FedALA: Adaptive Local Aggregation for Personalized Federated Learning

Jianqing Zhang[1], Yang Hua[2], Hao Wang[3], Tao Song[1], Zhengui Xue[1], Ruhui Ma[1*], Haibing Guan[1]

[1]Shanghai Jiao Tong University [2]Queen's University Belfast [3]Louisiana State University
{tsingz, songt333, zhenguixue, ruhuima, hbguan}@sjtu.edu.cn,
Y.Hua@qub.ac.uk, haowang@lsu.edu

Abstract

A key challenge in federated learning (FL) is the statistical heterogeneity that impairs the generalization of the global model on each client. To address this, we propose a method *Federated learning with Adaptive Local Aggregation* (**FedALA**) by capturing the desired information in the global model for client models in personalized FL. The key component of FedALA is an *Adaptive Local Aggregation* (**ALA**) module, which can adaptively aggregate the downloaded global model and local model towards the local objective on each client to initialize the local model before training in each iteration. To evaluate the effectiveness of FedALA, we conduct extensive experiments with five benchmark datasets in computer vision and natural language processing domains. FedALA outperforms eleven state-of-the-art baselines by up to 3.27% in test accuracy. Furthermore, we also apply ALA module to other federated learning methods and achieve up to 24.19% improvement in test accuracy. Code is available at https://github.com/TsingZ0/FedALA.

1 Introduction

Federated learning (FL) can leverage distributed user data while preserving privacy by iteratively downloading models, training models locally on the clients, uploading models, and aggregating models on the server. A key challenge in FL is statistical heterogeneity, *e.g.*, the not independent and identically distributed (Non-IID) and unbalanced data across clients. This kind of data makes it hard to obtain a global model that generalizes to each client (McMahan et al. 2017; Reisizadeh et al. 2020; T Dinh, Tran, and Nguyen 2020).

Personalized FL (pFL) methods have been proposed to tackle statistical heterogeneity in FL. Unlike traditional FL that seeks a high-quality global model via distributed training across clients, *e.g.*, FedAvg (McMahan et al. 2017), pFL methods are proposed to prioritize the training of a local model for each client. Recent pFL studies on model aggregation on the server can be classified into three categories: (1) methods that learn a single global model and fine-tune it, including Per-FedAvg (Fallah, Mokhtari, and Ozdaglar 2020) and FedRep (Collins et al. 2021), (2) methods that learn additional personalized models, including pFedMe (T Dinh,

Tran, and Nguyen 2020) and Ditto (Li et al. 2021a), and (3) methods that learn local models with personalized (local) aggregation, including FedAMP (Huang et al. 2021), Fed-PHP (Li et al. 2021b), FedFomo (Zhang et al. 2020), APPLE (Luo and Wu 2021) and PartialFed (Sun et al. 2021).

pFL methods in Category (1) and (2) take all the information in the global model for local initialization, *i.e.*, initializing the local model before local training in each iteration. However, only the desired information that improves the quality of the local model is beneficial for the client. The global model has poor generalization ability since it has desired and undesired information for an individual client simultaneously. Thus, pFL methods in Category (3) intend to capture the desired information in the global model through personalized aggregation.

However, pFL methods in Category (3) still have shortcomings. FedAMP/FedPHP performs personalized aggregation on the server/clients without considering the local objective. FedFomo/APPLE downloads other client models and locally aggregates them with the approximated/learned aggregating weights on each client. All the parameters in one client model are assigned with the same weight, *i.e.*, model-level weight. Besides, downloading client models among clients causes high communication overhead in each iteration and also has privacy concerns since the data from other clients can be recovered through these client models (Zhu, Liu, and Han 2019). In addition, FedFomo/APPLE also requires feeding data forward in the downloaded client models to obtain the aggregating weights, which introduces additional computation overhead. PartialFed locally learns aggregation strategies to select the parameters in the global model or the local model. Still, the layer-level and binary selection cannot precisely capture the desired information in the global model. Furthermore, PartialFed uses non-overlapping samples to learn the local model and the strategy, so it can hardly learn a strategy that fully satisfies the local objective. Due to the significant modification of the learning process in FedAvg, the personalized aggregation process in these methods cannot be directly applied to most existing FL methods.

To precisely capture the desired information in the downloaded global model for each client without additional communication overhead in each iteration, we propose a novel pFL method *Federated learning with Adaptive Local*
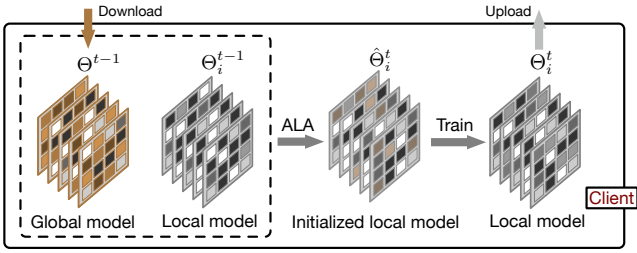
---

*Corresponding author.

Figure 1: Local learning process on client i in the t-th iteration. Specifically, client i downloads the global model from the server, locally aggregates it with the old local model by ALA module for local initialization, trains the local model, and finally uploads the trained local model to the server.

Aggregation (**FedALA**) to adaptively aggregates the downloaded global model and local model towards the local objective for local initialization. FedALA only downloads one global model and uploads one local model on each client with the same communication overhead as in FedAvg, which also has fewer privacy concerns and is more communication-efficient than FedFomo and APPLE. By adaptively learning real-value and element-wise aggregation weights towards the local objective on the full local dataset, FedALA can capture the desired information in the global model at the element level, which is more precise than the binary and layer-wise weight learning in PartialFed. Since the lower layers in a deep neural network (DNN) learn more generic information than the higher layers (Yosinski et al. 2014; LeCun, Bengio, and Hinton 2015), we can further reduce the computation overhead by only applying *Adaptive Local Aggregation* (**ALA**) module on the higher layers. The whole local learning process is shown in Figure 1.

To evaluate the effectiveness of FedALA, we conduct extensive experiments on five benchmark datasets. Results show that FedALA outperforms eleven state-of-the-art (SOTA) methods. Besides, we apply the ALA module to the traditional FL methods and pFL methods to improve their performance. To sum up, our contributions are as follows.

- We propose a novel pFL method FedALA that adaptively aggregates the global model and local model towards the local objective to capture the desired information from the global model in an element-wise manner.

- We empirically show the effectiveness of FedALA, which outperforms eleven SOTA methods by up to 3.27% in test accuracy without additional communication overhead in each iteration.

- Attributed to the minor modification of FL process, the ALA module in FedALA can be directly applied to existing FL methods to enhance their performance by up to 24.19% in test accuracy on Cifar100.

2 Related work

2.1 Traditional Federated Learning

The widely known traditional FL method FedAvg (McMahan et al. 2017) learns a single global model for all clients by aggregating their local models. However, it often suffers in statistically heterogeneous settings, *e.g.*, FL with Non-IID and unbalanced data (Kairouz et al. 2019; Zhao et al. 2018). To address this issue, FedProx (Li et al. 2020) improves the stability of the FL process through a proximal term. To counteract the bias introduced by the Non-IID data, FAVOR (Wang et al. 2020a) selects a subset of clients based on deep Q-learning (Van Hasselt, Guez, and Silver 2016) at each iteration. By generating the global model layer-wise, FedMA (Wang et al. 2020b) can adapt to statistical heterogeneity with the matched averaging approach. However, with statistical heterogeneity in FL, it is hard to obtain a single global model which generalizes well to each client (Kairouz et al. 2019; Huang et al. 2021; T Dinh, Tran, and Nguyen 2020).

2.2 Personalized Federated Learning

Recently, personalization has attracted much attention for tackling statistical heterogeneity in FL (Kairouz et al. 2019). We consider the following three categories of pFL methods that focus on aggregating models on the server:

(1) Methods that learn a single global model and fine-tune it. Per-FedAvg (Fallah, Mokhtari, and Ozdaglar 2020) considers the global model as an initial shared model based on MAML (Finn, Abbeel, and Levine 2017). By performing a few additional training steps locally, all the clients can easily fine-tune the reasonable initial shared model. FedRep (Collins et al. 2021) splits the backbone into a global model (representation) and a client-specific head and fine-tunes the head locally to achieve personalization.

(2) Methods that learn additional personalized models. pFedMe(T Dinh, Tran, and Nguyen 2020) learns an additional personalized model for each client with Moreau envelopes. In Ditto (Li et al. 2021a), each client learns its additional personalized model with a proximal term to fetch information from the downloaded global model.

(3) Methods that learn local models with personalized (local) aggregation. To further capture the personalization, recent methods try to generate client-specific models through personalized aggregation. For example, FedAMP (Huang et al. 2021) generates an aggregated model for an individual client by the attention-inducing function and personalized aggregation. To utilize the historical local model, FedPHP (Li et al. 2021b) locally aggregates the global model and local model with the rule-based moving average and a pre-defined weight (hyperparameter). Fed-Fomo (Zhang et al. 2020) focuses on aggregating other client models locally for local initialization in each iteration and approximates the client-specific weights for aggregation using the local models from other clients. Similar to FedFomo, APPLE (Luo and Wu 2021) also aggregates client models locally but learns the weights instead of approximation and performs the local aggregation in each training batch rather than just local initialization. FedAMP and APPLE are proposed for the cross-silo FL setting, which require all clients to join each iteration. By learning the binary and layer-level aggregation strategy for each client, PartialFed (Sun et al. 2021) selects the parameters in the global model or the local model to construct the new local model in each batch.

Our FedALA belongs to Category (3) but is more precise and requires less communication cost than FedFomo and APPLE. The fine-grained ALA in FedALA can element-wisely aggregate the global model and local model to adapt to the local objective on each client for local initialization. As FedALA only modifies the local initialization in FL, it can be applied to existing FL methods to improve their performance without modifying other learning processes.

3 Method

3.1 Problem Statement

Suppose we have N clients with their private training data $D_1, \ldots, D_N$, respectively. These datasets are heterogeneous (Non-IID and unbalanced). Specifically, $D_1, \ldots, D_N$ are sampled from N distinct distributions and have different sizes. With the help of a central server, our goal is to collaboratively learn individual local models $\hat{\Theta}_1, \ldots, \hat{\Theta}_N$ using $D_1, \ldots, D_N$ for each client, without exchanging the private data. The objective is to minimize the global objective and obtain the reasonable local models

$$\{\hat{\Theta}_1, \ldots, \hat{\Theta}_N\} = \arg\min \, \mathcal{G}(\mathcal{L}_1, \ldots, \mathcal{L}_N), \quad (1)$$

where $\mathcal{L}_i = \mathcal{L}(\hat{\Theta}_i, D_i; \Theta), \forall i \in [N]$ and $\mathcal{L}(\cdot)$ is the loss function. Θ is the global model, which brings external information to client i. Typically, $\mathcal{G}(\cdot)$ is set to $\sum_{i=1}^{N} k_i \mathcal{L}_i$, where $k_i = |D_i| / \sum_{j=1}^{N} |D_j|$ and $|D_i|$ is the amount of local data samples on client i.

3.2 Adaptive Local Aggregation (ALA)

The server generates a global model by aggregating trained client models in heterogeneous settings, but this global model generalizes poorly on each client. To address this problem, we propose a pFL method FedALA with a fine-grained ALA module that element-wisely aggregates the global model and local model to adapt to the local objective. We show the learning process in ALA in Figure 2.
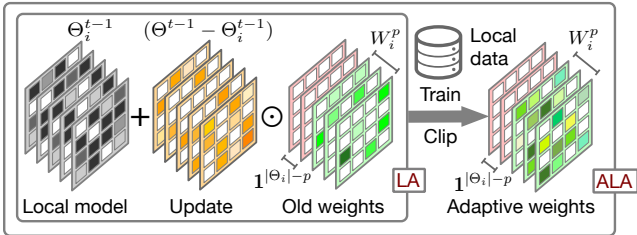

Figure 2: The learning process in ALA. LA denotes "local aggregation". Here, we consider a five-layer model and set $p = 3$. The lighter the color, the larger the value.

In traditional FL (e.g., FedAvg), after the server sends the old global model Θ^{t-1} to client i in iteration t, Θ^{t-1} overwrites the old local model Θ_i^{t-1} to obtain the initialized local model $\hat{\Theta}_i^t$ for local model training, i.e., $\hat{\Theta}_i^t := \Theta^{t-1}$. In FedALA, we element-wisely aggregate the global model

and local model instead of overwriting. Formally,

$$\hat{\Theta}_i^t := \Theta_i^{t-1} \odot W_{i,1} + \Theta^{t-1} \odot W_{i,2}, \\ s.t. \quad w_1^q + w_2^q = 1, \forall \text{ valid } q \quad (2)$$

where $\odot$ is a Hadamard product and w_1^q and w_2^q are the q-th parameters in the aggregating weights $W_{i,1}$ and $W_{i,2}$, respectively. For overwriting, $\forall$ valid $q, w_1^q \equiv 0$ and $w_2^q \equiv 1$.

However, it is hard to learn $W_{i,1}$ and $W_{i,2}$ with the constraint through the gradient-based learning method. Thus, we combine $W_{i,1}$ and $W_{i,2}$ by viewing Equation (2) as:

$$\hat{\Theta}_i^t := \Theta_i^{t-1} + (\Theta^{t-1} - \Theta_i^{t-1}) \odot W_i, \quad (3)$$

where we call the term $(\Theta^{t-1} - \Theta_i^{t-1})$ as the "update". Inspired by the previous methods (Courbariaux et al. 2016; Luo et al. 2018), we utilize element-wise weight clipping $\sigma(w) = \max(0, \min(1, w))$ for regularization (Arjovsky, Chintala, and Bottou 2017) and let $w \in [0, 1], \forall w \in W_i$.

Since the lower layers in the DNN learn more general information than the higher layers (Yosinski et al. 2014; Zhu, Hong, and Zhou 2021), the client desires most of the information in the lower layers of the global model. To reduce computation overhead, we introduce a hyperparameter p to control the range of ALA by applying it on p higher layers and overwriting the parameters in the lower layers like FedAvg for local initialization:

$$\hat{\Theta}_i^t := \Theta_i^{t-1} + (\Theta^{t-1} - \Theta_i^{t-1}) \odot [\mathbf{1}^{|\Theta_i|-p}; W_i^p], \quad (4)$$

where $|\Theta_i|$ is the number of layers (or blocks) in Θ_i^{t-1} and $\mathbf{1}^{|\Theta_i|-p}$ has the same shape of the lower layers in Θ_i^{t-1}. The elements in $\mathbf{1}^{|\Theta_i|-p}$ are ones (constants). The weight W_i^p has the same shape as the remaining p higher layers.

We initialize the value of each element in W_i^p to one in the beginning and learn W_i^p based on old W_i^p in each iteration. To further reduce computation overhead, we randomly sample s% of D_i in iteration t and denote it as $D_i^{s,t}$. Client i trains W_i^p through the gradient-based learning method:

$$W_i^p \leftarrow W_i^p - \eta \nabla_{W_i^p} \mathcal{L}(\hat{\Theta}_i^t, D_i^{s,t}; \Theta^{t-1}), \quad (5)$$

where η is the learning rate for weight learning. We freeze other trainable parameters in ALA, including the entire global model and entire local model. After local initialization, client i performs local model training as in FedAvg.

We can significantly reduce the number of the trainable parameters in ALA by choosing a small p with negligible performance decline. We show the details in Section 4.2. Besides, we observe that once we train W_i^p to converge in the second iteration (initial stage), it hardly changes in the subsequent iterations, as shown in Figure 3. In other words, W_i^p can be reused. Similar to APPLE and PartialFed, we train only one epoch for W_i^p when $t > 2$ to adapt to the changing model parameters. Note that ALA is meaningless and deactivated in the first iteration since $\Theta^0 = \Theta_i^0, \forall i \in [N]$. Algorithm 1 presents the overall FL process in FedALA.

3.3 Analysis of ALA

We omit $\sigma(\cdot)$ and set $p = |\Theta_i|$ here for simplicity without affecting the analysis. According to Equation (4) and Equation (5), we have $\nabla_{W_i} \mathcal{L}_i^t = \eta(\Theta^{t-1} - \Theta_i^{t-1}) \odot \nabla_{\hat{\Theta}_i} \mathcal{L}_i^t$,

Algorithm 1: FedALA

Input: N clients, ρ: client joining ratio, $\mathcal{L}$: loss function, Θ^0: initial global model, α: local learning rate, η: the learning rate in ALA, $s\%$: the percent of local data in ALA, p: the range of ALA, and $\sigma(\cdot)$: clip function.

Output: Reasonable local models $\hat{\Theta}_1, \ldots, \hat{\Theta}_N$

1: Server sends Θ^0 to all clients to initialize local models.
2: Clients initialize $W_i^p, \forall i \in [N]$ to ones.
3: **for** iteration $t = 1, \ldots, T$ **do**
4:     Server samples a subset $\mathcal{I}^t$ of clients according to ρ.
5:     Server sends Θ^{t-1} to $|\mathcal{I}^t|$ clients.
6:     **for** Client $i \in \mathcal{I}^t$ in parallel **do**
7:         Client i samples $s\%$ of local data.   ▷ **ALA**
8:         **if** $t = 2$ **then**   ▷ **Initial stage**
9:             **while** W_i^p does not converge **do**
10:                 Client i trains W_i^p by Equation (5).
11:                 Client i clips W_i^p using $\sigma(\cdot)$.
12:         **else if** $t > 2$ **then**
13:             Client i trains W_i^p by Equation (5).
14:             Client i clips W_i^p using $\sigma(\cdot)$.
15:         Client i obtains $\hat{\Theta}_i^t$ by Equation (4).
16:         Client i obtains Θ_i^t by   ▷ **Local model training**
            $\Theta_i^t \leftarrow \hat{\Theta}_i^t - \alpha \nabla_{\hat{\Theta}_i} \mathcal{L}(\hat{\Theta}_i^t, D_i; \Theta^{t-1})$.
17:         Client i sends Θ_i^t to the server.   ▷ **Uploading**
18:     Server obtains Θ^t by $\Theta^t \leftarrow \sum_{i \in \mathcal{I}^t} \frac{k_i}{\sum_{j \in \mathcal{I}^t} k_j} \Theta_i^t$.
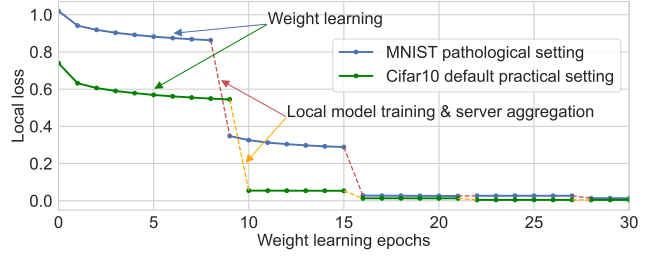19: **return** $\hat{\Theta}_1, \ldots, \hat{\Theta}_N$

Figure 3: The local loss on client #8 regarding weight learning epochs in ALA on MNIST and Cifar10. Here, we train the weights for at least six epochs in each iteration. The details of the settings are described in Section 4.

where $\mathcal{L}_i^t$ represents $\mathcal{L}(\hat{\Theta}_i^t, D_i^{s,t}; \Theta^{t-1})$. Based on Equation (4), we can view updating W_i as updating $\hat{\Theta}_i^t$ in ALA:

$$\hat{\Theta}_i^t \leftarrow \hat{\Theta}_i^t - \eta(\Theta^{t-1} - \Theta_i^{t-1}) \odot (\Theta^{t-1} - \Theta_i^{t-1}) \odot \nabla_{\hat{\Theta}_i} \mathcal{L}_i^t. \quad (6)$$

The gradient $\nabla_{\hat{\Theta}_i} \mathcal{L}_i^t$ is element-wisely scaled in iteration t. In contrast to local model training (or fine-tuning) that only focuses on the local data, the whole process of updating in Equation (6) can be aware of the generic information in the global model. Across iterations, the dynamic term $(\Theta^{t-1} - \Theta_i^{t-1})$ brings dynamic information for ALA to adapt to the complex scenarios, although it is a constant in iteration t.

4 Experiments

4.1 Experimental Setup

In this section, we firstly compare FedALA with eleven SOTA FL baselines including FedAvg (McMahan et al. 2017), FedProx (Li et al. 2020), Per-FedAvg (Fallah, Mokhtari, and Ozdaglar 2020), FedRep (Collins et al. 2021), pFedMe (T Dinh, Tran, and Nguyen 2020), Ditto (Li et al. 2021a), FedAMP (Huang et al. 2021), FedPHP (Li et al. 2021b), FedFomo (Zhang et al. 2020), APPLE (Luo and Wu 2021), and PartialFed (Sun et al. 2021). To show the superiority of weight learning in FedALA over additional local training steps, we also compare FedALA with FedAvg-C and FedProx-C, which locally fine-tune the global model for local initialization at each iteration. Then we apply ALA to SOTA FL methods and show that ALA can improve them.

We conduct extensive experiments in computer vision (CV) and natural language processing (NLP) domains.

For the CV domain, we study the image classification tasks with four widely used datasets including MNIST (LeCun et al. 1998), Cifar10/100 (Krizhevsky and Geoffrey 2009) and Tiny-ImageNet (Chrabaszcz, Loshchilov, and Hutter 2017) (100K images with 200 classes) using the 4-layer CNN (McMahan et al. 2017). For the NLP domain, we study the text classification tasks with AG News (Zhang, Zhao, and LeCun 2015) and fastText (Joulin et al. 2017). To evaluate the effectiveness of FedALA on a larger model, we also use ResNet-18 (He et al. 2016) on Tiny-ImageNet. We set the local learning rate to 0.005 for the 4-layer CNN (0.1 on MNIST following FedAvg) and 0.1 for both fastText and ResNet-18. We set the batch size to 10 and the number of local model training epochs to 1, following FedAvg. We run all the tasks for 2000 iterations to make all the methods converge empirically. Following pFedMe, we have 20 clients and set $\rho = 1$ by default.

We simulate the heterogeneous settings with two widely used scenarios. The first one is the pathological heterogeneous setting (McMahan et al. 2017; Shamsian et al. 2021), where we sample 2/2/10 classes for MNIST/Cifar10/Cifar100 from a total of 10/10/100 classes for each client, with disjoint data samples. The second scenario is the practical heterogeneous setting (Lin et al. 2020; Li, He, and Song 2021), which is controlled by the Dirichlet distribution, denoted as $Dir(\beta)$. The smaller the β is, the more heterogeneous the setting is. We set $\beta = 0.1$ for the default heterogeneous setting (Lin et al. 2020; Wang et al. 2020c).

We use the same evaluation metrics as pFedMe, which reports the test accuracy of the best single global model for the traditional FL and the average test accuracy of the best local models for pFL. To simulate the practical pFL setting, we evaluate the learned model on the client side. 25% of the local data forms the test dataset, and the remaining 75% data is used for training. We run all the tasks five times and report the mean and standard deviation.

We implement FedALA using PyTorch-1.8 and run all experiments on a server with two Intel Xeon Gold 6140 CPUs (36 cores), 128G memory, and eight NVIDIA 2080 Ti GPUs, running CentOS 7.8.

Table 1: The test accuracy (%) and the number of trainable parameters (in millions) for ALA on TINY*.

Items	$p=1$							$s=80$					
	$s=5$	$s=10$	$s=20$	$s=40$	$s=60$	$s=80$	$s=100$	$p=6$	$p=5$	$p=4$	$p=3$	$p=2$	$p=1$
Acc.	39.53	40.62	40.02	40.23	41.11	41.94	**42.11**	41.71	41.54	41.62	41.86	**42.47**	41.94
Param.	0.005							11.182	11.172	11.024	10.499	8.399	0.005

Table 2: The test accuracy (%) in the pathological heterogeneous setting and practical heterogeneous setting.

Settings	Pathological heterogeneous setting			Practical heterogeneous setting				
Methods	MNIST	Cifar10	Cifar100	Cifar10	Cifar100	TINY	TINY*	AG News
FedAvg	97.93±0.05	55.09±0.83	25.98±0.13	59.16±0.47	31.89±0.47	19.46±0.20	19.45±0.13	79.57±0.17
FedProx	98.01±0.09	55.06±0.75	25.94±0.16	59.21±0.40	31.99±0.41	19.37±0.22	19.27±0.23	79.35±0.23
FedAvg-C	99.79±0.00	92.13±0.03	66.17±0.03	90.34±0.01	51.80±0.02	30.67±0.08	36.94±0.10	95.89±0.25
FedProx-C	99.80±0.04	92.12±0.03	66.07±0.08	90.33±0.01	51.84±0.07	30.77±0.13	38.78±0.52	96.10±0.22
Per-FedAvg	99.63±0.02	89.63±0.23	56.80±0.26	87.74±0.19	44.28±0.33	25.07±0.07	21.81±0.54	93.27±0.25
FedRep	99.77±0.03	91.93±0.14	67.56±0.31	90.40±0.24	52.39±0.35	37.27±0.20	39.95±0.61	96.28±0.14
pFedMe	99.75±0.02	90.11±0.10	58.20±0.14	88.09±0.32	47.34±0.46	26.93±0.19	33.44±0.33	91.41±0.22
Ditto	99.81±0.00	92.39±0.06	67.23±0.07	90.59±0.01	52.87±0.64	32.15±0.04	35.92±0.43	95.45±0.17
FedAMP	99.76±0.02	90.79±0.16	64.34±0.37	88.70±0.18	47.69±0.49	27.99±0.11	29.11±0.15	94.18±0.09
FedPHP	99.73±0.00	90.01±0.00	63.09±0.04	88.92±0.02	50.52±0.16	35.69±3.26	29.90±0.51	94.38±0.12
FedFomo	99.83±0.00	91.85±0.02	62.49±0.22	88.06±0.02	45.39±0.45	26.33±0.22	26.84±0.11	95.84±0.15
APPLE	99.75±0.01	90.97±0.05	65.80±0.08	89.37±0.11	53.22±0.20	35.04±0.47	39.93±0.52	95.63±0.21
PartialFed	99.86±0.01	89.60±0.13	61.39±0.12	87.38±0.08	48.81±0.20	35.26±0.18	37.50±0.16	85.20±0.16
FedALA	**99.88±0.01**	**92.44±0.02**	**67.83±0.06**	**90.67±0.03**	**55.92±0.03**	**40.54±0.02**	**41.94±0.05**	**96.52±0.08**

4.2 Effect of Hyperparameters

Effect of s. From Table 1, high accuracy corresponds to large s, where "TINY*" represents using ResNet-18 on Tiny-ImageNet in the default heterogeneous setting. We can balance the performance and the computational cost by choosing a reasonable value for s. FedALA can also achieve excellent performance with only 5% local data for ALA. Since the improvement from $s=80$ to $s=100$ is negligible, we set $s=80$ for FedALA.

Effect of p. By decreasing the hyperparameter p, we can shrink the range of ALA with negligible accuracy decrease, as shown in Table 1. When p decreases from 6 to 1, the number of trainable parameters in ALA also decreases, especially from $p=2$ to $p=1$, as the last block in ResNet-18 contains most of the parameters (He et al. 2016). Although FedALA performs the best when $p=2$ here, we set $p=1$ for ResNet-18 to reduce computation overhead. Similarly, we also set $p=1$ for the 4-layer CNN and fastText. This also shows that the lower layers of the global model mostly contain generic information which is desired by the client.

4.3 Performance Comparison and Analysis

Pathological heterogeneous setting. Clients are separated into groups, which benefits the methods that measure the similarity among clients, such as FedAMP. Even so, Table 2 shows that FedALA outperforms all the baselines. Due to the poor generalization ability of the global model, FedAvg and FedProx perform poorly in this setting.

pFL methods in Category (1). Compared to the traditional FL methods, the personalized methods perform better. The accuracy for Per-FedAvg is the lowest among these methods since it only finds an initial shared model corresponding to the learning trends of all the clients, which may not capture the needs of an individual client. Fine-tuning the global model in FedAvg-C/FedProx-C generates client-specific local models, which improves the accuracy for FedAvg/Fed-Prox. However, fine-tuning only focuses on the local data and cannot be aware of the generic information during local training like FedALA. Although FedRep also fine-tunes the head at each iteration, it freezes the downloaded representation part when fine-tuning and keeps most of the generic information in the global model, thus performing excellently. However, the generic information of the head part is lost without sharing the head among clients.

pFL methods in Category (2). Although both pFedMe and Ditto use the proximal term to learn their additional personalized models, pFedMe learns the desired information from the local model while Ditto learns it from the global model. Thus, Ditto learns more generic information locally, and it performs better. However, learning the personalized model with the proximal term is an implicit way to extract the desired information.

pFL methods in Category (3). Aggregating the models using the rule-based method is aimless in capturing the desired information in the global model, so FedPHP performs worse than FedRep and Ditto. The model-level personalized aggregation in FedAMP, FedFomo, and APPLE, as well as the

Table 3: The computation overhead on TINY* and the communication overhead (transmitted parameters per iteration). Σ is the parameter amount in the backbone. α_f ($\alpha_f < 1$) is the ratio of the parameters of the feature extractor in the backbone. M ($M \geq 1$) is the number of the received client models on each client in FedFomo and APPLE.

	Computation		Communication
	Total time	Time/iter.	Param./iter.
FedAvg	365 min	1.59 min	$2 * \Sigma$
FedProx	325 min	1.99 min	$2 * \Sigma$
FedAvg-C	607 min	24.28 min	$2 * \Sigma$
FedProx-C	711 min	28.44 min	$2 * \Sigma$
Per-FedAvg	121 min	3.56 min	$2 * \Sigma$
FedRep	471 min	4.09 min	$2 * \alpha_f * \Sigma$
pFedMe	1157 min	10.24 min	$2 * \Sigma$
Ditto	318 min	11.78 min	$2 * \Sigma$
FedAMP	92 min	1.53 min	$2 * \Sigma$
FedPHP	264 min	4.06 min	$2 * \Sigma$
FedFomo	193 min	2.72 min	$(1 + M) * \Sigma$
APPLE	132 min	2.93 min	$(1 + M) * \Sigma$
PartialFed	693 min	2.13 min	$2 * \Sigma$
FedALA	7+116 min	1.93 min	$2 * \Sigma$

layer-level and binary selection in PartialFed, are imprecise, which may introduce undesired information in the global model to the local model. Furthermore, downloading multiple models on each client in each iteration has additional communication costs for FedFomo and APPLE.

By adaptively learning the aggregating weights, FedALA can explicitly capture the desired information precisely in the global model to facilitate the local model training.

Practical heterogeneous setting. We add two additional tasks using Tiny-ImageNet and one text classification task on AG News in the default practical heterogeneous setting. The results in the default heterogeneous setting with $Dir(0.1)$ are shown in Table 2, where "TINY" represents using the 4-layer CNN on Tiny-ImageNet. FedALA is still superior to other baselines, which achieves an accuracy of 3.27% higher than FedRep on TINY.

In the practical heterogeneous setting, due to the complex data distribution of each client, it is hard to measure the similarity among clients. Thus, FedAMP can not precisely assign importance to the local models through the attention-inducing function to generate aggregated models with personalized aggregation. After downloading the global model/representation, Ditto and FedRep can capture the generic information from it instead of measuring the similarity among local models. In this way, they achieve excellent performance in most of the tasks. The trainable weights are more informative than the approximated one, so APPLE performs better than FedFomo. Although FedPHP performs well on TINY, the standard deviation is relatively high. Since FedALA can adapt to the changing circumstances through ALA, it still outperforms all the baselines in the practical set-

ting. If we reuse the aggregated weights learned in the initial stage without adaptation, the accuracy drops to 33.81% on TINY. Due to the fine-grained feature of ALA, it still outperforms Per-FedAvg, pFedMe, Ditto, FedAMP, and FedFomo.

Compared to the 4-layer CNN, ResNet-18 is a larger backbone. With ResNet-18, most methods achieve a higher accuracy, including FedALA. However, it is harder for FedAMP to measure the model similarity, and the heuristic local aggregation in FedPHP performs worse when using ResNet-18. As we set $p = 1$, the number of trainable parameters in ALA is much less than that of ResNet-18, but FedALA can still achieve superior performance.

Computation Overhead. We record the total time cost for each method until convergence, as shown in Table 3. Except for the 7 min cost in the initial stage, FedALA costs 1.93 min (similar to FedAvg) in each iteration. In other words, ALA only costs additional 0.34 min for the great accuracy improvement. However, FedAvg-C, FedProx-C, and FedRep cost relatively more time than most of the methods because of the fine-tuning for the entire local model (or only the head). Due to the additional training steps for the personalized model, Ditto and pFedMe have the top computation overhead per iteration among SOTA methods. FedFomo and APPLE feed data forward in the downloaded client-side model to approximate aggregate weights and update directed relation (DR) vectors, respectively, taking additional time.

Communication Overhead. We show the communication overhead for one client in one iteration in Table 3. The communication overhead for most of the methods is the same as FedAvg, which uploads and downloads only one model. Since FedRep only transmits the representation and keeps the head locally, it has less communication overhead. FedFomo and APPLE require the highest communication overhead, as they download M client models in each iteration (Zhang et al. 2020; Luo and Wu 2021). To achieve the results in Table 2, we set $M = 20$ for them.

Heterogeneity. To study the effectiveness of FedALA in the settings with different degrees of heterogeneity, we vary the β in $Dir(\beta)$ on Tiny-ImageNet and AG News. The smaller the β is, the more heterogeneous the setting is. In Table 4, most of the pFL methods have better performance in the more heterogeneous settings. However, except for FedPHP, APPLE, PartialFed, and FedALA, these methods excessively focus on personalization but underestimate the significance of the generic information. When the data heterogeneity becomes moderate with $Dir(0.5)$ for Tiny-ImageNet, they perform worse than FedAvg.

Scalability. To show the scalability of FedALA, we conduct two experiments with 50 and 100 clients in the default heterogeneous setting. In Table 4, most of the pFL methods degenerate greatly as the client amount increases to 100, while FedALA drops less than 1% in accuracy. Since the data amount on Cifar100 is constant, the data amount on the client decreases with more clients. This aggravates the lack of local data, so precisely capturing the desired information in the global model becomes more critical.

Table 4: The test accuracy (%) (and improvement (%)) on Tiny-ImageNet, AG News, and Cifar100.

	Heterogeneity			Scalability		Applicability of ALA			
	Tiny-ImageNet		AG News	Cifar100		Tiny-ImageNet		Cifar100	
Datasets									
Methods	$Dir(0.01)$	$Dir(0.5)$	$Dir(1)$	50 clients	100 clients	Acc.	Imps.	Acc.	Imps.
FedAvg	15.70±0.46	21.14±0.47	87.12±0.19	31.90±0.27	31.95±0.37	40.54±0.17	21.08	55.92±0.15	24.03
FedProx	15.66±0.36	21.22±0.47	87.21±0.13	31.94±0.30	31.97±0.24	40.53±0.26	21.16	56.18±0.65	24.19
FedAvg-C	49.88±0.11	16.21±0.05	91.38±0.21	49.82±0.11	47.90±0.12	—	—	—	—
FedProx-C	49.84±0.02	16.36±0.19	92.03±0.19	49.79±0.14	48.02±0.02	—	—	—	—
Per-FedAvg	39.39±0.30	16.36±0.13	87.08±0.26	44.31±0.20	36.07±0.24	30.90±0.28	5.83	48.68±0.36	4.40
FedRep	55.43±0.15	16.74±0.09	92.25±0.20	47.41±0.18	44.61±0.20	37.89±0.31	0.62	53.02±0.11	0.63
pFedMe	41.45±0.14	17.48±0.61	87.08±0.18	48.36±0.64	46.45±0.18	27.30±0.24	0.37	47.91±0.21	0.57
Ditto	50.62±0.02	18.98±0.05	91.89±0.17	54.22±0.04	52.89±0.22	40.75±0.06	8.60	56.33±0.07	3.46
FedAMP	48.42±0.06	12.48±0.21	83.35±0.05	44.39±0.35	40.43±0.17	28.18±0.20	0.19	48.03±0.23	0.34
FedPHP	48.63±0.02	21.09±0.07	90.52±0.19	52.44±0.16	49.70±0.31	40.16±0.24	4.47	54.28±0.21	3.76
FedFomo	46.36±0.54	11.59±0.11	91.20±0.18	42.56±0.33	38.91±0.08	—	—	—	—
APPLE	48.04±0.10	24.28±0.21	84.10±0.18	55.06±0.20	52.81±0.29	—	—	—	—
PartialFed	49.38±0.02	24.20±0.10	91.01±0.28	48.95±0.07	39.31±0.01	35.40±0.02	0.14	48.99±0.05	0.18
FedALA	**55.75±0.02**	**27.85±0.06**	**92.45±0.10**	**55.61±0.02**	**54.68±0.57**	—	—	—	—

4.4 Applicability of ALA

As the ALA module only modifies the local initialization in FL, it can be applied to most existing FL methods. We apply ALA to the SOTA FL methods except for FedFomo and APPLE (as they download multiple client models) without modifying other learning processes to evaluate the effectiveness of ALA. We report the accuracy and improvements on Tiny-ImageNet and Cifar100 using the 4-layer CNN in default heterogeneous setting with $s = 80$ and $p = 1$.

In Table 4, the accuracy improvement for FedAvg and FedProx is apparent, and the improvement for Per-FedAvg, Ditto, and FedPHP is also remarkable. This indicates the applicability of ALA to the traditional FL and pFL methods.

However, the improvement to other pFL methods is relatively small. In FedAMP, only the information in the important local models is emphasized through the attention-inducing function with the generic information in unimportant models ignored, so ALA can hardly find the ignored information back without modifying other learning processes. According to Table 1, the representation part mostly contains generic information, which is desired by the client. It leaves little room for ALA to take effect, but ALA still improves FedRep by more than 0.60% in accuracy. pFedMe learns a personalized model to approximate the local model at each local training batch, so it benefits little (0.57% on Cifar100) from ALA. In PartialFed, the local aggregation happens in each training batch, so the initialized local model generated by ALA is later re-aggregated by its strategy, thus eliminating the effect of ALA.

4.5 Update Direction Correction

Compared to overwriting the local model with the downloaded global model, ALA can correct the update direction for the local model with the desired information in the global model. We visualize the learning trajectory of the local model on client #4 using the visualization method (Li
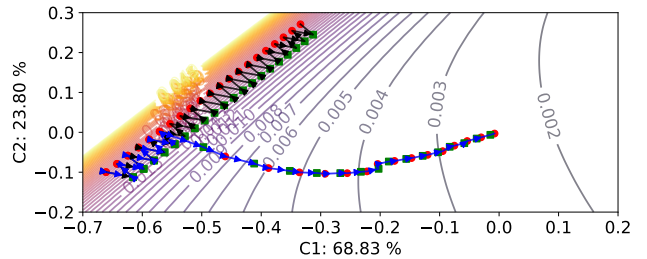

Figure 4: 2D visualization of local learning trajectory (from iteration 140 to 200) and the local loss surface on MNIST in the pathological heterogeneous setting. The green square dots and red circles represent the local model at the beginning and end of each iteration, respectively. The black and blue trajectories with the arrows represent FedAvg and FedALA, respectively. The local models are projected to the 2D plane using PCA. C1 and C2 are the two principal components generated by the PCA.

et al. 2018). We deactivate the ALA for FedALA in the first 155 iterations and activate it in the subsequent iterations.

Without capturing the desired information in the global model, the update direction of local model is misled by the global model in FedAvg, as shown by the black trajectory in Figure 4. Once the ALA is activated, the update direction of local model is corrected to the local loss reducing direction, as shown by the blue trajectory in Figure 4.

5 Conclusion

In this paper, we propose an adaptive and fine-grained method FedALA in pFL to facilitate the local model training with the received global model. The extensive experiments demonstrate the effectiveness of FedALA. Our method outperforms eleven SOTA methods. Also, the ALA module in FedALA can improve other FL methods in accuracy.

References

Arjovsky, M.; Chintala, S.; and Bottou, L. 2017. Wasserstein generative adversarial networks. In *ICML*.

Chrabaszcz, P.; Loshchilov, I.; and Hutter, F. 2017. A Downsampled Variant of Imagenet as an Alternative to the Cifar Datasets. *arXiv preprint arXiv:1707.08819*.

Collins, L.; Hassani, H.; Mokhtari, A.; and Shakkottai, S. 2021. Exploiting Shared Representations for Personalized Federated Learning. In *ICML*.

Courbariaux, M.; Hubara, I.; Soudry, D.; El-Yaniv, R.; and Bengio, Y. 2016. Binarized Neural Networks: Training Deep Neural Networks with Weights and Activations Constrained to +1 or -1. *arXiv preprint arXiv:1602.02830*.

Fallah, A.; Mokhtari, A.; and Ozdaglar, A. 2020. Personalized Federated Learning with Theoretical Guarantees: A Model-Agnostic Meta-Learning Approach. In *NeurIPS*.

Finn, C.; Abbeel, P.; and Levine, S. 2017. Model-Agnostic Meta-Learning for Fast Adaptation of Deep Networks. In *ICML*.

He, K.; Zhang, X.; Ren, S.; and Sun, J. 2016. Deep Residual Learning for Image Recognition. In *CVPR*.

Huang, Y.; Chu, L.; Zhou, Z.; Wang, L.; Liu, J.; Pei, J.; and Zhang, Y. 2021. Personalized Cross-Silo Federated Learning on Non-IID Data. In *AAAI*.

Joulin, A.; Grave, E.; Bojanowski, P.; and Mikolov, T. 2017. Bag of Tricks for Efficient Text Classification". In *EACL*.

Kairouz, P.; McMahan, H. B.; Avent, B.; Bellet, A.; Bennis, M.; Bhagoji, A. N.; Bonawitz, K.; Charles, Z.; Cormode, G.; Cummings, R.; et al. 2019. Advances and Open Problems in Federated Learning. *arXiv preprint arXiv:1912.04977*.

Krizhevsky, A.; and Geoffrey, H. 2009. Learning Multiple Layers of Features From Tiny Images. *Citeseer*.

LeCun, Y.; Bengio, Y.; and Hinton, G. 2015. Deep learning. *nature*.

LeCun, Y.; Bottou, L.; Bengio, Y.; and Haffner, P. 1998. Gradient-based Learning Applied to Document Recognition. *Proceedings of the IEEE*, 86(11): 2278–2324.

Li, H.; Xu, Z.; Taylor, G.; Studer, C.; and Goldstein, T. 2018. Visualizing the Loss Landscape of Neural Nets. In *NeurIPS*.

Li, Q.; He, B.; and Song, D. 2021. Model-Contrastive Federated Learning. In *CVPR*.

Li, T.; Hu, S.; Beirami, A.; and Smith, V. 2021a. Ditto: Fair and Robust Federated Learning Through Personalization. In *ICML*.

Li, T.; Sahu, A. K.; Zaheer, M.; Sanjabi, M.; Talwalkar, A.; and Smith, V. 2020. Federated Optimization in Heterogeneous Networks. In *MLSys*.

Li, X.-C.; Zhan, D.-C.; Shao, Y.; Li, B.; and Song, S. 2021b. FedPHP: Federated Personalization with Inherited Private Models. In *ECML PKDD*.

Lin, T.; Kong, L.; Stich, S. U.; and Jaggi, M. 2020. Ensemble Distillation for Robust Model Fusion in Federated Learning. In *NeurIPS*.

Luo, J.; and Wu, S. 2021. Adapt to Adaptation: Learning Personalization for Cross-Silo Federated Learning. *arXiv preprint arXiv:2110.08394*.

Luo, L.; Xiong, Y.; Liu, Y.; and Sun, X. 2018. Adaptive Gradient Methods with Dynamic Bound of Learning Rate. In *ICLR*.

McMahan, B.; Moore, E.; Ramage, D.; Hampson, S.; and y Arcas, B. A. 2017. Communication-Efficient Learning of Deep Networks from Decentralized Data. In *AISTATS*.

Reisizadeh, A.; Mokhtari, A.; Hassani, H.; Jadbabaie, A.; and Pedarsani, R. 2020. Fedpaq: A Communication-Efficient Federated Learning Method with Periodic Averaging and Quantization. In *AISTATS*.

Shamsian, A.; Navon, A.; Fetaya, E.; and Chechik, G. 2021. Personalized Federated Learning using Hypernetworks. In *ICML*.

Sun, B.; Huo, H.; Yang, Y.; and Bai, B. 2021. PartialFed: Cross-Domain Personalized Federated Learning via Partial Initialization. In *NeurIPS*.

T Dinh, C.; Tran, N.; and Nguyen, T. D. 2020. Personalized Federated Learning with Moreau Envelopes. In *NeurIPS*.

Van Hasselt, H.; Guez, A.; and Silver, D. 2016. Deep reinforcement learning with double q-learning. In *AAAI*.

Wang, H.; Kaplan, Z.; Niu, D.; and Li, B. 2020a. Optimizing Federated Learning on Non-IID Data with Reinforcement Learning. In *InfoComm*.

Wang, H.; Yurochkin, M.; Sun, Y.; Papailiopoulos, D.; and Khazaeni, Y. 2020b. Federated learning with matched averaging. *arXiv preprint arXiv:2002.06440*.

Wang, J.; Liu, Q.; Liang, H.; Joshi, G.; and Poor, H. V. 2020c. Tackling the Objective Inconsistency Problem in Heterogeneous Federated Optimization. In *NeurIPS*.

Yosinski, J.; Clune, J.; Bengio, Y.; and Lipson, H. 2014. How Transferable Are Features in Deep Neural Networks? In *NeurIPS*.

Zhang, M.; Sapra, K.; Fidler, S.; Yeung, S.; and Alvarez, J. M. 2020. Personalized Federated Learning with First Order Model Optimization. In *ICLR*.

Zhang, X.; Zhao, J.; and LeCun, Y. 2015. Character-level Convolutional Networks for Text Classification. In *NeurIPS*.

Zhao, Y.; Li, M.; Lai, L.; Suda, N.; Civin, D.; and Chandra, V. 2018. Federated learning with non-iid data. *arXiv preprint arXiv:1806.00582*.

Zhu, L.; Liu, Z.; and Han, S. 2019. Deep leakage from gradients. In *NeurIPS*.

Zhu, Z.; Hong, J.; and Zhou, J. 2021. Data-Free Knowledge Distillation for Heterogeneous Federated Learning. In *ICML*.