FEDRRA: RESIDUAL RANK ADAPTATION FOR HETEROGENEOUS FEDERATED FINE-TUNING

Anonymous authors
Paper under double-blind review

ABSTRACT

Combining Large Language Models (LLMs) with Low-Rank Adaptation (LoRA) and Federated Learning (FL) has emerged as a promising paradigm to enhance Parameter-Efficient Fine-Tuning (PEFT) while preserving data privacy. However, in practical deployments, where clients have heterogeneous resources and thus require different LoRA ranks for fine-tuning, the conventional paradigm of aggregating uniform LoRA adapters is no longer applicable, introducing significant challenges. While padding-based methods, such as zero-padding, have been proposed to address this issue, we find that they still suffer from two key challenges: limited adaptability to heterogeneous resource settings and aggregation noise. To address these challenges, we propose **Fed**erated **R**esidual **R**ank **A**daptation (FedRRA), a novel heterogeneous Federated Fine-Tuning (FFT) framework. Specifically, we design a new padding strategy, termed *Consensus-Aware Padding* (CAP), which performs weighted padding by combining rank-one signal strength with direction-aware soft gating, enabling improved adaptation to diverse resource-heterogeneous scenarios. In addition, we propose a residual correction method tailored for heterogeneous resource settings, which mitigates aggregation noise by adding a residual term to the frozen pretrained weights. Extensive experiments demonstrate that FedRRA consistently outperforms existing methods across different resource-heterogeneous settings.

1 INTRODUCTION

Large Language Models (LLMs) have demonstrated remarkable performance across a wide range of tasks (Achiam et al., 2023; Touvron et al., 2023a; Dong et al., 2023; Kelly et al., 2023; Chang et al., 2024). However, adapting pretrained LLMs to downstream tasks often requires fine-tuning in resource-constrained environments, such as some private healthcare clinics with limited memory and computational capacity (Singhal et al., 2023). To address this challenge, various Parameter-Efficient Fine-Tuning (PEFT) (Gao et al., 2025b; Houlsby et al., 2019; Liu et al., 2022; Li & Liang, 2021) methods have been proposed, among which Low-Rank Adaptation (LoRA) (Hu et al., 2022) is widely adopted for its substantial reduction in trainable parameters. Despite the progress in LLM fine-tuning, the data used for fine-tuning often raises privacy concerns in practice (Du et al., 2025; Babakniya et al., 2023; Qin et al., 2023). Federated learning (FL) therefore provides a viable solution for fine-tuning LLMs across distributed clients without sharing private data.

Previous works have explored integrating LoRA into the FL framework (Zhang et al., 2024a; Kuang et al., 2024; Xu et al., 2023; Ling et al., 2024; Nguyen et al., 2024; Wang et al., 2024a). For example FedIT (Zhang et al., 2024a), combines FedAvg (McMahan et al., 2017) algorithm with LoRA for efficient federated fine-tuning under homogeneous-client settings. However, these methods suffer from *aggregation noise*, where the actual aggregated update deviates from the ideal one. To address this issue, RoLoRA (Chen et al., 2025) reduces aggregation error by alternately training the A and B modules. FedEX-LoRA (Singhal et al., 2025) achieves exact updates by incorporating a residual term into the frozen pretrained weight matrix.

However, in practical scenarios, clients are often resource-heterogeneous, and resource-constrained clients can only afford limited LoRA ranks. Such heterogeneity breaks the standard global aggregation process, motivating federated fine-tuning methods that can adapt to heterogeneous clients. Several prior works have explored aggregation under client heterogeneity. For example, FLoRA (Wang et al.,

Reviewers: please read the Reviewer Guidelines (iclr.cc/Conferences/2027/ReviewerGuidelines) and the AI Policy for Reviewers (iclr.cc/Conferences/2027/AIPolicyForReviewers). If you used AI to expand, edit, or polish your review, please provide the input text to the LLM. Better still, consider skipping the LLM and submitting your original text: we, and the authors, are much more interested in your unedited thoughts than in what an LLM has to say. AI-assisted or not, you are putting your name and reputation behind your review: LLM-generated falsehoods, hallucinations or misrepresentations are subject to disciplinary action, which may include desk-rejecting all papers you have authored.

2024b) constructs the global adapter by stacking local adapters to support heterogeneous ranks, but this leads to communication costs that grow linearly with the number of clients. FlexLoRA (Bai et al., 2024), in contrast, applies SVD to the global weights and reallocates them based on each client's rank budget, but this incurs extra computational overhead and decomposition error.

In contrast, padding-based methods offer a simpler solution (Cho et al., 2024; Byun & Lee, 2025; Fan et al., 2025). The key idea is to expand all LoRA modules to a unified rank, apply standard FedAvg to obtain the global module, and then truncate it before redistributing it to clients for the next round of fine-tuning. For instance, HETLoRA (Cho et al., 2024) uses zero-padding to align LoRA modules across clients, while Byun & Lee (2025) proposes replication padding, which fills missing dimensions of low-rank clients using parameters from high-rank clients instead of zeros. These methods introduce no extra overhead and, unlike FLoRA and FlexLoRA, are naturally compatible with strategies for homogeneous settings. However, existing padding-based methods still have important limitations: zero-padding may dilute informative signals from high-rank clients, while replication-based padding is typically restricted to limited rank configurations, such as only two rank categories. More importantly, they overlook the *aggregation error* also observed in homogeneous settings. Due to space limitations, we defer the Related Works section to Appendix B.

In this paper, we experimentally compare several representative methods for heterogeneous federated fine-tuning, including existing padding-based approaches. Based on these insights, we propose **FedRRA** (**Fed**erated **R**esidual **R**ank **A**daptation). Specifically, we introduce a new padding strategy, *Consensus-Aware Padding* (CAP), which performs weighted padding by combining rank-one signal strength with direction-aware soft gating, enabling improved adaptation to diverse heterogeneous scenarios. In addition, we propose a residual correction method tailored to heterogeneous scenarios, which effectively mitigates the *aggregation noise* issue without incurring additional training cost. Our main contributions are summarized as follows:

- We design a novel padding strategy, termed *Consensus-Aware Padding* (CAP), which performs weighted padding by combining rank-one signal strength with direction-aware soft gating, enabling improved adaptation to diverse heterogeneous scenarios.
- We propose *Aggregation Residual Correction* (ARC), which mitigates *aggregation noise* by adding a residual term to the frozen pretrained weights, tailored to heterogeneous-rank settings where the server computes residuals based on each client's resource capacity.
- Through extensive experiments on three datasets and two evaluation benchmarks using TinyLlama, Qwen2.5-3B and Llama2-7B with three different heterogeneous rank allocation strategies, the results show that FedRRA consistently outperforms existing heterogeneous federated LoRA fine-tuning methods.

2 Preliminaries and Motivation

2.1 Low-Rank Adaptation

Low-Rank Adaptation (LoRA) (Hu et al., 2022) has emerged as a dominant parameter-efficient fine-tuning (PEFT) method that balances adaptation quality and resource consumption effectively (Huang et al., 2023; Lialin et al., 2023; Valipour et al., 2023). Concretely, given a pretrained weight matrix $W_0 \in \mathbb{R}^{m\times n}$, LoRA introduces a low-rank incremental update ΔW, which is factorized as the product of two low-rank matrices:

$$W = W_0 + \Delta W = W_0 + BA \tag{1}$$

where $B \in \mathbb{R}^{m\times r}$ and $A \in \mathbb{R}^{r\times n}$, with $r \ll \min(m, n)$. During fine-tuning, only A and B are updated while W_0 remains frozen, reducing the number of trainable parameters from mn to $(m+n)r$, thereby significantly reducing the parameter footprint and memory overhead.

2.2 Aggregation Noise in Federated Fine-Tuning LoRA

FedIT (Zhang et al., 2024a) integrates LoRA with FedAvg (McMahan et al., 2017) for homogeneous LoRA federated fine-tuning. Consider a federated system with N clients, where each client i holds a local dataset $\mathcal{D}_i$ with $|\mathcal{D}_i|$ samples. In each round, clients fine-tune the global LoRA module locally

and upload the updated adapters, the server then aggregates these adapters via weighted averaging:

$$\overline{A} = \sum_{i=1}^{K} p_i \, A_i, \quad \overline{B} = \sum_{i=1}^{K} p_i \, B_i \tag{2}$$

where $p_i = |\mathcal{D}_i| / \sum_{j=1}^{K} |\mathcal{D}_j|$ denotes the aggregation weight, and $\overline{A}$, $\overline{B}$ are the resulting global LoRA modules. However, this independent averaging introduces *aggregation noise*, as the ideal global update depends on the products $B_i A_i$, rather than the individual modules B_i or A_i:

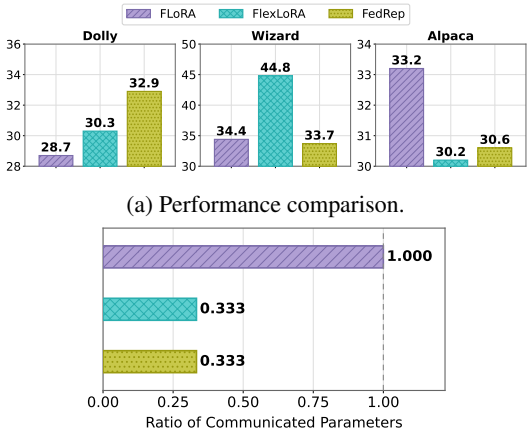
$$\Delta W = \sum_{i=1}^{K} p_i \, B_i A_i \;\neq\; \overline{B}\,\overline{A} = \left(\sum_{i=1}^{K} p_i \, B_i \right) \left(\sum_{i=1}^{K} p_i \, A_i \right) \tag{3}$$

where ΔW denotes the desired global update obtained by averaging the local updates ΔW_i. The discrepancy between the ideal update and the actual aggregated update accumulates over communication rounds and may progressively degrade model performance.

2.3 Observations

Observation 1: Comparison of Several Heterogeneous LoRA Federated Fine-Tuning Methods.

To compare mainstream heterogeneous LoRA federated fine-tuning methods, we design an experimental scenario where five clients are assigned different local LoRA ranks $[64, 64, 16, 8, 4]$, and fine-tuning is conducted on the Dolly, Wizard, and Alpaca datasets, followed by evaluation on a subset of MMLU. Note that FedRep is a replication-based padding method (Byun & Lee, 2025); more details are in Appendix C.2. As illustrated in Figure 1a, different methods achieve optimal performance on different datasets, with no single method dominating across all datasets. Meanwhile, Figure 1b shows that FlexLoRA and FedRep incur identical communication overhead, while FLoRA introduces the highest cost. Although FlexLoRA and FedRep share the same communication overhead, FlexLoRA additionally requires large-scale SVD operations on the server, incurring extra computational cost. These observations motivate us to investigate *whether padding-based methods can be further improved to achieve better performance while maintaining low communication cost.*



(a) Performance comparison.

(b) Communication cost comparison.

Figure 1: **Comparison of the heterogeneous LoRA federated fine-tuning methods.**

Observation 2: Replication Padding Underperforms Under Multi-Rank Data Heterogeneity.

According to (Byun & Lee, 2025), in federated learning scenarios with only two ranks, when high-rank clients possess high-quality data, replication padding outperforms zero padding. A natural question arises: *Does this advantage still hold when there are multiple different ranks and high-rank clients do not possess high-quality data?*

Table 1: Before/After aggregation accuracy of Zero and Replication Padding across rank groups on MMLU.

Rank	Padding	Round 1		Round 2		Round 3	
		Before	After	Before	After	Before	After
r=64	Zero	29.00	27.56	31.16	29.25	29.87	29.95
	Replication	29.06	28.00	28.32	28.78	29.50	29.42
r=32	Zero	23.72	27.37	27.70	29.95	27.84	28.25
	Replication	23.16	26.99	26.87	28.03	25.75	26.16
r=4	Zero	30.42	29.18	28.21	29.02	28.94	28.96
	Replication	30.42	26.36	26.81	26.77	26.17	27.86

To investigate this, we construct a federated learning scenario with six clients, whose ranks are $[64, 64, 32, 32, 4, 4]$. We conduct federated fine-tuning on the Dolly dataset, where rank-64 clients

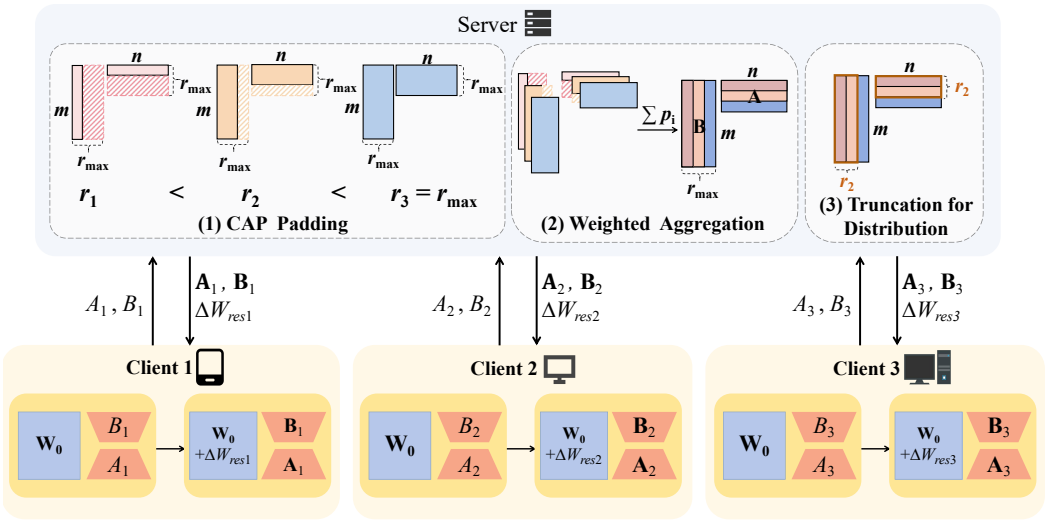


Figure 2: **FedRRA workflow.** The server applies CAP and aggregates LoRA modules into the global $\mathbf{B}$ and $\mathbf{A}$. It then truncates the global modules to each client's rank r_i and sends them back together with the residual correction ΔW_{res_i}, which is added to the pretrained weight matrix $\mathbf{W}_0$.

are assigned class-imbalanced data while the remaining clients receive data sampled from a Dirichlet distribution. As shown in Table 1, In most rank groups, replication padding consistently underperforms zero padding, with the gap most pronounced for clients $r = 32$. For instance, replication padding yields an After score of 28.03 in Round 2 versus 29.95 for zero padding, a deficit that persists throughout training. These results indicate that noisy information from low-quality high-rank clients propagates severely under replication padding, motivating us to design a padding strategy that adapts to diverse heterogeneous scenarios.

3 Proposed Method: FedRRA

In this section, we introduce our method FedRRA. As illustrated in Figure 2, FedRRA incorporates two key methods: *Consensus-Aware Padding* (CAP) and *Aggregation Residual Correction* (ARC), which are detailed in the following subsections. Additionally, Appendix A provides the detailed pseudocode of our FedRRA algorithm.

3.1 Consensus-Aware Padding

Motivated by the replication strategy (Byun & Lee, 2025) and the magnitude-based importance evaluation in AdaLoRA (Zhang et al., 2023), we propose a novel padding method named *Consensus-Aware Padding* (CAP), as illustrated in Figure 3. CAP is designed for general heterogeneous-rank settings, constructing fill values by computing a weighted aggregation of all clients that have trained the target dimension.

Rank-One Signal Scoring. According to the matrix multiplication mechanism and the update rule of LoRA (Equation 1), the low-rank update can be written as:

$$\Delta W = BA = \sum_{k=1}^{r} b_k a_k^\top \tag{4}$$

where $B \in \mathbb{R}^{m\times r}$ and $A \in \mathbb{R}^{r\times n}$, b_k denote the k-th column of B and $a_k^\top$ denote the k-th row of A. To quantify how strongly each dimension contributes to the learned update, we measure the magnitude of the corresponding rank-one component by its Frobenius norm:

$$s(k) = \left\| b_k a_k^\top \right\|_F = \|b_k\|_2 \, \|a_k\|_2 \tag{5}$$

For a nonzero rank-one component, $s(k)$ equals its sole nonzero singular value and spectral norm, quantifying the maximum output-update strength of that component. We examine alternative scoring

metrics (e.g., gradient) in Appendix F.4. However, under data non-IID heterogeneity, locally important dimensions may still deviate from the global optimization trend. Therefore, beyond relying solely on rank-one update magnitude, CAP further quantifies whether each client's update direction aligns with the consensus among other eligible clients.

Direction-aware Soft Gating. For each dimension k, let $\mathcal{K}_k = \{j : r_j \geq k\}$. Based on the observations in Appendix E.1, we construct the consensus direction exclusively from B. Specifically, for each client j, we define a personalized consensus direction $\mathbf{c}_{k,j}^B$ by uniformly averaging over all eligible clients except j itself, and then its cosine similarity to the consensus direction:

$$\mathbf{c}_{k,j}^B = \frac{1}{|\mathcal{K}_k| - 1} \sum_{\substack{i \in \mathcal{K}_k \\ i \neq j}} b_{i,k}, \qquad \rho_j(k) = \cos\big(b_{j,k},\, \mathbf{c}_{k,j}^B\big) \tag{6}$$

where $b_{i,k}$ denotes the k-th column of client i's local B matrix and $\rho_j(k)$ measures how well client j's local direction on dimension k aligns with its consensus direction. A natural approach is to apply zero-threshold clipping, retaining only clients with $\rho_j(k) > 0$. In practice, however, cosine similarities are rarely negative due to inherent correlations among clients, making zero-threshold filtering ineffective. To address this, we adopt a *mean-referenced soft gate*. Specifically, we calculate the mean cosine similarity of all eligible clients on dimension k and define the adaptive threshold and alignment factor as:

$$\mu_k = \frac{1}{|\mathcal{K}_k|} \sum_{j \in \mathcal{K}_k} \rho_j(k), \qquad \alpha_j(k) = \max(0, \rho_j(k) - \mu_k) \tag{7}$$

Here, μ_k denotes the average cosine similarity on dimension k, and $\alpha_j(k)$ measures how much client j's alignment exceeds this average level. This threshold automatically adapts to the degree of heterogeneity among clients: only clients whose alignment exceeds the group mean contribute to the completion, while clients below the mean are softly pruned to zero.

Weighted Consensus Padding. We combine the signal strength $s_j(k)$ and alignment factor $\alpha_j(k)$ into a composite coefficient $\omega_j^{(k)} = s_j(k) \cdot \alpha_j(k)$. After renormalization $\hat{\omega}_j^{(k)} = \omega_j^{(k)} \big/ \sum_{i \in \mathcal{K}_k} \omega_i^{(k)}$, the consensus fill vectors are:

$$\mathbf{b}_k^* = \sum_{j \in \mathcal{K}_k} \hat{\omega}_j^{(k)}\, b_{j,k}, \qquad \mathbf{a}_k^* = \sum_{j \in \mathcal{K}_k} \hat{\omega}_j^{(k)}\, a_{j,k} \tag{8}$$

For each client i with native rank $r_i < R$, the padded adapter matrices are formed by concatenating the original adapters with these fill vectors:

$$\tilde{B}_i = \big[\, B_i \,\big|\, \mathbf{b}_{r_i+1}^* \,\big|\, \cdots \,\big|\, \mathbf{b}_R^* \,\big] \in \mathbb{R}^{m \times R}, \quad \tilde{A}_i^\top = \big[\, A_i^\top \,\big|\, \mathbf{a}_{r_i+1}^{*\top} \,\big|\, \cdots \,\big|\, \mathbf{a}_R^{*\top} \,\big] \in \mathbb{R}^{n \times R} \tag{9}$$

In this way, CAP fills missing dimensions with consensus-supported components. If only one client covers dimension k, we use replication padding to copy its trained component to clients missing that dimension. If exactly two clients cover it, we combine their components according to their rank-one signal scores when their B directions are positively aligned and the total signal is nonzero; otherwise, we use zero-padding. If no client covers the dimension, CAP naturally reduces to zero-padding.

3.2 Aggregation Residual Correction

After applying CAP, all client adapters are padded to the common rank R. However, similar to the homogeneous-rank setting, this independent averaging still suffers from *aggregation noise*. To address this challenge, we propose a novel residual correction method. Consider the decomposition of the padded product:

$$\tilde{B}_i \tilde{A}_i = B_i A_i + \underbrace{B_i^* A_i^*}_{\text{padding artifact}} \tag{10}$$

where $B_i \in \mathbb{R}^{m \times r_i}$ and $A_i \in \mathbb{R}^{r_i \times n}$ are the native adapters, and B_i^*, A_i^* denote the filled dimensions. The first term captures the component learned through training, while the second term is solely induced by the padding strategy and has not been optimized during training. Including this second term in the correction target may introduce irreversible noise. Therefore, we define a *native correction target* based only on the first term:

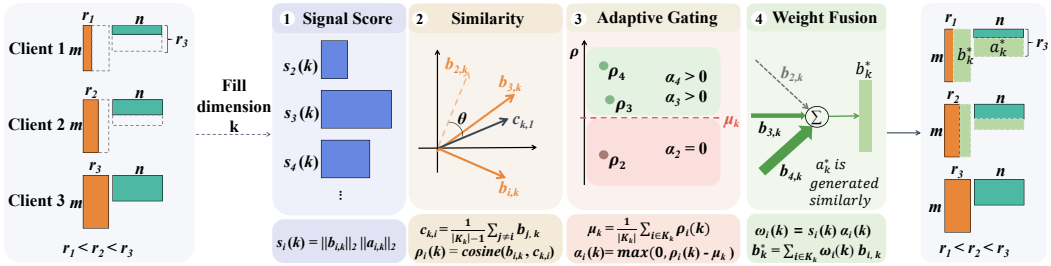


Figure 3: **Overview of CAP.** CAP first evaluates the signal strength of each dimension for every client, as well as its cosine similarity to the corresponding consensus direction. It then determines each client's contribution through an adaptive mean-referenced soft gating mechanism, and finally fuses these weights to generate the consensus fill vectors b_k^* and a_k^*.

$$\Delta W_{nat} = \sum_{i=1}^{K} p_i B_i A_i \tag{11}$$

In the heterogeneous-rank setting, after obtaining the global LoRA modules, the server truncates them back to each client's native rank r_i before broadcasting. More specifically:

$$\begin{aligned} \mathbf{B} &= \sum_{k=1}^{K} p_k \tilde{B}_k \in \mathbb{R}^{m\times R}, \qquad \mathbf{A} = \sum_{k=1}^{K} p_k \tilde{A}_k \in \mathbb{R}^{R\times n}, \\ \mathbf{B}_i &= \mathbf{B}_{[:,:r_i]} \in \mathbb{R}^{m\times r_i}, \qquad \mathbf{A}_i = \mathbf{A}_{[:r_i,:]} \in \mathbb{R}^{r_i\times n}. \end{aligned} \tag{12}$$

where $\mathbf{B}$ and $\mathbf{A}$ denote the global LoRA modules, and $\mathbf{B}_i$ and $\mathbf{A}_i$ represent their truncations to the client-specific rank r_i. To account for the discrepancy between the native correction target ΔW_{nat} and the actual low-rank aggregated update, we further introduce a residual term. Since this residual is generally high-rank and cannot be expressed within the low-rank parameterization, we absorb it into the frozen backbone weights. Accordingly, the update at the j^{th} round can be written as:

$$W_0^{j+1} = W_0^j + \underbrace{\Delta W_{nat} - \mathbf{BA}}_{\text{Residual } \Delta W_{res}} + \mathbf{BA} \tag{13}$$

$$= W_0^j + \underbrace{\Delta W_{nat} - \mathbf{B}_i\mathbf{A}_i}_{\text{Residual } \Delta W_{res_i}} + \mathbf{B}_i\mathbf{A}_i = \underbrace{W_0^j + \Delta W_{nat}}_{\text{Ideal aggregation}} \tag{14}$$

where W_0 denotes the frozen model weights. Note that Equation 13 represents the server-side update, whereas Equation 14 represents the update for client i.

Best Inexact Approximation. For a target rank r, we compute the rank-r truncated SVD approximation of the server-side residual ΔW_{res}:

$$\begin{aligned} U\Sigma V^T &\leftarrow \mathbf{SVD}(\Delta W_{res}) \\ \Delta W_{res}^r &\leftarrow U[:r]\,\Sigma[:r,:r]\,V^T[:r] \end{aligned} \tag{15}$$

Among all residual approximations of rank at most r, this choice minimizes the Frobenius approximation error. The resulting correction error is determined exactly by the discarded singular values. Under bounded-input and Lipschitz assumptions, we can further bound its effect on the output and loss; the derivation is given in Appendix A.1. The effect of the truncation rank on task performance and cost is evaluated in Appendix F.5.

4 Experiments

4.1 Experiment Setup

Baselines. We compare FedRRA with six federated baselines and a centralized reference. (1) **FLoRA** (Wang et al., 2024b) aggregates heterogeneous LoRA modules by stacking them together.

Table 2: Comparison of FedRRA with baselines on MMLU and MT-Bench. Centralized (in grey) sets the benchmark skyline. Results under other rank allocation strategies are provided in Appendix D.1.

Foundation model	Method Triangle	MMLU			MT-Bench
		Dolly	Wizard	Alpaca	Wizard
TinyLlama	Centralized	26.95 ±0.01	26.44 ±0.01	26.64 ±0.02	2.84 ±0.03
	HETLoRA	25.04 ±0.16	25.10 ±0.08	24.87 ±0.23	2.45 ±0.13
	FedRep	25.46 ±0.24	24.76 ±0.18	25.25 ±0.16	2.24 ±0.10
	HeLoRA-Pad	25.03 ±0.25	24.48 ±0.22	25.04 ±0.15	2.34 ±0.19
	FLoRA	25.72 ±0.18	25.46 ±0.15	25.61 ±0.13	2.50 ±0.08
	FlexLoRA	25.42 ±0.21	25.09 ±0.12	25.36 ±0.14	2.46 ±0.09
	FedPLoRA	25.99 ±0.14	25.67 ±0.16	25.73 ±0.11	2.55 ±0.11
	FedRRA	**26.37** ±0.12	**25.96** ±0.08	**26.08** ±0.05	**2.65** ±0.06
Llama2	Centralized	47.76 ±0.12	46.82 ±0.08	46.86 ±0.14	3.77 ±0.09
	HETLoRA	45.82 ±0.32	45.21 ±0.28	45.08 ±0.17	3.06 ±0.09
	FedRep	45.34 ±0.27	45.69 ±0.18	44.54 ±0.32	3.13 ±0.07
	HeLoRA-Pad	45.46 ±0.35	44.86 ±0.23	45.14 ±0.19	2.98 ±0.11
	FLoRA	46.42 ±0.28	45.79 ±0.17	45.78 ±0.19	3.28 ±0.12
	FlexLoRA	46.31 ±0.25	45.45 ±0.31	45.52 ±0.23	3.37 ±0.06
	FedPLoRA	46.78 ±0.31	46.02 ±0.28	45.58 ±0.25	3.44 ±0.09
	FedRRA	**47.14** ±0.18	**46.38** ±0.25	**46.21** ±0.22	**3.55** ±0.12

(2) **FlexLoRA** (Bai et al., 2024) applies SVD-based truncation aggregation. (3) **HETLoRA** (Cho et al., 2024) expands heterogeneous local ranks to the maximum rank using zero padding. (4) **HeLoRA-Pad** (Fan et al., 2025) fills the missing dimensions using the mean of its own weights. (5) **FedRep** (Byun & Lee, 2025) copies the weight information from the highest-rank client to lower-rank clients. (6) **FedPLoRA** (Zhang et al., 2026) uses parallel rank-one modules and folds unselected modules into the backbone before local training. (7) **Centralized Fine-tuning** serves as a benchmark, where LoRA fine-tuning is performed in a centralized setting without federated learning.

Datasets and Models. We use three datasets: the Wizard dataset (Luo et al., 2023), the Alpaca dataset (Taori et al., 2023), and Databricks-dolly-15k (Dolly) (Zhang et al., 2024a). For federated training, we follow the non-IID setting in FedIT and partition the datasets using a Dirichlet distribution ($\alpha = 0.5$). For evaluation, we consider both closed (Chia et al., 2024) and open benchmarks: MMLU (Hendrycks et al., 2020) is used for closed evaluation on QA tasks, and MT-Bench (Zheng et al., 2023) is used for open evaluation on chatbot tasks. We use three models of different scales: TinyLlama (Zhang et al., 2024b), Qwen2.5-3B (Qwen et al., 2025), and Llama2-7B (Touvron et al., 2023b). We apply the LoRA modules only to the query and value weight matrices.

Implementation Details. We use a total of 10 clients and assume that all clients participate in communication in every round. The rank of the residual SVD is 16; we discuss this choice in Appendix F.5. We repeat each federated experiment with three random seeds and report the mean performance. To simulate heterogeneous computational resource scenarios, we design three different rank allocation strategies to assign different local LoRA ranks to the 10 clients: (1) **Triangle**: $[64, 64, 32, 32, 16, 16, 8, 8, 4, 4]$, where high-, medium-, and low-rank clients coexist in relatively balanced proportions; (2) **Diamond**: $[32, 32, 32, 16, 16, 16, 8, 8, 4, 4]$, where no high-rank clients are present; (3) **Head**: $[64, 32, 32, 4, 4, 4, 4, 4, 4, 4]$, where only a few clients have high ranks and the majority are low-rank.

4.2 Experiment Results

Main Results. As shown in Table 2, FedRRA achieves the highest mean score among federated methods in all eight settings under the Triangle rank allocation strategy. With TinyLlama, FedRRA scores 26.37, 25.96, and 26.08 on Dolly, Wizard, and Alpaca MMLU, respectively, exceeding the second-best method, FedPLoRA, by 0.38, 0.29, and 0.35 points. With Llama2-7B, it scores 47.14 on Dolly and 46.38 on Wizard, exceeding FedPLoRA by 0.36 points on each dataset. On Alpaca, it scores 46.21, surpassing the second-best method, FLoRA, by 0.43 points. FedRRA also achieves MT-Bench scores of 2.65 with TinyLlama and 3.55 with Llama2-7B, exceeding FedPLoRA by 0.10 and 0.11 points, respectively. The centralized results remain higher, while FedRRA's lead over the federated baselines holds across all three fine-tuning datasets and both model scales. Additional results with more clients and stronger data heterogeneity ($\alpha = 0.1$) are reported in Appendices D.2 and D.3, respectively.

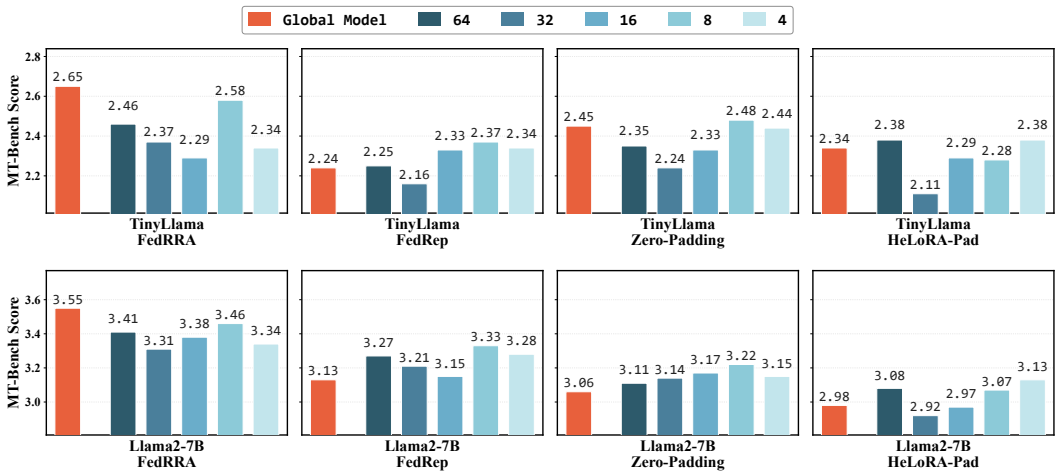


Figure 4: Impact of the **Triangle** rank allocation strategy on Wizard.

Impact of Rank Heterogeneity on Performance. To further analyze behavioral differences under rank heterogeneity, we evaluate both global and per-rank performance across different methods. As shown in Figure 4, FedRRA's global model consistently outperforms all local models across ranks, demonstrating effective aggregation of heterogeneous updates. In contrast, FedRep, Zero-Padding, and HeLoRA-Pad all exhibit cases where local models outperform the global model — on TinyLlama, the FedRep global model (2.24) is clearly inferior to its $r = 8$ and $r = 4$ local models (2.37 and 2.34), while the Zero-Padding global model (2.45) falls below its $r = 8$ local model (2.48) and the HeLoRA-Pad global model (2.34) falls below its $r = 64$ and $r = 4$ local models (both 2.38), indicating information loss during aggregation. From a per-rank perspective, higher-rank clients do not always achieve better performance than lower-rank ones, likely due to data heterogeneity. Notably, FedRRA's high-rank clients consistently achieve better performance than their counterparts in FedRep, Zero-Padding, and HeLoRA-Pad, confirming that FedRRA effectively reduces padding-induced noise and yields more robust performance across ranks. More results are provided in Appendix D.

The Communication Overhead of FedRRA. Because FedRRA transmits residuals to clients, it incurs additional communication overhead despite compressing them through approximation. To quantify this overhead, we compare the average total upload and download communication cost per round across methods. Further details are provided in Appendix E.4. As shown in Figure 5, FedRRA substantially reduces overall communication cost compared with FLoRA and FedPLoRA. Moreover, with only a modest additional communication cost, FedRRA achieves the best performance across several heterogeneous scenarios. These results demonstrate its advantage for heterogeneous federated fine-tuning.

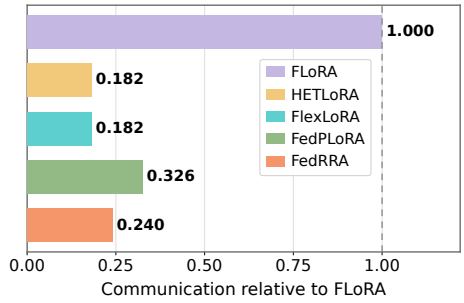


Figure 5: Comparison of the communication cost.

4.3 Ablation Study

Effect of CAP and ARC. To evaluate the effectiveness of CAP and ARC, we conduct ablation studies under different padding strategies. Specifically, "w/o ARC" denotes the variant without residual correction. As shown in Table 3, CAP achieves the best performance among all padding methods under the "w/o ARC" setting. Moreover, nearly all padding methods exhibit performance improvements after applying ARC. Additional results are in Appendix F.1.

Effect of ARC Design. ARC constructs its correction target from native-rank updates to avoid padding artifacts and computes client-specific residuals to account for rank-dependent truncation. ARC-PT instead uses padded updates as the target, while ARC-SR shares a minimum-rank residual across clients. As shown in Table 4, ARC scores higher than ARC-PT and ARC-SR in the vast

Table 3: CAP and ARC ablation on TinyLlama under Triangle rank allocation. A blank setting indicates ARC is enabled; parentheses show gains over the corresponding w/o ARC result.

Method	Setting	MMLU			MT-Bench
		Dolly	Wizard	Alpaca	Wizard
Zero-Padding	w/o ARC	25.04 25.12 (+0.08)	25.10 25.16 (+0.06)	24.87 25.15 (+0.28)	2.45 2.49 (+0.04)
HeLoRA-Pad	w/o ARC	25.03 25.27 (+0.24)	24.48 24.66 (+0.18)	25.04 25.64 (+0.60)	2.34 2.43 (+0.09)
Replication	w/o ARC	25.46 25.84 (+0.38)	24.76 25.02 (+0.26)	25.25 25.56 (+0.31)	2.24 2.42 (+0.18)
FedRRA	w/o ARC	**25.92** 26.37 (+0.45)	**25.41** 25.96 (+0.55)	**25.62** 26.08 (+0.46)	**2.52** 2.65 (+0.13)

Table 4: ARC design ablation on TinyLlama under Triangle rank allocation.

Method	Dolly			Wizard			Alpaca		
	ARC	ARC-PT	ARC-SR	ARC	ARC-PT	ARC-SR	ARC	ARC-PT	ARC-SR
Zero-Padding	25.12	**25.13**	24.67	**25.16**	25.15	24.80	**25.15**	25.14	24.83
HeLoRA-Pad	**25.27**	24.86	24.75	**24.66**	24.43	24.39	**25.64**	25.24	25.32
Replication	**25.84**	25.24	24.89	**25.02**	24.61	24.34	**25.56**	25.21	25.18
FedRRA	**26.37**	25.88	25.59	**25.96**	25.41	25.44	**26.08**	25.63	25.45

majority of tasks. These results suggest that both the native-rank target and client-specific residuals contribute to effective correction under heterogeneous ranks. See Appendix F.2 for more results.

Effect of Mean-Referenced Soft Gate. We conduct an ablation study comparing three configurations. "w/o Gate" removes gating by setting $\alpha_j(k) = 1$ for all eligible clients, while "Zero-Ref" replaces the mean-referenced threshold with a fixed zero baseline. As shown in Table 5, our default configuration performs best on all tasks, highlighting the importance of mean-referenced soft gating. We further examine fixed gate thresholds $\{0.1, 0.5, 0.9\}$. As shown in Figure 6, their effectiveness varies across datasets and tasks. For example, 0.5 performs best among the fixed thresholds on Wizard but worst on Alpaca, and remains below our mean-referenced strategy in every evaluated case. These results support the effectiveness of the proposed mean-referenced soft gate. Additional results are provided in Appendix F.3.

Table 5: Mean-referenced soft gate ablation on TinyLlama under Triangle rank allocation.

Method	MMLU			MT-Bench
	Dolly	Wizard	Alpaca	Wizard
FedRRA (w/o Gate)	25.79	25.50	25.92	2.43
FedRRA (Zero-Ref)	25.77	25.49	25.94	2.45
FedRRA	**26.37**	**25.96**	**26.08**	**2.65**

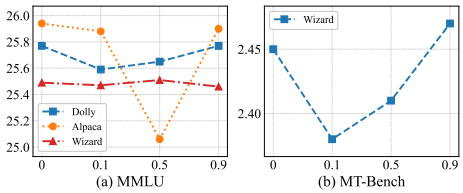


Figure 6: Effect of the gate thresholds on (a) MMLU and (b) MT-Bench.

5 Conclusion

In this work, we introduce FedRRA, a novel heterogeneous FFT method designed to address two challenges in existing padding-based heterogeneous approaches: limited adaptability to heterogeneous resource settings and aggregation noise. By leveraging Consensus-Aware Padding (CAP) with selective weighted padding, FedRRA better adapts to diverse resource-heterogeneous scenarios. In addition, we introduce an aggregation residual correction mechanism tailored for heterogeneous settings to effectively mitigate aggregation noise. Extensive experiments demonstrate that FedRRA consistently outperforms state-of-the-art heterogeneous LoRA-based FFT methods across three heterogeneous scenarios, providing a practical and effective solution for heterogeneous FFT.

AI USE STATEMENT

During the preparation of this manuscript, we used LLMs as writing assistants. Some sentences were drafted or refined by LLMs, but all text was reviewed and finalized by the human authors. We also used LLMs to assist with literature searches and the identification of related work. LLMs were not used for any core intellectual contributions, including research ideation or methodology formulation. We take responsibility for the final content of this work, including text, claims or artifacts produced with the aid of generative AI.

ETHICS STATEMENT

Our experiments use publicly available datasets and benchmarks and do not involve human subjects or the collection of new personal data. Although federated fine-tuning keeps raw client data local, exchanged model updates may still reveal information about local data and therefore do not provide a formal privacy guarantee. In addition, malicious or compromised clients may perform poisoning or backdoor attacks by manipulating local updates. FedRRA is designed to address resource heterogeneity and aggregation noise rather than privacy or adversarial robustness. Practical deployments, especially in privacy- or security-sensitive scenarios, should therefore incorporate appropriate safeguards.

REPRODUCIBILITY STATEMENT

To support reproducibility, we provide a source code package in the supplementary materials. The package includes:

- Configuration files and instructions for setting up the experimental environment.
- complete implementations of our method and the baselines.
- Scripts for training our method.
- Evaluation scripts for the main experiments.

In addition, Appendix C provides dataset and implementation details, including experimental configurations. We hope these materials enable researchers to reproduce and extend our work.

REFERENCES

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

Sara Babakniya, Ahmed Roushdy Elkordy, Yahya H Ezzeldin, Qingfeng Liu, Kee-Bong Song, Mostafa El-Khamy, and Salman Avestimehr. Slora: Federated parameter efficient fine-tuning of language models. *arXiv preprint arXiv:2308.06522*, 2023.

Jiamu Bai, Daoyuan Chen, Bingchen Qian, Liuyi Yao, and Yaliang Li. Federated fine-tuning of large language models under heterogeneous tasks and client resources. *Advances in Neural Information Processing Systems*, 37:14457–14483, 2024.

Jieming Bian, Lei Wang, Letian Zhang, and Jie Xu. Lora-fair: Federated lora fine-tuning with aggregation and initialization refinement. In *Proceedings of the IEEE/CVF International Conference on Computer Vision*, pp. 3737–3746, 2025.

Yuji Byun and Jaeho Lee. Towards federated low-rank adaptation of language models with rank heterogeneity. In *Proceedings of the 2025 Conference of the Nations of the Americas Chapter of the Association for Computational Linguistics: Human Language Technologies (Volume 2: Short Papers)*, pp. 356–362, 2025.

Yupeng Chang, Xu Wang, Jindong Wang, Yuan Wu, Linyi Yang, Kaijie Zhu, Hao Chen, Xiaoyuan Yi, Cunxiang Wang, Yidong Wang, et al. A survey on evaluation of large language models. *ACM transactions on intelligent systems and technology*, 15(3):1–45, 2024.

Shuangyi Chen, Yuanxin Guo, Yue Ju, Harik Dalal, Zhongwen Zhu, and Ashish Khisti. Robust federated finetuning of llms via alternating optimization of lora. *arXiv preprint arXiv:2502.01755*, 2025.

Yew Ken Chia, Pengfei Hong, Lidong Bing, and Soujanya Poria. Instructeval: Towards holistic evaluation of instruction-tuned large language models. In *Proceedings of the First edition of the Workshop on the Scaling Behavior of Large Language Models (SCALE-LLM 2024)*, pp. 35–64, 2024.

Yae Jee Cho, Luyang Liu, Zheng Xu, Aldi Fahrezi, and Gauri Joshi. Heterogeneous lora for federated fine-tuning of on-device foundation models. In *Proceedings of the 2024 conference on empirical methods in natural language processing*, pp. 12903–12913, 2024.

Xin Luna Dong, Seungwhan Moon, Yifan Ethan Xu, Kshitiz Malik, and Zhou Yu. Towards next-generation intelligent assistants leveraging llm techniques. In *Proceedings of the 29th ACM SIGKDD Conference on Knowledge Discovery and Data Mining*, pp. 5792–5793, 2023.

Hao Du, Shang Liu, Lele Zheng, Yang Cao, Atsuyoshi Nakamura, and Lei Chen. Privacy in fine-tuning large language models: Attacks, defenses, and future directions. In *Pacific-Asia Conference on Knowledge Discovery and Data Mining*, pp. 326–344. Springer, 2025.

Boyu Fan, Xiang Su, Sasu Tarkoma, and Pan Hui. Helora: Lora-heterogeneous federated fine-tuning for foundation models. *ACM Transactions on Internet Technology*, 25(2):1–22, 2025.

Chongyang Gao, Kezhen Chen, Jinmeng Rao, Ruibo Liu, Baochen Sun, Yawen Zhang, Daiyi Peng, Xiaoyuan Guo, and VS Subrahmanian. Mola: Moe lora with layer-wise expert allocation. In *Findings of the Association for Computational Linguistics: NAACL 2025*, pp. 5097–5112, 2025a.

Junyao Gao, Yanan Sun, Fei Shen, Xin Jiang, Zhening Xing, Kai Chen, and Cairong Zhao. Faceshot: Bring any character into life. *arXiv preprint arXiv:2503.00740*, 2025b.

Pengxin Guo, Shuang Zeng, Yanran Wang, Huijie Fan, Feifei Wang, and Liangqiong Qu. Selective aggregation for low-rank adaptation in federated learning. *arXiv preprint arXiv:2410.01463*, 2024.

Dan Hendrycks, Collin Burns, Steven Basart, Andy Zou, Mantas Mazeika, Dawn Song, and Jacob Steinhardt. Measuring massive multitask language understanding. *arXiv preprint arXiv:2009.03300*, 2020.

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

Edward J Hu, Yelong Shen, Phillip Wallis, Zeyuan Allen-Zhu, Yuanzhi Li, Shean Wang, Liang Wang, Weizhu Chen, et al. Lora: Low-rank adaptation of large language models. *Iclr*, 1(2):3, 2022.

Chengsong Huang, Qian Liu, Bill Yuchen Lin, Tianyu Pang, Chao Du, and Min Lin. Lorahub: Efficient cross-task generalization via dynamic lora composition. *arXiv preprint arXiv:2307.13269*, 2023.

Dominique Kelly, Yimin Chen, Sarah E Cornwell, Nicole S Delellis, Alex Mayhew, Sodiq Onaolapo, and Victoria L Rubin. Bing chat: The future of search engines? *Proceedings of the Association for Information Science and Technology*, 60(1):1007–1009, 2023.

Jabin Koo, Minwoo Jang, and Jungseul Ok. Towards robust and efficient federated low-rank adaptation with heterogeneous clients. In *Proceedings of the 63rd Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, pp. 416–429, 2025.

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

Xiang Lisa Li and Percy Liang. Prefix-tuning: Optimizing continuous prompts for generation. In *Proceedings of the 59th Annual Meeting of the Association for Computational Linguistics and the 11th International Joint Conference on Natural Language Processing (Volume 1: Long Papers)*, pp. 4582–4597, 2021.

Vladislav Lialin, Namrata Shivagunde, Sherin Muckatira, and Anna Rumshisky. Relora: High-rank training through low-rank updates. *arXiv preprint arXiv:2307.05695*, 2023.

Zhenqing Ling, Daoyuan Chen, Liuyi Yao, Yaliang Li, and Ying Shen. On the convergence of zeroth-order federated tuning for large language models. In *Proceedings of the 30th ACM SIGKDD Conference on Knowledge Discovery and Data Mining*, pp. 1827–1838, 2024.

Xiao Liu, Kaixuan Ji, Yicheng Fu, Weng Tam, Zhengxiao Du, Zhilin Yang, and Jie Tang. P-tuning: Prompt tuning can be comparable to fine-tuning across scales and tasks. In *Proceedings of the 60th Annual Meeting of the Association for Computational Linguistics (Volume 2: Short Papers)*, pp. 61–68, 2022.

Xiao Liu, Yanan Zheng, Zhengxiao Du, Ming Ding, Yujie Qian, Zhilin Yang, and Jie Tang. Gpt understands, too. *AI open*, 5:208–215, 2024.

Haipeng Luo, Qingfeng Sun, Can Xu, Pu Zhao, Jianguang Lou, Chongyang Tao, Xiubo Geng, Qingwei Lin, Shifeng Chen, and Dongmei Zhang. Wizardmath: Empowering mathematical reasoning for large language models via reinforced evol-instruct. *arXiv preprint arXiv:2308.09583*, 2023.

Brendan McMahan, Eider Moore, Daniel Ramage, Seth Hampson, and Blaise Aguera y Arcas. Communication-efficient learning of deep networks from decentralized data. In *Artificial intelligence and statistics*, pp. 1273–1282. Pmlr, 2017.

Duy Phuong Nguyen, J Pablo Munoz, and Ali Jannesari. Flora: Enhancing vision-language models with parameter-efficient federated learning. *arXiv preprint arXiv:2404.15182*, 2024.

Zhen Qin, Daoyuan Chen, Bingchen Qian, Bolin Ding, Yaliang Li, and Shuiguang Deng. Federated full-parameter tuning of billion-sized language models with communication cost under 18 kilobytes. *arXiv preprint arXiv:2312.06353*, 2023.

Qwen, :, An Yang, Baosong Yang, Beichen Zhang, Binyuan Hui, Bo Zheng, Bowen Yu, Chengyuan Li, Dayiheng Liu, Fei Huang, Haoran Wei, Huan Lin, Jian Yang, Jianhong Tu, Jianwei Zhang, Jianxin Yang, Jiaxi Yang, Jingren Zhou, Junyang Lin, Kai Dang, Keming Lu, Keqin Bao, Kexin Yang, Le Yu, Mei Li, Mingfeng Xue, Pei Zhang, Qin Zhu, Rui Men, Runji Lin, Tianhao Li, Tianyi Tang, Tingyu Xia, Xingzhang Ren, Xuancheng Ren, Yang Fan, Yang Su, Yichang Zhang, Yu Wan, Yuqiong Liu, Zeyu Cui, Zhenru Zhang, and Zihan Qiu. Qwen2.5 technical report, 2025. URL https://arxiv.org/abs/2412.15115.

Karan Singhal, Shekoofeh Azizi, Tao Tu, S Sara Mahdavi, Jason Wei, Hyung Won Chung, Nathan Scales, Ajay Tanwani, Heather Cole-Lewis, Stephen Pfohl, et al. Large language models encode clinical knowledge. *Nature*, 620(7972):172–180, 2023.

Raghav Singhal, Kaustubh Ponkshe, and Praneeth Vepakomma. Fedex-lora: Exact aggregation for federated and efficient fine-tuning of large language models. In *Proceedings of the 63rd Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, pp. 1316–1336, 2025.

Rohan Taori, Ishaan Gulrajani, Tianyi Zhang, Yann Dubois, Xuechen Li, Carlos Guestrin, Percy Liang, and Tatsunori B Hashimoto. Stanford alpaca: An instruction-following llama model, 2023.

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

Hugo Touvron, Louis Martin, Kevin Stone, Peter Albert, Amjad Almahairi, Yasmine Babaei, Nikolay Bashlykov, Soumya Batra, Prajjwal Bhargava, Shruti Bhosale, et al. Llama 2: Open foundation and fine-tuned chat models. *arXiv preprint arXiv:2307.09288*, 2023b.

Mojtaba Valipour, Mehdi Rezagholizadeh, Ivan Kobyzev, and Ali Ghodsi. Dylora: Parameter-efficient tuning of pre-trained models using dynamic search-free low-rank adaptation. In *Proceedings of the 17th Conference of the European Chapter of the Association for Computational Linguistics*, pp. 3274–3287, 2023.

Boxin Wang, Yibo Zhang, Yuan Cao, Bo Li, Hugh McMahan, Sewoong Oh, Zheng Xu, and Manzil Zaheer. Can public large language models help private cross-device federated learning? In *Findings of the Association for Computational Linguistics: NAACL 2024*, pp. 934–949, 2024a.

Lei Wang, Jieming Bian, Letian Zhang, and Jie Xu. Adaptive lora experts allocation and selection for federated fine-tuning. *arXiv preprint arXiv:2509.15087*, 2025.

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. *Advances in Neural Information Processing Systems*, 37:22513–22533, 2024b.

Mingbin Xu, Congzheng Song, Ye Tian, Neha Agrawal, Filip Granqvist, Rogier van Dalen, Xiao Zhang, Arturo Argueta, Shiyi Han, Yaqiao Deng, et al. Training large-vocabulary neural language models by private federated learning for resource-constrained devices. In *ICASSP 2023-2023 IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP)*, pp. 1–5. IEEE, 2023.

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 *Proceedings of the 30th ACM SIGKDD conference on knowledge discovery and data mining*, pp. 6137–6147, 2024.

Jianyi Zhang, Saeed Vahidian, Martin Kuo, Chunyuan Li, Ruiyi Zhang, Tong Yu, Guoyin Wang, and Yiran Chen. Towards building the federatedgpt: Federated instruction tuning. In *ICASSP 2024-2024 IEEE international conference on acoustics, speech and signal processing (ICASSP)*, pp. 6915–6919. IEEE, 2024a.

Peiyuan Zhang, Guangtao Zeng, Tianduo Wang, and Wei Lu. Tinyllama: An open-source small language model. *arXiv preprint arXiv:2401.02385*, 2024b.

Qingru Zhang, Minshuo Chen, Alexander Bukharin, Nikos Karampatziakis, Pengcheng He, Yu Cheng, Weizhu Chen, and Tuo Zhao. Adalora: Adaptive budget allocation for parameter-efficient fine-tuning. *arXiv preprint arXiv:2303.10512*, 2023.

Zikai Zhang, Rui Hu, and Jiahao Xu. Heterogeneous federated fine-tuning with parallel one-rank adaptation. *arXiv preprint arXiv:2602.16936*, 2026.

Lianmin Zheng, Wei-Lin Chiang, Ying Sheng, Siyuan Zhuang, Zhanghao Wu, Yonghao Zhuang, Zi Lin, Zhuohan Li, Dacheng Li, Eric Xing, et al. Judging llm-as-a-judge with mt-bench and chatbot arena. *Advances in neural information processing systems*, 36:46595–46623, 2023.

A THE ALGORITHM OF FEDRRA

Algorithm 1 FedRRA for Federated Learning

Require: Communication rounds T, clients $\mathcal{C}$ with ranks $\{r_i\}_{i\in\mathcal{C}}$, global rank R, pretrained weights W_0

Server distributes W_0 to all clients; each client i initializes B_i^0, A_i^0 with rank r_i
for $t = 1, \ldots, T$ **do**
  Server samples clients $\mathcal{C}^t \subseteq \mathcal{C}$, sends $\mathbf{B}_i, \mathbf{A}_i, \Delta W_{res_i}$ to $i \in \mathcal{C}^t$
  for client $i \in \mathcal{C}^t$ **in parallel do**
    Update local frozen weights: $W_0 \leftarrow W_0 + \Delta W_{res_i}$
    $\{B_i^t, A_i^t\} \leftarrow \text{LOCALUPDATE}(\mathbf{B}_i, \mathbf{A}_i)$
  end for
  $\{\mathbf{B}_i, \mathbf{A}_i, \Delta W_{res_i}\}_{i\in\mathcal{C}^t} \leftarrow \text{SERVERUPDATE}(\{B_i^t, A_i^t, r_i\}_{i\in\mathcal{C}^t})$
end for
return W_0, $\{B_i, A_i\}_{i\in\mathcal{C}}$

Algorithm 2 FedRRA Server Update

Require: $\{B_i, A_i, r_i\}_{i\in\mathcal{C}^t}$, global rank R

▷ *Consensus-Aware Padding (CAP)*
for each dimension $k = 1, \ldots, R$ **do**
  Identify eligible set $\mathcal{K}_k = \{j : r_j \geq k\}$
  Compute $s_j(k)$, $\rho_j(k)$, and $\alpha_j(k) = \max(0,\, \rho_j(k) - \mu_k)$
  Obtain consensus fill vectors b_k^*, a_k^* via weighted aggregation with $\omega_j^{(k)} = s_j(k) \cdot \alpha_j(k)$
end for
for each client i with $r_i < R$ **do**
  Pad: $\tilde{B}_i \leftarrow [B_i \mid b_{r_i+1}^* \mid \cdots \mid b_R^*]$, $\tilde{A}_i$ likewise
end for
▷ *Aggregation*
$\mathbf{B} = \sum_i p_i \tilde{B}_i, \quad \mathbf{A} = \sum_i p_i \tilde{A}_i$
▷ *Residual Correction*
Compute native target: $\Delta W_{nat} = \sum_i p_i B_i A_i$
for each client $i \in \mathcal{C}^t$ **do**
  Truncate: $\mathbf{B}_i = \mathbf{B}_{[:,:r_i]}$, $\mathbf{A}_i = \mathbf{A}_{[:r_i,:]}$
  Compute residual: $\Delta W_{res} = \Delta W_{nat} - \mathbf{BA}$ and $\Delta W_{res_i} = \Delta W_{nat} - \mathbf{B}_i\mathbf{A}_i$
end for
return $\{\mathbf{B}_i, \mathbf{A}_i, \Delta W_{res_i}\}_{i\in\mathcal{C}^t}$

A.1 APPROXIMATION GUARANTEE FOR RESIDUAL CORRECTION

Consider one weight matrix at a fixed communication round. Let

$$R = \Delta W_{\text{nat}} - \mathbf{BA} = U\Sigma V^\top = \sum_{j=1}^{p} \sigma_j u_j v_j^\top, \qquad R_q = \sum_{j=1}^{q} \sigma_j u_j v_j^\top,$$

where $p = \min(m, n)$, $1 \leq q \leq p$, and $\sigma_1 \geq \sigma_2 \geq \cdots \geq \sigma_p \geq 0$. We set $\sigma_{p+1} = 0$. Denote the effective weight obtained with the full residual by $W^\star = W_0^t + \mathbf{BA} + R$ and that obtained with the truncated residual by $\widehat{W}_q = W_0^t + \mathbf{BA} + R_q$.

Approximation error. The difference between the two effective weights is $W^\star - \widehat{W}_q = R - R_q$. Orthogonality of the singular vectors therefore gives

$$\|W^\star - \widehat{W}_q\|_F^2 = \|R - R_q\|_F^2 = \sum_{j=q+1}^{p} \sigma_j^2, \qquad \|W^\star - \widehat{W}_q\|_2 = \sigma_{q+1}. \tag{16}$$

Moreover, this is the smallest Frobenius error achievable with a rank-q residual. To see this, let X be any matrix with $\mathrm{rank}(X) \le q$, and let $\tau_1, \dots, \tau_q$ be its singular values, including zeros if necessary. Von Neumann's trace inequality yields $\langle R, X\rangle \le \sum_{j=1}^{q} \sigma_j \tau_j$. Hence,

$$\begin{aligned}\|R - X\|_F^2 &= \|R\|_F^2 + \|X\|_F^2 - 2\langle R, X\rangle \\ &\ge \sum_{j=1}^{q} (\sigma_j - \tau_j)^2 + \sum_{j=q+1}^{p} \sigma_j^2 \ge \sum_{j=q+1}^{p} \sigma_j^2. \end{aligned} \tag{17}$$

Equality is attained by $X = R_q$. Thus, truncated SVD minimizes the correction error among all residuals of rank at most q. In particular, the error is zero if $q \ge \mathrm{rank}(R)$.

Effect on the output and loss. For a fixed input vector h to this layer with $\|h\|_2 \le H$, Equation 16 implies

$$\|(W^\star - \widehat{W}_q)h\|_2 \le \|R - R_q\|_2 \|h\|_2 \le H\sigma_{q+1}. \tag{18}$$

Suppose that, with all other parameters and computational paths fixed, the remainder of the network is L_g-Lipschitz with respect to this layer's output, and the loss is L_ℓ-Lipschitz with respect to the network output. Then

$$\begin{aligned}&\Big|\ell\big(g(W^\star h), y\big) - \ell\big(g(\widehat{W}_q h), y\big)\Big| \\ &\qquad \le L_\ell L_g \|(W^\star - \widehat{W}_q)h\|_2 \le L_\ell L_g H \sigma_{q+1}. \end{aligned} \tag{19}$$

These bounds characterize the effect of truncating one server-side residual at a fixed communication round. Their magnitude depends on the discarded singular values and the stated Lipschitz constants. The observed task-level trade-off across truncation ranks is reported in Appendix F.5.

B Related Works

Parameter-Efficient Fine-Tuning of LLMs. Parameter-Efficient Fine-Tuning (PEFT) aims to reduce the number of trainable parameters. Representative works include Prefix Tuning (Li & Liang, 2021), Prompt Tuning (Liu et al., 2024), Adapters (Houlsby et al., 2019), and Low-Rank Adaptation (LoRA) (Hu et al., 2022). Among these, LoRA is one of the most representative approaches. It introduces trainable low-rank matrices to approximate weight updates, thereby effectively reducing memory overhead. In addition, several variants have been proposed. For example, AdaLoRA (Zhang et al., 2023) adaptively allocates parameter budgets based on the importance scores of weight matrices, enhancing the flexibility of LoRA; DyLoRA (Valipour et al., 2023) dynamically trains across all ranks during the training process, reducing training cost.

Federated Fine-tuning of LLMs. Federated fine-tuning has emerged as a prominent paradigm in LLM fine-tuning, enabling efficient adaptation while preserving data privacy. Several prior works have explored this direction (Zhang et al., 2024a; Bian et al., 2025; Koo et al., 2025; Guo et al., 2024; Singhal et al., 2025; Ye et al., 2024; Wang et al., 2025). For example, FedIT(Zhang et al., 2024a) combines LoRA with FedAvg and serves as an effective baseline for further research. LoRA-FAIR (Bian et al., 2025) improves aggregation efficiency and accuracy by introducing a correction term on the server side. LoRA-A^2 (Koo et al., 2025) adopts a strategy of alternating freezing and adaptive rank selection, demonstrating robustness in challenging environments with low-rank constraints and high data heterogeneity. FedSA-LoRA (Guo et al., 2024) explores selective aggregation in personalized FL systems, where only the A matrix is shared with the server for aggregation. FedEx-LoRA (Singhal et al., 2025) achieves precise updates by adding a residual term to the pretrained frozen weight matrices, maintaining the efficiency of LoRA. However, these methods assume that client resources are homogeneous, which is often inconsistent with the resource heterogeneity observed in real-world FL systems.

Federated Fine-Tuning with Heterogeneous LoRA Ranks. Due to the heterogeneous resources present on the client side, clients need to adjust the LoRA rank accordingly. Existing work has proposed methods to address this issue. FLoRA (Wang et al., 2024b) adopts a stacking-based aggregation approach that ensures error-free aggregation. However, this method leads to linear

growth in communication complexity, and the trainable LoRA modules for the next round are typically re-initialized, which introduces initialization noise. FlexLoRA (Bai et al., 2024) applies SVD to redistribute the global update, thereby accommodating client LoRA modules of different ranks. However, this introduces additional computational overhead as well as errors caused by the redistribution. HETLoRA (Cho et al., 2024) employs a zero-padding strategy to expand each client's local rank to the maximum rank, followed by truncation of the resulting global LoRA module before distributing it to clients for the next round of fine-tuning. Nevertheless, this padding approach introduces aggregation noise, and zero-padding dilutes the information captured by high-rank clients. Several studies have further built upon this method: Byun & Lee (2025) proposed a replication-based padding method to address the dilution problem that zero-padding causes for high-rank clients. However, we find that this method can only accommodate the extreme scenario where exactly two distinct ranks exist, and it requires high-rank clients to possess high-quality data. HeLoRA-Pad (Fan et al., 2025) proposes Context-based Padding, which fills in values correlated with the original matrix, offering three variants based on mean, maximum, and adjacency functions. However, this method does not account for the potential adverse effects caused by data heterogeneity across clients. FedPLoRA (Zhang et al., 2026) replaces the conventional LoRA adapter with parallel rank-one modules. Its Select-N-Fold strategy allows each client to train a subset matching its rank budget while folding unselected modules into the frozen backbone weights before local training.

C Experimental Settings

C.1 Datasets and Metrics

Dolly dataset. The Dolly dataset is an instruction-tuning corpus released by Databricks, comprising approximately 15,000 manually written samples. Unlike purely synthetic datasets, its data is generated by human annotators, which improves linguistic diversity and realism. The dataset covers a variety of task types, including question answering, text generation, classification, and information extraction.

Wizard dataset. The Wizard dataset is constructed from the training data of WizardLM and contains around 70,000 instruction-response pairs. It is characterized by its relatively complex and multi-step instructions, which often require deeper reasoning and contextual understanding.

Alpaca dataset. The Alpaca dataset consists of approximately 52,000 instruction-following examples generated via a self-instruct framework. High-quality responses are produced by a strong teacher model, ensuring consistency and coverage across diverse tasks.

MMLU. The MMLU benchmark is designed to evaluate the knowledge breadth and reasoning ability of language models. It spans multiple academic disciplines, including science, humanities, and mathematics.

MT-Bench. MT-Bench is a benchmark specifically designed to evaluate multi-turn conversational abilities. It consists of a collection of open-ended dialogue tasks that require coherent, context-aware responses across multiple turns.

C.2 Setting Details

Observation 1. For Observation 1, we fine-tune the TinyLlama model and perform a rapid evaluation using the MMLU test set. Specifically, we conduct a comprehensive selection across all categories in the dataset and finally choose 1,444 samples as our evaluation subset. The detailed configurations of communication rounds and local epochs are presented in Table 6.

Table 6: Communication rounds and local epochs of Observation 1.

Datasets	Rounds	Epochs
Dolly	3	1
Alpaca	3	1
Wizard	3	1

Observation 2. For Observation 2, we use the Dolly dataset and categorize its samples into two groups based on their alignment with the MMLU benchmark. We treat *closed_qa*, *open_qa*, and *general_qa* as MMLU-aligned categories. In contrast, *creative_writing*, *summarization*, *brainstorming*, *information_extraction*, and *classification* are treated as MMLU-misaligned categories. The highest-rank clients only receive data from the misaligned subset, while the other clients receive data from the aligned subset. Within each subset, we perform stratified sampling to preserve the category distribution. Ultimately, the high-rank clients have smaller and more imbalanced local datasets, while the other clients possess more balanced data distributions. The training and evaluation protocols are consistent with those used in Observation 1.

Main Experiments. The training hyperparameters for the main experiments are shown in Table 7. To enable fair comparison, we follow the experimental setup of FLoRA as a reference. Given the dataset scales and available computational resources, and further supported by our convergence analysis which shows that models typically converge within five communication rounds(Appendix D.6), we conduct experiments with a limited number of communication rounds. All experiments are conducted on 8 NVIDIA 4090 GPU and 1 A100.

Table 7: Training Hyperparameters of Main Experiments.

Foundation	Datasets	Rounds	Epochs	Micro Batch Size	Learning Rate
TinyLlama	Dolly	5	1	24	3e-4
	Alpaca	5	1	24	3e-4
	Wizard	5	1	24	3e-4
Llama2	Dolly	4	2	8	2e-4
	Alpaca	5	1	8	2e-4
	Wizard	5	1	8	2e-4
Qwen2.5	Dolly	4	2	12	3e-4
	Alpaca	5	1	12	3e-4
	Wizard	5	1	12	3e-4

D Additional Experimental Results

D.1 Main Results

In the main experiments, to compare the performance of our method FedRRA with baselines under different rank distributions, we adopt three allocation strategies: Triangle, Diamond, and Head. The main text reports TinyLlama and Llama2-7B under Triangle allocation. Here, we provide the Qwen2.5-3B results under Triangle allocation and the results for all three models under Diamond and Head allocation.

Table 8: Comparison of FedRRA with baselines on MMLU and MT-Bench using Qwen2.5-3B under the Triangle rank allocation strategy.

Foundation model	Method Triangle	MMLU			MT-Bench
		Dolly	Wizard	Alpaca	Wizard
Qwen2.5	Centralized	67.27 ±0.04	66.86 ±0.03	66.69 ±0.06	5.37 ±0.09
	HETLoRA	65.41 ±0.29	65.69 ±0.16	65.48 ±0.14	4.76 ±0.12
	FedRep	65.82 ±0.25	65.99 ±0.16	65.77 ±0.17	4.85 ±0.16
	HeLoRA-Pad	65.52 ±0.21	65.38 ±0.19	65.39 ±0.16	4.73 ±0.09
	FLoRA	66.22 ±0.25	65.59 ±0.17	66.12 ±0.13	4.89 ±0.13
	FlexLoRA	65.93 ±0.16	65.41 ±0.22	65.89 ±0.15	4.81 ±0.09
	FedPLoRA	66.34 ±0.26	66.12 ±0.24	65.58 ±0.18	5.03 ±0.08
	FedRRA	**66.89** ±0.18	**66.49** ±0.13	**66.21** ±0.15	**5.19** ±0.09

Triangle Rank Allocation. As shown in Table 8, FedRRA achieves the highest federated mean on all four Qwen2.5-3B metrics. Its Dolly, Wizard, and Alpaca MMLU scores are 66.89, 66.49, and 66.21, exceeding the strongest competing baselines by 0.55 (FedPLoRA), 0.37 (FedPLoRA), and

Table 9: Comparison of FedRRA with baselines on MMLU and MT-Bench under the Diamond rank allocation strategy.

Foundation model	Method Diamond	MMLU			MT-Bench
		Dolly	Wizard	Alpaca	Wizard
TinyLlama	Centralized	26.95 ±0.01	26.44 ±0.01	26.64 ±0.02	2.84 ±0.03
	HETLoRA	25.21 ±0.15	25.03 ±0.12	24.96 ±0.19	2.29 ±0.09
	FedRep	24.89 ±0.19	25.44 ±0.15	25.31 ±0.16	2.36 ±0.08
	HeLoRA-Pad	24.89 ±0.25	24.76 ±0.18	24.67 ±0.19	2.15 ±0.13
	FLoRA	25.41 ±0.24	25.71 ±0.18	25.39 ±0.13	2.43 ±0.11
	FlexLoRA	25.86 ±0.21	25.43 ±0.16	25.73 ±0.12	2.39 ±0.13
	FedPLoRA	25.72 ±0.26	25.86 ±0.24	25.53 ±0.21	2.49 ±0.07
	FedRRA	**26.38** ±0.14	**26.09** ±0.09	**26.21** ±0.08	**2.61** ±0.08
Llama2	Centralized	47.76 ±0.12	46.82 ±0.08	46.86 ±0.14	3.77 ±0.09
	HETLoRA	46.07 ±0.22	45.18 ±0.19	45.45 ±0.12	3.34 ±0.09
	FedRep	45.57 ±0.23	45.47 ±0.18	45.15 ±0.18	3.37 ±0.12
	HeLoRA-Pad	45.67 ±0.26	44.92 ±0.22	44.78 ±0.23	3.24 ±0.08
	FLoRA	46.49 ±0.23	45.81 ±0.16	45.77 ±0.19	3.52 ±0.14
	FlexLoRA	46.68 ±0.17	45.44 ±0.14	46.04 ±0.16	3.43 ±0.08
	FedPLoRA	46.25 ±0.19	45.58 ±0.17	45.44 ±0.21	3.39 ±0.07
	FedRRA	**47.16** ±0.18	**46.04** ±0.14	**46.33** ±0.11	**3.63** ±0.09
Qwen2.5	Centralized	67.27 ±0.04	66.86 ±0.03	66.69 ±0.06	5.37 ±0.09
	HETLoRA	65.76 ±0.24	65.89 ±0.18	65.43 ±0.17	4.63 ±0.14
	FedRep	65.41 ±0.26	66.14 ±0.18	65.06 ±0.19	5.02 ±0.12
	HeLoRA-Pad	65.72 ±0.26	65.44 ±0.17	64.69 ±0.24	4.48 ±0.09
	FLoRA	66.38 ±0.15	65.87 ±0.19	66.04 ±0.31	4.88 ±0.08
	FlexLoRA	66.12 ±0.21	65.76 ±0.12	65.73 ±0.16	4.79 ±0.13
	FedPLoRA	66.51 ±0.22	66.16 ±0.19	65.89 ±0.17	5.01 ±0.12
	FedRRA	**66.78** ±0.17	**66.41** ±0.23	**66.29** ±0.14	**5.11** ±0.08

0.09 (FLoRA) points, respectively. On MT-Bench, FedRRA scores 5.19, leading FedPLoRA by 0.16 points.

Diamond Rank Allocation. As shown in Table 9 under the Diamond allocation strategy, FedRRA achieves the highest federated mean in all comparisons.

FedRRA achieves the highest federated mean on all four TinyLlama metrics. It leads FlexLoRA by 0.52 points on Dolly MMLU and 0.48 points on Alpaca MMLU, while exceeding FedPLoRA by 0.23 points on Wizard MMLU and 0.12 points on MT-Bench. Its lead therefore holds across datasets despite the change in the strongest competing method.

The result extends to Llama2-7B: FedRRA leads FlexLoRA by 0.48 points on Dolly MMLU and 0.29 points on Alpaca MMLU, and FLoRA by 0.23 points on Wizard MMLU and 0.11 points on MT-Bench. The same ranking across both model scales suggests that FedRRA's advantage under Diamond allocation does not depend on the availability of high-rank clients.

For Qwen2.5-3B, FedRRA exceeds FedPLoRA by 0.27 points on Dolly MMLU and 0.25 points on Wizard MMLU. Its gains over the closest competitors are 0.25 points against FLoRA on Alpaca MMLU and 0.09 points against FedRep on MT-Bench.

Head Rank Allocation. As shown in Table 10, under the Head allocation strategy, FedRRA achieves the highest federated mean on all four metrics for each of the three models.

The Head allocation assigns rank 4 to seven of the ten clients, so only a small minority can train higher-rank adapters. Despite this uneven rank distribution, FedRRA achieves the highest federated mean on every reported metric. With TinyLlama, it leads FedPLoRA by 0.20 points on Dolly MMLU, 0.26 points on Wizard MMLU, and 0.09 points on MT-Bench. On Alpaca MMLU, its closest competitor is instead FlexLoRA, which it exceeds by 0.35 points. Thus, the lead is maintained across datasets even as the strongest baseline changes.

The same pattern extends to Llama2-7B. FedRRA exceeds FedPLoRA by 0.31 points on Dolly MMLU, 0.28 points on Wizard MMLU, and 0.10 points on MT-Bench, while leading FlexLoRA by

Table 10: Comparison of FedRRA with baselines on MMLU and MT-Bench under the Head rank allocation strategy.

Foundation model	Method Head	MMLU			MT-Bench
		Dolly	Wizard	Alpaca	Wizard
TinyLlama	Centralized	26.95 ±0.01	26.44 ±0.01	26.64 ±0.02	2.84 ±0.03
	HETLoRA	24.88 ±0.26	24.78 ±0.24	25.12 ±0.18	2.21 ±0.08
	FedRep	25.31 ±0.19	25.14 ±0.21	25.49 ±0.18	2.37 ±0.12
	HeLoRA-Pad	25.25 ±0.31	25.11 ±0.22	25.48 ±0.16	2.29 ±0.11
	FLoRA	25.64 ±0.16	25.27 ±0.22	25.36 ±0.19	2.33 ±0.07
	FlexLoRA	25.61 ±0.15	25.51 ±0.14	25.89 ±0.18	2.41 ±0.07
	FedPLoRA	26.04 ±0.14	25.72 ±0.13	25.71 ±0.23	2.49 ±0.09
	FedRRA	**26.24** ±0.12	**25.98** ±0.15	**26.24** ±0.19	**2.58** ±0.06
Llama2	Centralized	47.76 ±0.12	46.82 ±0.08	46.86 ±0.14	3.77 ±0.09
	HETLoRA	45.65 ±0.29	44.81 ±0.24	45.24 ±0.18	3.16 ±0.13
	FedRep	46.18 ±0.22	45.21 ±0.19	44.99 ±0.15	3.35 ±0.07
	HeLoRA-Pad	46.02 ±0.25	45.12 ±0.18	45.16 ±0.16	3.29 ±0.09
	FLoRA	46.54 ±0.19	45.64 ±0.22	45.52 ±0.18	3.44 ±0.11
	FlexLoRA	46.31 ±0.21	45.43 ±0.32	45.88 ±0.15	3.41 ±0.05
	FedPLoRA	46.81 ±0.22	45.71 ±0.17	45.61 ±0.15	3.49 ±0.12
	FedRRA	**47.12** ±0.17	**45.99** ±0.14	**46.21** ±0.16	**3.59** ±0.07
Qwen2.5	Centralized	67.27 ±0.04	66.86 ±0.03	66.69 ±0.06	5.37 ±0.09
	HETLoRA	65.40 ±0.19	65.62 ±0.05	65.50 ±0.16	4.75 ±0.03
	FedRep	65.97 ±0.21	65.41 ±0.16	65.12 ±0.14	4.71 ±0.07
	HeLoRA-Pad	65.75 ±0.17	65.21 ±0.23	65.22 ±0.27	4.62 ±0.09
	FLoRA	66.18 ±0.31	65.91 ±0.15	65.89 ±0.23	4.86 ±0.05
	FlexLoRA	65.81 ±0.22	66.09 ±0.12	66.08 ±0.16	4.92 ±0.05
	FedPLoRA	65.34 ±0.26	65.64 ±0.23	65.52 ±0.17	4.79 ±0.09
	FedRRA	**66.57** ±0.19	**66.34** ±0.18	**66.18** ±0.16	**5.05** ±0.06

0.33 points on Alpaca MMLU. Its advantage under Head allocation is therefore observed at both the TinyLlama and Llama2-7B model scales, rather than being confined to one backbone or evaluation dataset.

For Qwen2.5-3B, FLoRA is the strongest competing method on Dolly MMLU, where FedRRA leads by 0.39 points. FlexLoRA is closest on Wizard MMLU, Alpaca MMLU, and MT-Bench, with gaps of 0.25, 0.10, and 0.13 points, respectively. Taken together, these results suggest that FedRRA can maintain its advantage when most clients have small rank budgets. Across the three rank allocations and three backbones, it achieves the highest mean among federated methods.

D.2 Results with More Clients

We extend the Triangle rank allocation setting from 10 to 50 clients. Each of the five rank levels contains ten clients. We retain the default Dirichlet concentration parameter $\alpha = 0.5$ and the other experimental settings. Table 11 reports the MMLU results on TinyLlama and Qwen2.5-3B.

On TinyLlama, FedRRA exceeds the strongest federated baseline, FedPLoRA, by 0.25, 0.42, and 0.40 points on Dolly, Wizard, and Alpaca, respectively. On Qwen2.5-3B, it leads FedPLoRA by 0.34 points on Dolly and FedRep by 0.10 and 0.48 points on Wizard and Alpaca. Compared with the corresponding ten-client results, increasing the number of clients does not uniformly raise the absolute MMLU scores. Nevertheless, with 50 clients, FedRRA retains the highest federated mean across both models and all three datasets.

D.3 Results under Stronger Data Heterogeneity

We further evaluate the methods with a Dirichlet concentration parameter of $\alpha = 0.1$, compared with $\alpha = 0.5$ in the main experiments. We use ten clients and retain the Triangle rank allocation and the other experimental settings. Table 12 reports the MMLU results on TinyLlama and Qwen2.5-3B.

Under stronger data heterogeneity, FedRRA's MMLU scores are lower than in the default $\alpha = 0.5$ setting, indicating that this is a more challenging evaluation condition. On TinyLlama, it leads

Table 11: MMLU results with 50 clients under Triangle rank allocation.

Foundation model	Method Triangle	Dolly	Wizard	Alpaca
TinyLlama	Centralized	26.95 ±0.01	26.44 ±0.01	26.64 ±0.02
	HETLoRA	25.12 ±0.17	24.77 ±0.13	25.12 ±0.11
	FedRep	25.16 ±0.18	25.21 ±0.18	25.33 ±0.13
	HeLoRA-Pad	24.67 ±0.23	24.86 ±0.14	25.29 ±0.15
	FLoRA	25.64 ±0.14	25.59 ±0.23	25.46 ±0.16
	FlexLoRA	25.61 ±0.24	25.22 ±0.18	25.04 ±0.19
	FedPLoRA	25.86 ±0.17	25.61 ±0.26	25.59 ±0.15
	FedRRA	**26.11** ±0.18	**26.03** ±0.16	**25.99** ±0.15
Qwen2.5-3B	Centralized	67.27 ±0.04	66.86 ±0.03	66.69 ±0.06
	HETLoRA	65.62 ±0.25	65.48 ±0.24	65.79 ±0.16
	FedRep	65.46 ±0.21	66.21 ±0.17	65.94 ±0.14
	HeLoRA-Pad	65.21 ±0.22	65.54 ±0.15	65.42 ±0.16
	FLoRA	65.89 ±0.23	65.78 ±0.28	65.91 ±0.16
	FlexLoRA	65.75 ±0.21	65.89 ±0.27	65.78 ±0.19
	FedPLoRA	66.11 ±0.17	65.88 ±0.15	65.85 ±0.21
	FedRRA	**66.45** ±0.18	**66.31** ±0.13	**66.42** ±0.17

Table 12: MMLU results under stronger data heterogeneity ($\alpha = 0.1$) with ten clients and Triangle rank allocation.

Foundation model	Method Triangle	Dolly	Wizard	Alpaca
TinyLlama	Centralized	26.95 ±0.01	26.44 ±0.01	26.64 ±0.02
	HETLoRA	24.58 ±0.17	24.76 ±0.13	24.66 ±0.19
	FedRep	24.85 ±0.23	24.45 ±0.15	24.97 ±0.21
	HeLoRA-Pad	24.81 ±0.24	24.76 ±0.19	24.68 ±0.22
	FLoRA	25.23 ±0.24	25.11 ±0.19	25.24 ±0.18
	FlexLoRA	24.86 ±0.18	24.68 ±0.16	24.96 ±0.19
	FedPLoRA	25.51 ±0.23	25.14 ±0.21	25.71 ±0.16
	FedRRA	**25.85** ±0.17	**25.43** ±0.13	**25.76** ±0.18
Qwen2.5-3B	Centralized	67.27 ±0.04	66.86 ±0.03	66.69 ±0.06
	HETLoRA	64.86 ±0.32	64.95 ±0.24	65.12 ±0.23
	FedRep	65.14 ±0.33	65.31 ±0.18	65.23 ±0.22
	HeLoRA-Pad	64.76 ±0.24	64.77 ±0.23	65.11 ±0.14
	FLoRA	65.75 ±0.22	65.12 ±0.18	65.68 ±0.23
	FlexLoRA	65.42 ±0.18	65.08 ±0.15	65.21 ±0.19
	FedPLoRA	65.79 ±0.28	65.68 ±0.21	65.32 ±0.15
	FedRRA	**66.31** ±0.23	**65.97** ±0.18	**65.82** ±0.19

FedPLoRA by 0.34, 0.29, and 0.05 points on Dolly, Wizard, and Alpaca. On Qwen2.5-3B, the corresponding gains are 0.52 and 0.29 points over FedPLoRA on Dolly and Wizard, and 0.14 points over FLoRA on Alpaca. The TinyLlama–Alpaca margin is small, so that individual comparison should be interpreted cautiously. Overall, at $\alpha = 0.1$, FedRRA maintains the highest federated mean in all settings.

D.4 Impact of Rank Heterogeneity on Performance

Diamond Rank Allocation. As shown in Figure 7, consistent with the Triangle results, FedRRA continues to exhibit the desirable property that, on both TinyLlama (2.61 vs. 2.45–2.54) and Llama2-7B (3.63 vs. 3.39–3.52), the global model consistently outperforms all local models across different ranks, confirming that heterogeneous low-rank updates can be effectively aggregated without incurring information loss. In contrast, FedRep, Zero-Padding, and HeLoRA-Pad still exhibit the inverse phenomenon in some cases. On TinyLlama, the FedRep global model (2.36) is outperformed by its $r = 4$ local model (2.41), while the Zero-Padding global model (2.29) is surpassed by its $r = 8$ local model (2.32), and the HeLoRA-Pad global model (2.15) is surpassed by its $r = 16$ and $r = 4$ local models (2.19 and 2.24). A similar degradation is observed on Llama2-7B for Zero-Padding and HeLoRA-Pad, whose global models (3.34 and 3.24, respectively) are outperformed by their $r = 8$

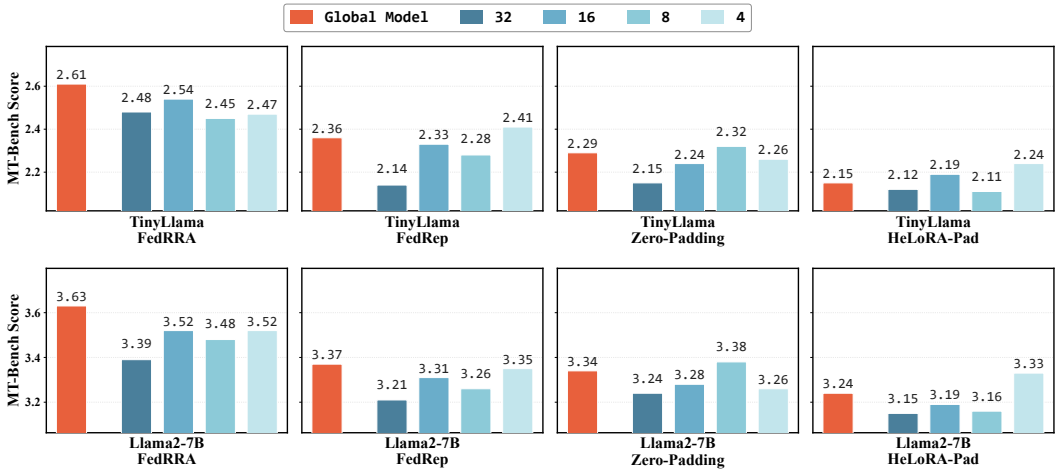


Figure 7: Impact of the Diamond rank allocation strategy on Wizard.

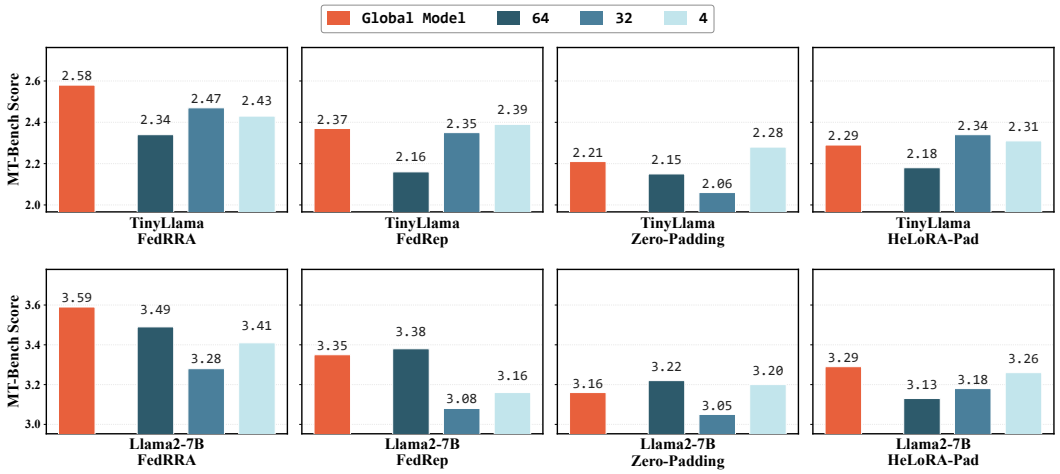


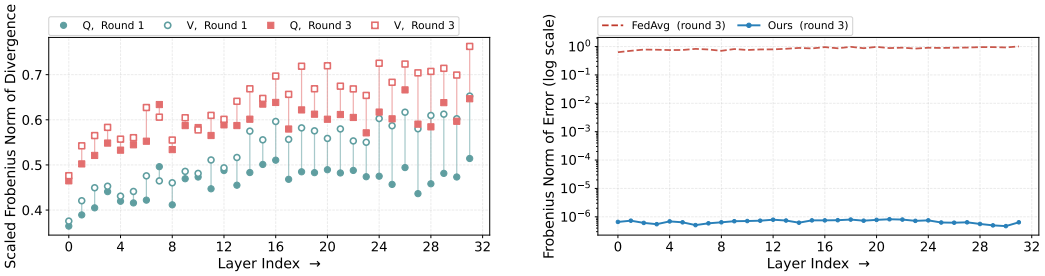
Figure 8: Impact of the Head rank allocation strategy on Wizard.

and $r = 4$ local models (3.38 and 3.33). The FedRep global model (3.37), however, remains above all its local models on Llama2-7B.

Head Rank Allocation. As shown in Figure 8, the FedRRA global model remains the best performer on both backbones (2.58 on TinyLlama and 3.59 on Llama2-7B), strictly dominating all of its local counterparts and indicating robust aggregation under extreme rank heterogeneity. FedRep, Zero-Padding, and HeLoRA-Pad, on the other hand, exhibit cases of instability: on TinyLlama, the FedRep global model (2.37) is surpassed by its $r = 4$ local model (2.39), the Zero-Padding $r = 32$ client falls to 2.06, below its global model (2.21), while its $r = 4$ client reaches 2.28, and the HeLoRA-Pad global model (2.29) is surpassed by its $r = 32$ and $r = 4$ local models (2.34 and 2.31). On Llama2-7B, both the FedRep and Zero-Padding global models (3.35 and 3.16) are outperformed by their $r = 64$ local models (3.38 and 3.22), whereas the HeLoRA-Pad global model (3.29) remains above its local models. From a per-rank perspective, the high-rank clients ($r = 64$ and $r = 32$) under FedRRA outperform their counterparts under FedRep, Zero-Padding, and HeLoRA-Pad on both backbones, suggesting greater robustness to padding-induced noise across ranks.

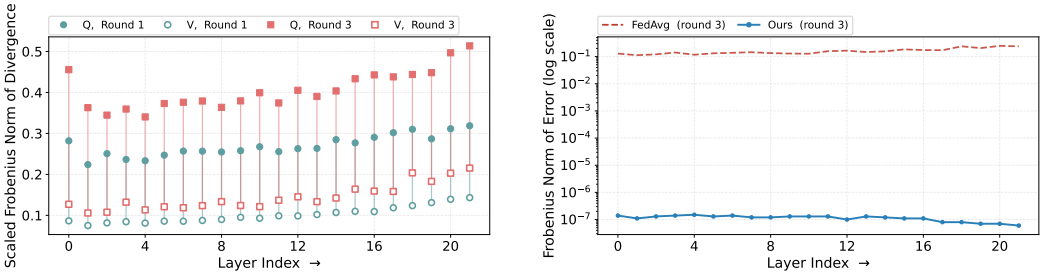
D.5 Impact of Aggregation Noise

Absolute Aggregation Deviation and Residual Correction. To quantify the deviation of FedAvg updates from the ideal, we measure the Frobenius norm of their difference in the query and value matrices in Rounds 1 and 3. As shown in Figure 9a, the deviation grows with model depth and



(a) Llama2 - Layer Query/Value Deviation (b) Llama2 - Residual correction effect

Figure 9: Aggregation deviation and the effect of residual correction.



(a) TinyLlama - Layer Query/Value Deviation (b) TinyLlama - Residual correction effect

Figure 10: Aggregation deviation and the effect of residual correction.

communication rounds, and is more pronounced in V than in Q matrices. To verify the effectiveness of our residual correction, we compare the aggregation error before and after applying it. As shown in Figure 9b, residual correction reduces the error by approximately six orders of magnitude in this analysis.

We further characterize aggregation error using the relative client residual ratio:

$$\rho_i = \frac{\|\Delta W_{res_i}\|_F}{\|\Delta W_{nat}\|_F}, \tag{20}$$

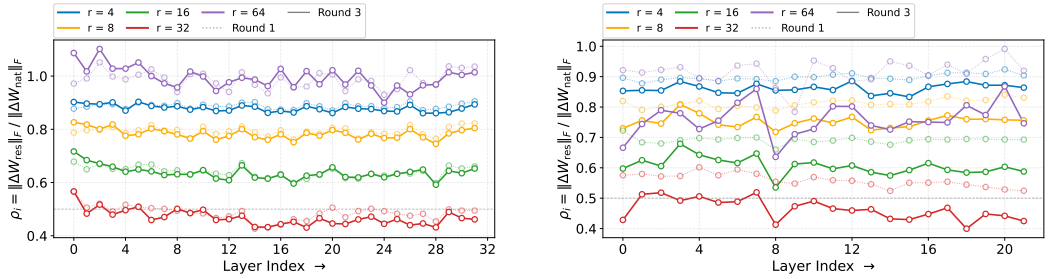
and the ideal update magnitude:

$$\|\Delta W_{nat}\|_F. \tag{21}$$

We also present the corresponding results on TinyLlama, as shown in Figure 10. The aggregation deviation and correction effect are consistent with the observations on Llama2-7B. Interestingly, however, we find a distinct phenomenon: on Llama2-7B, the deviation of the V projection is larger than that of Q, whereas the opposite holds on TinyLlama. We attribute this discrepancy to architectural differences: in smaller models, adaptation pressure concentrates more on the Q pathway, while the deeper Llama2-7B allocates more updates to V.

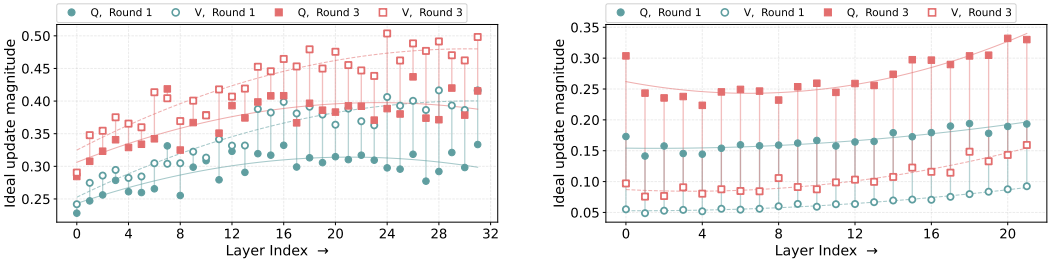
The Relative Residual Ratio. Figure 11 plots ρ_i. On both backbones, the curves form clearly separated horizontal bands ordered by rank, and each band is nearly flat across depth, indicating that the residual magnitude of a client is predominantly determined by its own rank r_i rather than by which layer it sits in. A more subtle observation is that the ordering of the bands differs between the two backbones. On Llama2-7B, the band order stays consistent with the client rank: $r = 64$ sits at the top with $\rho \geq 1$, while $r = 32$ remains the lowest. On TinyLlama, however, the order is inverted at later rounds: by Round 3, $r = 4$ rises to the top, while $r = 32$ and $r = 64$ settle in the middle-to-low region.

This divergence reflects the interplay between the effective rank of ΔW_{nat} and the capacity of the backbone. On a large model, the aggregated update lies in a genuinely low-dimensional subspace; once r_i exceeds this effective rank, additional dimensions inject off-direction noise rather than useful signal, pushing ρ to approach or exceed 1. On a smaller model, the aggregated update is closer to



(a) Llama2 (b) TinyLlama

Figure 11: Layer-wise relative residual ratio $\rho_i = \|\Delta W_{\text{res}}\|_F / \|\Delta W_{\text{nat}}\|_F$, evaluated at Round 1 and Round 3.



(a) Llama2 (b) TinyLlama

Figure 12: Layer-wise ideal update magnitude $\|\Delta W_{\text{nat}}\|_F$ of the Q and V projections at Round 1 and Round 3.

full rank relative to the module width, so higher-rank clients capture most of the signal, and only low-rank clients (e.g., $r = 4$) are left with a large uncovered component, driving their ρ to the top.

Despite the different ordering, the conclusion is consistent across both regimes: clients with different ranks exhibit inherently different biases.

The Ideal Update Magnitude. Figure 12 reports the ideal update magnitude $\|\Delta W_{\text{nat}}\|_F$. We observe that the variation of the ideal update magnitude aligns with the deviation trend, both of which increase with training rounds, indicating a strong correlation with the training process. Notably, in Llama2-7B, the overall trend follows an inverted-U shape: the magnitude first rises with depth and saturates around layers 20–24. Shallow layers exhibit small updates, while middle layers undertake significantly more updates than deep layers, which is consistent with findings in recent LoRA research (Gao et al., 2025a). In contrast, TinyLlama shows a monotonically increasing trend across layers. These observations motivate us to further investigate federated fine-tuning from a layer-wise perspective.

D.6 Convergence Analysis

To justify the round budgets used in our main experiments (Table 7) and to compare the convergence behavior of different methods, we extend training to eight communication rounds and track per-round performance of FedRRA, FLoRA, and FlexLoRA under the Triangle rank allocation strategy on TinyLlama and Qwen2.5-3B. As shown in Figures 13 and 14, the three methods exhibit consistent convergence patterns across both models, though with clear differences in speed and final performance level.

TinyLlama. FedRRA reaches its peak MMLU performance by round 4 (25.69 on Wizard, 25.34 on Dolly) and remains stable thereafter, with only marginal fluctuation. MT-Bench peaks earlier at round 2 (2.73) and exhibits a moderate decline in later rounds, which we attribute to mild overfitting

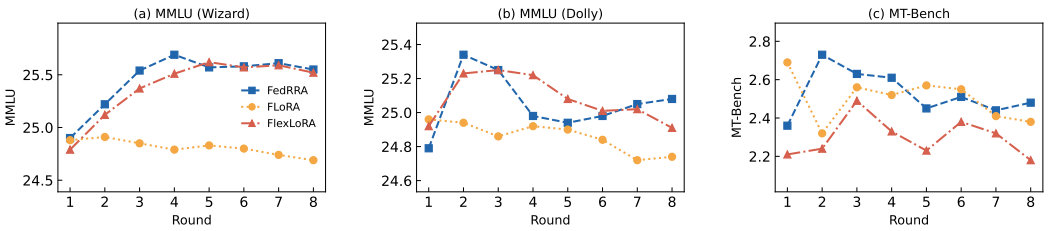


Figure 13: Per-round performance of FedRRA, FLoRA, and FlexLoRA on **TinyLlama** under the Triangle rank allocation strategy over eight communication rounds.

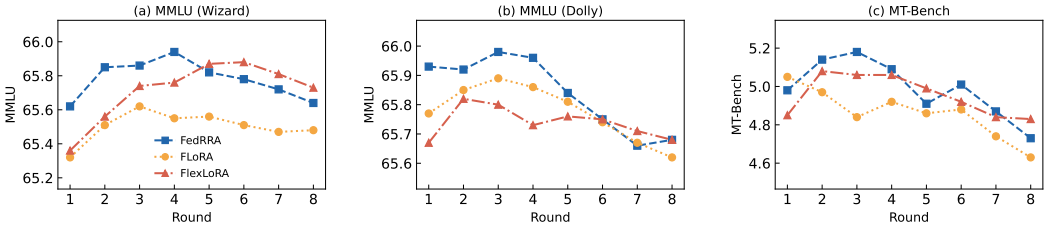


Figure 14: Per-round performance on **Qwen2.5-3B** over eight communication rounds. Layout identical to Figure 13.

under prolonged training on the relatively small federated datasets. FLoRA converges quickly from round 1 but plateaus at a consistently lower MMLU level (around 24.8–24.9), and shows a steady decline on both MMLU and MT-Bench after round 5, suggesting greater susceptibility to overfitting. FlexLoRA converges more slowly, reaching its MMLU plateau around round 5 (25.62 on Wizard), approximately two rounds later than FedRRA, and exhibits considerable instability on MT-Bench throughout training.

Qwen2.5-3B. On the stronger Qwen2.5-3B backbone, FedRRA again converges within three to four rounds, peaking at round 3 on MMLU (65.86 on Wizard, 65.98 on Dolly) and at round 3 on MT-Bench (5.18), before exhibiting a gradual decline in later rounds consistent with mild overfitting. FLoRA converges to a lower MMLU level than FedRRA across all rounds, and its MT-Bench score declines steadily from round 1 (5.05) to round 8 (4.63), indicating a more pronounced overfitting tendency on this backbone. FlexLoRA requires five to six rounds to stabilize on MMLU (peaking at round 6 with 65.88 on Wizard), roughly two rounds later than FedRRA, and shows a similar post-peak decline on MT-Bench.

Overall, FedRRA achieves the best performance while converging faster than FlexLoRA across both backbones, and the configured round budgets in our main experiments are sufficient to capture the peak performance of all methods under comparison.

E Additional Analysis

E.1 Why Consensus Direction Is Built on LoRA B Rather Than LoRA A

In Section 3.1, we construct the consensus direction using B rather than A. We explain this choice through LoRA's initialization and optimization dynamics, and then examine the directional variation observed across clients.

Under standard LoRA initialization, A is randomly initialized and B is initialized to zero. Let G_i denote the gradient of client i's loss with respect to $\Delta W_i = B_i A_i$. Then

$$\frac{\partial L_i}{\partial B_i} = G_i A_i^\top, \qquad \frac{\partial L_i}{\partial A_i} = B_i^\top G_i. \tag{22}$$

At initialization, the gradient of A_i is zero, whereas B_i can immediately receive updates induced by client i's local data. This suggests that B may capture client-dependent directions earlier in training.

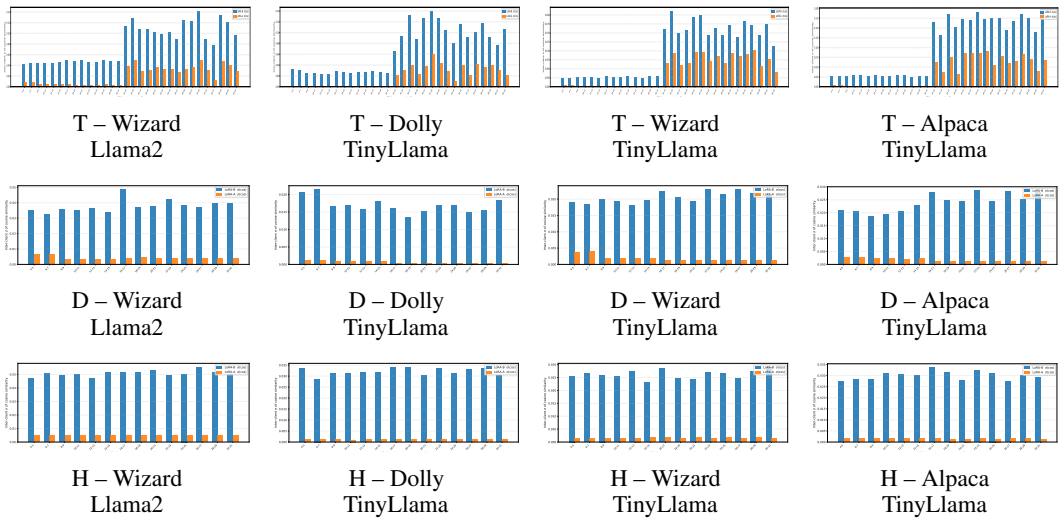


Figure 15: Inter-client standard deviation of pairwise cosine similarities along each dimension for LoRA-B (blue) and LoRA-A (orange), evaluated across different rank allocation strategies and datasets.

To examine this empirically, for each eligible rank dimension k, we compute the inter-client standard deviation of pairwise cosine similarities separately for the k-th columns of B and the k-th rows of A. A larger standard deviation indicates that the similarities vary more across client pairs, providing a more discriminative signal for directional gating. Figure 15 shows substantially greater variation on the B side than on the A side across the evaluated rank allocations and datasets. The A-side similarities are comparatively uniform, so using A alone provides less information for distinguishing client update directions. This observation is consistent with prior findings on the different roles of the two LoRA factors (Guo et al., 2024; Wang et al., 2025). CAP therefore uses B for directional gating, while retaining both A and B when constructing the padded adapter.

To directly evaluate the choice of consensus direction, we compare gating based on the A factor, the complete rank-one update BA, and the B factor. All other components of CAP and the experimental settings are kept fixed.

Table 13: Different consensus-direction choices under the Triangle rank allocation strategy.

Foundation model	**Consensus direction**	**MMLU**		
		Dolly	**Wizard**	**Alpaca**
TinyLlama	A	25.26	25.21	25.36
	BA	25.89	25.72	25.83
	B	**26.37**	**25.96**	**26.08**
Qwen2.5-3B	A	64.92	65.19	65.34
	BA	66.34	66.19	65.98
	B	**66.89**	**66.49**	**66.21**

Table 13 shows that using B for directional gating achieves the highest MMLU score on all three datasets for both TinyLlama and Qwen2.5-3B. Compared with using the complete rank-one update BA, the B-based variant improves MMLU by 0.24–0.48 points on TinyLlama and 0.23–0.55 points on Qwen2.5-3B, while using A alone performs worst in all six settings. These results support the use of B as the directional signal in CAP, consistent with its greater inter-client variation observed in Figure 15.

E.2 CAP Performance Under the Observation 2 Setting

Observation 2 in Section 2.3 demonstrates that replication padding can inject noise into $r=32$ and $r=4$ clients when high-rank clients carry low-quality data. Here we supplement that experiment by evaluating CAP under the same setting to verify whether CAP can better adapt to this scenario.

As shown in Table 14, CAP consistently achieves higher After scores than replication padding across all rank groups and rounds, and outperforms zero padding in most settings. The improvement is most pronounced for $r=32$ clients, where replication padding suffers the most severe degradation: CAP yields After scores of 28.01, 30.11, and 29.34 in rounds 1–3 respectively, compared to 26.99, 28.03, and 26.16 for replication padding — a consistent margin of approximately 1–3 points. Similarly, for $r=4$ clients, CAP consistently outperforms replication padding (e.g., 28.42 vs. 26.36 in Round 1), further confirming its ability to mitigate noise propagation across all lower-rank groups. This confirms that CAP's direction-aware soft gating effectively suppresses the contribution of low-quality $r=64$ clients, preventing their noisy updates from propagating to lower-rank groups. For $r=64$ clients themselves, CAP also achieves comparable or superior After scores (27.42, 30.01, 30.17) relative to zero padding (27.56, 29.25, 29.95), indicating that the gating mechanism does not discard useful signal when it is present.

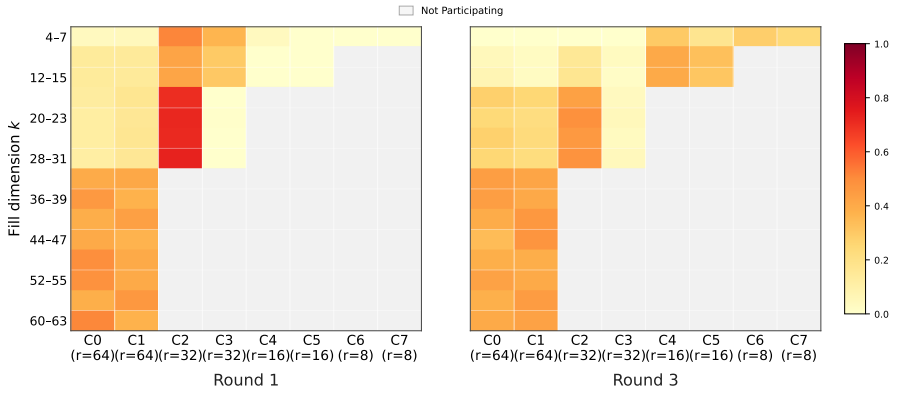
Table 14: Performance of Zero Padding, Replication Padding, and CAP under the Observation 2 setting.

Rank	**Padding**	**Round 1**		**Round 2**		**Round 3**	
		Before	After	Before	After	Before	After
r=64	Zero	29.00	27.56	31.16	29.25	29.87	29.95
	Replication	29.06	28.00	28.32	28.78	29.50	29.42
	CAP	29.15	27.42	31.50	30.01	30.56	30.17
r=32	Zero	23.72	27.37	27.70	29.95	27.84	28.25
	Replication	23.16	26.99	26.87	28.03	25.75	26.16
	CAP	23.35	28.01	27.85	30.11	27.97	29.34
r=4	Zero	30.42	29.18	28.21	29.02	28.94	28.96
	Replication	30.42	26.36	26.81	26.77	26.17	27.86
	CAP	30.42	28.42	28.63	29.16	28.98	29.21

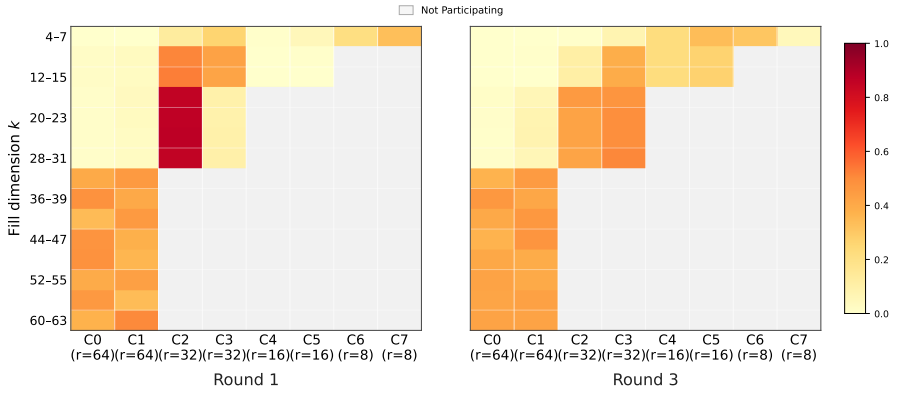
E.3 Client Contribution Weights in CAP

To further verify whether the direction-aware gating in CAP plays a discriminative role during training, we conduct experiments on the client $r = 4$ and visualize the normalized contribution weights across padded dimensions for Rounds = {1, 3}.

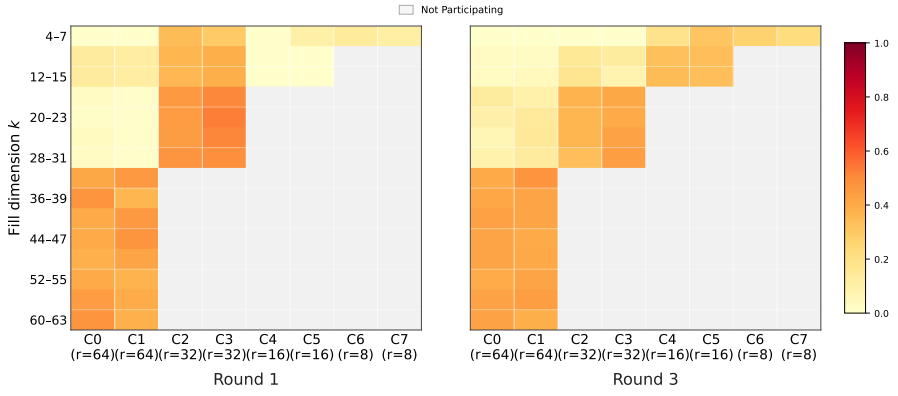
From Figure 16, we observe that at the early stage of training, the contribution weights exhibit clear non-uniformity across each padded dimension. For example, in Round 1, C2 ($r=32$) consistently attains the highest weight over $k \in [12, 31]$, indicating that the direction-aware gating $\alpha_j(k)$ effectively distinguishes clients on a per-dimension basis. Moreover, medium-rank clients tend to dominate in low-dimensional padding regions: in Round 1, for $k \in [4, 31]$, clients with $r = 32$ generally receive higher weights than those with $r=64$, rather than the weighting being biased simply toward higher-rank clients. As training progresses, the weight distribution becomes noticeably smoother. This is consistent with expectations, as multi-round federated training gradually aligns the directions of the B matrices across clients, thereby reducing inter-client cosine differences. Overall, these observations demonstrate that CAP produces meaningful and adaptive contribution differences under heterogeneous rank settings, while naturally converging toward a more uniform weighting as client representations become aligned.



(a) Dolly dataset



(b) Wizard dataset



(c) Alpaca dataset

Figure 16: Client contribution weights across datasets under Triangle strategy. Each heatmap shows the normalized contribution weight $\hat{\omega}_j(k) = \omega_j(k)/\sum_{j' \in \mathcal{K}_k} \omega_{j'}(k)$ with $\omega_j(k) = s_j(k) \cdot \alpha_j(k)$. Gray cells indicate that a client does not participate due to insufficient native rank. Results are shown for different training stages to illustrate the evolution of contribution weights.

E.4 Detailed Communication and Server Overhead

Let K be the number of participating clients, $S = \sum_{i=1}^{K} r_i$ the sum of their ranks, $R = \max_i r_i$, and r_s the residual compression rank. For adapted matrices of sizes $m_\ell \times n_\ell$, define $Q = \sum_\ell (m_\ell + n_\ell)$, $D = \sum_\ell m_\ell n_\ell$, and $V = \sum_\ell m_\ell n_\ell \min(m_\ell, n_\ell)$. A rank-$r$ adapter contains rQ parameters. Table 15

summarizes the per-round server complexity and total upload-plus-download parameter counts. These expressions count adapter payloads and exclude the initial backbone distribution and control metadata.

Table 15: Per-round server complexity and communication under heterogeneous ranks.

Method	Server complexity	Communication
HETLoRA	$\mathcal{O}(KRQ)$	$2SQ$
FLoRA	$\mathcal{O}(SQ)$	$(K+1)SQ$
FlexLoRA	$\mathcal{O}((S+K)D+V)$	$2SQ$
FedPLoRA	$\mathcal{O}(SQ+KR)$	$(S+KR)Q$
FedRRA	$\mathcal{O}(SQ+(S+K)D+V)$	$(2S+Kr_s)Q$

FedPLoRA uploads the selected rank-r_i components and broadcasts the full rank-R collection to each client (Zhang et al., 2026). The FedRRA communication results reported below are evaluated under the default setting of $r_s = 16$ used in this paper.

Tables 16 and 17 report results on Dolly under Triangle rank allocation with 10 and 50 clients. Communication denotes the upload-plus-download parameter count per round. Server time and arithmetic GFLOPs are averaged per round, whereas CPU and GPU memory entries report peak values. The arithmetic FLOP counter excludes SVD kernels, although server time includes their execution.

Table 16: Per-round communication and server cost on Dolly with 10 clients under Triangle rank allocation.

Foundation model	Method	Comm. (M params)	Server time (s)	GFLOPs	Peak CPU (GiB)	Peak GPU (GiB)
TinyLlama	HETLoRA	69.84	0.51	1.571	3.32	4.43
	FLoRA	384.10	0.52	0.035	2.51	4.28
	FlexLoRA	69.84	17.89	54.918	3.46	4.18
	FedPLoRA	125.03	0.98	0.035	3.70	4.37
	FedRRA	92.36	27.78	78.327	6.12	4.43
Qwen2.5-3B	HETLoRA	114.28	2.13	2.201	4.20	7.71
	FLoRA	628.53	0.96	0.057	3.42	7.13
	FlexLoRA	114.28	36.39	89.866	5.58	7.05
	FedPLoRA	204.60	1.57	0.057	4.86	7.56
	FedRRA	151.14	49.44	116.349	7.81	7.91

Table 17: Per-round communication and server cost on Dolly with 50 clients under Triangle rank allocation.

Foundation model	Method	Comm. (M params)	Server time (s)	GFLOPs	Peak CPU (GiB)	Peak GPU (GiB)
TinyLlama	HETLoRA	349.18	9.90	7.324	4.23	4.95
	FLoRA	8904.19	3.18	0.175	2.87	4.83
	FlexLoRA	349.18	23.75	273.332	6.32	5.18
	FedPLoRA	625.15	2.77	0.175	7.90	5.98
	FedRRA	461.82	43.11	380.850	9.71	6.54
Qwen2.5-3B	HETLoRA	571.39	24.65	13.413	6.19	7.38
	FLoRA	14570.50	4.69	0.286	3.97	7.52
	FlexLoRA	571.39	59.75	447.271	5.87	7.05
	FedPLoRA	1022.98	6.23	0.286	8.45	7.55
	FedRRA	745.71	71.93	561.366	9.21	8.08

From 10 to 50 clients, FLoRA's communication grows by approximately 23 times, whereas the other methods grow by about five times. FedRRA communicates substantially less than FLoRA: approximately 4.2 times less with 10 clients and 19–20 times less with 50 clients. It also transmits fewer parameters than FedPLoRA, although its residual correction requires more communication than

HETLoRA and FlexLoRA. This saving is accompanied by the highest server time and arithmetic cost among the profiled methods, with increased peak memory in several settings. Thus, FedRRA trades greater server computation for lower communication than methods that transmit larger global adapters. Because client–server exchange recurs in every round, communication is a primary scalability concern in bandwidth-constrained federated LoRA deployments.

Table 18: Ablation study on CAP and ARC with TinyLlama under the Diamond rank allocation strategy.

Method	Setting	MMLU			MT-Bench
		Dolly	Wizard	Alpaca	Wizard
Zero-Padding	w/o ARC	25.21 25.31 (+0.10)	25.03 25.21 (+0.18)	24.96 25.19 (+0.23)	2.29 2.32 (+0.03)
HeLoRA-Pad	w/o ARC	24.89 25.19 (+0.30)	24.76 25.08 (+0.32)	24.67 25.10 (+0.43)	2.15 2.26 (+0.11)
Replication	w/o ARC	24.89 25.42 (+0.53)	25.44 25.67 (+0.23)	25.31 25.61 (+0.30)	2.36 2.46 (+0.10)
FedRRA	w/o ARC	**25.82** 26.38 (+0.56)	**25.58** 26.09 (+0.51)	**25.86** 26.21 (+0.45)	**2.49** 2.61 (+0.12)

Table 19: Ablation study on CAP and ARC with TinyLlama under the Head rank allocation strategy.

Method	Setting	MMLU			MT-Bench
		Dolly	Wizard	Alpaca	Wizard
Zero-Padding	w/o ARC	24.88 25.04 (+0.16)	24.78 24.93 (+0.15)	25.12 25.26 (+0.14)	2.21 2.28 (+0.07)
HeLoRA-Pad	w/o ARC	25.25 25.49 (+0.24)	25.11 25.42 (+0.31)	25.48 25.74 (+0.26)	2.29 2.38 (+0.09)
Replication	w/o ARC	25.31 25.72 (+0.41)	25.14 25.51 (+0.37)	25.49 25.81 (+0.32)	2.37 2.46 (+0.09)
FedRRA	w/o ARC	**25.86** 26.24 (+0.48)	**25.52** 25.98 (+0.46)	**25.79** 26.24 (+0.45)	**2.48** 2.58 (+0.10)

Table 20: Ablation study on CAP and ARC with Llama2-7B under the Triangle rank allocation strategy.

Method	Setting	MMLU			MT-Bench
		Dolly	Wizard	Alpaca	Wizard
Zero-Padding	w/o ARC	45.82 46.18 (+0.36)	45.21 45.64 (+0.43)	45.08 45.43 (+0.35)	3.06 3.19 (+0.13)
HeLoRA-Pad	w/o ARC	45.46 46.24 (+0.78)	44.86 45.46 (+0.60)	45.14 45.91 (+0.77)	2.98 3.15 (+0.17)
Replication	w/o ARC	45.34 46.47 (+1.13)	45.69 46.12 (+0.43)	44.54 45.26 (+0.72)	3.13 3.28 (+0.15)
FedRRA	w/o ARC	**46.61** 47.14 (+0.53)	**45.93** 46.38 (+0.45)	**45.61** 46.21 (+0.60)	**3.28** 3.55 (+0.27)

F Additional Ablation Studies

F.1 Additional Ablation Study on CAP and ARC

Section 4.3 presents the ablation study on CAP and ARC using TinyLlama under the Triangle rank allocation strategy. Here we provide complementary results under two additional settings: (1)

TinyLlama with the Diamond and Head allocation strategies, and (2) Llama2-7B with the Triangle strategy. Together with the main-text results, these experiments cover all three rank allocation strategies on the smaller model and validate scalability to a larger model.

TinyLlama under Diamond and Head strategies. Tables 18 and 19 report the ablation results for the Diamond and Head rank allocation strategies, respectively. The trends are largely consistent with the Triangle results in the main text: CAP achieves the best performance among all padding methods under the "w/o ARC" setting in most columns, and applying ARC yields further improvements in the all cases. Applying ARC improves every reported entry, CAP combined with ARC remains the strongest configuration in most cases across both strategies.

Llama2-7B under Triangle strategy. To verify that the conclusions scale beyond smaller models, we repeat the ablation on Llama2-7B with the Triangle rank allocation strategy (Table 20). The same two findings hold: CAP remains the best-performing padding method without ARC, and ARC consistently yields further gains. Compared with TinyLlama, the absolute improvements brought by ARC tend to be larger, which suggests that residual correction is even more critical as model capacity grows, since larger models amplify the impact of aggregation errors accumulated across heterogeneous ranks.

Table 21: ARC design ablation on Llama2-7B under Triangle rank allocation.

Method	Dolly			Wizard			Alpaca		
	ARC	ARC-PT	ARC-SR	ARC	ARC-PT	ARC-SR	ARC	ARC-PT	ARC-SR
Zero-Padding	**46.18**	46.17	45.76	**45.64**	45.61	45.34	**45.43**	45.42	44.89
HeLoRA-Pad	**46.24**	45.52	45.46	**45.46**	44.86	44.69	**45.91**	45.27	45.32
Replication	**46.47**	45.79	45.61	**46.12**	45.61	45.74	**45.26**	44.76	44.61
FedRRA	**47.14**	46.45	46.51	**46.38**	45.82	45.77	**46.21**	45.68	45.59

F.2 Additional Ablation Study on ARC Design

We extend the ARC design ablation in Table 4 to Llama2-7B under Triangle rank allocation. ARC-PT changes the correction target to padded local updates, while ARC-SR replaces client-specific residuals with a shared minimum-rank residual. The default ARC results match the ARC-enabled results in the preceding CAP and ARC ablation.

Table 21 reports that the default ARC scores higher than both variants in all tasks, with average gains of 0.46 points over ARC-PT and 0.60 points over ARC-SR. ARC-SR consistently yields lower scores, supporting the use of residuals matched to each client's truncated adapter. These results agree with the TinyLlama findings in Table 4 and support both ARC design choices on a larger backbone.

Table 22: Mean-referenced soft gate ablation on Llama2-7B under Triangle rank allocation.

Method	MMLU			MT-Bench
	Dolly	Wizard	Alpaca	Wizard
FedRRA (w/o Gate)	46.65	44.83	44.50	3.33
FedRRA (Zero-Ref)	46.79	44.99	44.69	3.36
FedRRA	**47.14**	**46.38**	**46.21**	**3.55**

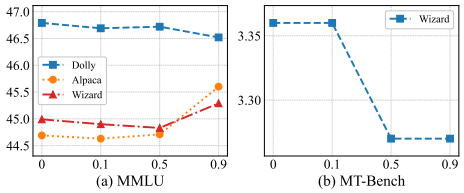


Figure 17: Effect of gate thresholds on (a) MMLU and (b) MT-Bench for Llama2-7B under Triangle rank allocation.

F.3 Additional Ablation Study on the Mean-Referenced Soft Gate

We extend the mean-referenced soft gate ablation in Table 5 to Llama2-7B under Triangle rank allocation and TinyLlama under Diamond and Head rank allocation. Each setting compares the default gate with "w/o Gate" and "Zero-Ref", alongside a fixed-threshold sweep.

Table 23: Mean-referenced soft gate ablation on TinyLlama under Diamond rank allocation.

Method	MMLU			MT-Bench
	Dolly	Wizard	Alpaca	Wizard
FedRRA (w/o Gate)	25.88	25.76	25.62	2.24
FedRRA (Zero-Ref)	25.87	25.82	25.67	2.25
FedRRA	**26.38**	**26.09**	**26.21**	**2.61**

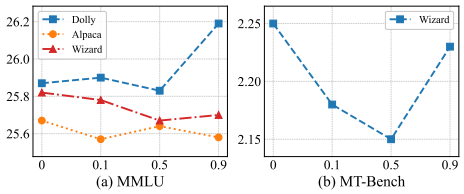


Figure 18: Effect of gate thresholds on (a) MMLU and (b) MT-Bench for TinyLlama under Diamond rank allocation.

Table 24: Mean-referenced soft gate ablation on TinyLlama under Head rank allocation.

Method	MMLU			MT-Bench
	Dolly	Wizard	Alpaca	Wizard
FedRRA (w/o Gate)	25.85	25.54	25.84	2.19
FedRRA (Zero-Ref)	25.88	25.63	25.80	2.25
FedRRA	**26.24**	**25.98**	**26.24**	**2.58**

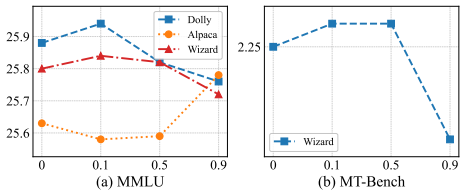


Figure 19: Effect of gate thresholds on (a) MMLU and (b) MT-Bench for TinyLlama under Head rank allocation.

Across the three additional settings, the default gate achieves the highest score in every reported column. Relative to "w/o Gate", its MMLU gains are 0.33–0.59 points on TinyLlama under Diamond and Head allocation and 0.49–1.71 points on Llama2-7B under Triangle allocation. The corresponding MT-Bench gains range from 0.22 to 0.39 points. The default also scores higher than "Zero-Ref" in every column.

The threshold plots show that no single fixed threshold performs best across all datasets and tasks, while the mean-referenced gate exceeds the tested fixed thresholds in these settings. Together with the TinyLlama Triangle results in the main text, these findings support mean-referenced gating across the evaluated backbones and rank allocation strategies.

F.4 Ablation Study on Rank-One Signal Scoring

CAP uses the Frobenius norm of a complete rank-one update as its signal score (Equation 5). This score measures update strength, whereas gradient-, Taylor-, and Fisher-based scores capture aspects of local loss sensitivity. To examine this choice, we replace only the signal score $s_i(k)$.

Let $\mathcal{B}_i$ denote the final local-training microbatch of client i. For each example $n \in \mathcal{B}_i$, we evaluate gradients at the trained adapter:

$$\mathbf{g}_{b,i,k}^{(n)} = \nabla_{\mathbf{b}_{i,k}} \ell_{i,n}, \qquad \mathbf{g}_{a,i,k}^{(n)} = \nabla_{\mathbf{a}_{i,k}} \ell_{i,n}, \tag{23}$$

where $\mathbf{b}_{i,k}$ is the k-th column of B_i and $\mathbf{a}_{i,k}^{\top}$ is the k-th row of A_i. For $x \in \{a, b\}$, define the microbatch-averaged gradient

$$\bar{\mathbf{g}}_{x,i,k} = \frac{1}{|\mathcal{B}_i|} \sum_{n \in \mathcal{B}_i} \mathbf{g}_{x,i,k}^{(n)}. \tag{24}$$

We also define the per-example directional quantity

$$d_i^{(n)}(k) = \frac{1}{2} \left(\left\langle \mathbf{g}_{b,i,k}^{(n)}, \mathbf{b}_{i,k} \right\rangle + \left\langle \mathbf{g}_{a,i,k}^{(n)}, \mathbf{a}_{i,k} \right\rangle \right). \tag{25}$$

When the loss depends on the two factors through B_iA_i, the factor of $1/2$ avoids counting the same rank-one direction twice. The alternative scores are

$$s_i^{\mathrm{Grad}}(k) = \sqrt{\|\bar{\mathbf{g}}_{b,i,k}\|_2^2 + \|\bar{\mathbf{g}}_{a,i,k}\|_2^2}, \tag{26}$$

$$s_i^{\mathrm{Taylor}}(k) = \left| \frac{1}{|\mathcal{B}_i|} \sum_{n \in \mathcal{B}_i} d_i^{(n)}(k) \right|, \tag{27}$$

$$s_i^{\mathrm{Fisher}}(k) = \frac{1}{|\mathcal{B}_i|} \sum_{n \in \mathcal{B}_i} \left(d_i^{(n)}(k) \right)^2. \tag{28}$$

The last expression is a Fisher-based empirical squared-gradient score along the rank-one direction.

We compare these scores with the default Frobenius score using TinyLlama and Qwen2.5-3B on Dolly, Wizard, and Alpaca under Triangle rank allocation. Table 25 reports MMLU accuracy.

Table 25: Rank-one signal scoring ablation under Triangle rank allocation. Entries report MMLU accuracy; bold indicates the best score for each model and dataset.

Foundation model	Dolly				Wizard				Alpaca			
	Grad.	Taylor	Fisher	Frob.	Grad.	Taylor	Fisher	Frob.	Grad.	Taylor	Fisher	Frob.
TinyLlama	25.89	26.12	25.97	**26.37**	25.42	25.79	25.62	**25.96**	25.69	25.52	25.81	**26.08**
Qwen2.5-3B	66.65	66.52	66.41	**66.89**	66.29	66.12	65.76	**66.49**	65.86	65.59	66.10	**66.21**

The Frobenius score is numerically highest in all settings. Relative to the strongest alternative in each setting, its gains range from 0.11 to 0.27 MMLU points, averaging 0.21 points. These results support measuring the strength of the complete rank-one update when weighting components for CAP's server-side padding, rather than relying on local loss sensitivity alone.

F.5 Ablation Study on SVD Truncation Rank

To analyze the trade-off between performance and communication cost introduced by SVD-compressed residual transmission, we evaluate FedRRA under different truncation ranks $r_{\mathrm{svd}} \in \{4, 8, 16, 32, 64\}$ across all three heterogeneity settings on both TinyLlama and Qwen2.5-3B.

Performance is non-monotone in truncation rank. As shown in Figures 20 and 21, neither MMLU nor MT-Bench improves monotonically with increasing r_{svd}. Across most settings, intermediate truncation ranks ($r_{\mathrm{svd}} \in \{16, 32\}$) achieve performance comparable to or better than transmitting the full residual, and certain ranks even outperform the no-truncation baseline in several settings. We attribute this to a regularization effect: truncated SVD retains only the dominant singular directions, which capture the consensus update across clients, while discarding low-energy directions that primarily encode inter-client disagreement—consistent with the low-rank assumption underlying LoRA itself. This is distinct from the exact aggregation property of ARC, which eliminates *aggregation noise* (i.e., the systematic discrepancy $\bar{B}\bar{A} \neq \sum_i p_i B_i A_i$ accumulating across rounds): SVD truncation instead introduces a one-shot, controlled approximation that we conjecture discards directions predominantly encoding client heterogeneity and padding artifacts, thereby acting as an additional denoising step. Residual correction substantially reduces aggregation noise regardless of whether SVD truncation is applied.

Communication cost decreases proportionally with truncation rank. The communication overhead of residual transmission scales as $\mathcal{O}((m+n) \cdot r_{\mathrm{svd}})$. As shown in panel (d) of Figures 20 and 21, reducing r_{svd} from 64 to 16 yields a substantial reduction in communication cost across both models and all heterogeneity settings, while incurring negligible performance degradation.

Balancing performance and communication cost, we select $r_{\mathrm{svd}} = 16$ as the default setting, which offers a favorable trade-off between model quality and communication efficiency across all evaluated configurations. This value is adopted in all main experiments.

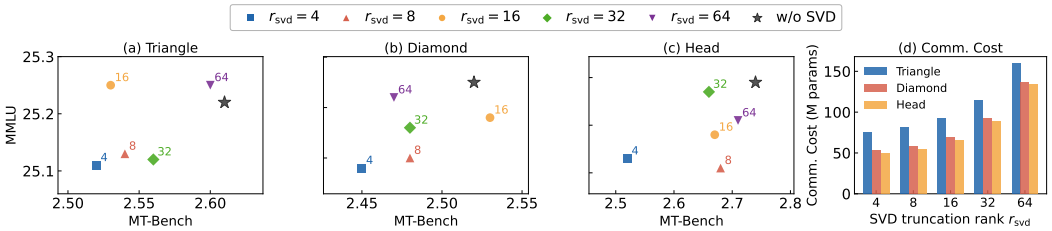


Figure 20: SVD truncation rank ablation on **TinyLlama** under Triangle, Diamond, and Head heterogeneity settings. Panels (a)–(c): scatter plots of MMLU vs. MT-Bench score for each truncation rank r_{svd}; the gray star marks the no-truncation reference. Panel (d): total communication cost (M params) per truncation rank under each heterogeneity setting.

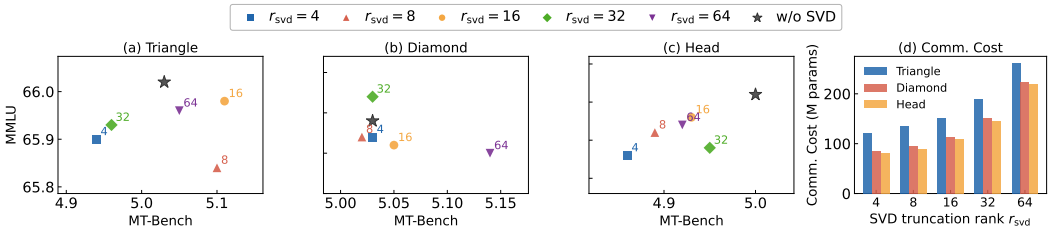


Figure 21: SVD truncation rank ablation on **Qwen2.5-3B**.

G LIMITATION

Due to resource constraints, our experiments are limited to three heterogeneous settings and models with up to 7B parameters. In future work, we plan to investigate a broader range of heterogeneous scenarios, larger-scale models, and more principled strategies for selecting the residual SVD rank to obtain more comprehensive insights. We also plan to explore how padding methods can be more effectively combined with other techniques for mitigating aggregation noise, which may further improve robustness in heterogeneous federated learning.