2] - (\mathbb{E}[x])^2$.

(c) follows from Assumption 5.2.

(d): we denote $B = \nabla(\mu \cdot \mathcal{L}_{\omega_{tE+0}})$, $C = \nabla((1-\mu) \cdot \mathcal{L}_{\theta_{tE+0}})$, we should prove that $|B + C|_2^2 \leq 2|B|_2^2$. According to the Cauchy-Schwarz Inequality, we can have $|B + C|_2^2 \leq 2|B|_2^2 + 2|C|_2^2$, which is a derivation of the Cauchy-Schwarz Inequality proved in W [38]. Given the above inequality, since $\mu \in [0.5, 1]$, as μ approaches 1, $(1-\mu)$ approaches 0, so the second term $2|C|_2^2 = 2|\nabla((1-\mu) \cdot \mathcal{L}_{\theta_{tE+0}})|_2^2$ can be omitted. Therefore, we can get $|B + C|_2^2 \leq 2|B|_2^2$, i.e., $|\nabla(\mu \cdot \mathcal{L}_{\omega_{tE+0}} + (1-\mu) \cdot \mathcal{L}_{\theta_{tE+0}})|_2^2 \leq 2|\nabla(\mu \cdot \mathcal{L}_{\omega_{tE+0}})|_2^2$.

Take the expectations of the heterogeneous local model ω on both sides across E local iterations, we have

$$\mathbb{E}[\mathcal{L}_{(t+1)E}] \leq \mathcal{L}_{tE+0} + (L_1 \eta^2 \mu^2 - \eta \mu) \sum_{e=0}^{E-1} \|\nabla \mathcal{L}_{tE+e}\|_2^2 + \frac{L_1 \eta^2 (\sigma^2 + \delta^2)}{2}. \quad (20)$$

□

C PROOF FOR THEOREM 5.4

PROOF. Eq. (20) can be adjusted further as

$$\sum_{e=0}^{E-1} \|\nabla \mathcal{L}_{tE+e}\|_2^2 \leq \frac{\mathcal{L}_{tE+0} - \mathbb{E}[\mathcal{L}_{(t+1)E}] + \frac{L_1 \eta^2 (\sigma^2 + \delta^2)}{2}}{\eta \mu - L_1 \eta^2 \mu^2}. \quad (21)$$

Take the expectations of the heterogeneous local model ω on both sides across T communication rounds, we have

$$\frac{1}{T} \sum_{t=0}^{T-1} \sum_{e=0}^{E-1} \|\nabla \mathcal{L}_{tE+e}\|_2^2 \leq \frac{\frac{1}{T} \sum_{t=0}^{T-1} (\mathcal{L}_{tE+0} - \mathbb{E}[\mathcal{L}_{(t+1)E}]) + \frac{L_1 \eta^2 (\sigma^2 + \delta^2)}{2}}{\eta \mu - L_1 \eta^2 \mu^2}. \quad (22)$$

Let $\Delta = \mathcal{L}_{t=0} - \mathcal{L}^* > 0$, then $\sum_{t=0}^{T-1} (\mathcal{L}_{tE+0} - \mathbb{E}[\mathcal{L}_{(t+1)E}]) \leq \Delta$, so we have

$$\frac{1}{T} \sum_{t=0}^{T-1} \sum_{e=0}^{E-1} \|\nabla \mathcal{L}_{tE+e}\|_2^2 \leq \frac{\frac{\Delta}{T} + \frac{L_1 \eta^2 (\sigma^2 + \delta^2)}{2}}{\eta \mu - L_1 \eta^2 \mu^2}. \quad (23)$$

If the above equation can converge to a constant ϵ , i.e.,

$$\frac{1}{T} \sum_{t=0}^{T-1} \sum_{e=0}^{E-1} \|\nabla \mathcal{L}_{tE+e}\|_2^2 \leq \frac{\frac{\Delta}{T} + \frac{L_1 \eta^2 (\sigma^2 + \delta^2)}{2}}{\eta \mu - L_1 \eta^2 \mu^2} < \epsilon, \quad (24)$$

then

$$T > \frac{2\Delta}{2\epsilon(\eta \mu - L_1 \eta^2 \mu^2) - L_1 \eta^2 (\sigma^2 + \delta^2)}. \quad (25)$$

Since $T > 0, \Delta > 0$, so we get

$$2\epsilon(\eta \mu - L_1 \eta^2 \mu^2) - L_1 \eta^2 (\sigma^2 + \delta^2) > 0. \quad (26)$$

After solving the above inequality, we can get

$$\eta < \frac{2\epsilon \mu}{L_1 (\sigma^2 + \delta^2 + 2\mu^2 \epsilon)}. \quad (27)$$

Since $\epsilon, \mu, L_1, \sigma^2, \delta^2 > 0$ are both constants, the learning rate η of the local heterogeneous model has solutions.

Therefore, when the learning rate of the local heterogeneous model satisfies the above condition, an arbitrary client's local heterogeneous local can converge. In addition, on the right side of Eq. (23), except for $\frac{\Delta}{T}$, Δ and other items are both constants, so the non-convex convergence rate $\epsilon \sim \mathcal{O}(\frac{1}{T})$. \square

D MORE DETAILED EXPERIMENTAL SETTINGS AND RESULTS

Table 3: Structures of 5 heterogeneous CNN models with 5×5 kernel size and 16 or 32 filters in convolutional layers.

Layer Name	CNN-1	CNN-2	CNN-3	CNN-4	CNN-5
Conv1	$5 \times 5, 16$				
Maxpool1	2×2				
Conv2	$5 \times 5, 32$	$5 \times 5, 16$	$5 \times 5, 32$	$5 \times 5, 32$	$5 \times 5, 32$
Maxpool2	2×2				
FC1	2000	2000	1000	800	500
FC2	500	500	500	500	500
FC3	10/100	10/100	10/100	10/100	10/100
model size	10.00 MB	6.92 MB	5.04 MB	3.81 MB	2.55 MB

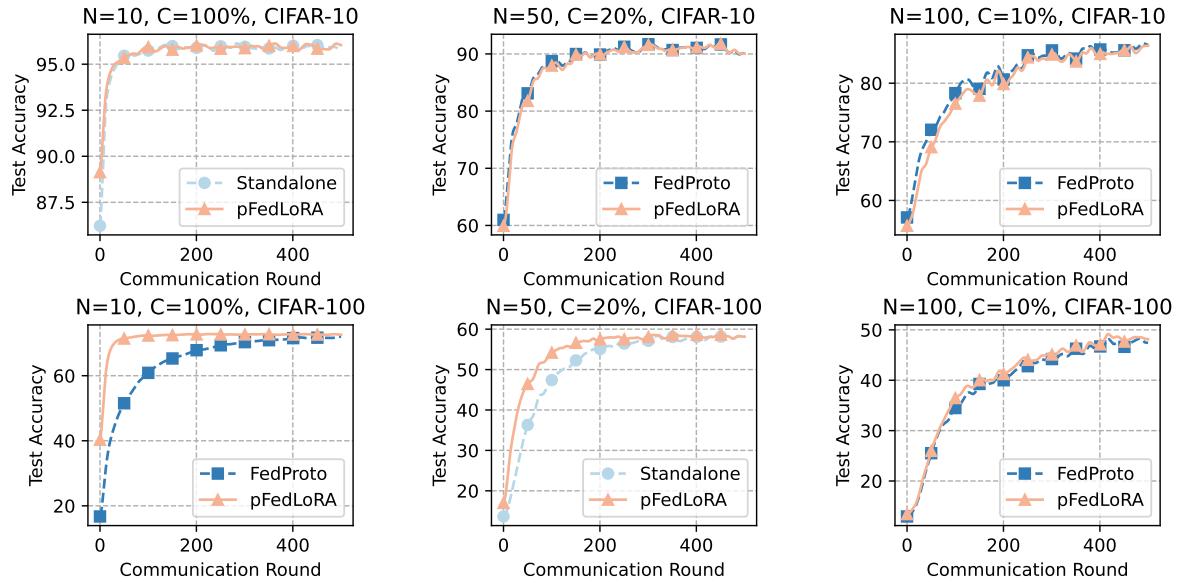


Figure 10: Average accuracy vs. communication rounds.