SAME UPDATES, DIFFERENT AGGREGATES: RETHINKING FEDERATED LORA THROUGH SHARED SUBSPACES

Anonymous authors
Paper under double-blind review

ABSTRACT

Federated LoRA communicates low-rank factors to adapt models across decentralized clients. Equivalent factorizations preserve each local model, yet can produce different server aggregates under factor averaging. We formulate *gauge invariance* as an aggregation principle: equivalent representations of the same client updates should yield the same server update. Controlled audits of ordinary training uploads show that product-preserving refactorization can change predictions after factor averaging. Guided by this principle, we introduce **GLoRA**, a **G**auge-invariant aggregation framework for federated **LoRA** that separates client subspace geometry from update coordinates. Invariant client projectors define a shared reference space, giving client updates a common coordinate frame for aggregation. The resulting aggregate is gauge-invariant at finite server rank and recovers the exact update average under sufficient subspace coverage. A common low-rank server state supplies adapters for heterogeneous client ranks. Extensive experiments on GLUE and SuperNI cover homogeneous-rank, heterogeneous-rank, and task-heterogeneous settings, including sparse participation and unseen-task generalization. Compared with the strongest baselines in each setting, GLoRA improves homogeneous-rank GLUE averages by 0.58–0.93 accuracy points and unseen-task ROUGE-L by an average of 1.52 points across three SuperNI rank distributions, while reducing server aggregation time in the tested low-rank regimes.

1 INTRODUCTION

Federated fine-tuning allows clients to adapt a shared model while retaining their data locally (McMahan et al., 2017; Zhang et al., 2024). Low-rank adaptation (LoRA) (Hu et al., 2022) reduces the computation and communication involved: each client trains and transmits two small factors whose product defines a model update. The server combines these updates and returns adapters that clients with different data and resources can train (Cho et al., 2024; Bai et al., 2024).

LoRA admits a representation freedom whose implications change when factors become objects of aggregation. For any invertible Q, an update $\Delta W = BA$ can also be represented as $(BQ)(Q^{-1}A)$. For a fixed model in centralized adaptation, these factorizations are functionally indistinguishable: they induce the same weight update and identical predictions. Federated learning introduces an additional operation that combines updates from independently trained clients. When the server averages their factors separately Zhang et al. (2024); Singhal et al. (2025), equivalent representations of the same client updates can produce different global updates, as illustrated in Figure 1. A change that leaves every local model unchanged can therefore alter the aggregated model. This reveals a distinction between local model equivalence and aggregation equivalence.

We study this issue through *gauge invariance*: aggregation should return the same update when a client replaces its factors with an equivalent representation. This principle applies even when the server must compress its aggregate to a limited rank. It asks that the compression depend on the updates being combined, while remaining unchanged under equivalent factor coordinates.

Our empirical study connects this representation freedom to federated training. Ordinary local optimization from shared initialization exhibits coordinate motion, while controlled SVD refactorization of real uploads changes 1.91%–14.95% of validation predictions after factor averaging without changing any client update (Figure 1). These observations motivate aggregation rules that preserve local model equivalence at the server.

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

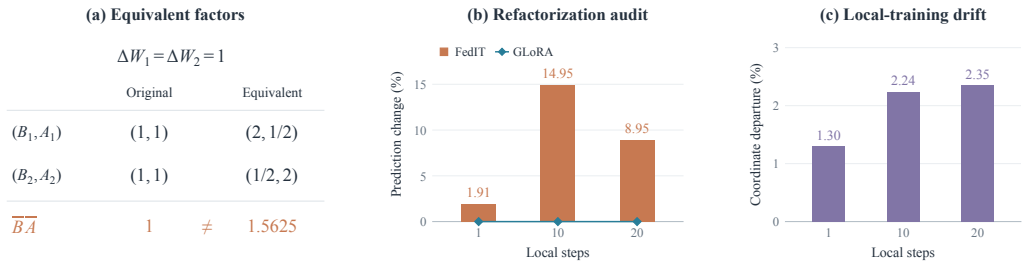


Figure 1: **Representation dependence in federated LoRA. (a)** Two equally weighted clients express identical updates through equivalent factors, yet separate factor averaging yields 1 and 1.5625. **(b)** SVD refactorization preserves every client update and aggregation weight, but changes FedIT's validation predictions; GLoRA's predictions remain unchanged. **(c)** Ordinary local training from shared factors exhibits coordinate departure, measured by aligning the trained factors toward their round-start representation while preserving products. Panels (b) and (c) use RoBERTa-base on SST-2.

Gauge invariance suggests a constructive way to organize aggregation: use invariant client geometry to establish a common frame, and combine updates through coordinates in that frame. Equivalent factors can change a client's basis while preserving its spanned subspace. This motivates separating *where an update lies* from *its coordinates within that space*.

Following this insight, we introduce **GLoRA**, a gauge-invariant aggregation framework for federated LoRA. Client subspace projectors define a rank-budgeted reference space that maximizes weighted overlap with the participating subspaces. The server projects client updates into this common frame and averages their coordinates, retaining the magnitudes and signs of the projected updates. Since these coordinates share a basis, their average remains within the same low-rank representation. The resulting aggregate is gauge-invariant at finite server rank and recovers the exact update average under sufficient subspace coverage. GLoRA operates on low-rank factors throughout, and its shared server state supplies adapters for heterogeneous client ranks.

Our contributions are threefold:

- **We identify** representation dependence in federated LoRA: equivalent factors can yield different aggregates despite preserving client updates. We connect this issue to coordinate motion observed during ordinary training, demonstrate its consequences through controlled refactorization, and formulate gauge invariance as an aggregation principle.
- **We propose** GLoRA, which separates invariant client subspace geometry from update coordinates to construct a shared aggregation space. GLoRA guarantees gauge invariance at finite server rank, recovers the exact update average under sufficient subspace coverage, and supports heterogeneous client ranks through a common low-rank server state.
- **We evaluate** GLoRA on GLUE Wang et al. (2018) and SuperNI Wang et al. (2022) across homogeneous-rank, heterogeneous-rank, and task-heterogeneous settings, including sparse participation and unseen-task generalization. Extensive experiments demonstrate its learning performance and server efficiency, while controlled comparisons and ablations clarify the contributions of shared-subspace aggregation and client readout.

2 Same Updates, Different Aggregates

2.1 A Representation-Invariance Requirement

For one adapted layer, client i represents its update as $\Delta W_i = B_i A_i$, where $B_i \in \mathbb{R}^{d_{\text{out}} \times r_i}$ and $A_i \in \mathbb{R}^{r_i \times d_{\text{in}}}$. We absorb any fixed LoRA scaling into these effective factors. For every invertible matrix $Q_i \in \mathrm{GL}(r_i)$,

$$(B_i Q_i)(Q_i^{-1} A_i) = B_i A_i. \tag{1}$$

We call factor pairs related by Eq.(1) *gauge-equivalent*. They induce the same local model.

Consider an aggregation rule Φ_R with server rank budget R, active clients $\mathcal{C}$, and fixed weights $p_i > 0$ summing to one. Its output is an induced update matrix; any history or other server inputs are held fixed.

Definition 1 (Gauge-invariant aggregation)**.** *An aggregation rule is gauge-invariant if, for every admissible set of invertible* $\{Q_i\}_{i\in\mathcal{C}}$*,*

$$\Phi_R\big(\{B_iQ_i, Q_i^{-1}A_i, p_i\}_{i\in\mathcal{C}}\big) = \Phi_R\big(\{B_i, A_i, p_i\}_{i\in\mathcal{C}}\big). \tag{2}$$

Invariance is defined for the represented update, allowing different server factors to encode the same matrix.

For equal client ranks, separate factor averaging gives

$$\bar{B}\bar{A} = \Big(\sum_i p_iB_i\Big)\Big(\sum_j p_jA_j\Big) = \sum_{i,j} p_ip_jB_iA_j. \tag{3}$$

Under equivalent input factors, a cross-client term becomes $B_iQ_iQ_j^{-1}A_j$, although every local product B_iA_i is unchanged. This exposes how factor coordinates enter the server aggregate. Exact update averaging is invariant by construction; a rank-limited approximation can satisfy the same principle by making its projection independent of those coordinates.

2.2 Auditing Ordinary Client Uploads

Experimental setup. We use RoBERTa-base on SST-2 with ten balanced-size non-IID clients, rank eight, and 20% participation. Clients train from shared round-start factors using the same model architecture and optimizer. We collect uploads at rounds 5, 10, and 20 under local-training budgets of 1, 10, and 20 steps. The audit uses one seed and summarizes the sampled rounds within each local-step setting. Aggregated models are evaluated on the same 872 validation examples.

Equivalent representations can change predictions. We isolate representation dependence by holding every client update and aggregation weight fixed while changing the uploaded factors. For each $B_iA_i = U_i\Sigma_iV_i^\top$, SVD refactorization produces the equivalent pair $(U_i\Sigma_i^{1/2}, \Sigma_i^{1/2}V_i^\top)$. The largest relative change in a client update is below 1.1×10^{-14}, preserving each local model to numerical precision. Despite this local equivalence, FedIT changes 1.91%–14.95% of validation predictions, whereas GLoRA's predictions remain unchanged (Figure 1(b)). Because the client updates and their weights are fixed, this difference isolates the effect of factor representation on aggregation. Measurements of the server updates show the same distinction: FedIT's aggregate changes, while GLoRA's changes remain at numerical precision. Table 8 in Appendix C reports the complete update and prediction measurements.

Ordinary training involves coordinate motion. We next examine how this representation freedom relates to local optimization from shared initialization. For each trained factor pair (B_i, A_i), we find an equivalent pair $(B_iQ_i, Q_i^{-1}A_i)$ that aligns the factors toward their round-start values. The displacement between the trained and aligned factors measures coordinate departure while holding the endpoint update fixed. This normalized departure ranges from 1.30% to 2.35% across the three local-step settings (Figure 1(c)), showing that shared initialization does not eliminate subsequent coordinate departure. To examine the optimization process directly, we also project recorded factor steps onto directions of the form $(B\Omega, -\Omega A)$, where $\Omega \in \mathbb{R}^{r\times r}$. These are gauge-tangent directions because their first-order effect on the update vanishes:

$$(B\Omega)A + B(-\Omega A) = 0. \tag{4}$$

Approximately 1.56% of factor-motion energy lies in these directions. The endpoint alignment and optimizer-step measurements thus provide complementary evidence of coordinate motion during ordinary training. The alignment objective and projection procedure are given in Appendix C.

Representation choices can affect subsequent learning. We run two ten-round branches from the same initialization with identical client participation and local data order. One branch uploads the original factors; the other applies SVD refactorization after each local training phase. FedIT ends with validation losses of 0.3136 and 0.3357, respectively, and 1.83% prediction disagreement. GLoRA reaches 89.56% accuracy with identical final predictions in both branches. This intervention extends the fixed-upload observation to subsequent learning; complete results appear in Appendix C.

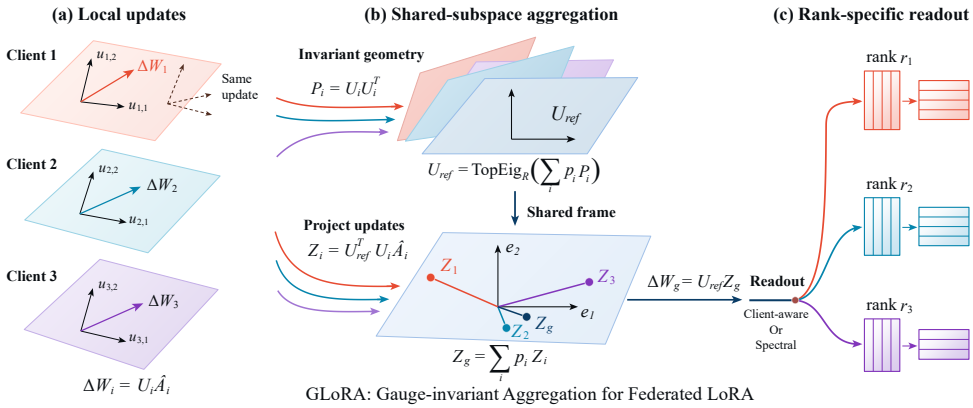


Figure 2: **Overview of GLoRA.** Client subspace projectors define an invariant shared reference space, where update coordinates are aggregated to produce a gauge-invariant server update. The shared server state then provides adapters for heterogeneous client ranks.

3 GLoRA: Aggregation in Shared Subspaces

GLoRA gives client updates a common coordinate frame before aggregation. We first choose that frame from invariant subspace geometry, then combine the update coordinates within it. For one layer and one round, the server stores

$$\Omega = (U_{\mathrm{ref}}, Z_g), \qquad \Delta W_g = U_{\mathrm{ref}} Z_g, \tag{5}$$

where $U_{\mathrm{ref}} \in \mathbb{R}^{d_{\mathrm{out}} \times R}$ has orthonormal columns and $Z_g \in \mathbb{R}^{R \times d_{\mathrm{in}}}$ contains shared coordinates.

3.1 From Local Factors to Invariant Geometry

A local orthonormal basis is useful for computation, but its orientation can still depend on the input factors. We therefore use it to represent a subspace through its projector. A reduced QR decomposition gives

$$B_i = U_i T_i, \qquad \widehat{A}_i = T_i A_i, \qquad \Delta W_i = U_i \widehat{A}_i, \qquad U_i^\top U_i = I. \tag{6}$$

We describe the construction for minimal nonzero factorizations, where $\mathrm{col}(B_i) = \mathrm{col}(\Delta W_i)$; Appendix A.1 gives the rank conventions.

The projector $P_i = U_i U_i^\top$ is unchanged by $B_i \mapsto B_i Q_i$. We combine these projectors to choose the reference space:

$$K = \sum_{i \in \mathcal{C}} p_i P_i, \qquad U_{\mathrm{ref}} = \mathrm{TopEig}_R(K), \qquad P_{\mathrm{ref}} = U_{\mathrm{ref}} U_{\mathrm{ref}}^\top. \tag{7}$$

This construction weights subspace directions: since $\mathrm{tr}(P_i) = r_i$, client i contributes total geometric weight $p_i r_i$. Higher-rank clients therefore contribute more geometric weight under equal client weights. We use this direction-weighted geometry by default and compare client-normalized and energy-aware alternatives in Appendix D.

The top eigenspace maximizes weighted overlap with the client subspaces:

$$U_{\mathrm{ref}} \in \arg\max_{U^\top U = I_R} \sum_i p_i \|U^\top U_i\|_F^2. \tag{8}$$

Equivalently, P_{ref} minimizes the weighted squared Frobenius distance to client projectors over rank-R orthogonal projectors (Appendix A.4). This defines a geometric consensus across client subspaces; truncated SVD instead minimizes reconstruction error of the averaged update. Update magnitudes remain in $\widehat{A}_i$ and enter the subsequent coordinate average.

The leading left singular vectors of the thin matrix $X = [\sqrt{p_i} U_i]_{i \in \mathcal{C}}$ yield this basis because $K = XX^\top$. This avoids forming a dense projector. Write $r_\Sigma = \sum_i r_i$ and $r_\cup = \mathrm{rank}(X)$, and choose $1 \le R \le r_\cup$. We assume a uniquely selected leading projector, as ensured by an eigengap at R; Appendix A.1 discusses ties.

3.2 Aggregating Updates in a Common Frame

Once a reference space has been chosen, all client updates have compatible coordinates:

$$Z_i = U_{\mathrm{ref}}^\top U_i \widehat{A}_i = U_{\mathrm{ref}}^\top \Delta W_i, \qquad Z_g = \sum_i p_i Z_i. \tag{9}$$

The projectors determine the shared reference space, while $Z_i = U_{\mathrm{ref}}^\top \Delta W_i$ encodes the magnitudes, signs, and row-side structure of each client's projected update. Only low-rank factors and intermediate matrices are multiplied. With $M = \sum_i p_i \Delta W_i$ denoting the update average for analysis, the induced server update satisfies

$$\Delta W_g = P_{\mathrm{ref}} M, \qquad M - \Delta W_g = (I - P_{\mathrm{ref}}) M. \tag{10}$$

Thus, GLoRA computes exactly the projection of the update average onto the reference space. This identity separates the effect of the rank budget from dependence on input coordinates.

Proposition 1 (Gauge invariance)**.** *Under the stated subspace-selection assumption, replacing each (B_i, A_i) by $(B_i Q_i, Q_i^{-1} A_i)$ leaves ΔW_g unchanged at every admissible server rank R.*

The input projectors, their aggregate K, and the updates ΔW_i are unchanged, so both factors on the right of Eq.(10) are unchanged.

Proposition 2 (Exact recovery under subspace coverage)**.** *If $P_{\mathrm{ref}} U_i = U_i$ for every active client, then $\Delta W_g = M$. In particular, retaining all $r_\cup$ nonzero directions of K gives exact recovery.*

At smaller rank, Eq.(10) identifies the approximation residual while Proposition 1 still applies. Proofs are in Appendix A.

3.3 Rank-Compatible Client Readout

The shared aggregate and its guarantees are established before client readout, which supplies adapters of different ranks. Since U_{ref} is orthonormal, a thin SVD of the coordinate matrix is sufficient:

$$Z_g = O \Sigma V^\top, \qquad U_s = U_{\mathrm{ref}} O, \qquad \Delta W_g = \sum_{j=1}^{q} \sigma_j u_j v_j^\top. \tag{11}$$

Here $q = \min(R, d_{\mathrm{in}})$ is the available thin-SVD dimension. *Spectral readout* selects the first $k_i = \min(r_i, q)$ components, providing a history-free adapter within each client's rank budget.

An optional *client-aware readout* (CAR) uses historical subspace alignment to tailor component selection to a client. It retains a shared core $\mathcal{G}_i = \{1, \ldots, \min(\lfloor \gamma r_i \rfloor, k_i)\}$, where $\gamma \in [0, 1]$. Let H_i be the latest available orthonormal basis of the client's historical update subspace. CAR fills the remaining slots with the $k_i - |\mathcal{G}_i|$ non-core directions having the highest scores $a_{ij} = \|H_i^\top u_j\|_2^2$. Without history, it uses spectral order. For either readout, the selected index set $\mathcal{I}_i$ defines factors:

$$B_{i,\mathrm{init}} = U_s[:, \mathcal{I}_i] \Sigma_{\mathcal{I}_i}^{1/2}, \qquad A_{i,\mathrm{init}} = \Sigma_{\mathcal{I}_i}^{1/2} V[:, \mathcal{I}_i]^\top. \tag{12}$$

Controlled comparisons in Section 4.3 separate the contributions of aggregation and readout. CAR provides additional gains in the evaluated GLUE settings, while spectral readout performs better in the SuperNI stale-history audit (Appendix F). Appendix B gives the complete algorithm.

4 Experiments

We ask three questions: does GLoRA improve learning across heterogeneous clients, how do the shared subspace and readout contribute, and what server cost does the representation require?

4.1 Experimental Setup

Tasks and models. We evaluate MNLI-m, MNLI-mm, SST-2, QQP, and QNLI from GLUE (Wang et al., 2018) with RoBERTa-base (Liu et al., 2019). For cross-task learning, we use SuperNI (Wang et al., 2022) with the Data-Juicer LLaMA-1B-dj-refine-150B backbone (Chen et al., 2024), 50 training clients, and 20 held-out task clients. Metrics are accuracy for GLUE and ROUGE-L (Lin, 2004) for SuperNI. LoRA adapts query and value projections. Additional evaluations use Qwen2-7B (Yang et al., 2024) and Gemma-2-9B (Team et al., 2024).

Table 1: **Performance comparison on GLUE tasks under different Dirichlet splits with homogeneous ranks.** Best and second-best results are bold and underlined.

Split	Method	MNLI-m	MNLI-mm	SST-2	QQP	QNLI	AVG
Dir(0.1)	FedIT	71.74±0.45	71.68±0.37	93.69±0.24	83.98±0.37	83.91±0.20	81.00
	FFA-LoRA	67.81±0.39	69.27±0.46	93.92±0.21	82.31±0.33	81.95±0.29	79.05
	FedEx-LoRA	71.95±0.48	71.97±0.48	**95.27±0.17**	84.27±0.37	83.17±0.35	81.33
	FedSA-LoRA	53.76±0.93	53.11±1.04	75.00±0.82	63.18±0.90	52.37±0.98	59.48
	FedRot-LoRA	70.53±0.47	71.05±0.44	93.48±0.31	82.68±0.52	83.21±0.38	80.19
	FLoRG	69.16±0.53	70.38±0.50	93.08±0.39	81.97±0.61	82.43±0.45	79.40
	GLoRA (Ours)	**72.89±0.41**	**72.66±0.38**	94.49±0.26	**85.07±0.29**	**84.42±0.29**	**81.91**
Dir(0.5)	FedIT	85.55±0.27	85.34±0.33	94.96±0.19	86.28±0.31	89.95±0.29	88.42
	FFA-LoRA	82.48±0.45	83.12±0.29	95.18±0.22	83.42±0.27	89.07±0.31	86.65
	FedEx-LoRA	84.66±0.37	84.64±0.30	95.64±0.15	86.28±0.24	90.94±0.29	88.43
	FedSA-LoRA	84.60±0.35	85.31±0.33	93.37±0.31	85.80±0.31	90.14±0.27	87.84
	FedRot-LoRA	83.13±0.36	84.02±0.34	94.71±0.24	85.40±0.43	89.62±0.31	87.38
	FLoRG	81.85±0.42	83.45±0.39	94.60±0.28	84.37±0.49	89.04±0.35	86.66
	GLoRA (Ours)	**85.71±0.31**	**85.73±0.34**	**96.33±0.13**	**87.21±0.32**	**91.82±0.25**	**89.36**

Federated settings. GLUE uses three rank-eight clients with full participation and Dirichlet skew $\alpha \in \{0.1, 0.5\}$, or 50 clients with 50% participation, $\alpha = 0.5$, and ranks 8–128 under normal, uniform, or heavy-tail distributions (Appendix F.1). GLUE runs 1,000 rounds with ten local steps; SuperNI runs 30 rounds with 10% participation and one local epoch. Main experiments use seeds 42–44. Budget sweeps vary this fraction; realized ranks are capped by the available span.

Comparisons. Homogeneous-rank baselines are FedIT, FFA-LoRA, FedEx-LoRA, FedSA-LoRA, FLoRG, and FedRot-LoRA; heterogeneous-rank baselines are HetLoRA and FlexLoRA (Section 5). FlexLoRA averages induced update matrices and applies truncated SVD for rank-specific redistribution, and thus serves as our update-average+SVD comparator. Unless otherwise stated, GLoRA uses CAR. In controlled rank-matched comparisons, GLoRA instead uses spectral readout to isolate the aggregation rule; separate spectral-versus-CAR ablations quantify CAR's additional contribution.

4.2 Learning across Heterogeneous Clients

Data heterogeneity. GLoRA achieves the highest average accuracy under both label-skew settings in Table 1, improving over the strongest baseline averages by 0.58 and 0.93 points. It leads on four of five tasks at $\mathrm{Dir}(0.1)$ and all five at $\mathrm{Dir}(0.5)$. These gains show that a shared aggregation space is useful even when client ranks match. Figure 3 illustrates the learning dynamics on SST-2: GLoRA reaches a high-accuracy plateau by roughly round 60 and maintains it through the displayed rounds.

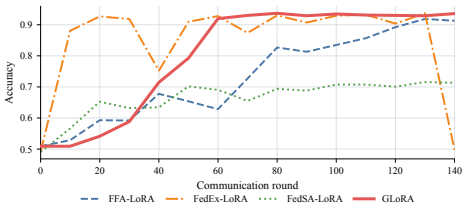


Figure 3: **Training dynamics under label skew.** SST-2 accuracy during different communication rounds at $\mathrm{Dir}(0.1)$. GLoRA maintains a high-accuracy plateau after its initial improvement.

Resource heterogeneity. GLoRA leads on 14 of 15 task–rank-distribution pairs in Table 2. Average gains over FlexLoRA are 2.55, 1.77, and 2.85 points for normal, uniform, and heavy-tail rank populations. The improvement across all three distributions shows that the shared server representation accommodates a broad range of client capacities.

Sparse participation and unseen tasks. With only 10% of clients participating per round, GLoRA achieves the highest ROUGE-L in all six seen/unseen settings (Table 3). Mean gains over the strongest baseline in each setting are 1.30 points on seen clients and 1.52 points on unseen clients. Figure 4 breaks these results down by task category. For example, GLoRA improves sentiment analysis under uniform ranks and language identification under heavy-tail ranks, illustrating benefits across both training and held-out task types.

Table 2: **Performance comparison on GLUE tasks with heterogeneous LoRA ranks under** $\mathrm{Dir}(0.5)$ **label skew**. Best and second-best results are bold and underlined.

Distribution	Method	MNLI-m	MNLI-mm	SST-2	QQP	QNLI	Average
Normal	HetLoRA	83.92±0.34	84.22±0.31	**95.07±0.18**	83.52±0.36	91.07±0.24	87.56
	FlexLoRA	81.63±0.43	82.75±0.39	93.32±0.27	80.71±0.48	90.54±0.29	85.79
	GLoRA (Ours)	**85.57±0.28**	**86.25±0.33**	94.04±0.23	**84.45±0.34**	**91.40±0.22**	**88.34**
Uniform	HetLoRA	82.74±0.38	82.96±0.36	94.61±0.20	82.99±0.41	89.57±0.31	86.57
	FlexLoRA	81.89±0.46	82.74±0.42	93.67±0.18	78.57±0.57	89.54±0.34	85.28
	GLoRA (Ours)	**83.02±0.35**	**83.66±0.32**	**95.07±0.17**	**83.39±0.39**	**90.12±0.28**	**87.05**
Heavy-tail	HetLoRA	85.37±0.30	85.29±0.34	94.18±0.25	83.96±0.37	90.81±0.26	87.92
	FlexLoRA	82.71±0.44	82.72±0.41	90.09±1.06	83.17±0.43	88.54±0.36	85.45
	GLoRA (Ours)	**85.59±0.27**	**85.84±0.29**	**94.50±0.21**	**84.70±0.33**	**90.89±0.25**	**88.30**

Table 3: **ROUGE-L comparison on SuperNI under sparse participation and heterogeneous LoRA ranks.** Results are reported on both in-domain training clients and unseen evaluation clients.

Method	In-domain Client Performance			Unseen Task Generalization		
	Normal	Uniform	Heavy-tail	Normal	Uniform	Heavy-tail
HetLoRA	50.31±0.42	48.59±0.51	49.49±0.48	35.95±0.63	33.04±0.78	33.88±0.72
FlexLoRA	48.72±0.57	42.53±0.84	44.10±0.76	35.12±0.71	29.21±1.02	31.11±0.91
GLoRA (Ours)	**52.26±0.36**	**49.61±0.44**	**50.42±0.41**	**37.97±0.58**	**34.95±0.69**	**34.50±0.73**

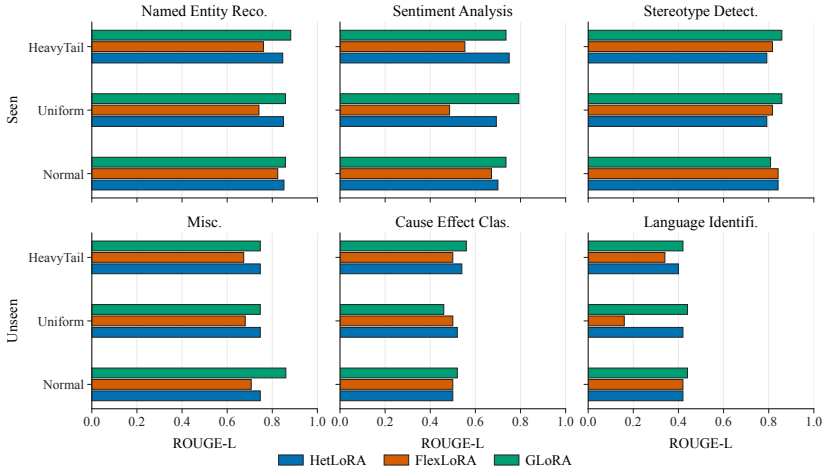


Figure 4: **Generalization across task categories.** SuperNI ROUGE-L on selected seen (top) and unseen (bottom) categories under three client-rank distributions. Scores here use a 0–1 scale.

A participation sweep with 5, 25, and 50 active clients improves both seen and unseen ROUGE-L over FlexLoRA at each level (Appendix D). On Qwen2-7B and Gemma-2-9B, GLoRA improves QNLI and QQP accuracy over FedEx-LoRA by 0.79–0.86 points (Appendix F.1).

4.3 Understanding the Shared Representation

From equivalent factors to a common frame. The paired audits in Section 2.2 establish the effect of input representation. We next compare aggregation rules under shared initialization, participation, and rank budget. Canonical-FedIT uses deterministic balanced factors before factor averaging; rank-matched FlexLoRA averages updates and applies SVD; GLoRA projects updates into a shared frame and uses spectral readout. Table 4 reports a single-seed, 30-round SST-2 comparison. GLoRA reaches 92.63% accuracy with CAR disabled. Canonical factor averaging retains the cross-client

Table 4: **Shared-subspace aggregation improves performance without CAR.** GLoRA and rank-matched FlexLoRA use spectral readout at the same server rank; Canonical-FedIT canonicalizes factors before averaging. All methods share initialization and participation.

Method	Accuracy (%)	Loss ↓	Worst-client acc. (%)
FedIT	90.71	0.3213	80.46
Canonical-FedIT	83.26	0.4286	62.07
Rank-matched FlexLoRA	91.40	0.2957	82.76
GLoRA w/o CAR	**92.63**	**0.2897**	**83.91**

terms in Eq.(3), whereas shared coordinates combine the projected updates themselves. The result shows that the shared update space can benefit learning with a history-free readout.

Choosing the server rank. A smaller reference space reduces the server representation while preserving gauge invariance. Figure 5 shows the associated learning trade-off. GLUE accuracy generally improves with the budget; SuperNI seen-task ROUGE-L rises from 0.444 at $\rho = 0.1$ to 0.523 at $\rho = 1$. Unseen-task scores vary less regularly, indicating that additional retained directions have task-dependent value.

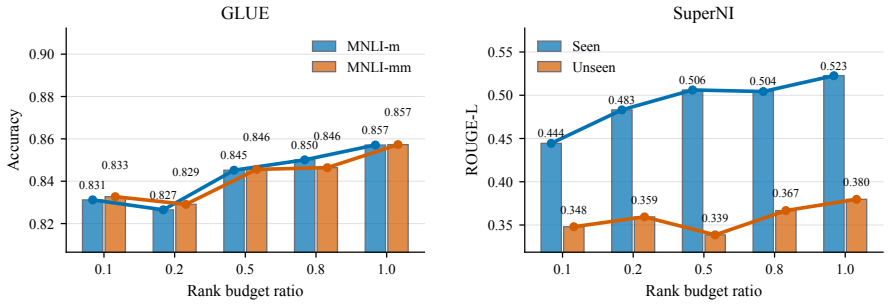


Figure 5: **Learning with a limited server rank.** Performance as the requested rank budget ρr_Σ varies on GLUE and SuperNI. Gauge invariance holds throughout the budget sweep.

The geometry audit helps explain why moderate budgets remain useful. With normal-rank SuperNI uploads, $R \approx 0.5 r_\Sigma$ retains 99.792%–99.993% of average-update energy across cohorts of 5–50 clients. The client union span is large, but much of the average's energy lies in the selected space. Rank-matched SVD achieves lower reconstruction error by optimizing that objective directly; GLoRA selects its space through client-subspace overlap. Appendix D examines this distinction across budgets and overlap levels.

Adapting the readout to clients. Table 5 separates projected aggregation from CAR. Spectral GLoRA improves all six cells over rank-matched update averaging; CAR adds a further 0.31–0.76 accuracy points, averaging 0.53 points. Thus, both the shared aggregate and the selection of client components contribute to the GLUE results. The core-ratio sweep and history audit in Appendix F clarify when historical alignment is helpful: CAR benefits these GLUE settings, while spectral readout performs better in the reported SuperNI stale-history comparison.

Table 5: **Aggregation and readout ablation on GLUE.** The heterogeneous setting uses uniform ranks. All methods use the same server-rank budget.

	Homogeneous Dir(0.1)			**Heterogeneous** Dir(0.5)		
Aggregation / readout	**MNLI-m**	**SST-2**	**QNLI**	**MNLI-m**	**SST-2**	**QNLI**
Update average / spectral	71.63	93.98	83.18	81.47	94.06	88.31
GLoRA / spectral	72.08	94.22	83.72	82.84	94.47	89.36
GLoRA / CAR	**72.84**	**94.53**	**84.37**	**83.15**	**94.96**	**90.03**

4.4 Communication and Server Cost

GLoRA communicates $r_i(d_{\text{out}} + d_{\text{in}})$ scalars per layer, matching the payload of ordinary LoRA factors. For rank eight over 24 RoBERTa adapters, this is 1.125 MiB per client in each direction using FP32. The thin-matrix construction also reduces server work when the total active rank is small relative to the layer dimensions.

Table 6: **Server aggregation time and peak memory for 24 rank-eight RoBERTa adapters.**

	5 active clients			50 active clients		
Method	**Time (ms)**	**Peak (MiB)**	**Speedup**	**Time (ms)**	**Peak (MiB)**	**Speedup**
FlexLoRA	2197.5	74.5	1.0×	2578.8	125.1	1.0×
GLoRA	**57.3**	**14.5**	38.3×	**496.3**	**109.7**	5.2×

Table 6 reports 38.3× and 5.2× faster server aggregation for 5 and 50 active clients, respectively. The advantage narrows as the total active rank approaches the layer dimensions. Appendix G gives the complexity, larger-cohort measurements, and CAR history storage.

5 Related Work

Parameter-efficient adaptation. Adapters (Houlsby et al., 2019), prefix and prompt tuning (Li & Liang, 2021; Lester et al., 2021), and multiplicative modulation (Liu et al., 2022) reduce the cost of adapting pretrained models. LoRA (Hu et al., 2022) parameterizes updates with low-rank factors; AdaLoRA (Zhang et al., 2023), QLoRA (Dettmers et al., 2023), and DoRA (Liu et al., 2024) further develop rank allocation, quantized fine-tuning, and weight decomposition. Our focus is how such local representations support aggregation across clients.

Federated LoRA aggregation. FedIT (Zhang et al., 2024) averages factors; FFA-LoRA (Sun et al., 2024), FedSA-LoRA (Guo et al., 2025), and LoRA-FAIR (Bian et al., 2025) modify factor training, sharing, or initialization. FedEx-LoRA (Singhal et al., 2025) uses residual correction, while FLoRA (Wang et al., 2024) stacks factors for exact aggregation. FLoRG (Meng et al., 2026) adopts a single-factor Gram parameterization and uses Procrustes alignment to mitigate decomposition drift across rounds. FedRot-LoRA (Zhang et al., 2026) addresses rotational misalignment through orthogonal alignment followed by factor averaging, with a common rank across clients. Our work studies aggregation invariance under independent, general invertible reparameterizations of standard LoRA updates. GLoRA realizes this requirement by constructing a shared frame from invariant client projectors and averaging projected update coordinates within it. This construction guarantees invariance at finite server rank, accommodates heterogeneous client ranks, and recovers the exact update average under sufficient subspace coverage.

Heterogeneous ranks and redistribution. HetLoRA (Cho et al., 2024) supports heterogeneous adapters through rank adaptation and sparsity-weighted aggregation. FlexLoRA (Bai et al., 2024) averages update matrices and uses SVD for rank-specific redistribution. These methods address rank heterogeneity, leaving consistency under equivalent client representations implicit. We make this requirement explicit through gauge invariance and expose representation dependence through controlled audits that hold every client update fixed. This perspective explains the invariance of update averaging and factor stacking, while revealing the coordinate dependence of factor averaging.

6 Conclusion

Gauge invariance captures a simple requirement: equivalent client updates should lead to the same server aggregate. Our controlled audits demonstrate why factor representations matter in federated LoRA. GLoRA addresses this through a shared subspace and common update coordinates, preserving invariance at finite rank and recovering exact averaging under sufficient coverage. Across GLUE and SuperNI, the resulting representation supports effective learning with heterogeneous data, client ranks, and tasks. Shared subspaces thus provide a compact interface between local adaptation and server aggregation.

AI USE STATEMENT

We used generative AI tools to polish an existing author-written draft, improving clarity, readability, and linguistic expression. We also used AI tools to assist with code development. The authors reviewed and verified all AI-assisted text and code and take full responsibility for the final content of the paper.

REPRODUCIBILITY STATEMENT

The source code has been uploaded as supplementary material to facilitate reproduction of the experimental results and further research.

REFERENCES

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

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

Daoyuan Chen, Yilun Huang, Zhijian Ma, Hesen Chen, Xuchen Pan, Ce Ge, Dawei Gao, Yuexiang Xie, Zhaoyang Liu, Jinyang Gao, et al. Data-juicer: A one-stop data processing system for large language models. In *Companion of the 2024 International Conference on Management of Data*, pp. 120–134, 2024.

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

Tim Dettmers, Artidoro Pagnoni, Ari Holtzman, and Luke Zettlemoyer. QLoRA: Efficient finetuning of quantized llms. In *Advances in Neural Information Processing Systems*, volume 36, pp. 10088–10115, 2023.

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

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

Edward J. Hu, Yelong Shen, Phillip Wallis, Zeyuan Allen-Zhu, Yuanzhi Li, Shean Wang, Lu Wang, and Weizhu Chen. LoRA: Low-rank adaptation of large language models. In *International Conference on Learning Representations*, 2022.

Brian Lester, Rami Al-Rfou, and Noah Constant. The power of scale for parameter-efficient prompt tuning. In *Proceedings of the 2021 conference on empirical methods in natural language processing*, pp. 3045–3059, 2021.

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

Chin-Yew Lin. ROUGE: A package for automatic evaluation of summaries. In *Text Summarization Branches Out*, pp. 74–81, 2004.

Haokun Liu, Derek Tam, Mohammed Muqeeth, Jay Mohta, Tenghao Huang, Mohit Bansal, and Colin A Raffel. Few-shot parameter-efficient fine-tuning is better and cheaper than in-context learning. In *Advances in Neural Information Processing Systems*, volume 35, pp. 1950–1965, 2022.

Shih-Yang Liu, Chien-Yi Wang, Hongxu Yin, Pavlo Molchanov, Yu-Chiang Frank Wang, Kwang-Ting Cheng, and Min-Hung Chen. DoRA: Weight-decomposed low-rank adaptation. In *Forty-first International Conference on Machine Learning*, 2024.

Yinhan Liu, Myle Ott, Naman Goyal, Jingfei Du, Mandar Joshi, Danqi Chen, Omer Levy, Mike Lewis, Luke Zettlemoyer, and Veselin Stoyanov. RoBERTa: A robustly optimized bert pretraining approach. *arXiv preprint arXiv:1907.11692*, 2019.

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

Chuiyang Meng, Ming Tang, and Vincent Wong. Florg: Federated fine-tuning with low-rank gram matrices and procrustes alignment. In *International Conference on Learning Representations*, volume 2026, pp. 82804–82824, 2026.

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

Youbang Sun, Zitao Li, Yaliang Li, and Bolin Ding. Improving LoRA in privacy-preserving federated learning. In *The Twelfth International Conference on Learning Representations*, 2024.

Gemma Team, Morgane Riviere, Shreya Pathak, Pier Giuseppe Sessa, Cassidy Hardin, Surya Bhupatiraju, Léonard Hussenot, Thomas Mesnard, Bobak Shahriari, Alexandre Ramé, et al. Gemma 2: Improving open language models at a practical size. *arXiv preprint arXiv:2408.00118*, 2024.

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*, pp. 353–355, 2018.

Yizhong Wang, Swaroop Mishra, Pegah Alipoormolabashi, Yeganeh Kordi, Amirreza Mirzaei, Atharva Naik, Arjun Ashok, Arut Selvan Dhanasekaran, Anjana Arunkumar, David Stap, et al. Super-NaturalInstructions: Generalization via declarative instructions on 1600+ nlp tasks. In *Proceedings of the 2022 conference on empirical methods in natural language processing*, pp. 5085–5109, 2022.

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

An Yang, Baosong Yang, Binyuan Hui, Bo Zheng, Bowen Yu, Chang Zhou, Chengpeng Li, Chengyuan Li, Dayiheng Liu, Fei Huang, Guanting Dong, Haoran Wei, Huan Lin, Jialong Tang, Jialin Wang, Jian Yang, Jianhong Tu, Jianwei Zhang, Jianxin Ma, Jianxin Yang, Jin Xu, Jingren Zhou, Jinze Bai, Jinzheng He, Junyang Lin, Kai Dang, Keming Lu, Keqin Chen, Kexin Yang, Mei Li, Mingfeng Xue, Na Ni, Pei Zhang, Peng Wang, Ru Peng, Rui Men, Ruize Gao, Runji Lin, Shijie Wang, Shuai Bai, Sinan Tan, Tianhang Zhu, Tianhao Li, Tianyu Liu, Wenbin Ge, Xiaodong Deng, Xiaohuan Zhou, Xingzhang Ren, Xinyu Zhang, Xipin Wei, Xuancheng Ren, Xuejing Liu, Yang Fan, Yang Yao, Yichang Zhang, Yu Wan, Yunfei Chu, Yuqiong Liu, Zeyu Cui, Zhenru Zhang, Zhifang Guo, and Zhihao Fan. Qwen2 technical report, 2024.

Haoran Zhang, Dongjun Kim, Seohyeon Cha, and Haris Vikalo. Fedrot-loRA: Mitigating rotational misalignment in federated loRA. In *Forty-third International Conference on Machine Learning*, 2026. URL https://openreview.net/forum?id=2X8Qi3VdjA.

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

Qingru Zhang, Minshuo Chen, Alexander Bukharin, Pengcheng He, Yu Cheng, Weizhu Chen, and Tuo Zhao. Adaptive budget allocation for parameter-efficient fine-tuning. In *International Conference on Learning Representations*, 2023.

A NOTATION AND PROOFS

A.1 NOTATION AND ADMISSIBLE RANKS

We analyze one adapted layer and suppress round indices where possible. The effective client update is $\Delta W_i = B_i A_i$, with any LoRA scale included in the factors. The weights of participating clients are positive and sum to one. The principal symbols are summarized in Table 7.

Table 7: Notation for one layer and one round.

Symbol	Meaning
$d_{\text{out}}, d_{\text{in}}$	Output and input dimensions
$\mathcal{C}, p_i, r_i$	Active clients, aggregation weights, and client ranks
B_i, A_i	Local LoRA factors
$U_i, T_i, \widehat{A}_i$	Reduced QR factors and coordinates, $\widehat{A}_i = T_i A_i$
$P_i = U_i U_i^\top$	Client subspace projector
$X = [\sqrt{p_i} U_i]_i$	Weighted concatenated client bases
$K = XX^\top$	Weighted sum of client projectors
$r_\Sigma, r_\cup$	Sum of client ranks and dimension of their union span
R, ρ	Realized server rank and requested rank-budget ratio
$U_{\text{ref}}, P_{\text{ref}}$	Reference basis and its projector
Z_i, Z_g	Client and aggregated coordinates in the reference frame
$M, \Delta W_g$	Update average and its GLoRA projection
O, Σ, V, U_s	Coordinate SVD factors and lifted left singular vectors
$H_i, \gamma, \mathcal{I}_i$	Client history basis, core ratio, and readout index set

For a minimal nonzero rank-r_i factorization, B_i is full column rank and A_i full row rank, so their product has the same column space as B_i. If B_i is rank deficient, a rank-revealing decomposition supplies a basis for its actual column space; the proofs below use that basis. For a nonminimal factorization with a rank-deficient A_i, the projector of B_i is still invariant under invertible factor reparameterization, but can contain directions absent from the product. Identifying that projector with the product's column space requires a minimal factorization. Zero updates can be represented with an empty basis.

The nonzero eigenspace of K has dimension $r_\cup \le \min(d_{\text{out}}, r_\Sigma)$. A requested budget is rounded to an integer and capped by this available dimension. If the request covers $r_\cup$, all nonzero directions are retained. For an empty aggregate, the server returns a zero update. The budget sweeps specify a requested rank; the realized basis width is capped by the available subspace dimension.

For truncated eigenspaces, either an eigengap at R or a selection convention depending only on K is required. Once the projector is selected, any orthonormal basis of its range gives the same represented aggregate. Readout selects at most the available spectral dimension, producing an adapter within the client's rank budget.

A.2 GAUGE INVARIANCE

Proof. For each invertible Q_i, the factors $\widetilde{B}_i = B_i Q_i$ and $\widetilde{A}_i = Q_i^{-1} A_i$ satisfy $\widetilde{B}_i \widetilde{A}_i = \Delta W_i$ and $\operatorname{col}(\widetilde{B}_i) = \operatorname{col}(B_i)$. Orthonormal bases of this column space differ by an orthogonal matrix: $\widetilde{U}_i = U_i S_i$. Consequently,

$$\widetilde{P}_i = U_i S_i S_i^\top U_i^\top = P_i, \qquad \widetilde{K} = \sum_i p_i \widetilde{P}_i = K.$$

The stated selection convention gives $\widetilde{P}_{\text{ref}} = P_{\text{ref}}$. Moreover, $\widetilde{M} = M$. Eq.(10) therefore yields

$$\widetilde{\Delta W}_g = \widetilde{P}_{\text{ref}} \widetilde{M} = P_{\text{ref}} M = \Delta W_g.$$

If the stored reference basis rotates as $\widetilde{U}_{\text{ref}} = U_{\text{ref}} S$, its coordinates rotate as $\widetilde{Z}_g = S^\top Z_g$, leaving their product unchanged. Thus, the server state represents the same update under either basis. □

A.3 Exact recovery and the finite-rank residual

Proof. Let $\mathcal{S} = \mathrm{span}([U_i]_i)$. Positive weights imply $\mathrm{range}(K) = \mathrm{range}(X) = \mathcal{S}$. Retaining the full nonzero eigenspace of K thus gives $P_{\mathrm{ref}} U_i = U_i$ for every client. Hence

$$\Delta W_g = \sum_i p_i P_{\mathrm{ref}} U_i \widehat{A}_i = \sum_i p_i U_i \widehat{A}_i = M.$$

For any smaller selected space, the same calculation gives $\Delta W_g = P_{\mathrm{ref}} M$, and subtracting from M gives $(I - P_{\mathrm{ref}})M$. □

A.4 The consensus objective

For an orthonormal reference basis U,

$$\mathrm{tr}(U^\top K U) = \sum_i p_i \|U^\top U_i\|_F^2.$$

The leading eigenspace maximizes this expression. Equivalently, over rank-R orthogonal projectors,

$$P_{\mathrm{ref}} \in \arg\min_P \sum_i p_i \|P - P_i\|_F^2,$$

because $\|P - P_i\|_F^2 = R + r_i - 2\,\mathrm{tr}(P P_i)$. The terms R and $\sum_i p_i r_i$ are fixed. This is a geometric consensus objective; it differs from minimizing reconstruction error of M.

B Algorithms

The backbone remains frozen during local adaptation. The server keeps a low-rank state and rank-compatible adapter initializations. Following the readout order, the histories available before the current uploads determine the next initializations; uploaded bases are then stored for future rounds. The set $\mathcal{D}_t$ denotes clients for which readout is prepared. Clients without history use spectral order.

Algorithm 1 GLoRA across communication rounds

Require: Frozen backbone W_0, initial adapters, client ranks $\{r_i\}$, rounds T, budget rule, core ratio γ

Initialize server state Ω, client adapters, and optional history cache $\{H_i\}$
for $t = 0, \ldots, T-1$ **do**
  Sample active clients $\mathcal{C}_t$ and set weights $\{p_i^t\}$
  for all $i \in \mathcal{C}_t$ in parallel **do**
    Send the latest prepared rank-r_i adapter to client i
    Perform local training on the client's data to obtain (B_i^t, A_i^t)
    Compute $B_i^t = U_i^t T_i^t$ and $\widehat{A}_i^t = T_i^t A_i^t$
    Upload $(U_i^t, \widehat{A}_i^t)$
  end for
  $\Omega \leftarrow$ ProjectedAggregate$(\{U_i^t, \widehat{A}_i^t, p_i^t\}, R_t)$
  Prepare rank-compatible adapters for $i \in \mathcal{D}_t$ from Ω and the cached histories
  Update $H_i \leftarrow U_i^t$ for active clients if CAR is enabled
end for
return the model induced by W_0 and Ω

Algorithm 2 PROJECTEDAGGREGATE

Require: Client bases and coordinates $\{U_i, \widehat{A}_i, p_i\}$, requested rank R
Form $X = [\sqrt{p_i}U_i]_i$ and cap R by $\mathrm{rank}(X)$
if $R = 0$ **then**
    return a zero server update
end if
Obtain U_{ref} from the leading R left singular vectors of X
Initialize $Z_g = 0$
for all active clients i **do**
    $Z_g \leftarrow Z_g + p_i(U_{\mathrm{ref}}^\top U_i)\widehat{A}_i$
end for
return $\Omega = (U_{\mathrm{ref}}, Z_g)$

Algorithm 3 RANKCOMPATIBLEREADOUT

Require: Server state (U_{ref}, Z_g), rank r_i, core ratio γ, optional history H_i
Compute a thin SVD $Z_g = O\Sigma V^\top$ and set $U_s = U_{\mathrm{ref}}O$
Set $q = \min(R, d_{\mathrm{in}})$ and $k_i = \min(r_i, q)$
if CAR is disabled or H_i is unavailable **then**
    $\mathcal{I}_i \leftarrow \{1, \ldots, k_i\}$
else
    $g_i \leftarrow \min(\lfloor \gamma r_i \rfloor, k_i)$, $\mathcal{G}_i \leftarrow \{1, \ldots, g_i\}$
    Compute $a_{ij} = \|H_i^\top u_j\|_2^2$ for $j \notin \mathcal{G}_i$
    $\mathcal{I}_i \leftarrow \mathcal{G}_i \cup \mathrm{TopK}_{k_i - g_i}\{a_{ij} : j \notin \mathcal{G}_i\}$
end if
Construct balanced factors using Eq.(12)
return the rank-compatible factors

C REPRESENTATION AUDITS AND TRAINING INTERVENTIONS

Paired-upload protocol. The primary audit uses RoBERTa-base on SST-2 with ten balanced-size non-IID clients, rank eight, 20% participation, and 24 query/value adapters. Clients train from shared round-start factors using a common model architecture and optimizer. We collect uploads at rounds 5, 10, and 20 under local-training budgets of 1, 10, and 20 steps, summarizing the audited rounds within each local-step setting. The paired audit uses one seed and the same 872 validation examples. Fixed-upload comparisons hold the client update matrices and aggregation weights fixed while changing their factor representations. The ten-round intervention instead follows two training branches from a shared initialization.

Server-update and prediction measurements. For each aggregation rule Φ, we compare the server updates induced by the original uploads ($\Delta W_\Phi^{\mathrm{nat}}$) and equivalent re-expressed uploads ($\Delta W_\Phi^{\mathrm{rep}}$). The normalized server-update change is

$$S_\Phi = \frac{\|\Delta W_\Phi^{\mathrm{nat}} - \Delta W_\Phi^{\mathrm{rep}}\|_F}{\sqrt{\sum_i p_i \|\Delta W_i\|_F^2}}, \tag{13}$$

where p_i is the aggregation weight of client i. Squared norms are summed over the 24 query/value adapters. Prediction disagreement is the fraction of the same 872 validation examples for which the two aggregated models predict different labels.

SVD refactorization. For $\Delta W_i = U_i\Sigma_i V_i^\top$, we use $B_i^{\mathrm{rep}} = U_i\Sigma_i^{1/2}$ and $A_i^{\mathrm{rep}} = \Sigma_i^{1/2}V_i^\top$. This construction distributes the singular values symmetrically between the factors. The maximum relative error in preserving a client update is below 1.1×10^{-14}. Table 8 reports the resulting server-update changes and prediction disagreement. GLoRA prediction disagreement is zero in all three local-step settings. A separate three-client, highly imbalanced partition covers three seeds and 18

ordinary-training checkpoints: mean update change is 0.462 for FedIT and 4.64×10^{-12} for GLoRA. Representation dependence is therefore also observed across seeds and client partitions.

Table 8: Balanced-SVD audit of fixed real uploads. All three GLoRA comparisons have zero prediction disagreement. Results aggregate audited rounds within each local-step setting.

Steps	FedIT update change	Prediction disagreement	GLoRA update change
1	0.3281	1.91%	2.83×10^{-15}
10	0.5347	14.95%	2.70×10^{-15}
20	0.5386	8.95%	2.67×10^{-15}

Round-start-relative alignment. To examine coordinate departure after ordinary local training, we search for an equivalent endpoint representation close to the shared round-start factors (B_g, A_g):

$$Q_i^\star \in \arg\min_{Q \in \mathrm{GL}(r)} \left[\frac{\|B_i Q - B_g\|_F^2}{\|B_i\|_F^2 + \|B_g\|_F^2 + \epsilon} + \frac{\|Q^{-1} A_i - A_g\|_F^2}{\|A_i\|_F^2 + \|A_g\|_F^2 + \epsilon} \right],$$

where $\epsilon > 0$ stabilizes the denominators. The aligned factors are $B_i^{\mathrm{align}} = B_i Q_i^\star$ and $A_i^{\mathrm{align}} = (Q_i^\star)^{-1} A_i$, so $B_i^{\mathrm{align}} A_i^{\mathrm{align}} = B_i A_i$. Alignment is applied after local training to measure coordinate departure while preserving the endpoint update.

Table 9 reports the normalized displacement between the trained and aligned factors, together with the resulting server-update changes and FedIT prediction disagreement. Alignment toward the initial factors produces smaller aggregate changes than SVD refactorization. In an identical-upload control, maximum coordinate departure is 2.2×10^{-18}, FedIT update change is 3.5×10^{-16}, GLoRA update change is 1.1×10^{-15}, and prediction disagreement is zero.

Table 9: Round-start-relative alignment of ordinary training endpoints. Equivalent factors are aligned toward the shared starting factors; each client update remains fixed.

Steps	Factor departure	FedIT update change	Prediction disagreement	GLoRA update change
1	1.30%	8.25×10^{-4}	7.68%	1.57×10^{-15}
10	2.24%	2.45×10^{-3}	10.24%	1.55×10^{-15}
20	2.35%	2.74×10^{-3}	11.51%	1.77×10^{-15}

Gauge-tangent motion. For a recorded optimizer step $(\delta B, \delta A)$, the first-order product-preserving directions have the form $(B\Omega, -\Omega A)$, where $\Omega \in \mathbb{R}^{r \times r}$. They satisfy $(B\Omega)A + B(-\Omega A) = 0$. We project the recorded step onto these directions by solving

$$\Omega^\star \in \arg\min_{\Omega} \big(\|\delta B - B\Omega\|_F^2 + \|\delta A + \Omega A\|_F^2 \big).$$

Under this Euclidean metric on factor steps, gauge-tangent motion accounts for approximately 1.56% of factor-motion energy. This measurement examines the recorded optimization process, complementing the endpoint alignment diagnostic. The fixed-upload audits separately measure the aggregate's sensitivity to equivalent representations.

Ten-round intervention. The original-upload and SVD-refactorized branches start from the same global checkpoint and share client participation and local data order. Refactorization occurs only after each local training phase. Table 10 reports the final single-seed results. FedIT ends with validation losses of 0.3136 and 0.3357, respectively, an increase of 0.0221 (7.1%), and 1.83% prediction disagreement. GLoRA retains identical predictions and 89.56% accuracy in both branches; its losses are 0.3093 and 0.3090.

Table 10: Ten-round paired training intervention. Prediction disagreement is measured against the natural-upload branch of the same aggregator.

Aggregator	Upload factors	Loss	Accuracy	Prediction disagreement
FedIT	Natural	0.3136	89.33%	0
FedIT	Balanced SVD	0.3357	89.11%	1.83%
GLoRA	Natural	0.3093	89.56%	0
GLoRA	Balanced SVD	0.3090	89.56%	0

Canonicalization and aggregation. The single-seed, 30-round SST-2 comparison in Table 4 uses shared initialization, client schedule, and server rank budget. Canonical-FedIT fixes a deterministic SVD factorization with symmetrically distributed singular values for each product before averaging the factors. Rank-matched FlexLoRA averages updates and applies SVD, while GLoRA uses shared-subspace aggregation with spectral readout. Canonical factor averaging retains the cross-client products in Eq.(3); update averaging and GLoRA instead combine complete local updates or their projections.

D Finite-Rank Geometry and Client Participation

Coverage and aggregate spectrum. We audit normal-rank SuperNI uploads with 5, 10, 25, or 50 active clients. Table 11 compares the sum of client ranks, their union rank, and the concentration of the average update's spectrum. The union span is nearly as large as the rank sum, while the average update concentrates much of its energy in a small number of directions.

Table 11: Union span and spectral concentration of the average update across active-cohort sizes.

Active	r_Σ	$r_\cup$	$r_\cup / r_\Sigma$	Effective rank of average	99%-energy rank
5	176	176	1.000	10.86	25.6
10	296	296	1.000	9.81	22–26
25	880	880	1.000	9.93	22–26
50	1816	1814	0.999	9.87	22–26

At a requested budget $R \approx 0.5 r_\Sigma$, Table 12 reports aggregate energy and client-update retention together with error to the pre-SVD average M. The rank-matched SVD result is the Frobenius-optimal rank-R approximation of M. Table 13 varies the budget within a five-client cohort. At full coverage, GLoRA recovers the average to approximately 10^{-14} relative error.

Table 12: Normal-rank SuperNI geometry audit at the half-rank-sum budget. Errors are relative to the same pre-SVD update average.

Active	Energy retained	p10 client retention	Minimum client retention	GLoRA error	Matched SVD error
5	99.792%	95.40%	89.83%	4.40%	1.19%
10	99.948%	96.78%	92.03%	2.20%	0.61%
25	99.977%	97.66%	94.92%	1.45%	0.24%
50	99.993%	98.78%	97.47%	0.80%	0.09%

Table 13: Server-rank budget sweep on a fixed five-client cohort. At full coverage, both representations recover the update average to numerical precision.

$R/r_\cup$	Energy retained	p10 retention	Minimum retention	GLoRA error	Matched SVD error
0.10	91.80%	68.82%	20.75%	28.18%	15.78%
0.25	98.03%	84.06%	51.13%	13.63%	3.42%
0.50	99.79%	95.40%	89.83%	4.40%	1.19%
0.75	99.97%	98.56%	97.07%	1.73%	0.45%
1.00	100%	100%	100%	numerical zero	numerical zero

Overlap strata. Table 14 groups preselected five-client subsets by high, medium, and low subspace overlap. At half-union-rank budgets, the reconstruction errors remain between 4.20% and 4.65%, and minimum client retention is about 92%.

Table 14: Five-client subsets stratified by subspace overlap, evaluated at quarter- and half-union-rank budgets.

Overlap	Error $0.25r_\cup$	Min. retention $0.25r_\cup$	Error $0.5r_\cup$	Min. retention $0.5r_\cup$
High	12.83%	70.70%	4.20%	91.95%
Medium	13.57%	73.88%	4.25%	91.96%
Low	14.09%	71.94%	4.65%	92.35%

Participation and learning. In a separate 30-round, single-seed SuperNI experiment, we vary participation among 50 clients from 5 to 25 to 50 active clients per round. Table 15 reports autoregressive ROUGE-L on a 0–1 scale. GLoRA improves both seen and unseen performance over FlexLoRA at all three participation levels.

Table 15: Single-seed, 30-round SuperNI participation sweep. ROUGE-L uses a 0–1 scale; differences are GLoRA minus FlexLoRA.

Active	GLoRA seen	FlexLoRA seen	Δ	GLoRA unseen	FlexLoRA unseen	Δ
5	0.4074	0.3849	+0.0225	0.2354	0.2144	+0.0210
25	0.3976	0.3456	+0.0520	0.2266	0.2160	+0.0106
50	0.3334	0.3206	+0.0128	0.1963	0.1748	+0.0215

Alternative geometry weights. Projector weighting assigns weight per subspace direction. Since $\mathrm{tr}(P_i) = r_i$, client i contributes total projector mass $p_i r_i$. We compare it with client-normalized projectors and an energy-aware diagnostic. The extreme population contains one rank-128 and seven rank-8 clients with equal client weights, giving $r_\Sigma = r_\cup = 184$ and $R = 92$. The high-rank identity is rotated across six clients and three checkpoints, producing 18 fixed-checkpoint units from one seed. Table 16 reports the geometric audit, and Table 17 gives separate single-seed, 30-round SuperNI task results. The weighting rules yield different trade-offs between average-update reconstruction and task performance across rank populations.

Table 16: Geometric selection under extreme rank imbalance: one rank-128 and seven rank-8 clients. Error is relative to the pre-SVD update average.

Geometry	High-rank mass share	p10 retention	Minimum retention	Error
Client-normalized projector	12.5%	88.7%	37.0%	0.179
Direction-weighted projector	69.6%	86.6%	68.1%	0.126
Energy-aware diagnostic	38.3%	99.3%	96.5%	0.030
Rank-matched FlexLoRA	—	95.4%	82.7%	0.020

Table 17: Geometry-weighting comparisons on SuperNI (one seed, 30 rounds). Entries are seen / unseen ROUGE-L on a 0–1 scale.

Rank population	Client-normalized seen / unseen	Direction-weighted seen / unseen	Energy-aware seen / unseen
One rank-128 + seven rank-8	0.3388 / 0.2083	**0.3914** / 0.2128	0.3316 / **0.2226**
Heavy-tail	**0.3913 / 0.2204**	0.3527 / 0.2137	0.3698 / 0.1734

E Client-Specific Directions and Row-Side Information

Unique client components. For a normal-rank SuperNI checkpoint at round 20, we examine 50 clients and 48 adapters, giving 2,400 client–layer observations. Let $P_{-i,\ell}$ project onto the union of the other clients' column spaces at layer ℓ. Define

$$\Delta W_{i,\ell}^{\perp} = (I - P_{-i,\ell})\Delta W_{i,\ell}, \quad u_{i,\ell} = \frac{\|\Delta W_{i,\ell}^{\perp}\|_F^2}{\|\Delta W_{i,\ell}\|_F^2}, \quad q_{i,\ell} = \frac{\|P_{R,\ell}\Delta W_{i,\ell}^{\perp}\|_F^2}{\|\Delta W_{i,\ell}^{\perp}\|_F^2}.$$

Table 18 reports unique energy and its retained fraction. To assess local utility, we remove only the evaluated client's weighted unique component, using $M_{-i} = M - p_i \Delta W_i^{\perp}$, and evaluate on that client's validation data. The 50 client-level loss changes are summarized in Table 19; 72% of clients have an increased loss. Table 20 compares the complete projected and spectral aggregates. Unique directions can therefore have local value even when their energy is small.

Table 18: Unique update energy and its retained fraction over 2,400 client–layer observations from one SuperNI checkpoint.

Statistic	Mean	Median	90th percentile	Maximum
Unique energy $u_{i,\ell}$	2.01%	1.72%	3.29%	10.52%
Unique retention $q_{i,\ell}$	10.00%	9.83%	15.46%	26.67%

Table 19: Local-loss increase after removing the evaluated client's weighted unique component, summarized across 50 clients.

Statistic	Mean	Median	90th percentile	Maximum
Local-loss increase	0.00377	0.00223	0.01105	0.03583

Table 20: Complete aggregate and client-specific deletion comparisons at the same SuperNI checkpoint. Deletion is performed separately for each evaluated client.

Aggregate	Mean local loss	Mean local ROUGE-L
FlexLoRA pre-SVD aggregate	1.32546	0.38712
Average minus own unique component	1.32923	0.38698
GLoRA, evaluated rank budget	**1.32542**	**0.38867**
Rank-matched FlexLoRA	1.32593	0.38586

Controlled row-side variation. We fix a 16-dimensional common left subspace and vary row-side alignment, gauge coordinates, and energy allocation. Table 21 compares server ranks 16 and 8. With full coverage, $Z_i = U_{\text{ref}}^{\top} \Delta W_i$ retains row-side information and all tested reconstructions are exact to numerical precision. At rank eight, the space-selection rule matters, including for energy-permuted updates.

Table 21: Controlled row-side changes within a common 16-dimensional left subspace. The rank-eight comparison tests finite-rank selection.

Controlled case	Rank	Direction projector energy / error	Energy-aware energy / error	Matched SVD error
Aligned / orthogonal rows	16	100% / numerical zero	100% / numerical zero	numerical zero
Gauge-rotated coordinates	16	100% / numerical zero	100% / numerical zero	numerical zero
Energy-permuted rows	16	100% / numerical zero	100% / numerical zero	numerical zero
Aligned / orthogonal rows	8	55.66% / 0.6659	86.36% / 0.3693	0.3693
Energy-permuted rows	8	51.65% / 0.6953	65.19% / 0.5900	0.5900

In real SuperNI uploads, column- and row-overlap Spearman correlations are 0.895, 0.910, and 0.950 for extreme, heavy-tail, and normal populations. We define strict mismatch as column overlap above its 75th percentile and row overlap below its 25th percentile. Table 22 reports the corresponding frequencies and errors. The normal population has no strict cases; its separate maximum-gap case has errors 0.4060, 0.0608, and 0.0516 for direction projectors, energy-aware geometry, and rank-matched SVD, respectively.

Table 22: Strict column/row overlap mismatches in real SuperNI uploads. No strict mismatch is observed in the normal population.

Ranks	Strict cases	Frequency	Direction error	Energy-aware error	Matched SVD error
Extreme	127 / 176,400	0.0720%	0.3799	0.0902	0.0680
Heavy-tail	113 / 176,400	0.0641%	0.3094	0.0613	0.0534
Normal	0 / 176,400	0%	—	—	—

F Client Readout and Additional Task Results

Component ablations. Table 5 reports the component ablation, and Table 23 isolates spectral versus client-aware readout with projected aggregation fixed. The six CAR gains average 0.53 accuracy points. The update-average row in Table 5 uses rank-matched SVD. The two GLoRA rows share the same projected aggregation rule.

Table 23: Readout ablation with projected aggregation fixed. Gains are accuracy points; the final row averages the six task settings.

Setting	Task	Spectral	CAR	Gain
Homo. Dir(0.1)	MNLI-m	72.08	72.84	+0.76
Homo. Dir(0.1)	SST-2	94.22	94.53	+0.31
Homo. Dir(0.1)	QNLI	83.72	84.37	+0.65
Hetero. Dir(0.5), uniform	MNLI-m	82.84	83.15	+0.31
Hetero. Dir(0.5), uniform	SST-2	94.47	94.96	+0.49
Hetero. Dir(0.5), uniform	QNLI	89.36	90.03	+0.67
Average	—	—	—	**+0.53**

Historical information. A separate single-seed SuperNI audit fixes the server state, client data, and local training protocol while varying the readout. Table 24 groups events by time since the last participation. Cached-to-current subspace alignment declines with the gap, and full-history CAR underperforms spectral readout on the reported group means. Spectral readout provides a history-free option for these settings.

Table 24: Matched-state, single-seed SuperNI history audit. Lower loss difference and higher ROUGE-L difference favor CAR.

Participation gap	Events	History alignment	Loss difference CAR − spectral	ROUGE-L difference CAR − spectral
1–2 rounds	32	0.555	+0.0088	-0.0276
3–5 rounds	31	0.414	+0.0451	-0.0132
More than 5 rounds	38	0.265	+0.0514	-0.0150

F.1 Training settings and further evaluations

Optimization and hardware. GLUE uses Adam, batch size 128, and ten local steps per round. Learning rates are selected for each method from $\{2\times10^{-4}, 5\times10^{-4}, 10^{-3}, 2\times10^{-3}, 5\times10^{-3}\}$. SuperNI uses Adam, batch size four, and one local epoch, with learning rates in $\{10^{-4}, 2\times10^{-4}, 5\times10^{-4}, 10^{-3}\}$. Main experiments use two RTX 5090 GPUs; Qwen2-7B and Gemma-2-9B experiments use one RTX PRO 6000.

Rank and data populations. We use five rank levels: 8, 16, 32, 64, and 128. The normal population assigns 5%, 15%, 60%, 15%, and 5% of clients to these levels. The uniform population assigns 20% to each level; the heavy-tail population assigns 70%, 10%, 10%, 5%, and 5%. Figure 6 displays these allocations. Figure 7 shows the homogeneous GLUE label partitions.

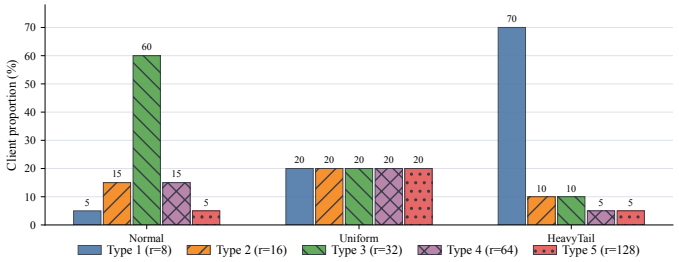


Figure 6: Client-rank populations used in heterogeneous-rank experiments.

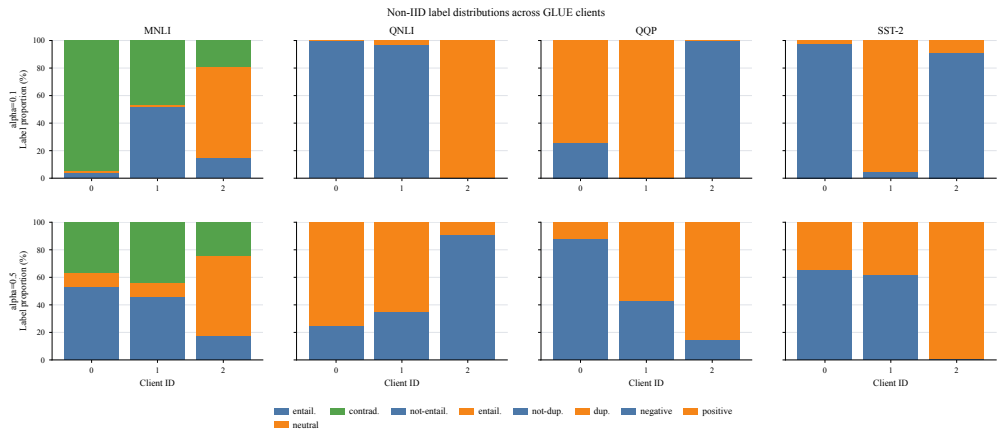


Figure 7: Label proportions across the three homogeneous-rank GLUE clients.

Table 25 reports the larger-backbone results under the homogeneous GLUE protocol. Figure 8 shows performance over the core ratio and learning rate. The preferred ratio varies across tasks and label partitions, with broader high-performing regions in several of the milder-skew settings.

Table 25: Larger-backbone accuracy under $\mathrm{Dir}(0.1)$, using the homogeneous-rank GLUE protocol.

Backbone	Method	QNLI	QQP
Qwen2-7B	FedIT	86.62	86.21
	FFA-LoRA	84.93	84.67
	FedEx-LoRA	<u>87.38</u>	<u>86.94</u>
	GLoRA	**88.24**	**87.73**
Gemma-2-9B	FedIT	87.25	86.74
	FFA-LoRA	85.61	85.18
	FedEx-LoRA	<u>88.02</u>	<u>87.42</u>
	GLoRA	**88.83**	**88.28**

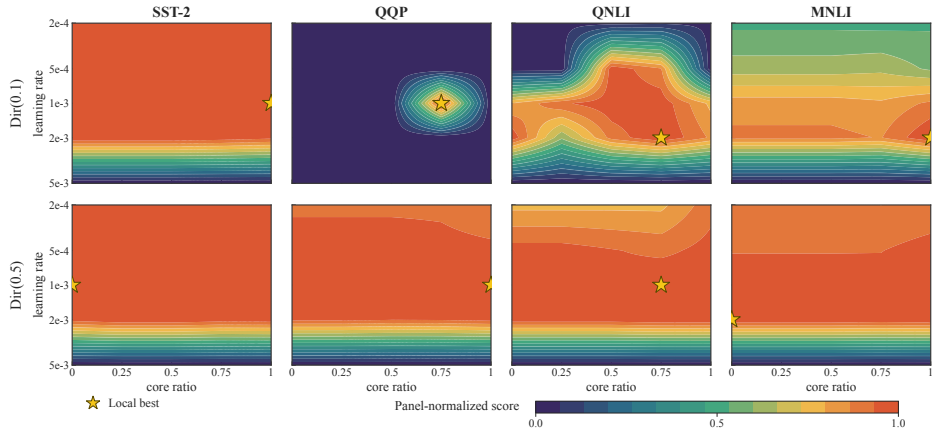


Figure 8: GLUE performance over the CAR core ratio and learning rate. Colors are normalized within each panel; stars mark panel optima.

G SERVER COMPUTATION, COMMUNICATION, AND STORAGE

Complexity. For one module, client QR and coordinate formation cost $O((d_{\text{out}} + d_{\text{in}})r_i^2)$. On the server, a Gram-based eigenspace construction uses $X^\top X$, costing $O(d_{\text{out}}r_\Sigma^2 + r_\Sigma^3)$, followed by $O(d_{\text{out}}r_\Sigma R)$ to obtain the reference basis. Forming $U_{\text{ref}}^\top U_i$ and aggregating coordinates costs $O(Rr_\Sigma(d_{\text{out}} + d_{\text{in}}))$. For $R \le d_{\text{in}}$, the coordinate SVD and lifting its left vectors cost $O(d_{\text{in}}R^2 + d_{\text{out}}R^2)$; more generally the SVD term is $O(Rd_{\text{in}}\min(R, d_{\text{in}}))$. CAR adds history-alignment work per readout client, approximately $O(d_{\text{out}}h_iq)$ for a history of width h_i and q spectral directions.

When $R = O(r_\Sigma)$ and r_Σ is below the layer dimensions, the leading aggregation and spectral-readout terms simplify to $O(r_\Sigma^2(d_{\text{out}} + d_{\text{in}}) + r_\Sigma^3)$. The FlexLoRA path evaluated here costs $O(d_{\text{out}}d_{\text{in}}r_\Sigma)$ to form the average and $O(d_{\text{out}}d_{\text{in}}\min(d_{\text{out}}, d_{\text{in}}))$ for SVD.

Communication. Table 26 uses FP32 and reports per-client, per-round payloads for 24 rank-eight RoBERTa adapters. Ordinary factor payloads halve to 0.5625 MiB in BF16. The FedEx-LoRA downlink corresponds specifically to the evaluated exact-residual redistribution protocol. GLoRA transmits the basis and coordinate factors.

Table 26: FP32 communication per client and round for 24 rank-eight RoBERTa adapters. The FedEx-LoRA row uses the evaluated exact-residual protocol.

Method	Uplink	Downlink	History
FedIT	1.125 MiB	1.125 MiB	No
FedEx-LoRA (exact residual)	1.125 MiB	55.125 MiB	No
HetLoRA	1.125 MiB	1.125 MiB	No
FlexLoRA	1.125 MiB	1.125 MiB	No
GLoRA, spectral readout	1.125 MiB	1.125 MiB	No
GLoRA + CAR	1.125 MiB	1.125 MiB	Optional

Kernel measurement protocol. Measurements use a common shape-matched GPU harness with warm-up and repeated timing, serially extrapolated across 24 adapters. They exclude networking, client training, and the shared frozen backbone. Tables 27 and 28 report timing and conservative peak memory for five methods. Client reduced QR takes approximately 2.5 ms per rank-eight client in the measured setting and can run in parallel across clients. A separate five-client normal heterogeneous-rank configuration takes 157.7 ms / 64.7 MiB for GLoRA and 2366.3 ms / 108.2 MiB for FlexLoRA.

Table 27: Server aggregation kernel time in milliseconds across 24 adapters. Local training and networking are excluded.

Active	FedIT	FedEx-LoRA	HetLoRA	GLoRA	FlexLoRA
5	2.8	10.8	14.6	57.3	2197.5
50	23.1	89.4	114.9	496.3	2578.8

Table 28: Conservative peak server memory (MiB) for the same kernel configurations.

Active	FedIT	FedEx-LoRA	HetLoRA	GLoRA	FlexLoRA
5	6.8	13.5	29.3	14.5	74.5
50	57.5	64.2	79.9	109.7	125.1

Increasing the active cohort. Table 29 extends the rank-eight measurement to 100 and 1,000 active clients. The requested budget is half the rank sum, with the realized reference dimension capped by the available span. At 1,000 clients, the request is 4,000 and the layer width is 768. The measured

implementation retains projector construction; at this scale it takes 2584.3 ms, compared with 2310.4 ms for dense accumulation and SVD.

Table 29: Server cost as the number of active rank-eight clients increases. The requested budget is capped by the available span.

Active	r_Σ	Nominal R	GLoRA time (ms)	FlexLoRA time (ms)	GLoRA peak (MiB)	FlexLoRA peak (MiB)
5	40	20	57.3	2197.5	14.5	74.5
50	400	200	496.3	2578.8	109.7	125.1
100	800	400	796.8	2319.4	217.2	181.3
1000	8000	4000	2584.3	2310.4	1356.0	1193.8

History storage. Spectral readout needs no persistent per-client history. For CAR, a rank-eight basis across 24 RoBERTa adapters contains 147,456 scalars per client. Eagerly caching the outgoing adapter adds 294,912 scalars per client. Table 30 summarizes BF16 tensor storage and the FP32 full-state footprint, excluding Python object overhead. Lazy readout avoids the eager-adapter cache, and limiting cached clients reduces the history footprint proportionally.

Table 30: CAR storage: BF16 history and eager-readout estimates, alongside the FP32 full-state footprint.

Registered clients / cached fraction	BF16 history basis (GiB)	BF16 history + eager readout (GiB)	Current FP32 full state (GiB)
1,000 / 100%	0.275	0.824	1.10
100,000 / 100%	27.47	82.40	109.86
1,000,000 / 100%	274.66	823.98	1098.63
1,000,000 / 5%	13.73	41.20	—
1,000,000 / 25%	68.66	205.99	—