

FLoRA-Chef: Making A Good LoRA Recipe in Federated Generalization

Wenwen He[1] **Wenke Huang**[1*] **Yiyan Qi**[2] **He Li**[3]
Ming Hu[4] **Yi Liu**[5] **Guansong Pang**[6*]

[1]College of Computing and Data Science, Nanyang Technological University
[2]International Digital Economy Academy (IDEA)
[3]Wuhan University [4]East China Normal University
[5]City University of Hong Kong
[6]Singapore Management University

Abstract

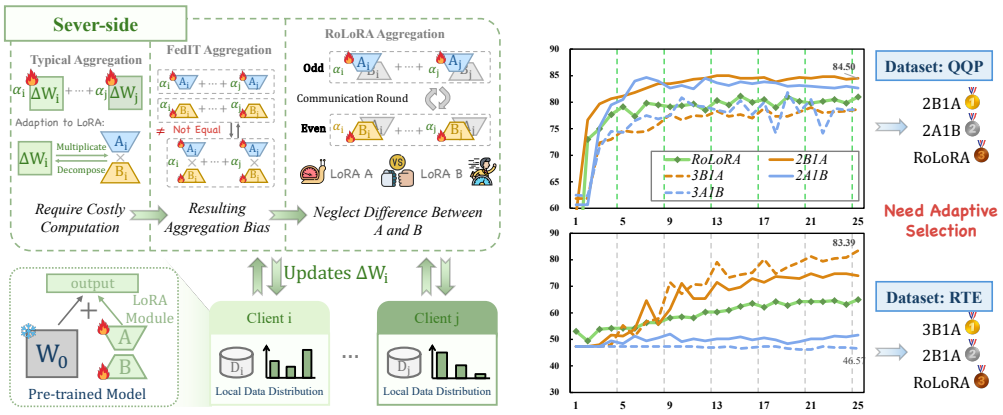
Low-Rank Adaptation (LoRA) is a popular parameter-efficient fine-tuning (PEFT) method that adapts the large model with few trainable parameters. Federated LoRA extends LoRA to federated learning (FL), enabling clients to collaboratively fine-tune a shared model without sharing their raw data by exchanging LoRA parameters. However, due to distributed data heterogeneity, local LoRA updates are highly inconsistent across clients, hindering robust global generalization. Moreover, common direct aggregation introduces *aggregated bias*, yielding noisy updates and further harming the global generalization. Existing bias-mitigation approaches often rely on symmetrically alternating LoRA modules, overlooking matrix distinct characteristics during training. Therefore, we propose **FLoRA-Chef**, an asymmetric Federated LoRA that decides *when* and *how* to aggregate LoRA components. *FLFire* adaptively selects the next aggregation target by tracking cross-clientupdate dispersion. *FLSauce* applies matrix-distinct reweighting to control diversity and prioritize stability. Experiments on various scenarios demonstrate consistent improvements, yielding an average gain of **6.94%** over counterparts, validating the benefit of adaptive asymmetric aggregation for federated LoRA.

1 Introduction

Federated large language models (LLMs) provide a feasible approach for collaboratively optimizing a shared model for downstream tasks without sharing local data. This formulation has attracted both academic interest [40] and industry investment [46]. Notably, typical federated learning (FL) paradigm relies on full parameter aggregation [26, 20], which introduces considerable computation and communication overhead. Parameter-efficient fine-tuning (PEFT), particularly Low-Rank Adaptation (LoRA) [14, 22, 21], has been adopted as an effective solution for adapting large models in federated settings [19, 42, 38, 2]. As shown in Figure 1a, LoRA adds a pair of trainable projection-up matrix $\boldsymbol{B}$ and projection-down matrix $\boldsymbol{A}$ to compute the model update $\boldsymbol{\Delta W}$, while the entire $\boldsymbol{W}$ is frozen. Building on this motivation, Federated LoRA (FedLoRA) provides a feasible solution to require multiple parties, *e.g.*, the kth client, to train their own $\{\boldsymbol{A}_k, \boldsymbol{B}_k\}$ and collaboratively communicate the LoRA modules, which enables efficient federated adaptation of large models by substantially reducing computation costs while maintaining strong performance [2, 4, 31].

*Corresponding authors: Wenke Huang (wenke.huang@ntu.edu.sg) and Guansong Pang (gspang@smu.edu.sg).

40th Conference on Neural Information Processing Systems (NeurIPS 2026).



(a) **Illustration of existing FedLoRA methods**. Full-update aggregation incurs high communication overhead. Module-wise LoRA aggregation reduces transmission but introduces *aggregation bias*. Alternating schemes further lower cost and mitigate bias by switching between $\mathbf{A}$ and $\mathbf{B}$, yet treat them symmetrically and ignore their distinct roles.

(b) **Matrix Update Frequency Observation**. We conduct empirical experiments on QQP and RTE to compare different LoRA update frequencies. Compared with the symmetric strategy RoLoRA, asymmetric alternation consistently yields better performance, suggesting that $\mathbf{A}$ and $\mathbf{B}$ should be updated with different schedules. More details are shown in Section 3.2.

According to typical federated formulations, their straightforward updates aggregation of $\boldsymbol{\Delta W_k}$ requires the costly matrix multiplication for $\boldsymbol{B}_k \cdot \boldsymbol{A}_k$ and the decomposition for $\boldsymbol{\Delta W_k}$. Previous studies, such as FedIT [42], directly aggregate client-side LoRA modules:

$$\boldsymbol{B}_g\boldsymbol{A}_g = (\sum_{k=1}^{K} \alpha_k \boldsymbol{A}_k)(\sum_{k=1}^{K} \alpha_k \boldsymbol{B}_k). \tag{1}$$

where α_k denotes aggregated weight of client k. This deviation from the original update formulation, *i.e.*, $\sum_{k=1}^{K} \alpha_k \boldsymbol{B}_k \boldsymbol{A}_k$, introduces *aggregated bias* ($\boldsymbol{B}_g\boldsymbol{A}_g \neq \sum_{k=1}^{K} \alpha_k \boldsymbol{B}_k \boldsymbol{A}_k$)[2], which brings the update noise and hinders generalization ability [3].

To mitigate aggregation bias, a straightforward solution is to stack client-specific $\boldsymbol{A}$ and $\boldsymbol{B}$ matrices [38, 5]. However, this strategy remains costly in federation, since stacking ranks increases computation and communication overhead as the number of clients and communication rounds grows. More recently, a stream of works has focused on selectively updating candidate matrices (e.g., $\boldsymbol{A}$ or $\boldsymbol{B}$) to avoid aggregation bias. In particular, these methods iteratively optimize and aggregate only one type of matrix per communication round [4, 17, 31]. Nevertheless, existing approaches treat the $\boldsymbol{A}$ and $\boldsymbol{B}$ matrices symmetrically, overlooking the fact that $\boldsymbol{A}$ and $\boldsymbol{B}$ exhibit different representational capacities and optimization dynamics, as demonstrated in [13, 33, 43].

Thus, motivated by the above analysis, we adopt an asymmetric scheme for FedLoRA to *incorporate matrix-specific characteristics*. As shown in Figure 1b, compared to some fixed alternative strategies, symmetric matrix aggregation (*e.g.*, RoLoRA) is not always optimal. Allowing one matrix to be updated for a few consecutive rounds before switching can sometimes improve the global generalization. The observation naturally raises a fundamental question: **i**) *How to determine the matrix dynamic alternation between $\boldsymbol{A}$ and $\boldsymbol{B}$ matrices.* Furthermore, existing LoRA matrix aggregation typically relies on the arithmetic mean, as shown in Equation (1). Such uniform aggregation neglects the distinct roles of $\boldsymbol{A}$ and $\boldsymbol{B}$, underusing their matrix-specific advantages. This leads to another critical problem: **ii**) *How to design distinct reweighting strategies for $\boldsymbol{A}$ and $\boldsymbol{B}$ aggregation?*

To address these challenges, we propose ***FLoRA-Chef***, a federated generalization method to choose a good time and a good way to aggregate the $\boldsymbol{A}$ and $\boldsymbol{B}$ matrix. For Q1, we propose the first module ***FLFire*** *(Federated Fire)*. This module adaptively selects the alternation points by the change of the similarity between inter-client $\boldsymbol{A}$ and $\boldsymbol{B}$ updates. For Q2, we propose the second module ***FLSauce*** *(Federated Sauce)*. We introduce an asymmetric reweighting strategy during aggregation. This strategy stabilizes $\boldsymbol{B}$ updates, while allowing controlled diversity in $\boldsymbol{A}$ to preserve generalization. This enables our method to achieve consistently stronger generalization than the existing methods. The main contributions can be summarized as follows:

[2]Detailed theoretical proofs are provided in the Section A.

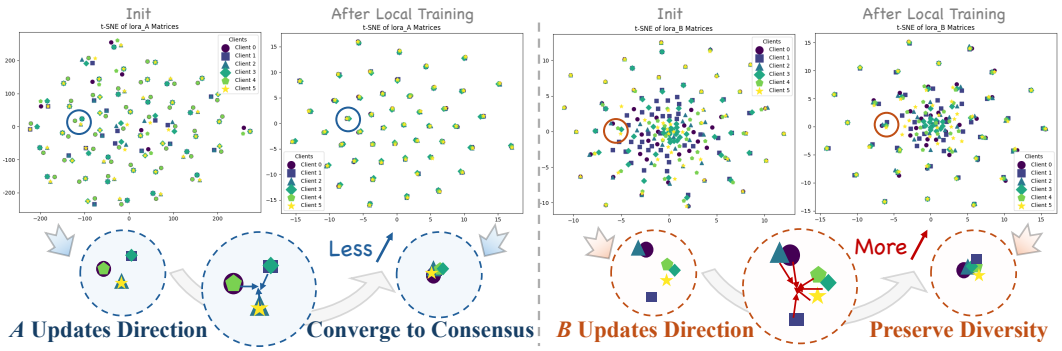


Figure 2: **Federated Matrix Behavior Analysis** shows t-SNE projections of LoRA $\boldsymbol{A}$ (left) and $\boldsymbol{B}$ (right) matrices from all clients at initialization and after local training. At initialization, $\boldsymbol{A}$ appears more consensus-oriented across clients, whereas $\boldsymbol{B}$ is more diverse, shaping their distinct optimization behaviors during local training. More details are shown in Section 3.2.

- We reveal the *aggregation bias* in Federated LoRA, showing that naively aggregating LoRA matrices in a matrix-wise manner introduces a non-negligible *noisy term*.
- We propose ***FLoRA-Chef***, an asymmetric matrix alternation and selective aggregation framework that jointly designs *when* to alternate and *how* to aggregate: it tracks client-wise dispersion changes of $\boldsymbol{A}$ and $\boldsymbol{B}$ to decide the alternation point, and performs selective aggregation through $\boldsymbol{A}$ and $\boldsymbol{B}$ heterogeneous optimization dynamics.
- We evaluate on general-domain (GLUE) and financial language understanding task (FLUE), where our approach achieves the +6.19% average gain over FedIT on GLUE.

2 Related Work

2.1 Low-Rank Adaption (LoRA)

Parameter-Efficient Fine-Tuning (PEFT) adapt the foundation model, *e.g.*, large language model, to downstream tasks with significantly computation and storage costs reduce [25]. Among them, Low-Rank Adaptation (LoRA) [14] is a widely adopted technique that decomposes the weight update into two trainable low-rank matrices added in parallel to pre-trained weights. During training, only these low-rank parameters are optimized and can be merged back at inference. Recent studies improve LoRA by enhancing optimization stability, generalization, or representational capacity through regularization, adaptive optimization, or architectural modifications [18, 39, 1, 13, 23, 36]. These improvements preserve the efficiency of LoRA while enabling effective adaptation of large models with minimal additional parameters and computational overhead.

2.2 Federated Large Language Models (FedLLMs)

With the rapid success of ChatGPT and related applications [32, 28, 27, 45], LLMs have demonstrated remarkable capabilities across a wide range of tasks [29, 37]. However, adapting LLMs to real-world applications often requires privacy-sensitive access to diverse and distributed data, which is difficult to collect in centralized settings due to legal, ethical, and ownership constraints. Federated Learning [34, 26] provides a promising solution by enabling multiple parties to collaboratively optimize a shared model through parameter or update sharing rather than raw data exchange. Federated LLMs (or FedLLMs) [44, 11] extend this paradigm to LLMs, allowing distributed clients to contribute task-specific knowledge while maintaining privacy guarantees and data locality.

2.3 Federation Meets LoRA

As LoRA reduces the computation and communication costs of LLM fine-tuning [14, 22, 15], combining it with federated learning has become increasingly attractive. Federated LoRA enables scalable and privacy-preserving LLM adaptation in multi-party settings [41]. A representative baseline, FedIT [42], directly averages clients LoRA parameters. However, the factorized LoRA structure makes naive low-rank averaging prone to aggregation bias and inconsistent global updates.

Existing solutions mainly fall into two categories. `I) Server-side stacking aggregation` stacks client LoRA matrices to reconstruct the exact summed update and avoid aggregation bias [12, 6, 30, 7]. For instance, FLoRA preserves local updates by stacking $\boldsymbol{A}_k$ and $\boldsymbol{B}_k$ along the rank dimension, but its rank size and server computation grow with the number of clients [12]. `II) Client-side selective updating` alternately updates and aggregates either $\boldsymbol{A}$ or $\boldsymbol{B}$ to reduce bias and overhead [4, 17]. However, these methods often treat $\boldsymbol{A}$ and $\boldsymbol{B}$ symmetrically or use fixed schedules, overlooking their asymmetric optimization dynamics.

3 Preliminray and Motivation

3.1 Preliminray

LoRA. LoRA is a PEFT method that adapts pre-trained models by learning low-rank updates while keeping the original weights frozen [14]. Given a pre-trained weight matrix $\boldsymbol{W}_0$, LoRA models the adapted weight as

$$\boldsymbol{W} = \boldsymbol{W}_0 + \boldsymbol{\Delta W} = \boldsymbol{W}_0 + \boldsymbol{BA}, \tag{2}$$

where $\boldsymbol{A}$ is the projection-down matrix and $\boldsymbol{B}$ is the projection-up matrix. Only $\boldsymbol{A}$ and $\boldsymbol{B}$ are optimized during training, which reduces resource overhead.

Federated LoRA. Federated LoRA enables multiple clients to collaboratively fine-tune a shared model without sharing raw data. Each client locally optimizes its LoRA parameters $\{\boldsymbol{A}_k, \boldsymbol{B}_k\}$ based on its private data, and then the server aggregates these updates to obtain a global adaptation. A typical aggregation can be written as follows

$$\boldsymbol{W_g} = \boldsymbol{W}_0 + \boldsymbol{\Delta W_g} = \boldsymbol{W_0} + \sum_{k=1}^{K} \alpha_k \boldsymbol{B}_k \boldsymbol{A}_k, \tag{3}$$

where K denotes the number of clients and α_k is the aggregation weight. A representative instantiation is FedIT [42], which performs separate aggregation of the $\boldsymbol{A}$ and $\boldsymbol{B}$ matrices across clients:

$$\boldsymbol{A_g} = \sum_{k=1}^{K} \alpha_k \boldsymbol{A}_k, \quad \boldsymbol{B_g} = \sum_{k=1}^{K} \alpha_k \boldsymbol{B}_k. \tag{4}$$

However, this matrix-wise aggregation is generally biased:

$$\boldsymbol{\mathcal{N}} \triangleq (\sum_{k=1}^{K} \alpha_k \boldsymbol{B}_k)(\sum_{k=1}^{K} \alpha_k \boldsymbol{A}_k) - \sum_{k=1}^{K} \alpha_k \boldsymbol{B}_k \boldsymbol{A}_k \neq 0. \tag{5}$$

which introduces noise $\boldsymbol{\mathcal{N}}$ and hinds generalization ability, also comfirmed in [38, 3].

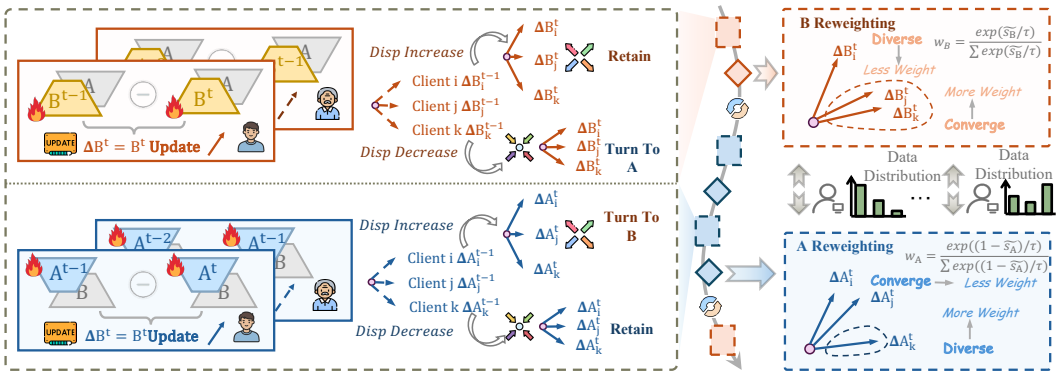
3.2 Motivation

To mitigate the aggregated bias, selectively updating $\boldsymbol{A}$ and $\boldsymbol{B}$ has emerged as a practical and effective solution [4, 17, 31]. Specifically, these methods alternately update and aggregate only one LoRA matrix while freezing the other unshared matrix, thereby avoiding matrix multiplication mismatch during aggregation. At round t, each client k receives $\{\boldsymbol{A}^t, \boldsymbol{B}^t\}$ and updates only one matrix:

$$\{\boldsymbol{A}_k^{t+1}, \boldsymbol{B}_k^{t+1}\} = \begin{cases} \{\boldsymbol{A}_k^{t+1},\ \boldsymbol{B}^t\}, & \boldsymbol{A}_k^{t+1} \leftarrow \arg\min_{\boldsymbol{A}}\ \mathcal{L}_k, \\ \{\boldsymbol{A}^t,\ \boldsymbol{B}_k^{t+1}\}, & \boldsymbol{B}_k^{t+1} \leftarrow \arg\min_{\boldsymbol{B}}\ \mathcal{L}_k. \end{cases} \tag{6}$$

Here, $\mathcal{L}_k$ denotes the local objective of client k. As a result, prior explorations typically apply **symmetric** schedules and aggregation operations to the two matrices. However, the $\boldsymbol{A}$ and $\boldsymbol{B}$ modules inherently play different functional roles and exhibit distinct optimization dynamics [13, 33, 10], which motivates us to treat them *asymmetrically* in the federated system.

Matrix Update Frequency Observation. We conduct an observational study by varying the update frequencies of $\boldsymbol{A}$ and $\boldsymbol{B}$. As illustrated in Figure 1b, selectively allocating asymmetric updates to either $\boldsymbol{A}$ or $\boldsymbol{B}$ can achieve better performance than the symmetric update scheme, indicating that equal-frequency alternation is not always optimal. Moreover, Figure 1b reveals a dataset dependence in the preferable update frequency of $\boldsymbol{A}$ and $\boldsymbol{B}$ (*e.g.*, QQP prefers 2B1A, while RTE favors 3B1A), suggesting that the optimal schedule should adapt the federated dynamics. This finding prompts us to rethink $\boldsymbol{A}$ and $\boldsymbol{B}$ as inherently different components that should be treated distinctly in federation.



FLFire: Adaptive Matrix Alternation *FLSauce: Selective Matrix Aggregation*

Figure 3: **Overview of FLoRA-Chef**. **Left: FLFire** adaptively decides whether to continue aggregating $\boldsymbol{A}$ or $\boldsymbol{B}$ by tracking cross-client *update dispersion*. **Right: FLSauce** performs selective matrix aggregation: for $\boldsymbol{A}$, it upweights *diverse* directions to preserve controlled diversity, while for $\boldsymbol{B}$, it upweights *convergent* directions to suppress outliers and stabilize training.

Federated Matrix Behavior Analysis. We further investigate this asymmetry via t-SNE projections of the $\boldsymbol{A}$ and $\boldsymbol{B}$ matrices from all clients, both at initialization and after local training. As shown in Figure 2, $\boldsymbol{A}$ becomes less dispersed after local updates: client representations move in less divergent directions, indicating that $\boldsymbol{A}$ updates are *consensus-seeking*. In contrast, $\boldsymbol{B}$ remains widely scattered and exhibits increased directional diversity, suggesting that $\boldsymbol{B}$ updates are *diversity-preserving*. These observations imply that $\boldsymbol{A}$ and $\boldsymbol{B}$ follow inherently different optimization dynamics, consistent with prior findings [10, 33]. Motivated by this, we revisit the federated pipeline, including *client-side update* and *server-side aggregation*, and argue that both stages should account for the distinct characteristics of $\boldsymbol{A}$ and $\boldsymbol{B}$ to improve generalization under heterogeneous federation. Thus, we propose FLoRA-Chef, an adaptive asymmetric operation framework for FedLoRA.

4 Methodology

4.1 FLFire: Adaptive Matrix Alternation.

FLFire is an adaptive update strategy for federated LoRA that dynamically **determines whether to aggregate the projection-down matrix $\boldsymbol{A}$ or the projection-up matrix $\boldsymbol{B}$** at each communication round. Instead of adopting a fixed alternation, FLFire monitors the cross-client updates consistency of LoRA matrixces and uses it as a signal to guide the alternation between $\boldsymbol{A}$ and $\boldsymbol{B}$ aggregation.

Local Updating. Since the local updating matrix varies across local training rounds (between $\boldsymbol{A}$ and $\boldsymbol{B}$), we denote the k-th client's trainable local LoRA matrix at round t as $\mathcal{M}_k^t$. Moreover, we denote the state signal as $m^t \in \{\text{update } A,\ \text{update } B\}$ to indicate the target matrix for the t-th local training round. After the server broadcasts the state signal m^t, each client performs local training by updating only the corresponding LoRA matrix while freezing the other. For the k-th client, the updated matrix at round t is

$$\mathcal{M}_k^t = \begin{cases} \boldsymbol{A}_k^t, & \text{if } m^t = \text{update } A, \\ \boldsymbol{B}_k^t, & \text{if } m^t = \text{update } B. \end{cases} \tag{7}$$

$\mathcal{M}_k^{t+1} = \boldsymbol{A}_k^{t+1}$ means updating $\boldsymbol{A}_k$ while freezing $\boldsymbol{B}_k$, and $\mathcal{M}_k^{t+1} = \boldsymbol{B}_k^{t+1}$ means updating $\boldsymbol{B}_k$ while freezing $\boldsymbol{A}_k$.

Matrix Update. We define the client-side update of the selected matrix at round t as

$$\Delta\mathcal{M}_k^t = \mathcal{M}_k^t - \mathcal{M}_g, \quad \mathcal{M}_g = \begin{cases} \boldsymbol{A}_g, & \text{if } m^t = \text{update } A, \\ \boldsymbol{B}_g, & \text{if } m^t = \text{update } B. \end{cases} \tag{8}$$

Here, $\mathcal{M}_g$ serves as the most recent server-side reference parameter and $\Delta\mathcal{M}_k^t \in \{\Delta A_k^t, \Delta B_k^t\}$. This definition allows us to compute update similarity and cross-client dispersion in a unified and comparable manner, regardless of whether $\boldsymbol{A}$ or $\boldsymbol{B}$ is alternated at each communication round.

Matrix Dispersion. Since the aggregated matrix varies across communication rounds (alternating between $\boldsymbol{A}$ and $\boldsymbol{B}$), we denote the matrix update optimized by client k at round t as $\Delta\mathcal{M}_k^t \in \{\Delta\boldsymbol{A}_k^t, \Delta\boldsymbol{B}_k^t\}$. To quantify cross-client disagreement in update directions, we define the φ as:

$$\varphi_{\mathcal{M}}^t = 1 - \frac{2}{K(K-1)} \sum_{1 \le i < j \le K} \cos\big(\Delta\mathcal{M}_i^t, \Delta\mathcal{M}_j^t\big)\,, \tag{9}$$

where a larger value indicates greater update inconsistency across clients.

Adaptive Alternation Rule. $\varphi_{A,g}$ and $\varphi_{B,g}$ denote the latest recorded update dispersions of $\boldsymbol{A}$ and $\boldsymbol{B}$, which serve as references for alternation decisions. At the end of communication round t, FLFire determines the next-round training/aggregation target. Specifically, the decision signal m^{t+1} is given by:

$$m^{t+1} = \begin{cases} \text{update } B, & \text{if } m^t = \text{update } A \ \wedge\ \varphi_A^t \ge \varphi_{A,g}, \\ \text{update } A, & \text{if } m^t = \text{update } B \ \wedge\ \varphi_B^t \le \varphi_{B,g}, \\ m^t, & \text{otherwise.} \end{cases} \tag{10}$$

where $\{\varphi_A^t, \varphi_B^t\}$ denotes the directional dispersion of $\boldsymbol{A}^t$, $\boldsymbol{B}^t$ updates and $\{\ \varphi_{A,g}, \varphi_{B,g}\ \}$ denotes latest dispersion.

4.2 FLSauce: Selective Matrix Aggregation.

FLSauce is a reweighting module by refining the aggregation of LoRA components through client-specific weights. While FLFire determines which LoRA matrix to aggregate at each round, FLSauce focuses on **how to aggregate the selected component** in a more stable and generalized manner. FLSauce introduces two respectively asymmetric reweighting mechanisms for the $\boldsymbol{A}$ and $\boldsymbol{B}$ updates.

Federated A Matrix Reweighting. When $\boldsymbol{A}$ is selected for aggregation, FLSauce adopts an *individuality-aware reweighting* strategy. $\boldsymbol{A}$ typically evolves more slowly and exhibits more consistent updates across clients, making it beneficial to emphasize relatively distinctive client updates to facilitate localized exploration and avoid over-smoothing. For each client k, we compute the cosine similarity $s_{A,k}$ between its $\boldsymbol{A}$ update $\Delta\boldsymbol{A}_k$ and a reference update formed by averaging the updates of the remaining clients $\{\Delta\boldsymbol{A}_i\}_{i=1,i\neq k}^K$. Therefore, clients whose updates are less aligned with the reference are assigned higher aggregation weights via a similarity-to-weight mapping:

$$\alpha_{A,k} = \frac{\exp((1-\tilde{s}_{A,k})/\tau)}{\sum_{i=1}^K \exp((1-\tilde{s}_{A,i})/\tau)}, \quad \boldsymbol{A}_g = \sum_{k=1}^K \alpha_{A,k}\,\boldsymbol{A}_k. \tag{11}$$

The symbol $s_{A,k} = \cos(\Delta\boldsymbol{A}_k, \Delta\boldsymbol{A}_{i,i\neq k})$, $\tilde{s}_{A,k} = (s_{A,k}+1)/2 \in [0,1]$ is the rescaled cosine similarity, and τ is a temperature parameter. This design encourages diverse update directions to contribute more strongly to $\boldsymbol{A}$ aggregation, preventing over-dominance by highly aligned clients.

Federated B Matrix Reweighting. When $\boldsymbol{B}$ is selected for aggregation, FLSauce adopts an *stability-aware reweighting* strategy. Unlike $\boldsymbol{A}$, $\boldsymbol{B}$ is optimized more aggressively and is more prone to divergence, making it crucial to emphasize directionally consistent updates during aggregation. For each client k, we compute the cosine similarity $s_{B,k} = \cos(\Delta\boldsymbol{B}_k, \Delta\boldsymbol{B}_{i,i\neq k})$ between its $\boldsymbol{B}$ update direction and the reference direction formed by averaging the remaining clients. Clients whose updates are more aligned with the reference are assigned higher aggregation weights via

$$\alpha_{B,k} = \frac{\exp(\tilde{s}_{B,k}/\tau)}{\sum_{i=1}^K \exp(\tilde{s}_{B,i}/\tau)}, \quad \boldsymbol{B}_g = \sum_{k=1}^K \alpha_{B,k}\,\boldsymbol{B}_k. \tag{12}$$

The symbol $\tilde{s}_{B,k} = (s_{B,k}+1)/2 \in [0,1]$ is the rescaled cosine similarity. This design suppresses outlier updates and stabilizes the aggregation of $\boldsymbol{B}$ by prioritizing clients that exhibit consistent optimization directions. Therefore, in our work, we utilize the proposed $\alpha_{A,k}$, $\alpha_{B,k}$ to replace orgianl aggregattion weight in Equation (4).

4.3 Discussion and Limitation

Privacy Concern Discussion. In FedIT [42], clients upload both $\boldsymbol{A}$ and $\boldsymbol{B}$ for server aggregation, which can increase the attack surface [35] for parameter inversion [9] and related inference attacks [16, 24, 8]. However, our approach does not transmit the entire LoRA module in each round: clients upload only one matrix (either $\boldsymbol{A}$ or $\boldsymbol{B}$) for aggregation. Moreover, compared to fixed alternation schedules [4], our dynamic alternation reduces the predictability of the uploaded components across rounds, making it harder for an attacker to reliably exploit a known upload pattern.

Table 1: FLFire Choice. The top panel visualizes four training schedules, and the bottom table reports accuracy across GLUE tasks. According to the results and the observation in Section 3.2, FLFire selects **A↑B↓**.



	MNLI	QNLI	MRPC	QQP	RTE	Avg.
A↓B↓	80.13	87.40	73.19	81.40	69.89	78.40
A↑B↑	82.03	**86.87**	77.21	83.27	**76.68**	81.21
A↓B↑	81.60	86.40	77.11	82.40	61.23	77.75
A↑B↓	**82.80**	86.13	**83.87**	**86.57**	66.93	**81.26**

Table 2: Ablation of Target Objective. Combining adaptive matrix aggregation FLFire with selective reweighting FLSauce achieves consistent gains across QNLI and MRPC. More details are provided in Section 5.2.

FLFire	FLSauce	QNLI	MRPC	Avg.
		79.67	81.77	80.72
✓		86.13 ↑6.46	83.87 ↑2.10	85.00 ↑4.28
✓	✓	**86.37** ↑6.70	**87.65** ↑5.88	**87.01** ↑6.29

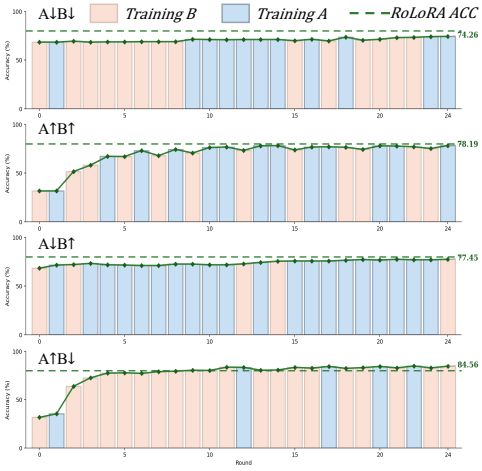


Figure 4: Training schedules and ACC trends on MRPC. Shaded bars indicate which LoRA component is optimized at each communication round. The dashed line represents the base strategy FedIT. A↓B↑ usually allocates more rounds to **B** and achieves better performance. Details are shown in Section 5.2.

Limitation. Our method requires the server to cache the aggregated matrix from the previous round and to compute cross-client similarities ($\mathcal{O}(K^2)$) to guide dynamic alternation and reweighting, which introduces additional computation and storage overhead compared to naive averaging.

5 Experiment

5.1 Experimental Setup

Foundation Models. We mainly adopt RoBERTa-large as the foundation model for natural language understanding experiments, and further evaluate LLaMA on the reasoning task to verify model generality. In our federated setting, all backbone parameters are frozen, and only the inserted LoRA modules are optimized and aggregated across clients.

Datasets. We conduct experiments on general-domain, financial-domain, and reasoning benchmarks. Specifically, GLUE includes representative natural language understanding tasks, including MNLI, QNLI, QQP, RTE, and MRPC. Headline is a financial-domain dataset from the FLUE benchmark, covering market Direction and Price subtasks. We further evaluate GSM8K to assess the reasoning ability of LLaMA-based federated LoRA adaptation.

Heterogeneity Setting. Under GLUE and Headline dataset, we simulate via Dirichlet distribution via the concentration parameter $\alpha \in \{0.5, 1.0\}$.

Training Details. For GLUE and FLUE tasks, we use batch size 128 and set the learning rates to 3×10^{-4} and 1×10^{-4}, respectively. Each GLUE client uses 1,000 training and 600 test samples with 20 local epochs, while each FLUE client uses 500 training and 300 test samples with 5 local iterations. All experiments are conducted with 6 clients under the same federation protocol, and we report the mean scores over the last 5 communication rounds. For GSM8K, we run a lightweight LLaMA-based setting with CoT reasoning, 2 local epochs and 5 global rounds, using 150 training samples and 25 test samples per client.

Counterparts. We compare our approach with five representative federated LoRA aggregation methods. The details are as follows:

- **FedIT** [ICASSP'24] [42]. FedIT aggregates LoRA by independently averaging the projection-down matrix $\boldsymbol{A}$ and the projection-up matrix $\boldsymbol{B}$ across clients.
- **FFA LoRA** [ICLR'24] [31]. FFA LoRA reconstructs the full LoRA update on each client and aggregates $\boldsymbol{B}$-side updates while freezing $\boldsymbol{A}$.

Table 3: Performance comparison under the label non-IID setting ($\beta = \{0.5, 1.0\}$) on GLUE (MNLI, QNLI, QQP, RTE, MRPC) and Headline tasks (Price, Direction). We report accuracy (%) for each task and the average score. The red arrows (↑) denote the absolute gains of **Ours** over FedIT, showing consistent improvements across both benchmark suites. More details are shown in Section 5.3.

Methods	GLUE						Headline		
	MNLI	QNLI	QQP	RTE	MRPC	Avg.	Price	Direction	Avg.
Label non-IID ($\beta = 0.5$)									
FedIT	83.27	79.67	72.83	69.53	81.77	77.41	91.73	94.67	93.20
RoLoRA	81.70	90.80	80.27	64.26	79.75	79.36	89.07	92.27	90.67
FFA-LoRA	81.43	89.70	82.20	66.14	78.58	79.61	91.87	92.13	92.00
FlexLoRA	83.30	89.97	**84.43**	76.61	80.78	83.02	92.20	94.60	93.40
LoRA-A^2	82.73	85.97	82.37	76.97	83.48	82.30	92.80	93.67	93.23
Ours	**85.30**	**86.37**	81.93	**80.51**	**87.65**	**84.35**	**93.60**	**94.93**	**94.27**
	↑ 2.03	↑ 6.70	↑ 9.10	↑ 10.98	↑ 5.88	↑ 6.94	↑ 1.87	↑ 0.26	↑ 1.07
Label IID ($\beta = 1.0$)									
FedIT	84.33	87.43	82.07	73.00	84.80	82.33	91.93	95.00	93.47
RoLoRA	86.00	85.90	83.60	72.64	84.61	82.55	93.40	91.47	92.44
FFA-LoRA	81.27	88.53	82.00	75.88	84.61	82.46	91.93	94.33	93.13
FlexLoRA	82.73	88.17	82.03	77.40	85.44	83.15	92.93	93.67	93.30
LoRA-A^2	81.10	90.43	82.10	66.71	82.55	80.58	93.53	92.67	93.10
Ours	**87.97**	**90.73**	**91.93**	**79.50**	**86.17**	**87.26**	**93.66**	**95.07**	**94.37**
	↑ 3.64	↑ 3.30	↑ 9.86	↑ 6.50	↑ 1.37	↑ 4.93	↑ 1.73	↑ 0.07	↑ 0.90

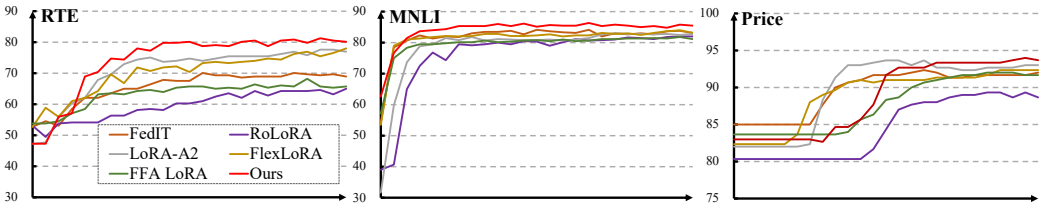


Figure 5: **Convergence Curves.** Test accuracy (%) versus communication rounds on RTE, MNLI, and Price. Our method consistently achieves strong accuracy across rounds and maintains a competitive convergence trend. More details can be seen in Section 5.3.

Table 4: Comparison between *Anti FLSauce* and *FLSauce*. *Anti FLSauce* reverses the selective weights in Equations (11) and (12). *Gain* denotes the absolute improvement of FLSauce over *Anti FLSauce*.

Task	Anti FLSauce	FLSauce	Gain
MNLI	82.80	**85.30**	↑ 2.50
QNLI	86.13	**86.37**	↑ 0.24
QQP	**86.57**	81.93	↓ 4.64
RTE	66.93	**80.51**	↑ 13.58
MRPC	83.87	**87.65**	↑ 3.78
Avg.	81.26	**84.35**	↑ 3.09

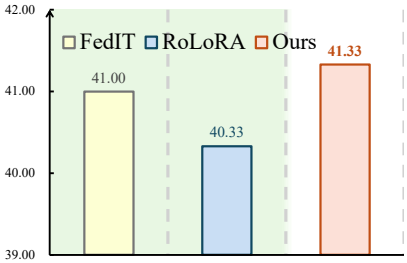


Figure 6: Performance comparison on LLaMA structure based on GSM8k dataset. FLSauce achieves better accuracy than FedIT and RoLoRA.

- **RoLoRA** [NeurIPS'25] [4]. RoLoRA adopts a fixed alternating aggregation strategy, where only one LoRA matrix ($\boldsymbol{A}$ or $\boldsymbol{B}$) is aggregated in each round.
- **FlexLoRA** [NeurIPS'24] [2]. FlexLoRA aggregates reconstructed LoRA updates with heterogeneous ranks and applies singular value decomposition to redistribute global LoRA parameters.
- **LoRA-$\boldsymbol{A}^2$** [ACL'25] [17]. LoRA-$\boldsymbol{A}^2$ alternatively aggregates A and B across rounds and adaptively selects only communicate important low-rank components.

5.2 Diagnostic Analysis

FLFire Choice. Table 1 summarizes the averaged GLUE results of four update strategies. $A \uparrow B \downarrow$ achieves the best overall average, and is therefore selected as the default choice of FLFire. Figure 4 further shows on MRPC that FLFire allocates more rounds to training $\boldsymbol{B}$ and fewer updates to $\boldsymbol{A}$, consistent with the observation in Figure 1b. These results suggest that emphasizing $\boldsymbol{B}$ updates is crucial for improving generalization under aggregation, while $\boldsymbol{A}$ updates play a complementary role.

FLSauce Choice. In Table 4, FLSauce consistently outperforms Anti FLSauce on most tasks, improving the average from 81.26 to 84.35. This confirms the practical benefit of the asymmetric design. Specifically, for the slowly evolving $\boldsymbol{A}$, FLSauce assigns higher weights to clients with more distinctive directions to preserve diversity; for the more volatile $\boldsymbol{B}$, it favors clients aligned with the majority to suppress outliers and stabilize training. Thus, FLSauce yields robust improvements under heterogeneous GLUE tasks, rather than gains on only a single dataset.

Target Objective Ablation. Table 2 evaluates the effects of FLFire and FLSauce on QNLI and MRPC. Starting from the baseline (80.72 Avg.), enabling FLFire improves the average to 85.00 ($\uparrow 4.28$), showing the importance of selecting the proper LoRA component for aggregation. Adding FLSauce further increases the average to 87.01 ($\uparrow 6.29$), with consistent gains on both tasks. Overall, FLFire provides the major boost, while FLSauce further stabilizes aggregation and improves generalization.

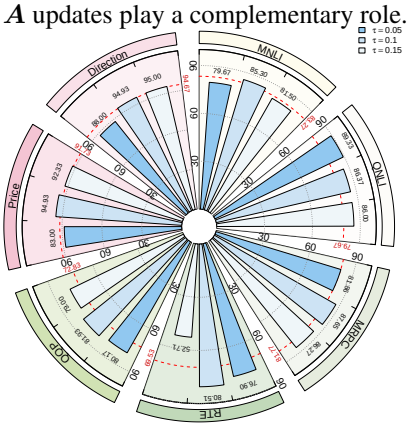


Figure 7: Ablation on temperature τ for client-selective aggregation (τ in Equation (11)). We use $\tau = 0.1$ as the default setting for stable performance.

Temperature τ Selection. We analyze $\tau \in \{0.05, 0.1, 0.15\}$ in the weighting scheme on GLUE and Headline tasks in Figure 7. A smaller τ (*e.g.*, 0.05) produces overly sharp weights and unstable gains, while a larger τ (*e.g.*, 0.15) over-smooths the weights and weakens FLSauce aggregation. Overall, $\tau = 0.1$ provides a suitable trade-off between client-selective aggregation and stable improvement, and is used in all experiments.

5.3 Comparison to State-of-the-Arts

Overall Performance. We compare FLoRA-Chef with representative federated LoRA aggregation baselines under $\beta \in \{0.5, 1.0\}$. As shown in Table 3, FLoRA-Chef achieves the best average accuracy on both GLUE and Headline, reaching 84.35/87.26 and 94.27/94.37 for $\beta = 0.5/1.0$, respectively. It brings consistent gains over FedIT across most tasks, with particularly clear improvements on RTE and QQP under the more heterogeneous setting $\beta = 0.5$. Moreover, Figure 6 shows that FLoRA-Chef also improves LLaMA-based reasoning on GSM8K, achieving higher accuracy than FedIT and RoLoRA. These results confirm that our asymmetric aggregation design is robust across different tasks, model architectures, and federation regimes.

Convergence Behavior. We further compare training trajectories via the convergence curves in Figure 5. Our method maintains strong accuracy across communication rounds and shows competitive convergence trends on RTE, MNLI, and Price. This is mainly because we dynamically alternate between aggregating $\boldsymbol{A}$ and $\boldsymbol{B}$ according to cross-client dispersion, avoiding over-aggregating the matrix to deepen diversity. Moreover, we apply matrix-distinct reweighting to suppress outlier updates while emphasizing more informative ones. Together, these designs yield more reliable aggregation signals across rounds, leading to consistent improvements and stable performance across tasks.

6 Conclusion

In this paper, we investigated the aggregation bias issue in Federated LoRA and showed that symmetric, matrix-wise aggregation can be misaligned with the distinct roles and optimization dynamics of the $\boldsymbol{A}$ and $\boldsymbol{B}$ matrices. To address this, we proposed FLoRA-Chef, an asymmetric matrix alternation and selective aggregation framework that jointly decides *when* to alternate and *how* to aggregate by monitoring cross-client directional dispersion and applying matrix-specific aggregation

rules. Extensive experiments on general-domain (GLUE), financial language understanding (FLUE), and mathematical reasoning (GSM8K) benchmarks demonstrate that FLoRA-Chef consistently outperforms representative federated LoRA baselines, achieving strong and stable improvements in generalization. We hope our findings encourage future federated adaptation methods to explicitly account for matrix-specific behaviors when designing efficient and robust aggregation strategies.

Acknowledgments and Disclosure of Funding

This work was supported by the National Research Foundation (NRF), Singapore, under Grant ID NRF-PA2025-NTU, and by Nanyang Technological University (NTU); the National Natural Science Foundation of China (Grant No. 623B2080); the Agency for Science, Technology and Research (A*STAR) under its MTC Young Individual Research Grant (YIRG, Grant No. M24N8c0103); the Singapore Ministry of Education (MOE) Academic Research Fund (AcRF) Tier 1 (Grant No. 24-SIS-SMU-008); and the Lee Kong Chian Fellowship (T050273); Fundamental Research Funds for the Central Universities

References

[1] Reda S Agiza A, Neseem M. Mtlora: Low-rank adaptation approach for efficient multi-task learning. In *CVPR*, 2024.

[2] Jiamu Bai, Daoyuan Chen, Bingchen Qian, Liuyi Yao, and Yaliang Li. Federated fine-tuning of large language models under heterogeneous tasks and client resources. In *NeurIPS*, 2024.

[3] Jieming Bian, Lei Wang, Letian Zhang, and Jie Xu. Lora-fair: Federated lora fine-tuning with aggregation and initialization refinement. In *ICCV*, pages 3737–3746, 2025.

[4] Shuangyi Chen, Yue Ju, Hardik Dalal, Zhongwen Zhu, and Ashish J Khisti. Robust federated finetuning of foundation models via alternating minimization of loRA. In *NeurIPS*, 2025.

[5] Yae Jee Cho, Luyang Liu, Zheng Xu, Aldi Fahrezi, and Gauri Joshi. Heterogeneous lora for federated fine-tuning of on-device foundation models. In *EMNLP*, 2024.

[6] Yae Jee Cho, Luyang Liu, Zheng Xu, Aldi Fahrezi, and Gauri Joshi. Heterogeneous lora for federated fine-tuning of on-device foundation models. *arXiv preprint arXiv:2401.06432*, 2024.

[7] Boyu Fan, Xiang Su, Sasu Tarkoma, and Pan Hui. Helora: Lora-heterogeneous federated fine-tuning for foundation models. *ACM TOIT*, 2025.

[8] Chong Fu, Xuhong Zhang, Shouling Ji, Jinyin Chen, Jingzheng Wu, Shanqing Guo, Jun Zhou, Alex X Liu, and Ting Wang. Label inference attacks against vertical federated learning. In *USENIX*, pages 1397–1414, 2022.

[9] Jonas Geiping, Hartmut Bauermeister, Hannah Dröge, and Michael Moeller. Inverting gradients-how easy is it to break privacy in federated learning? In *NeurIPS*, pages 16937–16947, 2020.

[10] Pengxin Guo, Shuang Zeng, Yanran Wang, Huijie Fan, Feifei Wang, and Liangqiong Qu. Selective aggregation for low-rank adaptation in federated learning. In *ICLR*, 2025.

[11] Tao Guo, Song Guo, Junxiao Wang, Xueyang Tang, and Wenchao Xu. Promptfl: Let federated participants cooperatively learn prompts instead of models-federated learning in age of foundation model. *IEEE TMC*, 2023.

[12] Yongchang Hao, Yanshuai Cao, and Lili Mou. Flora: Low-rank adapters are secretly gradient compressors. In *ICML*, 2024.

[13] Soufiane Hayou, Nikhil Ghosh, and Bin Yu. Lora+: Efficient low rank adaptation of large models. In *ICML*, 2024.

[14] 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. In *ICLR*, pages 1–26, 2022.

[15] Chengsong Huang, Qian Liu, Bill Yuchen Lin, Tianyu Pang, Chao Du, and Min Lin. Lorahub: Efficient cross-task generalization via dynamic lora composition. In *COLM*, 2024.

[16] Yigitcan Kaya and Tudor Dumitras. When does data augmentation help with membership inference attacks? In *ICML*, pages 5345–5355, 2021.

[17] Jabin Koo, Minwoo Jang, and Jungseul Ok. Towards robust and efficient federated low-rank adaptation with heterogeneous clients. In *ACL*, pages 416–429, 2025.

[18] Dawid J Kopiczko, Tijmen Blankevoort, and Yuki M Asano. Vera: Vector-based random matrix adaptation. In *ICLR*, 2023.

[19] 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 *ACM SIGKDD*, pages 5260–5271, 2024.

[20] Tian Li, Anit Kumar Sahu, Manzil Zaheer, Maziar Sanjabi, Ameet Talwalkar, and Virginia Smith. Federated optimization in heterogeneous networks. In *MLSys*, volume 2, pages 429–450, 2020.

[21] Tianwei Lin, Jiang Liu, Wenqiao Zhang, Zhaocheng Li, Yang Dai, Haoyuan Li, Zhelun Yu, Wanggui He, Juncheng Li, Hao Jiang, et al. Teamlora: Boosting low-rank adaptation with expert collaboration and competition. In *ACL*, 2025.

[22] Shih-Yang Liu, Chien-Yi Wang, Hongxu Yin, Pavlo Molchanov, Yu-Chiang Frank Wang, Kwang-Ting Cheng, and Min-Hung Chen. Dora: Weight-decomposed low-rank adaptation. In *ICML*, 2024.

[23] Yuheng Lu, Bingshuo Qian, Caixia Yuan, Huixing Jiang, and Xiaojie Wang. Controlled low-rank adaptation with subspace regularization for continued training on large language models. *arXiv preprint arXiv:2410.16801*, 2024.

[24] Xinjian Luo, Yuncheng Wu, Xiaokui Xiao, and Beng Chin Ooi. Feature inference attack on model predictions in vertical federated learning. In *ICDE*, pages 181–192. IEEE, 2021.

[25] Sourab Mangrulkar, Sylvain Gugger, Lysandre Debut, Younes Belkada, Sayak Paul, and B Bossan. Peft: State-of-the-art parameter-efficient fine-tuning methods. 2022.

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

[27] OpenAI. Gpt-4 technical report. *arXiv preprint arXiv:2303.08774*, 2023.

[28] Alec Radford, Jeffrey Wu, Rewon Child, David Luan, Dario Amodei, Ilya Sutskever, et al. Language models are unsupervised multitask learners. *OpenAI blog*, 1(8):9, 2019.

[29] Yifei Shen, Jiawei Shao, Xinjie Zhang, Zehong Lin, Hao Pan, Dongsheng Li, Jun Zhang, and Khaled B Letaief. Large language models empowered autonomous edge ai for connected intelligence. *IEEE Communications Magazine*, 62(10):140–146, 2024.

[30] Yang Su, Na Yan, Yansha Deng, Mischa Dohler, and Robert Schober. Haflq: Heterogeneous adaptive federated lora fine-tuned llm with quantization. *arXiv preprint arXiv:2411.06581*, 2024.

[31] Youbang Sun, Zitao Li, Yaliang Li, and Bolin Ding. Improving loRA in privacy-preserving federated learning. In *ICLR*, 2024.

[32] Ruixiang Tang, Xiaotian Han, Xiaoqian Jiang, and Xia Hu. Does synthetic data generation of llms help clinical text mining? *arXiv preprint arXiv:2303.04360*, 2023.

[33] Chunlin Tian, Zhan Shi, Zhijiang Guo, Li Li, and Chengzhong Xu. Hydralora: An asymmetric lora architecture for efficient fine-tuning. In *NeurIPS*, 2024.

[34] Omar Abdel Wahab, Azzam Mourad, Hadi Otrok, and Tarik Taleb. Federated machine learning: Survey, multi-level classification, desirable criteria and future directions in communication and networking systems. *IEEE CST*, pages 1342–1397, 2021.

[35] Ching Pui Wan and Qifeng Chen. Robust federated learning with attack-adaptive aggregation. In *IJCAI Workshop*, 2021.

[36] Hanqing Wang, Yixia Li, Shuo Wang, Guanhua Chen, and Yun Chen. Milora: Harnessing minor singular components for parameter-efficient llm finetuning. *arXiv preprint arXiv:2406.09044*, 2024.

[37] Wenxuan Wang, Zizhan Ma, Zheng Wang, Chenghan Wu, Wenting Chen, Xiang Li, and Yixuan Yuan. A survey of llm-based agents in medicine: How far are we from baymax? In *ACL*, 2025.

[38] 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. In *NeurIPS*, 2024.

[39] Taiqiang Wu, Jiahao Wang, Zhe Zhao, and Ngai Wong. Mixture-of-subspaces in low-rank adaptation. In *EMNLP*, 2024.

[40] Rui Ye, Wenhao Wang, Jingyi Chai, Dihan Li, Zexi Li, Yinda Xu, Yaxin Du, Yanfeng Wang, and Siheng Chen. Openfedllm: Training large language models on decentralized private data via federated learning. In *ACM SIGKDD*, 2024.

[41] Liping Yi, Han Yu, Gang Wang, Xiaoguang Liu, and Xiaoxiao Li. pfedlora: Model-heterogeneous personalized federated learning with lora tuning. *arXiv preprint arXiv:2310.13283*, 2023.

[42] 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.

[43] Juzheng Zhang, Jiacheng You, Ashwinee Panda, and Tom Goldstein. Lori: Reducing cross-task interference in multi-task low-rank adaptation. *arXiv preprint arXiv:2504.07448*, 2025.

[44] Tuo Zhang, Tiantian Feng, Samiul Alam, Mi Zhang, Shrikanth S Narayanan, and Salman Avestimehr. Gpt-fl: Generative pre-trained model-assisted federated learning. *arXiv preprint arXiv:2306.02210*, 2023.

[45] Deyao Zhu, Jun Chen, Xiaoqian Shen, Xiang Li, and Mohamed Elhoseiny. Minigpt-4: Enhancing vision-language understanding with advanced large language models. *arXiv preprint arXiv:2304.10592*, 2023.

[46] Weiming Zhuang, Chen Chen, and Lingjuan Lyu. When foundation model meets federated learning: Motivations, challenges, and future directions. *arXiv preprint arXiv:2306.15546*, 2023.

A Proof of Aggregated Bias

We provide a formal proof that factor-wise aggregation in federated LoRA introduces aggregation inconsistency. Recall that the ideal federated update is given by

$$\Delta \boldsymbol{W}_g = \sum_{k=1}^{K} \alpha_k \boldsymbol{B}_k \boldsymbol{A}_k,$$

while FedIT-style aggregation computes

$$\Delta \boldsymbol{W}_g^{'} = \Big(\sum_{k=1}^{K} \alpha_k \boldsymbol{B}_k \Big) \Big(\sum_{k=1}^{K} \alpha_k \boldsymbol{A}_k \Big).$$

Expanding the aggregated factors, we have

$$\sum_{k=1}^{K} \alpha_k \boldsymbol{B}_k = \alpha_1 \boldsymbol{B}_1 + \alpha_2 \boldsymbol{B}_2 + \cdots + \alpha_K \boldsymbol{B}_K, \quad \sum_{k=1}^{K} \alpha_k \boldsymbol{A}_k = \alpha_1 \boldsymbol{A}_1 + \alpha_2 \boldsymbol{A}_2 + \cdots + \alpha_K \boldsymbol{A}_K.$$

Taking the product yields

$$\begin{aligned} \Delta \boldsymbol{W}_g^{'} &= (\alpha_1 \boldsymbol{B}_1 + \alpha_2 \boldsymbol{B}_2 + \cdots + \alpha_K \boldsymbol{B}_K)(\alpha_1 \boldsymbol{A}_1 + \alpha_2 \boldsymbol{A}_2 + \cdots + \alpha_K \boldsymbol{A}_K) \\ &= \alpha_1^2 \boldsymbol{B}_1 \boldsymbol{A}_1 + \alpha_2^2 \boldsymbol{B}_2 \boldsymbol{A}_2 + \cdots + \alpha_K^2 \boldsymbol{B}_K \boldsymbol{A}_K + \sum_{i \neq j} \alpha_i \alpha_j \boldsymbol{B}_i \boldsymbol{A}_j. \end{aligned}$$

In contrast, the ideal aggregated update contains only the diagonal terms

$$\Delta \boldsymbol{W}_g = \alpha_1 \boldsymbol{B}_1 \boldsymbol{A}_1 + \alpha_2 \boldsymbol{B}_2 \boldsymbol{A}_2 + \cdots + \alpha_K \boldsymbol{B}_K \boldsymbol{A}_K.$$

Define the aggregation noise as the deviation between the FedIT-style update and the ideal update:

$$\boldsymbol{\mathcal{N}} \triangleq \Delta \boldsymbol{W}_g^{'} - \Delta \boldsymbol{W}_g.$$

Using the expansions above, we obtain

$$\begin{aligned} \boldsymbol{\mathcal{N}} &= \left(\sum_{k=1}^{K} \alpha_k^2 \boldsymbol{B}_k \boldsymbol{A}_k + \sum_{i \neq j} \alpha_i \alpha_j \boldsymbol{B}_i \boldsymbol{A}_j \right) - \sum_{k=1}^{K} \alpha_k \boldsymbol{B}_k \boldsymbol{A}_k \\ &= \sum_{k=1}^{K} (\alpha_k^2 - \alpha_k) \boldsymbol{B}_k \boldsymbol{A}_k \; + \; \sum_{i \neq j} \alpha_i \alpha_j \boldsymbol{B}_i \boldsymbol{A}_j. \end{aligned} \tag{13}$$

The first term shows that ***factor-wise aggregation*** distorts the weighting of each client update (from α_k to α_k^2), and the second term introduces ***cross-client interaction terms*** $\boldsymbol{B}_i \boldsymbol{A}_j$ that do not appear in the ideal update. Therefore, unless restrictive conditions hold (e.g., $K = 1$ or all clients share identical LoRA factors so that cross terms cancel), we have $\boldsymbol{\mathcal{N}} \neq \boldsymbol{0}$ and thus $\Delta \boldsymbol{W}_g^{'} \neq \Delta \boldsymbol{W}_g$.

To highlight that the noise can grow with the number of clients and heterogeneity, consider the uniform weighting case $\alpha_k = \frac{1}{K}$. Then Equation (13) becomes

$$\boldsymbol{\mathcal{N}} = -\frac{K-1}{K^2} \sum_{k=1}^{K} \boldsymbol{B}_k \boldsymbol{A}_k \; + \; \frac{1}{K^2} \sum_{i \neq j} \boldsymbol{B}_i \boldsymbol{A}_j. \tag{14}$$

If client updates are heterogeneous so that cross-client products are non-negligible, the second term in (14) can dominate as K increases due to the $K(K-1)$ cross terms. For example, under a mild boundedness assumption $\|\boldsymbol{B}_k \boldsymbol{A}_j\|_F \leq M$ for all k, j,

$$\|\boldsymbol{\mathcal{N}}\|_F \leq \frac{K-1}{K^2} \sum_{k=1}^{K} \|\boldsymbol{B}_k \boldsymbol{A}_k\|_F + \frac{K(K-1)}{K^2} M = \frac{K-1}{K^2} \sum_{k=1}^{K} \|\boldsymbol{B}_k \boldsymbol{A}_k\|_F + \frac{K-1}{K} M,$$

which indicates that the cross-term component scales as $\mathcal{O}(1)$ in K (i.e., does not vanish) in the worst case. In contrast, the ideal update $\Delta \boldsymbol{W}_g$ is a weighted sum without cross-client interaction terms.

In standard federated optimization, $\Delta \boldsymbol{W}_g = \sum_{k=1}^{K} \alpha_k \Delta \boldsymbol{W}_k$ is an unbiased aggregation of client updates that aligns with the FedAvg update rule, and thus admits convergence guarantees under common assumptions (e.g., smoothness and bounded gradient variance). However, $\Delta \boldsymbol{W}_g^{'} = \Delta \boldsymbol{W}_g + \boldsymbol{\mathcal{N}}$ adds a structured perturbation $\boldsymbol{\mathcal{N}}$ that is generally *not* a weighted sum of valid client updates. Consequently, the global update may deviate from a descent direction, which can slow convergence and degrade generalization, especially under heterogeneous client distributions where the cross-client terms in (13) are amplified.

B Notation Claim.

Table 5: **Notation used in FLoRA-Chef**. Summary of key variables and operations in our method. The *Definition* column indicates where each symbol first appears in the main text.

Symbol	Description	Definition
$\boldsymbol{A}_g, \boldsymbol{B}_g$	Aggregated LoRA matrices	Eq. (1)
$\boldsymbol{A}_k, \boldsymbol{B}_k$	Client-k LoRA matrices	Eq. (1)
K	Number of participating clients	Eq. (1)
$\boldsymbol{W}$	Adapted model weights with LoRA	Eq. (2)
$\boldsymbol{W}_0$	Frozen pre-trained base model weights	Eq. (2)
$\boldsymbol{W}_g$	Global model weights	Eq. (3)
$\boldsymbol{\mathcal{N}}$	Aggregation noise induced by factor-wise averaging	Eq. (5)
t	Communication round index	Eq. (6)
$\mathcal{L}_k$	Local training objective of client k	Eq. (6)
$\mathcal{M}_k^t$	Client-k trainable local LoRA matrix at round t	Eq. (7)
$\mathcal{M}_g$	Server-side reference matrix	Eq. (8)
m^t	State signal indicating the target matrix at round t	Eq. (7)
$\varphi_{\mathcal{M}}^t$	Update dispersion over $\{\Delta \mathcal{M}_k^t\}_{k=1}^K$	Eq. (9)
α_k	Aggregation weight of client k	Eq. (1)
$\tilde{s}_{A,k}, \tilde{s}_{B,k}$	Rescaled cosine similarity scores in $[0, 1]$	Eq. (11–12)
$\alpha_{A,k}, \alpha_{B,k}$	Selective aggregation weights for $\boldsymbol{A}$ / $\boldsymbol{B}$	Eq. (11–12)
τ	Temperature parameter in softmax weighting	Eq. (11–12)

C Additional Experiments and Analysis

We further examine the two decisions underlying FLoRA-Chef: when to update each LoRA factor and how to aggregate client updates for the selected factor. The additional studies distinguish adaptive switching from a preference for more frequent B updates, characterize the geometry of induced updates, and investigate sensitivity to client heterogeneity and initialization. We also report protocol-level communication and storage costs. Throughout, $\Delta W = BA$ denotes the low-rank adapter, and A and B denote its two factors.

C.1 Adaptive Switching versus Fixed and Randomized Schedules

Fixed schedules with more frequent B updates. A possible explanation for the gains of FLFire is that it simply allocates more rounds to B. To examine this explanation, we compare FLFire with six fixed schedules on MNLI and MRPC under the same training setting. A schedule denoted by $mB\,nA$ repeatedly allocates m rounds to B and n rounds to A. As shown in Table 6, the strongest fixed schedule depends on the task: $3B\,1A$ is strongest on MNLI, whereas $2B\,1A$ is strongest on MRPC. FLFire achieves higher accuracy than every tested fixed schedule on both tasks. These results support adapting the schedule to the evolving optimization dynamics rather than selecting one fixed ratio across tasks.

Randomized schedules with identical update counts. To isolate update ordering from the realized ratio, we retain exactly the numbers of A and B rounds selected by FLFire and randomly permute their order. The realized ratios are $A : B = 7 : 18$ for MNLI and $A : B = 5 : 20$ for MRPC. We

Table 6: Accuracy (%) under fixed schedules with more frequent B updates. These comparisons vary the update schedule; they do not enforce an identical realized A/B ratio.

Schedule	MNLI	MRPC
$2B\,1A$	79.67	85.59
$3B\,1A$	83.87	78.77
$3B\,2A$	82.00	83.09
$4B\,1A$	82.87	82.35
$4B\,2A$	82.43	82.65
$4B\,3A$	82.53	82.65
FLFire	**85.30**	**87.65**

Table 7: Accuracy (%) with identical realized A/B update counts. Randomized results are averaged over five permutations.

Method	MNLI $(7:18)$	MRPC $(5:20)$
Randomized matched-ratio	82.62	83.17
FLFire	**85.30**	**87.65**

evaluate five random permutations and report their mean accuracy. Consequently, the factor-update counts and the computation and communication budgets are identical, while the placement of the updates differs. Table 7 shows gains of 2.68 and 4.48 percentage points for FLFire on MNLI and MRPC, respectively. The advantage therefore cannot be explained by the update ratio alone. This experiment evaluates the importance of scheduling under the tested settings; it does not imply that every random ordering is inferior.

C.2 Controlled Comparisons from Identical Checkpoints

Protocol and matrix-specific dispersion. We compare alternative continuations from identical checkpoints to examine whether following the switching signal improves subsequent optimization. The fixed continuation repeatedly updates the current factor, whereas FLFire follows its dispersion-based switching rule. Each paired comparison uses the same number of update rounds. The initial study includes two QNLI checkpoints, followed by six additional comparisons covering early, middle, and late stages on MRPC and MNLI, for eight comparisons in total.

Dispersion changes are defined for the same factor at the two endpoints. If the path begins and ends with factor $M \in \{A, B\}$, we compute

$$\Delta\phi_M = \phi_{M,\mathrm{end}} - \phi_{M,\mathrm{start}}. \tag{15}$$

For example, both BAB and BBB use the endpoint change of ϕ_B. Similarly, paths beginning and ending with A use ϕ_A. Intermediate rounds change the model state, but the dispersions of A and B are never pooled into one trajectory-level statistic. Validation-loss changes are also computed as endpoint minus starting value.

QNLI results. Table 8 shows that FLFire produces better accuracy and validation-loss changes at both checkpoints. At the early checkpoint, inserting an A update between B updates increases accuracy by 4.33 points and reduces validation loss by 0.202, whereas continuing with B yields a 2.00-point accuracy gain and a 0.093 loss increase. At the middle checkpoint, continuing with A causes a 20.83-point accuracy drop and a 4.466 loss increase. FLFire instead improves accuracy by 4.50 points and limits the loss increase to 0.602. The latter comparison illustrates a relative benefit rather than a loss decrease: validation loss still rises under FLFire.

Extension across tasks and training stages. In all six additional pairs in Table 9, FLFire yields a smaller validation-loss increase or a larger reduction than the fixed continuation. The magnitude varies considerably: the early MNLI difference is small, while the middle MNLI comparison changes a loss increase into a decrease. Together with QNLI, these results extend the observation across three tasks and multiple training stages, without establishing a universal monotonic relationship between dispersion and optimization quality.

Table 8: Controlled continuations on QNLI from early (approximately 25% of training) and middle (approximately 50%) checkpoints. Accuracy changes are in percentage points. Each FLFire continuation is compared with a fixed continuation of equal length.

Stage	Strategy	Path	$\Delta\phi_M$	ΔAcc.	$\Delta\mathcal{L}_{\text{val}}$
Early	Fixed	BBB	$+0.0138$	$+2.00$	$+0.093$
	FLFire	BAB	-0.0257	$+4.33$	-0.202
Middle	Fixed	AAAAAAAAAA	-0.1811	-20.83	$+4.466$
	FLFire	ABBBBBBBBA	$+0.0223$	$+4.50$	$+0.602$

Table 9: Six additional controlled checkpoint comparisons on MRPC and MNLI. Smaller validation-loss changes indicate a better continuation relative to the same starting checkpoint.

Task	Stage	Strategy	Path	$\Delta\phi_M$	$\Delta\mathcal{L}_{\text{val}}$
MRPC	Early	Fixed	BBB	$+0.164852$	$+0.277000$
		FLFire	BAB	-0.062042	-0.132600
	Middle	Fixed	AAAAAA	-0.000011	$+0.261665$
		FLFire	ABBBBA	$+0.004022$	$+0.097105$
	Late	Fixed	AAAA	-0.003253	$+0.229960$
		FLFire	ABBA	$+0.005448$	$+0.066135$
MNLI	Early	Fixed	AAAAA	-0.068446	$+1.065445$
		FLFire	ABBBA	$+0.073315$	$+1.057887$
	Middle	Fixed	BBBB	$+0.030241$	$+1.076817$
		FLFire	BAAB	-0.051495	-0.032428
	Late	Fixed	BBB	$+0.002169$	$+0.095004$
		FLFire	BAB	-0.005848	-0.015568

Interpretation of the two factor-specific patterns. For the tested A-endpoint paths, repeated A updates reduce or nearly preserve A dispersion but yield worse loss changes. Introducing an intermediate B phase leads to higher endpoint A dispersion and better optimization. This pattern is consistent with retaining useful cross-client variation instead of repeatedly concentrating already aligned updates. For the tested B-endpoint paths, repeated B updates increase B dispersion and yield worse loss changes; an intermediate A phase reduces endpoint B dispersion and improves the continuation. This is consistent with consolidating shared information before further client-specific adaptation. These are interpretations of controlled trajectory comparisons. Endpoint dispersion is distinct from the instantaneous change that triggers a switch, and the results do not prove that dispersion alone causally determines update quality.

C.3 Quantitative Geometry of Induced Client Updates

To complement qualitative visualizations, we measure the average pairwise cosine similarity between client updates (client–client similarity) and the average cosine similarity between the aggregated global update and each client update (global–client similarity). The similarities are computed in the induced weight-update space, rather than directly between the low-rank factors. Specifically,

$$\Delta W_k = \begin{cases} B_g\,\Delta A_k, & \text{for an } A\text{-update round,} \\ \Delta B_k\,A_g, & \text{for a } B\text{-update round,} \end{cases} \tag{16}$$

where the complementary global factor is frozen at the start of the round.

Table 10 places FLoRA-Chef between FedIT and RoLoRA in client–client alignment. Its global–client similarity is close to that of FedIT and below that of RoLoRA. The statistics are consistent with retaining directional variation while regulating disagreement, rather than uniformly maximizing alignment. They do not by themselves establish aggregation quality or attribute a separate geometric effect to each module. We assess optimization quality independently through the checkpoint comparisons and downstream accuracy.

Table 10: Cosine-similarity statistics in the induced update space. These quantities characterize geometry; neither a larger value nor an intermediate value is assumed to be intrinsically optimal.

Method	Client–client	Global–client
FedIT	0.023760	0.423935
RoLoRA	0.043867	0.443027
FLoRA-Chef	0.033338	0.424005

Table 11: Accuracy (%) of FLSauce and Anti-FLSauce under different client heterogeneity levels. Bold indicates the better result within each task and β.

β	Strategy	QQP	MRPC	MNLI
0.3	Anti-FLSauce	**76.60**	81.18	75.73
	FLSauce	73.70	**84.22**	**84.50**
1.0	Anti-FLSauce	83.73	85.78	83.60
	FLSauce	**91.93**	**86.17**	**87.97**

C.4 Client Heterogeneity and Task-dependent Reweighting

FLSauce versus reversed reweighting. Anti-FLSauce reverses the matrix-specific reweighting direction of FLSauce. We compare the two strategies on QQP, MRPC, and MNLI at $\beta \in \{0.3, 1.0\}$, where the smaller value corresponds to stronger client heterogeneity. Table 11 shows that FLSauce outperforms Anti-FLSauce on MRPC and MNLI at both heterogeneity levels. QQP is an exception at $\beta = 0.3$, where Anti-FLSauce is better by 2.90 points. At $\beta = 1.0$, FLSauce is better on all three tasks, including an 8.20-point gain on QQP. Thus, the reversal on QQP depends on the partition setting and is not a universal preference for reversed reweighting.

Distribution of difficult QQP examples. To further examine the QQP exception, we count high-lexical-overlap negative pairs, referred to here as hard negatives, across six clients. Such pairs require distinguishing non-paraphrases despite their shared words. Table 12 compares the partitions with $\beta = 0.5$ and $\beta = 1.0$. These settings are distinct from the $\beta = 0.3$ comparison in Table 11. Hard-negative proportions range from 6.2% to 12.0% across clients at $\beta = 0.5$, compared with 5.7% to 8.0% at $\beta = 1.0$.

The wider range under $\beta = 0.5$ suggests that clients receive different exposure to challenging lexical and structural patterns. Preserving their distinct directions may therefore be beneficial on QQP. However, these counts provide descriptive evidence rather than a direct measurement of update complementarity. Moreover, the total hard-negative prevalence also changes, from 8.93% to 6.90%. The comparison therefore does not isolate uneven allocation from overall example difficulty. We interpret it as context for the task-dependent result, rather than a causal explanation of the Anti-FLSauce advantage.

Comparison with baselines under stronger heterogeneity. We also evaluate the full method against FedIT and RoLoRA at $\beta = 0.3$. Table 13 shows that FLoRA-Chef improves over FedIT by 5.00 points on MRPC and 5.40 points on MNLI. The corresponding improvements over RoLoRA are 6.33 and 10.77 points. These comparisons support the effectiveness of the combined scheduling and aggregation procedure under the tested strongly heterogeneous setting.

C.5 Sensitivity to Reversed LoRA Initialization

To test whether the observed asymmetry depends on the standard initialization, we reverse it by initializing A to zero and B randomly. We compare RoLoRA, two fixed asymmetric schedules, and FLFire on QQP and RTE. Table 14 shows that the large performance difference between $2A\,1B$ and $2B\,1A$ persists after reversing initialization. FLFire attains the highest accuracy among the tested methods on both tasks, although its advantage over $2B\,1A$ is modest: 0.50 points on QQP and 0.22 points on RTE. These results support the robustness of FLFire to this initialization change; they do not establish invariance to arbitrary initialization schemes.

Table 12: QQP hard-negative counts and within-client proportions. The total row gives the pooled count and proportion. The two partitions differ in both the total prevalence and the cross-client distribution of these examples.

	$\beta = 0.5$		$\beta = 1.0$	
Client	Count	Proportion	Count	Proportion
0	84	8.4%	80	8.0%
1	62	6.2%	57	5.7%
2	101	10.1%	61	6.1%
3	120	12.0%	64	6.4%
4	84	8.4%	78	7.8%
5	85	8.5%	74	7.4%
Total	536	8.93%	414	6.90%

Table 13: Accuracy (%) under stronger client heterogeneity, $\beta = 0.3$.

Method	MRPC	MNLI
FedIT	79.22	79.10
RoLoRA	77.89	73.73
FLoRA-Chef	**84.22**	**84.50**

C.6 Communication and Storage Accounting

We quantify adapter communication under the RoBERTa-large setting with LoRA rank $r = 8$, 48 query/value LoRA modules, and FP32 parameters. Each factor occupies 1.50 MiB, and the complete adapter occupies 3.00 MiB. Table 15 reports per-client, per-round communication and persistent LoRA state under the stated protocol-level accounting.

FedIT and FlexLoRA exchange both factors, yielding 6.00 MiB of bidirectional communication per client per round. FFA-LoRA, RoLoRA, and FLoRA-Chef exchange one factor, yielding 3.00 MiB. Thus, FLoRA-Chef reduces adapter communication by 50% relative to FedIT under this setting. The reported FLoRA-Chef storage counts a 1.50 MiB current factor and a 1.50 MiB reference factor, totaling 3.00 MiB. This accounting is not a measurement of peak runtime memory: it does not include temporary client-update buffers, optimizer state, activations, or other implementation-dependent allocations.

To summarize accuracy relative to these costs, we normalize each quantity by its FedIT value:

$$R_{\text{comm}} = \frac{\text{Acc}/\text{Acc}_{\text{FedIT}}}{C/C_{\text{FedIT}}}, \qquad R_{\text{storage}} = \frac{\text{Acc}/\text{Acc}_{\text{FedIT}}}{S/S_{\text{FedIT}}}, \tag{17}$$

where C is bidirectional per-round adapter communication, and S includes the shared RoBERTa-large backbone and the counted persistent LoRA state. The reported accuracy averages over seven datasets and two non-IID settings.

FLoRA-Chef increases the reported average accuracy from 83.7164% to 88.2371%, corresponding to a gain of 4.5207 percentage points, while halving adapter communication. Its normalized ratios are 2.1080 for communication and 1.0540 for storage. Because the storage denominator includes the large shared backbone, differences in the counted adapter state have limited influence on that ratio. The result should therefore be read as an accuracy–cost trade-off under the specified accounting.

C.7 Further Interpretation of Scheduling and Reweighting

Why switching rules produce different update frequencies. Each FLFire pattern specifies two transition conditions. $A \uparrow$ and $A \downarrow$ switch from A to B when A dispersion increases or decreases, respectively; $B \uparrow$ and $B \downarrow$ analogously switch from B to A. The frequencies emerge from the interaction between the selected conditions and the training dynamics. Under $A \uparrow B \downarrow$, an increase in the dispersion of relatively aligned A updates can shorten an A stage, while a B stage can persist until its dispersion decreases. This can produce the observed greater allocation to B. Reversing the A

Table 14: Accuracy (%) with zero-initialized A and randomly initialized B. The adaptive method evaluated here is FLFire.

Method	QQP	RTE
RoLoRA	79.83	56.68
$2A\,1B$	71.33	68.95
$2B\,1A$	84.17	78.70
FLFire	**84.67**	**78.92**

Table 15: Communication and storage overhead, and accuracy–cost ratios of different methods. Upload and download are measured in MiB per client per round. Persistent LoRA state excludes the shared backbone, while the storage ratio includes it. All ratios are normalized to FedIT.

Method	Overhead (MiB)			Accuracy–Cost Ratio	
	Upload	Download	LoRA State	R_{comm}	R_{storage}
FedIT	3.00	3.00	3.00	1.0000	1.0000
FFA-LoRA	1.50	1.50	1.50	2.0146	1.0084
RoLoRA	1.50	1.50	1.50	2.0063	1.0043
FlexLoRA	3.00	3.00	3.00	1.0275	1.0275
FLoRA-Chef	1.50	1.50	3.00	**2.1080**	**1.0540**

condition can prolong A stages, and $B \uparrow$ can shorten B stages when their dispersion grows. These observations explain the recorded trajectories; they do not prescribe a universal A/B ratio.

Temperature in FLSauce. The temperature τ controls the concentration of the client aggregation weights. A smaller value produces sharper weights and can make the aggregate depend heavily on a few clients. In the reported comparison, $\tau = 0.05$ is slightly worse than $\tau = 0.1$ and $\tau = 0.15$. One possible explanation is that sharp weights repeatedly applied during frequent B rounds suppress useful contributions from heterogeneous clients. Larger tested temperatures distribute weight more smoothly. This interpretation is consistent with the observed sensitivity but was not isolated through an additional causal intervention. The temperature results belong to the original evaluation; this paragraph adds interpretation rather than reporting a new sweep.

Relationship to LoRA-A^2. Both approaches distinguish the LoRA factors, but they address different decisions. LoRA-A^2 uses a predefined odd–even alternation and adaptively selects rank components within the factor scheduled for a round. FLFire instead chooses the factor-update timing from cross-client dynamics, while FLSauce assigns matrix-specific weights across clients. The distinction is therefore between rank-component selection within a schedule and adaptive factor scheduling together with client reweighting. This conceptual distinction does not imply an empirical advantage over LoRA-A^2, which is not evaluated in the additional comparisons above.

NeurIPS Paper Checklist

The checklist is designed to encourage best practices for responsible machine learning research, addressing issues of reproducibility, transparency, research ethics, and societal impact. Do not remove the checklist: **The papers not including the checklist will be desk rejected.** The checklist should follow the references and follow the (optional) supplemental material. The checklist does NOT count towards the page limit.

Please read the checklist guidelines carefully for information on how to answer these questions. For each question in the checklist:

- You should answer [Yes], [No], or [N/A].
- [N/A] means either that the question is Not Applicable for that particular paper or the relevant information is Not Available.
- Please provide a short (1–2 sentence) justification right after your answer (even for [N/A]).

The checklist answers are an integral part of your paper submission. They are visible to the reviewers, area chairs, senior area chairs, and ethics reviewers. You will also be asked to include it (after eventual revisions) with the final version of your paper, and its final version will be published with the paper.

The reviewers of your paper will be asked to use the checklist as one of the factors in their evaluation. While [Yes] is generally preferable to [No], it is perfectly acceptable to answer [No] provided a proper justification is given (e.g., error bars are not reported because it would be too computationally expensive" or "we were unable to find the license for the dataset we used"). In general, answering [No] or [N/A] is not grounds for rejection. While the questions are phrased in a binary way, we acknowledge that the true answer is often more nuanced, so please just use your best judgment and write a justification to elaborate. All supporting evidence can appear either in the main paper or the supplemental material, provided in appendix. If you answer [Yes] to a question, in the justification please point to the section(s) where related material for the question can be found.

IMPORTANT, please:

- **Delete this instruction block, but keep the section heading "NeurIPS Paper Checklist"**,
- **Keep the checklist subsection headings, questions/answers and guidelines below.**
- **Do not modify the questions and only use the provided macros for your answers**.

1. **Claims**

   Question: Do the main claims made in the abstract and introduction accurately reflect the paper's contributions and scope?

   Answer: [Yes].

   Justification: The abstract and introduction clearly state the main contributions, including the aggregation-bias analysis, the FLFire and FLSauce modules, and the empirical improvements. The claims are supported by the theoretical analysis and experiments reported in the paper.

   Guidelines:

   - The answer [N/A] means that the abstract and introduction do not include the claims made in the paper.
   - The abstract and/or introduction should clearly state the claims made, including the contributions made in the paper and important assumptions and limitations. A [No] or [N/A] answer to this question will not be perceived well by the reviewers.
   - The claims made should match theoretical and experimental results, and reflect how much the results can be expected to generalize to other settings.
   - It is fine to include aspirational goals as motivation as long as it is clear that these goals are not attained by the paper.

2. **Limitations**

   Question: Does the paper discuss the limitations of the work performed by the authors?

Answer: [Yes].

Justification: The paper includes a discussion of limitations in Section 4.3, including the need to cache previous aggregated matrices and compute cross-client similarities, which introduces additional computation and storage overhead.

Guidelines:

- The answer [N/A] means that the paper has no limitation while the answer [No] means that the paper has limitations, but those are not discussed in the paper.
- The authors are encouraged to create a separate "Limitations" section in their paper.
- The paper should point out any strong assumptions and how robust the results are to violations of these assumptions (e.g., independence assumptions, noiseless settings, model well-specification, asymptotic approximations only holding locally). The authors should reflect on how these assumptions might be violated in practice and what the implications would be.
- The authors should reflect on the scope of the claims made, e.g., if the approach was only tested on a few datasets or with a few runs. In general, empirical results often depend on implicit assumptions, which should be articulated.
- The authors should reflect on the factors that influence the performance of the approach. For example, a facial recognition algorithm may perform poorly when image resolution is low or images are taken in low lighting. Or a speech-to-text system might not be used reliably to provide closed captions for online lectures because it fails to handle technical jargon.
- The authors should discuss the computational efficiency of the proposed algorithms and how they scale with dataset size.
- If applicable, the authors should discuss possible limitations of their approach to address problems of privacy and fairness.
- While the authors might fear that complete honesty about limitations might be used by reviewers as grounds for rejection, a worse outcome might be that reviewers discover limitations that aren't acknowledged in the paper. The authors should use their best judgment and recognize that individual actions in favor of transparency play an important role in developing norms that preserve the integrity of the community. Reviewers will be specifically instructed to not penalize honesty concerning limitations.

3. **Theory assumptions and proofs**

   Question: For each theoretical result, does the paper provide the full set of assumptions and a complete (and correct) proof?

   Answer: [Yes].

   Justification: The paper provides the derivation of aggregation bias in the main text and gives a detailed proof in Appendix A, including the expansion of factor-wise aggregation noise and its behavior under client heterogeneity.

   Guidelines:

   - The answer [N/A] means that the paper does not include theoretical results.
   - All the theorems, formulas, and proofs in the paper should be numbered and cross-referenced.
   - All assumptions should be clearly stated or referenced in the statement of any theorems.
   - The proofs can either appear in the main paper or the supplemental material, but if they appear in the supplemental material, the authors are encouraged to provide a short proof sketch to provide intuition.
   - Inversely, any informal proof provided in the core of the paper should be complemented by formal proofs provided in appendix or supplemental material.
   - Theorems and Lemmas that the proof relies upon should be properly referenced.

4. **Experimental result reproducibility**

   Question: Does the paper fully disclose all the information needed to reproduce the main experimental results of the paper to the extent that it affects the main claims and/or conclusions of the paper (regardless of whether the code and data are provided or not)?

Answer: [Yes].

Justification: The paper reports the foundation models, datasets, heterogeneity settings, client number, sample sizes, learning rates, batch sizes, local training steps, and evaluation protocol needed to reproduce the main experiments.

Guidelines:

- The answer [N/A] means that the paper does not include experiments.
- If the paper includes experiments, a [No] answer to this question will not be perceived well by the reviewers: Making the paper reproducible is important, regardless of whether the code and data are provided or not.
- If the contribution is a dataset and/or model, the authors should describe the steps taken to make their results reproducible or verifiable.
- Depending on the contribution, reproducibility can be accomplished in various ways. For example, if the contribution is a novel architecture, describing the architecture fully might suffice, or if the contribution is a specific model and empirical evaluation, it may be necessary to either make it possible for others to replicate the model with the same dataset, or provide access to the model. In general. releasing code and data is often one good way to accomplish this, but reproducibility can also be provided via detailed instructions for how to replicate the results, access to a hosted model (e.g., in the case of a large language model), releasing of a model checkpoint, or other means that are appropriate to the research performed.
- While NeurIPS does not require releasing code, the conference does require all submissions to provide some reasonable avenue for reproducibility, which may depend on the nature of the contribution. For example
  (a) If the contribution is primarily a new algorithm, the paper should make it clear how to reproduce that algorithm.
  (b) If the contribution is primarily a new model architecture, the paper should describe the architecture clearly and fully.
  (c) If the contribution is a new model (e.g., a large language model), then there should either be a way to access this model for reproducing the results or a way to reproduce the model (e.g., with an open-source dataset or instructions for how to construct the dataset).
  (d) We recognize that reproducibility may be tricky in some cases, in which case authors are welcome to describe the particular way they provide for reproducibility. In the case of closed-source models, it may be that access to the model is limited in some way (e.g., to registered users), but it should be possible for other researchers to have some path to reproducing or verifying the results.

5. **Open access to data and code**

Question: Does the paper provide open access to the data and code, with sufficient instructions to faithfully reproduce the main experimental results, as described in supplemental material?

Answer: [No].

Justification: The current submission does not include released code or detailed execution commands. However, the experiments are conducted on standard public benchmarks, and the main experimental settings are described in the paper.

Guidelines:

- The answer [N/A] means that paper does not include experiments requiring code.
- Please see the NeurIPS code and data submission guidelines (`https://neurips.cc/public/guides/CodeSubmissionPolicy`) for more details.
- While we encourage the release of code and data, we understand that this might not be possible, so [No] is an acceptable answer. Papers cannot be rejected simply for not including code, unless this is central to the contribution (e.g., for a new open-source benchmark).
- The instructions should contain the exact command and environment needed to run to reproduce the results. See the NeurIPS code and data submission guidelines (`https://neurips.cc/public/guides/CodeSubmissionPolicy`) for more details.

- The authors should provide instructions on data access and preparation, including how to access the raw data, preprocessed data, intermediate data, and generated data, etc.
- The authors should provide scripts to reproduce all experimental results for the new proposed method and baselines. If only a subset of experiments are reproducible, they should state which ones are omitted from the script and why.
- At submission time, to preserve anonymity, the authors should release anonymized versions (if applicable).
- Providing as much information as possible in supplemental material (appended to the paper) is recommended, but including URLs to data and code is permitted.

6. **Experimental setting/details**

   Question: Does the paper specify all the training and test details (e.g., data splits, hyperparameters, how they were chosen, type of optimizer) necessary to understand the results?

   Answer: [Yes].

   Justification: Section 5.1 describes the datasets, foundation models, heterogeneity setting, training/test sample sizes, learning rates, batch size, number of clients, local training steps, and evaluation protocol.

   Guidelines:

   - The answer [N/A] means that the paper does not include experiments.
   - The experimental setting should be presented in the core of the paper to a level of detail that is necessary to appreciate the results and make sense of them.
   - The full details can be provided either with the code, in appendix, or as supplemental material.

7. **Experiment statistical significance**

   Question: Does the paper report error bars suitably and correctly defined or other appropriate information about the statistical significance of the experiments?

   Answer: [No].

   Justification: The paper reports mean scores over the last five communication rounds and provides convergence curves, but it does not report error bars, confidence intervals, or statistical significance tests.

   Guidelines:

   - The answer [N/A] means that the paper does not include experiments.
   - The authors should answer [Yes] if the results are accompanied by error bars, confidence intervals, or statistical significance tests, at least for the experiments that support the main claims of the paper.
   - The factors of variability that the error bars are capturing should be clearly stated (for example, train/test split, initialization, random drawing of some parameter, or overall run with given experimental conditions).
   - The method for calculating the error bars should be explained (closed form formula, call to a library function, bootstrap, etc.)
   - The assumptions made should be given (e.g., Normally distributed errors).
   - It should be clear whether the error bar is the standard deviation or the standard error of the mean.
   - It is OK to report 1-sigma error bars, but one should state it. The authors should preferably report a 2-sigma error bar than state that they have a 96% CI, if the hypothesis of Normality of errors is not verified.
   - For asymmetric distributions, the authors should be careful not to show in tables or figures symmetric error bars that would yield results that are out of range (e.g., negative error rates).
   - If error bars are reported in tables or plots, the authors should explain in the text how they were calculated and reference the corresponding figures or tables in the text.

8. **Experiments compute resources**

Question: For each experiment, does the paper provide sufficient information on the computer resources (type of compute workers, memory, time of execution) needed to reproduce the experiments?

Answer: [No].

Justification: The current paper does not provide detailed hardware information, memory usage, runtime, or total compute cost for reproducing each experiment.

Guidelines:

- The answer [N/A] means that the paper does not include experiments.
- The paper should indicate the type of compute workers CPU or GPU, internal cluster, or cloud provider, including relevant memory and storage.
- The paper should provide the amount of compute required for each of the individual experimental runs as well as estimate the total compute.
- The paper should disclose whether the full research project required more compute than the experiments reported in the paper (e.g., preliminary or failed experiments that didn't make it into the paper).

9. **Code of ethics**

Question: Does the research conducted in the paper conform, in every respect, with the NeurIPS Code of Ethics `https://neurips.cc/public/EthicsGuidelines`?

Answer: [Yes].

Justification: The research studies federated LoRA aggregation on standard language understanding and reasoning benchmarks. It does not involve human subjects, private user data, or unsafe data collection.

Guidelines:

- The answer [N/A] means that the authors have not reviewed the NeurIPS Code of Ethics.
- If the authors answer [No], they should explain the special circumstances that require a deviation from the Code of Ethics.
- The authors should make sure to preserve anonymity (e.g., if there is a special consideration due to laws or regulations in their jurisdiction).

10. **Broader impacts**

Question: Does the paper discuss both potential positive societal impacts and negative societal impacts of the work performed?

Answer: [Yes].

Justification: The paper discusses the positive impact of privacy-preserving federated adaptation and also notes privacy-related concerns such as parameter inversion and inference attacks in Section 4.3.

Guidelines:

- The answer [N/A] means that there is no societal impact of the work performed.
- If the authors answer [N/A] or [No], they should explain why their work has no societal impact or why the paper does not address societal impact.
- Examples of negative societal impacts include potential malicious or unintended uses (e.g., disinformation, generating fake profiles, surveillance), fairness considerations (e.g., deployment of technologies that could make decisions that unfairly impact specific groups), privacy considerations, and security considerations.
- The conference expects that many papers will be foundational research and not tied to particular applications, let alone deployments. However, if there is a direct path to any negative applications, the authors should point it out. For example, it is legitimate to point out that an improvement in the quality of generative models could be used to generate Deepfakes for disinformation. On the other hand, it is not needed to point out that a generic algorithm for optimizing neural networks could enable people to train models that generate Deepfakes faster.

- The authors should consider possible harms that could arise when the technology is being used as intended and functioning correctly, harms that could arise when the technology is being used as intended but gives incorrect results, and harms following from (intentional or unintentional) misuse of the technology.
- If there are negative societal impacts, the authors could also discuss possible mitigation strategies (e.g., gated release of models, providing defenses in addition to attacks, mechanisms for monitoring misuse, mechanisms to monitor how a system learns from feedback over time, improving the efficiency and accessibility of ML).

11. **Safeguards**

    Question: Does the paper describe safeguards that have been put in place for responsible release of data or models that have a high risk for misuse (e.g., pre-trained language models, image generators, or scraped datasets)?

    Answer: [N/A].

    Justification: The paper does not release a new high-risk pretrained model, image generator, or scraped dataset. It evaluates an aggregation method using existing foundation models and public benchmarks.

    Guidelines:

    - The answer [N/A] means that the paper poses no such risks.
    - Released models that have a high risk for misuse or dual-use should be released with necessary safeguards to allow for controlled use of the model, for example by requiring that users adhere to usage guidelines or restrictions to access the model or implementing safety filters.
    - Datasets that have been scraped from the Internet could pose safety risks. The authors should describe how they avoided releasing unsafe images.
    - We recognize that providing effective safeguards is challenging, and many papers do not require this, but we encourage authors to take this into account and make a best faith effort.

12. **Licenses for existing assets**

    Question: Are the creators or original owners of assets (e.g., code, data, models), used in the paper, properly credited and are the license and terms of use explicitly mentioned and properly respected?

    Answer: [No].

    Justification: The paper cites the existing models, methods, and benchmarks used in the experiments, but the current draft does not explicitly list the licenses or terms of use for all existing assets.

    Guidelines:

    - The answer [N/A] means that the paper does not use existing assets.
    - The authors should cite the original paper that produced the code package or dataset.
    - The authors should state which version of the asset is used and, if possible, include a URL.
    - The name of the license (e.g., CC-BY 4.0) should be included for each asset.
    - For scraped data from a particular source (e.g., website), the copyright and terms of service of that source should be provided.
    - If assets are released, the license, copyright information, and terms of use in the package should be provided. For popular datasets, `paperswithcode.com/datasets` has curated licenses for some datasets. Their licensing guide can help determine the license of a dataset.
    - For existing datasets that are re-packaged, both the original license and the license of the derived asset (if it has changed) should be provided.
    - If this information is not available online, the authors are encouraged to reach out to the asset's creators.

13. **New assets**

Question: Are new assets introduced in the paper well documented and is the documentation provided alongside the assets?

Answer: [N/A].

Justification: The paper does not introduce or release a new dataset, benchmark, model checkpoint, or software asset as a primary contribution.

Guidelines:

- The answer [N/A] means that the paper does not release new assets.
- Researchers should communicate the details of the dataset/code/model as part of their submissions via structured templates. This includes details about training, license, limitations, etc.
- The paper should discuss whether and how consent was obtained from people whose asset is used.
- At submission time, remember to anonymize your assets (if applicable). You can either create an anonymized URL or include an anonymized zip file.

14. **Crowdsourcing and research with human subjects**

Question: For crowdsourcing experiments and research with human subjects, does the paper include the full text of instructions given to participants and screenshots, if applicable, as well as details about compensation (if any)?

Answer: [N/A].

Justification: The paper does not involve crowdsourcing, user studies, or research with human subjects.

Guidelines:

- The answer [N/A] means that the paper does not involve crowdsourcing nor research with human subjects.
- Including this information in the supplemental material is fine, but if the main contribution of the paper involves human subjects, then as much detail as possible should be included in the main paper.
- According to the NeurIPS Code of Ethics, workers involved in data collection, curation, or other labor should be paid at least the minimum wage in the country of the data collector.

15. **Institutional review board (IRB) approvals or equivalent for research with human subjects**

Question: Does the paper describe potential risks incurred by study participants, whether such risks were disclosed to the subjects, and whether Institutional Review Board (IRB) approvals (or an equivalent approval/review based on the requirements of your country or institution) were obtained?

Answer: [N/A].

Justification: The paper does not involve human subjects, crowdsourcing participants, or personally identifiable data, so IRB approval is not applicable.

Guidelines:

- The answer [N/A] means that the paper does not involve crowdsourcing nor research with human subjects.
- Depending on the country in which research is conducted, IRB approval (or equivalent) may be required for any human subjects research. If you obtained IRB approval, you should clearly state this in the paper.
- We recognize that the procedures for this may vary significantly between institutions and locations, and we expect authors to adhere to the NeurIPS Code of Ethics and the guidelines for their institution.
- For initial submissions, do not include any information that would break anonymity (if applicable), such as the institution conducting the review.

16. **Declaration of LLM usage**

Question: Does the paper describe the usage of LLMs if it is an important, original, or non-standard component of the core methods in this research? Note that if the LLM is used only for writing, editing, or formatting purposes and does *not* impact the core methodology, scientific rigor, or originality of the research, declaration is not required.

Answer: [Yes].

Justification: The paper studies federated LoRA adaptation of foundation models and explicitly describes the use of RoBERTa-large and LLaMA as experimental backbones. No LLM is used as an autonomous annotator or evaluator for generating the main scientific results.

Guidelines:

- The answer [N/A] means that the core method development in this research does not involve LLMs as any important, original, or non-standard components.
- Please refer to our LLM policy in the NeurIPS handbook for what should or should not be described.