SDFLoRA: Selective Decoupled Federated LoRA for Privacy-preserving Fine-tuning with Heterogeneous Clients

Zhikang Shen[1] , **Jianrong Lu**[1] , **Haiyuan Wan**[2] and **Jianhai Chen**[1,*]
[1]Zhejiang University
[2]Tsinghua University
{hydrolysis02, jrong.alvin}@gmail.com, wanhy24@mails.tsinghua.edu.cn, chenjh919@zju.edu.cn

Abstract

Federated learning (FL) has emerged as a promising paradigm for adapting large language models (LLMs) to distributed data. To mitigate the high communication and memory overhead, parameter-efficient techniques such as Low Rank Adaptation (LoRA) are widely adopted. However, practical deployments often exhibit *rank and data heterogeneity*, making direct aggregation of LoRA updates biased and unstable. Existing approaches enforce a unified rank or align heterogeneous updates into a single shared subspace, which undermines personalization and accuracy. Moreover, under differential privacy (DP), adding noise to such mixed updates could perturb client-specific directions that should remain local, resulting in utility loss. To address these issues, we propose **S**elective **D**ecoupled **F**ederated **LoRA** (***SDFLoRA***), a structure-aware LoRA framework that decouples each client update into a shared component for updating, and a private component that preserves client-specific semantics. Subspace alignment and aggregation are applied selectively to the shared module, while private modules remain local. We further design low rank re-compression to maintain a fixed rank budget for the shared module. This structure supports privacy aware optimization by injecting DP noise exclusively into the shared module. Experiments on multiple benchmarks demonstrate that SDFLoRA outperforms federated LoRA baselines and exhibits strong robustness under privacy constraints.

1 Introduction

Parameter-efficient federated fine-tuning (PEFT) has emerged as a critical paradigm for adapting Large Language Models (LLMs) to distributed, privacy-sensitive datasets. By optimizing only lightweight adapters, clients avoid full-model communication and memory overhead, enabling deployments in domains such as healthcare, finance, and on-device intelligence. Among PEFT approaches, low-rank adaptation (LoRA) [Hu *et al.*, 2022] is the dominant choice in federated LLM adaptation and is widely adopted in recent FL-LLM systems [Wang *et al.*, 2024; Yang *et al.*, 2025; Long *et al.*, 2024].

Despite these advances, practical Federated Learning (FL) deployments could exhibit strong heterogeneity, as clients originate from different domains, tasks, and user populations. LoRA updates naturally contain both transferable directions shared across clients and client-specific directions that encode local semantics and personalization. However, existing federated LoRA pipelines typically enforce shared alignment and aggregation over all update directions, which suppresses client-specific adaptations and limits personalization. This issue is further exacerbated by rank heterogeneity, where clients adopt different low rank configurations due to diverse system constraints, making naïve aggregation even more biased and unstable.

Existing solutions for federated LoRA under heterogeneous ranks can be grouped into two paradigms. The first paradigm enforces a unified rank across all clients, which reduces flexibility and can waste resources on constrained devices [Cho *et al.*, 2024]. The second paradigm aligns heterogeneous LoRA updates into a shared subspace to enable aggregation. However, it tends to align all directions indiscriminately, including client-specific components that should remain private and unshared, thereby harming personalization [Zhao *et al.*, 2023]. Meanwhile, differential privacy (DP) is widely adopted in FL to protect private client information. When such structurally entangled updates are globally aligned, adding noise to the entire update could perturb client-specific components that are not meant to be shared, leading to utility loss without delivering a meaningful privacy guarantee [Wei *et al.*, 2023].

Motivated by these observations, we propose **S**elective **D**ecoupled **F**ederated **LoRA** (***SDFLoRA***), which decomposes client adapter into a shared module that captures transferable knowledge and a private module that preserves client-specific adaptations. Only the shared module participates in selective subspace alignment and aggregation across clients, while private modules remain private and unaggregated. To control rank growth induced by stacking-based aggregation, we further introduce low rank re-compression to maintain a fixed rank budget for the shared module. This structure naturally supports privacy aware optimization by applying DP

*Corresponding author.

mechanisms exclusively to the shared module updates, avoiding unnecessary perturbations on local adaptations.

Our contributions are summarized as follows:

- We propose SDFLoRA, a structure-aware federated LoRA framework that decouples each client adapter into a shared module and a private module, and performs selective subspace alignment and aggregation exclusively on the shared module. The shared module is further constrained by a fixed rank budget via low rank recompression, enabling stable and efficient aggregation.
- We introduce a DP compatible optimization enabled by the decoupling, where differential privacy noise is injected only into the aggregated shareable update. This design avoids unnecessary perturbations on client-specific directions and leads to improved robustness under privacy constraints.
- Extensive experiments on multiple benchmarks with heterogeneous clients demonstrate that SDFLoRA outperforms representative federated LoRA baselines, particularly in achieving a superior balance between privacy protection and model performance.

2 Related Work

2.1 Federated Optimization under Client Heterogeneity

FL enables collaborative model training without centralizing data by alternating between local optimization and server-side aggregation. Federated Averaging (FedAvg) [McMahan *et al.*, 2017] is the canonical approach, but its simple weighted averaging mechanism often leads to unstable convergence under non-IID data distributions, partial client participation, and multiple local update steps, a phenomenon commonly referred to as client drift.

To mitigate heterogeneity-induced optimization challenges, a lot of work has proposed algorithmic modifications from different perspectives. FedProx [Li *et al.*, 2020] introduces a proximal regularization term to constrain local updates from deviating excessively from the shared model. FedDyn [Durmus *et al.*, 2021] and FedDC [Gao *et al.*, 2022] further address client drift by incorporating dynamic correction terms that explicitly decouple local objectives from the shared optimization trajectory. Another line of work focuses on correcting biases caused by heterogeneous local computation. FedNova [Wang *et al.*, 2020] normalizes client updates according to their effective number of local steps, while SCAFFOLD [Karimireddy *et al.*, 2020b] reduces update variance through control variates shared between the server and clients. From an optimization-centric viewpoint, MIME and MIME-Lite [Karimireddy *et al.*, 2020a] reinterpret federated learning as biased stochastic optimization and improve convergence by aligning federated updates with centralized SGD via momentum and gradient correction.

Despite their demonstrated effectiveness, these methods are predominantly designed for full-parameter federated optimization and implicitly assume homogeneous model structures across clients. When parameter-efficient fine-tuning is adopted, additional forms of heterogeneity emerge. In particular, low-rank adapters introduce *structural heterogeneity*, where clients may employ different adapter ranks or decompositions due to resource constraints or task diversity, which cannot be directly handled by existing optimization-centric approaches.

2.2 Parameter-Efficient Fine-Tuning and Federated LoRA

PEFT has become a standard approach for adapting large models under resource constraints. LoRA [Hu *et al.*, 2022] freezes the backbone and learns low-rank updates for selected layers, greatly reducing trainable parameters and communication costs, which makes it especially attractive in FL.

Several studies extend LoRA to federated settings. FedIT [Zhang *et al.*, 2024] performs FedAvg-style aggregation over LoRA parameters, enabling efficient collaboration but implicitly assumes homogeneous adapter configurations and can be sensitive to rank mismatch. To mitigate rank heterogeneity, recent work has explored *stacking-based aggregation* to align heterogeneous low-rank updates into a shared subspace. FLoRA [Wang *et al.*, 2024] is a representative approach that constructs an expanded low-rank subspace via stacking, providing a principled way to aggregate clients with different ranks without padding or truncation. Other variants improve initialization, aggregation bias, or fairness properties for federated LoRA training [Bian *et al.*, 2025; Guo *et al.*, 2025].

These methods demonstrate that stacking is an effective tool for handling *rank heterogeneity*. However, they typically treat each client adapter as a monolithic update and implicitly assume that *all* low-rank directions should be globally aligned and aggregated. In federated LLM fine-tuning, client updates may contain a mixture of transferable knowledge and locally specific adaptations; indiscriminate alignment can over-regularize personalization and become fragile when additional constraints (e.g., privacy noise) are introduced.

2.3 Personalization and Privacy in Federated Learning

Personalized FL aims to account for client-specific objectives and data distributions by learning both shared and private components. Representative methods include Ditto [Gao *et al.*, 2024], which couples a shared model with personalized local models via regularization, and FedRep [Collins *et al.*, 2021], which separates shared representations from client-specific heads. These approaches highlight the importance of explicitly distinguishing between shared and locally specialized parameters, but they are not designed around the structural properties of low-rank adapters in LLM fine-tuning.

Privacy preservation is another fundamental concern in FL. Differential privacy (DP) [Dwork, 2006] provides formal guarantees, and DP-SGD [Abadi *et al.*, 2016] has become a standard mechanism to protect training updates by gradient clipping and noise injection. In federated settings, differential privacy is typically enforced through client-side or server-side perturbations, such as DP-SGD-based noise injection on local updates or aggregated models [Kairouz *et*

al., 2021]. However, applying DP noise uniformly to all parameters can significantly degrade utility, particularly for large models and heterogeneous clients, and adversarial or poisoned updates further exacerbate this instability in federated aggregation [Lu *et al.*, 2024]. Recent work suggests that selective or structured privacy mechanisms may yield better utility–privacy trade-offs, yet existing federated LoRA approaches do not explicitly leverage adapter structure to support such selective protection.

2.4 Positioning of Our Work

Compared to prior federated LoRA methods, our work focuses on the semantic roles of low-rank updates and the resulting need for structural awareness in aggregation. Rather than proposing another stacking mechanism, we introduce SDFLoRA that separates globally alignable updates from locally personalized ones. Stacking-based aggregation is then applied *selectively* to the shared module, while private modules remain private and unaggregated. This perspective addresses not only how to aggregate heterogeneous ranks, but also which updates should be aggregated under personalization and privacy constraints, enabling robust federated LLM fine-tuning with improved stability under DP noise.

3 Method

3.1 Preliminaries: Federated LoRA Fine-tuning

We consider federated fine-tuning with K clients. In round t, a subset S_t of clients participates, each optimizing a local objective $F_k(\cdot)$ on its private data. Let W denote the frozen backbone parameters of an LLM. LoRA [Hu *et al.*, 2022] augments a target linear layer $W \in R^{d_{\text{out}} \times d_{\text{in}}}$ with a low-rank update

$$\Delta W_k = B_k A_k, \quad A_k \in R^{r_k \times d_{\text{in}}},\ B_k \in R^{d_{\text{out}} \times r_k} \tag{1}$$

where r_k is the adapter rank (potentially different across clients). During local training, client k updates $\{A_k, B_k\}$ (and keeps W fixed). The server aggregates client updates and broadcasts the aggregated adapter for the next round.

Challenge: rank heterogeneity and semantic heterogeneity. When ranks differ across clients, directly averaging low-rank factors is ill-defined without padding or truncation, and may introduce biased updates. Stacking-based aggregation alleviates dimensional mismatch by aligning heterogeneous updates into an expanded subspace, but implicitly assumes that *all* updates should be globally aligned and aggregated. In federated LLM fine-tuning, client updates often contain a mixture of globally transferable knowledge and client-specific adaptations; forcing shared alignment on all updates can harm personalization and becomes fragile under differential privacy noise. Our goal is therefore to design an aggregation scheme that is robust to rank heterogeneity while respecting the semantic roles of low-rank updates.

3.2 Dual-Model LoRA Decomposition

We propose to decompose each client adapter into two structurally separated low-rank modules:

$$\Delta W_k = \Delta W_k^{(g)} + \Delta W_k^{(l)} = B_k^{(g)} A_k^{(g)} + B_k^{(l)} A_k^{(l)} \tag{2}$$

where $\Delta W_k^{(g)}$ is a **shared module** intended to capture transferable directions shared across clients, and $\Delta W_k^{(l)}$ is a **private module** for client-specific adaptations.

This decomposition reflects an important observation in federated LLM fine-tuning: client-side low-rank updates are *semantically heterogeneous*. Some update directions correspond to transferable knowledge that should be aligned and shared across clients, while others encode client-specific or domain-specific adaptations whose forced alignment can be harmful. By explicitly separating $\Delta W_k^{(g)}$ and $\Delta W_k^{(l)}$, we impose a structural constraint on the subspaces that are allowed to participate in cross-client aggregation.

As illustrated in Figure 1, we decouple transferable (public) updates from client-specific (private) adaptations, enabling selective aggregation and privacy-preserving training.

Federated roles. The two modules play different roles in the federated process: (i) the shared module participates in cross-client aggregation and is broadcast each round; (ii) the private module is *never aggregated* and remains private to the client. This structural separation is the key design choice that allows us to restrict subspace alignment to semantically appropriate updates, rather than treating each LoRA adapter as a monolithic update.

Optimization objective. In each round t, client $k \in S_t$ receives the current shared module $\theta^{(g,t)}$ (the collection of all $\{A^{(g)}, B^{(g)}\}$ across adapted layers) and keeps its private module $\theta_k^{(l,t)}$. The client performs E steps of local optimization to minimize:

$$\min_{\theta^{(g)},\, \theta_k^{(l)}} \; F_k\big(W, \theta^{(g)}, \theta_k^{(l)}\big) \tag{3}$$

and uploads only the updated shared module $\theta_k^{(g,t)}$ (or its update) to the server. The private module is kept on-device and continues across rounds.

3.3 Selective Stacking Aggregation on the Shared Module

We propose *selective stacking*, a stacking-based aggregation strategy that is applied exclusively to the shared module while explicitly excluding client-private updates from cross-client alignment. Let $\theta_k^{(g,t)}$ denote the shared-module parameters trained by client k at round t for a given layer, i.e., $\theta_k^{(g,t)} = \{A_k^{(g,t)}, B_k^{(g,t)}\}$ with rank $r_k^{(g)}$. To aggregate heterogeneous ranks without padding or truncation, we perform **stacking-based** aggregation only on the shared module.

Stacking construction. For a target layer, we form a stacked representation:

$$A_t^{(g)} = \text{Concat}\Big(p_k\, A_k^{(g,t)}\Big)_{k \in S_t} \tag{4}$$

$$B_t^{(g)} = \text{Concat}\Big(p_k\, B_k^{(g,t)}\Big)_{k \in S_t} \tag{5}$$

where p_k is the aggregation weight (e.g., proportional to the local data size) and $\text{Concat}(\cdot)$ concatenates along the rank dimension. This yields an aggregated update

$$\Delta W_t^{(g)} = B_t^{(g)} A_t^{(g)} \tag{6}$$

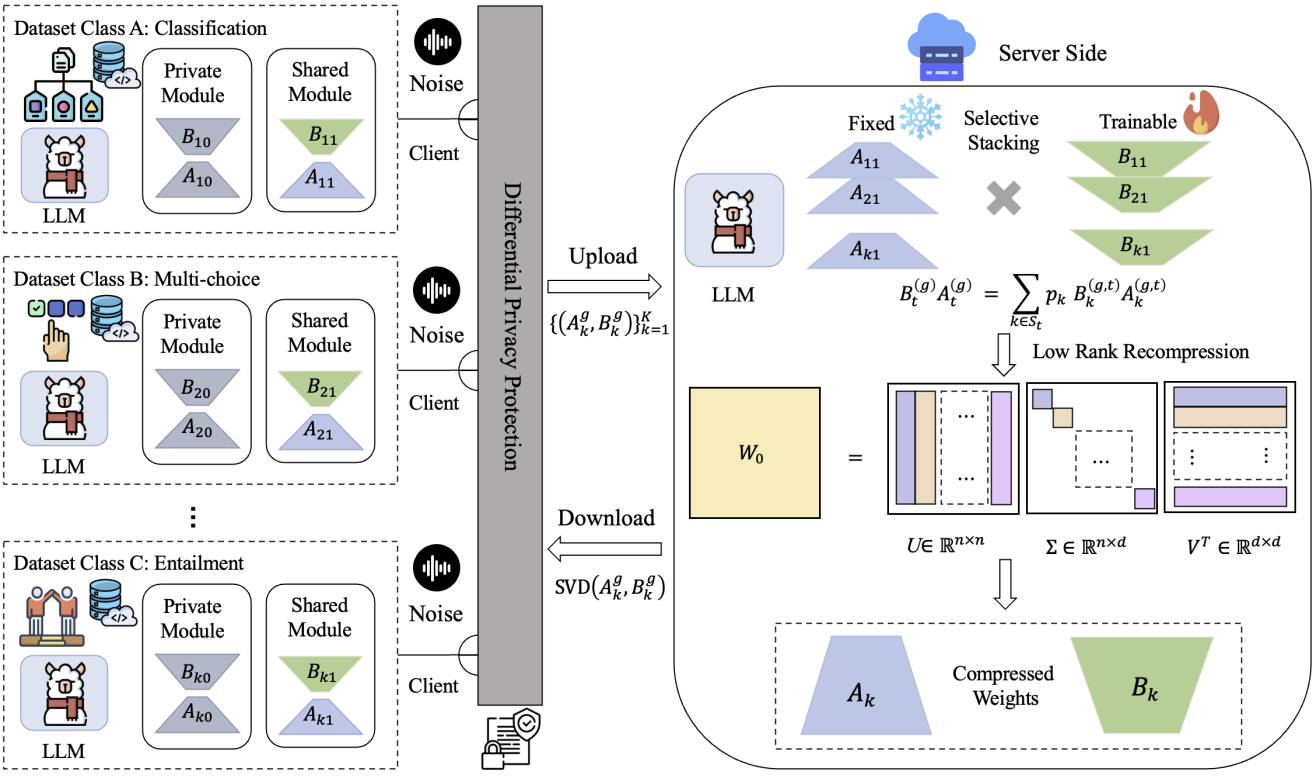

Figure 1: Framework overview. Each client keeps a private module and trains a shared module on a frozen LLM. Only the shared module is communicated and aggregated via selective stacking; Differential Privacy noise is applied to the shared update.

which is well-defined under rank heterogeneity. Importantly, *private modules are excluded from stacking and aggregation*, avoiding forced alignment of client-specific directions.

The stacking operation preserves the following equivalence:

$$B_t^{(g)} A_t^{(g)} = \sum_{k \in S_t} p_k B_k^{(g,t)} A_k^{(g,t)} \tag{7}$$

while avoiding direct averaging of incompatible low-rank factors. In contrast to padding-based alignment, stacking delays subspace merging until after aggregation, preventing premature mixing of heterogeneous update directions.

Rank control by re-compression. A practical consequence of stacking is that the effective rank of the aggregated shared update grows with the number of participating clients and communication rounds. Concretely, the concatenation in Eqs. (5)–(6) expands the rank dimension, which increases both the transmission cost (larger adapter factors) and the memory footprint on the server and clients.

Moreover, under privacy-aware training, stochasticity (e.g., DP noise) and client heterogeneity tend to inject low-energy directions that accumulate across rounds, leading to unstable optimization and diminished generalization if left uncontrolled. To keep the shared module within a fixed budget, we apply low-rank re-compression to $\Delta W_t^{(g)}$:

$$\Delta W_t^{(g)} \approx \tilde{B}_t^{(g)} \tilde{A}_t^{(g)}, \quad \operatorname{rank}(\tilde{B}_t^{(g)} \tilde{A}_t^{(g)}) = r_{\max}. \tag{8}$$

In practice, we compute a truncated SVD (or an equivalent low-rank factorization) $\Delta W_t^{(g)} = U \Sigma V^\top$ and retain only the top-$r_{\max}$ singular components, yielding $\Delta W_t^{(g)} \approx U_r \Sigma_r V_r^\top$. We then convert this truncated form back to LoRA-style factors by setting $\tilde{B}_t^{(g)} = U_r \Sigma_r^{1/2}$ and $\tilde{A}_t^{(g)} = \Sigma_r^{1/2} V_r^\top$, so that $\tilde{B}_t^{(g)} \tilde{A}_t^{(g)}$ matches the best rank-$r_{\max}$ approximation under the Frobenius norm. This step acts as a spectral filter that preserves dominant, transferable directions while discarding tail components that are more likely to be noise-dominated, and it keeps the communicated shared module compact for the next round. Finally, the server broadcasts $(\tilde{A}_t^{(g)}, \tilde{B}_t^{(g)})$ as the next-round shared module.

3.4 Privacy Aware Optimization (DP Compatibility)

Our decoupled design is naturally compatible with differential privacy (DP). The key idea is to apply DP mechanisms *only* to the shared module, while keeping private modules noise-free.

Client-side DP-SGD on the shared module. During local training, client k clips per-sample (or per-microbatch) gradients for the shared module and adds Gaussian noise:

$$g_k^{(g)} \leftarrow \operatorname{Clip}\left(g_k^{(g)}, C\right) + \mathcal{N}\left(0, \sigma^2 C^2 I\right) \tag{9}$$

where C is the clipping norm and σ is the noise multiplier. The perturbed gradients update $\theta_k^{(g)}$, which is then uploaded

Algorithm 1 Selective Decoupled Federated LoRA (One Round)

Input: Frozen backbone W; shared module $\theta^{(g,t)}$; client set S_t; weights $\{p_k\}$; rank budget $r_{\max}$; DP parameters (C, σ)
Output: Updated shared module $\theta^{(g,t+1)}$
Server broadcasts $\theta^{(g,t)}$ to clients $k \in S_t$.
for each client $k \in S_t$ **in parallel do**
  Initialize decoupled adapters with $\theta^{(g,t)}$ and local state $\theta_k^{(l,t)}$.
  for local step $e = 1, \ldots, E$ **do**
    Compute gradients on batch; update $\theta_k^{(g)}$ (with DP-SGD if enabled) and $\theta_k^{(l)}$.
  end for
  Upload $\theta_k^{(g,t)}$ to server; keep $\theta_k^{(l,t)}$ locally.
end for
Server aggregates shared modules via selective stacking:
  $\theta^{(g)} \leftarrow \mathrm{StackAgg}(\{\theta_k^{(g,t)}\}_{k \in S_t}, \{p_k\})$.
Re-compress to rank budget $r_{\max}$ to obtain $\theta^{(g,t+1)}$; otherwise set $\theta^{(g,t+1)} \leftarrow \theta^{(g)}$.

to the server. The private module $\theta_k^{(l)}$ is updated without noise, preserving client-specific expressiveness and improving optimization stability.

Why selective privacy helps. Injecting noise into the entire adapter amplifies variance and can destabilize learning, especially for heterogeneous clients. By restricting DP to the shared module, our framework protects transferable knowledge while avoiding unnecessary corruption of client-specific adaptations, yielding a better balance of utility and privacy.

3.5 Algorithm Summary

Algorithm 1 summarizes one communication round of our method. Each client receives the shared module, performs local training with decoupled adapters, uploads only the shared-module parameters, and the server performs selective stacking aggregation to produce the next shared module.

4 Experiments

4.1 Experimental Setup and Baselines

Tasks and datasets. We evaluate federated parameter-efficient fine-tuning on GLUE-style and MMLU benchmarks [Wang *et al.*, 2018; Hendrycks *et al.*, 2021]. We report results on QNLI, RTE, QQP, MNLI, and SST-2, which cover natural language inference, paraphrase identification, and sentiment classification. We use the official train/dev splits and report dev-set performance.

Backbone and adapter. We adopt LLaMA-7B [Touvron *et al.*, 2023] as the frozen backbone and insert LoRA adapters [Hu *et al.*, 2022] into linear projection layers. Only LoRA parameters are optimized while the backbone weights remain fixed.

Item	Setting
Backbone	LLaMA-7B
Adapter	LoRA
Tasks	GLUE (RTE, QQP, MNLI, SST-2), MMLU
Clients (K)	8
Rounds (T)	30
Rank (hetero.)	$\{r_k\} = [4, 4, 8, 8, 8, 8, 16, 16]$
Training	Optimizer: Adam; learning rate 2×10^{-4}
	Local steps per round: 10; batch size: 128
Hardware	Intel Xeon Gold 5318Y; 128GB RAM
	2× RTX A6000
Software	Ubuntu 22.04.5; Python 3.10; CUDA 12.4

Table 1: Experimental configuration and system environment.

Federated protocol and rank heterogeneity. We simulate a cross-device FL setting with $K = 8$ clients and run $T = 30$ communication rounds. Unless otherwise stated, all clients participate in each round and the server aggregates client updates with weights proportional to local data sizes, i.e., $\alpha_k = n_k / \sum_j n_j$. To model structural heterogeneity induced by rank mismatch, we set client ranks as $\{r_k\} = [4, 4, 8, 8, 8, 8, 16, 16]$. When a homogeneous setting is required, we use a shared rank $r_k = r$ for all clients. In an additional scalability study, we vary the number of participating clients K while keeping the remaining hyperparameters consistent.

Local optimization. Each client uses Adam with a fixed learning rate of 2×10^{-4} and batch size 128. Within each communication round, clients perform a fixed number of local update steps (10) on mini-batches before uploading their LoRA updates. All remaining optimizer hyperparameters follow the default settings in our implementation. These hyperparameters are kept identical across methods unless the experiment explicitly controls a specific factor, ensuring fair comparisons.

Evaluation metrics. We report Accuracy on QNLI/RTE/MNLI/SST-2. And for QQP, we also report Accuracy to maintain a unified metric across tasks.

Baselines. We compare the proposed aggregation strategy against representative alternatives under identical training budgets: **(i) Zero-padding** aligns heterogeneous LoRA updates by padding lower-rank factors to the maximum rank before weighted averaging; **(ii) FedAvg** performs standard weighted averaging of LoRA parameters in the homogeneous-rank setting; **(iii) Stacking(FLoRA)** concatenates client low-rank updates along the rank dimension to preserve the union of client subspaces.

Reproducibility. All experiments are conducted on a machine equipped with an Intel Xeon Gold 5318Y CPU, 128GB RAM, and 2×NVIDIA RTX A6000 GPUs, running Ubuntu 22.04.5, Python 3.10, and CUDA 12.4. Key hyperparameters and system configurations are summarized in Table 1.

Task	Hetero				Homo								
					Rank = 4			Rank = 8			Rank = 16		
	Padding	FLoRA	Ours	Δ	FedAvg	Ours	Δ	FedAvg	Ours	Δ	FedAvg	Ours	Δ
QNLI	95.81	96.10	**96.96**	+0.86	95.81	**97.78**	+1.97	96.50	**98.47**	+1.97	97.00	**98.65**	+1.65
RTE	95.45	97.20	**99.71**	+2.51	85.94	**88.89**	+2.95	85.62	**90.05**	+4.43	86.10	**90.80**	+4.70
QQP	86.72	87.10	**87.97**	+0.87	84.40	**87.60**	+3.20	83.10	**85.30**	+2.20	86.50	**89.87**	+3.37
MNLI(MATCHED)	65.48	69.30	**72.19**	+2.89	70.20	**73.05**	+2.85	69.85	**73.60**	+3.75	70.10	**74.00**	+3.90
MNLI(MISMATCH)	64.90	68.80	**71.62**	+2.82	69.55	**72.48**	+2.93	69.20	**73.05**	+3.85	69.50	**73.70**	+4.20
SST-2	81.10	81.70	**82.30**	+0.60	82.06	**82.14**	+0.08	**81.90**	81.88	−0.02	82.00	**82.20**	+0.20
MMLU	41.55	44.10	**45.60**	+1.50	44.00	**46.30**	+2.30	44.80	**47.00**	+2.20	45.25	**47.60**	+2.35
Avg.	75.86	77.76	**79.48**	+1.72	75.99	**78.32**	+2.33	75.85	**78.48**	+2.63	76.64	**79.55**	+2.91

Table 2: Federated aggregation results under rank-heterogeneous and rank-homogeneous clients. Δ denotes absolute improvement (percentage points) of **Ours** over the strongest baseline in each setting: $\max(\text{FLoRA}, \text{Padding})$ for Hetero, and FedAvg for Homo (Rank=4/8/16).

4.2 Results under Rank and Data Heterogeneity

Rank heterogeneous clients. Table 2 reports results under mismatched client ranks. Overall, **Ours** consistently outperforms zero-padding and FLoRA across all tasks. For example, it improves QNLI from 95.81% to 96.96% and QQP from 86.72% to 87.97%, and achieves notably larger gains on RTE and MNLI, which are more sensitive to cross-client directional mismatch.

These improvements stem from how different methods handle structural mismatch under rank heterogeneity. Zero-padding dilutes informative low-rank updates by enforcing a common high-rank parameterization, while FLoRA aggregates adapters in a monolithic manner. By contrast, Ours performs selective aggregation only on the shared component, leading to more stable and reliable updates under mismatched ranks.

Homogeneous rank sanity check. To verify that the benefit of our method is not merely due to resolving dimensional incompatibility, we further compare it with FedAvg-style averaging when all clients use the same rank (Table 2). Even in this homogeneous setting, stacking remains consistently superior or comparable.

For rank $r = 4$, Ours improves QQP by 3.20%, QNLI by 1.97%, and RTE by 2.95%, while maintaining comparable performance on SST-2. For rank $r = 8$, Ours again improves QQP (2.20%), QNLI (1.97%), and RTE (4.43%), with only a negligible difference on SST-2 (-0.02%). For rank $r = 16$, Ours yields further gains on QQP (3.37%), QNLI (1.65%), and RTE (4.70%), while slightly improving performance on SST-2 (0.20%). These results indicate that Ours acts as a subspace-preserving aggregator that remains beneficial even when the rank is fixed, rather than being a mere workaround for heterogeneous dimensions.

Effect of client population size. We study the impact of the federated population size by varying the number of participating clients K while keeping all other settings fixed. As shown in Figure 2, increasing K generally improves test accuracy, reflecting the benefit of broader data coverage in larger federations. With very few clients (K=4), aggregation is less effective due to limited diversity of update directions.

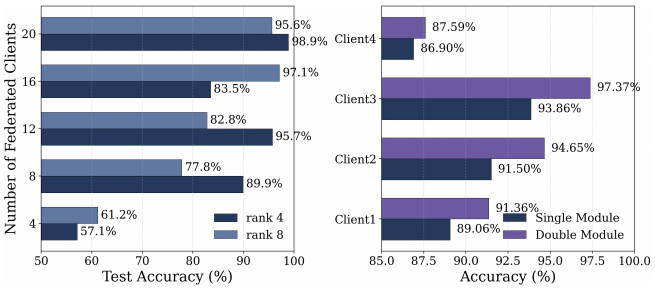

Figure 2: (Left) Test accuracy under varying numbers of federated clients (K) for two rank settings. Increasing K generally improves performance due to broader data coverage, while non-IID update discrepancy can cause transient fluctuations. (Right) Per-client accuracy comparison between single-module and dual-module adapters.

The performance trend is not strictly monotonic, and a temporary drop is observed at intermediate K values (e.g., around K=16), which is consistent with non-IID federated optimization. Comparing the two rank settings, lower ranks exhibit better robustness in the small-to-medium regime, whereas higher ranks achieve a higher accuracy ceiling as the client population grows. Overall, these results indicate that our method scales well with the federation size while exhibiting a non-trivial interaction between rank and population size.

Robustness to data heterogeneity. We further examine robustness under Non-IID data partitions controlled by a Dirichlet parameter α using the client-level accuracy standard deviation in Figure 3. Lower variance indicates more consistent performance across heterogeneous clients.

Under strong heterogeneity ($\alpha = 0.1$), stacking achieves lower variance than padding on all tasks, e.g., on SST-2 (0.344 vs. 0.379) and QNLI (0.387 vs. 0.420), indicating improved stability when client data distributions are highly skewed. At moderate heterogeneity ($\alpha = 0.5$), stacking again consistently reduces variance, for example lowering RTE from 0.372 to 0.343 and QNLI from 0.111 to 0.104, showing better robustness across clients.

At $\alpha = 1.0$, stacking remains slightly more stable on RTE

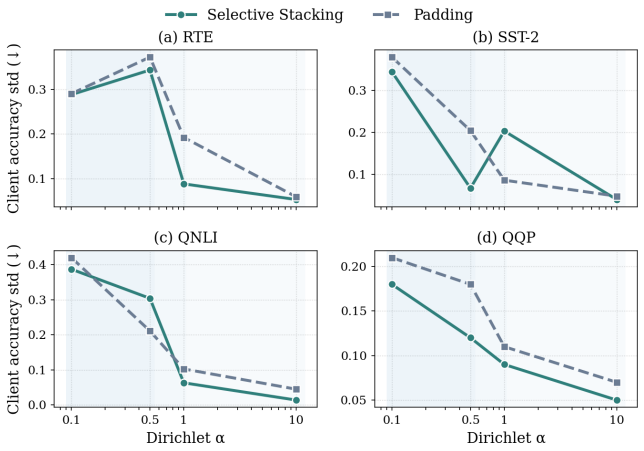

Figure 3: Client-level accuracy standard deviation under varying degrees of data heterogeneity (Dirichlet α) for selective stacking vs. padding.

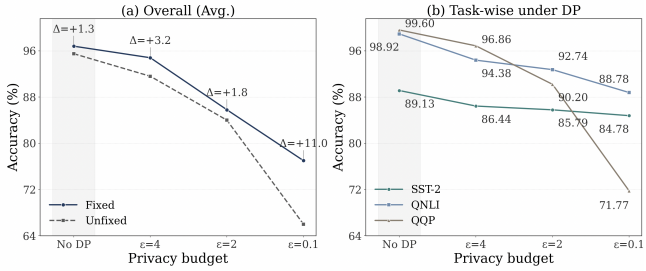

Figure 4: Accuracy under different privacy budgets ϵ. Fixed denotes applying structural constraints (selective stacking and rank control) on the shared module, while Unfixed corresponds to unconstrained aggregation over all updates. The fixed design exhibits substantially better stability under strong privacy constraints.

(0.088 vs. 0.092), but exhibits higher variance on SST-2 and QNLI. This suggests that when data heterogeneity is moderate, preserving a broader set of client-specific low-rank directions may amplify inter-client discrepancies on certain tasks, even though the shared model remains competitive. Importantly, when the partition approaches near-IID ($\alpha = 10$), stacking again achieves uniformly lower variance across all tasks, showing that stacking does not inherently harm stability.

4.3 Privacy and Rank Sensitivity

Privacy and utility trade-off. We apply DP-SGD during private training and report task accuracies under different privacy budgets ϵ (Figure 4). As ϵ increases (weaker privacy), the required noise scale decreases and performance improves.

Figure 4 further shows that enforcing structural constraints on the shared module significantly improves robustness under strong privacy constraints. When ϵ is small, unconstrained aggregation suffers from severe performance degradation, whereas the fixed shared-module design maintains substantially higher accuracy.

Rank r	Setting	QNLI	MNLI (M)	MNLI (MM)	QQP
4	No DP	**94.39**	74.62	**89.63**	86.02
	DP ($\epsilon = 1$)	92.41	72.45	85.42	84.83
8	No DP	88.78	**79.60**	79.22	82.06
	DP ($\epsilon = 1$)	84.49	73.34	74.18	81.25
16	No DP	84.06	71.04	76.93	**89.87**
	DP ($\epsilon = 1$)	81.41	70.75	69.32	87.32
32	No DP	88.89	66.46	73.41	80.55
	DP ($\epsilon = 1$)	85.47	62.04	71.02	77.99

Table 3: Rank sensitivity with and without differential privacy (DP).

Rank sensitivity with and without DP. Table 3 analyzes how the DP noise affects performance. Without DP, smaller ranks can be more stable on some tasks, while overly large ranks may introduce redundancy and hurt aggregation. With DP ($\epsilon = 1$), the optimal rank can shift due to noise, but small-to-medium ranks often remain a good trade-off.

4.4 Evidence for Decoupled Benefit

We further provide direct evidence that the dual-adapter (decoupled) design itself contributes to performance, independent of selective stacking and rank budgeting. As shown in Figure 2 (Right), under matched training budgets the dual-adapter improves accuracy on every client, raising the average from 90.33% to 92.74% with an improvement of 2.41 percentage points. Per-client gains range from 0.69% on Client 4 to 3.51% on Client 3, indicating that the shared and private branches capture complementary low-rank subspaces rather than redundant capacity.

5 Conclusion

We studied federated fine-tuning of LLMs with LoRA under practical client heterogeneity, including differences in data distributions, system constraints, adapter ranks, and privacy requirements. While stacking-based aggregation provides a natural way to reconcile rank mismatch, we argue that stacking alone is insufficient, as it implicitly forces shared alignment on all client updates, suppressing personalization and becoming fragile under differential privacy noise.

We proposed the **SDFLoRA** framework, which structurally decomposes each client LoRA adapter into shared and private components. Building on this decomposition, we introduce selective stacking that applies stacking-based alignment only to the shared module while keeping private modules private and unaggregated. This configuration supports privacy-aware optimization by injecting DP noise exclusively into the shared module. Experiments on multiple benchmarks show that our approach outperforms representative federated LoRA baselines and achieves a better balance between utility and privacy under DP noise. In future work, we plan to extend this structural separation to other adapter families and investigate adaptive mechanisms for allocating shared versus private capacity based on client characteristics and task demands.

Acknowledgments

This work was supported by the Key R&D Program of Zhejiang Province under Grant No. 2025C01084 and the CAAI–MindSpore Open Fund, developed on the OpenI Community, under Contract No. CAAIXSJLJJ 2025 MindSpore 04.

References

[Abadi *et al.*, 2016] Martin Abadi, Andy Chu, Ian Goodfellow, H Brendan McMahan, Ilya Mironov, Kunal Talwar, and Li Zhang. Deep learning with differential privacy. In *Proceedings of the 2016 ACM SIGSAC conference on computer and communications security*, pages 308–318, 2016.

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

[Cho *et al.*, 2024] Yae Jee Cho, Luyang Liu, Zheng Xu, Aldi Fahrezi, and Gauri Joshi. Heterogeneous lora for federated fine-tuning of on-device foundation models. In *Proceedings of the 2024 Conference on Empirical Methods in Natural Language Processing*, pages 12903–12913, 2024.

[Collins *et al.*, 2021] Liam Collins, Hamed Hassani, Aryan Mokhtari, and Sanjay Shakkottai. Exploiting shared representations for personalized federated learning. In *International conference on machine learning*, pages 2089–2099. PMLR, 2021.

[Durmus *et al.*, 2021] Alp Emre Durmus, Zhao Yue, Matas Ramon, Mattina Matthew, Whatmough Paul, and Saligrama Venkatesh. Federated learning based on dynamic regularization. In *International conference on learning representations*, 2021.

[Dwork, 2006] Cynthia Dwork. Differential privacy. In *International colloquium on automata, languages, and programming*, pages 1–12. Springer, 2006.

[Gao *et al.*, 2022] Liang Gao, Huazhu Fu, Li Li, Yingwen Chen, Ming Xu, and Cheng-Zhong Xu. Feddc: Federated learning with non-iid data via local drift decoupling and correction. In *Proceedings of the IEEE/CVF conference on computer vision and pattern recognition*, pages 10112–10121, 2022.

[Gao *et al.*, 2024] Lingzhi Gao, Zhenyuan Zhang, and Chao Wu. Feddtg: Federated data-free knowledge distillation via three-player generative adversarial networks. In *International Conference on Neural Information Processing*, pages 16–30. Springer, 2024.

[Guo *et al.*, 2025] Pengxin Guo, Shuang Zeng, Yanran Wang, Huijie Fan, Feifei Wang, and Liangqiong Qu. Selective aggregation for low-rank adaptation in federated learning. In *13th International Conference on Learning Representations*, 2025.

[Hendrycks *et al.*, 2021] Dan Hendrycks, Collin Burns, Steven Basart, Andy Zou, Mantas Mazeika, Dawn Song, and Jacob Steinhardt. Measuring massive multitask language understanding. In *International Conference on Learning Representations*, 2021.

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

[Kairouz *et al.*, 2021] Peter Kairouz, H Brendan McMahan, Brendan Avent, Aurélien Bellet, Mehdi Bennis, Arjun Nitin Bhagoji, Kallista Bonawitz, Zachary Charles, Graham Cormode, Rachel Cummings, et al. Advances and open problems in federated learning. *Foundations and trends® in machine learning*, 14(1–2):1–210, 2021.

[Karimireddy *et al.*, 2020a] Sai Praneeth Karimireddy, Martin Jaggi, Satyen Kale, Mehryar Mohri, Sashank J Reddi, Sebastian U Stich, and Ananda Theertha Suresh. Mime: Mimicking centralized stochastic algorithms in federated learning. *arXiv preprint arXiv:2008.03606*, 2020.

[Karimireddy *et al.*, 2020b] Sai Praneeth Karimireddy, Satyen Kale, Mehryar Mohri, Sashank Reddi, Sebastian Stich, and Ananda Theertha Suresh. Scaffold: Stochastic controlled averaging for federated learning. In *International conference on machine learning*, pages 5132–5143. PMLR, 2020.

[Li *et al.*, 2020] Tian Li, Anit Kumar Sahu, Manzil Zaheer, Maziar Sanjabi, Ameet Talwalkar, and Virginia Smith. Federated optimization in heterogeneous networks. *Proceedings of Machine learning and systems*, 2:429–450, 2020.

[Long *et al.*, 2024] Guodong Long, Tao Shen, Jing Jiang, Michael Blumenstein, et al. Dual-personalizing adapter for federated foundation models. *Advances in Neural Information Processing Systems*, 37:39409–39433, 2024.

[Lu *et al.*, 2024] Jianrong Lu, Shengshan Hu, Wei Wan, Minghui Li, Leo Yu Zhang, Lulu Xue, and Hai Jin. Depriving the survival space of adversaries against poisoned gradients in federated learning. *IEEE Transactions on Information Forensics and Security*, 19:5405–5418, 2024.

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

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

[Wang *et al.*, 2018] Alex Wang, Amanpreet Singh, Julian Michael, Felix Hill, Omer Levy, and Samuel Bowman. Glue: A multi-task benchmark and analysis platform for natural language understanding. In *Proceedings of the 2018 EMNLP workshop BlackboxNLP: Analyzing and interpreting neural networks for NLP*, pages 353–355, 2018.

[Wang *et al.*, 2020] Jianyu Wang, Qinghua Liu, Hao Liang, Gauri Joshi, and H Vincent Poor. Tackling the objective inconsistency problem in heterogeneous federated optimiza-

tion. *Advances in neural information processing systems*, 33:7611–7623, 2020.

[Wang *et al.*, 2024] Ziyao Wang, Zheyu Shen, Yexiao He, Guoheng Sun, Hongyi Wang, Lingjuan Lyu, and Ang Li. Flora: Federated fine-tuning large language models with heterogeneous low-rank adaptations. *Advances in Neural Information Processing Systems*, 37:22513–22533, 2024.

[Wei *et al.*, 2023] Kang Wei, Jun Li, Chuan Ma, Ming Ding, Wen Chen, Jun Wu, Meixia Tao, and H Vincent Poor. Personalized federated learning with differential privacy and convergence guarantee. *IEEE Transactions on Information Forensics and Security*, 18:4488–4503, 2023.

[Yang *et al.*, 2025] Yiyuan Yang, Guodong Long, Qinghua Lu, Liming Zhu, Jing Jiang, and Chengqi Zhang. Federated low-rank adaptation for foundation models: A survey. *arXiv preprint arXiv:2505.13502*, 2025.

[Zhang *et al.*, 2024] Jianyi Zhang, Saeed Vahidian, Martin Kuo, Chunyuan Li, Ruiyi Zhang, Tong Yu, Guoyin Wang, and Yiran Chen. Towards building the federatedgpt: Federated instruction tuning. In *ICASSP 2024-2024 IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP)*, pages 6915–6919. IEEE, 2024.

[Zhao *et al.*, 2023] Xuyang Zhao, Huiyuan Wang, and Wei Lin. The aggregation–heterogeneity trade-off in federated learning. In *The Thirty Sixth Annual Conference on Learning Theory*, pages 5478–5502. PMLR, 2023.