LoRA-FAIR: Federated LoRA Fine-Tuning with Aggregation and Initialization Refinement

Jieming Bian[1]* Lei Wang[1]* Letian Zhang[2] Jie Xu[1]

[1] University of Florida [2]Middle Tennessee State University

jieming.bian@ufl.edu, leiwang1@ufl.edu, letian.zhang@mtsu.edu, jie.xu@ufl.edu

Abstract

Foundation models (FMs) achieve strong performance across diverse tasks with task-specific fine-tuning, yet full parameter fine-tuning is often computationally prohibitive for large models. Parameter-efficient fine-tuning (PEFT) methods like Low-Rank Adaptation (LoRA) reduce this cost by introducing low-rank matrices for tuning fewer parameters. While LoRA allows for efficient fine-tuning, it requires significant data for adaptation, making Federated Learning (FL) an appealing solution due to its privacy-preserving collaborative framework. However, combining LoRA with FL introduces two key challenges: the ***Server-Side Aggregation Bias****, where server-side averaging of LoRA matrices diverges from the ideal global update, and the* ***Client-Side Initialization Lag****, emphasizing the need for consistent initialization across rounds. Existing approaches address these challenges individually, limiting their effectiveness. We propose LoRA-FAIR, a novel method that tackles both issues by introducing a correction term on the server, enhancing aggregation efficiency and accuracy. LoRA-FAIR maintains computational and communication efficiency, yielding superior performance over state-of-the-art methods. Experimental results on ViT and MLP-Mixer models across large-scale datasets demonstrate that LoRA-FAIR consistently achieves performance improvements in FL settings.*

1. Introduction

Emerging foundation models (FMs) [1, 4, 38, 45, 49] have demonstrated remarkable capabilities by providing robust and versatile architectures that can be adapted to a wide array of tasks through fine-tuning with task-specific data. These models excel across diverse applications, including image generation from prompts, language translation, mathematical problem-solving, and natural language conversation, among others [49]. However, the standard method of fine-tuning all model parameters, known as full parameter fine-tuning, entails prohibitively high computational costs, particularly for large-scale models. To alleviate this problem, parameter-efficient fine-tuning (PEFT) methods [13] have been proposed. One of the most important PEFT approaches is low-rank adaptation (LoRA) [16], which significantly reduces the number of trainable parameters by introducing low-rank matrices into the model.

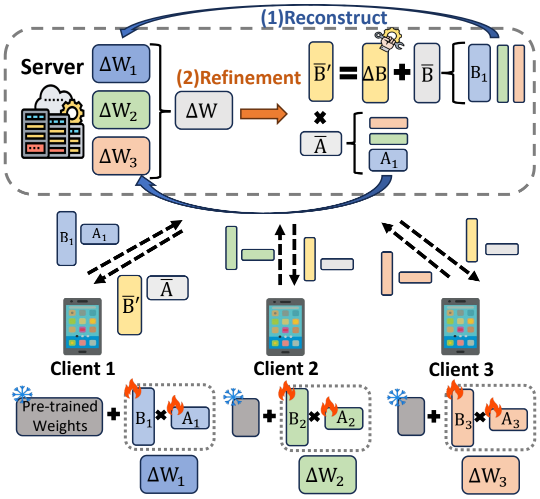

Figure 1. **Illustration of LoRA-FAIR.** Instead of directly averaging the local LoRA modules $\mathbf{A}_k$ and $\mathbf{B}_k$ collected from each client k on the server side and sending the averaged LoRA modules $\bar{\mathbf{A}}$ and $\bar{\mathbf{B}}$ back to clients, LoRA-FAIR reconstructs the ideal global update $\mathbf{\Delta W}$ using Eq. (6), finds the residual LoRA module $\mathbf{\Delta B}$ using Eq. (8), and replaces $\bar{\mathbf{B}}$ with the corrected LoRA modules $\bar{\mathbf{B}}' = \bar{\mathbf{B}} + \mathbf{\Delta B}$. See details in Sec. 4.

LoRA introduces a parallel branch of trainable low-rank matrices, $\mathbf{A}$ and $\mathbf{B}$, to compute the model update $\mathbf{\Delta W}$, where the ranks of $\mathbf{A}$ and $\mathbf{B}$ are significantly smaller than the parameters of the pre-trained model, $\mathbf{W}$. In LoRA fine-tuning, only $\mathbf{A}$ and $\mathbf{B}$ are updated, while $\mathbf{W}$ remains frozen. This approach greatly reduces the computational resources required, allowing for efficient fine-tuning with performance comparable to that of full parameter fine-tuning.

*The first two authors contributed equally to this work, and their names are listed in random order.

Despite these advantages, LoRA still requires substantial data to adapt effectively to specific downstream tasks. However, data from a single device may not be sufficient for this purpose, and fine-tuning often involves multiple devices that collectively hold the necessary data. This multi-device setup can raise privacy concerns, as fine-tuning with data from multiple parties may expose sensitive information. Federated Learning (FL) [26] offers a feasible solution to this issue. By enabling collaborative learning without requiring data sharing, FL allows participants to fine-tune models while addressing privacy concerns effectively.

Compared to studies on LoRA fine-tuning in centralized settings, fine-tuning LoRA within a FL environment remains relatively unexplored and presents unique challenges. In this paper, we investigate traditional FL in conjunction with parameter-efficient fine-tuning methods, specifically focusing on LoRA. We argue that fine-tuning LoRA modules presents two key challenges. First, which we refer to as the **Challenge 1: Server-Side Aggregation Bias**, arises because averaging the LoRA components ($\mathbf{A}$ and $\mathbf{B}$) independently at the server does not capture the ideal global update, potentially introducing noise into the aggregated model. Second, **Challenge 2: Client-Side Initialization Lag** highlights the importance of properly allocating global updates to each client's pre-trained model and LoRA modules at the start of the next local training phase to ensure a consistent initialization and mitigate initialization lag. Existing FL methods for fine-tuning fail to consider these two key points simultaneously. While some methods, such as FLoRA [42], attempt to address Challenge 1 by altering the aggregation process, they fail to address Challenge 2, which limits the performance to a level comparable to that of directly combining FedAvg and LoRA (i.e., FedIT [46]).

Taking both Challenge 1 and Challenge 2 into consideration simultaneously is essential for maximizing the performance of LoRA fine-tuning in a federated learning setting. In this work, we propose a simple yet effective method, LoRA-FAIR (short for LoRA with Federated Aggregation and Initialization Refinement), designed to tackle both challenges concurrently. Specifically, we propose that, on the server side, the original averaged LoRA modules (e.g., $\bar{\mathbf{A}}$ and $\bar{\mathbf{B}}$) be kept fixed while introducing a correction term $\mathbf{\Delta B}$ to $\bar{\mathbf{B}}$. This way, the product of the fine-tuned $\bar{\mathbf{B}} + \mathbf{\Delta B}$ and $\bar{\mathbf{A}}$ will closely approximate the ideal server update. To further enhance stability, we introduce a normalization term to ensure that the fine-tuned LoRA module remains close to its original averaged value, thereby preserving the average information collected from each client. Through this simple yet effective design, LoRA-FAIR provides an approach that approximates an ideal solution to both challenges by preserving the shared average information in the initial model while striving for accurate aggregation on the server side. Consequently, LoRA-FAIR maximizes the efficacy of LoRA fine-tuning within an FL framework, balancing performance improvements with computational efficiency. Our key contributions are summarized as follows:

- We investigate the problem of fine-tuning with LoRA in federated learning setting. Through an initial set of motivation experiments, we identify two key challenges that currently limit the application of LoRA in FL.
- In response to these challenges, we introduce a novel method named LoRA-FAIR. LoRA-FAIR is the first in the federated fine-tuning domain to simultaneously consider both the two challenges while maintaining computational and communication efficiency.
- We conduct experiments using two pre-trained foundation models, ViT [11] and MLP-Mixer [37], across various large-scale datasets. Our proposed LoRA-FAIR consistently outperforms state-of-the-art methods.

2. Preliminaries

2.1. PEFT with LoRA

LoRA (Low-Rank Adaptation) is a PEFT (parameter-efficient fine-tuning) approach that significantly reduces the number of trainable parameters in large-scale models by introducing low-rank matrices into the model. Consider a pre-trained model with parameters $\mathbf{W}_0 \in \mathbb{R}^{d\times l}$, where $\mathbf{W}_0$ represents the fixed parameters of the model, and $\mathbf{\Delta W} \in \mathbb{R}^{d\times l}$ denotes the trainable update matrix applied during fine-tuning. Rather than updating all elements in $\mathbf{\Delta W}$, LoRA decomposes $\mathbf{\Delta W}$ into two low-rank matrices $\mathbf{A} \in \mathbb{R}^{r\times l}$ and $\mathbf{B} \in \mathbb{R}^{d\times r}$, where $r \ll \min(d, l)$. Thus, the model update is expressed as $\mathbf{\Delta W} = \mathbf{BA}$, allowing the fine-tuning process to focus on the much smaller low-rank matrices $\mathbf{A}$ and $\mathbf{B}$ instead of the full matrix $\mathbf{\Delta W}$. Consequently, the total number of parameters that need to be trained is reduced from $d \times l$ to $r \times (d + l)$, where r is significantly smaller than both d and l. The updated model parameters after fine-tuning are given by:

$$\mathbf{W} = \mathbf{W}_0 + \mathbf{\Delta W} = \mathbf{W}_0 + \mathbf{BA}. \tag{1}$$

In practice, $\mathbf{A}$ is typically initialized with random Gaussian values, while $\mathbf{B}$ is initialized to zero to ensure a stable start to the fine-tuning process. This low-rank adaptation enables LoRA to achieve performance comparable to full fine-tuning while significantly reducing the computational and memory overhead.

2.2. Federated Learning

In a standard federated learning setup, multiple clients collaboratively train a shared global model without sharing their local data, thereby preserving privacy. Each client trains on its local data and then transmits its local model updates back to the server, which aggregates these updates to refine the global model.

Consider an FL setup with K clients, starting with an initial model $\mathbf{W}_0$. The server collects the local updates from the clients and calculates the global update as follows:

$$\boldsymbol{\Delta}\mathbf{W} = \sum_{k=1}^{K} p_k \boldsymbol{\Delta}\mathbf{W}_k, \tag{2}$$

where $\mathcal{D}_k$ is the client k's local dataset, the weights $p_k = \frac{|\mathcal{D}_k|}{\sum_k |\mathcal{D}_k|}$ are proportional to the size of each client's local dataset, and $\boldsymbol{\Delta}\mathbf{W}_k$ denotes the local update from client k. To start the next round of local training, the server uses the global update $\boldsymbol{\Delta}\mathbf{W}$ to generate an updated global model, which is then distributed to each client as the initial model for the subsequent round. The next round of training for each client can be represented as follows, assuming clients train for E epochs during local training:

$$\begin{aligned} &\mathbf{W}_{k,0} = \mathbf{W}_0 + \boldsymbol{\Delta}\mathbf{W}; \\ &\mathbf{W}_{k,e+1} = \mathbf{W}_{k,e} - \eta g_{k,e}, \quad e = 0, \ldots, E-1; \\ &\boldsymbol{\Delta}\mathbf{W}_k = -\sum_{e=0}^{E-1} \eta g_{k,e}, \end{aligned} \tag{3}$$

where η is the local learning rate, and $g_{k,e}$ represents the stochastic gradient for client k at epoch e.

3. Challenges when Combining LoRA with Federated Learning

Fine-tuning foundation models in federated learning using full-parameter updates aligns with traditional FL methods. However, incorporating LoRA introduces unique challenges that diverge from those in centralized settings.

3.1. Challenge 1: Server-Side Aggregation Bias

To discuss this challenge, we first introduce a basic method that combines LoRA directly with FL, known as FedIT [46]. In FedIT, each of the K clients starts with a fixed pre-trained foundation model $\mathbf{W}_0$ and trains the local LoRA modules represented as low-rank matrices $\mathbf{A}_k$ and $\mathbf{B}_k$ on its private dataset $\mathcal{D}_k$. The server then aggregates these local matrices uploaded by clients into global LoRA modules, $\bar{\mathbf{A}}$ and $\bar{\mathbf{B}}$, through a weighted average based on data size:

$$\bar{\mathbf{A}} = \sum_{k=1}^{K} p_k \mathbf{A}_k, \quad \bar{\mathbf{B}} = \sum_{k=1}^{K} p_k \mathbf{B}_k, \tag{4}$$

where $p_k = \frac{|\mathcal{D}_k|}{\sum_{k=1}^{K} |\mathcal{D}_k|}$ reflects each client's data proportion. Using these averaged matrices, the server distributes them back to the clients for subsequent training rounds. In FedIT, the actual global update received by each client is:

$$\boldsymbol{\Delta}\mathbf{W}' = \bar{\mathbf{B}}\bar{\mathbf{A}} = \left(\sum_{k=1}^{K} p_k \mathbf{B}_k\right)\left(\sum_{k=1}^{K} p_k \mathbf{A}_k\right). \tag{5}$$

However, this aggregated update deviates from the ideal global model update in the typical FL setting, which should be the weighted sum of all local model updates:

$$\boldsymbol{\Delta}\mathbf{W} = \sum_{k=1}^{K} p_k \boldsymbol{\Delta}\mathbf{W}_k = \sum_{k=1}^{K} p_k \mathbf{B}_k \mathbf{A}_k \neq \boldsymbol{\Delta}\mathbf{W}'. \tag{6}$$

This discrepancy, termed **Server-Side Aggregation Bias**, occurs because the approximate global update $\boldsymbol{\Delta}\mathbf{W}'$ fails to accurately capture the ideal global update $\boldsymbol{\Delta}\mathbf{W}$. To demonstrate this, we compare the two aggregation methods under a single global round with 50 local epochs independent of the client-side initialization on the DomainNet dataset. As shown in Fig. 2, **AvgToMul** and **MulToAvg** denotes the aggregated update using $\boldsymbol{\Delta}\mathbf{W}'$ and $\boldsymbol{\Delta}\mathbf{W}$ respectively. Although **AvgToMul** reduces communication costs by only transmitting the LoRA modules, it does so at the expense of alignment with the intended global model update. This challenge highlights the need for more refined aggregation methods when integrating LoRA into FL frameworks.

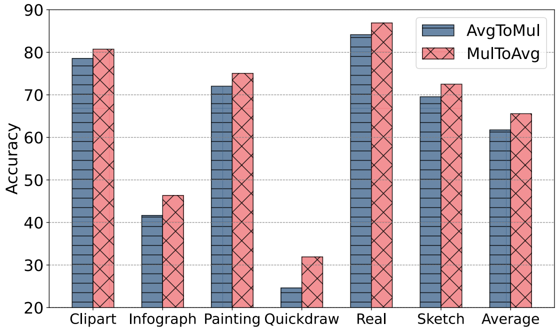

Figure 2. **Comparison of two aggregation strategies: AvgToMul and MulToAvg. AvgToMul** averages the LoRA matrices $\mathbf{A}_k$ and $\mathbf{B}_k$ from clients, then multiplies the averages to obtain the approximate global update $\boldsymbol{\Delta}\mathbf{W}'$ using Eq. (5). **MulToAvg** first multiplies each client's matrices (yielding $\mathbf{B}_k\mathbf{A}_k$) and then averages these products for the true global update $\boldsymbol{\Delta}\mathbf{W}$ using Eq. (6). While **AvgToMul** is communication-efficient, **MulToAvg** better captures the intended global model update. See details in Sec. 3.1.

3.2. Challenge 2: Client-Side Initialization Lag

To mitigate server-side aggregation bias, FFA-LoRA [33] was proposed, which freezes the non-zero-initialized low-rank matrix $\mathbf{A}$ while updating only the zero-initialized matrix $\mathbf{B}$. However, this approach slows fine-tuning and limits its performance due to the reduced number of trainable parameters. A more recent method, FLoRA [42], stacks local LoRA modules from all clients and transmits the aggregated modules back to each client to reconstruct global

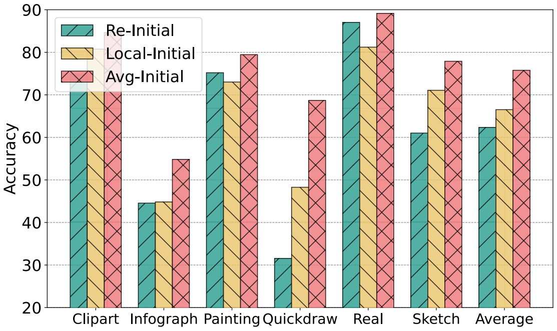

Figure 3. **Comparison of three initialization strategies: Avg-Initial, Re-Initial, Local-Initial.** The **Avg-Initial** method is the most effective as it balances continuity and unification across clients, mitigating client initialization lag and promoting better performance. For more details, refer to Sec. 3.2.

updates. These updates are then added directly to each local pre-trained model, while the local LoRA modules are reinitialized for the next training round. FLoRA effectively addresses **Challenge 1** by stacking local LoRA modules, ensuring that each client receives an ideal $\mathbf{\Delta W}$ update to add to the pre-trained model. However, this method incurs high communication costs proportional to the number of clients and raises privacy concerns, as it requires distributing all clients' LoRA modules to each client rather than only the averaged modules, as in FedIT.

Furthermore, in FLoRA, the client's local LoRA modules are reinitialized (randomizing $\mathbf{A}$ with a Gaussian distribution and setting $\mathbf{B}$ to zero). This reinitialization strategy can lead to **Client-Side Initialization Lag**. Given an input x and output y at a layer, a forward pass with LoRA modules is represented as: $y = x(\mathbf{W}_0 + \mathbf{BA})$. Accordingly, the gradients of $\mathbf{A}$ and $\mathbf{B}$ are:

$$\frac{\partial L}{\partial \mathbf{A}} = x^\top \frac{\partial L}{\partial y} \mathbf{B}^\top, \quad \frac{\partial L}{\partial \mathbf{B}} = \mathbf{A}^\top x^\top \frac{\partial L}{\partial y}. \tag{7}$$

When $\mathbf{A}$ is initialized with Gaussian noise and $\mathbf{B}$ is set to zero, the initial gradients are small and uninformative (i.e., $\frac{\partial L}{\partial \mathbf{A}} \to 0$, $\frac{\partial L}{\partial \mathbf{B}} \to$ *random direction*), leading to a slow learning start. LoRA then spends significant time near its initialization before meaningful updates occur [27]. In FL setting, where clients often have non-IID data, a prolonged local training phase can exacerbate global convergence issues [48]. Under a limited number of local training epochs, reinitialization may prevent the model from effectively capturing optimal local updates. Consequently, the locally learned information sent to the server may be ***suboptimal***, which in turn degrades the global model's performance—despite FLoRA's ability to mitigate server-side aggregation bias.

This raises the question: *what is the optimal way for a client to allocate the received global update to mitigate Client-Side Initialization Lag?* To evaluate the impact of different client-side initialization methods on model performance, we consider three strategies in an FL setup with six clients, each assigned a unique domain from the DomainNet dataset. To isolate the effect of initialization strategies from server-side aggregation, all clients receive the bias $\mathbf{\Delta W'}$ from the server in different formats based on the applied strategies. The three strategies are described in Tab. 1. **Local-Initial** requires an additional randomly selected client's last-round local LoRA modules $\mathbf{A_s}, \mathbf{B_s}$ as the LoRA initialization point for all clients. Although all approaches result in the same overall initial model (i.e. $\mathbf{W_0} + \mathbf{B_k A_k} \leftarrow \mathbf{W_0} + \mathbf{\Delta W'}$) at the start of the current training round, as shown in Fig. 3, the **Avg-Initial** method is the most effective compared to **Re-Initial** and **Local-Initial**. Compared to **Re-Initial**, Avg-Initial ensures continuity in LoRA module training, preventing the risk of uploading suboptimal client updates to the server. Compared to **Local-Initial**, it effectively disseminates global information across LoRA module training, promoting better knowledge sharing. By averaging local LoRA modules, this method captures a representative update, smooths extreme deviations, and fosters a more stable and consistent training.

Strategies	$\mathbf{W_0} \leftarrow$	$\mathbf{A_k} \leftarrow$	$\mathbf{B_k} \leftarrow$	Overall Initial Model
Avg-Initial	$\mathbf{W_0}$	$\bar{\mathbf{A}}$	$\bar{\mathbf{B}}$	$\mathbf{W_0} + \mathbf{\Delta W'}$
Re-Initial	$\mathbf{W_0} + \Delta\mathbf{W'}$	*Random Gaussian*	$\mathbf{0}$	$\mathbf{W_0} + \mathbf{\Delta W'}$
Local-Initial	$\mathbf{W_0} + \Delta\mathbf{W'} - \mathbf{B_s A_s}$	$\mathbf{A_s}$	$\mathbf{B_s}$	$\mathbf{W_0} + \mathbf{\Delta W'}$

Table 1. **Different Client Initialization Strategies.** Note that $\mathbf{\Delta W'} = \bar{\mathbf{B}}\bar{\mathbf{A}}$ is reconstructed locally. $\mathbf{B_s}$ and $\mathbf{A_s}$ represent a randomly selected client's last-round local LoRA modules. See details in Sec. 3.2.

4. LoRA-FAIR: Simple but Effective Solution

Building on the challenges outlined in previous sections, we propose a novel aggregation mechanism, LoRA-FAIR (shown in Fig. 1), designed to address both server-side aggregation bias and client-side initialization lag simultaneously. LoRA-FAIR employs a residual-based approach to refine the global model update. Rather than relying solely on the averaged LoRA matrices $\bar{\mathbf{A}}$ and $\bar{\mathbf{B}}$, LoRA-FAIR introduces a correction term for $\bar{\mathbf{B}}$, denoted as the residual LoRA module $\mathbf{\Delta B}$, to tackle both the server-side and client-side issues concurrently. Notably, LoRA-FAIR refines the global LoRA matrices at the server, without introducing additional communication or computational costs on the client side. In this section, we outline the key steps of LoRA-FAIR and demonstrate how it simultaneously addresses both Challenge 1 and Challenge 2.

To illustrate the process, consider a FL setup with K clients participating in fine-tuning at round $t+1$.

Server Side. After fine-tuning in round t, each client k sends its locally fine-tuned LoRA modules $\mathbf{A}_k$ and $\mathbf{B}_k$ back to the server. The server first aggregates these local modules to obtain the global modules $\bar{\mathbf{A}}$ and $\bar{\mathbf{B}}$ using Eq. (4). Rather than directly distributing $\bar{\mathbf{A}}$ and $\bar{\mathbf{B}}$ to the clients, LoRA-

FAIR refines the server-side aggregation by introducing a residual update $\mathbf{\Delta B}$, optimizing the following:

$$\arg\min_{\mathbf{\Delta B}} \underbrace{\mathcal{S}\left(\mathbf{\Delta W}, (\bar{\mathbf{B}} + \mathbf{\Delta B})\bar{\mathbf{A}}\right)}_{\text{correction}} + \underbrace{\lambda ||\mathbf{\Delta B}||}_{\text{regularization}}, \quad (8)$$

where $\mathbf{\Delta W}$ represents the ideal global update from Eq. (6), and $\mathcal{S}(\cdot)$ is a similarity metric (i.e. cosine similarity [8]) that measures the discrepancy between $(\bar{\mathbf{B}} + \mathbf{\Delta B})\bar{\mathbf{A}}$ and $\mathbf{\Delta W}$. The regularization weight λ balances the correction term and the regularization term. We denote the corrected averaged LoRA $\mathbf{B}$ with the residual as $\bar{\mathbf{B}}' = \bar{\mathbf{B}} + \mathbf{\Delta B}$. The application of the residual update to LoRA $\bar{\mathbf{B}}$ is validated through experiments and analysis in Sec. 5.2. The optimization problem in Eq. (8) can be approximately solved using SGD, with its computational cost detailed in the Appendix.

Upon determining $\mathbf{\Delta B}$, the server distributes $\bar{\mathbf{B}}' = \bar{\mathbf{B}} + \mathbf{\Delta B}$ and $\bar{\mathbf{A}}$ to the clients for the next training round. This approach introduces no additional communication costs. Unlike existing methods that require large-matrix SVD computations [2] or transmission of all client-stacked LoRA modules [42], LoRA-FAIR achieves computational and communication efficiency.

Client Side. Once client k receives $\bar{\mathbf{B}}'$ and $\bar{\mathbf{A}}$, it begins local fine-tuning for round $t + 1$ using its local dataset. The client initializes its LoRA module as $\mathbf{B}_k = \bar{\mathbf{B}}'$ and $\mathbf{A}_k = \bar{\mathbf{A}}$, while keeping the pre-trained model fixed.

4.1. LoRA-FAIR for Challenge 1

LoRA-FAIR tackles the server-side aggregation bias by introducing the residual correction term $\mathbf{\Delta B}$, which refines the aggregated LoRA matrix $\bar{\mathbf{B}}$ on the server. In contrast to straightforward averaging, which leads to $\bar{\mathbf{B}}\bar{\mathbf{A}}$ diverging from the ideal global update $\mathbf{\Delta W} = \sum_{k=1}^{K} p_k \mathbf{B}_k \mathbf{A}_k$, LoRA-FAIR computes a residual update that minimizes the difference between the aggregated update and the ideal. By optimizing $\mathbf{\Delta B}$, LoRA-FAIR approximates the target global model update more accurately, reducing the bias introduced by direct averaging. This correction ensures that the server-generated update better captures the interactions between local LoRA matrices, aligning $(\bar{\mathbf{B}} + \mathbf{\Delta B})\bar{\mathbf{A}}$ with the true aggregated update.

4.2. LoRA-FAIR for Challenge 2

LoRA-FAIR also addresses the client-side initialization lag by adopting the principle of **Avg-Initial**. Specifically, $\mathbf{W} \leftarrow \mathbf{W}$, $\mathbf{A} \leftarrow \bar{\mathbf{A}}$, $\mathbf{B} \leftarrow \bar{\mathbf{B}} + \Delta\mathbf{B}$. The regularization term in LoRA-FAIR's objective function prevents $\bar{\mathbf{B}}'$ from deviating excessively from $\bar{\mathbf{B}}$, thus preserving the global average information obtained from the previous round. This approach maintains continuity between rounds, allowing clients to build upon a stable and consistent initialization that incorporates both local updates and global insights. By incorporating this regularization, LoRA-FAIR fosters a smoother transition and more effective local fine-tuning.

5. Experiments

Foundation Models. This paper primarily utilizes two foundation models commonly applied in computer vision (CV) tasks. **ViT** [11]: We use a pre-trained Vision Transformer (ViT) model with 12 transformer layers as a foundation model, pre-trained on ImageNet-21k [9] (specifically, "vit base patch16 224"). **MLP-Mixer** [37]: In addition to ViT, we also use the MLP-Mixer model with 12 layers, pretrained on ImageNet-21k, specifically "mixer b16 224". We follow the step in [31] for fine-tuning and the rank of LoRA is set as 16 for experiments.

Datasets. We conduct experiments on two real-world image datasets to simulate real client data distributions. **DomainNet** [28]: DomainNet is a large multi-domain dataset containing around 600k images across 345 categories, distributed over six domains: clipart, infograph, painting, quickdraw, real, and sketch. Following the setup in [31], we use the first 100 categories. **NICO++** [14]: NICO++ is an enhanced version of NICO dataset, containing approximately 90k images across 60 categories, representing six styles: autumn, dim, grass, outdoor, rock, and water.

To emulate real client data distribution, we focus on the **feature non-IID** setting, where each client has data from different domains. In this setting, we simulate six clients, each associated with one of the six distinct domains. Additionally, we conduct experiments under the **feature and label non-IID** setting, where we consider 30 clients in total, with each domain distributed among five clients. Label non-IID conditions among the five clients from each domain are generated using a Dirichlet distribution [20] with a concentration parameter of 0.5.

Training Details. The reported results are averaged over three independent runs. We use a mini-batch size of 128 and set the number of local iterations to 2 in feature non-IID setting and 5 in feature and label nonIID setting. We set the global rounds as 50 and 30 for DomainNet and NICO++ datasets respectively. The learning rate for local training is set to 0.01, with SGD as the optimizer. In the feature non-IID experiments, all 6 clients participate in the training. For the feature and label non-IID experiments, we consider that 18 clients participate in each communication round to simulate a partial participation setting.

Baselines. To evaluate the performance of our proposed method, LoRA-FAIR, we compare it with several state-of-the-art methods in federated fine-tuning with LoRA. **1. FedIT**: FedIT [46] is the earliest approach to integrate LoRA with FedAvg. **2. FFA-LoRA**: FFA-LoRA [33] addresses server-side aggregation bias by fixing matrix $\mathbf{A}$ and fine-tuning only matrix $\mathbf{B}$. **3. FLoRA**: FLoRA [42] stacks local LoRA modules and transmits the stacked modules

			Clipart	Infograph	Painting	Quickdraw	Real	Sketch	Average
DomainNet	ViT	Centralized	85.20 ± 0.018	57.15 ± 0.037	81.48 ± 0.014	73.09 ± 0.005	90.90 ± 0.003	78.81 ± 0.036	77.77
		FFA-LoRA	81.75 ± 0.038	51.96 ± 0.058	77.51 ± 0.029	61.83 ± 0.095	88.68 ± 0.011	75.20 ± 0.050	72.82
		FedIT	84.37 ± 0.069	54.17 ± 0.127	79.67 ± 0.047	69.00 ± 0.085	89.20 ± 0.012	78.08 ± 0.035	75.75
		FLoRA	83.70 ± 0.041	53.51 ± 0.075	79.43 ± 0.046	70.09 ± 0.046	89.25 ± 0.011	77.20 ± 0.060	75.53
		FlexLoRA	85.15 ± 0.034	53.93 ± 0.132	79.82 ± 0.034	70.01 ± 0.058	89.42 ± 0.010	77.85 ± 0.048	76.02
		LoRA-FAIR	**86.25 ± 0.032**	**56.26 ± 0.062**	**80.09 ± 0.072**	**71.25 ± 0.039**	**89.52 ± 0.014**	**79.06 ± 0.061**	**77.07**
	MLP-Mixer	Centralized	74.61 ± 0.020	43.27 ± 0.019	71.54 ± 0.048	58.13 ± 0.039	85.90 ± 0.005	66.40 ± 0.048	66.64
		FFA-LoRA	69.74 ± 0.021	37.15 ± 0.045	66.43 ± 0.018	38.66 ± 0.081	80.94 ± 0.006	57.49 ± 0.047	58.40
		FedIT	74.69 ± 0.074	41.89 ± 0.089	**70.57 ± 0.029**	51.53 ± 0.030	83.25 ± 0.007	64.31 ± 0.130	64.37
		FLoRA	74.39 ± 0.024	41.33 ± 0.072	69.91 ± 0.021	53.83 ± 0.039	82.75 ± 0.008	64.08 ± 0.017	64.38
		FlexLoRA	75.11 ± 0.039	41.62 ± 0.146	70.49 ± 0.033	53.29 ± 0.051	83.41 ± 0.006	64.79 ± 0.028	64.79
		LoRA-FAIR	**75.92 ± 0.039**	**43.21 ± 0.104**	70.42 ± 0.089	**55.62 ± 0.041**	**83.43 ± 0.011**	**66.62 ± 0.039**	**65.87**
			Autumn	**Dim**	**Grass**	**Outdoor**	**Rock**	**Water**	**Average**
NICO++	ViT	Centralized	92.74 ± 0.063	89.63 ± 0.059	93.93 ± 0.024	91.07 ± 0.074	90.96 ± 0.036	90.71 ± 0.054	91.51
		FFA-LoRA	91.26 ± 0.019	88.19 ± 0.053	93.29 ± 0.012	89.84 ± 0.024	90.51 ± 0.019	88.60 ± 0.048	90.28
		FedIT	91.64 ± 0.024	88.87 ± 0.047	93.09 ± 0.015	90.05 ± 0.028	90.87 ± 0.075	88.96 ± 0.029	90.58
		FLoRA	91.48 ± 0.043	89.47 ± 0.063	93.33 ± 0.037	90.38 ± 0.040	90.83 ± 0.041	90.05 ± 0.057	90.93
		FlexLoRA	91.26 ± 0.065	88.91 ± 0.042	93.16 ± 0.013	90.41 ± 0.026	90.78 ± 0.029	89.09 ± 0.043	90.60
		LoRA-FAIR	**92.47 ± 0.032**	**89.35 ± 0.054**	**93.73 ± 0.016**	**90.56 ± 0.025**	**91.01 ± 0.060**	**90.34 ± 0.035**	**91.24**
	MLP-Mixer	Centralized	86.59 ± 0.042	82.15 ± 0.072	87.75 ± 0.012	83.67 ± 0.025	84.25 ± 0.037	82.60 ± 0.036	84.50
		FFA-LoRA	83.34 ± 0.015	76.82 ± 0.030	84.70 ± 0.010	80.14 ± 0.016	79.30 ± 0.008	75.97 ± 0.023	80.05
		FedIT	85.21 ± 0.021	79.62 ± 0.066	86.01 ± 0.010	82.44 ± 0.031	83.10 ± 0.075	78.65 ± 0.026	82.51
		FLoRA	85.10 ± 0.029	79.70 ± 0.068	86.03 ± 0.031	82.12 ± 0.055	82.24 ± 0.024	75.52 ± 0.023	82.29
		FlexLoRA	86.31 ± 0.082	79.82 ± 0.051	86.60 ± 0.012	82.77 ± 0.023	83.05 ± 0.012	79.73 ± 0.045	83.08
		LoRA-FAIR	**86.09 ± 0.037**	**81.06 ± 0.048**	**86.79 ± 0.022**	**82.71 ± 0.018**	**84.09 ± 0.033**	**80.60 ± 0.033**	**83.56**

Table 2. **Performance comparison** with baselines across different domains on DomainNet and NICO++ datasets using ViT and MLP-Mixer models in a **feature non-IID setting**. **Average** means the average accuracy across all domains. See details in Sec. 5.1.

to all participating clients to mitigate server-side aggregation bias. **4. FlexLoRA**: FlexLoRA [2] reformulates each client's local LoRA modules into a local update, sums these updates to generate a global update, and then applies SVD to update the local LoRA modules. **5. Centralized**: We also include an ideal centralized LoRA fine-tuning setting, where all data are held by a single entity, serving as a potential upper bound for comparison.

5.1. Experiments Results

Performance Comparisons. We first compare the performance of the global model across different domains under the **feature non-IID** setting. In Tab. 2, we present the results of our proposed method, LoRA-FAIR, alongside baseline methods on the DomainNet and NICO++ datasets across each domain using ViT as the foundation model. FFA-LoRA, despite reducing computation costs and addressing server-side aggregation bias by fixing the LoRA module $\mathbf{A}$, achieves the lowest performance due to limited parameter flexibility, as only $\mathbf{B}$ is fine-tuned, constraining optimization capacity. The state-of-the-art baseline method, FLoRA, which addresses server-side aggregation bias by stacking and transmitting local LoRA modules to each client, also underperforms compared to LoRA-FAIR. Although FLoRA effectively transmits the exact server aggregation update to clients, it even shows comparable performance to FedIT, a basic combination of FedAVG and LoRA, on the DomainNet dataset with ViT. These observations underscore the importance of client initialization, as discussed in Challenge 2, where the starting point of client models significantly affects federated fine-tuning results. FlexLoRA, which uses SVD to decompose summed local updates, performs better than other baselines but still falls short of LoRA-FAIR. Our proposed method which considers both server-side aggregation bias and client initialization lag, achieves superior performance in individual domain assessments and overall average accuracy. Additional experiments on both datasets using the MLP-Mixer model show similar performance trends, further supporting our findings.

We then conduct experiments under the **feature and label non-IID** setting to further validate our proposed method. In this setup, we consider a total of 30 clients, with each group of 5 clients sharing the same data domain but having non-IID label distributions (using a Dirichlet distribution with a concentration parameter of 0.5). To simulate partial participation, we increase the number of local iterations to 5 and allow 18 clients to participate in each communication round. Results in Tab. 3 indicate that, even in this more challenging setting, our proposed method, LoRA-FAIR, continues to outperform the baseline methods.

Communication Overhead. Here, we analyze the communication efficiency of our proposed method. As shown in Fig. 4, LoRA-FAIR only requires the server to distribute $\bar{\mathbf{B}}'$ and $\bar{\mathbf{A}}$ to the clients each round, incurring no additional communication cost compared to FedIT and FlexLoRA. In contrast, FLoRA, which stacks all clients' local LoRA modules and distributes them to all clients, introduces significant communication overhead. FFA-LoRA has the lowest communication cost since it keeps the LoRA module $\mathbf{A}$ fixed

			Clipart	**Infograph**	**Painting**	**Quickdraw**	**Real**	**Sketch**	**Average**
DomainNet	**ViT**	Centralized	85.20 ± 0.018	57.15 ± 0.037	81.48 ± 0.014	73.09 ± 0.005	90.90 ± 0.003	78.81 ± 0.036	77.77
		FFA-LoRA	81.75 ± 0.018	51.96 ± 0.022	77.51 ± 0.051	61.83 ± 0.025	88.68 ± 0.106	75.20 ± 0.091	72.82
		FedIT	84.08 ± 0.029	52.94 ± 0.024	79.62 ± 0.067	61.03 ± 0.019	88.94 ± 0.074	76.70 ± 0.125	73.89
		FLoRA	83.97 ± 0.042	53.57 ± 0.043	80.01 ± 0.063	62.77 ± 0.058	88.95 ± 0.060	76.30 ± 0.082	74.26
		FlexLoRA	84.29 ± 0.018	53.60 ± 0.036	79.54 ± 0.084	62.05 ± 0.079	89.23 ± 0.055	76.76 ± 0.085	74.25
		LoRA-FAIR	**84.99 ± 0.024**	**55.15 ± 0.058**	**80.51 ± 0.038**	**62.77 ± 0.059**	**89.48 ± 0.026**	**77.03 ± 0.053**	**74.99**
	MLP-Mixer	Centralized	74.61 ± 0.020	43.27 ± 0.019	71.54 ± 0.048	58.13 ± 0.039	85.90 ± 0.005	66.40 ± 0.048	66.64
		FFA-LoRA	62.91 ± 0.025	33.65 ± 0.062	64.47 ± 0.022	25.76 ± 0.024	79.85 ± 0.040	50.63 ± 0.017	52.88
		FedIT	71.53 ± 0.072	39.00 ± 0.089	68.76 ± 0.084	42.44 ± 0.060	82.34 ± 0.021	60.58 ± 0.071	60.77
		FLoRA	70.06 ± 0.052	37.26 ± 0.095	67.48 ± 0.095	41.56 ± 0.090	81.37 ± 0.016	60.01 ± 0.106	59.62
		FlexLoRA	71.58 ± 0.058	39.50 ± 0.033	68.89 ± 0.024	43.85 ± 0.091	82.39 ± 0.012	60.99 ± 0.096	61.20
		LoRA-FAIR	**72.79 ± 0.013**	**40.91 ± 0.043**	**69.49 ± 0.064**	**45.99 ± 0.073**	**82.59 ± 0.054**	**61.91 ± 0.059**	**62.28**
			Autumn	**Dim**	**Grass**	**Outdoor**	**Rock**	**Water**	**Average**
NICO++	**ViT**	Centralized	92.74 ± 0.063	89.63 ± 0.059	93.93 ± 0.024	91.07 ± 0.074	90.96 ± 0.036	90.71 ± 0.054	91.51
		FFA-LoRA	91.42 ± 0.013	86.99 ± 0.056	92.06 ± 0.045	88.83 ± 0.048	90.10 ± 0.065	87.29 ± 0.035	89.45
		FedIT	91.31 ± 0.035	86.91 ± 0.029	92.33 ± 0.008	89.01 ± 0.093	89.97 ± 0.042	87.37 ± 0.068	89.48
		FLoRA	91.28 ± 0.105	87.07 ± 0.062	92.27 ± 0.020	89.52 ± 0.064	90.04 ± 0.056	87.43 ± 0.019	89.60
		FlexLoRA	**91.81** ± 0.031	87.23 ± 0.046	92.45 ± 0.007	89.25 ± 0.021	89.79 ± 0.033	87.37 ± 0.071	89.65
		LoRA-FAIR	91.79 ± 0.040	**87.59 ± 0.060**	**92.90 ± 0.016**	**89.98 ± 0.028**	**90.39 ± 0.025**	**87.60 ± 0.018**	**90.04**
	MLP-Mixer	Centralized	86.59 ± 0.042	82.15 ± 0.072	87.75 ± 0.012	83.67 ± 0.025	84.25 ± 0.037	82.60 ± 0.036	84.50
		FFA-LoRA	80.04 ± 0.075	72.98 ± 0.048	82.07 ± 0.044	77.68 ± 0.081	76.23 ± 0.051	71.65 ± 0.068	76.78
		FedIT	81.47 ± 0.021	74.50 ± 0.092	83.64 ± 0.075	78.67 ± 0.084	78.72 ± 0.019	74.20 ± 0.081	78.53
		FLoRA	80.92 ± 0.018	74.58 ± 0.055	83.15 ± 0.027	79.21 ± 0.053	78.36 ± 0.035	74.25 ± 0.122	78.41
		FlexLoRA	82.02 ± 0.032	75.02 ± 0.015	83.33 ± 0.021	78.88 ± 0.012	78.94 ± 0.030	74.25 ± 0.098	78.73
		LoRA-FAIR	**82.46 ± 0.049**	**76.02 ± 0.027**	**83.79 ± 0.040**	**79.84 ± 0.041**	**80.16 ± 0.042**	**74.90 ± 0.091**	**79.53**

Table 3. **Performance comparison** with baselines across different domains on DomainNet and NICO++ datasets using ViT and MLP-Mixer models in a **feature and label non-IID setting**. **Average** means the average accuracy across all domains. See details in Sec. 5.1.

and only transmits $\mathbf{B}$ each round. However, as shown in Tab. 2 and Tab. 3, FFA-LoRA performs the worst across all settings. These results demonstrate that our proposed method achieves the best trade-off between communication cost and fine-tuned model performance.

5.2. Ablation Studies

Impact of Residual LoRA Module Position. In our proposed method, we apply the residual update $\mathbf{\Delta B}$ to the LoRA module $\mathbf{B}$. To examine the impact of this choice, we conduct an ablation study by adding the residual update (denoted as $\mathbf{\Delta A}$) to the LoRA module $\mathbf{A}$ or applying the residual update to both LoRA modules, $\mathbf{A}$ and $\mathbf{B}$. This study is conducted on the DomainNet dataset using ViT as the foundation model. As shown in Tab. 4, adding the residual update to the LoRA module $\mathbf{B}$ achieves slightly better performance than the others. This finding aligns with the observation in [35] that LoRA modules serve distinct functions, where $\mathbf{A}$ primarily captures general information and benefits from stability with averaged updates.

Residual	Clipart	Infograph	Painting	Quickdraw	Real	Sketch	Average
$\mathbf{\Delta A}$	84.93	54.55	80.08	71.13	89.48	78.39	76.42
$\mathbf{\Delta A}, \mathbf{\Delta B}$	84.69	54.64	78.56	68.73	88.28	78.39	75.55
$\mathbf{\Delta B}$	**86.25**	**56.26**	**80.09**	**71.25**	**89.52**	**79.06**	**77.07**

Table 4. **Performance comparison under different choices of residual LoRA modules position**. See details in Sec. 5.2.

Impact of Regularization Weight λ. In our proposed method, we optimize the objective in Eq. (8) to address both server aggregation bias and client initialization lag. Notably, we include a regularization term $\lambda||\mathbf{\Delta B}||$ to balance the similarity measure with the correction term. Here, we conduct experiments to investigate the impact of the regularization weight λ on model performance. As shown in Fig. 5, varying λ affects the performance of LoRA-FAIR, highlighting the importance of this parameter. Specifically, when $\lambda = 0$, LoRA-FAIR achieves its lowest performance.

Regularization Term $\lambda\|\|\mathbf{\Delta B}\|\|$	$\mathcal{S}(\bar{\mathbf{B}}, \bar{\mathbf{B}} + \mathbf{\Delta B})$	$\mathcal{S}\left(\mathbf{\Delta W}, (\bar{\mathbf{B}} + \mathbf{\Delta B})\bar{\mathbf{A}}\right)$	Average Accuracy
w/o ($\lambda = 0$)	0.971488	**0.999847**	73.22
w/ ($\lambda = 0.01$)	**0.999808**	0.999701	**77.07**

Table 5. **Impact of the regularization term on the similarity and the average accuracy metrics.** See details in Sec. 5.2.

This occurs because, as shown in Tab. 5, while setting $\lambda = 0$ helps address server aggregation bias by approximating $(\bar{\mathbf{B}} + \mathbf{\Delta B})\bar{\mathbf{A}}$ to $\mathbf{\Delta W}$, it reduces the similarity between $(\bar{\mathbf{B}} + \mathbf{\Delta B})$ and $\bar{\mathbf{B}}$, failing to mitigate client initialization lag. This result highlights the significant role of client initialization in influencing model performance. Additionally, with small regularization values (e.g., $\lambda = 0.01, 0.02$), performance remains stable. Thus, we recommend setting the regularization weight to a small positive value. In our experimental setup, we set the regularization weight to 0.01.

Impact of LoRA Rank. In this subsection, we investigate the impact of different LoRA ranks by conducting experiments with ranks set to $\{4,\ 8,\ 16,\ 32\}$. Notably, FLoRA fails to converge when the rank is 32, highlighting the limitations of its approach, which involves direct updates to the pre-trained model. We observe that increasing the LoRA rank does not necessarily lead to better final performance,

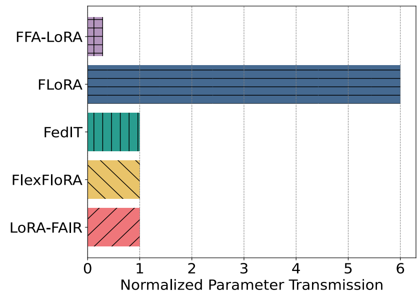

Figure 4. **Communication cost comparison.** LoRA-FAIR matches the communication cost of FedIT and FlexLoRA and avoids FLoRA's high overhead. Details in Sec. 5.1.

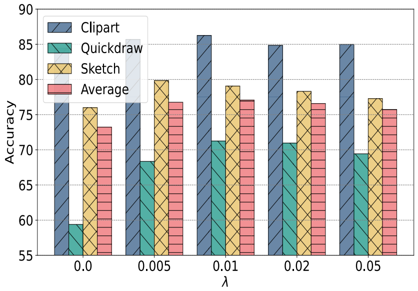

Figure 5. **Impact of Regularization Weight** λ**.** With $\lambda = 0$, LoRA-FAIR results in the lowest performance, underscoring the importance of this term. Details in Sec. 5.2.

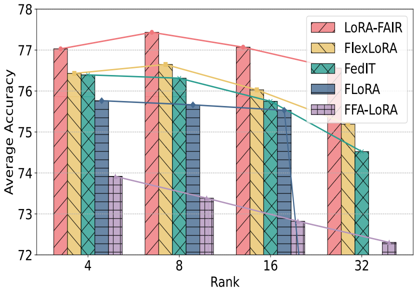

Figure 6. **Impact of LoRA Rank.** LoRA-FAIR outperforms baselines across ranks {4, 8, 16, 32}, with higher ranks not always improving performance, consistent with [7].

consistent with findings from previous studies [7]. Additionally, the results in Fig. 6 demonstrate that our proposed method consistently outperforms baselines across all rank settings, validating its effectiveness.

Additional experiment results on convergence performance, adaptation to clients with heterogeneous LoRA ranks, server-side computational overhead, and the limitation of FLoRA can be found in the Appendix.

6. Related Work

Parameter-Efficient Fine-Tuning. The increasing size of foundation models makes full fine-tuning computationally and storage-intensive. To address these challenges, Parameter-Efficient Fine-Tuning (PEFT) methods [10, 12, 13, 22] have been proposed to reduce the number of trainable parameters. PEFT techniques introduce a limited set of additional trainable parameters to enhance model performance while keeping most pre-trained parameters frozen. Some approaches, such as [15], add trainable parameters called adapters to each layer of the pre-trained network, updating only the adapters during fine-tuning. Other approaches, like [5], focus on fine-tuning only the bias terms of the pre-trained model. Techniques such as prefix-tuning [21] and prompt-tuning [19] add trainable dimensions to the input or hidden layers of the network. Among PEFT methods, a key approach is LoRA [16], which uses low-rank matrices to approximate the pre-trained weight matrix, updating only the low-rank matrices. In this paper, we focus on LoRA due to its demonstrated efficiency, achieving comparable performance to full-parameter fine-tuning.

Federated Learning. FedAvg [26], the foundational work in FL, demonstrates the advantages of this approach in terms of privacy and communication efficiency by aggregating local model parameters to train a shared global model. Numerous FL studies [3, 23, 24, 26, 29, 39–41, 44] have addressed various challenges within FL settings. For example, several works explore the impact of different initialization strategies on model performance. [34] shows that initializing with pre-trained weights can enhance the stability of FedAvg's global aggregation, while [36] confirms the effectiveness of using a pre-trained model as an initial starting point. However, these methods primarily focus on smaller models and do not extend to foundation models or incorporate parameter-efficient fine-tuning; instead, they adhere to conventional FL training practices.

Federated Fine-Tuning. Several studies [2, 7, 17, 33, 42, 43] have explored federated fine-tuning approaches. For example, Kuang et al. [17] proposes federated fine-tuning with all parameters updated, while Sun et al. [32] introduces federated fine-tuning with PEFT using prefix-tuning. A closely related area to our work involves federated fine-tuning using LoRA. Zhang et al. [46, 47] apply LoRA in a federated context; however, these methods overlook potential server aggregation bias. Several subsequent works have been proposed: FFA-LoRA [33] freezes the non-zero initialized low-rank matrices and updates only the zero-initialized matrices, FlexLoRA [2] uses SVD to redistribute weights, and FLoRA [42] stacks local LoRA modules and transmits them to each client. However, these methods do not address client initialization lag. A more detailed discussion of related federated LoRA works can be found in Appendix Sec. 10.

7. Conclusion

In this work, we proposed LoRA-FAIR to address the key challenges of server-side aggregation bias and client-side initialization lag in federated fine-tuning with LoRA. LoRA-FAIR approximates an ideal solution by maintaining shared average information while ensuring dynamic server-side adjustments. Our experiments on large-scale datasets demonstrated its superior performance over state-of-the-art methods. Future work will explore extending LoRA-FAIR beyond computer vision datasets and adapting it for scenarios where clients use different LoRA ranks to enhance its applicability in diverse federated learning environments.

8. Acknowledgments

The work of J. Bian, L. Wang and J. Xu is partially supported by NSF under grants 2505381 and 2515982. The work of L. Zhang is partially supported by NSF under grant 2348279.

References

[1] Josh Achiam, Steven Adler, Sandhini Agarwal, Lama Ahmad, Ilge Akkaya, Florencia Leoni Aleman, Diogo Almeida, Janko Altenschmidt, Sam Altman, Shyamal Anadkat, et al. Gpt-4 technical report. *arXiv preprint arXiv:2303.08774*, 2023. 1

[2] Jiamu Bai, Daoyuan Chen, Bingchen Qian, Liuyi Yao, and Yaliang Li. Federated fine-tuning of large language models under heterogeneous language tasks and client resources. *arXiv e-prints*, pages arXiv–2402, 2024. 5, 6, 8, 2

[3] Jieming Bian, Lei Wang, and Jie Xu. Prioritizing modalities: Flexible importance scheduling in federated multimodal learning. *arXiv preprint arXiv:2408.06549*, 2024. 8

[4] Rishi Bommasani, Drew A Hudson, Ehsan Adeli, Russ Altman, Simran Arora, Sydney von Arx, Michael S Bernstein, Jeannette Bohg, Antoine Bosselut, Emma Brunskill, et al. On the opportunities and risks of foundation models. *arXiv preprint arXiv:2108.07258*, 2021. 1

[5] Han Cai, Chuang Gan, Ligeng Zhu, and Song Han. Tinytl: Reduce memory, not parameters for efficient on-device learning. *Advances in Neural Information Processing Systems*, 33:11285–11297, 2020. 8

[6] Dengsheng Chen, Jie Hu, Vince Junkai Tan, Xiaoming Wei, and Enhua Wu. Elastic aggregation for federated optimization. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pages 12187–12197, 2023. 2

[7] Yae Jee Cho, Luyang Liu, Zheng Xu, Aldi Fahrezi, Matt Barnes, and Gauri Joshi. Heterogeneous lora for federated fine-tuning of on-device foundation models. In *International Workshop on Federated Learning in the Age of Foundation Models in Conjunction with NeurIPS 2023*, 2023. 8, 1

[8] Gobinda G Chowdhury. *Introduction to modern information retrieval*. Facet publishing, 2010. 5

[9] Jia Deng, Wei Dong, Richard Socher, Li-Jia Li, Kai Li, and Li Fei-Fei. Imagenet: A large-scale hierarchical image database. In *2009 IEEE conference on computer vision and pattern recognition*, pages 248–255. Ieee, 2009. 5

[10] Ning Ding, Yujia Qin, Guang Yang, Fuchao Wei, Zonghan Yang, Yusheng Su, Shengding Hu, Yulin Chen, Chi-Min Chan, Weize Chen, et al. Parameter-efficient fine-tuning of large-scale pre-trained language models. *Nature Machine Intelligence*, 5(3):220–235, 2023. 8

[11] Alexey Dosovitskiy. An image is worth 16x16 words: Transformers for image recognition at scale. *arXiv preprint arXiv:2010.11929*, 2020. 2, 5

[12] Zihao Fu, Haoran Yang, Anthony Man-Cho So, Wai Lam, Lidong Bing, and Nigel Collier. On the effectiveness of parameter-efficient fine-tuning. In *Proceedings of the AAAI conference on artificial intelligence*, pages 12799–12807, 2023. 8

[13] Zeyu Han, Chao Gao, Jinyang Liu, Jeff Zhang, and Sai Qian Zhang. Parameter-efficient fine-tuning for large models: A comprehensive survey. *arXiv preprint arXiv:2403.14608*, 2024. 1, 8

[14] Yue He, Zheyan Shen, and Peng Cui. Towards non-iid image classification: A dataset and baselines. *Pattern Recognition*, 110:107383, 2021. 5

[15] Neil Houlsby, Andrei Giurgiu, Stanislaw Jastrzebski, Bruna Morrone, Quentin De Laroussilhe, Andrea Gesmundo, Mona Attariyan, and Sylvain Gelly. Parameter-efficient transfer learning for nlp. In *International conference on machine learning*, pages 2790–2799. PMLR, 2019. 8

[16] Edward J Hu, Yelong Shen, Phillip Wallis, Zeyuan Allen-Zhu, Yuanzhi Li, Shean Wang, Lu Wang, and Weizhu Chen. Lora: Low-rank adaptation of large language models. *arXiv preprint arXiv:2106.09685*, 2021. 1, 8

[17] Weirui Kuang, Bingchen Qian, Zitao Li, Daoyuan Chen, Dawei Gao, Xuchen Pan, Yuexiang Xie, Yaliang Li, Bolin Ding, and Jingren Zhou. Federatedscope-llm: A comprehensive package for fine-tuning large language models in federated learning. In *Proceedings of the 30th ACM SIGKDD Conference on Knowledge Discovery and Data Mining*, pages 5260–5271, 2024. 8

[18] Sunwoo Lee, Tuo Zhang, and A Salman Avestimehr. Layer-wise adaptive model aggregation for scalable federated learning. In *Proceedings of the AAAI Conference on Artificial Intelligence*, pages 8491–8499, 2023. 2

[19] Brian Lester, Rami Al-Rfou, and Noah Constant. The power of scale for parameter-efficient prompt tuning. *arXiv preprint arXiv:2104.08691*, 2021. 8

[20] Qinbin Li, Yiqun Diao, Quan Chen, and Bingsheng He. Federated learning on non-iid data silos: An experimental study. In *2022 IEEE 38th international conference on data engineering (ICDE)*, pages 965–978. IEEE, 2022. 5

[21] Xiang Lisa Li and Percy Liang. Prefix-tuning: Optimizing continuous prompts for generation. *arXiv preprint arXiv:2101.00190*, 2021. 8

[22] Haokun Liu, Derek Tam, Mohammed Muqeeth, Jay Mohta, Tenghao Huang, Mohit Bansal, and Colin A Raffel. Few-shot parameter-efficient fine-tuning is better and cheaper than in-context learning. *Advances in Neural Information Processing Systems*, 35:1950–1965, 2022. 8

[23] Junkang Liu, Fanhua Shang, Yuanyuan Liu, Hongying Liu, Yuangang Li, and YunXiang Gong. Fedbcgd: Communication-efficient accelerated block coordinate gradient descent for federated learning. In *Proceedings of the 32nd ACM International Conference on Multimedia*, pages 2955–2963, 2024. 8

[24] Junkang Liu, Yuanyuan Liu, Fanhua Shang, Hongying Liu, Jin Liu, and Wei Feng. Improving generalization in federated learning with highly heterogeneous data via momentum-based stochastic controlled weight averaging. In *Forty-second International Conference on Machine Learning*, 2025. 8

[25] Xiaosong Ma, Jie Zhang, Song Guo, and Wenchao Xu. Layer-wised model aggregation for personalized federated

learning. In *Proceedings of the IEEE/CVF conference on computer vision and pattern recognition*, pages 10092–10101, 2022. 2

[26] Brendan McMahan, Eider Moore, Daniel Ramage, Seth Hampson, and Blaise Aguera y Arcas. Communication-efficient learning of deep networks from decentralized data. In *Artificial intelligence and statistics*, pages 1273–1282. PMLR, 2017. 2, 8

[27] Fanxu Meng, Zhaohui Wang, and Muhan Zhang. Pissa: Principal singular values and singular vectors adaptation of large language models. *Advances in Neural Information Processing Systems*, 37:121038–121072, 2025. 4

[28] Xingchao Peng, Qinxun Bai, Xide Xia, Zijun Huang, Kate Saenko, and Bo Wang. Moment matching for multi-source domain adaptation. In *Proceedings of the IEEE/CVF international conference on computer vision*, pages 1406–1415, 2019. 5

[29] Yuanzhe Peng, Jieming Bian, and Jie Xu. Fedmm: Federated multi-modal learning with modality heterogeneity in computational pathology. In *ICASSP 2024-2024 IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP)*, pages 1696–1700. IEEE, 2024. 8

[30] Yasar Abbas Ur Rehman, Yan Gao, Pedro Porto Buarque De Gusmão, Mina Alibeigi, Jiajun Shen, and Nicholas D Lane. L-dawa: Layer-wise divergence aware weight aggregation in federated self-supervised visual representation learning. In *Proceedings of the IEEE/CVF international conference on computer vision*, pages 16464–16473, 2023. 2

[31] Shangchao Su, Bin Li, and Xiangyang Xue. Fedra: A random allocation strategy for federated tuning to unleash the power of heterogeneous clients. *arXiv preprint arXiv:2311.11227*, 2023. 5

[32] Guangyu Sun, Umar Khalid, Matias Mendieta, Taojiannan Yang, and Chen Chen. Conquering the communication constraints to enable large pre-trained models in federated learning. *arXiv preprint arXiv:2210.01708*, 2022. 8

[33] Youbang Sun, Zitao Li, Yaliang Li, and Bolin Ding. Improving lora in privacy-preserving federated learning. *arXiv preprint arXiv:2403.12313*, 2024. 3, 5, 8, 2

[34] Yue Tan, Guodong Long, Jie Ma, Lu Liu, Tianyi Zhou, and Jing Jiang. Federated learning from pre-trained models: A contrastive learning approach. *Advances in neural information processing systems*, 35:19332–19344, 2022. 8

[35] Chunlin Tian, Zhan Shi, Zhijiang Guo, Li Li, and Cheng-Zhong Xu. Hydralora: An asymmetric lora architecture for efficient fine-tuning. *Advances in Neural Information Processing Systems*, 37:9565–9584, 2025. 7

[36] Yuanyishu Tian, Yao Wan, Lingjuan Lyu, Dezhong Yao, Hai Jin, and Lichao Sun. Fedbert: When federated learning meets pre-training. *ACM Transactions on Intelligent Systems and Technology (TIST)*, 13(4):1–26, 2022. 8

[37] Ilya O Tolstikhin, Neil Houlsby, Alexander Kolesnikov, Lucas Beyer, Xiaohua Zhai, Thomas Unterthiner, Jessica Yung, Andreas Steiner, Daniel Keysers, Jakob Uszkoreit, et al. Mlp-mixer: An all-mlp architecture for vision. *Advances in neural information processing systems*, 34:24261–24272, 2021. 2, 5

[38] Hugo Touvron, Thibaut Lavril, Gautier Izacard, Xavier Martinet, Marie-Anne Lachaux, Timothée Lacroix, Baptiste Rozière, Naman Goyal, Eric Hambro, Faisal Azhar, et al. Llama: Open and efficient foundation language models. *arXiv preprint arXiv:2302.13971*, 2023. 1

[39] Lei Wang, Jieming Bian, and Jie Xu. Federated learning with instance-dependent noisy label. In *ICASSP 2024-2024 IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP)*, pages 8916–8920. IEEE, 2024. 8

[40] Lei Wang, Jieming Bian, Letian Zhang, Chen Chen, and Jie Xu. Taming cross-domain representation variance in federated prototype learning with heterogeneous data domains. *arXiv preprint arXiv:2403.09048*, 2024.

[41] Yuan Wang, Huazhu Fu, Renuga Kanagavelu, Qingsong Wei, Yong Liu, and Rick Siow Mong Goh. An aggregation-free federated learning for tackling data heterogeneity. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pages 26233–26242, 2024. 8

[42] Ziyao Wang, Zheyu Shen, Yexiao He, Guoheng Sun, Hongyi Wang, Lingjuan Lyu, and Ang Li. Flora: Federated fine-tuning large language models with heterogeneous low-rank adaptations. *arXiv preprint arXiv:2409.05976*, 2024. 2, 3, 5, 8

[43] Feijie Wu, Zitao Li, Yaliang Li, Bolin Ding, and Jing Gao. Fedbiot: Llm local fine-tuning in federated learning without full model. In *Proceedings of the 30th ACM SIGKDD Conference on Knowledge Discovery and Data Mining*, pages 3345–3355, 2024. 8

[44] Zhiqin Yang, Yonggang Zhang, Yu Zheng, Xinmei Tian, Hao Peng, Tongliang Liu, and Bo Han. Fedfed: Feature distillation against data heterogeneity in federated learning. *Advances in Neural Information Processing Systems*, 36, 2024. 8

[45] Aohan Zeng, Xiao Liu, Zhengxiao Du, Zihan Wang, Hanyu Lai, Ming Ding, Zhuoyi Yang, Yifan Xu, Wendi Zheng, Xiao Xia, et al. Glm-130b: An open bilingual pre-trained model. *arXiv preprint arXiv:2210.02414*, 2022. 1

[46] Jianyi Zhang, Saeed Vahidian, Martin Kuo, Chunyuan Li, Ruiyi Zhang, Tong Yu, Guoyin Wang, and Yiran Chen. Towards building the federatedgpt: Federated instruction tuning. In *ICASSP 2024-2024 IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP)*, pages 6915–6919. IEEE, 2024. 2, 3, 5, 8

[47] Zhuo Zhang, Yuanhang Yang, Yong Dai, Qifan Wang, Yue Yu, Lizhen Qu, and Zenglin Xu. Fedpetuning: When federated learning meets the parameter-efficient tuning methods of pre-trained language models. In *Annual Meeting of the Association of Computational Linguistics 2023*, pages 9963–9977. Association for Computational Linguistics (ACL), 2023. 8

[48] Yue Zhao, Meng Li, Liangzhen Lai, Naveen Suda, Damon Civin, and Vikas Chandra. Federated learning with non-iid data. *arXiv preprint arXiv:1806.00582*, 2018. 4, 2

[49] Ce Zhou, Qian Li, Chen Li, Jun Yu, Yixin Liu, Guangjing Wang, Kai Zhang, Cheng Ji, Qiben Yan, Lifang He, et al. A comprehensive survey on pretrained foundation models: A history from bert to chatgpt. *arXiv preprint arXiv:2302.09419*, 2023. 1