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

Jieming Bian *, Lei Wang *, Jie Xu
University of Florida
Gainesville, FL 32611
jieming.bian@ufl.edu, leiwang1@ufl.edu, jie.xu@ufl.edu

Letian Zhang
Middle Tennessee State University
Murfreesboro, TN 37132
letian.zhang@mtsu.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 LoRA Aggregation Bias****, where server-side averaging of LoRA matrices diverges from the ideal global update, and the* ***Client-Side LoRA Initialization Drift****, 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 while keeping the original LoRA modules, 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, 30, 37, 39] 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

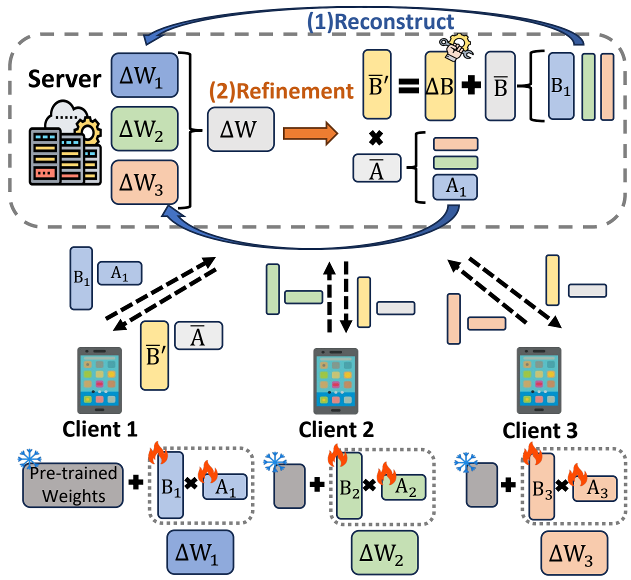


Figure 1. **Illustration of LoRA-FAIR.** Instead of directly averaging the local LoRA modules $\mathbf{A}_k$ and $\mathbf{B}_k$ collected from each client $k$ on the server side and sending the averaged LoRA modules $\bar{\mathbf{A}}$ and $\bar{\mathbf{B}}$ back to clients, LoRA-FAIR reconstructs the ideal global update $\boldsymbol{\Delta}\mathbf{W}$ using Eq. (7), finds the residual LoRA module $\boldsymbol{\Delta}\mathbf{B}$ using Eq. (8), and replaces $\bar{\mathbf{B}}$ with the corrected LoRA modules $\bar{\mathbf{B}}' = \bar{\mathbf{B}} + \boldsymbol{\Delta}\mathbf{B}$. See details in Sec. 4.

data. These models excel across diverse applications, including image generation from prompts, language translation, mathematical problem-solving, and natural language conversation, among others [39]. 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, a range of parameter-efficient fine-tuning (PEFT) methods [12] has been proposed. One of the most important PEFT approaches is low-rank adaptation (LoRA) [15], which significantly reduces the number of trainable parameters by introducing low-rank matrices

*The first two authors contributed equally to this work, and their names are listed in alphabetical order by last name.

into the model.

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. 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) [21] 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 federated learning 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, the **Challenge 2: Client-Side LoRA Initialization Drift** highlights the importance of starting each training round with LoRA modules that incorporate the previous round's averaged global information. Existing FL methods for fine-tuning fail to consider these two key points simultaneously. While some methods, such as FLoRA [34], 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 [38]).

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 $\bar{\mathbf{B}}$ 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 $\bar{\mathbf{B}}$ remains close to its original averaged value, thereby preserving the average information from $\mathbf{B}$ collected from each client. This modification not only maintains the global average insights embedded in $\bar{\mathbf{B}}$ but also allows $\bar{\mathbf{B}}$ to adjust dynamically to approximate the ideal global update $\mathbf{\Delta W}$. 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 [10] and MLP-Mixer [29], across various large-scale datasets. The results demonstrate that 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 $\mathbf{A} \in \mathbb{R}^{d\times r}$ and $\mathbf{B} \in \mathbb{R}^{r\times l}$, where $r \ll \min(d, l)$. Thus, the model update is expressed as:

$$\mathbf{\Delta W} = \mathbf{BA}, \tag{1}$$

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:

$$\mathbf{W} = \mathbf{W}_0 + \mathbf{\Delta W} = \mathbf{W}_0 + \mathbf{BA}. \tag{2}$$

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:

$$\mathbf{\Delta W} = \sum_{k=1}^{K} p_k \mathbf{\Delta W}_k, \tag{3}$$

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 $\mathbf{\Delta W}_k$ denotes the local update from client $k$. To start the next round of local training, the server uses the global update $\mathbf{\Delta W}$ 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:

$$\begin{aligned}
&\mathbf{W}_{k,0} = \mathbf{W}_0 + \mathbf{\Delta W};\\
&\mathbf{W}_{k,e+1} = \mathbf{W}_{k,e} - \eta g_{k,e}, \quad e = 0, \dots, E-1;\\
&\mathbf{\Delta W}_k = -\sum_{e=0}^{E-1} \eta g_{k,e},
\end{aligned} \tag{4}$$

where $\eta$ 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 [38]. 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:

$$\bar{\mathbf{A}} = \sum_{k=1}^{K} p_k \mathbf{A}_k, \quad \bar{\mathbf{B}} = \sum_{k=1}^{K} p_k \mathbf{B}_k, \tag{5}$$

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:

$$\mathbf{\Delta W}' = \bar{\mathbf{B}}\bar{\mathbf{A}} = \left(\sum_{k=1}^{K} p_k \mathbf{B}_k\right)\left(\sum_{k=1}^{K} p_k \mathbf{A}_k\right). \tag{6}$$

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:

$$\mathbf{\Delta W} = \sum_{k=1}^{K} p_k \mathbf{\Delta W}_k = \sum_{k=1}^{K} p_k \mathbf{B}_k \mathbf{A}_k \neq \mathbf{\Delta W}'. \tag{7}$$

This discrepancy, termed **Server-Side Aggregation Bias**, occurs because the approximate global update $\mathbf{\Delta W}'$ fails to accurately capture the ideal global update $\mathbf{\Delta W}$. 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 $\mathbf{\Delta W}'$ and $\mathbf{\Delta W}$ respectively. Although **AvgToMul** reduces communication costs by only

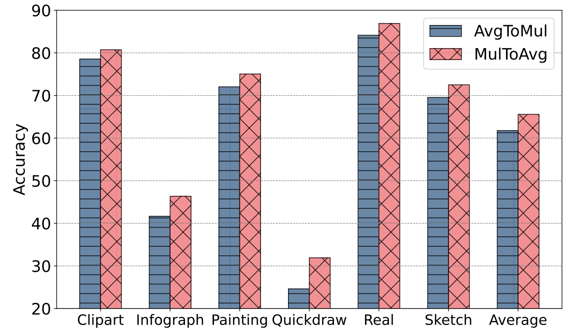


Figure 2. **Comparison of two aggregation strategies: AvgToMul and MulToAvg. AvgToMul** averages the LoRA matrices $\mathbf{A}_k$ and $\mathbf{B}_k$ from clients, then multiplies the averages to obtain the approximate global update $\mathbf{\Delta W}'$ using Eq. (6). **MulToAvg** first multiplies each client's matrices (yielding $\mathbf{B}_k\mathbf{A}_k$) and then averages these products for the true global update $\mathbf{\Delta W}$ using Eq. (7). While **AvgToMul** is communication-efficient, **MulToAvg** better captures the intended global model update. See details in Sec. 3.1.

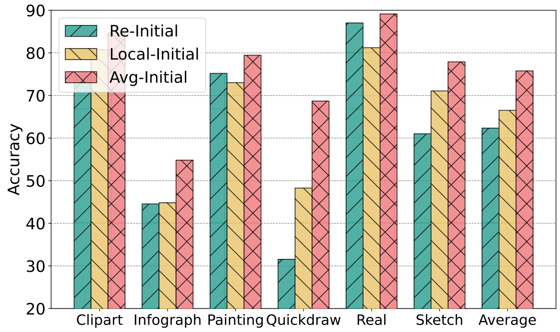


Figure 3. **Comparison of three initialization strategies: Avg-Initial, Re-Initial, Local-Initial.** The **Avg-Initial** method is the most effective as it balances continuity and unification across clients, reducing initialization drift and promoting better convergence. For more details, refer to Sec.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

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.

### 3.2. Challenge 2: Client-Side Initialization Drift

To mitigate server-side aggregation bias, FFA-LoRA [26] was proposed, freezing the non-zero initialized low-rank matrix $\mathbf{A}$ and updating only the zero-initialized matrix $\mathbf{B}$. However, this approach slows fine-tuning and limits performance due to fewer trainable parameters. A more recent method, FLoRA [34], stacks local LoRA modules from all clients and transmits the stacked LoRA modules back to each client to reconstruct global updates, which are then added directly to each local pre-trained model while reinitializing local LoRA modules for the next training round. Although FLoRA addresses **Challenge 1** effectively, it incurs high communication cost proportional to the number of clients and poses privacy concerns, as it requires distributing all clients' LoRA modules to each client rather than only the averaged modules, as in FedIT.

Furthermore, FLoRA's reinitialization strategy for LoRA modules (typically randomizing $\mathbf{A}$ with Gaussian distribution and setting $\mathbf{B}$ to zero) introduces **Client-Side Initialization Drift**. The gradient of $\mathbf{A}$ depends on the current state of $\mathbf{B}$, and the gradient of $\mathbf{B}$ similarly depends on $\mathbf{A}$. This interdependence implies that updates to one matrix are influenced by the configuration of the other, affecting the fine-tuning process. Frequent reinitialization produces small gradient updates, leading to inefficient training and potentially suboptimal performance.

To understand the effects of different client-side initialization methods on model performance, we evaluate three strategies in an FL setup with 6 clients, each assigned a unique domain from the DomainNet dataset. To isolate the impact of client-side initialization, we ensure that in all methods the server performs the same aggregation method, $\Delta W'$, and each client adjusts the pre-trained model part accordingly. The three strategies are as follows: **1. Avg-Initial:** The averaged LoRA modules aggregated from all clients are used as the initialization for the next round, maintaining parameter continuity across rounds and achieving the best performance. **2. Re-Initial:** LoRA modules are reinitialized at each round (with random Gaussian values for $\mathbf{A}$ and zeros for $\mathbf{B}$). This approach helps prevent overfitting to client-specific data but limits gradient updates, slowing convergence. **3. Local-Initial:** A randomly selected client's last-round local LoRA modules are used as the starting LoRA point for all clients. As shown in Fig. 3, the **Avg-Initial** method is the most effective, as it balances continuity and unification across clients, reducing initialization drift and promoting stable convergence. By averaging local LoRA modules, this method captures a representative update, smoothing extreme deviations and fostering a consistent training path. In summary, both server-side aggregation bias and client-side initialization drift are substantial challenges when combining LoRA with federated learning. Addressing these issues is critical for effective foundation model fine-tuning in FL settings, ensuring performance and convergence stability.

## 4. LoRA-FAIR: A 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 drift 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. (5). 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:

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

where $\mathbf{\Delta W}$ represents the ideal global update from Eq. (7), and $\mathcal{S}(\cdot)$ is a similarity metric (cosine similarity [7] in our experiments) that measures the discrepancy between $(\bar{\mathbf{B}} + \mathbf{\Delta B})\bar{\mathbf{A}}$ and $\mathbf{\Delta W}$. We denote the corrected averaged LoRA $\mathbf{B}$ with the residual as $\bar{\mathbf{B}}' = \bar{\mathbf{B}} + \mathbf{\Delta B}$. The regularization weight $\lambda$ balances the correction term and the regularization term. Rather than directly finding an analytical solution, we can use methods such as SGD to obtain a solution, which is computationally feasible and efficient.

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 [34], 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 uses $\bar{\mathbf{B}}'$ as the initialization for its LoRA module $\mathbf{B}_k$ and $\bar{\mathbf{A}}$ as the initialization for $\mathbf{A}_k$.

### 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 drift by distributing the refined LoRA matrices $\bar{\mathbf{B}}'$ and $\bar{\mathbf{A}}$ to each client, ensuring that each round begins with globally-informed parameters. The regularization term in LoRA-FAIR's objective function prevents $\mathbf{\Delta B}$ 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 ensures that the refined matrix $\bar{\mathbf{B}}'$ stays close to the globally averaged $\bar{\mathbf{B}}$, fostering a smoother transition and more effective local fine-tuning across rounds.

## 5. Experiments

The reported results are averaged over three independent runs. Due to space limitations, we only present the average results.

**Foundation Models.** This paper primarily utilizes two foundation models commonly applied in computer vision (CV) tasks. **ViT** [10]: We use a pre-trained Vision Transformer (ViT) model with 12 transformer layers as a foundation model, pre-trained on ImageNet-21k [8] (specifically, "vit base patch16 224"). **MLP-Mixer** [29]: In addition to ViT, we also use the MLP-Mixer model with 12 layers, pre-trained on ImageNet-21k, specifically "mixer b16 224". We follow the step in [24] 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** [22]: 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 [24], we use the first 100 categories. **NICO++** [13]: 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 [18] with a concentration parameter of 0.5.

**Training Details.** 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 [38] is the earliest approach to integrate LoRA with FedAvg. **2. FFA-LoRA**: FFA-LoRA [26] addresses server-side aggregation bias by fixing matrix $\mathbf{A}$ and

| | | | Clipart | Infograph | Painting | Quickdraw | Real | Sketch | Average |
|---|---|---|---|---|---|---|---|---|---|
| DomainNet | ViT | FFA-LoRA | 81.75 | 51.96 | 77.51 | 61.83 | 88.68 | 75.20 | 72.82 |
| | | FedIT | 84.37 | 54.17 | 79.67 | 69.00 | 89.20 | 78.08 | 75.75 |
| | | FLoRA | 83.70 | 53.51 | 79.43 | 70.09 | 89.25 | 77.20 | 75.53 |
| | | FlexLoRA | 85.15 | 53.93 | 79.82 | 70.01 | 89.42 | 77.85 | 76.02 |
| | | **LoRA-FAIR** | **86.25** | **56.26** | **80.09** | **71.25** | **89.52** | **79.06** | **77.07** |
| | MLP-Mixer | FFA-LoRA | 69.74 | 37.15 | 66.43 | 38.66 | 80.94 | 57.49 | 58.40 |
| | | FedIT | 74.69 | 41.89 | 70.57 | 51.53 | 83.25 | 64.31 | 64.37 |
| | | FLoRA | 74.39 | 41.33 | 69.91 | 53.83 | 82.75 | 64.08 | 64.38 |
| | | FlexLoRA | 75.11 | 41.62 | 70.49 | 53.29 | 83.41 | 64.79 | 64.79 |
| | | **LoRA-FAIR** | **75.92** | **43.21** | **70.42** | **55.62** | **83.43** | **66.62** | **65.87** |
| | | | **Autumn** | **Dim** | **Grass** | **Outdoor** | **Rock** | **Water** | **Average** |
| NICO++ | ViT | FFA-LoRA | 91.26 | 88.19 | 93.29 | 89.84 | 90.51 | 88.60 | 90.28 |
| | | FedIT | 91.64 | 88.87 | 93.09 | 90.05 | 90.87 | 88.96 | 90.58 |
| | | FLoRA | 91.48 | 89.47 | 93.33 | 90.38 | 90.83 | 90.05 | 90.93 |
| | | FlexLoRA | 91.26 | 88.91 | 93.16 | 90.41 | 90.78 | 89.09 | 90.60 |
| | | **LoRA-FAIR** | **92.47** | **89.35** | **93.73** | **90.56** | **91.01** | **90.34** | **91.24** |
| | MLP-Mixer | FFA-LoRA | 83.34 | 76.82 | 84.70 | 80.14 | 79.30 | 75.97 | 80.05 |
| | | FedIT | 85.21 | 79.62 | 86.01 | 82.44 | 83.10 | 78.65 | 82.51 |
| | | FLoRA | 85.10 | 79.70 | 86.03 | 82.12 | 82.24 | 75.52 | 82.29 |
| | | FlexLoRA | 86.31 | 79.82 | 86.60 | 82.77 | 83.05 | 79.73 | 83.08 |
| | | **LoRA-FAIR** | **86.09** | **81.06** | **86.79** | **82.71** | **84.09** | **80.60** | **83.56** |

Table 1. **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.

fine-tuning only matrix $\mathbf{B}$. **3. FLoRA**: FLoRA [34] stacks local LoRA modules and transmits the stacked modules 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.

## 5.1. Experiments Results

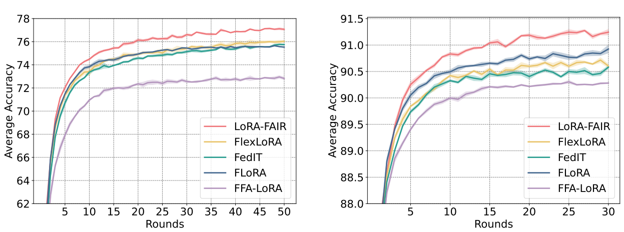


Figure 4. **Comparison of average accuracy** across training rounds on DomainNet (**left**) and NICO++ (**right**) datasets using the ViT model. The shaded area indicates the variance across multiple runs. For more details, refer to Sec. 5.1.

**Performance Comparisons.** We first compare the performance of the global model across different domains under the **feature non-IID** setting. In Tab. 1 and Fig. 4, 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, LoRA-FAIR, which considers both server-side aggregation bias and client initialization drift, 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. 2 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. 5, 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 and only transmits $\mathbf{B}$ each round. However, as shown in Tab. 1 and Tab. 2, 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 investigate the effect of this choice, we conduct an ablation study by adding the residual update (represented as $\mathbf{\Delta A}$) to another LoRA module $\mathbf{A}$. This study is performed on the DomainNet dataset using ViT as the foundation model. As shown in Tab. 3, adding the residual update to LoRA module $\mathbf{B}$ achieves slightly better performance compared to adding it to $\mathbf{A}$. Additionally, it is worth noting that applying the residual update to $\mathbf{A}$ still outperforms baseline methods, as shown in Tab. 1.

**Impact of Regularization Weight $\lambda$.** In our proposed method, we optimize the objective in Eq. (8) to address both server aggregation bias and client initialization drift. 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 $\lambda$ on model performance. As shown in Fig. 6, varying $\lambda$ affects the performance of LoRA-FAIR, highlighting the importance of this parameter. Specifically, when $\lambda = 0$, LoRA-FAIR achieves its lowest performance.

This occurs because, as shown in Tab. 4, 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 drift. 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, consistent with findings from previous studies [6]. Additionally, the results in Fig. 7 demonstrate that our proposed method consistently outperforms baselines across all rank settings, validating its effectiveness.

## 6. Related Work

**Parameter-Efficient Fine-Tuning.** The increasing size of foundation models makes full-parameter fine-tuning computationally and storage-intensive. To address these challenges, Parameter-Efficient Fine-Tuning (PEFT) methods [9, 11, 12, 20] 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 [14], 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 [19] and prompt-tuning [17] add trainable dimensions to the input or hidden layers of the network.

Among PEFT methods, a key approach is LoRA [15], which uses low-rank matrices to approximate the pre-trained weight matrix, updating only the low-rank matrices. In this paper, we utilize LoRA as our PEFT method due to its demonstrated efficiency, achieving comparable performance to full-parameter fine-tuning while modifying fewer than 5% of the parameters.

**Federated Learning.** FedAvg [21], 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, 21, 23, 31–33, 36] have addressed various challenges within FL settings. For example, several works explore the impact of different initialization strategies on model performance. [27] shows that initializing with pre-trained weights can enhance the stability of FedAvg's global aggregation, while [28] 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, 6, 16, 26, 34, 35] have explored federated fine-tuning approaches. For example, Kuang et al. [16] proposes federated fine-tuning with all parameters updated, while Sun et al. [25] introduces federated fine-tuning with PEFT using prefix-tuning.

| | | | Clipart | Infograph | Painting | Quickdraw | Real | Sketch | Average |
|---|---|---|---|---|---|---|---|---|---|
| DomainNet | ViT | FFA-LoRA | 81.75 | 51.96 | 77.51 | 61.83 | 88.68 | 75.20 | 72.82 |
| | | FedIT | 84.08 | 52.94 | 79.62 | 61.03 | 88.94 | 76.70 | 73.89 |
| | | FLoRA | 83.97 | 53.57 | 80.01 | 62.77 | 88.95 | 76.30 | 74.26 |
| | | FlexLoRA | 84.29 | 53.60 | 79.54 | 62.05 | 89.23 | 76.76 | 74.25 |
| | | **LoRA-FAIR** | **84.99** | **55.15** | **80.51** | **62.77** | **89.48** | **77.03** | **74.99** |
| | MLP-Mixer | FFA-LoRA | 62.91 | 33.65 | 64.47 | 25.76 | 79.85 | 50.63 | 52.88 |
| | | FedIT | 71.53 | 39.00 | 68.76 | 42.44 | 82.34 | 60.58 | 60.77 |
| | | FLoRA | 70.06 | 37.26 | 67.48 | 41.56 | 81.37 | 60.01 | 59.62 |
| | | FlexLoRA | 71.58 | 39.50 | 68.89 | 43.85 | 82.39 | 60.99 | 61.20 |
| | | **LoRA-FAIR** | **72.79** | **40.91** | **69.49** | **45.99** | **82.59** | **61.91** | **62.28** |
| | | | **Autumn** | **Dim** | **Grass** | **Outdoor** | **Rock** | **Water** | **Average** |
| NICO++ | ViT | FFA-LoRA | 91.42 | 86.99 | 92.06 | 88.83 | 90.10 | 87.29 | 89.45 |
| | | FedIT | 91.31 | 86.91 | 92.33 | 89.01 | 89.97 | 87.37 | 89.48 |
| | | FLoRA | 91.28 | 87.07 | 92.27 | 89.52 | 90.04 | 87.43 | 89.60 |
| | | FlexLoRA | **91.81** | 87.23 | 92.45 | 89.25 | 89.79 | 87.37 | 89.65 |
| | | **LoRA-FAIR** | 91.79 | **87.59** | **92.90** | **89.98** | **90.39** | **87.60** | **90.04** |
| | MLP-Mixer | FFA-LoRA | 80.04 | 72.98 | 82.07 | 77.68 | 76.23 | 71.65 | 76.78 |
| | | FedIT | 81.47 | 74.50 | 83.64 | 78.67 | 78.72 | 74.20 | 78.53 |
| | | FLoRA | 80.92 | 74.58 | 83.15 | 79.21 | 78.36 | 74.25 | 78.41 |
| | | FlexLoRA | 82.02 | 75.02 | 83.33 | 78.88 | 78.94 | 74.25 | 78.73 |
| | | **LoRA-FAIR** | **82.46** | **76.02** | **83.79** | **79.84** | **80.16** | **74.90** | **79.53** |

Table 2. **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.

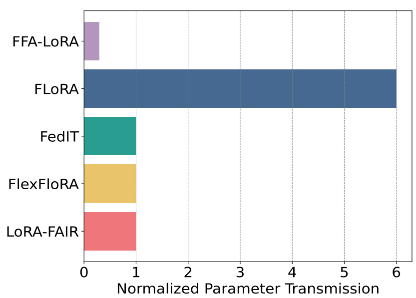


Figure 5. **Communication cost comparison.** LoRA-FAIR matches the communication cost of FedIT and FlexLoRA and avoids FLoRA's high overhead. Details in Sec. 5.1.

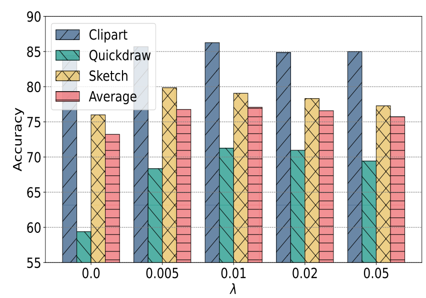


Figure 6. **Impact of Regularization Weight** $\lambda$. With $\lambda = 0$, LoRA-FAIR results in the lowest performance, underscoring the importance of this term. Details in Sec. 5.2.

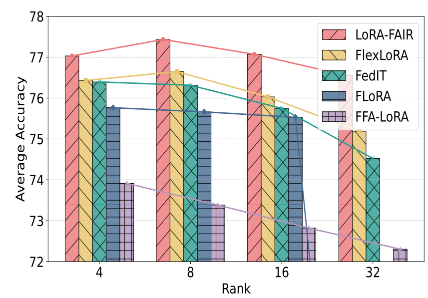


Figure 7. **Impact of LoRA Rank.** LoRA-FAIR outperforms baselines across ranks $\{4, 8, 16, 32\}$, with higher ranks not always improving performance, consistent with [6].

| Residual | Clipart | Infograph | Painting | Quickdraw | Real | Sketch | Average |
|---|---|---|---|---|---|---|---|
| $\Delta$A | 84.93 | 54.55 | 80.08 | 71.13 | 89.48 | 78.39 | 76.42 |
| $\Delta$B | **86.25** | **56.26** | **80.09** | **71.25** | **89.52** | **79.06** | **77.07** |

Table 3. **Performance comparison under different choices of residual LoRA modules position**. See details in Sec. 5.2.

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

Table 4. **Impact of the regularization term on the similarity and the average accuracy metrics.** See details in Sec. 5.2.

A closely related area to our work involves federated fine-tuning using LoRA. Zhang et al. [38] is the first study to apply LoRA in a federated context; however, this method overlooks potential server aggregation bias. Several subsequent works have been proposed: FFA-LoRA [26] freezes the non-zero initialized low-rank matrices and updates only the zero-initialized matrices, FlexLoRA [2] uses SVD to redistribute weights, and FLoRA [34] stacks local LoRA modules and transmits them to each client. However, these methods do not address client initialization drift.

## 7. Conclusion

In this work, we proposed LoRA-FAIR to address the key challenges of server-side aggregation bias and client-side initialization drift 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.

## 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, 7, 8

[3] Jieming Bian, Lei Wang, and Jie Xu. Prioritizing modalities: Flexible importance scheduling in federated multimodal learning. *arXiv preprint arXiv:2408.06549*, 2024. 7

[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. 7

[6] 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. 7, 8

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

[8] 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

[9] 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. 7

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

[11] 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. 7

[12] 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, 7

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

[14] 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. 7

[15] 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, 7

[16] 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. 7

[17] Brian Lester, Rami Al-Rfou, and Noah Constant. The power of scale for parameter-efficient prompt tuning. *arXiv preprint arXiv:2104.08691*, 2021. 7

[18] 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

[19] Xiang Lisa Li and Percy Liang. Prefix-tuning: Optimizing continuous prompts for generation. *arXiv preprint arXiv:2101.00190*, 2021. 7

[20] 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. 7

[21] 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, 7

[22] 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

[23] 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. 7

[24] 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

[25] 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. 7

[26] Youbang Sun, Zitao Li, Yaliang Li, and Bolin Ding. Improving lora in privacy-preserving federated learning. *arXiv preprint arXiv:2403.12313*, 2024. 4, 5, 7, 8

[27] 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. 7

[28] 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. 7

[29] 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

[30] 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

[31] 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. 7

[32] 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.

[33] 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. 7

[34] 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, 4, 5, 6, 7, 8

[35] 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. 7

[36] 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. 7

[37] 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

[38] 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

[39] 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