Communication-Efficient Federated Low-Rank Update Algorithm and its Connection to Implicit Regularization

Haemin Park [1] **Diego Klabjan** [1]

Abstract

Federated Learning (FL) faces significant challenges related to communication efficiency and performance reduction when scaling to many clients. To address these issues, we explore the potential of using low-rank updates and provide the first theoretical study of rank properties in FL. Our theoretical analysis shows that a client's loss exhibits a higher-rank structure (i.e., gradients span higher-rank subspaces of the Hessian) compared to the server's loss, and that low-rank approximations of the clients' gradients have greater similarity. Based on this insight, we hypothesize that constraining client-side optimization to a low-rank subspace could provide an implicit regularization effect while reducing communication costs. Consequently, we propose **FedLoRU**, a general low-rank update framework for FL. Our framework enforces low-rank client-side updates and accumulates these updates to form a higher-rank model. Additionally, variants of FedLoRU can adapt to environments with statistical and model heterogeneity by employing multiple or hierarchical low-rank updates. Experimental results demonstrate that FedLoRU performs comparably to full-rank algorithms and exhibits robustness to heterogeneous and large numbers of clients.

1. Introduction

Federated learning (FL, (McMahan et al., 2017)) is a collaborative learning framework designed to enhance privacy preservation by training models on clients' local data without sharing raw information. While FL offers privacy benefits, it trades off some performance compared to centralized learning, largely due to communication overhead and heterogeneity. Despite improvements in computation and memory capacities, communication speeds have only slightly improved, making communication overhead a major factor in slowing down FL (Zheng et al., 2020). Additionally, various forms of heterogeneity—statistical, system, and device—further complicate FL (Ye et al., 2023; Kairouz et al., 2021). These issues are especially pronounced with a large number of clients, where frequent, less impactful updates slow down training and reduce performance.

Addressing these challenges is becoming increasingly critical, for example, training large language models (LLMs) in FL. Utilizing private datasets on edge devices for LLM training is promising due to the limited availability of public data (Ye et al., 2024). However, this approach presents significant issues, notably in terms of communication overhead, as edge devices possess heterogeneous resources and data. Additionally, the need for effective regularization across clients is required. To address the two main challenges of communication overhead and performance reduction with increasing local clients in FL, we analyze the rank nature of loss landscape in FL and leverage low-rank updates.

There has been substantial research focusing on the low-rank characteristics in centralized learning. By low rank, we refer to gradients spanning a low rank subspace of Hessian at any given weights or the weight matrix being of the form $\boldsymbol{AB}$ where the number of columns of $\boldsymbol{A}$ is low. Methods such as LoRA (Hu et al., 2021), DyLoRA (Valipour et al., 2022), and QLoRA (Dettmers et al., 2024) utilize this factorization to decrease the number of trainable parameters, thus conserving memory and computational resources. Further observations (Huh et al., 2021; Ji & Telgarsky, 2018) indicate that over-parameterized models tend to find low-rank solutions, which provide implicit regularization effects.

However, the rank properties of the loss landscape in FL remain under-explored. Herein, we first analyze the difference in the stable rank—defined as the squared ratio of the Frobenius norm to the spectral norm—between client Hessians and the server Hessian of any weights, discovering that a client exhibits a higher-rank structure. We also show that low-rank approximations of local gradients align better in direction than their full-rank counterparts. Based on this insight, we hypothesize that the client's higher-rank Hessian amplifies cross-client discrepancies, and that restricting

[1]Department of Industrial Engineering & Management Sciences, Northwestern University, Evanston, USA. Correspondence to: Haemin Park <haemin.park1@northwestern.edu>, Diego Klabjan <d-klabjan@northwestern.edu>.

client-side updates could offer both implicit regularization and reduced communication costs.

To address this, we propose the Federated Low-Rank Updates (FedLoRU) algorithm, which mitigates communication overhead and accommodates many clients through low-rank updates. FedLoRU factorizes client-side update matrices into $\boldsymbol{A}$ and $\boldsymbol{B}$ and applies iterative optimization to these low-rank factorized matrices. Clients and the server share the factorized matrices, which the server then aggregates. Matrices $\boldsymbol{A}$ and $\boldsymbol{B}$ are being communicated between the clients and server, rather than the much larger matrix $\boldsymbol{AB}$. To make the model's weight rank high, FedLoRU successively accumulates low-rank matrices. We also generalize the low-rank update strategy within federated learning for various heterogeneous settings.

Our comprehensive approach underscores the potential of low-rank updates not only to enhance communication efficiency but also to impose implicit regularization. Our contributions can be summarized as follows. 1) We propose FedLoRU, the first algorithm using successive low-rank updates for both pre-training and fine-tuning in federated learning, and introduce variants of FedLoRU for personalization and model heterogeneity settings; 2) We investigate the rank properties of client and server losses, analytically showing that under stochastic sampling and sufficiently large model, the stable rank of the Hessian of the loss function increases with smaller sample sizes; 3) We provide empirical evidence of the higher rank structure of client losses and demonstrate that restricting the rank of local updates aids in implicit regularization; 4) On average, FedLoRU improves state-of-the-art communication-efficient federated learning algorithms on a variety of datasets, including LLM fine-tuning, and exhibits superior performance as the number of clients increases.

2. Related Work

Communication-Efficient Federated Learning Extensive research has addressed communication challenges in FL (Shahid et al., 2021). FedPAQ (Reisizadeh et al., 2020) and AdaQuantFL (Jhunjhunwala et al., 2021) employ quantization to reduce the precision of weights, while Fed-Dropout (Caldas et al., 2018) and FedMP (Jiang et al., 2023) apply pruning to remove less important weights. Since quantization and sparsification do not alter the core network structure, they can be easily combined with other algorithms (e.g., FedLoRU) to reduce communication overhead.

In contrast, model compression techniques modify the model structure itself by compressing the original model before communication and restoring it afterward. FedDLR (Qiao et al., 2021) compresses using low-rank approximation for both server-to-client and client-to-server communication but reverts to the full model for local training. FedHM (Yao et al., 2021) compresses only during server-to-client communication, where clients train factorized low-rank models that are aggregated by the server. Although both methods reduce communication overhead, their server-side compression approaches can lead to performance degradation. To mitigate potential information loss during server-side compression, we focus on client-side factorization, avoiding compression processes.

Low-rank nature of centralized and federated learning Numerous studies (Gur-Ari et al., 2018; Li et al., 2018; Sagun et al., 2016) assert that the training process in deep learning inherently possesses a low-rank nature. Low-Rank Adaptation (LoRA, (Hu et al., 2021)) is a representative algorithm that leverages this low-rank characteristic, particularly for fine-tuning tasks, by freezing pre-trained weights and applying low-rank updates via the decomposition $\boldsymbol{W} = \boldsymbol{W}_0 + \boldsymbol{AB}$, where $\boldsymbol{W}_0 \in \mathbb{R}^{m\times n}$, $\boldsymbol{A} \in \mathbb{R}^{m\times r}$, $\boldsymbol{B} \in \mathbb{R}^{r\times n}$, $r \ll m, n$. However, effectively leveraging the low-rank structure during pre-training remains a challenge, as the weights do not inherently exhibit a low-rank nature (Yu & Wu, 2023; Zhao et al., 2024). To address this, ReLoRA (Lialin et al., 2023) seeks to achieve a higher-rank model by accumulating multiple low-rank updates, expressed as $\boldsymbol{W} = \boldsymbol{W}_0 + \sum_{i=1}^{M} \boldsymbol{A}_i\boldsymbol{B}_i$ where $\boldsymbol{A}_i \in \mathbb{R}^{m\times r}$, $\boldsymbol{B}_i \in \mathbb{R}^{r\times n}$.

In federated learning, some research has aimed to exploit the low-rank nature observed in centralized learning. LBGM (Azam et al., 2021) and FedLRGD (Jadbabaie et al., 2023) approximate gradients using past or sampled gradients, assuming gradients lie in a low-rank subspace. However, there is a noticeable gap in analyzing rank characteristics specific to federated learning. In the context of federated learning, there is a complex loss landscape involving multiple client-side and a single server-side optimization, and leveraging a low-rank structure needs to consider their respective rank structures. To our knowledge, no prior work has examined the rank structure in federated learning contexts without making very stringent assumptions. Our study is pioneering in addressing this gap, using analytical results and insights to develop a novel algorithm.

Low-Rank Adaptation in Federated Learning Recent studies have studied the application of LoRA within federated learning frameworks. Notable algorithms, such as FedLoRA (Wu et al., 2024; Yi et al., 2023), FFALoRA (Sun et al., 2024), and Hyperflora (Lu et al., 2024), employ LoRA adapters to facilitate personalization. These methods apply low-rank adaptation to a pre-trained model during the local personalization training phase. On the other hand, other works (Zhang et al., 2023; Kuo et al., 2024; Cho et al., 2023) apply LoRA for fine-tuning within federated learning

environments.

These approaches use only one low-rank matrix that restricts the model to a low-rank subspace. In contrast, we utilize multiple accumulated low-rank matrices allowing the model to achieve higher rank. Specifically, we extend the concept of LoRA by incorporating client-side low-rank updates and server-side accumulation to address the low-rank limitation of LoRA as well as the challenges posed by communication and client-server rank disparity. We also generalize the low-rank strategy within federated learning for both pre-training and fine-tuning, and for heterogeneous environments.

3. Analyzing Rank Nature in FL

In centralized learning, neural network losses exhibit a low-rank structure, indicating that the gradient lies within the subspace spanned by the Top-k eigenvectors of the Hessian during training (Gur-Ari et al., 2018). Although efforts have been made to utilize this low-rank structure to enhance federated learning algorithms, there is a lack of studies that analyze the rank structure of federated learning. To the best of our knowledge, we are the first to provide a theoretical analysis of the rank structure in FL. Specifically, we examine the rank properties of FL and explore the effect of low-rank updates on enhancing gradient alignment.

Notation and problem setup Suppose $\psi(\boldsymbol{x}, \boldsymbol{y})$ is a data generating distribution for an input-output pair $(\boldsymbol{x}, \boldsymbol{y}) \in \mathbb{R}^{d_x} \times \mathbb{R}^{d_y}$. We consider the problem of finding a prediction function $h^R(\cdot;\cdot) : \mathbb{R}^{d_x} \times \mathbb{R}^R \to \mathbb{R}^{d_y}$ parameterized by a R-dim weight vector $\omega^R \in \mathbb{R}^R$. Given a loss function $\ell(\cdot,\cdot) : \mathbb{R}^{d_y} \times \mathbb{R}^{d_y} \to \mathbb{R}$, the true risk $\mathcal{L}_{\text{true}}(h^R, \omega^R) = \int \ell(h^R(\boldsymbol{x}; \omega^R), \boldsymbol{y}) d\psi(\boldsymbol{x}, \boldsymbol{y})$ is defined as the loss over the data-generating distribution $\psi(\boldsymbol{x}, \boldsymbol{y})$. The corresponding true Hessian is $\boldsymbol{H}_{\text{true}}(h^R, \omega^R) = \nabla^2 \mathcal{L}_{\text{true}}(h^R, \omega^R)$. If $\mathcal{D}_N = \{(\boldsymbol{x}_1, \boldsymbol{y}_1), \cdots, (\boldsymbol{x}_N, \boldsymbol{y}_N)\}$ is a dataset generated from the distribution ψ, the empirical loss and Hessian for $\mathcal{D}_N$ are $f_N(h^R, \omega^R) = \sum_{(x,y)\in\mathcal{D}_N} \frac{1}{N} \ell(h^R(x; \omega^R), y)$ and $\boldsymbol{H}_N(h^R, \omega^R) = \sum_{(x,y)\in\mathcal{D}_N} \frac{1}{N} \frac{\partial^2}{\partial(\omega^R)^2} \ell(h^R(x; \omega^R), y)$.

We consider a random selection of M samples without replacement from $\mathcal{D}_N$ to form a sub-dataset $\mathcal{D}_M \subseteq \mathcal{D}_N$. Let $f_M(h^R, \omega^R)$ and $\boldsymbol{H}_M(h^R, \omega^R)$ denote the loss and Hessian for the sub-dataset $\mathcal{D}_M$. In federated learning, f_N can be considered as the loss that the server optimizes, while f_M represents the loss of a local client assuming the homogeneous setting.

3.1. Higher Rank Nature of Clients in FL

In this section, we demonstrate that the local Hessian possesses a higher stable rank than the server's Hessian when the model size is sufficiently large. This indicates that the loss landscape at a client is more complex than that of the server, which may contribute to divergence of local models after local training.

Stable rank To compare the rank properties of Hessians of a client and the server, we use the stable rank $\text{srank}(\boldsymbol{A}) = \frac{\|\boldsymbol{A}\|_F^2}{\|\boldsymbol{A}\|_2^2} = \frac{\sum_{i=1}^n \sigma_i^2(\boldsymbol{A})}{\sigma_1^2(\boldsymbol{A})}$, where n is the rank of matrix $\boldsymbol{A}$ and $\sigma_i(\boldsymbol{A})$ denotes its i-th singular value. Unlike traditional rank, which discretely counts non-zero singular values, the stable rank provides a continuous and more informative proxy, effectively capturing the low-rank nature of deep learning since stable rank is sensitive to the distribution of the singular values. This property is particularly useful in deep learning, where gradient descent trajectories are often dominated by a few large eigenvalues, and the subspace spanned by the corresponding eigenvectors critically influences training dynamics (Gur-Ari et al., 2018; Sagun et al., 2016; Sabanayagam et al., 2023). By emphasizing the contribution of large eigenvalues, the stable rank serves as a practical tool for quantifying the curvature of the loss landscape.

Moreover, the stable rank exhibits robustness to small perturbations in the Hessian. In practice, minor changes in model parameters or data points can lead to significant variations in the traditional rank, but these do not substantially affect the stable rank. This robustness ensures that stable rank provides consistent insights to the loss landscape, even under small variations in the training process.

Comparing the stable rank of the client and server Hessians For non-zero real numbers $\theta_1, \cdots, \theta_k$, we define $\Omega^R(\theta_1, \cdots, \theta_k)$ as the family of pairs (h^R, ω^R), where h^R is an R-dimensional prediction function and ω^R is a weight vector, such that the true Hessian has non-zero eigenvalues $\theta_1, \cdots, \theta_k$. Specifically, $\Omega^R(\theta_1, \cdots, \theta_k) = \{(h^R, \omega^R) : \boldsymbol{H}_{\text{true}}(h^R, \omega^R)$ has non-zero eigenvalues $\theta_1, \cdots, \theta_k\}$. Let $\Omega(\theta_1, \cdots, \theta_k) = \bigcup_R \Omega^R(\theta_1, \cdots, \theta_k)$, representing the union of $\Omega^R(\theta_1, \cdots, \theta_k)$ over all dimensions R. We aim to show that the difference in the stable rank between the Hessians of the server and a client eventually becomes positive as dimension R approaches infinity within the space of $\Omega(\theta_1, \cdots, \theta_k)$, which contains infinitely many R for which $\Omega^R(\theta_1, \cdots, \theta_k) \neq \emptyset$, as proved in Appendix A.1.

We next focus on comparing the stable rank of the client and server Hessians. For given $p, q \in \mathbb{N}$, let $\theta_1 > \cdots > \theta_p > 0 > \theta_{p+1} > \cdots > \theta_{p+q}$ be deterministic non-zero real numbers. Let R be an integer such that $R \geq \bar{R}$, where $\bar{R}$ is the smallest integer for which $\Omega^{\bar{R}}(\theta_1, \cdots, \theta_{p+q})$ is non-empty and consider any $(h^R, \omega^R) \in \Omega^R(\theta_1, \cdots, \theta_{p+q})$. Here, $\boldsymbol{H}_N(h^R, \omega^R)$ and $\boldsymbol{H}_M(h^R, \omega^R)$ represent server and local Hessians in FL, respectively. To compare the stable rank of two Hessians, we consider the additive perturbed model of the true Hessian as described by (Baskerville et al., 2022):

$$\boldsymbol{H}_N(h^R, \omega^R) = \boldsymbol{H}_{\text{true}}(h^R, \omega^R) + \epsilon_N^R, \quad (1)$$
$$\boldsymbol{H}_M(h^R, \omega^R) = \boldsymbol{H}_{\text{true}}(h^R, \omega^R) + \epsilon_M^R. \quad (2)$$

Here, $\epsilon_N^R, \epsilon_M^R \in \mathbb{R}^{R\times R}$ are random error matrices associated with each Hessian. These matrices are assumed to be scaled according to $\epsilon_N^R = s_N X^R$, where $X^R \in \mathbb{R}^{R\times R}$ is a random real symmetric matrix where each element is independently drawn from a distribution with mean 0 and variance σ^2/R. The scaling factor $s_N = s(N)$ is defined as a monotonic decreasing function mapping $\mathbb{N}$ to $(0, 1)$. For simplicity in notation, we use $\boldsymbol{H}_N^R = \boldsymbol{H}_N(h^R, \omega^R)$ and $\boldsymbol{H}_{\text{true}}^R = \boldsymbol{H}_{\text{true}}(h^R, \omega^R)$ whenever the context is clear.

Notably, X^R is a Wigner matrix, commonly used in Random Matrix Theory (RMT) as an error or perturbation matrix. In this role, Wigner matrices capture statistical fluctuations in the eigenvalues and eigenvectors of a Hessian, making it particularly suitable for analyzing a loss landscape. For further discussion on X^R, please refer to Appendix A.5.

(Granziol et al., 2022) employs the model $\boldsymbol{H}_M^R = \boldsymbol{H}_N^R + \epsilon^R$, implying a dependency structure between $\boldsymbol{H}_M^R$ and $\boldsymbol{H}_N^R$. However, their analysis assumes independence between these matrices, which is problematic given the underlying model and practical considerations. In contrast, we address this issue by introducing two decoupled additive perturbed models.

Next, we determine the limiting eigenvalues of the Hessians $\boldsymbol{H}_N^R$ in relation to the eigenvalues of $\boldsymbol{H}_{\text{true}}^R$ as $R \to \infty$.

Proposition 3.1 (Limiting eigenvalues of $\boldsymbol{H}_N^R$ (modified from (Baskerville et al., 2022)))**.** *Let $\boldsymbol{H}_N^R$ defined as in (1). If $\lambda_i(\boldsymbol{H}_N^R)$ denotes the i-th eigenvalue of $\boldsymbol{H}_N^R$, then for $i = 1, \cdots, p$, the following holds:*

$$\lambda_i(\boldsymbol{H}_N^R) \to \begin{cases} g_N^{-1}(\theta_i) & \text{if } g_N^{-1}(\theta_i) > U_N \\ U_N & \text{otherwise} \end{cases} \quad (3)$$

as $R \to \infty$, and for $i = 0, \cdots, q-1$, we have

$$\lambda_{R-i}(\boldsymbol{H}_N^R) \to \begin{cases} g_N^{-1}(\theta_{p+q-i}) & \text{if } g_N^{-1}(\theta_{p+q-i}) < L_N \\ L_N & \text{otherwise.} \end{cases} \quad (4)$$

Here, $g_N^{-1}(\theta) = \theta + \frac{\sigma^2 s_N^2}{\theta}$, $U_N = 2\sigma s_N$, and $L_N = -2\sigma s_N$. In addition, for $p < i \leq P - q$, we have $\lambda_i(\boldsymbol{H}_N^R) \to \{L_N, U_N\}$.

Convergence in our analysis is almost sure uniform convergence. Compared to (Baskerville et al., 2022), which focuses solely on outlier eigenvalues with a particular form of μ, we extend the analysis to bulk eigenvalues and use a general μ with compact support. By the proposition, the i-th largest or smallest limiting eigenvalues of $\boldsymbol{H}_N$ are determined by the values of $g_N^{-1}(\theta_i)$. If $g_N^{-1}(\theta_i)$ falls within the support of μ_N, the corresponding limiting eigenvalues converge to the bounds. If $g_N^{-1}(\theta_i)$ does not lie within this support, it converges to $g_N^{-1}(\theta_i)$ itself; these eigenvalues are typically referred to as outlier eigenvalues in the literature. The detailed proof is provided in Appendix A.2 and is similar to the proof in (Baskerville et al., 2022).

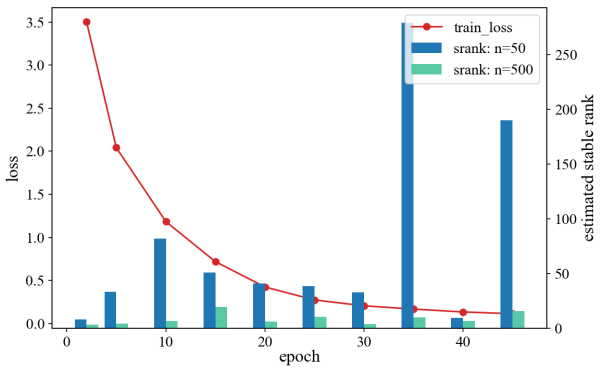


Figure 1: The estimated stable ranks of the Hessians are compared for dataset sizes of 50 and 500 (averaged over multiple runs). The estimated stable rank for the size of 50 consistently exceeds that of 500. For details of the experiment, see Appendix C.3

In the following theorem, we demonstrate that a smaller dataset results in a higher stable rank in the limit except for the extremely ill-conditioned situation.

Theorem 3.2. *Let $\boldsymbol{H}_N^R$ and $\boldsymbol{H}_M^R$ be the Hessians as defined in (1) and (2) and define $\theta_0 = \theta_1 \cdot \mathbf{1}_{|\theta_1|\geq|\theta_{p+q}|} + \theta_{p+q} \cdot \mathbf{1}_{|\theta_1|<|\theta_{p+q}|}$. Assume $\theta_0^2 \geq \sigma^2 s_M^2$. Then the difference in the limiting stable rank between $\boldsymbol{H}_N^R$ and $\boldsymbol{H}_M^R$ is positive and bounded below as follow*

$$\hat{srank}(\boldsymbol{H}_M) - \hat{srank}(\boldsymbol{H}_N) \geq \frac{s_M^2 - s_N^2}{g_M^{-1}(\theta_0)^2 g_N^{-1}(\theta_0)^2} \left[\sum_{j\in\mathcal{P}_N\cup\mathcal{Q}_N} 8\sigma^4 s_M s_N \left| \frac{\theta_0}{\theta_j} - \frac{\theta_j}{\theta_0} \right| + 4\sigma^2 B_N \left(\theta_0^2 - \frac{\sigma^4 s_M^2 s_N^2}{\theta_0^2} \right) \right], \quad (5)$$

where $B_N = |\{i : \lambda_i(\boldsymbol{H}_N^R) \to U_N \text{ or } L_N\}|$, $\mathcal{P}_N = \{i \leq p : g_N^{-1}(\theta_i) > U_N\}$, and $\mathcal{Q}_N = \{i > p : g_N^{-1}(\theta_i) < L_N\}$. Furthermore, the lower bound decreases with M.

This theorem characterizes the stable rank difference between $\boldsymbol{H}_M^R$ and $\boldsymbol{H}_N^R$ by showing that it is bounded below by a term proportional to $(s_M^2 - s_N^2)$. As M decreases relative to N, this term increases. In the special case where $\theta_0^2 \leq \sigma^2 s_M^2$, the gap can become negative; however, this scenario arises only when the Hessian is extremely ill-conditioned, meaning that the largest singular value is

extremely small. Under a typical scaling assumption such as $s_M = 1/M$, $\sigma^2 s_M^2$ remains sufficiently small in most practical settings, making such ill-conditioning unlikely. Consequently, except for highly degenerate settings, the stable rank difference between local and global Hessians remains strictly positive, and the magnitude of this difference grows as the size of each local dataset becomes smaller relative to the aggregate dataset.

Our empirical results in Figure 1 further support this by demonstrating that smaller datasets exhibit higher estimated stable ranks.

3.2. Gradient Alignment Effect of Local Low-Rank Updates

In this section, we examine how low-rank approximations of local gradients promote alignment among clients in an FL setting. Intuitively, as the approximation rank r decreases, the components of each local gradient become more concentrated along the most significant directions of its Hessian, which in turn improves similarity across different clients. This phenomenon provides an important insight into the benefits of performing local low-rank updates: even when clients operate on potentially smaller datasets, restricting updates to low-rank directions can enhance overall gradient alignment.

Building on results from (Benaych-Georges & Nadakuditi, 2011), we know the limiting eigenvector transition. For $i \in \mathcal{P}_N \cup \mathcal{Q}_N$, let v_i be the unit-norm eigenvector associated with the eigenvalue θ_i of H^R_{true} and let u_i be the corresponding unit-norm eigenvector of $\boldsymbol{H}^R_N$. Then for $j \in \{j \in \mathcal{P}_N \cup \mathcal{Q}_N : j \neq i\}$, we have

$$|\langle v_i, u_i\rangle|^2 \to 1 - \frac{\sigma^2 s_N^2}{\theta_i^2}, \tag{6}$$

$$|\langle v_j, u_i\rangle|^2 \to 0. \tag{7}$$

In other words, each limiting eigenvector of $\boldsymbol{H}^R_N$ lies in a cone around the corresponding eigenvector of $\boldsymbol{H}^R_{\text{true}}$. When N is small, $\langle v_i, u_i\rangle$ remains farther from unity. This implies that the similarity between the eigenvectors of $\boldsymbol{H}^R_N$ and $\boldsymbol{H}^R_{\text{true}}$ is diminished in the regime of small N. Moreover, for a client operating with a dataset size $M < N$, the spectral similarity $\langle v_i, u_i\rangle$ becomes smaller than that of a client with a larger dataset. This phenomenon can degrade performance when a client holds very limited local data, as its local Hessian captures fewer reliable directions than one computed from a larger dataset.

For eigenvectors corresponding to bulk eigenvalues, numerous studies (Anderson et al., 2010; Antti Knowles, 2013) have demonstrated that these eigenvectors exhibit an isotropic distribution, with each component behaving as an independent and identically distributed random variable with zero mean and variance $O(1/R)$. This indicates that no single element of the eigenvector dominates. Additionally, the bulk eigenvectors primarily arise from the random noise in ϵ^R_N. Accordingly, we assume that the bulk eigenvectors are random vectors residing in the subspace orthogonal to that spanned by the edge eigenvectors.

Gradient alignment We define the full-rank approximation of $\nabla f_N(h^R, \omega^R)$ with respect to $\boldsymbol{H}^R_N$ as

$$\nabla \hat{f}_{N,\text{full}}(h^R, \omega^R) \;=\; \sum_{i=1}^{s} \partial_{u_i} f_N(h^R, \omega^R)\, u_i, \tag{8}$$

where $u_1, \ldots, u_s$ are eigenvectors of $\boldsymbol{H}^R_N$ associated with the eigenvalues $\theta_1, \ldots, \theta_s$, ordered by magnitude, and $\partial_u f_N(h^R, \omega^R)$ is the directional derivative of $f_N(h^R, \omega^R)$ with respect to u. A rank-r approximation then restricts this sum to only the top-r eigenvectors

$$\nabla \hat{f}_{N,r} \;=\; \sum_{i=1}^{r} \partial_{u_i} f_N(h^R, \omega^R)\, u_i. \tag{9}$$

Given K clients, each with a dataset of size N, we denote their corresponding Hessians by $\boldsymbol{H}^{(k)}_N$ for $k \in \{1, \ldots, K\}$. Let

$$C^R_{N,r}(k_1, k_2) \;=\; \cos\Big(\nabla \hat{f}^{(k_1)}_{N,r}\,,\, \nabla \hat{f}^{(k_2)}_{N,r}\Big) \tag{10}$$

be the cosine similarity between the rank-r approximations of the gradients of clients k_1 and k_2.

Theorem 3.3. *For any $r \in \mathbb{N}$ and $k_1, k_2 \in \{1, \ldots, K\}$ with $k_1 \neq k_2$,*

$$\left|\mathbb{E}\left[C^R_{N,r}(k_1, k_2) - C^R_{N,r+1}(k_1, k_2)\right] - g_N(r)\right| \to 0 \tag{11}$$

as $R \to \infty$, where $g_N(r)$ is strictly positive and expressed in the proof in Appendix A.4.

According to Theorem 3.3, the expected cosine similarity between two clients' rank-r gradient approximations decreases as r increases for large R. Specifically, once r is large enough to include all dominant directions, adding an additional component contributes random noise from the bulk eigenvectors, thereby reducing the directional alignment. For small r, incorporating the $(r+1)$-th principal direction also reduces similarity, because although it remains more critical than the bulk noise directions, it contributes less universally aligned signal than the top-r directions.

One major limitation of Theorem 3.3 is that it analyzes the alignment between the gradient approximations of the clients based on a single update step. However, in practical federated learning settings, each local client typically performs multiple gradient descent steps during each round.

Algorithm 1 FedLoRU.

Require: model $\boldsymbol{W}$, initial low-rank update matrices $\boldsymbol{A}_0, \boldsymbol{B}_0$
Require: scaling factor α, accumulation cycle τ, total round T
  Initialize: Server sends $\boldsymbol{W}$ to each client.
  for $t = 1, \cdots, T$ **do**
    Server selects M clients $\mathcal{K}_M$.
    Server distributes $\boldsymbol{A}_{t-1}, \boldsymbol{B}_{t-1}$ to clients in $\mathcal{K}_M$.
    for each client $k \in \mathcal{K}_M$ **do**
      Find $\boldsymbol{A}_t^{(k)}, \boldsymbol{B}_t^{(k)}$ by solving (12) starting from $\boldsymbol{A}_{t-1}, \boldsymbol{B}_{t-1}$.
      Send $\boldsymbol{A}_t^{(k)}, \boldsymbol{B}_t^{(k)}$ to the server.
    end for
    Server aggregation:
    $\boldsymbol{A}_t \leftarrow \sum_{k\in\mathcal{K}_M} p^{(k)} \boldsymbol{A}_t^{(k)}$, $\boldsymbol{B}_t \leftarrow \sum_{k\in\mathcal{K}_M} p^{(k)} \boldsymbol{B}_t^{(k)}$.
    if $t \bmod \tau = 0$ **then**
      Server distributes $\boldsymbol{A}_t, \boldsymbol{B}_t$ to all clients .
      Each client k updates its local copy of the global model: $\boldsymbol{W} \leftarrow \boldsymbol{W} + \alpha \boldsymbol{A}_t \boldsymbol{B}_t$.
    end if
  end for
  Return: $\boldsymbol{W} + \alpha \sum_{t=1:\ t \bmod \tau=0}^{T} \boldsymbol{A}_t \boldsymbol{B}_t$.

Thus, it does not directly guarantee that the overall alignment of the representative gradients, defined as the difference between a client's updated model and the global model after one round of training, would exhibit the same behavior. Further analysis is needed, but we leave this for future research.

4. Federated Low-Rank Update (FedLoRU) Algorithm

Consider a federated learning system with K clients, where each client k has its own loss function $f^{(k)} : \mathbb{R}^{m\times n} \to \mathbb{R}$. The server aims to find a global model $\boldsymbol{W} \in \mathbb{R}^{m\times n}$ that minimizes the aggregated loss function $f(\boldsymbol{W}) = \sum_{k=1}^{K} p^{(k)} f^{(k)}(\boldsymbol{W})$, where $p^{(k)}$ is the weight of client k.

Fedeated low-rank update algorithm Our theoretical analysis suggests that local Hessians exhibit a more complex loss landscape, which can lead to potential gradient divergence across local clients. Low-rank updates help align client updates along shared directions, thereby reducing client discrepancies. Building on this insight, FedLoRU constraints clients' local updates to low-rank to enhance the communication efficiency and improve client alignment.

Analogous to the LoRA (Hu et al., 2021) approach[1], at each iteration, client k holds a frozen local copy of the global model $\boldsymbol{W}$ and performs local training to find low-rank matrices $\boldsymbol{A}_t^{(k)} \in \mathbb{R}^{m\times r}$ and $\boldsymbol{B}_t^{(k)} \in \mathbb{R}^{r\times n}$ by solving:

$$\boldsymbol{A}_t^{(k)},\ \boldsymbol{B}_t^{(k)} = \underset{\boldsymbol{A},\ \boldsymbol{B}}{\arg\min}\, f^{(k)}(\boldsymbol{W} + \alpha \boldsymbol{A}\boldsymbol{B}) \qquad (12)$$

where α is a fixed scaling hyperparameter. At the end of each iteration, the server collects $\boldsymbol{A}_t^{(k)}$ and $\boldsymbol{B}_t^{(k)}$ and aggregates them by averaging: $\boldsymbol{A}_t = \sum_{k\in\mathcal{K}_M} p^{(k)} \boldsymbol{A}_t^{(k)}$, $\boldsymbol{B}_t = \sum_{k\in\mathcal{K}_M} p^{(k)} \boldsymbol{B}_t^{(k)}$ where $\mathcal{K}_M$ is the set of participating clients. After the aggregation, the server broadcasts $\boldsymbol{A}_t$ and $\boldsymbol{B}_t$ to the clients, who continue local training using these matrices as starting $\boldsymbol{A}$ and $\boldsymbol{B}$.

Unlike LoRA, FedLoRU periodically accumulates low-rank updates into the global model after aggregation to achieve a higher-rank global model. Clients subsequently update their local copies of the global model by $\boldsymbol{W} \leftarrow \boldsymbol{W} + \alpha \boldsymbol{A}_t \boldsymbol{B}_t$. When low-rank updates are accumulated every τ rounds from the initial global model $\boldsymbol{W}$, the final global model at round T is $\boldsymbol{W}_T = \boldsymbol{W} + \sum_{t=1,\ t \bmod \tau=0}^{T} \boldsymbol{A}_t \boldsymbol{B}_t$.

We average each matrix $\boldsymbol{A}$ and $\boldsymbol{B}$ individually, but acknowledge that alternative low-rank approaches, such as freezing one factor or alternating updates, may offer different mathematical justifications. In practice, however, we have found that our chosen scheme is the most effective among them. Furthermore, since our primary objective is to demonstrate the practicality and implicit regularization effect of low-rank updates, we defer a deeper investigation of these alternatives to future work.

FedLoRU for Fine-tuning For fine-tuning tasks, FedLoRU retains a series of low-rank matrices alongside the frozen pre-trained model. Although storing multiple low-rank matrices requires more memory than storing a single matrix, their size remains significantly smaller than that of the original model. This enables a modular, plug-and-play approach where low-rank matrices can be easily integrated with the pre-trained model. Consequently, FedLoRU maintains the same level of flexibility and extensibility as LoRA. The detailed fine-tuning algorithm is provided in the Appendix B.1.

Practical Advantages FedLoRU enables training a higher-rank global model alongside low-rank local updates. With each accumulation of low-rank update matrices, the global model's rank is incrementally enhanced, enabling the initiation of new learning phases. Moreover, by constraining updates to a low-rank subspace, FedLoRU implicitly regularizes local training, aligning local updates along major directions and reducing client divergence. Such regularization addresses one of the most significant challenges in federated learning: performance degradation when scaling to many clients.

FedLoRU also reduces communication overhead from

[1]While we use a low-rank factorized model, alternatives like LoKr (Edalati et al., 2022) or LoHa (Hyeon-Woo et al., 2021) can be employed, differing only in the factorization scheme but based on the same principles.

Table 1: Top-1 test accuracy comparison with different communication-efficient federated learning methods under various FL settings. The parameter ratio refers to the proportion of trainable parameters in the model compared to the full-rank model used in FedAvg and it implies the rank.

(a) Fashion-MNIST

Setting	IID - #clients=20			IID - #clients=100			NonIID - #clients=20		
Param Ratio	**44%**	**33%**	**22%**	**44%**	**33%**	**22%**	**44%**	**33%**	**22%**
FedLoRA	91.22	90.29	90.15	88.63	88.14	88.01	73.89	74.00	73.19
FedHM	91.16	91.10	**90.94**	**89.43**	**89.37**	**88.86**	85.15	**85.45**	**85.33**
FedLoRU	**91.25**	**91.16**	90.59	89.01	88.88	88.37	**85.33**	80.02	80.17

(b) CIFAR-10

Setting	IID - #clients=20			IID - #clients=100			NonIID - #clients=20		
Param Ratio	**41%**	**31%**	**21%**	**41%**	**31%**	**21%**	**41%**	**31%**	**21%**
FedLoRA	91.65	88.96	89.35	79.48	85.71	85.06	69.60	66.13	67.61
FedHM	90.76	90.32	90.77	81.41	81.58	82.12	70.55	66.39	65.48
FedLoRU	**92.43**	**90.71**	**90.85**	**81.46**	**86.01**	**86.10**	**75.19**	**69.71**	**67.88**

(c) CIFAR-100

Setting	IID - #clients=20			IID - #clients=100			NonIID - #clients=20		
Param Ratio	**41%**	**31%**	**21%**	**41%**	**31%**	**21%**	**41%**	**31%**	**21%**
FedLoRA	65.53	57.36	55.14	53.79	52.20	51.20	14.41	10.58	12.97
FedHM	59.43	58.40	58.52	43.35	41.84	41.62	**16.88**	15.04	14.13
FedLoRU	**66.81**	**60.78**	**61.42**	**57.96**	**53.25**	**53.53**	16.46	**15.70**	**14.52**

Kmn to $Kr(m+n)$ when $r \ll m$ or $r \ll n$. Additionally, since no compression process is involved, there is no additional computation compared to conventional compression-based communication-efficient federated learning algorithms.

5. Experiments

In this section, we extensively evaluate FedLoRU in both pre-training and fine-tuning and demonstrate the benefits of low-rank updates in scenarios with a large number of clients. We first provide the experiment setup, then move on to the performance evaluation.

5.1. Experiment setup

Datasets and Baseline Algorithms We evaluate our proposed algorithms on four datasets: Fashion MNIST (Xiao et al., 2017), CIFAR-10, CIFAR-100 (Krizhevsky & Hinton, 2009), and Alpaca (Taori et al., 2023). ResNet-10 and ResNet-18 (He et al., 2016) are used for the image datasets, and LLaMA2-3B (Touvron et al., 2023) is used for fine-tuning on Alpaca. For the image datasets, we allocated 10,000 samples each for the validation and test sets, while for the Alpaca dataset, we partitioned the data into training, validation, and test sets consisting of 48,000, 2,000, and 2,000 samples, respectively. We compare FedLoRU with several benchmarks: FedAvg (McMahan et al., 2017), the standard federated learning algorithm that trains full-rank models; FedLoRA (Zhang et al., 2023), which trains low-rank modules without accumulating low-rank updates; and FedHM (Yao et al., 2021), the prior state-of-the-art in communication-efficient federated learning.

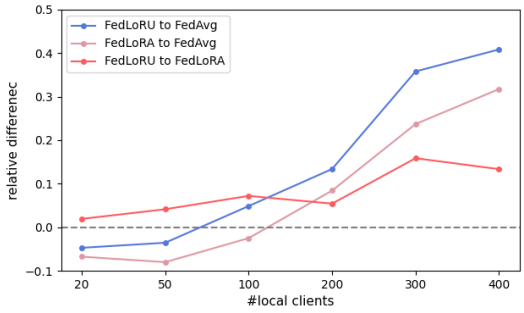


Figure 2: The relative difference in test accuracy between two algorithms is measured by the number of clients. The relative difference of Alg_1 to Alg_2 is defined as $\frac{\text{Alg}_1 - \text{Alg}_2}{\text{Alg}_1}$. For detailed numbers, see Table 6.

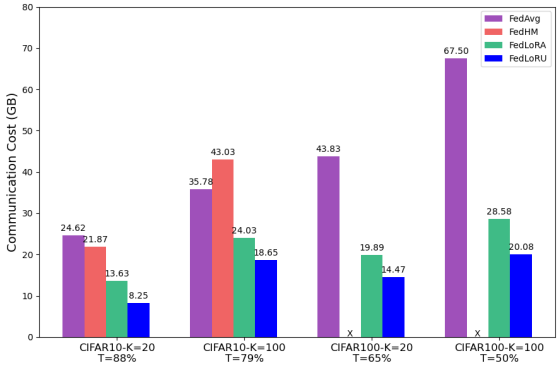


Figure 3: Evaluating different low-rank federated learning algorithms in terms of the communication cost to achieve target test accuracy. Here, "X" indicates that the algorithm did not reach the target accuracy.

Implementation During pre-training on the image datasets, we vary the number of clients between 20 and 400, sampling 50% of clients per round, as is standard in the FL literature, with each client training for 5 local epochs. For fine-tuning the language model, we use 10 clients with a 50% participation and 1 local epoch. The selection of local epochs balances the trade-off between communication overhead and potential performance degradation. Learning rates and accumulation cycles are selected via grid search, and different rank configurations are tested for FedHM, FedLoRA, and FedLoRU. In fact, while we use FedAvg as the training scheme, FedLoRU techniques can be easily integrated into other federated learning schemes such as FedAdam and FedAdagrad (Reddi et al., 2020). Model parameters are initialized following LoRA best practices, Kaiming initialization (He et al., 2015) for $\boldsymbol{A}$-matrix, and zeros for $\boldsymbol{B}$-matrix. For full details of the implementation,

including the selection of parameters such as α, τ, and T, as well as their sensitivity, see Appendix C. We run each setting 3 times and the numbers reported in the tables are averages with very low standard deviation (¡ 0.005).

In the statistically heterogeneous setting, we generate disjoint non-IID client data using a Dirichlet distribution, Dir(ψ), with a concentration parameter ψ set to 0.5, as described in (Hsu et al., 2019).

5.2. Performance Evaluation

Performance of Pre-training We evaluate the Top-1 accuracy of models with varying parameter sizes in both IID and Non-IID scenarios across different federated learning configurations. Table 1 shows the performance of FedLoRU and baseline algorithms. The standard deviation for each setting is relatively small in the IID scenario, with a maximum value of 0.382. In contrast, the non-IID setting exhibits a relatively higher standard deviation, with a maximum of 0.969. However, these variations do not impact the overall comparison between the algorithms.

In our experimental evaluation, FedLoRU consistently achieves competitive or superior accuracy compared to FedAvg, whose results can be found in Appendix E. Although FedLoRU's accuracy is slightly lower than FedAvg's in most settings, the difference is minimal given the significant reduction in parameters, with at most a 5% decrease and typically only a 1-2% difference. Notably, in the CIFAR-10 and CIFAR-100 IID settings with 100 clients, FedLoRU surpasses FedAvg. Overall, FedLoRU achieves the best accuracy in 20 out of 27 cases and demonstrates improvements over FedHM ranging from -6% to 33.7%. Additionally, FedLoRU consistently outperforms FedLoRA, highlighting the effectiveness of accumulating low-rank updates. The client regularization effect of FedLoRU, as predicted by our theoretical analysis, suggests that using client-side low-rank updates is particularly beneficial in environments with a large number of clients. This benefit is evident in experiments under IID conditions with 100 clients, where FedLoRU attains the highest accuracy among the tested methods.

Scalability and Performance of FedLoRU in Large-Client Federated Learning Table 6 and Figure 2 compare FedAvg and FedLoRU across varing number of clients. As the number of clients increases, the scalability of the algorithm becomes a crucial factor. Our experiments show a sharp decline in FedAvg's performance, demonstrating its difficulty in maintaining accuracy as the number of clients grows.

In contrast, FedLoRU and FedLoRA outperform FedAvg when the number of clients exceeds 100 and 200, respectively. This trend is further reinforced in settings with a lower participation ratio, as shown in Table 7. Furthermore, the performance gap between low-rank algorithms and FedAvg continues to expand as K increases. These findings emphasize that constraining updates to a low-rank subspace is particularly beneficial in federated learning environments with a large number of clients, and FedLoRU provides the most effective strategy among the compared low-rank approaches.

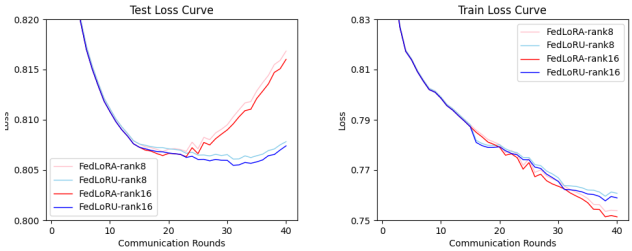


Figure 4: Loss curve of FedLoRU and FedLoRA for fine-tuning LLaMA2-3B.

Performance of LLM Fine-tuning Figure 4 presents the loss curves of FedLoRA and FedLoRU during fine-tuning of the LLaMA2-3B model on the Alpaca dataset. The train loss curves show that both algorithms achieve similar convergence rates, with minimal differences in training optimization. However, a notable distinction emerges in the test loss results, where FedLoRU consistently outperforms FedLoRA after the 25th communication round.

In this fine-tuning experiment, we accumulate the results every 15 communication rounds. Notably, despite FedLoRU performing an additional accumulation at round 30, the test loss does not show any further improvement. This suggests that beyond a certain point, further accumulation may not necessarily enhance the model's generalization performance.

6. Conclusion

In this paper, we theoretically show that client-side optimization exhibits a higher-rank structure compared to server-side optimization and hypothesize that using low-rank updates in client-side optimization can promote an implicit regularization effect across clients. We are the first to establish a theoretical foundation supporting the use of low-rank updates in federated learning. Our proposed algorithm, FedLoRU, achieves comparable performance to FedAvg while significantly reducing the number of communicated parameters. Moreover, as the number of clients increases, FedLoRU consistently outperforms FedAvg, highlighting its scalability and effectiveness in large-scale federated learning environments.

Impact Statement

This paper presents work whose goal is to advance the field of Machine Learning. There are many potential societal consequences of our work, none which we feel must be specifically highlighted here.

References

Anderson, G. W., Guionnet, A., and Zeitouni, O. *An introduction to random matrices*. Number 118. Cambridge university press, 2010.

Antti Knowles, J. Y. The isotropic semicircle law and deformation of wigner matrices. *Communications on Pure and Applied Mathematics*, 66(11):1663–1749, 2013.

Azam, S. S., Hosseinalipour, S., Qiu, Q., and Brinton, C. Recycling model updates in federated learning: Are gradient subspaces low-rank? In *International Conference on Learning Representations*, 2021.

Baskerville, N. P., Keating, J. P., Mezzadri, F., Najnudel, J., and Granziol, D. Universal characteristics of deep neural network loss surfaces from random matrix theory. *Journal of Physics A: Mathematical and Theoretical*, 55 (49):494002, 2022.

Benaych-Georges, F. and Nadakuditi, R. R. The eigenvalues and eigenvectors of finite, low rank perturbations of large random matrices. *Advances in Mathematics*, 227(1):494–521, 2011.

Benaych-Georges, F. and Nadakuditi, R. R. The singular values and vectors of low rank perturbations of large rectangular random matrices. *Journal of Multivariate Analysis*, 111:120–135, 2012.

Brody, T. A., Flores, J., French, J. B., Mello, P., Pandey, A., and Wong, S. S. Random-matrix physics: Spectrum and strength fluctuations. *Reviews of Modern Physics*, 53(3): 385, 1981.

Caldas, S., Konečny, J., McMahan, H. B., and Talwalkar, A. Expanding the reach of federated learning by reducing client resource requirements. *arXiv preprint arXiv:1812.07210*, 2018.

Capitaine, M. Additive/multiplicative free subordination property and limiting eigenvectors of spiked additive deformations of wigner matrices and spiked sample covariance matrices. *Journal of Theoretical Probability*, 26: 595–648, 2013.

Chen, Y., Cheng, C., and Fan, J. Asymmetry helps: Eigenvalue and eigenvector analyses of asymmetrically perturbed low-rank matrices. *Annals of Statistics*, 49(1):435, 2021.

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.

Dettmers, T., Pagnoni, A., Holtzman, A., and Zettlemoyer, L. QLoRA: Efficient finetuning of quantized LLMs. *Advances in Neural Information Processing Systems*, 36, 2024.

Edalati, A., Tahaei, M., Kobyzev, I., Nia, V. P., Clark, J. J., and Rezagholizadeh, M. Krona: Parameter efficient tuning with kronecker adapter. *arXiv preprint arXiv:2212.10650*, 2022.

Granziol, D., Zohren, S., and Roberts, S. Learning rates as a function of batch size: A random matrix theory approach to neural network training. *Journal of Machine Learning Research*, 23(173):1–65, 2022.

Guhr, T., Müller-Groeling, A., and Weidenmüller, H. A. Random-matrix theories in quantum physics: Common concepts. *Physics Reports*, 299(4-6):189–425, 1998.

Gur-Ari, G., Roberts, D. A., and Dyer, E. Gradient descent happens in a tiny subspace. *arXiv preprint arXiv:1812.04754*, 2018.

He, K., Zhang, X., Ren, S., and Sun, J. Delving deep into rectifiers: Surpassing human-level performance on imagenet classification. In *Proceedings of the IEEE International Conference on Computer Vision*, pp. 1026–1034, 2015.

He, K., Zhang, X., Ren, S., and Sun, J. Deep residual learning for image recognition. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, pp. 770–778, 2016.

Hsu, T.-M. H., Qi, H., and Brown, M. Measuring the effects of non-identical data distribution for federated visual classification. *arXiv preprint arXiv:1909.06335*, 2019.

Hu, E. J., Shen, Y., Wallis, P., Allen-Zhu, Z., Li, Y., Wang, S., Wang, L., and Chen, W. LoRA: Low-rank adaptation of large language models. *arXiv preprint arXiv:2106.09685*, 2021.

Huh, M., Mobahi, H., Zhang, R., Cheung, B., Agrawal, P., and Isola, P. The low-rank simplicity bias in deep networks. *arXiv preprint arXiv:2103.10427*, 2021.

Hyeon-Woo, N., Ye-Bin, M., and Oh, T.-H. Fedpara: Low-rank hadamard product for communication-efficient federated learning. *arXiv preprint arXiv:2108.06098*, 2021.

Jadbabaie, A., Makur, A., and Shah, D. Federated optimization of smooth loss functions. *IEEE Transactions on Information Theory*, 2023.

Jhunjhunwala, D., Gadhikar, A., Joshi, G., and Eldar, Y. C. Adaptive quantization of model updates for communication-efficient federated learning. In *ICASSP IEEE International Conference on Acoustics, Speech and Signal Processing*, pp. 3110–3114. IEEE, 2021.

Ji, Z. and Telgarsky, M. Gradient descent aligns the layers of deep linear networks. *arXiv preprint arXiv:1810.02032*, 2018.

Jiang, Z., Xu, Y., Xu, H., Wang, Z., Liu, J., Chen, Q., and Qiao, C. Computation and communication efficient federated learning with adaptive model pruning. *IEEE Transactions on Mobile Computing*, 23(3):2003–2021, 2023.

Kairouz, P., McMahan, H. B., Avent, B., Bellet, A., Bennis, M., Bhagoji, A. N., Bonawitz, K., Charles, Z., Cormode, G., Cummings, R., et al. Advances and open problems in federated learning. *Foundations and Trends in Machine Learning*, 14(1–2):1–210, 2021.

Krizhevsky, A. and Hinton, G. Learning multiple layers of features from tiny images. *https://www. cs. toronto. edu/kriz/learning-features-2009-TR. pdf.*, 2009.

Kuo, K., Raje, A., Rajesh, K., and Smith, V. Federated LoRA with sparse communication. *arXiv preprint arXiv:2406.05233*, 2024.

Li, C., Farkhoor, H., Liu, R., and Yosinski, J. Measuring the intrinsic dimension of objective landscapes. *arXiv preprint arXiv:1804.08838*, 2018.

Lialin, V., Muckatira, S., Shivagunde, N., and Rumshisky, A. ReLoRA: High-rank training through low-rank updates. In *The Twelfth International Conference on Learning Representations*, 2023.

Loshchilov, I. and Hutter, F. Decoupled weight decay regularization. *arXiv preprint arXiv:1711.05101*, 2017.

Lu, Q., Niu, D., Khoshkho, M. S., and Li, B. Hyperflora: Federated learning with instantaneous personalization. In *Proceedings of the 2024 SIAM International Conference on Data Mining*, pp. 824–832. SIAM, 2024.

Mangrulkar, S., Gugger, S., Debut, L., Belkada, Y., Paul, S., and Bossan, B. Peft: State-of-the-art parameter-efficient fine-tuning methods. https://github.com/huggingface/peft, 2022.

McMahan, B., Moore, E., Ramage, D., Hampson, S., and y Arcas, B. A. Communication-efficient learning of deep networks from decentralized data. In *Artificial Intelligence and Statistics*, pp. 1273–1282. PMLR, 2017.

Péché, S. The largest eigenvalue of small rank perturbations of hermitian random matrices. *Probability Theory and Related Fields*, 134:127–173, 2006.

Pielaszkiewicz, J. and Singull, M. Closed form of the asymptotic spectral distribution of random matrices using free independence. *Linkoping University Electronic Press*, 2015.

Qiao, Z., Yu, X., Zhang, J., and Letaief, K. B. Communication-efficient federated learning with dual-side low-rank compression. *arXiv preprint arXiv:2104.12416*, 2021.

Reddi, S., Charles, Z., Zaheer, M., Garrett, Z., Rush, K., Konečnỳ, J., Kumar, S., and McMahan, H. B. Adaptive federated optimization. *arXiv preprint arXiv:2003.00295*, 2020.

Reisizadeh, A., Mokhtari, A., Hassani, H., Jadbabaie, A., and Pedarsani, R. FedPAQ: A communication-efficient federated learning method with periodic averaging and quantization. In *International Conference on Artificial Intelligence and Statistics*, pp. 2021–2031. PMLR, 2020.

Sabanayagam, M., Behrens, F., Adomaityte, U., and Dawid, A. Unveiling the hessian's connection to the decision boundary. *arXiv preprint arXiv:2306.07104*, 2023.

Sagun, L., Bottou, L., and LeCun, Y. Eigenvalues of the hessian in deep learning: Singularity and beyond. *arXiv preprint arXiv:1611.07476*, 2016.

Shahid, O., Pouriyeh, S., Parizi, R. M., Sheng, Q. Z., Srivastava, G., and Zhao, L. Communication efficiency in federated learning: Achievements and challenges. *arXiv preprint arXiv:2107.10996*, 2021.

Sun, Y., Li, Z., Li, Y., and Ding, B. Improving LoRA in privacy-preserving federated learning. *arXiv preprint arXiv:2403.12313*, 2024.

Taori, R., Gulrajani, I., Zhang, T., Dubois, Y., Li, X., Guestrin, C., Liang, P., and Hashimoto, T. B. Stanford Alpaca: An instruction-following LLaMA model. https://github.com/tatsu-lab/stanford_alpaca, 2023.

Touvron, H., Martin, L., Stone, K., Albert, P., Almahairi, A., Babaei, Y., Bashlykov, N., Batra, S., Bhargava, P., Bhosale, S., et al. Llama 2: Open foundation and fine-tuned chat models. *arXiv preprint arXiv:2307.09288*, 2023.

Tulino, A. M., Verdú, S., et al. Random matrix theory and wireless communications. *Foundations and Trends in Communications and Information Theory*, 1(1):1–182, 2004.

Valipour, M., Rezagholizadeh, M., Kobyzev, I., and Ghodsi, A. Dylora: Parameter efficient tuning of pre-trained models using dynamic search-free low-rank adaptation. *arXiv preprint arXiv:2210.07558*, 2022.

Wu, X., Liu, X., Niu, J., Wang, H., Tang, S., and Zhu, G. FedLoRA: When personalized federated learning meets low-rank adaptation. https://openreview.net/forum?id=bZh06ptG9r, 2024.

Xiao, H., Rasul, K., and Vollgraf, R. Fashion-MNIST: a novel image dataset for benchmarking machine learning algorithms. *arXiv preprint arXiv:1708.07747*, 2017.

Yao, D., Pan, W., O'Neill, M. J., Dai, Y., Wan, Y., Jin, H., and Sun, L. Fedhm: Efficient federated learning for heterogeneous models via low-rank factorization. *arXiv preprint arXiv:2111.14655*, 2021.

Yao, Z., Gholami, A., Keutzer, K., and Mahoney, M. W. Pyhessian: Neural networks through the lens of the hessian. In *2020 IEEE International Conference on Big Data*, pp. 581–590. IEEE, 2020.

Ye, M., Fang, X., Du, B., Yuen, P. C., and Tao, D. Heterogeneous federated learning: State-of-the-art and research challenges. *ACM Computing Surveys*, 56(3):1–44, 2023.

Ye, R., Wang, W., Chai, J., Li, D., Li, Z., Xu, Y., Du, Y., Wang, Y., and Chen, S. OpenFedLLM: Training large language models on decentralized private data via federated learning. *arXiv preprint arXiv:2402.06954*, 2024.

Yi, L., Yu, H., Wang, G., and Liu, X. Fedlora: Model-heterogeneous personalized federated learning with lora tuning. *arXiv preprint arXiv:2310.13283*, 2023.

Yu, H. and Wu, J. Compressing transformers: features are low-rank, but weights are not! In *Proceedings of the AAAI Conference on Artificial Intelligence*, volume 37, pp. 11007–11015, 2023.

Zhang, Z., Yang, Y., Dai, Y., Wang, Q., Yu, Y., Qu, L., and Xu, Z. Fedpetuning: When federated learning meets the parameter-efficient tuning methods of pre-trained language models. In *Annual Meeting of the Association of Computational Linguistics 2023*, pp. 9963–9977. Association for Computational Linguistics (ACL), 2023.

Zhao, J., Zhang, Z., Chen, B., Wang, Z., Anandkumar, A., and Tian, Y. GaLore: Memory-efficient LLM training by gradient low-rank projection. *arXiv preprint arXiv:2403.03507*, 2024.

Zheng, S., Shen, C., and Chen, X. Design and analysis of uplink and downlink communications for federated learning. *IEEE Journal on Selected Areas in Communications*, 39(7):2150–2167, 2020.

A. Proof of the Main Theorems

In this section, we provide proofs of Proposition 3.1, Theorem 3.2, and Proposition A.4. We begin by presenting some lemmas that are required for our analysis and then proceed to prove the propositions and the theorem.

Lemma A.1 (Theorem 2.2 from (Pielaszkiewicz & Singull, 2015))**.** *Let μ_n be a sequence of probability measures on $\mathbb{R}$ and let g_{μ_n} denote the Stieltjes transform of μ_n. We have*

a) if $\mu_n \to \mu$ weakly, where μ is a measure on $\mathbb{R}$, then $g_{\mu_n}(z) \to g_\mu(z)$ pointwise for any $z \in \{z \in \mathbb{C} : \ z = u+iv, v > 0\}$

b) if $g_{\mu_n}(z) \to g(z)$ pointwise, for all $z \in \{z \in \mathbb{C} : \ z = u + iv, v > 0\}$, then there exists a unique non-negative and finite measure such that $g = g_\mu$ and $\mu_n \to \mu$ weakly.

Lemma A.2 (cf. (Capitaine, 2013))**.** *Let $\boldsymbol{X}_N$ be an $N \times N$ random real-symmetric Wigner matrix, and let $\boldsymbol{D}$ be a $N \times N$ deterministic symmetric matrix with uniformly bounded operator norm $\|\boldsymbol{D}\|$ in N. Let $\hat{\mu}_{\boldsymbol{X}}$, $\hat{\mu}_{\boldsymbol{D}}$ be the empirical spectral measures of the sequence of matrices $\boldsymbol{X}, \boldsymbol{D}$ and assume there exist deterministic limit measures $\mu_{\boldsymbol{X}}$, $\mu_{\boldsymbol{D}}$. Then $\boldsymbol{H} = \boldsymbol{X} + \boldsymbol{D}$ has a limiting spectral measure and is given by the free convolution $\mu_{\boldsymbol{X}} \boxplus \mu_{\boldsymbol{D}}$.*

Lemma A.2 states that in our additive perturbed model $\boldsymbol{H}_N(h^R, \omega^R) = \boldsymbol{H}_{\text{true}}(h^R, \omega^R) + \epsilon_N^R$, the matrix $\boldsymbol{H}_N(h^R, \omega^R)$ has a limiting spectral measure given by the free additive convolution $\mu_\nu \boxplus \mu_{\epsilon_N^R}$, where μ_ν is the limiting spectral measure of $\boldsymbol{H}_{\text{true}}(h^R, \omega^R)$ and $\mu_{\epsilon_N^R}$ corresponds to the limiting spectral measure of ϵ_N^R. The subsequent lemma, Weyl's inequality, examines the changes to eigenvalues of an Hermitian matrix that is perturbed.

Lemma A.3 (Weyl's inequality)**.** *For Hermitian matrices $\boldsymbol{A}, \boldsymbol{B} \in \mathbb{C}^{n\times n}$ and $i, j \in \{1, 2, \cdots, n\}$,*

$$\lambda_{i+j-1}(\boldsymbol{A} + \boldsymbol{B}) \leq \lambda_i(\boldsymbol{A}) + \lambda_j(\boldsymbol{B}), \quad i + j \leq n + 1, \tag{13}$$

$$\lambda_{i+j-n}(\boldsymbol{A} + \boldsymbol{B}) \geq \lambda_i(\boldsymbol{A}) + \lambda_j(\boldsymbol{B}), \quad i + j \geq n + 1, \tag{14}$$

where $\lambda_i(\boldsymbol{D})$ is i-th eigenvalue of $\boldsymbol{D}$.

A.1. Proof of the Richness of $\Omega(\theta_1, \cdots, \theta_k)$.

In our theoretical analysis, we show the difference in stable rank between the Hessians of a server and a client eventually becomes positive as dimension R approaches infinity within the space of $\Omega(\theta_1, \cdots, \theta_k)$. In this section, we discuss about the richness of $\Omega(\theta_1, \cdots, \theta_k)$ and characteristics of $\Omega^R(\theta_1, \cdots, \theta_k)$. They are defined as:

$$\Omega^R(\theta_1, \cdots, \theta_k) = \{(h^R, \omega^R) : \boldsymbol{H}_{\text{true}}(h^R, \omega^R) \text{ has non-zero eigenvalues } \theta_1, \cdots, \theta_k\}, \tag{15}$$

$$\Omega(\theta_1, \cdots, \theta_k) = \bigcup_R \Omega^R(\theta_1, \cdots, \theta_k). \tag{16}$$

In fact, the set of all possible pairs (h^R, ω^R) is represented by the union over all dimensions R, integers $k \leq R$, and non-zero real values $\theta_1, \cdots, \theta_k$ as follows:

$$\bigcup_{R=1}^{\infty} \{(h^R, \omega^R) : \text{ any pair } (h^R, \omega^R) \text{ of dimension } R\} = \bigcup_{R=1}^{\infty} \bigcup_{k=1}^{R} \bigcup_{(\theta_1, \cdots, \theta_k) \in \mathbb{R}^k} \Omega^R(\theta_1, \cdots, \theta_k).$$

Thus, for any given pair (h^R, ω^R), there exist $\theta_1, \cdots, \theta_k$ such that $(h^R, \omega^R) \in \Omega^R(\theta_1, \cdots, \theta_k)$. According to the following proposition, either the set $\Omega(\theta_1, \cdots, \theta_k)$ is empty or there exist infinitely many values of R for which $\Omega^R(\theta_1, \cdots, \theta_k) \neq \emptyset$.

Proposition A.4. *Let $\theta_1, \cdots, \theta_k$ be fixed non-zero real numbers, and suppose there exists $\tilde{R} > k$ such that $\Omega^{\tilde{R}}(\theta_1, \cdots, \theta_k)$ is non-empty. Then $\Omega^R(\theta_1, \cdots, \theta_k)$ is non-empty for all $R \geq \tilde{R}$.*

Proof. To establish the proposition, it suffices to demonstrate that $\Omega^{\tilde{R}}(\theta_1, \cdots, \theta_k) \neq \emptyset$ implies $\Omega^{\tilde{R}+1}(\theta_1, \cdots, \theta_k) \neq \emptyset$. To this end, let $(h^{\tilde{R}}, \omega^{\tilde{R}}) \in \Omega^{\tilde{R}}(\theta_1, \cdots, \theta_k)$. Our objective is to show that there exists $(h^{\tilde{R}+1}, \omega^{\tilde{R}+1}) \in \Omega^{\tilde{R}+1}(\theta_1, \cdots, \theta_k)$.

To construct a prediction function $h^{\tilde{R}+1}$ and a weight $\omega^{\tilde{R}+1}$ of dimension $\tilde{R}+1$ such that the true Hessian retains the same non-zero eigenvalues, we define $h^{\tilde{R}+1}: \mathbb{R}^{d_x} \times \mathbb{R}^{\tilde{R}+1} \to \mathbb{R}^{d_y}$ and $\omega^{\tilde{R}+1} \in \mathbb{R}^{\tilde{R}+1}$ as

$$h^{\tilde{R}+1}(x;\omega) = \tilde{h}^{\tilde{R}+1}(x;\omega), \quad \forall x \in \mathbb{R}^{d_x}, \forall \omega \in \mathbb{R}^{\tilde{R}+1}, \tag{17}$$

$$\omega^{\tilde{R}+1} = (\omega^{\tilde{R}}, 0) \tag{18}$$

where $\tilde{h}^{\tilde{R}+1}: \mathbb{R}^{d_x} \times \mathbb{R}^{\tilde{R}+1} \to \mathbb{R}^{d_y}$ is defined as

$$\tilde{h}^{\tilde{R}+1}(x;\omega^{\tilde{R}+1}) = \tilde{h}^{\tilde{R}+1}(x;(\omega^{\tilde{R}},0)) = h^{\tilde{R}}(x;\omega^{\tilde{R}}) \tag{19}$$

which is independent of the last variable $z \in \mathbb{R}$ for all $\omega \in \mathbb{R}^R$. Expanding the Hessian of the loss function at $\omega^{\tilde{R}+1}$ for any $(x,y) \sim \psi$, we obtain

$$\begin{aligned}\nabla^2_\omega \ell(h^{\tilde{R}+1}(x;\omega^{\tilde{R}+1}),y) &= \mathbf{J}_\omega(h^{\tilde{R}+1}(x;\omega^{\tilde{R}+1}))^T \nabla^2_y \ell(h^{\tilde{R}+1}(x;\omega^{\tilde{R}+1}),y) \mathbf{J}_\omega(h^{\tilde{R}+1}(x;\omega^{\tilde{R}+1})) \\ &\quad + \sum_{i=1}^{d_y} \frac{\partial \ell}{\partial y_i}(h^{\tilde{R}+1}(x;\omega^{\tilde{R}+1}),y) \cdot \nabla^2_\omega h_i^{\tilde{R}+1}(x;\omega^{\tilde{R}+1})\end{aligned} \tag{20}$$

where $h^{\tilde{R}+1} = [h_1^{\tilde{R}+1}, \cdots, h_{d_y}^{\tilde{R}+1}]^T$ and $\mathbf{J}_\omega(h^{\tilde{R}+1}(x;\omega^{\tilde{R}+1}))$ is the Jacobian of the function $h^{\tilde{R}+1}$ with respect to $R+1$ dimensional input ω. Then, by the definition of $h^{\tilde{R}+1}$ and $\omega^{\tilde{R}+1}$, we have:

$$h^{\tilde{R}+1}(x;\omega^{\tilde{R}+1}) = h^{\tilde{R}}(x;\omega^{\tilde{R}}), \tag{21}$$

$$\mathbf{J}_\omega(h^{\tilde{R}+1}(x;\omega^{\tilde{R}+1})) = \begin{bmatrix} \mathbf{J}_\omega(h^{\tilde{R}}(x;\omega^{\tilde{R}})) & \begin{matrix}|\\0\\|\end{matrix} \end{bmatrix}, \tag{22}$$

$$\nabla^2_\omega h_i^{\tilde{R}+1}(x;\omega^{\tilde{R}+1}) = \begin{bmatrix} \nabla^2_\omega h_i^{\tilde{R}}(x;\omega^{\tilde{R}}) & \begin{matrix}|\\0\\|\end{matrix} \\ \begin{matrix}\text{——}\ 0\ \text{——}\end{matrix} & \end{bmatrix}, \ \forall i. \tag{23}$$

Substituting these expressions into the expanded Hessian equation (20), we conclude that $\nabla^2_w \ell(h^{\tilde{R}+1}(x;\omega^{\tilde{R}+1}),y)$ is identical to $\nabla^2_w \ell(h^{\tilde{R}}(x;\omega^{\tilde{R}}),y)$ for any $(x,y) \sim \psi$ except for a final zero row and column. Thus, $(h^{\tilde{R}+1},\omega^{\tilde{R}+1})$ and $(h^{\tilde{R}},\omega^{\tilde{R}})$ have the same true Hessian, except for the zero-row and the zero-column, which have no impact on the non-zero eigenvalues of the Hessian. It follows that $(h^{\tilde{R}+1},\omega^{\tilde{R}+1}) \in \Omega^{\tilde{R}+1}(\theta_1,\cdots,\theta_k)$.

For example, if we consider feedforward neural networks as prediction functions, one can easily construct a larger neural network that maintains the same non-zero eigenvalues by adding an additional neuron with a single connection to a neuron in the previous layer. This additional neuron does not affect the final output, thereby preserving the desired eigenvalue properties. □

A.2. Proof of Proposition 3.1

Numerous studies (Benaych-Georges & Nadakuditi, 2011; 2012; Chen et al., 2021; Péché, 2006) have investigated the eigenvalue behavior of perturbed matrices. In this proposition, we analyze the limiting eigenvalues of a perturbed random matrix when the perturbation is given by a Wigner matrix and the original matrix has fixed eigenvalues.

To prove Proposition 3.1, we decompose the eigenvalue analysis into two distinct parts. First, we demonstrate that the i-th eigenvalues, where $i \in \{p+1,\cdots,P-q-1\}$, converge to the upper or lower bounds of the spectral density of μ_N. Here,

μ_N is the limiting spectral density of ϵ_N^R. This portion of the proof parallels the approach employed by (Benaych-Georges & Nadakuditi, 2011). Second, we show that the remaining eigenvalues converge to the Stieltjes transformation. This part of the proof follows the methodology outlined by (Baskerville et al., 2022).

Proof. In this proof, we drop dependency on (h^R, ω^R) and simplify the notation by representing $\boldsymbol{H}_N(h^R, \omega^R)$ and $\boldsymbol{H}_{\text{true}}(h^R, \omega^R)$ as $\boldsymbol{H}_N^R$ and $\boldsymbol{H}_{\text{true}}^R$, respectively. Let us consider $\lambda_i(\boldsymbol{H}_N^R)$ for the index range $p < i < R - q$. Applying Lemma A.3, we obtain

$$\lambda_i(\boldsymbol{H}_N^R) \leq \lambda_{1+i-j}(\boldsymbol{H}_{\text{true}}^R) + \lambda_{1+i-k}(\epsilon^R(N)), \quad i = j + k - 1 \leq R,\ j,k \in \{1, \cdots, R\}, \tag{24}$$

$$\lambda_i(\boldsymbol{H}_N^R) \geq \lambda_{R+i-j}(\boldsymbol{H}_{\text{true}}^R) + \lambda_{R+i-k}(\epsilon^R(N)), \quad i = j + k - R \geq 1,\ j,k \in \{1, \cdots, R\}. \tag{25}$$

By letting $k = 1 + p$ in (24) and $k = R - q$ in (25), we derive

$$\lambda_i(\boldsymbol{H}_N^R) \leq \lambda_{1+i-j}(\boldsymbol{H}_{\text{true}}^R) + \lambda_{i-p}(\epsilon^R(N)), \quad i = j + p \leq R,\ j \in \{1, \cdots, R\}, \tag{26}$$

$$\lambda_i(\boldsymbol{H}_N^R) \geq \lambda_{R+i-j}(\boldsymbol{H}_{\text{true}}^R) + \lambda_{i+q}(\epsilon^R(N)), \quad i = j - q \geq 1,\ j \in \{1, \cdots, R\}. \tag{27}$$

By substituting $i - j = p$ in (26) and $i - j = -q$ in (27), and utilizing the facts that $\lambda_{1+p}(\boldsymbol{H}_{\text{true}}^R) = 0$ and $\lambda_{R-q}(\boldsymbol{H}_{\text{true}}^R) = 0$, we deduce

$$\lambda_{i+q}(\epsilon_N^R) \leq \lambda_i(\boldsymbol{H}_N^R) \leq \lambda_{i-p}(\epsilon_N^R), \quad \forall i \in \{1, \cdots, R\}, \tag{28}$$

where $\lambda_k(\epsilon_N^R) = -\infty$ if $k > R$, and $+\infty$ if $k \leq 0$. Additionally, since ϵ_N^R has the limiting spectral density μ_N and L_N, U_N are lower and upper bounds of μ_N, we have, for all $i \geq 1$ fixed,

$$\liminf_{R\to\infty} \lambda_i(\epsilon_N^R) \geq U_N \quad \text{and} \quad \limsup_{R\to\infty} \lambda_{R+1-i}(\epsilon_N^R) \leq L_N, \tag{29}$$

$$\lambda_1(\epsilon_N^R) \to U_N \quad \text{and} \quad \lambda_R(\epsilon_N^R) \to L_N. \tag{30}$$

From these relations, it follows that for any fixed $i \geq 1$, $\lambda_i(\epsilon_N^R)$ converges to U_N and $\lambda_{R+1-i}(\epsilon_N^R)$ converges to L_N as $R \to \infty$. By applying (29) in (28), we obtain, for all fixed $i \geq 1$,

$$\liminf_{R\to\infty} \lambda_i(\boldsymbol{H}_N^R) \geq U_N \quad \text{and} \quad \limsup_{R\to\infty} \lambda_i(\boldsymbol{H}_N^R) \leq L_N \tag{31}$$

By combining (28), (30), and (31), for all $i > p$ (respectively, $i \geq q$) fixed, we have

$$\lambda_i(\boldsymbol{H}_N^R) \to U_N \quad (\text{respectively, } \lambda_{R-i}(\boldsymbol{H}_N^R) \to L_N). \tag{32}$$

Next, we aim to prove the behavior of the remaining eigenvalues $\lambda_i(\boldsymbol{H}_N^R)$ for $i \in \{1, \cdots, p, R - q + 1, \cdots R\}$. Note that, since $p + q \ll R$ when R is sufficiently large, the limiting spectral density of H_{true}^R converges to $\nu = \delta_0$. Furthermore, because X^R is a Wigner matrix, its limiting spectral density is given by the semicircular distribution, denoted by μ.

Let us consider $\lambda_i(\boldsymbol{H}_N^R)$ where $i \leq p$ or $i \geq R - q$. According to Lemma A.2, the limiting spectral density $\mu_{\boldsymbol{H}_N^R}$ of $\boldsymbol{H}_N^R$ is given by $\mu_N \boxplus \nu$, where μ_N is the limiting spectral density of ϵ_N^R. By Lemma A.1, the Stieltjes transform $g_{\mu_{\boldsymbol{H}_N^R}}(z)$ converges pointwise to $g_{\nu \boxplus \mu_N}(z)$ for any $z \in \{z : z \in \mathbb{C},\ z = u + iv, v > 0\}$. Consequently, we have:

$$\begin{aligned} \widehat{g}_{\boldsymbol{H}_N^R}(z) &= g_{\mu_{\boldsymbol{H}_N^R}}(z) + o(1) \\ &= g_{\mu_N \boxplus \nu}(z) + o(1) \\ &= g_\nu(k(z)) + o(1) \\ &= \widehat{g}_{\boldsymbol{H}_{\text{true}}^R}(k(z)) + o(1), \end{aligned} \tag{33}$$

where k is the subordination function such that $g_{\mu_N \boxplus \nu}(z) = g_\nu(k(z))$.

Let $\lambda \in \mathbb{R}\backslash\text{supp}(\mu_N \boxplus \nu)$ be an eigenvalue of $\boldsymbol{H}_N^R$. Then $\widehat{g}_{\boldsymbol{H}_N^R}$ has a singularity at λ, and thus $\widehat{g}_{\boldsymbol{H}_{\text{true}}^R}$ must also have a singularity at $k(\lambda)$. Thus, for any R, this singularity persists, implying that $k(\lambda)$ must correspond to one of the outlier eigenvalues of $\boldsymbol{H}_N^R$. In other words, θ_i is an outlier eigenvalue of $\boldsymbol{H}_{\text{true}}^R$ if and only if there exists an eigenvalue λ of $\boldsymbol{H}_N^R$ in $\mathbb{R}\backslash\text{supp}(\mu_N \boxplus \nu)$ such that $k(\lambda) = \theta_i$. Thus, the family of the outliers of $\boldsymbol{H}_N^R$ can be expressed as

$$\{k^{-1}(\theta_j) : k^{-1}(\theta_j) \in \mathbb{R}\backslash\text{supp}(\mu_N \boxplus \nu)\}. \tag{34}$$

Note that $\text{supp}(\mu_N \boxplus \nu) = \text{supp}(\mu_N \boxplus \delta_0) = \text{supp}(\mu_N)$. Our next goal is to determine the form of $k^{-1}(\theta_j)$. From the subordination function relation, we have:

$$\begin{aligned} k^{-1}(\theta) &= g^{-1}_{\mu_N \boxplus \nu}(g_\nu(\theta)) \\ &= \mathcal{R}_{\mu_N}(g_\nu(\theta) + g_\nu^{-1}(g_\nu(\theta)) \\ &= \mathcal{R}_{\mu_N}(1/\theta) + \theta. \end{aligned} \tag{35}$$

Note that by the definition of Stieltjes transformation and $\mathcal{R}$-transform, we have $g_\nu(\theta) = g_{\delta_0}(\theta) = 1/\theta$.

Let $m_n^{(\mu)}$ denote the n-th moment of a distribution μ, and let $C_n^{(\mu)}$ denote the n-th cumulant of μ. The relationship between $m_n^{(\mu)}$ and $C_n^{(\mu)}$ is given by (Anderson et al., 2010) as

$$m_n^{(\mu)} = \sum_{r=1}^{n} \sum_{\substack{0 \le i_1, \cdots, i_r \le n-r \\ i_1 + \cdots + i_r = n-r}} C_r^{(\mu)} \left[\Pi_{j=1}^{r} m_{i_j}^{(\mu)}\right]. \tag{36}$$

Using the scaling property of moments, $m_n^{\mu_N} = s_N^n m_n^\mu$, we can derive the corresponding scaling relation for the cumulants as $C_n^{(\mu_N)} = s_N^n C_n^{(\mu)}$. Consequently, the $\mathcal{R}$-transform exhibits the scaling property

$$\mathcal{R}_{\mu_N}(\theta) = s_N \mathcal{R}_\mu(s_N \theta). \tag{37}$$

Finally, we have an expression for the outliers of $\boldsymbol{H}_N^R$ as

$$k^{-1}(\theta) = s_N \mathcal{R}_\mu(s_N/\theta) + \theta. \tag{38}$$

Since $\mathcal{R}$-transform of a semicircle law μ is given by $\mathcal{R}_\mu(x) = \sigma^2 x$, we have $k^{-1}(\theta) = \theta + \frac{\sigma^2 s_N^2}{\theta}$.

□

A.3. Proof of Theorem 3.2

Proof. Define the sets $\mathcal{P}_N = \{i \le p : g_N^{-1}(\theta_i) > U_N\} = \{i \le p : \lambda_i(\boldsymbol{H}_N^R) \to g_N^{-1}(\theta_i)\}$ and $\mathcal{Q}_N = \{i > p : g_N^{-1}(\theta_i) < L_N\} = \{i > p : \lambda_i(\boldsymbol{H}_N^R) \to g_N^{-1}(\theta_i)\}$, which represent the indices of eigenvalues $\lambda_i(\boldsymbol{H}_N^R)$ converging to $g_N^{-1}(\theta_i)$. Let $N_u = |\{i : \lambda_i(\boldsymbol{H}_N^R) \to U_N\}|$ and $N_l = |\{i : \lambda_i(\boldsymbol{H}_N^R) \to L_N\}|$ denote their cardinalities of the set of indices whose corresponding limiting eigenvalues converge to U_N and L_N, respectively. Similarly, define $\mathcal{P}_M$, $\mathcal{Q}_M$, M_u, and M_l for $\boldsymbol{H}_M^R$ analogously.

It is possible that $g_N^{-1}(\theta_i) \le U_N$ for all $i \in \{1, \cdots, p\}$ or $g_N^{-1}(\theta_i) \ge L_N$ for all $i \in \{p+1, \cdots, p+q\}$. In this case, we can just let $\mathcal{P}_N = \emptyset$ or $\mathcal{Q}_N = \emptyset$, respectively.

Define $\theta_0 = \theta_1 \cdot \mathbf{1}_{|\theta_1| \ge |\theta_{p+q}|} + \theta_{p+q} \cdot \mathbf{1}_{|\theta_1| < |\theta_{p+q}|}$ to represent the limiting eigenvalue based on the larger magnitude between θ_1 and θ_{p+q}. Using the limiting eigenvalues of $\boldsymbol{H}_N^R$, define the estimated stable rank as:

$$\hat{\text{srank}}(\boldsymbol{H}_N^R) = \sum_{j \in \mathcal{P}_N \cup \mathcal{Q}_N} \frac{g_N^{-1}(\theta_j)^2}{g_N^{-1}(\theta_0)^2} + N_u \frac{U_N^2}{g_N^{-1}(\theta_0)^2} + N_l \frac{L_N^2}{g_N^{-1}(\theta_0)^2}. \tag{39}$$

Similarly, $\hat{\text{srank}}(\boldsymbol{H}_M^R)$ is defined in the same manner. By Proposition 3.1, it follows that $\left|\text{srank}(\boldsymbol{H}_N^R) - \hat{\text{srank}}(\boldsymbol{H}_N^R)\right| \to 0$ and $\left|\text{srank}(\boldsymbol{H}_M^R) - \hat{\text{srank}}(\boldsymbol{H}_M^R)\right| \to 0$. Consequently, we have

$$\left|\big(\text{srank}(\boldsymbol{H}_M^R) - \text{srank}(\boldsymbol{H}_N^R)\big) - \big(\hat{\text{srank}}(\boldsymbol{H}_M^R) - \hat{\text{srank}}(\boldsymbol{H}_N^R)\big)\right| \to 0. \tag{40}$$

Given that $U_N < U_M$ and $L_N > L_M$, it follows that $\mathcal{P}_N \subseteq \mathcal{P}_M$ and $\mathcal{Q}_N \subseteq \mathcal{Q}_M$. Furthermore, since $U_N^2 = L_N^2$, by matching the indices in $\hat{\text{srank}}(\boldsymbol{H}_N^R)$ and $\hat{\text{srank}}(\boldsymbol{H}_M^R)$, we can express the difference between the limiting stable rank as

$$\begin{aligned}\hat{\text{srank}}(\boldsymbol{H}_M^R) - \hat{\text{srank}}(\boldsymbol{H}_N^R) = &\sum_{j\in\mathcal{P}_N\cup\mathcal{Q}_N}\left(\frac{g_M^{-1}(\theta_j)^2}{g_M^{-1}(\theta_0)^2} - \frac{g_N^{-1}(\theta_j)^2}{g_N^{-1}(\theta_0)^2}\right)\\ &+\sum_{j\in(\mathcal{P}_N^c\cap\mathcal{P}_M)\cup(\mathcal{Q}_N^c\cap\mathcal{Q}_M)}\left(\frac{g_M^{-1}(\theta_j)^2}{g_M^{-1}(\theta_0)^2} - \frac{U_N^2}{g_N^{-1}(\theta_0)^2}\right)\\ &+(M_u+M_l)\left(\frac{U_M^2}{g_M^{-1}(\theta_0)^2} - \frac{U_N^2}{g_N^{-1}(\theta_0)^2}\right).\end{aligned} \tag{41}$$

(i) We begin by showing that the first summation term, $\sum_{j\in\mathcal{P}_N\cup\mathcal{Q}_N}\left(\frac{g_M^{-1}(\theta_j)^2}{g_M^{-1}(\theta_0)^2} - \frac{g_N^{-1}(\theta_j)^2}{g_N^{-1}(\theta_0)^2}\right)$, is positive and increasing with respect to M. To achieve this, we analyze the individual term $F_j = \frac{g_M^{-1}(\theta_j)^2}{g_M^{-1}(\theta_0)^2} - \frac{g_N^{-1}(\theta_j)^2}{g_N^{-1}(\theta_0)^2}$ for $j \in \mathcal{P}_N \cup \mathcal{Q}_N$, which appears in the first summation of (41).

Expanding F_j and factoring the numerator, we have

$$\begin{aligned}F_j &= \frac{g_M^{-1}(\theta_j)^2 g_N^{-1}(\theta_0)^2 - g_N^{-1}(\theta_j)^2 g_M^{-1}(\theta_0)^2}{g_M^{-1}(\theta_0)^2 g_N^{-1}(\theta_0)^2}\\ &= \frac{\big(g_M^{-1}(\theta_j)g_N^{-1}(\theta_0) + g_N^{-1}(\theta_j)g_M^{-1}(\theta_0)\big)\big(g_M^{-1}(\theta_j)g_N^{-1}(\theta_0) - g_N^{-1}(\theta_j)g_M^{-1}(\theta_0)\big)}{g_M^{-1}(\theta_0)^2 g_N^{-1}(\theta_0)^2}\end{aligned}$$

Substituting $g_M^{-1}(\theta) = \theta + \frac{\sigma^2 s_M^2}{\theta}$ and $g_N^{-1}(\theta) = \theta + \frac{\sigma^2 s_N^2}{\theta}$ and simplifying, we can express the difference as

$$\begin{aligned}g_M^{-1}(\theta_j)g_N^{-1}(\theta_0) - g_N^{-1}(\theta_j)g_M^{-1}(\theta_0) &= \left(\theta_j + \frac{\sigma^2 s_M^2}{\theta_j}\right)\left(\theta_0 + \frac{\sigma^2 s_N^2}{\theta_0}\right) - \left(\theta_j + \frac{\sigma^2 s_N^2}{\theta_j}\right)\left(\theta_0 + \frac{\sigma^2 s_M^2}{\theta_0}\right)\\ &= \sigma^2(s_M^2 - s_N^2)\left(\frac{\theta_0}{\theta_j} - \frac{\theta_j}{\theta_0}\right).\end{aligned}$$

Thus, F_j becomes

$$F_j = \frac{g_M^{-1}(\theta_j)g_N^{-1}(\theta_0) + g_N^{-1}(\theta_j)g_M^{-1}(\theta_0)}{g_M^{-1}(\theta_0)^2 g_N^{-1}(\theta_0)^2}\cdot\sigma^2(s_M^2 - s_N^2)\left(\frac{\theta_0}{\theta_j} - \frac{\theta_j}{\theta_0}\right). \tag{42}$$

For the sign analysis, the term $g_M^{-1}(\theta_j)g_N^{-1}(\theta_0) + g_N^{-1}(\theta_j)g_M^{-1}(\theta_0)$ takes the sign of $\theta_0\theta_j$, as the sign of $g_M^{-1}(\theta)$ and $g_N^{-1}(\theta)$ are dependent of the sign of θ. The difference $s_M^2 - s_N^2$ is positive, and the term $\frac{\theta_0}{\theta_j} - \frac{\theta_j}{\theta_0}$ also has the sign of $\theta_0\theta_j$. Combining these observations, the overall sign of F_j is positive because all contributing terms either maintain a positive sign or do not introduce a sign change. Further, since $g_M^{-1}(\theta_j) \geq U_M$ for $j \in \mathcal{P}_M$ or $g_M^{-1}(\theta_j) \leq L_M$ for $j \in \mathcal{Q}_M$, a lower bound for F_j can be established as follows:

$$F_j \geq \frac{8\sigma^4 s_M s_N(s_M^2 - s_N^2)}{g_M^{-1}(\theta_0)^2 g_N^{-1}(\theta_0)^2}\cdot\left|\frac{\theta_0}{\theta_j} - \frac{\theta_j}{\theta_0}\right|. \tag{43}$$

To show that the first summation term $\sum_{j\in\mathcal{P}_N\cup\mathcal{Q}_N} F_j$ is a decreasing function with respect to M, we compute the derivative of F_j with respect to M. The derivative can be expressed as

$$\frac{\partial F_j}{\partial M} = \frac{\partial F_j}{\partial s_M}\cdot\frac{\partial s_M}{\partial M} = \frac{4\sigma^2 s_M\left(\theta_0^2-\theta_j^2\right)}{\theta_0\theta_j}\cdot\frac{g_M^{-1}(\theta_j)}{g_M^{-1}(\theta_0)^3}\cdot\frac{\partial s_M}{\partial M} \tag{44}$$

Since s_M is a decreasing function of M, it follows that $\frac{\partial s_M}{\partial M}<0$. Additionally, the term $\frac{4\sigma^2 s_M\left(\theta_0^2-\theta_j^2\right)}{\theta_0\theta_j}\cdot\frac{g_M^{-1}(\theta_j)}{g_M^{-1}(\theta_0)^3}$ is positive as same way in the sign analysis. Consequently, the product is negative, implying $\frac{\partial F_j}{\partial M}<0$. This shows that F_j decreases with M. Therefore, the first summation term, which is a sum of such $\sum_{j\in\mathcal{P}_N\cup\mathcal{Q}_N} F_j$ is a decreasing function of M.

(ii) We next show the lower bound of remaining terms $\sum_{j\in(\mathcal{P}_N^c\cap\mathcal{P}_M)\cup(\mathcal{Q}_N^c\cap\mathcal{Q}_M)}\left(\frac{g_M^{-1}(\theta_j)^2}{g_M^{-1}(\theta_0)^2}-\frac{U_N^2}{g_N^{-1}(\theta_0)^2}\right)+(M_u+M_l)\left(\frac{U_M^2}{g_M^{-1}(\theta_0)^2}-\frac{U_N^2}{g_N^{-1}(\theta_0)^2}\right)$ is positive and decreases with M. Since $g_M^{-1}(\theta_j)\geq U_M$ for $j\in(\mathcal{P}_N^c\cap\mathcal{P}_M)\cup(\mathcal{Q}_N^c\cap\mathcal{Q}_M)$, it follows that

$$\begin{aligned}
&\sum_{j\in(\mathcal{P}_N^c\cap\mathcal{P}_M)\cup(\mathcal{Q}_N^c\cap\mathcal{Q}_M)}\left(\frac{g_M^{-1}(\theta_j)^2}{g_M^{-1}(\theta_0)^2}-\frac{U_N^2}{g_N^{-1}(\theta_0)^2}\right)+(M_u+M_l)\left(\frac{U_M^2}{g_M^{-1}(\theta_0)^2}-\frac{U_N^2}{g_N^{-1}(\theta_0)^2}\right)\\
&\geq\sum_{j\in(\mathcal{P}_N^c\cap\mathcal{P}_M)\cup(\mathcal{Q}_N^c\cap\mathcal{Q}_M)}\left(\frac{U_M^2}{g_M^{-1}(\theta_0)^2}-\frac{U_N^2}{g_N^{-1}(\theta_0)^2}\right)+(M_u+M_l)\left(\frac{U_M^2}{g_M^{-1}(\theta_0)^2}-\frac{U_N^2}{g_N^{-1}(\theta_0)^2}\right)\\
&=B_N\left(\frac{U_M^2}{g_M^{-1}(\theta_0)^2}-\frac{U_N^2}{g_N^{-1}(\theta_0)^2}\right),
\end{aligned} \tag{45}$$

where $B_N=|\{i:\lambda_i(H_N^R)\to U_N \text{ or } L_N\}|$. Now consider the difference $\frac{U_M^2}{g_M^{-1}(\theta_0)^2}-\frac{U_N^2}{g_N^{-1}(\theta_0)^2}$. By expanding and simplifying, we have

$$\begin{aligned}
\frac{U_M^2}{g_M^{-1}(\theta_0)^2}-\frac{U_N^2}{g_N^{-1}(\theta_0)^2} &= \frac{U_M^2 g_N^{-1}(\theta_0)^2-U_N^2 g_M^{-1}(\theta_0)^2}{g_M^{-1}(\theta_0)^2 g_N^{-1}(\theta_0)^2}\\
&= \frac{4\sigma^2 s_M^2\left(\theta_0+\sigma^2 s_N^2/\theta_0\right)^2-4\sigma^2 s_N^2\left(\theta_0+\sigma^2 s_M^2/\theta_0\right)^2}{g_M^{-1}(\theta_0)^2 g_N^{-1}(\theta_0)^2}\\
&= \frac{4\sigma^2\left(\theta_0^2(s_M^2-s_N^2)+\sigma^4 s_M^2 s_N^2\left(s_N^2/\theta_0^2-s_M^2/\theta_0^2\right)\right)}{g_M^{-1}(\theta_0)^2 g_N^{-1}(\theta_0)^2}\\
&= \frac{4\sigma^2(s_M^2-s_N^2)\left(\theta_0^2-\sigma^4 s_M^2 s_N^2/\theta_0^2\right)}{g_M^{-1}(\theta_0)^2 g_N^{-1}(\theta_0)^2}.
\end{aligned} \tag{46}$$

Given the assumption that $\theta_0^2\geq\sigma^2 s_M^2>\sigma^2 s_M s_N$, the numerator is positive, ensuring that (46) is positive. To establish that this bound decreases with M, we compute the derivative with respect to M:

$$\begin{aligned}
\frac{\partial}{\partial M}\left(\frac{U_M^2}{g_M^{-1}(\theta_0)^2}-\frac{U_N^2}{g_N^{-1}(\theta_0)^2}\right) &= \frac{\partial}{\partial s_M}\left(\frac{U_M^2}{g_M^{-1}(\theta_0)^2}-\frac{U_N^2}{g_N^{-1}(\theta_0)^2}\right)\cdot\frac{\partial s_M}{\partial M}\\
&= \frac{2U_M(U_M)' g_M^{-1}(\theta_0)^2-2g_M^{-1}(\theta_0)(g_M^{-1}(\theta_0))' U_M^2}{g_M^{-1}(\theta_0)^4}\cdot\frac{\partial s_M}{\partial M}\\
&= \frac{4\sigma U_M g_M^{-1}(\theta_0)-4\sigma^2 s_M U_M^2 g_M^{-1}(\theta_0)/\theta_0}{g_M^{-1}(\theta_0)^4}\cdot\frac{\partial s_M}{\partial M}\\
&= \frac{4\sigma U_M(\theta_0^2-\sigma^2 s_M^2)}{\theta_0 g_M^{-1}(\theta_0)^3}\cdot\frac{\partial s_M}{\partial M}.
\end{aligned} \tag{47}$$

Since $\theta_0^2\geq\sigma^2 s_M^2$ and $\frac{\partial s_M}{\partial M}<0$, the derivative is negative, indicating that the lower bound of (45) is a decreasing function of M.

By (i) and (ii), the difference in the limiting stable rank between $\boldsymbol{H}_N^R$ and $\boldsymbol{H}_M^R$ is

$$\hat{\text{srank}}(\boldsymbol{H}_M) - \hat{\text{srank}}(\boldsymbol{H}_N) \geq \frac{s_M^2 - s_N^2}{g_M^{-1}(\theta_0)^2 g_N^{-1}(\theta_0)^2} \left[\sum_{j \in \mathcal{P}_N \cup \mathcal{Q}_N} 8\sigma^4 s_M s_N \left| \frac{\theta_0}{\theta_j} - \frac{\theta_j}{\theta_0} \right| + 4\sigma^2 B_N \left(\theta_0^2 - \frac{\sigma^4 s_M^2 s_N^2}{\theta_0^2} \right) \right], \quad (48)$$

thus it is positive and its lower bound is a decreasing function of M. □

A.4. Proof of Theorem 3.3

We rearrange the indices such that eigenvalues $\theta_1, \cdots, \theta_{p+q}$ of H_{true}^R satisfy $|\theta_1| \geq \cdots \geq |\theta_{p+q}|$. Let v_i be the unit-norm eigenvector associated with the eigenvalue θ_i of H_{true}^R, and let $u_i^{(k)}$ be the unit-norm eigenvector of $\boldsymbol{H}_N^{(k)}$ corresponding to the eigenvalue whose limiting eigenvalue is $g_N^{-1}(\theta_i)$. Define $U^{(k)}$ be the subspace spanned by $\{u_i^{(k)}\}$. For all $k \in \{1, \cdots, K\}$, the dimension of $U^{(k)}$ is identical for each client and is denoted as $\tilde{r} = |U^{(k)}|$. Additionally, let $W_N^{(k)} = \{w_i^{(k)}\}_{i=1}^{l_k}$ be remaining limiting eigenvectors of $H_N^{(k)}(h^R, \omega^R)$. The indices of $\{w_i^{(k)}\}_{i=1}^{l_k}$ are rearranged in descending order based on the magnitudes of their associated singular values. We formalize the assumption stated in the main text:

Assumption A.5. $\{w_i^{(k)}\}_{i=1}^{l_k}$ are random unit-norm orthonormal vectors such that $w_i^{(k)} \perp U_N^{(k)}, \forall i$, and the limiting value of expected directional derivative $\mathbb{E}[\partial_{w_i^{(k)}} f_N^{(k)}(h^R, \omega^R)]$ have same values for all i and k.

Define $\phi_i = \sqrt{1 - \frac{\sigma^2 s_N^2}{\theta_i^2}}$, and note that $|\langle v_i, u_i^{(k)} \rangle| \to \phi_i^2$ by (6). The following lemma provides the limiting value of the expected inner product between eigenvectors of different clients, which is used in proving Theorem 3.3.

Lemma A.6. *For any $k_1 \neq k_2 \in \{1, \cdots, K\}$ and for any i, j, the limiting value of the expected inner product between eigenvectors of different clients k_1 and k_2 is as follows:*

a) $\mathbb{E}\left[\langle u_i^{(k_1)}, u_j^{(k_2)} \rangle\right] \to \phi_i \phi_j \mathbb{1}\{i = j\}$,

b) $\mathbb{E}\left[\langle w_i^{(k_1)}, w_j^{(k_2)} \rangle\right] \to 0$,

c) $\mathbb{E}\left[\langle u_i^{(k_1)}, w_j^{(k_2)} \rangle\right] \to 0$.

Proof. Let $\phi_i^{(k)}$ be the angle between v_i and $u_i^{(k)}$. By (6), $\phi_i^{(k)} \to \phi_i$. Using this, $u_i^{(k_1)}$ and $u_j^{(k_2)}$ can be expressed as

$$u_i^{(k_1)} = \phi_i^{(k_1)} v_i + \sqrt{1 - (\phi_i^{(k_1)})^2}\, r_i^{(k_1)}, \quad (49)$$

$$u_j^{(k_2)} = \phi_j^{(k_2)} v_j + \sqrt{1 - (\phi_j^{(k_2)})^2}\, r_j^{(k_2)}, \quad (50)$$

where $r_i^{(k_1)}$ and $r_j^{(k_2)}$ are random vectors orthogonal to v_i and v_j, respectively. The inner product between $u_i^{(k_1)}$ and $u_j^{(k_2)}$ is given by

$$\begin{aligned} \langle u_i^{(k_1)}, u_j^{(k_2)} \rangle = {} & \langle \phi_i^{(k_1)} v_i, \phi_j^{(k_2)} v_j \rangle + \langle \phi_i^{(k_1)} v_i, \sqrt{1 - (\phi_j^{(k_2)})^2}\, r_j^{(k_2)} \rangle \\ & + \langle \sqrt{1 - (\phi_i^{(k_1)})^2}\, r_i^{(k_1)}, \phi_j^{(k_2)} v_j \rangle + \langle \sqrt{1 - (\phi_i^{(k_1)})^2}\, r_i^{(k_1)}, \sqrt{1 - (\phi_j^{(k_2)})^2}\, r_j^{(k_2)} \rangle. \end{aligned} \quad (51)$$

Since $r_i^{(k_1)}$ and $r_j^{(k_2)}$ are uniformly distributed on the subspaces orthogonal to v_i and v_j, respectively, all cross terms involving $r_i^{(k_1)}$ and $r_j^{(k_2)}$ average to zero as $R \to \infty$. Consequently, the expected value reduces to $\mathbb{E}\left[\langle u_i^{(k_1)}, u_j^{(k_2)} \rangle\right] \to \phi_i \phi_j \mathbb{1}\{i = j\}$.

For eigenvectors $w_i^{(k_1)}$ and $w_j^{(k_2)}$, these are independent random vectors uniformly distributed within $(U^{(k_1)})^\perp$ and $(U^{(k_2)})^\perp$, respectively, i.e., they are chosen uniformly on the sphere in $(U^{(k_1)})^\perp$ and $(U^{(k_2)})^\perp$. Due to the rotational symmetry of these spaces, the expected inner product averages to zero:

$$\mathbb{E}\left[\langle w_i^{(k_1)}, w_j^{(k_2)}\rangle\right] \to 0. \tag{52}$$

Similarly, since $w_i^{(k_1)} \perp U_N^{(k_1)}$ and $u_j^{(k_2)} \in U_N^{(k_2)}$, the expected inner product between $w_i^{(k_1)}$ and $u_j^{(k_2)}$ also averages to zero:

$$\mathbb{E}[\langle u_i^{(k_1)}, w_j^{(k_2)}\rangle] \to 0. \tag{53}$$

□

Now we provide the proof for Theorem 3.3.

Proof. Let $\alpha_i^R = \mathbb{E}[\partial_{u_i^{(k)}} f_N^{(k)}(h^R, \omega^R)]$ and $\beta = \mathbb{E}[\partial_{w_i^{(k)}} f_N^{(k)}(h^R, \omega^R)]$. Since the dataset $\mathcal{D}_N^{(k)}$, $\forall k \in \{1, \cdots, K\}$, is random and the eigenvector distributions are identical across clients, the expected values of the directional derivatives are the same for all clients. The cosine similarity between two rank-r approximations of client k_1 and k_2 is $C_{N,r}^R(k_1, k_2) = \cos\left(\nabla \hat{f}_{N,r}^{(k_1)}, \nabla \hat{f}_{N,r}^{(k_2)}\right) = \frac{\langle \nabla \hat{f}_{N,r}^{(k_1)}, \nabla \hat{f}_{N,r}^{(k_2)}\rangle}{\|\nabla \hat{f}_{N,r}^{(k_1)}\| \cdot \|\nabla \hat{f}_{N,r}^{(k_2)}\|}$.

(i) For $r < \tilde{r}$, we can write rank-r approximation of $\nabla f_N^{(k_1)}(h^R, \omega^R)$ and $\nabla f_N^{(k_2)}(h^R, \omega^R)$ as

$$\nabla \hat{f}_{N,r}^{(k_1)} = \sum_{i \le r} \partial_{u_i^{(k_1)}} f_N^{(k_1)}(h^R, \omega^R)\, u_i^{(k_1)},$$
$$\nabla \hat{f}_{N,r}^{(k_2)} = \sum_{i \le r} \partial_{u_i^{(k_2)}} f_N^{(k_2)}(h^R, \omega^R)\, u_i^{(k_2)}.$$

We drop h^R and ω^R since the context is clear. The expectation of the cosine similarity between these two rank-r approximations is

$$\mathbb{E}\left[C_{N,r}^R(k_1, k_2)\right] = \mathbb{E}\left[\frac{\langle \sum_{i \le r} \partial_{u_i^{(k_1)}} f_N^{(k_1)}\, u_i^{(k_1)},\ \sum_{i \le r} \partial_{u_i^{(k_2)}} f_N^{(k_2)}\, u_i^{(k_2)}\rangle}{\|\nabla \hat{f}_{N,r}^{(k_1)}\| \cdot \|\nabla \hat{f}_{N,r}^{(k_2)}\|}\right]. \tag{54}$$

The denominator in (59) is $\mathbb{E}\left[\|\nabla \hat{f}_{N,r}^{(k_1)}\| \cdot \|\nabla \hat{f}_{N,r}^{(k_2)}\|\right] = \sum_{i \le r} (\alpha_i^R)^2$ because of the independence between client k_1 and k_2. The numerator can be expressed as

$$\begin{aligned}
&\mathbb{E}\left[\langle \sum_{i \le r} \partial_{u_i^{(k_1)}} f_N^{(k_1)}\, u_i^{(k_1)},\ \sum_{i \le r} \partial_{u_i^{(k_2)}} f_N^{(k_2)}\, u_i^{(k_2)}\rangle\right] \\
&= \mathbb{E}\left[\sum_{i \le r} \partial_{u_i^{(k_1)}} f_N^{(k_1)}\, \partial_{u_i^{(k_2)}} f_N^{(k_2)}\, \langle u_i^{(k_1)}, u_i^{(k_2)}\rangle\right] + \mathbb{E}\left[\sum_{i \ne j \le r} \partial_{u_i^{(k_1)}} f_N^{(k_1)}\, \partial_{u_j^{(k_2)}} f_N^{(k_2)}\, \langle u_i^{(k_1)}, u_j^{(k_2)}\rangle\right]
\end{aligned} \tag{55}$$

By Lemma A.6, we know $\mathbb{E}\left[\langle u_i^{(k_1)}, u_i^{(k_2)}\rangle\right] \to \phi_i^2$ and $\mathbb{E}\left[\langle u_i^{(k_1)}, u_j^{(k_2)}\rangle\right] \to 0$ for $i \ne j$, thus the numerator satisfies

$$\left|\mathbb{E}\left[\langle \sum_{i \le r} \partial_{u_i^{(k_1)}} f_N^{(k_1)}\, u_i^{(k_1)},\ \sum_{i \le r} \partial_{u_i^{(k_2)}} f_N^{(k_2)}\, u_i^{(k_2)}\rangle\right] - \sum_{i \le r} (\alpha_i^R)^2 \phi_i^2\right| \to 0. \tag{56}$$

Therefore we have

$$\left| \mathbb{E}\left[C_{N,r}^R(k_1,k_2)\right] - \frac{\sum_{i\le r}(\alpha_i^R)^2\phi_i^2}{\sum_{i\le r}(\alpha_i^R)^2} \right| \to 0. \tag{57}$$

Finally, we have the following term

$$\left| \mathbb{E}\left[C_{N,r}^R(k_1,k_2) - C_{N,r+1}^R(k_1,k_2)\right] - \frac{(\alpha_{r+1}^R)^2\sum_{i\le r}(\alpha_i^R)^2(\phi_i^2-\phi_{r+1}^2)}{\sum_{i\le r}(\alpha_i^R)^2 \cdot \sum_{i\le r+1}(\alpha_i^R)^2} \right| \to 0 \tag{58}$$

and here $g(r) = \frac{(\alpha_{r+1}^R)^2\sum_{i\le r}(\alpha_i^R)^2(\phi_i^2-\phi_{r+1}^2)}{\sum_{i\le r}(\alpha_i^R)^2 \cdot \sum_{i\le r+1}(\alpha_i^R)^2}$ is strictly positive since $\phi_i^2 = 1 - \frac{\sigma^2 s_N^2}{\theta_i^2} \ge 1 - \frac{\sigma^2 s_N^2}{\theta_{r+1}^2} = \phi_{r+1}^2$.

(ii) For $r \ge \tilde{r}$, we can write rank-r approximation of $\nabla f_N^{(k_1)}(h^R,\omega^R)$ and $\nabla f_N^{(k_2)}(h^R,\omega^R)$ as

$$\nabla \hat{f}_{N,r}^{(k_1)} = \sum_{i\le\tilde{r}} \partial_{u_i^{(k_1)}} f_N^{(k_1)}\, u_i^{(k_1)} + \sum_{i\le r-\tilde{r}} \partial_{w_i^{(k_1)}} f_N^{(k_1)}\, w_i^{(k_1)},$$
$$\nabla \hat{f}_{N,r}^{(k_2)} = \sum_{i\le\tilde{r}} \partial_{u_i^{(k_2)}} f_N^{(k_2)}\, u_i^{(k_2)} + \sum_{i\le r-\tilde{r}} \partial_{w_i^{(k_2)}} f_N^{(k_2)}\, w_i^{(k_2)}.$$

By the same argument in (i), we have

$$\mathbb{E}\left[C_{N,r}^R(k_1,k_2)\right] = \mathbb{E}\left[\frac{\langle \sum_{i\le\tilde{r}} \partial_{u_i^{(k_1)}} f_N^{(k_1)}\, u_i^{(k_1)} + \sum_{i\le r-\tilde{r}} \partial_{w_i^{(k_1)}} f_N^{(k_1)}\, w_i^{(k_1)},\ \sum_{i\le\tilde{r}} \partial_{u_i^{(k_2)}} f_N^{(k_2)}\, u_i^{(k_2)} + \sum_{i\le r-\tilde{r}} \partial_{w_i^{(k_2)}} f_N^{(k_2)} \rangle}{\|\nabla \hat{f}_{N,r}^{(k_1)}\| \cdot \|\nabla \hat{f}_{N,r}^{(k_2)}\|} \right]. \tag{59}$$

and by applying Lemma A.6, we have

$$\left| \mathbb{E}\left[C_{N,r}^R(k_1,k_2)\right] - \frac{\sum_{i\le\tilde{r}}(\alpha_i^R)^2\phi_i^2}{(r-\tilde{r})\beta^2 + \sum_{i\le\tilde{r}}(\alpha_i^R)^2} \right| \to 0. \tag{60}$$

Finally,

$$\left| \mathbb{E}\left[C_{N,r}^R(k_1,k_2) - C_{N,r+1}^R(k_1,k_2)\right] - \frac{\beta^2\sum_{i\le\tilde{r}}(\alpha_i^R)^2\phi_i^2}{\left((r-\tilde{r})\beta^2 + \sum_{i\le\tilde{r}}(\alpha_i^R)^2\right)\left((r-\tilde{r}+1)\beta^2 + \sum_{i\le\tilde{r}}(\alpha_i^R)^2\right)} \right| \to 0 \tag{61}$$

and here $g(r) = \frac{\beta^2\sum_{i\le\tilde{r}}(\alpha_i^R)^2\phi_i^2}{\left((r-\tilde{r})\beta^2+\sum_{i\le\tilde{r}}(\alpha_i^R)^2\right)\left((r-\tilde{r}+1)\beta^2+\sum_{i\le\tilde{r}}(\alpha_i^R)^2\right)}$ is strictly positive.

□

A.5. Discussion on the Additive Perturbed Model

In our framework, we express the perturbed Hessians as

$$\boldsymbol{H}_N(h^R,\omega^R) = \boldsymbol{H}_{\text{true}}(h^R,\omega^R) + \epsilon_N^R,$$

with the error matrices defined as $\epsilon_N^R = s_N X^R$, where X^R is a Wigner matrix.Wigner matrices have long been established as a canonical model for random perturbations in high-dimensional settings, such as perturbations in quantum systems

(Guhr et al., 1998; Brody et al., 1981) or as noise models in signal processing (Tulino et al., 2004), making them particularly well-suited as error matrices in our additive perturbation model. The use of a Wigner matrix is justified by its ability to capture intrinsic statistical fluctuations in the eigenvalues and eigenvectors, a property that has been extensively verified both theoretically and empirically in Random Matrix Theory.

Additionally, we scale the variance of the entries of X^R by σ^2/R rather than σ^2. This scaling is crucial because it prevents the eigenvalues of the perturbed Hessian from diverging as the matrix dimension R increases. If a variance of σ^2 were used, the eigenvalues of $\boldsymbol{H}_N(h^R, \omega^R)$ would diverge. In practice, the loss landscape displays controlled fluctuations, and the σ^2/R scaling maintains consistency with the reasonable distribution of eigenvalues.

B. Detail of the algorithms

In this section, we provide detailed explanation of fine-tuning version of FedLoRU and introduces variants of FedLoRU to adapt to environments with statistical and model heterogeneity by employing multiple or hierarchical low-rank updates.

B.1. FedLoRU for Fine-Tuning

In the fine-tuning version of FedLoRU, the approach deliberately avoids merging the low-rank update matrices into the frozen pre-trained model. Instead, these low-rank matrices are stored separately, enabling a plug-and-play mechanism. This design choice allows the pre-trained model to remain intact while the task-specific adaptations are provided solely by the auxiliary low-rank matrices. As a result, this framework not only minimizes storage overhead and communication costs but also maintains flexibility during fine-tuning — clients can easily swap or update the low-rank components without altering the core model, ensuring efficient and adaptable federated learning.

B.2. Personalized Federated Low-Rank Updates (pFedLoRU)

We develop the personalized FedLoRU (pFedLoRU) algorithm to address statistical heterogeneity (non-IID) in federated learning, building on the FedLoRU approach. The pFedLoRU algorithm enables each client k to train a personalized model adapted to its data distribution.

Algorithm 2 pFedLoRU.

Require: model $\boldsymbol{W}$, initial global low-rank update matrices $\boldsymbol{A}_0, \boldsymbol{B}_0$
Require: initial personal low-rank update matrices $\boldsymbol{L}_0, \boldsymbol{U}_0$
Require: scaling factors α_{global} and α_{per}, accumulation cycle τ, total round T
  Initialize: Server sends $\boldsymbol{W}$ to each client.
  for $t = 1, \cdots, T$ **do**
    Server selects M clients $\mathcal{K}_M$ and distributes $\boldsymbol{A}_{t-1}, \boldsymbol{B}_{t-1}$ to the clients in $\mathcal{K}_M$.
    for each client $k \in \mathcal{K}_M$ **do**
      Local training:
        Find $\boldsymbol{L}_t^{(k)}, \boldsymbol{U}_t^{(k)}$ by solving (62) starting from $\boldsymbol{W} + \alpha_{\text{global}} \boldsymbol{A}_{t-1}\boldsymbol{B}_{t-1} + \alpha_{\text{per}} \boldsymbol{L}_{t-1}^{(k)} \boldsymbol{U}_{t-1}^{(k)}$.
        Find $\boldsymbol{A}_t^{(k)}, \boldsymbol{B}_t^{(k)}$ by solving (63) starting from $\boldsymbol{W} + \alpha_{\text{global}} \boldsymbol{A}_{t-1}\boldsymbol{B}_{t-1} + \alpha_{\text{per}} \boldsymbol{L}_t^{(k)} \boldsymbol{U}_t^{(k)}$.
      Send $\boldsymbol{A}_t^{(k)}, \boldsymbol{B}_t^{(k)}$ to the server.
    end for
    Server aggregation: $\boldsymbol{A}_t \leftarrow \sum_{k \in \mathcal{K}_M} p^{(k)} \boldsymbol{A}_t^{(k)}$, $\boldsymbol{B}_t \leftarrow \sum_{k \in \mathcal{K}_M} p^{(k)} \boldsymbol{B}_t^{(k)}$.
    if $t \bmod \tau = 0$ **then**
      Server distributes $\boldsymbol{A}_t, \boldsymbol{B}_t$ to all clients .
      Each client k updates its local copy of the global model: $\boldsymbol{W} \leftarrow \boldsymbol{W} + \alpha_{\text{global}} \boldsymbol{A}_t \boldsymbol{B}_t$
    end if
  end for
  Return: The final model for client k is $\boldsymbol{W} + \sum_{t=1:\ t \bmod \tau = 0}^{T} \boldsymbol{A}_t \boldsymbol{B}_t + \boldsymbol{L}_T^{(k)} \boldsymbol{U}_T^{(k)}$.

In pFedLoRU, each client k maintains a local copy of the global model $\boldsymbol{W}$, global low-rank matrices $\boldsymbol{A}^{(k)}$ and $\boldsymbol{B}^{(k)}$, and personal matrices $\boldsymbol{L}^{(k)}$ and $\boldsymbol{U}^{(k)}$. The matrices $\boldsymbol{A}^{(k)}$ and $\boldsymbol{B}^{(k)}$ are shared with the server to update the global model, while $\boldsymbol{L}^{(k)}$ and $\boldsymbol{U}^{(k)}$ are tailored to adapt to the local distribution. In each round t, client k optimizes the personal matrices for

E_{per} epochs and the global matrices for E_{global} by solving

$$\boldsymbol{L}_t^{(k)},\ \boldsymbol{U}_t^{(k)} = \underset{\boldsymbol{L},\,\boldsymbol{U}}{\arg\min}\, f^{(k)}(\boldsymbol{W} + \alpha_{\text{global}}\boldsymbol{A}_{t-1}\boldsymbol{B}_{t-1} + \alpha_{\text{per}}\boldsymbol{L}\boldsymbol{U}), \tag{62}$$

$$\boldsymbol{A}_t^{(k)},\ \boldsymbol{B}_t^{(k)} = \underset{\bar{\boldsymbol{A}},\,\bar{\boldsymbol{B}}}{\arg\min}\, f^{(k)}(\boldsymbol{W} + \alpha_{\text{global}}\bar{\boldsymbol{A}}\bar{\boldsymbol{B}} + \alpha_{\text{per}}\boldsymbol{L}_t^{(k)}\boldsymbol{U}_t^{(k)}). \tag{63}$$

Subsequently, the server collects the global update matrices $\boldsymbol{A}_t^{(k)}$ and $\boldsymbol{B}_t^{(k)}$ from the clients, performs aggregation $\boldsymbol{A}_t \leftarrow \sum_{k\in\mathcal{K}_M} p^{(k)}\boldsymbol{A}_t^{(k)}$, $\boldsymbol{B}_t \leftarrow \sum_{k\in\mathcal{K}_M} p^{(k)}\boldsymbol{B}_t^{(k)}$, and broadcasts $\boldsymbol{A}_t$ and $\boldsymbol{B}_t$ to the clients. The clients then accumulate the low-rank updates accordingly as in FedLoRU. If clinet k performs inference, it is based on model $\boldsymbol{W} + \alpha_{\text{per}}\boldsymbol{L}_T^{(k)}\boldsymbol{U}_T^{(k)}$.

In pFedLoRU, the communication between the server and clients involves only the low-rank matrices $\boldsymbol{A}^{(k)}$ and $\boldsymbol{B}^{(k)}$, which substantially reduces communication overhead. In practice, since the global model incorporates general knowledge from the all clients' dataset, and the personalized model is essentially a fine-tuned version of the global model, we typically assign higher ranks to $\boldsymbol{A}^{(k)}$ and $\boldsymbol{B}^{(k)}$. Additionally, although we use the same rank for $\boldsymbol{L}^{(k)}$ and $\boldsymbol{U}^{(k)}$ across all clients in our experiments, each client can, in practice, use different ranks based on the complexity and size of their local dataset. It is also noteworthy that different ranks for $\boldsymbol{A}^{(k)}$ and $\boldsymbol{B}^{(k)}$ can be employed by integrating pFedLoRU and mFedLoRU.

B.3. Model-Heterogeneous Federated Low-Rank Updates (mFedLoRU)

When local clients possess varying hardware resources, it becomes impractical to use uniform low-rank matrices across all clients. To address this issue, we develop the model-heterogeneous FedLoRU (mFedLoRU) algorithm, which employs hierarchical low-rank updates that allows clients to use their adaptive update ranks. In mFedLoRU, at each round t, each client k receives $\boldsymbol{A}_{t-1}$ and $\boldsymbol{B}_{t-1}$ and updates its local copy of the global model as in FedLoRU. For local training, each client k generates and optimizes nested low-rank matrices $\boldsymbol{A}_{\text{d}}^{(k)}\boldsymbol{A}_{\text{u}}^{(k)}$ and $\boldsymbol{B}_{\text{d}}^{(k)}\boldsymbol{B}_{\text{u}}^{(k)}$ by solving

$$\boldsymbol{A}_{\text{d}}^{(k)}, \boldsymbol{A}_{\text{u}}^{(k)}, \boldsymbol{B}_{\text{d}}^{(k)}, \boldsymbol{B}_{\text{u}}^{(k)} = \underset{\bar{\boldsymbol{A}}_{\text{d}},\bar{\boldsymbol{A}}_{\text{u}},\bar{\boldsymbol{B}}_{\text{d}},\bar{\boldsymbol{B}}_{\text{u}}}{\arg\min}\ f^{(k)}(\boldsymbol{W} + \alpha(\boldsymbol{A}_{t-1} + \alpha_A^{(k)}\bar{\boldsymbol{A}}_{\text{d}}\bar{\boldsymbol{A}}_{\text{u}})(\boldsymbol{B}_{t-1} + \alpha_B^{(k)}\bar{\boldsymbol{B}}_{\text{d}}\bar{\boldsymbol{B}}_{\text{u}})). \tag{64}$$

Here, $\boldsymbol{A}_{t-1}\boldsymbol{B}_{t-1}$ are the rank-r low-rank matrices, and $\boldsymbol{A}_{\text{d}}^{(k)}\boldsymbol{A}_{\text{u}}^{(k)}$ and $\boldsymbol{B}_{\text{d}}^{(k)}\boldsymbol{B}_{\text{u}}^{(k)}$ are rank-r_A and rank-r_B low-rank matrices used to update $\boldsymbol{A}_{t-1}$ and $\boldsymbol{B}_{t-1}$. After local training, the server collects $\boldsymbol{A}_d^{(k)}$, $\boldsymbol{A}_u^{(k)}$, recovers the low-rank update matrix $\boldsymbol{A}_t^{(k)} \leftarrow \boldsymbol{A}_{t-1} + \alpha_A^{(k)}\boldsymbol{A}_d^{(k)}\boldsymbol{A}_u^{(k)}$, and finally aggregates $\boldsymbol{A}_t \leftarrow \sum_{k\in\mathcal{K}_M} p^{(k)}\boldsymbol{A}_{t-1}^{(k)}$. The same process applies for the low-rank matrices $\boldsymbol{B}_d^{(k)}$ and $\boldsymbol{B}_d^{(k)}$.

Model-heterogeneous FedLoRU (mFedLoRU) algorithm enables each client k to utilize a rank tailored to its resource constraints. Similar to FedLoRU, client k maintains low-rank update matrices $\boldsymbol{A}^{(k)} \in \mathbb{R}^{m\times r}$ and $\boldsymbol{B}^{(k)} \in \mathbb{R}^{r\times n}$, but Each client k decides whether to use nested low-rank updates or not. If a client opts out of nested low-rank updates, it updates its low-rank modules like in FedLoRU. However, if client k chooses nested low-rank updates, it determines the locally adapted rank $r_A^{(k)}, r_B^{(k)} < r$ based on its resources. At each round, it initializes nested low-rank update matrices $\boldsymbol{A}_{\text{d}}^{(k)} \in \mathbb{R}^{m\times r_A^{(k)}}$, $\boldsymbol{A}_{\text{u}}^{(k)} \in \mathbb{R}^{r_A^{(k)}\times r}$ and $\boldsymbol{B}_{\text{d}}^{(k)} \in \mathbb{R}^{r\times r_B^{(k)}}$, $\boldsymbol{B}_{\text{u}}^{(k)} \in \mathbb{R}^{r_B^{(k)}\times n}$ such that $\boldsymbol{A}_{\text{d}}^{(k)}\boldsymbol{A}_{\text{u}}^{(k)} = 0$ and $\boldsymbol{B}_{\text{d}}^{(k)}\boldsymbol{B}_{\text{u}}^{(k)} = 0$. After local training by solving (64), we update client k's original low-rank matrices as follows:

$$\boldsymbol{A}^{(k)} \leftarrow \boldsymbol{A}^{(k)} + \alpha_A^{(k)}\boldsymbol{A}_{\text{d}}^{(k)}\boldsymbol{A}_{\text{u}}^{(k)}, \quad \boldsymbol{B}^{(k)} \leftarrow \boldsymbol{B}^{(k)} + \alpha_A^{(k)}\boldsymbol{B}_{\text{d}}^{(k)}\boldsymbol{B}_{\text{u}}^{(k)}. \tag{65}$$

After local training, to reduce communication overhead, the client does not recover its original low-rank matrices directly. Instead, it sends the nested low-rank matrices to the server, which recovers them into rank-r low-rank matrices $\boldsymbol{A}^{(k)} \leftarrow \boldsymbol{A} + \alpha_A^{(k)}\boldsymbol{A}_{\text{d}}^{(k)}\boldsymbol{A}_{\text{u}}^{(k)}$, and $\boldsymbol{B}^{(k)} \leftarrow \boldsymbol{B} + \alpha_B^{(k)}\boldsymbol{B}_{\text{d}}^{(k)}\boldsymbol{B}_{\text{u}}^{(k)}$, and then performs aggregation using these rank-r low-rank matrices as in FedLoRU. By using this strategy, the communication overhead is reduced from $2mn$ to $r(m+n) + r_A(m+r) + r_B(n+r)$.

B.4. Personalized Federated Low-Rank Adaptation (pFedLoRA)

We outline two variants of the personalized FedLoRA algorithm here. We use these algorithms to compare our pFedLoRU. Both versions of pFedLoRA follow a similar framework, where each client maintains a full-rank global model $\boldsymbol{W}$ and its own personalization models $\boldsymbol{L}^{(k)}$ and $\boldsymbol{U}^{(k)}$.

Algorithm 3 mFedLoRU.

Require: model $\boldsymbol{W}$, initial low-rank update matrices $\boldsymbol{A}_0, \boldsymbol{B}_0$
Require: scaling factors $\alpha, \alpha_A^{(k)}, \alpha_B^{(k)}$
Require: accumulation cycle τ, total round T
  Initialize: Server sends $\boldsymbol{W}$ to each client.
  for $t = 1, \cdots, T$ **do**
    Server selects M clients $\mathcal{K}_M$ and distributes $\boldsymbol{A}_{t-1}, \boldsymbol{B}_{t-1}$.
    for each client $k \in \mathcal{K}_M$ **do**
      Initializes nested low-rank updates $\boldsymbol{A}_{\text{d}}^{(k)}$, $\boldsymbol{A}_{\text{u}}^{(k)}$ and $\boldsymbol{B}_{\text{d}}^{(k)}$, $\boldsymbol{B}_{\text{u}}^{(k)}$.
      Local training:
        Find $\boldsymbol{A}_{\text{d}}^{(k)}$, $\boldsymbol{A}_{\text{u}}^{(k)}$, $\boldsymbol{B}_{\text{d}}^{(k)}$, $\boldsymbol{B}_{\text{u}}^{(k)}$ by solving (64)
        starting from $\boldsymbol{W} + \alpha(\boldsymbol{A}_{t-1} + \alpha_A^{(k)} \boldsymbol{A}_{\text{d}}^{(k)} \boldsymbol{A}_{\text{u}}^{(k)})(\boldsymbol{B}_{t-1} + \alpha_B^{(k)} \boldsymbol{B}_{\text{d}}^{(k)} \boldsymbol{B}_{\text{u}}^{(k)})$.
      Sends $\boldsymbol{A}_{\text{d}}^{(k)} \boldsymbol{A}_{\text{u}}^{(k)}$ and $\boldsymbol{B}_{\text{d}}^{(k)} \boldsymbol{B}_{\text{u}}^{(k)}$ to the server.
    end for
    Recover rank-r low-rank updates from hierarchical low-rank updates:
      $\boldsymbol{A}_t^{(k)} \leftarrow \boldsymbol{A}_{t-1} + \alpha_A^{(k)} \boldsymbol{A}_{\text{d}}^{(k)} \boldsymbol{A}_{\text{u}}^{(k)}, \quad \boldsymbol{B}_t^{(k)} \leftarrow \boldsymbol{B}_{t-1} + \alpha_B^{(k)} \boldsymbol{B}_{\text{d}}^{(k)} \boldsymbol{B}_{\text{u}}^{(k)}$.
    Server aggregation: $\boldsymbol{A}_t \leftarrow \sum_{k \in \mathcal{K}_M} p^{(k)} \boldsymbol{A}_t^{(k)}$, $\boldsymbol{B}_t \leftarrow \sum_{k \in \mathcal{K}_M} p^{(k)} \boldsymbol{B}_t^{(k)}$.
    if $t \bmod \tau = 0$ **then**
      Server distributes $\boldsymbol{A}_t, \boldsymbol{B}_t$ to all clients.
      Each client k updates its local model: $\boldsymbol{W} \leftarrow \boldsymbol{W} + \alpha \boldsymbol{A}_t \boldsymbol{B}_t$.
    end if
  end for
  Return: $\boldsymbol{W} + \sum_{t=1:\ t \bmod \tau = 0}^{T} \boldsymbol{A}_t \boldsymbol{B}_t$.

In pFedLoRA(1), the first variant, as suggested by (Wu et al., 2024) and other FedLoRA algorithms, the personalization models are optimized separately from the global model. Specifically, the algorithm first optimizes the personalization models for E_{per} iterations and subsequently optimizes the global full-rank model for E_{global} iterations by solving:

$$\boldsymbol{L}_t^{(k)}, \boldsymbol{U}_t^{(k)} = \arg\min_{\boldsymbol{L}, \boldsymbol{U}} f^{(k)}(\boldsymbol{W}_{t-1} + \alpha_{\text{per}} \boldsymbol{L}\boldsymbol{U}), \tag{66}$$

$$\boldsymbol{W}_t^{(k)} = \arg\min_{\boldsymbol{W}} f^{(k)}(\boldsymbol{W} + \alpha_{\text{per}} \boldsymbol{L}_t^{(k)} \boldsymbol{U}_t^{(k)}). \tag{67}$$

However, pFedLoRA(1) has been found to be less effective compared to our modified version pFedLoRA(2). The second variant, pFedLoRA(2), optimizes both the personalization modules and the global full-rank model simultaneously for $E = E_{\text{per}} + E_{\text{global}}$ iterations by solving:

$$\boldsymbol{W}_t^{(k)}, \boldsymbol{L}_t^{(k)}, \boldsymbol{U}_t^{(k)} = \arg\min_{\boldsymbol{W}, \boldsymbol{L}, \boldsymbol{U}} f^{(k)}(\boldsymbol{W} + \alpha_{\text{per}} \boldsymbol{L}\boldsymbol{U}). \tag{68}$$

C. Detail of the experiment setting

In this section, we provide a detailed explanation of the experiments, including the datasets and hyperparameters used. We use PyTorch 11.4 version and 4 TITAN Xp GPUs. Additionally, we present the experiment for pFedLoRU and mFedLoRU, which are not included in the main text.

C.1. Datasets and Models

The federated learning experiments were performed using four datasets: Fashion-MNIST (FMNIST, (Xiao et al., 2017)), CIFAR-10, CIFAR-100 (Krizhevsky & Hinton, 2009), and Alpaca (Taori et al., 2023). The Alpaca dataset, consisting of

Algorithm 4 pFedLoRA.

Require: model $\boldsymbol{W}$, initial personal low-rank update matrices$\boldsymbol{L}_0, \boldsymbol{U}_0$
Require: scaling factors α_{per}, total round T
  Initialize: Server sends $\boldsymbol{W}$ to each client.
  for $t = 1, \cdots, T$ **do**
    Server selects M clients $\mathcal{K}_M$ and distributes $\boldsymbol{W}_{t-1}$ and client k initializes it as a local copy of the global model.
    for each client $k \in \mathcal{K}_M$ **do**
      Local training - pFedLoRA(1):
        Find $\boldsymbol{L}_t^{(k)}, \boldsymbol{U}_t^{(k)}$ by solving (66) starting from $\boldsymbol{W}_{t-1} + \alpha_{\text{per}} \boldsymbol{L}_{t-1}^{(k)} \boldsymbol{U}_{t-1}^{(k)}$.
        Find $\boldsymbol{W}_t^{(k)}$ by solving (67) starting from $\boldsymbol{W}_{t-1} + \alpha_{\text{per}} \boldsymbol{L}_t^{(k)} \boldsymbol{U}_t^{(k)}$.
      Local training - pFedLoRA(2):
        Find $\boldsymbol{W}_t^{(k)}, \boldsymbol{L}_t^{(k)}, \boldsymbol{U}_t^{(k)}$ together by solving (68) starting from $\boldsymbol{W}_{t-1} + \alpha_{\text{per}} \boldsymbol{L}_{t-1}^{(k)} \boldsymbol{U}_{t-1}^{(k)}$.
      Send $\boldsymbol{W}_t^{(k)}$ to the server.
    end for
    Server aggregation: $\boldsymbol{W}_t \leftarrow \sum_{k \in \mathcal{K}_M} p^{(k)} \boldsymbol{W}_t^{(k)}$.
  end for
  Return: The final model for client k is $\boldsymbol{W}_T + \boldsymbol{L}_T^{(k)} \boldsymbol{U}_T^{(k)}$.

52,000 instruction and demonstration samples, was divided into 50,000 instances for training and 2,000 for testing in our fine-tuning experiment.

We construct datasets for clients by evenly splitting the training data among K clients in a statistically homogeneous (i.e., IID) federated learning setting. For the heterogeneous statistical setting, we follow the procedure outlined in (Hsu et al., 2019), which involves applying latent Dirichlet allocation (LDA) over the dataset labels to create clients' datasets. In this approach, each client is assigned a multinomial distribution over the labels, from which its examples are sampled. The multinomial distribution is drawn from a symmetric Dirichlet distribution with parameter ψ. For the non-IID setting, we use $\psi = 0.5$ to simulate a severely heterogeneous environment.

C.2. Implementation and training details

Detailed implementation of FedLoRA, FedLoRU, and FedHM In FedLoRA, FedLoRU, FedHM, and their variant algorithms, we apply low-rank factorization to the convolutional layers in ResNet-based models and to the self-attention modules in LLaMA2-3B. Specifically, for ResNet10 and ResNet18, we factorize the convolutional layers in layer1 through layer4, and for LLaMA2-3B, we factorize the self-attention modules in q_proj, k_proj, v_proj, and o_proj. We explore various low-rank configurations, setting the ranks of the factorized modules to 16, 32, 64, and 128 for FedLoRA and FedLoRU. We use rank $r = 128$ as the largest rank since our initial experiments showed it to have the best performance/memory trade-off. For FedHM, since its factorization scheme differs from that of FedLoRA and FedLoRU, we determine equivalent rank factors that yield the same number of trainable parameters as the ranks used in FedLoRA and FedLoRU.

We employ two strategies for initializing the low-rank update matrices in FedLoRU. For random initialization, as adopted in (Hu et al., 2021), we initialize $\boldsymbol{A}$ with a random Gaussian distribution and set $\boldsymbol{B}$ to zero, ensuring that $\boldsymbol{AB}$ is zero at the start. Alternatively, for momentum initialization, we retain the existing weights of the matrices, continuing to use the previous low-rank update matrices. This approach leverages momentum effects as described in the ReLoRA(Lialin et al., 2023). The scheduling of accumulations is also critical due to the varying nature of the training phases across different rounds; in this study, we employ periodic accumulation with the accumulation cycle determined through a grid search over the values {20, 30, 40, 50, 60, 70, 80}, though this area warrants further investigation. We assess the performance by evaluating Top-1 test accuracy across experiments. In the non-IID setting, due to significant fluctuations in performance, we report the average of the last five test accuracy values.

Federated learning setting The federated learning experiments were conducted using four datasets: FMNIST, CIFAR-10, CIFAR-100, and Alpaca. The client sampling rate, representing the proportion of clients selected per communication round, was set at 0.5 for all datasets. Each client performed 5 local epochs per communication round on the image datasets with a

batch size of 32, while client performed 1 local epochs on Alpaca with a batch size of 16.

For training FMNIST, CIFAR-10, and CIFAR-100, we utilized stochastic gradient descent (SGD) with a momentum of 0.9 as the local optimizer. The learning rate was selected through a grid search over 0.3, 0.2, 0.1, 0.05, 0.01, and a Cosine-Annealing learning rate scheduler was applied throughout the training process, with a minimum learning rate of 0.001 and a cycle step set to 50 or the total number of communication rounds. For fine-tuning LLaMA2-3B, we used AdamW (Loshchilov & Hutter, 2017) as the local optimizer, with a learning rate of 3×10^{-4} and betas set to (0.9, 0.999), without employing a learning rate scheduler.

Fine-tuning setting We assess the fine-tuning performance of FedLoRA and FedLoRU using two different ranks, 8 and 16. For the low-rank matrix factorization of LLaMA2-3B, we employ the PEFT library (Mangrulkar et al., 2022). The percentage of trainable parameters is 0.124% for rank 8 and 0.248% for rank 16.

Personalization and model heterogeneous setting We compare pFedLoRU against pFedLoRA (Wu et al., 2024), and for mFedLoRU, we compare with the model-heterogeneous version of FedHM. For the model heterogeneous setting, we simulate virtual environments where each client is assigned a different nominal rank, thereby restricting them to use low-rank update matrices of varying ranks. In particular, we tested two different model heterogeneous configurations in mFedLoRU experiments where the clients had different ranks, denoted as r, which reflect the computational resources or constraints of each client. For FedHM, we match the number of trainable parameters corresponding to the model with specific rank in mFedLoRU experiments.

Table 2: Detailed model heterogeneous settings in our experiments. Both settings include total 20 clients.

Rank of a client		$r = 128$	$r = 64$	$r = 32$	$r = 16$
#Clients	**setting 1**	5	5	5	5
	setting 2	-	6	6	7

The motivation behind these settings is to establish a challenging model heterogeneous environment. This is particularly important as we observed that FedLoRU with $r = 128$ produces similar results to FedAvg with a full-rank model. Therefore, these configurations were designed to test the algorithm's adaptability under more demanding and diverse client conditions. In addition, we set α_A and α_B to satisfy $\alpha_A/r_A = \alpha_A/r_B = 1/2$, as our empirical observations indicate that the choice of α values in the range of $1/4$ to 1 has minimal effect on overall performance.

C.3. Detail of the estimated stable rank experiment

We conduct an experiment to support our theoretical analysis that the Hessians of loss functions trained on smaller datasets exhibit larger stable ranks. In this experiment, we randomly select either 50 or 500 samples from the CIFAR-100 dataset and train a ResNet-18 model using only these 50 or 500 samples. Every 5 epochs, we compute an estimated stable rank of the Hessian, as calculating the true stable rank is computationally challenging due to the need to determine all singular values. Instead, we estimate the empirical spectral density using pyHessian (Yao et al., 2020), which provides the empirical singular values $\sigma_i(H)$ of a Hessian H and their corresponding densities $p(\sigma_i)$, $i = 1, \cdots, Q$. Based on this, we calculate the estimated stable rank as follows:

$$\hat{\text{srank}}(H) = \frac{\sum_{i=1}^{Q} p(\sigma_i)\ \sigma_i^2(H)}{p(\sigma_1)\ \sigma_1^2(H)} \tag{69}$$

Figure 1 shows the results of the experiment, demonstrating that the Hessians trained on the smaller dataset ($n = 50$) consistently exhibits higher estimated stable ranks compared to those trained on the larger dataset ($n = 500$).

D. Experiment Result for pFedLoRU and mFedLoRU

We evaluate the performance of pFedLoRU and mFedLoRU on statistical heterogeneous and model heterogeneous FL environments. Table 3 shows the performance of pFedLoRU and pFedLoRA. We use two variants of pFedLoRA, each utilizing different optimization schemes. For a comprehensive description of pFedLoRA(1) and pFedLoRA(2), see Appendix B.4. Under both non-IID levels ($\psi = 0.1$ and $\psi = 0.5$), pFedLoRU shows a clear advantage in terms of accuracy compared

to pFedLoRA. In addition, despite having less than half the number of parameters, pFedLoRU consistently achieves higher accuracy.

Table 3: Comparison of the average test accuracy across local models for pFedLoRA and pFedLoRU with varying non-IIDness (ψ) on CIFAR100.

Algorithm	**#params**	**Non-IIDness**	
		$\psi = 0.1$	$\psi = 0.5$
pFedLoRA(1)	11.22M	45.36	42.14
pFedLoRA(2)	11.22M	47.45	42.28
pFedLoRU	4.63M	**49.65**	**46.50**

On the other hands, Table 4 shows the performanec of mFedLoRU and FedHM. FedHM outperforms mFedLoRU in both heterogeneous settings (setting 1 and setting 2) on the CIFAR-10 dataset, indicating that FedHM handles model heterogeneity more effectively for simpler tasks. This suggests that FedHM is better suited for less complex datasets such as CIFAR-10, where its approach proves more efficient. However, mFedLoRU outperforms FedHM in both heterogeneous settings for the more complex CIFAR-100 dataset, demonstrating its potential in addressing the model-heterogeneous problem in federated learning. A key advantage of mFedLoRU is that it does not require additional computational steps, such as the weight factorization used in FedHM, making it a more efficient solution in scenarios involving more challenging tasks.

Table 4: Comparison of test accuracy for FedHM and mFedLoRU in two model-heterogeneous settings.

Dataset	**Setting**	**FedHM**	**mFedLoRU**
CIFAR-10	setting 1	**88.09**	84.81
	setting 2	**88.68**	84.36
CIFAR-100	setting 1	49.84	**51.16**
	setting 2	50.52	**50.89**

E. Further Discussion on Experimental Results

In this section, we present learning curve plots and additional experimental results that were not included in the main text. Furthermore, we provide a more detailed analysis and discussion of the experimental outcomes.

E.1. Experiment Results for FedAvg

To emphasize the comparison between FedLoRU and other communication-efficient federated learning algorithms, we have excluded the FedAvg results from the main text. The FedAvg outcomes are instead provided in Table 5.

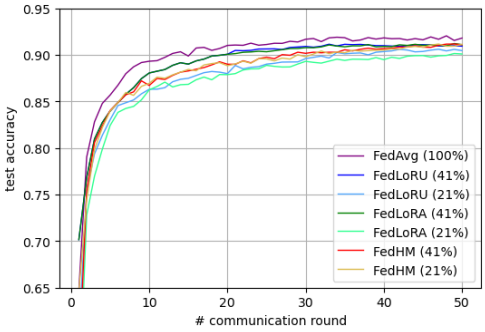
Table 5: Top-1 test accuracy of FedAvg under different federated learning settings and datasets

Dataset		**FMNIST**	**CIFAR-10**	**CIFAR-100**
FL setting	**IID - K=20**	91.81	93.48	69.97
	IID - K=100	90.19	85.14	55.14
	NonIID - K=20	80.03	79.65	19.18

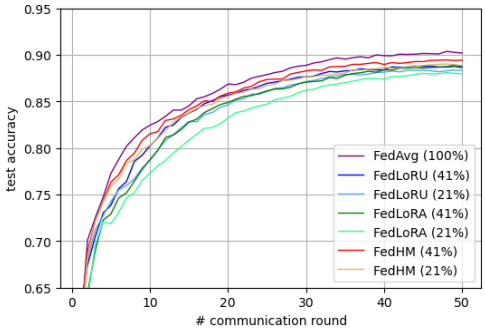
From Table 1 and Table 5, we observe that FedAvg consistently performs well across different datasets and settings, but its performance tends to drop as the number of clients increases and in non-IID scenarios. For example, in the CIFAR-100 dataset under the IID setting with 100 clients, FedAvg achieves a test accuracy of 55.14%, while its accuracy drops significantly to 19.18% in the non-IID setting with 20 clients. This illustrates FedAvg's limitations in handling large client numbers and heterogeneous data distributions.

In comparison, FedLoRU demonstrates competitive performance relative to FedAvg. While FedLoRU is at most 5% less accurate than FedAvg in some cases, it sometimes outperforms FedAvg, particularly in scenarios with a larger number of clients. For instance, in the CIFAR-100 IID setting with 100 clients, FedLoRU achieves a test accuracy of 57.96%, which

surpasses FedAvg's accuracy of 55.14%. This suggests that FedLoRU's low-rank update approach scales better with an increasing number of clients and is more robust in large-scale federated learning environments.

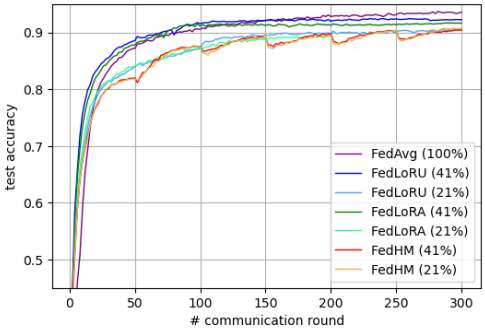


(a) FMNIST - IID - K=20

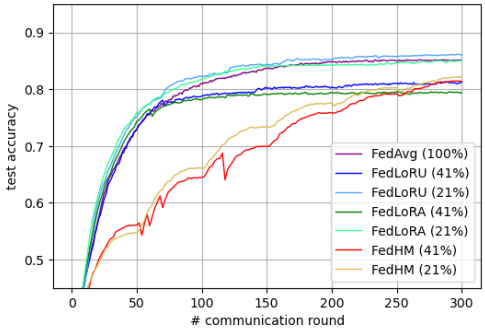


(b) FMNIST - IID - K=100

Figure 5: The test accuracy curves for FMNIST under an IID setting with K=20 and K=100.

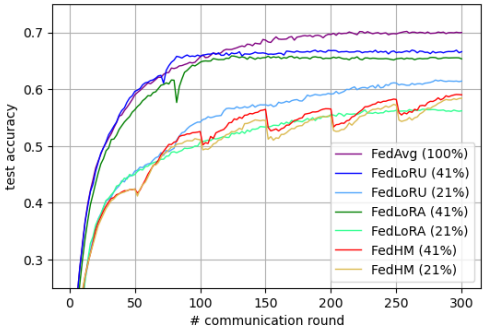


(a) CIFAR-10 - IID - K=20

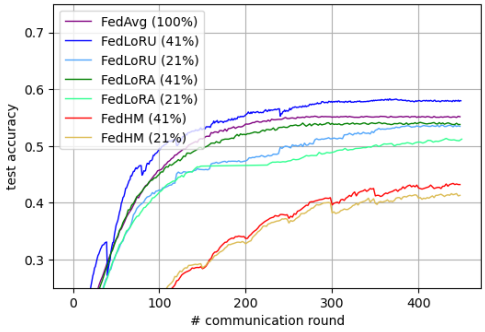


(b) CIFAR-10 - IID - K=100

Figure 6: The test accuracy curves for CIFAR-10 under an IID setting with K=20 and K=100.



(a) CIFAR-100 - IID - K=20



(b) CIFAR-100 - IID - K=100

Figure 7: The test accuracy curves for CIFAR-100 under an IID setting with K=20 and K=100.

E.2. Learning Curve Plots For IID Setting

We present the test accuracy curves for experiments conducted under a statistically homogeneous setting. Figure 5, Figure 6 and Figure 7 shows the test accuracy with respect to communication round under the IID setting. The fluctuations observed

in the graphs are attributable to the use of a cosine-annealing learning rate scheduler.

E.3. Discussion on Communication Cost

One of the main motivation of FedLoRU is to reduce the communication cost by using low-rank updates while maintaining reasonable performances. When the original weight matrix $\boldsymbol{W} \in \mathbb{R}^{m\times n}$ requires mn parameters to be communicated, FedLoRU with rank r requires $r(m+n)$ parameters. Additionally, as we can see in Figure 5, Figure 6 and Figure 7, the convergence speed is similar to FedAvg, resulting in much lower communication overheads.

Building on the motivation to reduce communication costs, Figure 3 compares the communication overheads across several federated learning algorithms—FedAvg, FedHM, FedLoRA, and FedLoRU—using the CIFAR-10 and CIFAR-100 datasets. The figure evaluates the communication cost in gigabytes (GB) required to reach specific target test accuracy (denoted as $T\%$) for different numbers of clients (K) and datasets. We compute the communication cost as 2 $\times$ (#clients) $\times$ (participation rate) $\times$ (#parameters) $\times$ (parameter memory size) $\times$ (#round). It is evident that FedLoRU consistently achieves significantly lower communication costs compared to the other methods.

E.4. Relative difference in performance in terms of the number of clients

Table 6 presents a comparison of test accuracy between FedAvg, FedLoRA, and FedLoRU across varying number of clients, illustrating the relative performance of these algorithms as the number of clients increases. FedLoRU consistently outperforms FedAvg when the number of clients exceeds 100, demonstrating its scalability and effectiveness in cross-device federated learning environments. Interestingly, even FedLoRA, which does not accumulate low-rank updates as in FedLoRU, outperforms FedAvg, particularly when the number of clients reaches 200 and above. This result suggests that simply adopting low-rank updates in high-client FL can significantly improve performance. These findings align with our theoretical insights, highlighting the potential benefits of leveraging low-rank structures in federated learning, even without the accumulation strategy employed by FedLoRU.

Table 6: A comparison between FedAvg, FedLoRA, and FedLoRU accuracy across varying client numbers. The ratio is the relative difference in accuracy between two algorithms. Here, we compute the ratio of FedLoRA and FedLoRU compared to FedAvg. For example, ratio of FedLoRU is defined as Ratio $= \frac{\text{FedLoRU}-\text{FedAvg}}{\text{FedLoRU}}$.

		FedLoRA		**FedLoRU**	
#Clients	**FedAvg**	acc	ratio	acc	ratio
20	69.97	65.53	-0.063	66.81	-0.046
50	64.68	59.87	-0.074	62.45	-0.034
100	55.14	53.79	-0.024	57.96	+0.051
200	38.85	42.42	+0.092	44.85	+0.154
300	24.94	32.69	+0.311	36.79	+0.475
400	21.44	31.41	+0.465	35.86	+0.673

We extended our experiments to settings with a lower participation ratio and a larger number of clients. Specifically, we examined $K = 100, 200$ with $C = 0.1$, using an IID CIFAR-100 dataset, which is more challenging than FMNIST and CIFAR-10. For these tests, we used the ResNet18 model, applying full parameter training for FedAvg and 41% parameter training for low-rank methods. The results, averaged over three runs with minimal standard deviation (¡ 0.005), are presented in Table 7.

Table 7: A comparison between FedAvg, FedHM, FedLoRA, and FedLoRU accuracy for experiments under large client numbers $K = 100, 200$ with lower participation ratio $C = 0.1$.

	FedAvg	FedHM	FedLoRA	FedLoRU
K=100	0.5382	0.5732	0.5506	**0.5837**
K=200	0.3885	0.4872	0.5227	**0.5393**

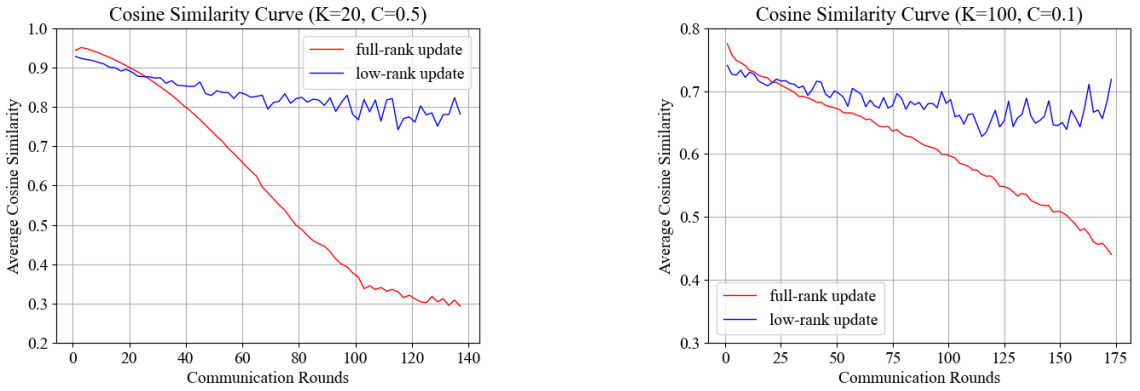
The results indicate that low-rank training methods consistently outperform full-rank training when the participation ratio is low and the number of clients increases. Among the low-rank approaches, FedLoRU achieves the highest accuracy,

demonstrating its effectiveness in large-scale federated learning. These findings reinforce the advantages of using low-rank updates, particularly in settings with a large number of clients and limited participation per round.

E.5. Model alignment of FedLoRU

Theorem 3.2 shows that clients exhibit higher stable ranks, indicating a more complex loss landscape that exacerbates client discrepancies. Further, Theorem 3.3 demonstrates that low-rank approximations of client gradients are more closely aligned compared to higher-rank approximations. This behavior implies that constraining updates to a low-rank space, as implemented in FedLoRU, inherently regularizes client training by aligning updates along major directions and reducing variations between clients.

To empirically validate the alignment of client updates with global updates, we conducted experiments to calculate the average cosine similarity between the global update (difference between the aggregated global model and the previous global model) and the local updates (differences between the locally trained models and the previous global model). These experiments were conducted on CIFAR-100 in two configurations: (1) 20 clients with a participation rate of 0.5 and (2) 100 clients with a participation rate of 0.1, both in iid setting. The average cosine similarity across clients serves as a proxy for the degree of alignment, with higher values indicating stronger alignment between local and global updates.



(a) CIFAR-100 - IID - K=20 - C=0.5

(b) CIFAR-100 - IID - K=100 - C=0.1

Figure 8: Average cosine similarity between global updates and local updates was calculated. Model weights were vectorized, and the cosine similarity between each participating client's update and the global update was computed.

In the first configuration with 20 clients, full-rank updates (FedAvg) initially exhibit higher cosine similarity, indicating stronger alignment with the global update in the early training stages. As training progresses, the alignment for both full-rank and low-rank updates decreases. Notably, after approximately 20 communication rounds, the low-rank updates consistently achieve higher cosine similarity than full-rank updates. This observation suggests that while low-rank updates initially align less closely with global updates due to their constrained nature, they adapt over time, improving alignment and maintaining stronger consistency during later communication rounds.

In the second configuration with 100 clients, a similar trend is observed. Full-rank updates initially achieve higher cosine similarity, reflecting better alignment in the early training stages. However, as training proceeds, low-rank updates surpass full-rank updates in alignment. The slightly lower cosine similarity of low-rank updates in the early stages likely reflects the initial adaptation of client updates within the constraints of the low-rank subspace.