FLoRIST: Singular Value Thresholding for Efficient and Accurate Federated Fine-Tuning of Large Language Models

Hariharan Ramesh [1] **Jyotikrishna Dass** [1]

Abstract

Integrating Low-Rank Adaptation (LoRA) into federated learning offers a promising solution for parameter-efficient fine-tuning of Large Language Models (LLMs) without sharing local data. However, several methods designed for federated LoRA present significant challenges in balancing communication efficiency, model accuracy, and computational cost, particularly among heterogeneous clients. These methods either rely on simplistic averaging of local adapters, which introduces aggregation noise, require transmitting large stacked local adapters, leading to poor download communication efficiency, or necessitate reconstructing memory-dense global weight-update matrix and performing computationally expensive decomposition to design client-specific low-rank adapters. In this work, we propose `FLoRIST`, a federated fine-tuning framework that achieves mathematically accurate aggregation without incurring high communication or computational overhead. Instead of constructing the full global weight-update matrix at the server, `FLoRIST` employs an efficient decomposition pipeline by performing singular value decomposition on stacked local adapters separately. This approach operates within a compact intermediate space to represent the accumulated information from local LoRAs. We introduce tunable singular value thresholding for server-side optimal rank selection to construct a pair of global low-rank adapters shared by all clients. Extensive empirical evaluations across multiple datasets and LLMs demonstrate that `FLoRIST` consistently strikes the best balance achieving superior download communication efficiency while maintaining competitively better performance than baselines in both homogeneous and heterogeneous setups.

1 Introduction

Large Language Models (LLMs) have emerged as powerful general-purpose learners, enabling impressive progress in dialogue systems (Bill & Eriksson, 2023; Dong et al., 2023), information retrieval (Kelly et al., 2023), healthcare (Thirunavukarasu et al., 2023), and scientific research (AI4Science & Quantum, 2023). However, adapting these models to specific downstream tasks (Howard & Ruder, 2018) remains resource-intensive, often requiring fine-tuning hundreds of millions of parameters. Parameter-Efficient Fine-Tuning (PEFT) methods such as Low-Rank Adaptation (LoRA) (Hu et al., 2022) alleviate this by inserting lightweight, trainable low-rank matrices into LLM layers, dramatically reducing memory and compute costs during adaptation. In privacy-sensitive settings where the data needed for fine-tuning LLMs reside in a distributed network of edge devices or institutions, Federated Learning (FL) (McMahan et al., 2017) offers a promising paradigm by allowing collaborative model fine-tuning without sharing local data. Integrating LoRA into FL enables clients to train only low-rank adapters locally and transmit compact updates to a central server rather than original weight updates in full fine-tuning, reducing communication overhead while preserving privacy. But this brings us to a deeper question:

> *What is the intrinsic dimensionality of these aggregated local adapters derived from heterogeneous LoRAs? Is it essential to preserve every component to maintain model performance? Could we further enhance communication-efficiency by identifying and eliminating hidden redundancies resulting in unified global LoRA?*

Most existing methods fail to comprehensively answer these questions. Prior works either enforce fixed homogeneous ranks across all layers and clients (e.g., FedIT (Zhang et al., 2024a), FFA-LoRA (Sun et al., 2024)) or handle heterogeneity by stacking and communicating dense full-rank adapters (e.g., FLoRA (Wang et al., 2024)), leading to significant communication or computational burdens. Even more recent methods like FlexLoRA (Bai et al., 2024) perform expensive singular value decomposition (SVD) computation on the full weight-update matrix, and later construct several global adapters to match the heterogeneous ranks

[1]Department of Electrical and Computer Engineering, University of Arizona, Tucson, AZ, USA. Correspondence to: Jyotikrishna Dass <jdass@arizona.edu>.

Proceedings of the 9^{th} MLSys Conference, Bellevue, WA, USA, 2026.

to client capacity rather than the intrinsic dimensionality of the global update, leading to increased communication overhead. Table 1 summarizes the trade-offs, and Figures 1-4 in Appendix A illustrate the key ideas and gaps in the existing federated LoRA fine-tuning methods.

To address the above limitations and comprehensively answer the above questions, we propose FLoRIST, a novel framework for Federated Low-Rank Integration with Singular value Thresholding. Instead of building and decomposing the full-weight update into multiple global adapters, FLoRIST performs aggregation directly in the low-rank latent space by operating on the stacked client adapters. It then applies an energy-based threshold to retain only the most significant singular values, producing compact pair of global low-rank adapters shared by all clients, that match or exceed the performance of larger baselines. Our layer-wise rank analysis further reveals that different layers, and even different attention projections (e.g., q_proj vs. v_proj), have varying intrinsic dimensionalities, many of which are significantly lower than commonly assumed. Main contributions are as follows:

1. We propose FLoRIST, a federated fine-tuning framework for accurate and compact aggregation in the low-rank latent space, supporting heterogeneous client ranks with higher download communication efficiency.
2. We introduce a computationally fast SVD-based aggregation that avoids constructing the full-weight update. By employing singular value thresholding, it optimally selects the unified global adapter rank to balance performance and download communication efficiency.
3. We provide empirical evidence, including a fine-grained layer-wise analysis, that demonstrates the low intrinsic dimensionality of the aggregated local adapters, revealing that some layers require ranks as low as 6–10, even when clients use ranks up to 64, thereby motivating low-rank unified global adapters.
4. We compare and contrast various federated LoRA fine-tuning methods in literature where we empirically demonstrate that FLoRIST achieves higher download communication efficiency and comparable to superior performance than state-of-the-art methods such as FedIT, FFA-LoRA, FLoRA, and FlexLoRA across multiple datasets and LLM architectures.

2 Related Work

Finetuning of LLMs. LLMs have demonstrated remarkable capabilities across various natural language processing tasks. However, fine-tuning these models for specific applications can be computationally intensive due to their vast number of parameters. LoRA (Hu et al., 2022) is a parameter-efficient fine-tuning method that significantly reduces memory and compute costs. LoRA introduces trainable low-rank matrices into each layer of the pre-trained model. Specifically, a model update matrix $\Delta W \in \mathbb{R}^{m\times n}$ is decomposed into two low-rank adapters $A \in \mathbb{R}^{r\times n}$ and $B \in \mathbb{R}^{m\times r}$, where $r \ll \min(m, n)$. The updated model is expressed as $W' = W + \Delta W = W + BA$, where W remains frozen, and only A and B are updated during fine-tuning. This reduces the number of trainable parameters dramatically. For instance, a LlaMA-3.2-1B attention model $W \in \mathbb{R}^{8192\times 8192}$ on fine-tuning with LoRA, $r = 16$, results in much smaller adapters, $A \in \mathbb{R}^{16\times 8192}$ and $B \in \mathbb{R}^{8192\times 16}$.

Federated fine-tuning methods for LLMs. FL (McMahan et al., 2017) enables distributed model training across multiple clients while preserving privacy by not sharing local data. In classical FL, local model updates are aggregated at the server using Federated Averaging (FedAvg) (Sun et al., 2022), where the global update is: $\Delta W = \sum_{k=1}^{K} \frac{n_k}{N} \Delta W_k$, where n_k is the number of local samples at client k, and $N = \sum_k n_k$. We discuss recent works integrating FL and LoRA for federated fine-tuning of LLMs below and provide visual workflow in Appendix A.

FedIT (Zhang et al., 2024a) incorporates LoRA into FL by allowing each client to fine-tune low-rank adapters locally and transmit them back to the server. The server aggregates the adapters separately using FedAvg: $A_{FedIT} = \sum_{k=1}^{K} \frac{n_k}{N} A_k$, $B_{FedIT} = \sum_{k=1}^{K} \frac{n_k}{N} B_k$. **Challenges.** However, this independent averaging leads to a *mathematically inaccurate global update* by introducing cross-term noise $B_i A_j$ for $i \neq j$ in the product of $(B_{FedIT})(A_{FedIT})$. This can affect the convergence and the model performance. Furthermore, FedIT inherently supports only *homogeneous client* ranks. Although zero-padding (HetLoRA (Cho et al., 2023)) can be used to handle heterogeneous ranks, it inflates communication and memory costs and could introduce significant performance drops, as shown in our empirical analysis.

FFA-LoRA (Sun et al., 2024) improves upon FedIT by addressing the aggregation inaccuracy with higher communication efficiency. In FFA-LoRA, each client fine-tunes only one LoRA adapter, typically B_k, while freezing the other adapter A_k to its initialization. Thus, the local model update becomes $\Delta W_k = B_k A_{\text{init}}$. Since A_{init} is shared across clients, the server aggregates only the trainable B_k matrices via FedAvg: $B_{FFA} = \sum_{k=1}^{K} \frac{n_k}{N} B_k$ and reconstructs the global update as $\Delta W = B_{FFA} A_{\text{init}}$ ensuring noise-free aggregation without cross-terms. **Challenges:** While FFA-LoRA corrects the aggregation noise and reduces communication cost by half compared to FedIT, it still *lacks support for heterogeneous client* ranks natively. In addition, since only half of the LoRA parameters are used,

Table 1. Comparison of methods across four critical metrics: heterogeneity support, performance, communication efficiency, and computational cost. Bars indicate relative magnitudes, longer bars represent higher performance and efficiency, or higher computational cost (server). The proposed FLoRIST strikes the best balance.

METHOD	HETEROGENEITY	PERFORMANCE	COMM. EFF.	COMP. COST
FEDIT	✗	LOW HIGH	LOW HIGH	LOW HIGH
FFA-LORA	✗	LOW HIGH	LOW HIGH	LOW HIGH
FLORA	✓	LOW HIGH	LOW HIGH	LOW HIGH
FLEXLORA	✓	LOW HIGH	LOW HIGH	LOW HIGH
FLORIST **(OURS)**	✓	LOW HIGH	LOW HIGH	LOW HIGH

convergence can be slower, and model expressivity may be reduced compared to methods with both LoRA adapters.

FLoRA (Wang et al., 2024) introduces a stacking-based aggregation strategy to ensure mathematically correct updates and support heterogeneous client configurations. In FLoRA, clients transmit their local adapters, and the server concatenates these adapters across clients. The global update ΔW eliminates cross-term noise while naturally accommodating clients with different ranks. FLoRA ensures mathematical correctness and reduces communication overhead by transmitting only LoRA modules instead of full model updates. **Challenges:** However, it suffers with poor scalability from transmitting stacked local LoRA modules back to all clients, where the global rank grows linearly as sum of local LoRA ranks, leading to *higher communication overhead (download) and increased memory requirements* on resource-constrained clients. Moreover, the FLoRA clients merge the downloaded adapters to the frozen model, deviating from other methods which perform standard PEFT by re-using the downloaded adapters for the next local updates.

FlexLoRA (Bai et al., 2024) addresses the scalability issue associated with transmitting stacked adapters in FLoRA by constructing a set of customized global adapters to match the rank of each client. To achieve this, FlexLoRA reconstructs the global update ΔW from local model updates ΔW_k from all clients. It decomposes the ΔW into its corresponding singular value factors to partition and redistribute those as set of customized global adapters tailored for each client. **Challenges:** However, the communication cost still grows proportionally to clients ranks which run the risk of some clients missing out on key singular values thereby degrading the model performance. Moreover, FlexLoRA incurs *significant server-side computational cost* due to the explicit construction and decomposition of the *full update matrix* $\Delta W \in \mathbb{R}^{m\times n}$, which can be *prohibitively large in memory* for LLMs. Furthermore, singular values were limited to enable partitioning of global update to serve heterogeneous client ranks and lacks in-depth analysis of the full-weight update for balancing performance and download communication efficiency.

These observations raise several key questions: *Can we avoid constructing the full weight-update (product) matrix for global aggregation by working directly in the low-rank latent adapter space? Can we identify and retain only the most informative components for improving communication efficiency and enabling faster computation? Can we verify that only a small number of components in the global aggregation are actually needed to preserve model performance?*

3 PROPOSED METHOD

We propose FLoRIST to address the above key questions. FLoRIST is a novel federated fine-tuning framework designed for parameter-efficient adaptation of LLMs using heterogeneous LoRA modules. Specifically, FLoRIST simultaneously tackles three key challenges in existing methods: (i) cross-term noise during adapter aggregation in FedIT, (ii) the computational overhead of performing Singular Value Decomposition (SVD) on dense update matrices in FlexLoRA, and (iii) poor communication efficiency in FLoRA resulting from broadcasting stacked local LoRAs. Our method achieves noise-free global aggregation, introduces a computationally efficient SVD strategy that avoids forming the full global update matrix altogether, and employs singular value thresholding for optimal rank selection to drastically improve communication efficiency without sacrificing performance. We present the workflow in Figure 1 and corresponding pseudocode in Algorithm 1.

Noise-free aggregation via weighted stacking. Each client k fine-tunes local LoRA adapters B_k, A_k with a client-specific rank r_k, producing $B_k \in \mathbb{R}^{m\times r_k}$ and $A_k \in \mathbb{R}^{r_k\times n}$. These are sent to the server along with weighting factor n_k/N, where n_k is the client's local dataset size. The server then stacks: $B_{\text{stack}} = B_1 \oplus \cdots \oplus B_K \in \mathbb{R}^{m\times r}$ and $A_{\text{stack}} = \frac{n_1}{N}A_1 \oplus \cdots \oplus \frac{n_K}{N}A_K \in \mathbb{R}^{r\times n}$, where $r = \sum_{k=1}^{K} r_k$ and $\oplus$ denotes horizontal stacking for B_k and vertical stacking for A_k. Rather than computing $\Delta W = \sum_{k=1}^{K} \frac{n_k}{N}\Delta W_k \in \mathbb{R}^{m\times n}$ as in FlexLoRA, we leverage the equivalence $\Delta W = B_{\text{stack}}A_{\text{stack}}$, where stacking includes the weighting.

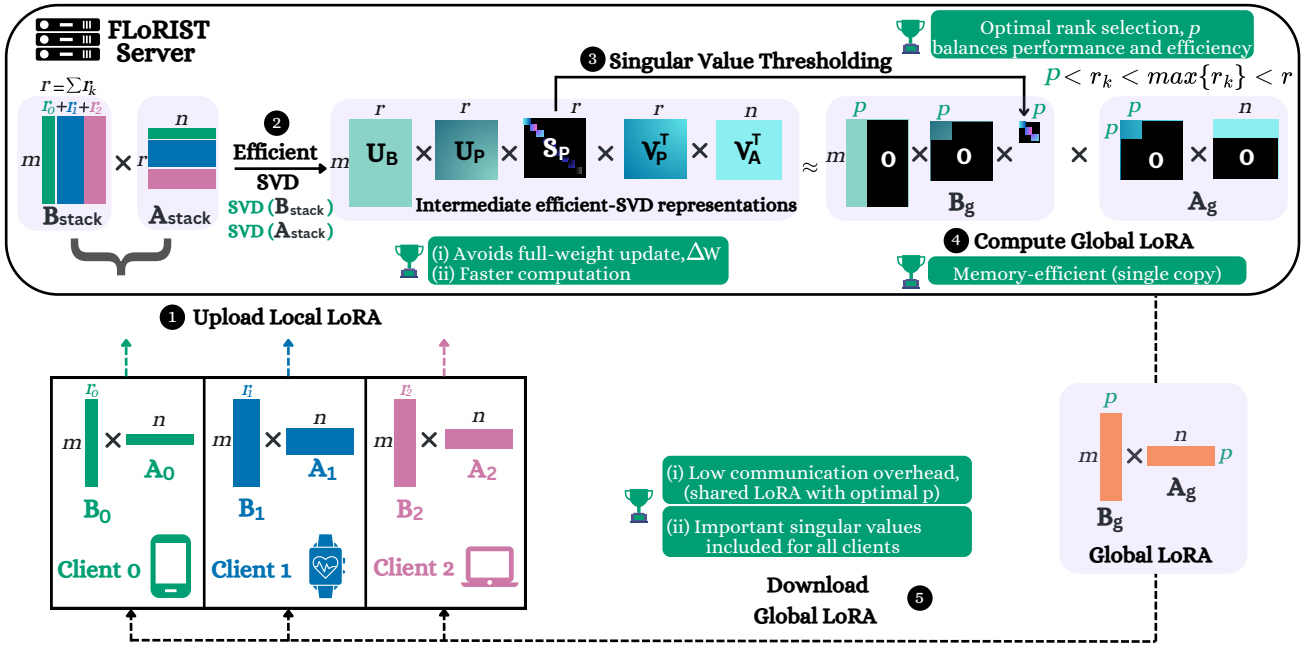


Figure 1. Workflow for the proposed FLoRIST: **(1)** Each client computes its local LoRA adapters, which are then uploaded onto the server. In contrast to FedAvg of local adapters in FedIT and constructing local full-weight updates in FlexLoRA, FLoRIST adopts stacking-based aggregation similar to FLoRA to maintain mathematical correctness. **(2)** Then, FLoRIST performs efficient SVD on stacked adapters independently to generate intermediate efficient-SVD representations. **(3)** Next, we use Singular Value Thresholding to determine the optimal rank (p) corresponding to the most informative components in the aggregated local adapters, where, $p < r_k \leq \max\{r_k\} < \Sigma r_k$, i.e. Rank (FLoRIST $<$ FlexLoRA $\leq$ FedIT $<$ FLoRA). **(4)** Using optimal rank, FLoRIST constructs a unified global low-rank adapters. **(5)** Finally, the server broadcasts the global LoRA adapters which are downloaded by all the clients for local fine-tuning.

Unlike, FlexLoRA which performs SVD on the full dense matrix ΔW, FLoRIST applies SVD (without truncation based on singular values) to B_{stack} and A_{stack} matrices independently, avoiding prohibitive construction and memory ($m \times n$) required for ΔW:

$$B_{\text{stack}} = U_B S_B V_B^T \ , \ A_{\text{stack}} = U_A S_A V_A^T \tag{1}$$

This results in reformulating the global update as $\Delta W = U_B S_B V_B^T U_A S_A V_A^T$ ensuring the proposed FLoRIST mitigates cross-term noise during adapter aggregation in FedIT.

Efficient SVD via intermediate matrix decomposition. Rather than directly multiplying the sequence of decomposed matrices above directly, FLoRIST computes an intermediate product P towards efficient SVD for the prohibitive global update without actually constructing it.

$$P = S_B Q S_A \in \mathbb{R}^{r \times r}, \tag{2}$$

where, $Q = V_B^T U_A \in \mathbb{R}^{r \times r}$ is an orthogonal matrix and recall, $r = \sum_{k=1}^{K} r_k$. The matrix P *captures the cross-adapter interaction across the local LoRA updates while maintaining low dimensionality in* $r < \{m, n\}$. Since S_B and S_A are diagonal and Q is orthogonal, the resulting matrix P *preserves spectral information from both local adapter sets*.

Using the above intermediate matrix P, we construct the global adapters as

$$B_g = U_B U_P S_P \ , \ A_g = V_P^T V_A^T, \tag{3}$$

where, $P = U_P S_P V_P^T$ is created from SVD. This gives the global weight update:

$$\Delta W \approx B_g A_g = (U_B U_P S_P)(V_P^T V_A^T). \tag{4}$$

Here, S_P *is the diagonal matrix of singular values of the global update* ΔW *without explicitly forming* ΔW. Thus, the final representation (B_g, A_g) corresponds to the SVD of the true aggregated update ΔW, computed in a memory- and time-efficient manner compared to direct SVD on ΔW in FlexLoRA.

Singular value thresholding for optimal rank selection. To justify the need for adaptive rank selection, we begin by analyzing the singular value spectrum of the aggregated update matrix ΔW. Figure 2 presents a heatmap of singular values across all q_proj layers of TinyLlaMA fine-tuned on the Wizard dataset in a heterogeneous setting. Despite

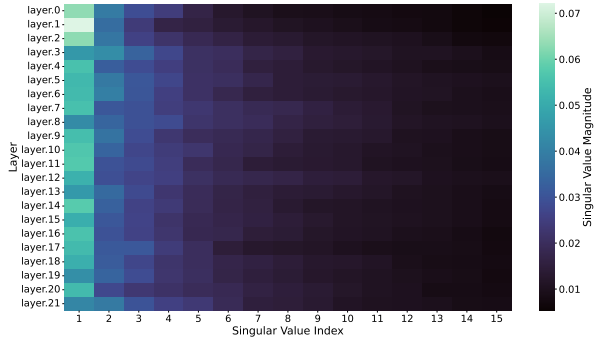


Figure 2. Singular value spectrum of the `q_proj` layers in TinyLLaMA fine-tuned with heterogeneous LoRA ranks on the Wizard dataset after 1 round. We observe that most singular values drop off sharply and become negligible within the first 6 to 10 components across layers, indicating that the effective rank required to reconstruct ΔW is far lower than the maximum client rank (64) used in FlexLoRA.

the maximum client rank being 64, we observe that in most layers, the singular values decay rapidly, often becoming negligible within the first 6 to 10 components. This indicates that the effective dimensionality of ΔW is substantially lower than the total transmitted rank. However, existing methods such as FLoRA and FlexLoRA overlook this redundancy and transmit stacked local adapters and partition full-SVD components to match specific client ranks, respectively, incurring excessive communication overhead and missing out on important singular values (resource-constrained clients) or transmitting redundant components than required (resource-rich clients). Motivated by this observation, `FLoRIST` introduces an energy-based truncation criterion that retains only the top-p singular components corresponding to the original ΔW without reconstructing it. Specifically, we apply thresholding on S_P, using a tunable hyperparameter $\tau \in (0, 1]$, and retain the smallest p resulting in singular values $(S_P)_{ii} = \sigma_i\ ,\ i \in \{1, \dots, p\}$.

$$\frac{\sum_{i=1}^{p}(S_P)_{ii}^2}{\sum_{i=1}^{\min(m,n)}(S_P)_{ii}^2} \geq \tau$$

The global adapters are then constructed as: $B_g = (U_B U_P)[:, :p](S_P)[:p, :p]$ and $A_g = (V_P^T V_A^T)[:p, :]$. These global adapters are broadcasted to all clients, who update their local models as $W' = W + B_g A_g$. Since the thresholded rank p is typically much smaller than $\max\{r_k\}$, `FLoRIST` achieves superior communication efficiency while maintaining competitive accuracy. Our experiments (Section 4) validate that `FLoRIST` outperforms all baselines in communication efficiency and matches or exceeds them in accuracy. Notably, $p < r_k \leq \max\{r_k\} < \sum_{k=1}^{K} r_k$, implying:

Rank: `FLoRIST` $<$ FlexLoRA $\leq$ FedIT $<$ FLoRA

By avoiding explicit construction of ΔW while still computing its singular values, S_P, `FLoRIST` provides a mathematically accurate, highly efficient federated model aggregation, supporting to heterogeneous client ranks, and scalable to large model sizes.

Theoretical analysis. We now formalize the approximation guarantees of `FLoRIST` using the Eckart–Young–Mirsky theorem (Golub et al., 1987). Let $M \in \mathbb{R}^{m \times n}$ with singular values $\sigma_1 \geq \sigma_2 \geq \cdots \geq \sigma_{r^*} > 0$, where $r^* = \text{rank}(M)$. The theorem states that the best rank-r approximation of M under the Frobenius norm is obtained by its top-r singular components:

$$\min_{\text{rank}(\hat{M}) \leq r} \|M - \hat{M}\|_F = \left(\sum_{i=r+1}^{r^*} \sigma_i^2 \right)^{1/2}.$$

In `FLoRIST`, we consider $M = \Delta W = \sum_k A_k B_k$, and the `FLoRIST` output is a rank-p factorization $B_g A_g$. By the theorem, the approximation error is bounded by the tail energy of the discarded singular values:

$$\|\Delta W - B_g A_g\|_F \leq \left(\sum_{i=p+1}^{r^*} \sigma_i^2 \right)^{1/2}. \tag{5}$$

Energy-based Rank Selection. To ensure that at least a τ-fraction of the total variance is preserved, we select the smallest rank p such that

$$\frac{\sum_{i=1}^{p} \sigma_i^2}{\sum_{i=1}^{r^*} \sigma_i^2} \geq \tau. \tag{6}$$

This criterion is consistent with standard practices in PCA and SVD-based compression, while providing a principled guarantee on the retained information.

Complexity analysis. Let m and n denote the embedding and context dimensions respectively, $\mathcal{T}(m, n, r_k, |D_k|)$ the per-epoch training cost, $|D_k|$ the number of local training samples, r_k the LoRA rank used by client k, p_l the rank retained after thresholding at layer l, $r = \sum_k r_k$, and L the total number of attention layers. The client-side computational cost of `FLoRIST` is $\mathcal{O}(E \cdot \mathcal{T}(m, n, r_k, |D_k|)) + \mathcal{O}(\sum_{l=1}^{L} m p_l n)$, and the server-side complexity for aggregation and SVD-based decomposition is $\mathcal{O}(Lr^2(m + n + r)) + \mathcal{O}(\sum_{l=1}^{L} p_l^2(m + n))$, which is significantly lower than FlexLoRA's $\mathcal{O}(LKmn) + \mathcal{O}(L \min(m, n)mn) + \mathcal{O}(L(mp^2 + p^2 n))$ server cost that arises from full-matrix SVD. We note that `FLoRIST`'s server-side advantage holds when $r = \sum_k r_k << \min(m, n)$, which is generally satisfied in practical federated setups where only a fraction of clients are sampled per round and each client performs low-rank adaptation. While the server workload grows with

Algorithm 1 FLoRIST: Federated Low-Rank Integration with Singular Value Thresholding

Input: Pretrained model weights W_0, number of rounds T, clients $\mathcal{C}$ with LoRA ranks $\{r_k\}_{k\in\mathcal{C}}$, dataset sizes $\{n_k\}$, threshold τ
Output: Global LoRA adapters (B_g, A_g)
Initialize global LoRA adapters (B_g, A_g) **for** $t = 1$ **to** T **do**
  /* Server selects clients and broadcasts global adapters */
  Server: Sample clients $\mathcal{C}^t \subset \mathcal{C}$ Broadcast(B_g, A_g) to all $k \in \mathcal{C}^t$ /* Clients perform local fine-tuning */
  foreach $k \in \mathcal{C}^t$ **do in parallel**
    Client k**:**
    if $t == 1$ **then**
      Initialize $B_k \leftarrow 0$, $A_k \sim \mathcal{N}(0, \sigma^2)$ (random init)
    else
      /* Match global rank p to local rank r_k */
      Let $p \leftarrow \text{rank}(B_g, A_g)$ **if** $p < r_k$ **then**
        Zero-pad: $B_k \leftarrow [B_g \mid \mathbf{0}_{m\times(r_k-p)}]$, $A_k \leftarrow \begin{bmatrix} A_g \\ \mathbf{0}_{(r_k-p)\times n} \end{bmatrix}$
      else if $p > r_k$ **then**
        Truncate: $B_k \leftarrow (B_g)_{[:,:r_k]}$, $A_k \leftarrow (A_g)_{[:r_k,:]}$
      else
        $(B_k, A_k) \leftarrow (B_g, A_g)$
    $(B_k, A_k) \leftarrow \text{LocalUpdate}(W_0, B_k, A_k)$
    Upload(B_k, A_k) to server
  /* Server aggregates without forming ΔW */
  Server:
  Stack all B_k horizontally and weighted A_k vertically
  $B_{\text{stack}} \leftarrow B_1 \oplus \cdots \oplus B_K$
  $A_{\text{stack}} \leftarrow \frac{n_1}{N} A_1 \oplus \cdots \oplus \frac{n_K}{N} A_K$ Perform SVD: $B_{\text{stack}} = U_B S_B V_B^T$, $A_{\text{stack}} = U_A S_A V_A^T$
  Compute: $Q \leftarrow V_B^T U_A$, $P \leftarrow S_B Q S_A$
  Perform SVD: $P = U_P S_P V_P^T$ /* Energy-based thresholding */
  Find smallest p such that $\frac{\sum_{i=1}^{p}(S_P)_{ii}^2}{\sum_{i=1}^{\min(m,n)}(S_P)_{ii}^2} \geq \tau$
  Truncate: $B_g \leftarrow (U_B U_P)_{[:,:p]}(S_P)_{[:p,:p]}$, $A_g \leftarrow (V_P^T V_A^T)_{[:p,:]}$
return (B_g, A_g)

the number of participating clients K, this cost is borne by the server, which is assumed to be resource-rich in cross-device FL, rather than by the clients. In contrast, methods such as FLoRA shift the aggregation burden to clients by requiring them to process large stacked adapters, which is undesirable under heterogeneous client resources. We report the raw server computational cost (in FLOPs) in Table 4. A detailed analysis and comparison with other methods across computation, communication, and memory is provided in Appendix B.

4 Experiments

4.1 Experimental Setup

Datasets and configurations. We evaluate FLoRIST on federated fine-tuning of LLaMA-based models (TinyLlaMA (Zhang et al., 2024b), and LlaMA-3.2-1B (lla) using three instruction-tuning datasets: Dolly (Zhang et al., 2024a), Alpaca (Dubois et al., 2023), and Wizard (Luo et al., 2025). LoRA is applied only to self-attention layers following (Hu et al., 2022). We report performance on a 1,444-sample subset of MMLU (Hendrycks et al., 2021). Our federated setup consists of 100 clients, with 10 clients randomly sampled in each communication round. Consistent with prior works (Zhang et al., 2024a; He et al., 2020; Lai et al., 2022), we use Dirichlet distribution-based non-IID partitions of the dataset, with a concentration parameter of $\alpha = 0.5$, to create local data for clients. All experiments are run for a total of 75 communication rounds to ensure convergence. In the homogeneous configuration, all clients use LoRA rank 16. In the heterogeneous configuration, we adopt an extreme heavy-tail-light distribution of client ranks with $16\times$ rank disparity ($4 \rightarrow 64$) to rigorously evaluate the model under a regime of high heterogeneity: 40 clients use rank 4, 20 clients use rank 8, 20 clients use rank 16, 10 clients use rank 32, and 10 clients use rank 64. This distribution reflects realistic variations in client capacity, where the majority of participants operate at lower ranks and a small fraction contributes higher-rank updates. Each communication round is followed by local fine-tuning with a learning rate of 0.0003. Training is conducted on a large-scale cluster equipped with NVIDIA H100 GPUs under both homogeneous and heterogeneous settings.

Baselines. We compare our proposed FLoRIST method against the following related works: FedIT (Zhang et al., 2024a) integrates LoRA with FedAvg and only supports homogeneous LoRA ranks across clients. It relies on zero-padding (HetLoRA (Cho et al., 2023)) to handle heterogeneity upto maximum client rank. Zero-padding is used by both FedIT and FFA-LoRA to accommodate rank differences. FLoRA (Wang et al., 2024) is a stacking-based aggregation strategy with heterogeneous LoRA spport. FlexLoRA (Bai et al., 2024) redistributes SVD of full-weight update to create multiple global adapters to match the client's local rank. FFA-LoRA (Sun et al., 2024) freezes one of the LoRA

adapters during training and only transmits the other half of the adapters. We refer to Appendix E for detailed descriptions of datasets, and baseline methods used for our experimental results.

Threshold variants. We report FLoRIST under two threshold configurations. **FLoRIST [τ^*]** uses an optimally tuned threshold selected via binary search over $[0.80, 0.99]$, choosing the smallest τ that achieves performance equal to or better than all baselines for each model–dataset–client combination; this variant serves as a diagnostic upper bound on the accuracy–efficiency trade-off. **FLoRIST [τ=0.9]** uses a single fixed threshold across all settings, as a practical deployment choice. Both variants are reported in Table 2.

Communication cost and efficiency definitions. Throughout this section, *communication cost* refers exclusively to the *download* cost, i.e., the total number of parameters transmitted from the server to the selected clients per round. Correspondingly, *communication efficiency* is defined as the inverse of this download cost: $\frac{1}{\text{Total Parameters Downloaded}}$. Since all compared methods communicate LoRA matrices whose size scales linearly with rank, we approximate efficiency by $\frac{1}{\text{Total Download Rank}}$, providing a consistent and interpretable proxy across methods.

Code and Reproducibility. Our implementation is publicly available at https://github.com/DASS-Lab-Group/FLoRIST. Full details on how to access, install, and run all experiments, including hardware and software requirements, expected outputs, and instructions for reproducing the FLoRIST [τ=0.9] rows in Table 2, are provided in Appendix F.

4.2 Performance Analysis

Convergence Analysis. We compare the convergence behavior of various federated fine-tuning methods on TinyLlama using the Alpaca dataset in a homogeneous client setting. Figure 3 plots MMLU accuracy over 70 communication rounds, with solid lines representing raw performance and dashed lines showing fitted trends. FLoRIST demonstrates both faster convergence and superior final accuracy compared to all baselines. This validates its ability to balance expressivity and communication efficiency through low-rank aggregation. Notably, FFA-LoRA converges more slowly due to freezing half of its parameters during training, but still achieves the second-best final accuracy, highlighting the benefits of partial update regularization. FedIT performs worst, as expected, due to its mathematically inaccurate aggregation of client updates. Although FLoRA uses mathematically correct averaging, it merges global adapters after every round, deviating from the PEFT paradigm. This

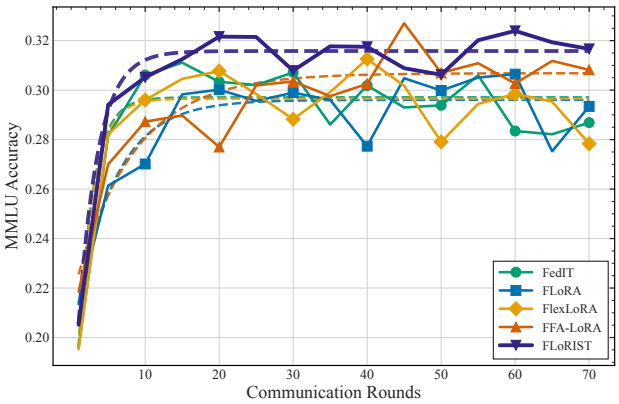


Figure 3. MMLU accuracy over communication rounds for TinyLlama on the Alpaca dataset (homogeneous setting, 10 clients). Solid lines denote raw accuracy; dashed lines show fitted trends. FLoRIST converges faster and achieves the highest final accuracy. FFA-LoRA, despite slower convergence due to partial parameter freezing, achieves the second-best final accuracy. FedIT performs worst due to mathematically inaccurate aggregation. FLoRA, while mathematically accurate, suffers from lower accuracy due to reinitializing local adapters each round.

results in lower accuracy, since local LoRA adapters are reinitialized each round rather than being fine-tuned from the global adapter, a strategy preserved in other baselines. Additional convergence plots are shown in Appendix D.

Homogeneous setup. In the homogeneous setting, where all clients utilize LoRA adapters of the same rank, FLoRIST consistently achieves the best overall trade-off between accuracy and communication efficiency across nearly all model–dataset combinations (Table 2). For example, with TinyLlama on the Wizard dataset, FLoRIST [τ^*] attains the highest accuracy of 40.95% while being nearly 30$\times$ more communication-efficient than FLoRA (54.38 vs. 1.78). Similarly, on Alpaca with TinyLlama, FLoRIST [τ^*] reaches 32.26% accuracy exceeding FedIT (28.35%), FLoRA (28.99%), FlexLoRA (29.22%) and FFA-LoRA (31.58%), with an efficiency of 35.08. With Llama-3.2-1B, FLoRIST [τ^*] again provides strong results: on Alpaca it achieves the top accuracy of 30.46%, and on Dolly it delivers 28.28% accuracy, far surpassing all baselines.

The only exception occurs on the Wizard dataset with Llama-3.2-1B, where FedIT slightly edges out FLoRIST [τ^*] in accuracy (30.48% vs. 29.31%). However, FLoRIST [τ^*] is still more than 2.6$\times$ as communication-efficient in this case (51.19 vs. 19.50). Overall, across all homogeneous configurations, FLoRIST [τ^*] either achieves the highest accuracy or remains highly competitive while consistently offering substantial communication savings, underscoring its robustness and efficiency advantages.

Table 2. MMLU performance across models, client configurations (homogeneous or heterogeneous rank), and federated fine-tuning methods on three datasets. Acc. denotes Accuracy (%), and Eff. denotes Communication Efficiency, defined as $\frac{1}{\text{Total Parameters Downloaded}}$. Accuracy values reflect the converged accuracy after 75 communication rounds. Highest and second-highest values in a column, within a particular client configuration, are represented in **bold** and underline, respectively. FLoRIST [τ^*] uses a optimally tuned threshold as a diagnostic analysis; FLoRIST [τ=0.9] uses a fixed threshold, as a practical deployment choice, across all configurations.

Model	Client	Method	Dolly		Alpaca		Wizard	
			Acc. (%)	Eff. ($\times 10^{-4}$)	Acc. (%)	Eff. ($\times 10^{-4}$)	Acc. (%)	Eff. ($\times 10^{-4}$)
TinyLlama	Homo	FedIT	27.46	14.20	28.35	14.20	36.61	14.20
		FLoRA	28.99	1.78	28.99	1.78	34.20	1.78
		FlexLoRA	28.06	14.20	29.22	14.20	39.75	14.20
		FFA-LoRA	27.79	28.40	31.58	28.40	36.01	28.40
		FLoRIST [τ^*]	29.16	**53.54**	**32.26**	35.08	**40.95**	54.38
		FLoRIST [τ=0.9]	**30.94**	23.48	31.68	**60.06**	38.92	**63.09**
	Heter	FedIT (Zero-pad)	25.73	3.55	31.04	3.55	**44.19**	3.55
		FLoRA	27.20	0.50	28.58	0.50	33.74	0.50
		FlexLoRA	28.49	11.96	29.61	11.96	36.39	11.96
		FFA-LoRA	18.54	7.10	23.19	7.10	23.75	7.10
		FLoRIST [τ^*]	**28.87**	**37.87**	**31.33**	**45.09**	38.20	13.60
		FLoRIST [τ=0.9]	27.69	34.93	29.69	34.90	41.51	**36.16**
Llama-3.2-1B	Homo	FedIT	25.00	19.50	29.44	19.50	**30.48**	19.50
		FLoRA	22.48	2.44	29.16	2.44	28.57	2.44
		FlexLoRA	27.39	19.50	29.24	19.50	30.03	19.50
		FFA-LoRA	26.15	39.06	29.18	**39.06**	28.40	39.06
		FLoRIST [τ^*]	28.28	**51.71**	**30.46**	33.30	29.31	**51.19**
		FLoRIST [τ=0.9]	**29.48**	33.64	29.73	33.42	29.62	**51.19**
	Heter	FedIT (Zero-pad)	21.41	4.88	28.37	4.88	28.42	4.88
		FLoRA	23.85	2.06	30.15	2.06	27.73	2.06
		FlexLoRA	**26.74**	16.44	29.95	16.44	29.13	16.44
		FFA-LoRA	22.45	9.77	22.68	9.77	28.78	9.77
		FLoRIST [τ^*]	24.01	46.35	**30.53**	45.22	**29.79**	47.82
		FLoRIST [τ=0.9]	24.10	**49.38**	30.24	**48.24**	29.45	**50.22**

Practical threshold (τ=0.9) in the homogeneous setup. The fixed-threshold variant FLoRIST [τ=0.9] remains highly competitive without any per-task tuning. On the Dolly dataset, it achieves the best accuracy across *all* methods, including the optimally tuned FLoRIST [τ^*], for both TinyLlama (30.94% vs. 29.16%) and Llama-3.2-1B (29.48% vs. 28.28%). Across all homogeneous settings the accuracy gap between τ=0.9 and τ^* choices is within $\pm$ 1% on average, and in several cases the practical threshold setup achieves markedly higher efficiency (e.g., TinyLlama–Alpaca: 60.06 vs. 35.08; TinyLlama–Wizard: 63.09 vs. 54.38). These results demonstrate that τ=0.9 is a practical, deployment-ready default, delivering strong performance out of the box without per-task threshold search.

Heterogeneous setup. In the heterogeneous setting, where clients employ LoRA adapters of varying ranks, FLoRIST emerges as the most communication-efficient method across all model–dataset combinations (Table 2). Beyond efficiency, it also achieves the best accuracy in the majority of cases. For instance, with TinyLlama on Dolly and Alpaca, FLoRIST [τ^*] delivers both the highest accuracy (28.87% and 31.33%, respectively) and the strongest efficiency (37.87 and 45.09). Similarly, with Llama-3.2-1B on Alpaca and Wizard, FLoRIST [τ^*] again leads in accuracy (30.53% and 29.79%) while being more than 2.7$\times$ as efficient as the next best baseline.

The only notable exception occurs on Llama-3.2-1B with Dolly, where FlexLoRA achieves a slightly higher accuracy (26.74% vs. 24.01%). However, FLoRIST [τ^*] is over 2.8$\times$ more communication-efficient in this case (46.35 vs. 16.44), underscoring its superior trade-off. Another case is TinyLlama on Wizard, where FedIT (zero-padding) attains marginally higher accuracy (44.19% vs. 38.20), but this comes at the cost of instability: FedIT and FFA-LoRA both collapse on Dolly (25.73% and 18.54%, respectively), highlighting the inconsistency of zero-padding approaches. These fluctuations are consistent with prior observations in the FLoRA paper, where FedIT (zero-padding) struggled to generalize in heterogeneous environments.

Practical threshold (τ=0.9) in the heterogeneous setup. The fixed-threshold variant FLoRIST [τ=0.9] shows notable strengths under the 16$\times$ rank-disparity regime as well. On TinyLlama–Wizard it achieves the second-best accuracy (41.51%) while delivering the highest efficiency (36.16), substantially outperforming τ^* in both metrics for this set-

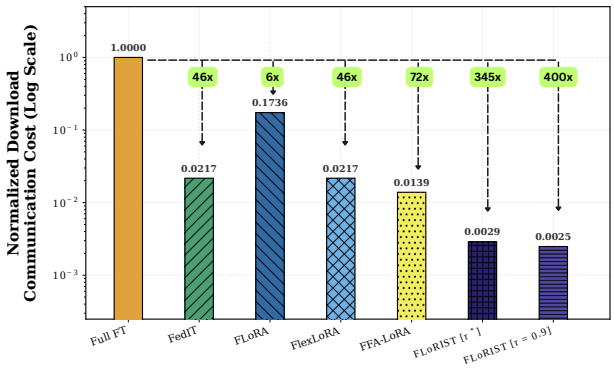


Figure 4. Normalized per-round download communication cost (log scale) for various federated fine-tuning methods on TinyLlama (Wizard dataset, homogeneous setting). All methods significantly reduce communication cost compared to Full Fine-Tuning. Proposed FLoRIST [τ=0.9] achieves the lowest cost, with a 400× reduction relative to Full FT.

ting (38.20% at 13.60). On Llama-3.2-1B, τ=0.9 consistently achieves the highest efficiency across all three datasets (49.38, 48.24, and 50.22), while its accuracy remains within $\pm$ 1% of τ^* in all cases, confirming that τ primarily controls the accuracy–efficiency trade-off. Combined with the homogeneous results, these findings establish τ=0.9 as a practical default across diverse client configurations, while automating threshold selection remains a promising direction for future work.

Overall, FLoRIST demonstrates remarkable stability, consistently achieving the best communication efficiency and competitive or better accuracy across nearly all settings. This robustness makes it a more reliable choice for heterogeneous federated fine-tuning compared to baselines that suffer from accuracy drops or poor scalability.

4.3 Communication Cost and Efficiency

As defined in Section 4, communication cost refers to the download cost, the total number of parameters transmitted from the server to all selected clients per round, corresponding to the size of LoRA adapters or full model weights downloaded by the clients.

Figure 4 presents the normalized per-round download communication cost for different federated fine-tuning methods on the TinyLlama model using the Wizard dataset in a homogeneous client setting with 10 clients. All methods demonstrate substantial reductions in communication cost compared to Full Fine-Tuning (Full FT). Notably, FLoRA, which requires downloading stacked LoRA adapters, incurs higher cost than other adapter-based methods. In contrast, the proposed FLoRIST achieves the lowest download communication cost by transmitting a unified global adapter with minimal rank. Specifically, FLoRIST [τ^*] achieves

Table 3. Communication cost (MB) of federated fine-tuning methods on TinyLlama using the Wizard dataset (homogeneous setting).

Method	Comm. Cost (MB) Upload	Download
Full FT	2076.17	2076.17
FedIT	45.05	45.05
FLoRA	45.05	360.45
FlexLoRA	45.05	45.05
FFA-LoRA	**22.52**	28.83
FLoRIST [τ^*]	45.05	**5.95**
FLoRIST [τ=0.9]	45.05	**5.15**

Table 4. Server computational cost (FLOPs) on TinyLlama using the Wizard dataset (homogeneous setting).

Method	Server FLOPs
FedIT	4.76M
FFA-LoRA	0.52M
FLoRA	0B
FlexLoRA	3516.01M
FLoRIST (Ours)	466.95M

around 5× reduction compared to FFA-LoRA, 60× compared to FLoRA, and a 345× reduction relative to Full FT. With a fixed threshold of τ=0.9, FLoRIST further reduces cost to 70× compared to FLoRA and a remarkable 400× reduction relative to Full FT.

From Table 2, FLoRIST [τ^*] is the most communication-efficient method across all datasets, models, and client configurations, while consistently maintaining competitive or better accuracy. For instance, on the TinyLlama–Dolly–homo combination, FLoRIST [τ^*] achieves an efficiency of 53.54, which is 1.9× higher than FFA-LoRA (28.40), 3.8× higher than both FedIT and FlexLoRA (14.20), and nearly 30× higher than FLoRA (1.78), all while also delivering the best accuracy (29.16%). Across tasks, FLoRIST [τ^*] reaches up to 108× higher efficiency than FLoRA (the least efficient baseline, e.g., TinyLlama–Wizard–homo: 54.38 vs. 0.50) and more than 1.8× higher than FFA-LoRA (e.g., Llama-3.2-1B–Dolly–homo: 51.71 vs. 28.40). The fixed-threshold variant FLoRIST [τ=0.9] achieves even higher download efficiency in several settings, e.g., TinyLlama–Alpaca–homo: 60.06 vs. 35.08 for τ^*, and Llama-3.2-1B–Wizard–heter: 50.22 vs. 47.82, demonstrating that a single practical threshold can further improve communication savings without sacrificing accuracy. These results highlight FLoRIST's scalability and practical relevance: it consistently delivers state-of-the-art communication efficiency while preserving or exceeding the accuracy of competing methods. Table 3 further reports the raw communication cost (upload and download, in MB) for TinyLlama–Wizard–homo.

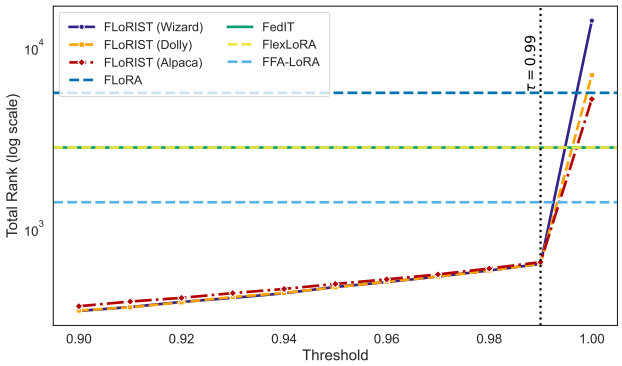


Figure 5. Total Rank (across all model layers) vs. Threshold for TinyLlama model on various datasets. Lower singular value thresholds lead to more memory-efficient global LoRA adapters and improved communication efficiency.

4.4 Server Computational Cost

We report the raw computational cost on the server where methods like FlexLoRA incur significant cost due to full-weight update matrix decomposition using full-SVD. Table 4 reports the server-side FLOPs required for each method on the TinyLlama model. FLoRA requires 0 FLOPs since it performs only concatenation and broadcasting (memory operations) at the server. We note that FlexLoRA requires over $3516.01M$ FLOPs whereas FLoRIST takes 466.9 FLOPs adopting efficient SVD scheme where SVD is applied directly on stacked LoRA adapters, making it nearly $7.5\times$ faster, while maintaining strong MMLU performance.

4.5 Impact of Thresholding

Unlike previous methods that maintain a fixed rank across all layers, FLoRIST dynamically adjusts the rank for each layer based on its unique weight distribution. Since different layers exhibit varying intrinsic dimensionalities, this adaptive approach enables more efficient parameterization compared to static-rank methods like FLoRA and FedIT. We compare *total rank* across layers to understand the trade-off between thresholding and rank compression.

Lower threshold achieves higher communication efficiency. As illustrated in Figure 5, the total rank of FLoRIST across all layers decreases as the threshold is lowered, demonstrating its ability to aggressively reduce redundancy in global weight representations, thereby boosting communication efficiency. Despite a significant reduction in rank at lower thresholds, FLoRIST maintains strong performance, as evidenced by our findings in Table 2. This highlights the effectiveness of adaptive rank decomposition in reducing communication overhead while maintaining or even surpassing the performance of state-of-the-art methods. An interesting observation from Figure 5 is that despite having double the number of LoRA adapters than FFA-LoRA

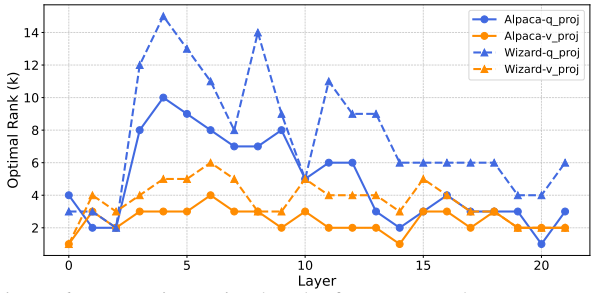


Figure 6. Layer-wise optimal rank of q_proj and v_proj matrices across attention layers.

(most efficient baseline with a frozen adapter), the proposed FLoRIST achieves superior efficiency for practical threshold values, $\tau \leq 0.99$ across the three datasets (TinyLlama-Wizard, TinyLlama-Dolly, and TinyLlama-Alpaca). Moreover, while methods like FLoRA need a higher total rank to maintain accuracy, FLoRIST balances rank reduction and performance retention through its adaptive thresholding mechanism. This validates that FLoRIST has lower communication overhead than all baselines at most practical thresholds, while still outperforming them in accuracy.

Layer-wise rank analysis reveals varying intrinsic dimensionality. While Figure 5 highlights the overall rank reduction trends with varying thresholds, a more granular analysis reveals deeper insights into the intrinsic dimensionality of different layers. To understand this, we visualize the optimal ranks of the attention projection matrices, across layers in a heterogeneous setup, using empirically chosen thresholds for FLoRIST. Specifically, we show the rank distribution of q_proj and v_proj in TinyLlama-Wizard at threshold $\tau = 0.8$ and TinyLlama-Alpaca at threshold $\tau = 0.9$ in Figure 6. Several key observations emerge: (1) The rank varies significantly across layers, indicating non-uniform intrinsic dimensionality in the model. Intermediate layers consistently require higher ranks, while both initial and final layers tend to be intrinsically low rank. This aligns with the findings of (Zhao et al., 2024) that intermediate layers carry richer representations. (2) We observe that v_proj consistently requires lower ranks compared to q_proj mostly across all layers, suggesting higher redundancy in the v_proj. These insights emphasize the utility of singular value thresholding in FLoRIST for adapting rank at a fine-grained level, leading to a more communication-efficient yet expressive global adapter.

Threshold helps regularize for improved performance. The energy threshold in FLoRIST serves as a key hyperparameter that governs the rank of the aggregated global LoRA adapter via Singular Value Thresholding (SVT). By filtering out low-energy components in the stacked client updates, SVT acts as a denoising mechanism that suppresses client-specific noise and enhances generalization. This de-

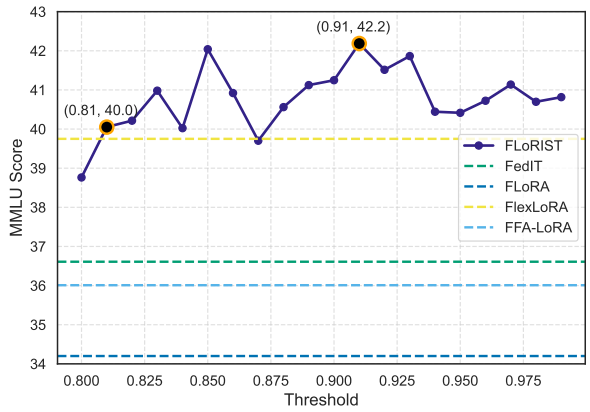


Figure 7. Energy Threshold vs. MMLU score of TinyLlama on the Wizard dataset (homogeneous rank). At $\tau = 0.91$, FLoRIST achieves the highest MMLU score. At $\tau = 0.81$, it still surpasses other methods in MMLU while being the most communication-efficient (corresponding to smallest optimal rank for global LoRA).

composition regularizes the global update, enhancing robustness to client heterogeneity and reducing overfitting. Figure 7 shows the impact of varying the threshold τ on MMLU performance for TinyLlama (Wizard dataset, homogeneous setting). As τ decreases from 1.0, performance initially improves, reaching a peak score of 42.2 at $\tau = 0.92$, before degrading as the threshold becomes too aggressive, discarding useful latent structure and causing loss of critical information. This behavior mirrors regularization techniques such as dropout (Srivastava et al., 2014) and weight pruning (Han et al., 2015), where controlled noise improves generalization, but excessive pruning reduces expressivity. Unlike prior methods, FLoRIST leverages SVT to select a reduced but informative subspace for global updates. Mathematically, SVT is the proximal operator for nuclear norm minimization, a widely used technique in matrix denoising and low-rank completion (Oh et al., 2015; Nadakuditi, 2013; Cai et al., 2010). Since singular values reflect the importance of latent directions across clients, thresholding retains only the shared signal while discarding noisy, client-specific variations. This low-rank regularization is especially beneficial in federated settings, where updates are inherently noisy due to data heterogeneity and prolonged local training. The optimal threshold varies across models and datasets, underscoring the need for tuning based on architecture and data characteristics. However, as shown in Table 2, a fixed threshold of τ=0.9 performs competitively across all 12 model–dataset–client configurations, with accuracy within $\pm 1\%$ of the optimally tuned τ^*. This confirms that the energy threshold serves not only as a regularization mechanism, but also as a robust and tunable lever to navigate the accuracy–efficiency trade-off. While a fixed τ=0.9 is a strong practical default, automating threshold selection remains a promising direction for future work.

5 Conclusion

FLoRIST is a novel approach for federated fine-tuning of LLMs that identifies and leverages the low intrinsic dimensionality of aggregated local LoRA adapters to reduce redundancy and optimize the trade-off between model performance and communication efficiency. By applying fast, independent SVD on aggregated local adapters and operating in a compact intermediate space at the server, FLoRIST avoids full-weight updates and identifies the most informative components (corresponding to optimal rank) via singular value thresholding. We present the first comprehensive evaluation of LoRA-based federated fine-tuning methods across both homogeneous and heterogeneous settings, showing FLoRIST achieves the best balance of performance and efficiency. While a fixed threshold of τ=0.9 proves to be a robust practical default (Section 4), automating threshold selection remains a promising direction. Candidate strategies include: (i) *singular value decay analysis*, using lightweight knee-point detection on the spectrum to identify the natural rank cutoff per layer; (ii) *cross-validation on held-out clients*, selecting τ that maximizes generalization across unseen client distributions; and (iii) *bilevel optimization*, jointly optimizing τ and model parameters within the federated training loop. Beyond threshold automation, leveraging the layer-wise rank analysis of the query and value projections offers a further avenue for improving the accuracy–efficiency trade-off.

Acknowledgment

This work used DeltaAI at the National Center for Supercomputing Applications (NCSA) through allocation $\#CIS250561$ from the Advanced Cyberinfrastructure Coordination Ecosystem: Services & Support (ACCESS) program (Boerner et al., 2023), which is supported by U.S. National Science Foundation grants $\#2138259$, $\#2138286$, $\#2138307$, $\#2137603$, and $\#2138296$. Research also supported by the NVIDIA Academic Grant Program using NVIDIA A100 (80 GiB) GPUs accessed via the NVIDIA Brev cloud platform.

References

Llama 3.2: Revolutionizing edge AI and vision with open, customizable models. https://ai.meta.com/blog/llama-3-2-connect-2024-vision-edge-mobile-devices/.

AI4Science, M. R. and Quantum, M. A. The impact of large language models on scientific discovery: a preliminary study using gpt-4. *arXiv preprint arXiv:2311.07361*, 2023.

Babakniya, S., Elkordy, A., Ezzeldin, Y., Liu, Q., Song,

K.-B., EL-Khamy, M., and Avestimehr, S. SLoRA: Federated parameter efficient fine-tuning of language models. In *International Workshop on Federated Learning in the Age of Foundation Models in Conjunction with NeurIPS 2023*, 2023. URL https://openreview.net/forum?id=06quMTmtRV.

Bai, J., Chen, D., Qian, B., Yao, L., and Li, Y. Federated fine-tuning of large language models under heterogeneous tasks and client resources. In *The Thirty-eighth Annual Conference on Neural Information Processing Systems*, 2024. URL https://openreview.net/forum?id=gkOzoHBXUw.

Bill, D. and Eriksson, T. Fine-tuning a llm using reinforcement learning from human feedback for a therapy chatbot application, 2023.

Boerner, T. J., Deems, S., Furlani, T. R., Knuth, S. L., and Towns, J. Access: Advancing innovation: Nsf's advanced cyberinfrastructure coordination ecosystem: Services & support. In *Practice and Experience in Advanced Research Computing 2023: Computing for the Common Good*, PEARC '23, pp. 173–176, New York, NY, USA, 2023. Association for Computing Machinery. ISBN 9781450399852. doi: 10.1145/3569951.3597559. URL https://doi.org/10.1145/3569951.3597559.

Cai, J.-F., Candès, E. J., and Shen, Z. A singular value thresholding algorithm for matrix completion. *SIAM Journal on Optimization*, 20(4):1956–1982, 2010. doi: 10.1137/080738970.

Cho, Y. J., Liu, L., Xu, Z., Fahrezi, A., Barnes, M., and Joshi, G. Heterogeneous loRA for federated fine-tuning of on-device foundation models. In *International Workshop on Federated Learning in the Age of Foundation Models in Conjunction with NeurIPS 2023*, 2023. URL https://openreview.net/forum?id=EmV9sGpZ7q.

Dong, X. L., Moon, S., Xu, Y. E., Malik, K., and Yu, Z. Towards next-generation intelligent assistants leveraging llm techniques. In *Proceedings of the 29th ACM SIGKDD Conference on Knowledge Discovery and Data Mining*, KDD '23, pp. 5792–5793, New York, NY, USA, 2023. Association for Computing Machinery. ISBN 9798400701030. doi: 10.1145/3580305.3599572. URL https://doi.org/10.1145/3580305.3599572.

Dubois, Y., Li, C. X., Taori, R., Zhang, T., Gulrajani, I., Ba, J., Guestrin, C., Liang, P. S., and Hashimoto, T. B. Alpacafarm: A simulation framework for methods that learn from human feedback. *Advances in Neural Information Processing Systems*, 36:30039–30069, 2023.

Ghiasvand, S., Yang, Y., Xue, Z., Alizadeh, M., Zhang, Z., and Pedarsani, R. Communication-efficient and tensorized federated fine-tuning of large language models, 2024. URL https://arxiv.org/abs/2410.13097.

Golub, G. H., Hoffman, A., and Stewart, G. W. A generalization of the eckart-young-mirsky matrix approximation theorem. *Linear Algebra and its applications*, 88:317–327, 1987.

Han, S., Pool, J., Tran, J., and Dally, W. Learning both weights and connections for efficient neural network. *Advances in neural information processing systems*, 28, 2015.

He, C., Li, S., So, J., Zhang, M., Wang, H., Wang, X., Vepakomma, P., Singh, A., Qiu, H., Shen, L., Zhao, P., Kang, Y., Liu, Y., Raskar, R., Yang, Q., Annavaram, M., and Avestimehr, S. Fedml: A research library and benchmark for federated machine learning. *CoRR*, abs/2007.13518, 2020. URL https://arxiv.org/abs/2007.13518.

Hendrycks, D., Burns, C., Basart, S., Zou, A., Mazeika, M., Song, D., and Steinhardt, J. Measuring massive multitask language understanding. In *International Conference on Learning Representations*, 2021. URL https://openreview.net/forum?id=d7KBjmI3GmQ.

Hou, C., Shrivastava, A., Zhan, H., Conway, R., Le, T., Sagar, A., Fanti, G., and Lazar, D. Pre-text: training language models on private federated data in the age of llms. In *Proceedings of the 41st International Conference on Machine Learning*, ICML'24. JMLR.org, 2024.

Howard, J. and Ruder, S. Universal language model fine-tuning for text classification. In Gurevych, I. and Miyao, Y. (eds.), *Proceedings of the 56th Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, pp. 328–339, Melbourne, Australia, July 2018. Association for Computational Linguistics. doi: 10.18653/v1/P18-1031. URL https://aclanthology.org/P18-1031/.

Hu, E. J., yelong shen, Wallis, P., Allen-Zhu, Z., Li, Y., Wang, S., Wang, L., and Chen, W. LoRA: Low-rank adaptation of large language models. In *International Conference on Learning Representations*, 2022. URL https://openreview.net/forum?id=nZeVKeeFYf9.

Huang, X., Liu, Z., Liu, S.-Y., and Cheng, K.-T. RoLoRA: Fine-tuning rotated outlier-free LLMs for effective weight-activation quantization. In Al-Onaizan, Y., Bansal, M., and Chen, Y.-N. (eds.), *Findings of the Association*

for Computational Linguistics: EMNLP 2024, pp. 7563–7576, Miami, Florida, USA, November 2024. Association for Computational Linguistics. doi: 10.18653/v1/2024.findings-emnlp.444. URL https://aclanthology.org/2024.findings-emnlp.444/.

Kelly, D., Chen, Y., Cornwell, S. E., Delellis, N. S., Mayhew, A., Onaolapo, S., and Rubin, V. L. Bing chat: The future of search engines? *Proceedings of the Association for Information Science and Technology*, 60(1):1007–1009, October 2023. doi: 10.1002/pra2.927. URL https://doi.org/10.1002/pra2.927.

Kim, Y., Kim, J., Mok, W.-L., Park, J.-H., and Lee, S. Client-customized adaptation for parameter-efficient federated learning. In Rogers, A., Boyd-Graber, J., and Okazaki, N. (eds.), *Findings of the Association for Computational Linguistics: ACL 2023*, pp. 1159–1172, Toronto, Canada, July 2023. Association for Computational Linguistics. doi: 10.18653/v1/2023.findings-acl.75. URL https://aclanthology.org/2023.findings-acl.75/.

Lai, F., Dai, Y., Singapuram, S., Liu, J., Zhu, X., Madhyastha, H., and Chowdhury, M. Fedscale: Benchmarking model and system performance of federated learning at scale. In *International conference on machine learning*, pp. 11814–11827. PMLR, 2022.

Li, Y., Sun, J., Liu, Y., Zhang, Y., Li, A., Chen, B., Roth, H. R., Xu, D., Chen, T., and Chen, Y. Federated black-box prompt tuning system for large language models on the edge. In *Proceedings of the 30th Annual International Conference on Mobile Computing and Networking*, ACM MobiCom '24, pp. 1775–1777, New York, NY, USA, 2024. Association for Computing Machinery. ISBN 9798400704895. doi: 10.1145/3636534.3698856. URL https://doi.org/10.1145/3636534.3698856.

Lin, Z., Hu, X., Zhang, Y., Chen, Z., Fang, Z., Chen, X., Li, A., Vepakomma, P., and Gao, Y. SplitLoRA: A Split Parameter-Efficient Fine-Tuning Framework for Large Language Models. *arXiv preprint arXiv:2407.00952*, 2024.

Luo, H., Sun, Q., Xu, C., Zhao, P., Lou, J.-G., Tao, C., Geng, X., Lin, Q., Chen, S., Tang, Y., and Zhang, D. Wizardmath: Empowering mathematical reasoning for large language models via reinforced evol-instruct. In *The Thirteenth International Conference on Learning Representations*, 2025. URL https://openreview.net/forum?id=mMPMHWOdOy.

McMahan, B., Moore, E., Ramage, D., Hampson, S., and Arcas, B. A. y. Communication-Efficient Learning of Deep Networks from Decentralized Data. In Singh, A. and Zhu, J. (eds.), *Proceedings of the 20th International Conference on Artificial Intelligence and Statistics*, volume 54 of *Proceedings of Machine Learning Research*, pp. 1273–1282. PMLR, 20–22 Apr 2017. URL https://proceedings.mlr.press/v54/mcmahan17a.html.

Nadakuditi, R. R. Optshrink: An algorithm for improved low-rank signal matrix denoising by optimal, data-driven singular value shrinkage. *IEEE Transactions on Information Theory*, 60:3002–3018, 2013. URL https://api.semanticscholar.org/CorpusID:15057477.

Oh, T.-H., Matsushita, Y., Tai, Y.-W., and Kweon, I. S. Fast randomized singular value thresholding for nuclear norm minimization. In *2015 IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, pp. 4484–4493, 2015. doi: 10.1109/CVPR.2015.7299078.

Qin, Z., Chen, D., Qian, B., Ding, B., Li, Y., and Deng, S. Federated full-parameter tuning of billion-sized language models with communication cost under 18 kilobytes. In *Proceedings of the 41st International Conference on Machine Learning*, ICML'24. JMLR.org, 2024.

Srivastava, N., Hinton, G., Krizhevsky, A., Sutskever, I., and Salakhutdinov, R. Dropout: A simple way to prevent neural networks from overfitting. *Journal of Machine Learning Research*, 15(56):1929–1958, 2014. URL http://jmlr.org/papers/v15/srivastava14a.html.

Sun, T., Li, D., and Wang, B. Decentralized federated averaging. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 45(4):4289–4301, 2022.

Sun, Y., Li, Z., Li, Y., and Ding, B. Improving lora in privacy-preserving federated learning. In *The Twelfth International Conference on Learning Representations*, 2024. URL https://openreview.net/forum?id=NLPzL6HWNl.

Thirunavukarasu, A. J., Ting, D. S. J., Elangovan, K., Gutierrez, L., Tan, T. F., and Ting, D. S. W. Large language models in medicine. *Nature medicine*, 29(8):1930–1940, 2023.

Wang, Z., Shen, Z., He, Y., Sun, G., Wang, H., Lyu, L., and Li, A. Flora: Federated fine-tuning large language models with heterogeneous low-rank adaptations. In *Advances in Neural Information Processing Systems*, volume 37, pp. 22513–22533, 2024.

Zhang, J., Vahidian, S., Kuo, M., Li, C., Zhang, R., Yu, T., Wang, G., and Chen, Y. Towards building the federatedgpt: Federated instruction tuning. In *ICASSP 2024 - 2024 IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP)*, pp. 6915–6919, 2024a. doi: 10.1109/ICASSP48485.2024.10447454.

Zhang, P., Zeng, G., Wang, T., and Lu, W. Tinyllama: An open-source small language model. *arXiv preprint arXiv:2401.02385*, 2024b.

Zhang, Q., Chen, M., Bukharin, A., Karampatziakis, N., He, P., Cheng, Y., Chen, W., and Zhao, T. Adalora: Adaptive budget allocation for parameter-efficient fine-tuning, 2023. URL https://arxiv.org/abs/2303.10512.

Zhao, Z., Ziser, Y., and Cohen, S. B. Layer by layer: Uncovering where multi-task learning happens in instruction-tuned large language models. In Al-Onaizan, Y., Bansal, M., and Chen, Y.-N. (eds.), *Proceedings of the 2024 Conference on Empirical Methods in Natural Language Processing*, pp. 15195–15214, Miami, Florida, USA, November 2024. Association for Computational Linguistics. URL https://aclanthology.org/2024.emnlp-main.847.

APPENDIX

In this Appendix, we provide additional figures, analyses, and technical clarifications to support and expand upon the main paper:

- **Appendix A** highlights key limitations of existing federated fine-tuning methods using LoRA through a visual workflow.
- **Appendix B** provides a detailed computational, communication, and memory complexity comparison across all methods.
- **Appendix C** discusses additional related work beyond the main text, contextualizing our method within the broader landscape of federated and parameter-efficient tuning techniques.
- **Appendix D** includes additional convergence plots.
- **Appendix E** details the experimental setup, including datasets and baseline methods.
- **Appendix F** provides the Artifact Appendix, describing the artifact, how to access and run it, and the key results it reproduces.

A GAPS IN RELATED WORKS

This section visually illustrates the core design and limitations of existing federated fine-tuning methods that use LoRA. Each method attempts to balance fine-tuning efficiency with communication and heterogeneity support, but distinct challenges persist, especially in aggregation strategies, rank handling, and computational overhead.

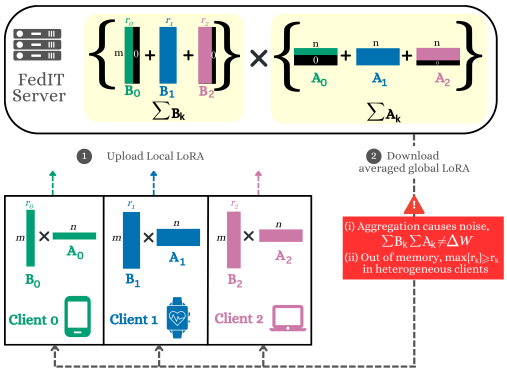


Figure 8. Workflow of FedIT (Zhang et al., 2024a) and its unique challenges. Does not support heterogeneity, natively.

FlexLoRA vs. FLoRIST. While FlexLoRA also employs an SVD-based aggregation mechanism, it differs from FLoRIST in two critical ways:

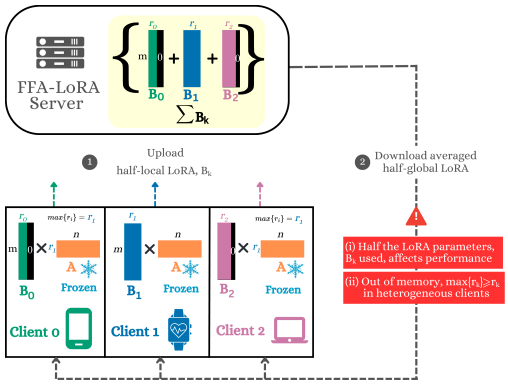


Figure 9. Workflow of FFA-LoRA (Sun et al., 2024) and its unique challenges. Does not support heterogeneity, natively.

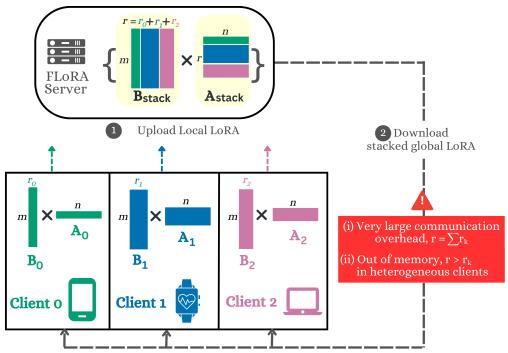


Figure 10. Workflow of FLoRA (Wang et al., 2024) and its unique challenges. Supports heterogeneity.

- **Computational Cost:** FlexLoRA explicitly constructs the global update matrix $\Delta W = \sum_{k=1}^{K} \frac{n_k}{N} \Delta W_k \in \mathbb{R}^{m \times n}$ and applies a full SVD, resulting in substantial computational and memory overhead. In contrast, FLoRIST bypasses this by operating entirely in the much smaller $r \times r$ space through separate decompositions of B_{stack} and A_{stack}.
- **Truncation Strategy.** FlexLoRA redistributes the decomposed components to clients based on their original adapter ranks, effectively matching rank to client capacity without considering global information retention. FLoRIST, in contrast, employs an *energy-based thresholding* mechanism to determine the smallest rank p such that a specified proportion τ of the singular value energy is preserved. This principled truncation leads to substantially lower communication overhead while preserving essential task-specific information.

B Complexity Analysis

Tables (5, 6, 7) summarize and compare the computational cost, communication overhead, and memory usage for all

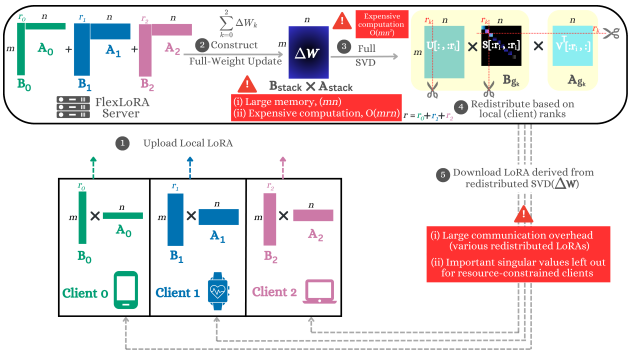


Figure 11. Workflow of FlexLoRA (Bai et al., 2024) and its unique challenges. Supports heterogeneity.

federated fine-tuning methods for LLMs using LoRA. The computational complexity of FLoRIST can be analyzed across three key stages: local client training, server-side aggregation and decomposition, and communication overhead.

Client-Side Computation. Each client k trains LoRA adapters $\{B_k, A_k\}$ for E local epochs. The cost depends on the base model, optimizer, dataset, and rank r_k. We abstract this cost as:

$$\mathcal{O}(E \cdot \mathcal{T}(m, n, r_k, |D_k|))$$

Here, m is the embedding size, n is the context length, r_k is the LoRA rank used by client k, K is the total number of clients, $|D_k|$ is the number of local training samples.

Server-Side Aggregation and Efficient SVD. In our new method, the server avoids constructing the dense update matrix $\Delta W \in \mathbb{R}^{m \times n}$ and instead performs the following operations:

1. SVD on stacked matrices:

$$B_{\text{stack}} \in \mathbb{R}^{m \times r}, \quad A_{\text{stack}} \in \mathbb{R}^{r \times n}, \quad r = \sum_{k=1}^{K} r_k$$

Each has complexity $\mathcal{O}(Lmr^2 + Lnr^2)$

2. Computing intermediate matrix:

$$Q = V_B^T U_A \in \mathbb{R}^{r \times r}, \quad P = S_B Q S_A \in \mathbb{R}^{r \times r}$$

3. SVD on P: $\mathcal{O}(Lr^3)$

4. Constructing global adapters:

$$B_g = U_B U_P S_P, \quad A_g = V_P^T V_A^T$$

$$\Rightarrow \mathcal{O}(\sum_{l=1}^{L} p_l^2 (m + n))$$

where, L is the total number of layers and p_l is the rank of the global adapters at layer l.

Overall Per-Round Complexity.

$$\mathcal{O}(E \cdot \mathcal{T}(m, n, r_k, |D_k|))$$

$$+\mathcal{O}(Lr^2(m+n+r)) + \mathcal{O}(\sum_{l=1}^{L} p_l^2(m+n))$$

The efficient decomposition approach used in FLoRIST leads to significantly reduced computational overhead compared to FlexLoRA, which performs SVD on the full matrix $\Delta W \in \mathbb{R}^{m \times n}$.

Impact of communication efficiency on end-to-end training time. To understand how FLoRIST's communication savings translate into wall-clock improvements, we analyze the per-round end-to-end (E2E) training time:

$$T_{\text{round}} \approx T_{\text{client}} + T_{\text{upload}} + T_{\text{download}} + T_{\text{server}},$$

where communication cost is $T_{\text{comm}} = c_{\text{net}} \cdot |\text{parameters}|$ and computation cost is $T_{\text{comp}} = c_{\text{flops}} \cdot \text{FLOPs}$, with c_{net} denoting the transmission cost per bit (inversely proportional to network bandwidth) and c_{flops} the computation cost per operation. As shown in Tables 5 and 6, client-side computation and upload cost are nearly identical across all methods (except FFA-LoRA, which transmits only half the adapters). Hence, the differential E2E time is dominated by $T_{\text{download}} + T_{\text{server}}$.

Defining $\alpha = c_{\text{net}}/c_{\text{flops}}$ as the ratio of communication to computation cost, the per-round time scales as:

$$T_{\text{round}} \propto \text{Server FLOPs} + \alpha \cdot |\text{Download Parameters}|.$$

In *bandwidth-limited* regimes (large α), communication latency dominates. Here, FLoRIST's substantial download reduction (up to 403× vs. Full FT, 70× vs. FLoRA; cf. Table 3) yields maximal E2E speedups. In *bandwidth-abundant* regimes (small α), computation becomes the bottleneck, yet FLoRIST retains its advantage through 7.5× lower server-side FLOPs compared to FlexLoRA (Table 4). These efficiency gains are further amplified under heterogeneous ranks, where baseline download costs grow with the maximum or summed client ranks while FLoRIST's thresholded rank p remains compact.

C Other Related Work

This section discusses additional relevant works not included in the main comparison table.

AdaLoRA. AdaLoRA (Zhang et al., 2023) adaptively allocates ranks to LoRA layers prior to training to optimize the number of trainable parameters within a fixed budget using SVD. However, it operates entirely on the client side and determines ranks *before* local training begins. This makes it unsuitable for communication-efficient federated settings where post-training compression is essential. In contrast, FLoRIST performs *server-side rank reduction after* clients upload their LoRA adapters. By applying Singular Value Thresholding (SVT) to the aggregated global adapters, it adaptively truncates them based on retained energy, enabling aggressive compression based on what was actually learned.

Key differences include:

- **Compression Timing.** FLoRIST compresses updates *post-training*, while AdaLoRA allocates rank *pre-training*.
- **Impact on Communication.** AdaLoRA does not reduce communication cost since it transmits full adapters; FLoRIST explicitly reduces global rank for efficient broadcast.
- **Training Flow.** In FLoRIST, each client initializes its local adapters from the compressed global adapters (B_g, A_g) by matching the global rank p to the client's local rank r_k via zero-padding (if $p < r_k$) or truncation (if $p > r_k$), as described in Algorithm 1. This decouples the server-side compression rank from the client-side training rank, allowing aggressive rank reduction during aggregation without constraining client expressivity.

Complementarity. Because AdaLoRA and FLoRIST act on different stages (client vs. server), they are *mutually compatible*. AdaLoRA can be integrated with FLoRIST to optimize trainable parameters locally while still benefiting from global compression.

SLoRA. SLoRA (Babakniya et al., 2023) introduces a two-phase procedure: a sparse update phase followed by LoRA fine-tuning. It applies SVD only once to initialize LoRA matrices but does not use SVD for communication efficiency or aggregation. Adapter aggregation is done via FedAvg, and no compression is applied post-training. Hence, SLoRA is **orthogonal** to FLoRIST and could potentially benefit from applying our post-training SVT compression scheme on top.

Split-LoRA. Split-LoRA (Lin et al., 2024) integrates Split Learning and LoRA to reduce per-client computational load. The model is partitioned between clients and server, with only a subset of layers trained locally. While this addresses system heterogeneity, it does not optimize communication or aggregation. Thus, Split-LoRA addresses a different challenge and is orthogonal to our focus. FLoRIST could, in principle, be combined with Split-LoRA to further reduce communication cost on the LoRA layers.

Table 5. Computational complexity of federated fine-tuning methods. $\mathcal{T}(m, n, r_k, |D_k|)$ is the per-epoch training cost; where, m is the embedding size, n is the context length, r_k is the LoRA rank used by client k, $r = \sum_{k=1}^{K} r_k$, K is the total number of clients, $|D_k|$ is the number of local training samples, p_l is the rank for layer l, L is the total number of attention layers.

Method	Client	Server
Full FT	$\mathcal{O}(E \cdot \mathcal{T}_F(m, n, \|D_k\|))$	$\mathcal{O}(LKmn)$
FedIT	$\mathcal{O}(E \cdot \mathcal{T}(m, n, r_k, \|D_k\|))$	$\mathcal{O}(L(m+n)r)$
FLoRA	$\mathcal{O}(E \cdot \mathcal{T}(m, n, r_k, \|D_k\|)) + \mathcal{O}(Lm\Sigma r_k n)$	None
FlexLoRA	$\mathcal{O}(E \cdot \mathcal{T}(m, n, r_k, \|D_k\|))$	$\mathcal{O}(Lmnr) + \mathcal{O}(LKmn) + \mathcal{O}(L\min(m,n)mn) + \mathcal{O}(Lmr^2)$
FFA-LoRA	$\mathcal{O}(E \cdot \mathcal{T}(n, r_k, \|D_k\|))$	$\mathcal{O}(Lnr)$
FLoRIST (ours)	$\mathcal{O}(E \cdot \mathcal{T}(m, n, r_k, \|D_k\|))$	$\mathcal{O}(Lr^2(m+n+r)) + \mathcal{O}(\sum_{l=1}^{L} p_l^2(m+n))$

Table 6. Communication overhead (upload and download costs for all clients per round).

Method	Upload	Download
Full FT	$\mathcal{O}(LKmn)$	$\mathcal{O}(LKmn)$
FedIT	$\mathcal{O}(L(m+n)r)$	$\mathcal{O}(LK(m+n)max(r_k)$
FLoRA	$\mathcal{O}(L(m+n)r)$	$\mathcal{O}(LK(m+n)r)$
FlexLoRA	$\mathcal{O}(L(m+n)r)$	$\mathcal{O}(L(m+n)r)$
FFA-LoRA	$\mathcal{O}(Lnr)$	$\mathcal{O}(LKn(max(r_k)))$
FLoRIST (ours)	$\mathcal{O}(L(m+n)r)$	$\mathcal{O}(K(m+n)\sum_{l=1}^{L} p_l)$

Table 7. Memory complexity (asymptotic) of federated fine-tuning methods. Client memory for LoRA-based methods are reported without the frozen base model.

Method	Client Memory	Server Memory
Full Fine Tuning	$\mathcal{O}(Lmn)$	$\mathcal{O}(Lmn)$
FedIT	$\mathcal{O}(r_k(m+n)) + \mathcal{O}(L(m+n)max(r_k))$	$\mathcal{O}(L(m+n)r)$
FLoRA	$\mathcal{O}(Lr_k(m+n)) + \mathcal{O}(L(m+n)r)$	$\mathcal{O}(L(m+n)r)$
FlexLoRA	$\mathcal{O}(Lr_k(m+n))$	$\mathcal{O}(LKmn) + \mathcal{O}(L(m^2+mn+n^2)) + \mathcal{O}(Lr(m+n))$
FFA-LoRA	$\mathcal{O}(Lr_k(m+n)) + \mathcal{O}(L(m+n)max(r_k))$	$\mathcal{O}(LKnr))$
FLoRIST (ours)	$\mathcal{O}(Lr_k(m+n))$	$\mathcal{O}(LK(m+n)r) + \mathcal{O}(Lr^2) + \mathcal{O}(\sum_{l=1}^{L} p_l(m+n))$

C2A. C2A (Kim et al., 2023) employs hypernetworks to generate personalized adapters conditioned on client-specific metadata. This method addresses client drift and personalization but does not modify aggregation or reduce communication. It is thus complementary to FLoRIST, which could serve as the backend aggregation engine while C2A handles local personalization.

RoLoRA. RoLoRA (Huang et al., 2024) improves convergence and quantization robustness by applying rotations to eliminate outliers in adapter weight space before fine-tuning. Like C2A, it operates locally and does not involve adapter aggregation or rank reduction. RoLoRA is orthogonal and could be used at the client side alongside FLoRIST's server-side aggregation.

FedTT. FedTT (Ghiasvand et al., 2024) introduces tensorized adapters to reduce parameter and communication cost. While it shares a goal with FLoRIST, its approach differs substantially—it compresses adapters using tensor decomposition rather than post-hoc SVD on aggregated weights. FedTT could be seen as an alternative approach, though it could potentially benefit from additional SVT-based compression.

FedBPT. FedBPT (Li et al., 2024) replaces adapter tuning entirely with prompt tuning. It transmits only small prompt vectors between clients and server, making it extremely communication-efficient but limited in adaptation capacity. Since it bypasses LoRA altogether, it is not comparable to FLoRIST and is considered incompatible for our setting.

FedKSeed and PrE-Text. FedKSeed (Qin et al., 2024) and PrE-Text (Hou et al., 2024) shift the focus from model-centric to data-centric personalization. FedKSeed seeds clients with shared knowledge, while PrE-Text generates synthetic local data to preserve privacy. Both are orthogonal to FLoRIST and can potentially be integrated as upstream personalization or privacy-enhancing modules.

Summary. In contrast to existing methods, FLoRIST focuses on scalable, communication-efficient *aggregation* through principled post-training rank truncation. Several works such as AdaLoRA, C2A, and RoLoRA can be layered with FLoRIST, while others such as FedTT or Split-LoRA solve complementary challenges and may benefit from integrating our SVT-based aggregation strategy.

D Additional Convergence Plots

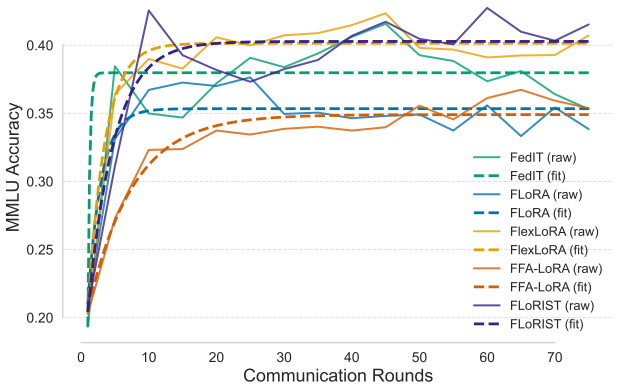
Figure 12 shows additional plots demonstrating convergence behavior of different federated fine-tuning methods across models and datasets. Each plot shows MMLU accuracy over multiple communication rounds. FLoRIST consistently converges faster and achieves higher final accuracy compared to baselines.

Training loss analysis. Figure 13 presents the average client training loss over communication rounds for two representative settings. A notable observation is that FLoRIST does not achieve the lowest training loss among all methods, in both settings, baselines such as FlexLoRA reach slightly lower loss values. This is expected and, in fact, desirable: FLoRIST's singular value thresholding explicitly discards low-energy components from the aggregated updates, which acts as a regularizer that suppresses client-specific overfitting and mitigates local client drift. As a result, while individual clients may not minimize their local training loss as aggressively, the global model generalizes better to the held-out MMLU evaluation, as evidenced by the superior accuracy in Table 2 and the convergence plots in Figures 3 and 12. This gap between training loss and evaluation accuracy underscores the regularization benefit of SVT: methods that fit local data more tightly (lower training loss) do not necessarily produce better global models, particularly under non-IID data heterogeneity, whereas FLoRIST's controlled rank reduction retains only the shared signal across clients.

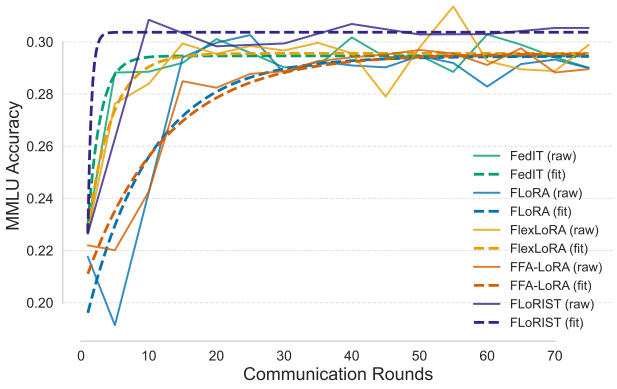
E Additional Experimental Details

E.1 Datasets

- **Dolly** (Zhang et al., 2024a) is an open-source instruction-tuning dataset consisting of 15,000 examples created by Databricks employees. It includes a broad range of instruction types across categories such as brainstorming, classification, closed and open-ended QA, summarization, information extraction, and generation. It is designed to reflect real-world user prompts and was used to train the original Dolly model series.



(a) TinyLlaMA on Wizard.



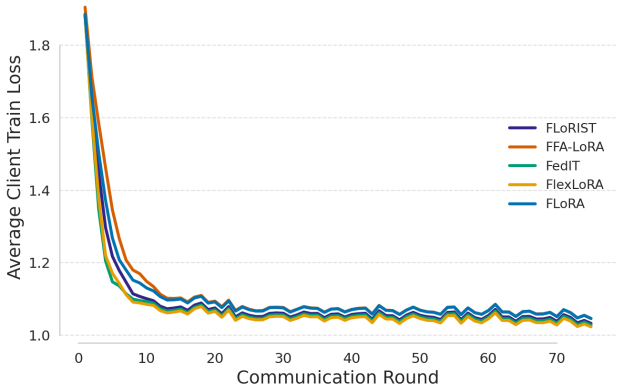
(b) LlaMA-3.2-1B on Alpaca.

Figure 12. Convergence on various model and datasets.

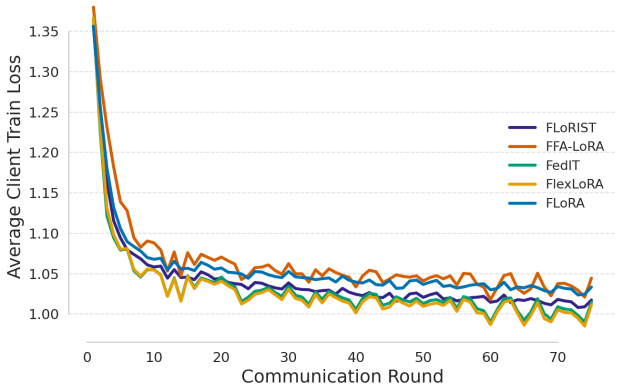
- **Alpaca** (Dubois et al., 2023) dataset contains 52,000 instruction-following samples generated by self-instructing a LlaMA model using GPT-3.5. It spans a diverse array of natural language instructions and was created to train the Alpaca model. Its wide coverage of tasks makes it a standard benchmark for instruction-tuned LLMs.

- **Wizard** (Luo et al., 2025) comprises 70,000 instruction-output pairs and serves as the training set for the WizardLM series. Compared to Dolly and Alpaca, the instructions in Wizard are typically more complex and abstract, making it a useful benchmark for evaluating instruction generalization and multi-step reasoning.

MMLU benchmark. The MMLU benchmark (Hendrycks et al., 2021) contains 14,024 multiple-



(a) TinyLlaMA on Alpaca.



(b) LlaMA-3.2-1B on Wizard.

Figure 13. Average client training loss over communication rounds. `FLoRIST` does not achieve the lowest training loss, as its singular value thresholding acts as a regularizer that suppresses client-specific drift. Despite higher local loss, the global model generalizes better (cf. Table 2).

choice questions spanning 57 diverse subjects, such as mathematics, history, law, and medicine. It is widely used to evaluate reasoning and knowledge recall in large language models. In our experiments, we sample 1,444 questions uniformly for evaluation due to resource constraints.

E.2 Baseline methods

We compared our proposed `FLoRIST` method with the following baseline approaches:

1. **FedIT** (Zhang et al., 2024a): Integrates LoRA with FedAvg to achieve communication efficiency but only supports homogeneous LoRA ranks across clients. It relies on zero-padding (HetLoRA (Cho et al., 2023)) to handle heterogeneity. HetLoRA is a simple method to enable support for heterogeneous LoRA ranks by zero-padding the smaller matrices to match the largest rank before aggregation. It is used by both FedIT and FFA-LoRA to accommodate rank differences.

2. **FLoRA** (Wang et al., 2024): Employs a stacking-based aggregation strategy that enables noise-free combination of heterogeneous LoRA modules. It achieves high performance but incurs additional communication cost proportional to client rank.

3. **FlexLoRA** (Bai et al., 2024): Allows clients to use different LoRA ranks by applying singular value decomposition (SVD) to change the rank of global adapters to match the client's local rank before fine-tuning. It avoids zero-padding and balances communication efficiency with flexibility in client ranks.

4. **FFA-LoRA** (Sun et al., 2024): Enhances communication efficiency by freezing one of the LoRA matrices during fine-tuning and transmitting only the remaining matrix. Like FedIT, it supports heterogeneity through zero-padding (HetLoRA).

F Artifact Appendix

F.1 Abstract

This artifact provides the official PyTorch implementation of **FLoRIST** (Federated Low-Rank Integration with Singular Value Thresholding), including training scripts for all five federated fine-tuning methods (FLoRIST, FedIT, FFA-LoRA, FLoRA, FlexLoRA), pre-split datasets, and evaluation code. The artifact supports federated fine-tuning of TinyLlama and LLaMA-3.2-1B on Dolly, Alpaca, and WizardLM under homogeneous and heterogeneous client rank configurations, and reproduces the FLoRIST [τ=0.9] rows in Table 2. Key reproducible results include up to 349$\times$ lower download communication cost than full fine-tuning and competitive or superior MMLU accuracy relative to all baselines across all 12 model–dataset–setting combinations.

F.2 Artifact check-list (meta-information)

- **Algorithm:** FLoRIST: federated LoRA aggregation via weighted stacking, efficient SVD in a compact $r \times r$ intermediate space, and energy-based singular value thresholding. Baselines: FedIT, FFA-LoRA, FLoRA, FlexLoRA.
- **Program:** Custom Python framework; LoRA on self-attention layers (q_proj, v_proj, k_proj, o_proj) via HuggingFace PEFT.
- **Model:** TinyLlama-1.1B (open-source, download script provided); LLaMA-3.2-1B (requires HuggingFace approval and `HF_TOKEN`). Approx. 2–5 GB each.
- **Data set:** Dolly-15k, Alpaca-52k, WizardLM-70k (all public HuggingFace datasets, pre-split in repository). MMLU subset (1,444 questions) sampled automatically.
- **Run-time environment:** Linux, Python 3.11+, PyTorch 2.x, CUDA 12.

- **Hardware:** NVIDIA GPU with 40 GB+ VRAM (A100/H100) recommended. Full experiments run on H100 cluster.
- **Execution:** 8–16 hours per full run (75 rounds, 100 clients); approx. 4 hours for LLaMA-3.2-1B/Alpaca/heterogeneous on a single H200.
- **Metrics:** MMLU accuracy (%). Communication efficiency $= 1/(\texttt{total\_rank}/2)$; communication cost (MB) $= \texttt{total\_parameters} \times 2/(1024^2)$ (FP16). Both derived from values logged to stdout.
- **Run-time state:** Results may vary slightly across runs due to non-deterministic GPU operations and stochastic client sampling. Reported accuracy values reflect convergence after 75 communication rounds.
- **Execution:** 8–16 hours per full run (75 rounds, 100 clients) on a single A100. Approx. 4 hours for LLaMA-3.2-1B/Alpaca/heterogeneous on a single H200.
- **Disk space:** Approx. 20–50 GB total (2–5 GB per model, 1–3 GB per dataset, plus checkpoints and logs).
- **Publicly available?:** Yes. https://github.com/DASS-Lab-Group/FLoRIST
- **Code licenses:** Apache 2.0.
- **Data licenses:** Dolly: CC BY-SA 3.0; Alpaca: CC BY NC 4.0; WizardLM: see dataset page; MMLU: MIT.
- **Archived (DOI):** https://doi.org/10.5281/zenodo.18945831

F.3 Description

F.3.1 How to access

Clone from https://github.com/DASS-Lab-Group/FLoRIST or download the archived release at https://doi.org/10.5281/zenodo.18945831.

F.3.2 Hardware dependencies

NVIDIA GPU with 40 GB+ VRAM (A100/H100). TinyLlama can run at 16–24 GB VRAM with reduced batch size.

F.3.3 Software dependencies

Python 3.11+, PyTorch 2.x (CUDA 12), HuggingFace `transformers`, `peft`, `datasets`, `numpy`, `scipy`, `tqdm`. Install via `pip install -r requirements.txt`.

F.3.4 Data sets

All three datasets are included pre-split in the repository (`./data/`, `./data_alpaca/`, `./data_wiz/`). No manual download required.

F.3.5 Models

TinyLlama-1.1B must be downloaded locally using the provided `download.py` script followed by a `wget` for the model weights (see README). LLaMA-3.2-1B is loaded from HuggingFace Hub at runtime, no local download needed, but requires accepting Meta's license at https://huggingface.co/meta-llama/Llama-3.2-1B and setting `export HF_TOKEN=your_token_here`.

F.4 Installation

```
# 1. Clone repository
git clone \
https://github.com/DASS-Lab-Group/FLoRIST.git
cd FLoRIST

# 2. Install dependencies
pip install -r requirements.txt

# 3. Download TinyLlama (if needed)
python download.py
cd tinyllama && wget -O model.safetensors \
  "https://huggingface.co/TinyLlama/
  TinyLlama-1.1B-Chat-v1.0/resolve/
  main/model.safetensors"

# 4. For LLaMA-3.2-1B only: set HF token
export HF_TOKEN=your_token_here
```

F.5 Experiment workflow

All experiments are launched via `main.py`. Key flags: `--global_model` (`tinyllama`/`llama3.2-1b`), `--method` (`florist`/`fedit`/`flora`/`flex`/`ffa`), `--threshold` ($\tau \in [0.80, 0.99]$, FLoRIST only), `--heter True` (heterogeneous client ranks), `--zero_padding True` (for FedIT/FFA-LoRA in heterogeneous setting). Shell scripts for all paper configurations are provided in the repository. Each run logs per-round MMLU accuracy, total LoRA rank, and total parameters communicated to stdout.

F.6 Evaluation and expected results

Running FLoRIST at $\tau = 0.9$ reproduces the FLoRIST [τ=0.9] rows in Table 2. We recommend the **LLaMA-3.2-1B / Alpaca / Heterogeneous** configuration as the primary target, as it completes in approximately 4 hours on a single H200 and exercises the full pipeline including heterogeneous rank aggregation. A reference output log for this configuration is included in `logs/` in the repository.

Key expected outcomes (FLoRIST [τ=0.9], Table 2):

- TinyLlama / Homo / Dolly: 30.94%, 23.48×10^{-4}
- TinyLlama / Homo / Alpaca: 31.68%, 60.06×10^{-4}
- TinyLlama / Homo / Wizard: 38.92%, 63.09×10^{-4}

- TinyLlama / Heter / Wizard: 41.51%, 36.16×10^{-4}
- LLaMA-3.2-1B / Homo / Dolly: 29.48%, 33.64×10^{-4}
- LLaMA-3.2-1B / Heter / Alpaca: 30.24%, 48.24×10^{-4}

Allowable variation: MMLU accuracy $\pm0.5\%$; efficiency $\pm5\%$ across runs due to non-deterministic GPU operations and stochastic client sampling.

F.7 Experiment customization

Threshold τ: `--threshold` accepts any value in $[0.80, 0.99]$. Lower values yield stronger compression (lower rank, higher efficiency) at a small accuracy cost; $\tau = 0.9$ is a robust default. Figure 7 in the paper shows the full accuracy–efficiency curve.

Client heterogeneity: `--heter True/False` switches between homogeneous (all rank 16) and heterogeneous (heavy-tail: 40 clients at rank 4, 20 at rank 8, 20 at rank 16, 10 at rank 32, 10 at rank 64).

Quick sanity check: Reduce `--num_communication_rounds` to 10 and `--clients_per_round` to 5 for a fast end-to-end functional check (∼1–2 hours).

Baseline comparison: Swap `--method` among `florist`, `flora`, `fedit`, `flex`, `ffa` to reproduce any baseline on the same data split and hardware.

F.8 Methodology

Submission, reviewing and badging methodology:

- https://www.acm.org/publications/policies/artifact-review-and-badging-current
- https://cTuning.org/ae