WHEN DOES PERSONALIZATION HELP IN FEDERATED LORA FINE-TUNING?

Anonymous authors
Paper under double-blind review

ABSTRACT

Low-rank adaptation (LoRA) makes fine-tuning foundation models inexpensive. Pairing LoRA with federated learning lets clients adapt collaboratively without sharing data. The aim is increasingly a model personalized to each client, but how much personalization helps depends on how clients differ in the amount of local data, label skew, feature shift, and adapter rank. Existing federated LoRA methods each handle only some of these four axes. We propose O4A (All for One, One for All), a single aggregation rule for personalized federated LoRA. O4A distills a shared consensus across clients while preserving each client's personalization. Across the four axes on CIFAR-100 and DomainNet, O4A supports clients at different ranks and improves over existing baselines where personalization matters, giving a regime map of when personalization helps in federated LoRA.

1 INTRODUCTION

Foundation models are usually adapted to a task by fine-tuning, but updating all parameters is expensive. Low-rank adaptation (LoRA) (Hu et al., 2022) has become a standard lightweight alternative. When the adaptation data are private and spread across many parties, combining LoRA with federated learning (Zhang et al., 2024a) is natural. Each client trains a small adapter on its own data, and only the low-rank factors A and B are communicated (Bian et al., 2025; Bai et al., 2024; He et al., 2026). As these models are increasingly used as personal assistants, the aim shifts from a single shared model to one that fits each client. This per-client model is the goal of personalized federated learning (pFL).

How much personalization helps depends on how clients differ. Federated learning on non-IID data (clients' data are not identically distributed) already recognizes three axes of statistical difference (Li et al., 2022): the amount of local data a client holds, the skew of its labels toward some classes (label skew (Hsu et al., 2019)), and the shift of its features across domains (feature shift (Li et al., 2021b)). The axes that matter vary by deployment, from on-device assistants to clinical imaging across hospitals (Kairouz et al., 2021). LoRA adds a fourth axis, adapter rank, a property not of the data but of the setting. Devices in a real federation differ in compute and communication budgets (Kairouz et al., 2021; Li et al., 2020), so each client can afford only a certain adapter size. Once federated learning adopts LoRA, it cannot avoid heterogeneous rank (Cho et al., 2024). Existing federated LoRA methods each handle only some of these axes (Zhang et al., 2024a; Bai et al., 2024; He et al., 2026; Guo et al., 2025; Wang et al., 2026), and we study personalization across all four jointly.

Personalizing with LoRA along these four axes raises two problems specific to the low-rank form. Problem 1 is personalization versus rank compatibility. Factor-sharing methods keep a per-client part but disagree on which factor, A or B (Guo et al., 2025; He et al., 2026), and break when ranks differ, while the methods that handle different ranks fall back to one global model (Cho et al., 2024; Wang et al., 2024). Problem 2 is how much to personalize. Along the three data axes, whether a per-client model outperforms a shared one changes with the setting, and when a shared model helps or hurts is unclear. Section 3 shows the impact of each problem on existing methods.

This paper. Our contributions are:

- We frame personalized federated LoRA around four axes of heterogeneity and two problems specific to the low-rank form (Sections 2, 3).

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.

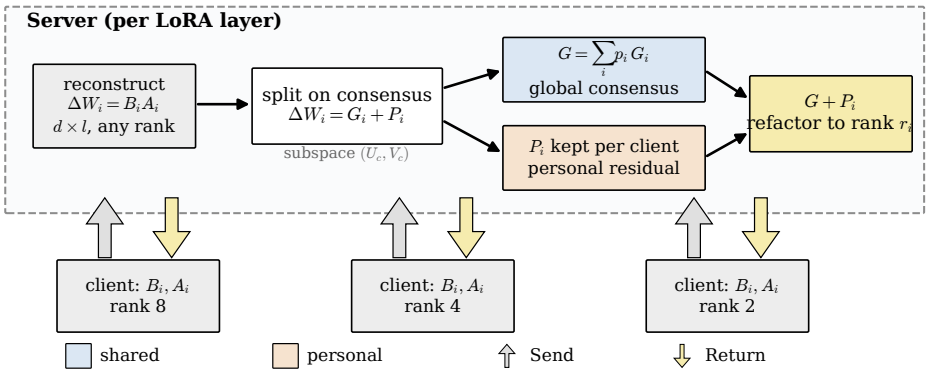

Figure 1: Overview of O4A (one LoRA layer, server side). Clients send factors at possibly different ranks. The server reconstructs each update in the ambient space and returns a shared consensus G plus each client's personal residual P_i, refactored to the client's own rank.

- We propose O4A (All for One, One for All; Figure 1), a single server-side rule that aggregates each client's reconstructed update into a shared consensus and a per-client residual. It addresses the heterogeneous rank that federated LoRA cannot avoid, and the other three axes of data heterogeneity (Section 4).
- On CIFAR-100 and DomainNet, O4A stays effective across all four axes, from their heterogeneous extremes to the near-homogeneous boundary, and we map when personalization helps (Section 5).

2 Related Work and Problem Setting

Personalized federated learning expresses each client's model as a shared part plus a client-specific one (Arivazhagan et al., 2019; Collins et al., 2021; Li et al., 2021a). Following the same decomposition, the update returned to client i splits into a shared component and a per-client residual,

$$\widetilde{\Delta W}_i = G + P_i, \tag{1}$$

where G is common across clients and P_i is client-specific.

In federated LoRA, client i's own update is the reconstructed product of its two low-rank factors A_i and B_i (Zhang et al., 2024a),

$$\Delta W_i = B_i A_i, \qquad B_i \in \mathbb{R}^{d \times r_i}, \quad A_i \in \mathbb{R}^{r_i \times l}, \tag{2}$$

where the rank r_i may differ across clients while $d \times l$, the size of the weight matrix being adapted, stays fixed. We write p_i for client i's aggregation weight among the K clients.

Existing federated LoRA methods fall into three levels by how they realize this split. Whether a method keeps a per-client residual separates the single-global methods (L1 and L2, $P_i = 0$) from the personalized ones (L3, $P_i \neq 0$). Whether G is defined for heterogeneous rank separates the two single-global levels. We note how each level fares along the four axes introduced in Section 1: data amount, label skew, feature shift, and adapter rank.

- **L1 (single-global, factor space)** averages A_i and B_i and returns the same model for every client, which needs a common rank (Zhang et al., 2024a). FedIT is the basic case (Zhang et al., 2024a), and LoRA-FAIR adds a server-side correction for the bias of averaging the two factors separately (Bian et al., 2025). L1 cannot personalize along the data axes and does not handle heterogeneous rank.
- **L2 (single-global, rank-aware)** defines G for heterogeneous ranks. Methods do this by padding factors to a common rank (Cho et al., 2024), aggregating the reconstructed products in the ambient space (Bai et al., 2024; Ramesh & Dass, 2026; Lee et al., 2025), stacking factors of differing widths (Wang et al., 2024), debiasing the aggregate (Peng et al., 2025),

or sketching client submatrices (Fang et al., 2026). L2 handles the rank axis but, with one model for all, cannot personalize along the data axes.

- **L3 (personalized)** keeps a shared G with a per-client residual, and personalizes along the data axes. The dominant realization is *factor-sharing*: the two LoRA factors are kept separate, and one is shared across clients while the other stays local (He et al., 2026; Guo et al., 2025; Sun et al., 2024). This forces a choice between A and B that the field disagrees on (Section 3.1), and a raw shared factor cannot be aggregated under heterogeneous rank. Other realizations place a local module beside a shared one (Bian et al., 2026), use a client hierarchy (Peng et al., 2026) or a hypernetwork (Gupta et al., 2026), or split the ambient update per client (Wang et al., 2026).

Adjacent lines vary client model capacity with subnetworks or low-rank factorization (Diao et al., 2021; Horváth et al., 2021; Yao et al., 2025) or place LoRA on a subset of layers by device budget (Zhang et al., 2025; 2024b), both orthogonal to how updates are aggregated under heterogeneous rank.

Across the three levels, no method covers all four axes at once. L2 handles heterogeneous rank but returns one model for all, so it does not personalize. Factor-sharing L3 personalizes but forfeits rank, the axis federated LoRA cannot avoid. These are the two problems described in Section 1, and no method escapes both.

3 Two problems in federated LoRA

Section 1 raised two problems, and Section 2 reviewed how existing methods handle them. To motivate our method, this section dissects both problems along the four axes.

3.1 Problem 1: personalization versus rank compatibility

To personalize in federated LoRA, a method must do two things at once. One is to aggregate a shared consensus while keeping each client's own residual, the other is to serve clients that train at different ranks.

Methods that keep a per-client residual share a single raw factor, but which factor should carry the shared information is not well-posed. For any invertible $R \in \mathbb{R}^{r\times r}$,

$$\Delta W = BA = (BR)(R^{-1}A), \tag{3}$$

so what a client holds in A versus B is only a change of basis, and only the product ΔW is basis-invariant (Zhang et al., 2026). The literature splits accordingly, with He et al. (2026) sharing B and others sharing A (Guo et al., 2025; Tian et al., 2024) (Appendix A.1, Figure 4). A raw factor also cannot be combined across ranks, since factors trained at different ranks have different shapes. These methods therefore produce no model under heterogeneous rank (Figure 2(d)). In deployment this forces a common rank on every client.

Methods that do support heterogeneous rank reconcile the shapes instead, by padding factors (Cho et al., 2024) or reconstructing the update (Bai et al., 2024), but they return one global model and discard the per-client residual. The tension is therefore not which factor to share but personalization against rank compatibility, and no existing method meets both requirements.

3.2 Problem 2: when and how much to personalize?

When personalization helps is a long-studied question in personalized FL (Li et al., 2021a; Arivazhagan et al., 2019), and the answer depends on how clients differ. We look at three conditions, namely scarce data, label skew, and feature shift, each measured on existing methods under a common protocol (Section 5, with per-level methods and the full setup in Appendix A.7).

Scarce data. Sharing should matter most when data are scarce, yet this is where existing methods struggle. As we reduce each client's data on CIFAR-100 (Krizhevsky, 2009) (Dirichlet label skew α=0.1, a strong skew; Figure 2(a)), `local-only` (each client trains only on its own data) drops sharply, since a client with little data cannot do well alone and has no way to borrow from others.

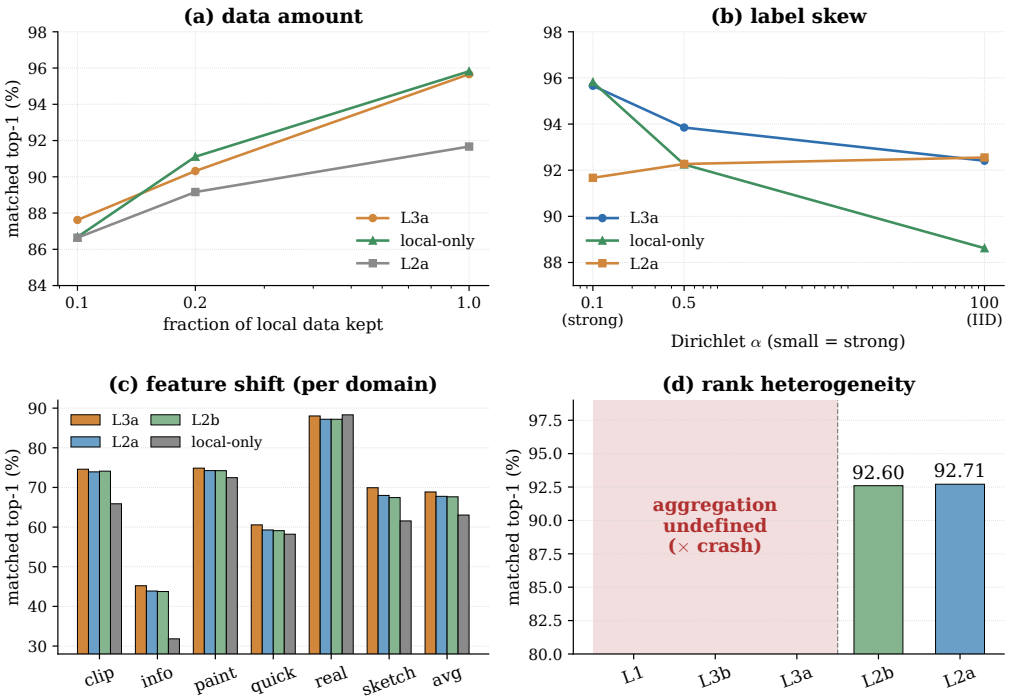

Figure 2: The four axes of heterogeneity on existing methods: **(a)** data amount and **(b)** label skew (Dirichlet α) on CIFAR-100, **(c)** feature shift on DomainNet (each domain restricted to 10% of its data), and **(d)** rank heterogeneity (ranks $8/2$) on CIFAR-100. Curves are labeled by level (L1–L3, Section 2).

The single-global methods (L2) do not rescue it, performing no better than training alone in this regime. Scarcity calls for both borrowing and specialization, yet neither training alone nor a single global model supplies both.

Label skew. How much clients' labels overlap changes which model performs better. Across the Dirichlet concentration α (Hsu et al., 2019), where a smaller α gives a stronger skew (Figure 2(b)), `local-only` and L3 lead at strong skew, and L2 leads near the homogeneous end. At strong skew, L2 sits *below* `local-only`, so aggregation lowers accuracy here. Sharing pays off only to the extent clients have structure in common (Li et al., 2021a; Arivazhagan et al., 2019). Under strong skew, each client adapts to a nearly disjoint set of classes, and the genuinely shared subspace is small. Forcing more directions than that into a shared model mixes conflicting updates. A method that keeps a shared component must therefore control *how much* it shares (Section 4).

Feature shift. Clients can also differ in their features (feature shift (Li et al., 2021b)). On DomainNet (Peng et al., 2019), six visual domains share the same classes (Figure 2(c)). To bring out the feature-shift effect, we restrict each domain to a fraction of its data (Appendix A.7). Under this restriction, `local-only` collapses on the hardest domains, most sharply on infograph, where a single client has too little signal to learn alone. The L2 methods stay flat across domains but cannot specialize to the hard ones. The gap is largest where training alone struggles.

Taken together, these measurements show that no method we compare handles all four axes. The L3 factor-sharing methods leave the factor choice unsettled and produce no model under heterogeneous rank, and L1 and L2 cannot personalize along the data axes. Meeting all four at once requires a single rule that aggregates the reconstructed update and controls how much clients share. These two requirements motivate O4A, the rule we build in Section 4.

4 METHOD

We propose O4A (Figure 1), a server-side aggregation rule for heterogeneous federated LoRA. O4A reconstructs each client's update $\Delta W_i = B_i A_i$ (equation 2) and aggregates these updates directly in the ambient $d \times l$ space. O4A then splits each update into a shared consensus and a per-client residual, and returns a personalized adapter at each client's own rank. We drop the layer index, since every step below is applied per layer. We write $p_i = 1/K$ for the aggregation weight and k for the consensus rank.

4.1 CONSENSUS SUBSPACE

O4A aggregates inside a k-dimensional subspace (U_c, V_c) that the clients share. These are the update directions many clients agree on. O4A pools each update's component in this subspace across clients and leaves the rest with the client that produced it. To find the subspace, O4A averages the clients' update projectors,

$$\mathrm{Proj}_L = \tfrac{1}{K} \sum_{i=1}^{K} U_i^{(r)} \big(U_i^{(r)}\big)^\top, \tag{4}$$

formed from each client's SVD $\Delta W_i = U_i \Sigma_i V_i^\top$, where $U_i^{(r)}$ denotes the top-r columns of U_i and r is the LoRA rank. The top-k eigenvectors of Proj_L are the shared left basis $U_c \in \mathbb{R}^{d \times k}$. Each client then contributes weight at most $1/K$, so no large-norm client dominates the shared directions. The spectrum of Proj_L measures how much the clients can share (Appendix A.3). The matching right basis $V_c \in \mathbb{R}^{k \times l}$ is coupled to U_c, with the construction in Appendix A.2. A concurrent method also builds a basis-invariant consensus subspace (Zheng et al., 2026), but returns a single global subspace under a differential-privacy objective and keeps no per-client residual.

The consensus rank. The consensus rank k is the method's single hyperparameter, setting how many shared directions the aggregate keeps and thus how much O4A personalizes. It is bounded and discrete. Since $k \leq r$ and the per-layer clamp (Section 4.3) further caps it at $\min_i r_i$, only a few values are admissible for the rank we use (r=8). Setting k too high forces private, conflicting directions into the shared component, the over-sharing that makes aggregation hurt under strong label skew (Section 3.2). Setting it too low discards structure the clients could share. Because the admissible values are so few, we select k by a cheap ablation (Appendix A.4) rather than a closed-form rule, and the best k tracks how much clients actually share.

4.2 SHARED COMPONENT AND PERSONALIZED RESIDUAL

With the subspace fixed, O4A separates what each client shares from what it keeps. The shared component G_i is the update's projection onto (U_c, V_c), and the personal residual P_i is the orthogonal remainder,

$$G_i = U_c U_c^\top \Delta W_i V_c^\top V_c, \qquad P_i = \Delta W_i - G_i. \tag{5}$$

O4A pools the shared components of equation 5 across clients,

$$G = \sum_{i=1}^{K} p_i \, G_i, \tag{6}$$

and gives each client this consensus together with its own residual, the return $\widetilde{\Delta W}_i$ of equation 1. Keeping P_i out of the pool makes O4A an L3 rule (Section 2). Splitting an update into orthogonal shared and personal principal subspaces has precedent outside federated LoRA (Meng et al., 2024; Shi & Al Kontar, 2024). There the split applies to a single model's weights, not a cross-client consensus.

Why sharing the consensus helps. The shared component G is a K-way average of the clients' projections onto the common subspace. Each G_i estimates the same shared directions but is corrupted by that client's own noise, and a small or highly skewed local dataset gives a noisier estimate. Averaging the K estimates keeps the aligned signal while the independent noise falls by roughly

$\sqrt{K}$. We measure this scaling directly (the cross-client mean of the residual directions has about $1/\sqrt{K}$ of a single client's norm, as expected for independent components, Appendix A.5). Returning G in place of a client's own G_i thus replaces a noisy estimate of the shared part with a cleaner one. The gain is largest for the clients whose estimate is noisiest, namely those with little data or a small rank. This is the mechanism behind the data-amount and feature-shift behavior in Section 3.2. The consensus G denoises the under-resourced clients, while the residual P_i preserves what is genuinely theirs. The effect is variance reduction, not a transfer of unique knowledge across clients. Appendix A.6 examines which clients benefit on real data and what makes a domain hard.

4.3 Returning to each client's rank

Finally, O4A refactors the returned update $\widetilde{\Delta W}_i$ to client i's rank by its best rank-r_i approximation,

$$(B_i', A_i') = \arg \min_{\mathrm{rank}(BA) \le r_i} \left\| \widetilde{\Delta W}_i - BA \right\|_F. \tag{7}$$

By the Eckart–Young theorem, this approximation is the truncated SVD of $\widetilde{\Delta W}_i$ kept to r_i components (Eckart & Young, 1936). Client i then fine-tunes from (B_i', A_i'). This is the only step that depends on r_i, and it runs per client. A client with $r_i = 2$ receives a rank-2 adapter and one with $r_i = 8$ a rank-8 adapter, while all server-side computation stays in the shared $d \times l$ space.

Guarding the residual under low rank. If the consensus rank k exceeds a client's true rank r_i, that client's update can be fully absorbed by the k-dimensional shared component, leaving an empty residual P_i and no personalization. We therefore clamp the consensus rank per layer, $k \leftarrow \min\big(k,\ \min_i r_i\big)$. The final consensus rank is the smaller of the shared dimension we select and this per-layer bound. Every client then keeps a non-empty residual, regardless of its rank.

4.4 Why O4A addresses both problems

The two problems set out in Section 2 both come from handling the four axes unevenly, and O4A answers them with the same rule. Working on the reconstructed ΔW lets clients of any adapter rank enter one aggregation, and the per-client refactor returns each client an adapter at its own rank, handling the rank axis. The shared consensus denoises clients with little data or on hard domains, addressing the data-amount and feature-shift axes. The consensus rank k holds back the over-sharing that strong label skew punishes, addressing the label-skew axis. Section 5 verifies O4A along each axis in turn.

5 Experiments

Sections 3 and 4 reduce the four axes of heterogeneity to two problems and give one rule, O4A, that meets all four. Here we verify O4A along the axes, grouped by the two problems: the adapter-rank axis (Problem 1, Section 5.1), the data amount, label skew, and feature shift axes (Problem 2, Section 5.2), and all conditions at once (Section 5.3).

Setup. We fine-tune a frozen ViT-B/16 backbone (Dosovitskiy et al., 2021) with LoRA adapters (Hu et al., 2022) of rank $r = 8$, except where a rank-heterogeneous setting is stated. Two vision benchmarks isolate different axes. **DomainNet** (Peng et al., 2019) maps six domains to six clients as a fixed feature shift over the first 100 categories, following LoRA-FAIR (Bian et al., 2025). **CIFAR-100** (Krizhevsky, 2009) splits ten clients under a Dirichlet(α) label skew (Hsu et al., 2019), with a smaller α giving a stronger skew. In one pipeline we run methods from every level described in Section 2 on both benchmarks: FedIT (L1) (Zhang et al., 2024a; McMahan et al., 2017), the rank-aware FlexLoRA (L2) (Bai et al., 2024) and HetLoRA (L2) (Cho et al., 2024), and the personalized FedPissa (L3) (He et al., 2026), with our rule and a `local-only` floor. Each benchmark adds one further baseline suited to its heterogeneity: FedSA (L3, share A) (Guo et al., 2025) on CIFAR-100, the stronger factor choice under label skew, and LoRA-FAIR (L1) (Bian et al., 2025) on DomainNet, whose feature-shift protocol we follow. Both L3 baselines are factor-sharing methods, the dominant personalized route in recent work (He et al., 2026).

Table 1: CIFAR-100 (α=0.1) top-1 accuracy (%) at full ($\sum_i r_i$=80, homogeneous rank 8) and reduced ($\sum_i r_i$=50, rank-heterogeneous $8/2$) budget. "−": the method stops working under heterogeneous rank. Bold marks the best at reduced budget.

Budget	Method	Level	Accuracy
$\sum_i r_i$=80	local-only	–	95.82 ± 0.23
	FedIT	L1	91.76 ± 0.32
	FlexLoRA	L2	91.69 ± 0.34
	HetLoRA	L2	91.76 ± 0.29
	FedPissa (share B)	L3	95.66 ± 0.06
	FedSA (share A)	L3	96.41 ± 0.07
	Ours	L3	96.45 ± 0.11
$\sum_i r_i$=50	FedIT	L1	−
	FlexLoRA	L2	92.71 ± 0.25
	HetLoRA	L2	92.60 ± 0.20
	FedPissa (share B)	L3	−
	FedSA (share A)	L3	−
	Ours	L3	$\mathbf{96.36 \pm 0.19}$

The consensus rank defaults to $k = \lfloor r/2 \rfloor = 4$ and drops to $k = 2$ under strong skew ($\alpha = 0.1$; Section 5.2). Following pFL practice (Arivazhagan et al., 2019; Li et al., 2021a; Peng et al., 2026), we evaluate each client on its own distribution and report the mean per-client top-1 accuracy. Except where marked †, each reported number is a three-seed mean with its standard deviation across seeds. The full training configuration, per-client rank assignments, and representative per-seed values are in Appendix A.7 and A.8.

5.1 Answering Problem 1: personalization with rank compatibility

Problem 1 (Section 3.1) asks a method to personalize and to serve clients at different ranks at once. Existing methods give one or the other. Our rule aggregates the reconstructed product ΔW and does both. Aggregating ΔW personalizes as well as the best factor-sharing method when ranks match, and keeps working, at lower cost, when they do not.

With homogeneous rank, aggregating ΔW matches the best factor-sharing method. We fix the strong-heterogeneity operating point ($\alpha = 0.1$, homogeneous rank, full data), where every method has a defined aggregation step, and report the full comparison in Table 1 (upper block). The methods that specialize to each client (`local-only`, our rule) and the factor-sharing methods (FedPissa, FedSA; L3) lead. The single-global methods (L1/L2) trail by a clear margin. Our rule is on par with the best factor-sharing method (FedSA). This match shows that aggregating ΔW buys rank compatibility without giving up accuracy. The same rule then serves the heterogeneous-rank regime (Table 1, lower block), where factor-sharing methods produce no model.

Under heterogeneous rank, only ΔW aggregation remains defined. Here, half the clients use rank 8 and half rank 2, keeping $\alpha = 0.1$. The methods that aggregate raw factors (FedIT, FedSA, FedPissa) have no defined aggregation step here and produce no model (Problem 1). The strongest factor-sharing method tied our rule at full budget, so the empty cells are not omissions. They mark where those baselines stop working, the rank-heterogeneous regime our rule is built for. Only the rank-aware baselines and our rule run (Table 1, lower block). Our rule holds its full-budget accuracy while cutting the rank budget by 37.5%, and upload by the same fraction, and leads the surviving rank-aware baselines by a wide margin. Aggregating ΔW thus lets a client cut cost by lowering its rank. A rank-2 client, for example, receives its own rank-2 adapter where factor-sharing methods produce none, and the same holds when the rank varies across rounds (Appendix A.9).

5.2 Answering Problem 2: when and how much to personalize

Whether a per-client model beats a shared one depends on how clients differ (Problem 2, Section 3.2). We take the three conditions in turn, then make the "how much to share" knob explicit.

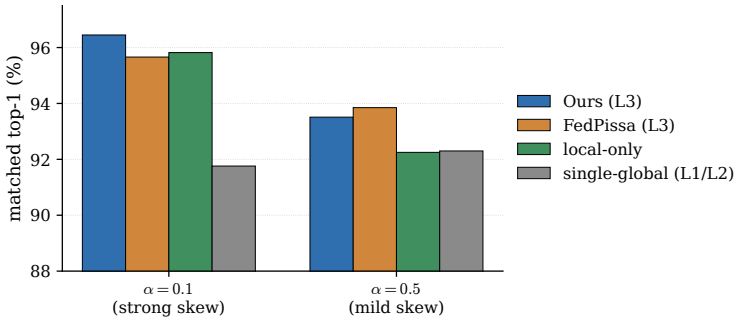

Figure 3: CIFAR-100 label skew at strong (α=0.1) and mild (α=0.5) skew. The near-IID boundary, where the order reverses, is in Appendix A.12.

Data amount. A client with plenty of data can already do well alone, so the benefit of borrowing from other clients should grow as data shrinks. Keeping ranks homogeneous, we reduce every client's training data to a fraction of its original size on CIFAR-100 ($\alpha = 0.1$). Against `local-only` (training alone), our lead is small at full data and widens as data shrinks, largest for the data-poorest clients (Appendix A.10). The same mechanism appears as targeted rescue on DomainNet. When a single domain is reduced to a small fraction of its data, the shared consensus raises its accuracy substantially, most when the domain is deeply reduced but still has signal to recover (Appendix A.11).

Label skew. Sweeping α (full data, homogeneous rank) gives a clean crossover across the full range (Appendix A.12). At strong skew (α=0.1), the personalized methods (L3) and `local-only` lead. At the near-homogeneous end, the single-global methods (L1/L2) are best, and our rule reduces toward one of them (a larger consensus rank at milder skew), staying slightly below the best model. The near-homogeneous end is the boundary of the map, not a counterexample. Figure 3 shows the strong- and mild-skew regime.

Feature shift. On DomainNet, the heterogeneity is a feature shift. At the *full* rank budget (Table 2), our rule and FedPissa are on par. Against `local-only`, our rule leads clearly on the data-scarce, harder domains such as infograph and quickdraw, while on the easier domains it runs about level, since a single client already has enough data there. At full budget with homogeneous rank, only the feature-shift axis is stressed, so both L3 methods (FedPissa and ours) already capture the client-specific signal and little separates them. Our rule separates from FedPissa once the other axes are stressed. At *half* the budget under rank heterogeneity, our rule holds its full-budget accuracy, while the rank-aware L2 methods drop sharply and the factor-sharing methods stop working. A clean consensus (Section 4.2) lets a low-rank client behave like a higher-rank one, so the same accuracy is reached at half the cost. When data are also made scarce, our rule and FedPissa are on par (Appendix A.11).

How much to share: the consensus rank. The ablation in Appendix A.4 (Table 4) makes the dependence concrete. At $\alpha = 0.1$ (strong skew), $k = 2$ is best, while a larger $k = 4$ over-shares and degrades over rounds to below `local-only`. At milder skew ($\alpha = 0.5$), the order reverses and $k = 4$ is best. The best k therefore depends on how much clients share. We choose k per setting (Section 5).

5.3 The combined setting and a regime map

In the combined setting, all three conditions hold simultaneously on CIFAR-100: strong skew ($\alpha = 0.1$), rank heterogeneity ($8/2$), and reduced data. The methods that aggregate raw factors stop working here, so only the rank-aware L2 methods and our rule survive. Our rule leads them by a clear margin (Table 3). Compounding the conditions does not dilute the benefit of personalization.

A regime map. Across the four axes, personalization helps and our rule leads exactly where clients differ: label skew (Figure 3), scarce data (Appendix A.10), rank heterogeneity (Tables 1 and 2),

Table 2: DomainNet per-domain top-1 accuracy (%) at full ($\sum_i r_i$=48, homogeneous rank 8) and half ($\sum_i r_i$=24, rank-heterogeneous; per-client ranks in Appendix A.7) budget. "–": the method stops working under heterogeneous rank. †: from Bian et al. (2025); bold: best per column.

Budget	Method	Lvl	Avg	clip	info	paint	quick	real	sketch
$\sum_i r_i$=48	local-only	–	79.14±0.12	85.62	56.42	**83.62**	74.75	**91.94**	**82.46**
	FedIT	L1	75.75†	84.37	54.17	79.67	69.00	89.20	78.08
	LoRA-FAIR	L1	77.07†	86.25	56.26	80.09	71.25	89.52	79.06
	FlexLoRA	L2	75.87±0.03	83.74	54.45	79.18	71.24	89.69	76.92
	HetLoRA	L2	76.06±0.06	83.73	54.24	79.51	71.68	89.75	77.46
	FedPissa	L3	79.27±0.10	86.05	57.43	82.89	75.78	91.64	81.82
	Ours	L3	79.64±0.10	**86.36**	**58.03**	83.27	**76.20**	91.69	82.28
$\sum_i r_i$=24	FedIT	L1	–	–	–	–	–	–	–
	LoRA-FAIR	L1	–	–	–	–	–	–	–
	FlexLoRA	L2	73.87±0.24	82.25	54.15	78.96	62.48	88.87	76.50
	HetLoRA	L2	73.86±0.18	82.46	54.21	78.43	62.94	88.42	76.69
	FedPissa	L3	–	–	–	–	–	–	–
	Ours	L3	**79.75±0.14**	**86.18**	**58.73**	**83.32**	**76.12**	**91.59**	**82.57**

Table 3: Combined setting on CIFAR-100 (α=0.1, ranks $8/2$, reduced data), top-1 accuracy (%). "frac" is the fraction of each client's training data retained.

Method	frac 0.1	frac 0.2
Ours	$\mathbf{92.02 \pm 0.48}$	$\mathbf{94.36 \pm 0.18}$
FlexLoRA (L2)	87.81 ± 0.54	90.03 ± 0.32
HetLoRA (L2)	87.42 ± 0.48	89.87 ± 0.15
FedPissa / FedSA / FedIT	n/a (shape mismatch)	

and their combination (Table 3). The boundary is equally informative. When clients are near-homogeneous, a single shared model is the right tool. Our rule always keeps a per-client residual, so it sits a little below the shared model there, where the residual is mostly noise (Appendix A.12). The benefit of the residual therefore scales with heterogeneity, helping where clients differ and costing little when they do not.

6 Discussion and Limitations

We approached personalized federated LoRA through four axes of heterogeneity: data amount, label skew, feature shift, and adapter rank. Existing methods (L1–L3, Section 2) each leave open two problems: personalization versus rank compatibility, and how much to personalize. O4A answers both with one rule. Aggregating each client's reconstructed update in the ambient $d \times l$ space removes the factor choice and the rank dependence, and the shared-plus-residual split controls how much each client personalizes. O4A combines two established ingredients, ambient reconstruction (Bai et al., 2024) and a shared-plus-personal split (He et al., 2026; Shen et al., 2026; Peng et al., 2026), into one rule that spans all four axes, and maps when personalization helps. The benefit is regime-dependent. O4A leads under scarce data, label skew, and feature shift. It stays defined under the heterogeneous rank that federated LoRA cannot avoid, where factor-sharing methods break.

Limitations. Our study is on vision classification with a frozen ViT-B/16 and a modest number of clients (cross-silo). We do not evaluate language tasks, cross-device scale, or varying client participation. The server-side SVD is dense in our implementation ($O(d^2 \min(d, l))$ per layer), and a factored variant that exploits the aggregate's low-rank structure would reduce this cost. Communication is rank-r per client, as in other LoRA methods. Finally, our rule always keeps a per-client residual, so near homogeneity it sits slightly below the single-global methods, a small, bounded cost (Appendix A.13) where personalization is not needed.

REPRODUCIBILITY STATEMENT

All methods are re-implemented in a single pipeline and run under matched conditions. Section 5 states the backbone, adapter ranks, datasets, non-IID splits, evaluation protocol, and number of seeds. The exact per-client rank assignments, Dirichlet parameters, data fractions, and per-seed numbers are listed in Appendix A. We will release the code and configuration files needed to reproduce every table.

AI USE STATEMENT

We used a generative AI assistant for two things: refactoring the experiment code, and editing the text of this paper. The authors reviewed all AI-assisted output and take responsibility for the final content of this paper.

REFERENCES

Manoj Ghuhan Arivazhagan, Vinay Aggarwal, Aaditya Kumar Singh, and Sunav Choudhary. Federated learning with personalization layers. *arXiv preprint arXiv:1912.00818*, 2019.

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 (NeurIPS)*, 2024. method named FlexLoRA.

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 (ICCV)*, 2025.

Jieming Bian, Lei Wang, Letian Zhang, and Jie Xu. FedALT: Federated fine-tuning through adaptive local training with rest-of-world lora. In *Proceedings of the AAAI Conference on Artificial Intelligence (AAAI)*, 2026.

Yae Jee Cho, Luyang Liu, Zheng Xu, Aldi Fahrezi, and Gauri Joshi. Heterogeneous LoRA for federated fine-tuning of on-device foundation models. In *Conference on Empirical Methods in Natural Language Processing (EMNLP)*, 2024.

Liam Collins, Hamed Hassani, Aryan Mokhtari, and Sanjay Shakkottai. Exploiting shared representations for personalized federated learning. In *International Conference on Machine Learning (ICML)*, 2021.

Enmao Diao, Jie Ding, and Vahid Tarokh. HeteroFL: Computation and communication efficient federated learning for heterogeneous clients. In *International Conference on Learning Representations (ICLR)*, 2021.

Alexey Dosovitskiy, Lucas Beyer, Alexander Kolesnikov, Dirk Weissenborn, Xiaohua Zhai, Thomas Unterthiner, Mostafa Dehghani, Matthias Minderer, Georg Heigold, Sylvain Gelly, Jakob Uszkoreit, and Neil Houlsby. An image is worth 16x16 words: Transformers for image recognition at scale. In *International Conference on Learning Representations (ICLR)*, 2021.

Carl Eckart and Gale Young. The approximation of one matrix by another of lower rank. *Psychometrika*, 1(3):211–218, 1936.

Wenzhi Fang, Dong-Jun Han, Liangqi Yuan, Seyyedali Hosseinalipour, and Christopher G. Brinton. Federated sketching LoRA: A flexible framework for heterogeneous collaborative fine-tuning of LLMs. In *International Conference on Machine Learning (ICML)*, 2026.

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

Sunny Gupta, Shambhavi Shanker, and Amit Sethi. Amortizing federated adaptation: Hypernetwork driven lora for personalized foundation models. *arXiv preprint arXiv:2606.06154*, 2026.

Wenwen He, Wenke Huang, Yi Liu, Jian Liang, Xirui Li, Guansong Pang, and Mang Ye. FedPissa: Towards federated personalized adaptation of foundation models via lora subspace mapping. In *International Conference on Machine Learning (ICML)*, 2026.

Samuel Horváth, Stefanos Laskaridis, Mario Almeida, Ilias Leontiadis, Stylianos Venieris, and Nicholas D. Lane. FjORD: Fair and accurate federated learning under heterogeneous targets with ordered dropout. In *Advances in Neural Information Processing Systems (NeurIPS)*, 2021.

Tzu-Ming Harry Hsu, Hang Qi, and Matthew Brown. Measuring the effects of non-identical data distribution for federated visual classification. *arXiv preprint arXiv:1909.06335*, 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 (ICLR)*, 2022.

Peter Kairouz, H. Brendan McMahan, et al. Advances and open problems in federated learning. *Foundations and Trends in Machine Learning*, 2021.

Alex Krizhevsky. Learning multiple layers of features from tiny images. Technical report, University of Toronto, 2009.

Seanie Lee, Sangwoo Park, Dong Bok Lee, Dominik Wagner, Haebin Seong, Tobias Bocklet, Juho Lee, and Sung Ju Hwang. FedSVD: Adaptive orthogonalization for private federated learning with lora. In *Advances in Neural Information Processing Systems (NeurIPS)*, 2025.

Qinbin Li, Yiqun Diao, Quan Chen, and Bingsheng He. Federated learning on non-IID data silos: An experimental study. In *IEEE International Conference on Data Engineering (ICDE)*, 2022.

Tian Li, Anit Kumar Sahu, Manzil Zaheer, Maziar Sanjabi, Ameet Talwalkar, and Virginia Smith. Federated optimization in heterogeneous networks. In *Proceedings of Machine Learning and Systems (MLSys)*, 2020.

Tian Li, Shengyuan Hu, Ahmad Beirami, and Virginia Smith. Ditto: Fair and robust federated learning through personalization. In *International Conference on Machine Learning (ICML)*, 2021a.

Xiaoxiao Li, Meirui Jiang, Xiaofei Zhang, Michael Kamp, and Qi Dou. FedBN: Federated learning on non-IID features via local batch normalization. In *International Conference on Learning Representations (ICLR)*, 2021b.

Brendan McMahan, Eider Moore, Daniel Ramage, Seth Hampson, and Blaise Agüera y Arcas. Communication-efficient learning of deep networks from decentralized data. In *Artificial Intelligence and Statistics (AISTATS)*, 2017.

Fanxu Meng, Zhaohui Wang, and Muhan Zhang. PiSSA: Principal singular values and singular vectors adaptation of large language models. In *Advances in Neural Information Processing Systems (NeurIPS)*, 2024.

Xingchao Peng, Qinxun Bai, Xide Xia, Zijun Huang, Kate Saenko, and Bo Wang. Moment matching for multi-source domain adaptation. In *Proceedings of the IEEE/CVF International Conference on Computer Vision (ICCV)*, 2019.

Zihao Peng, Jiandian Zeng, Boyuan Li, Guo Li, Shengbo Chen, and Tian Wang. FedHL: Federated learning for heterogeneous low-rank adaptation via unbiased aggregation. *arXiv preprint arXiv:2505.18494*, 2025.

Zihao Peng, Nan Zou, Jiandian Zeng, Guo Li, Ke Chen, Boyuan Li, and Tian Wang. HiLoRA: Hierarchical low-rank adaptation for personalized federated learning. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*, 2026.

Hariharan Ramesh and Jyotikrishna Dass. Spectral transformation for layer-wise global rank discovery in federated lora for vision transformers. In *European Conference on Machine Learning and Principles and Practice of Knowledge Discovery in Databases (ECML-PKDD)*, 2026.

Zhikang Shen, Jianrong Lu, Haiyuan Wan, and Jianhai Chen. SDFLoRA: Selective decoupled federated lora for privacy-preserving fine-tuning with heterogeneous clients. *arXiv preprint arXiv:2601.11219*, 2026.

Naichen Shi and Raed Al Kontar. Personalized PCA: Decoupling shared and unique features. *Journal of Machine Learning Research*, 25(41):1–82, 2024.

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

Chunlin Tian, Zhan Shi, Zhijiang Guo, Li Li, and Chengzhong Xu. HydraLoRA: An asymmetric LoRA architecture for efficient fine-tuning. In *Advances in Neural Information Processing Systems (NeurIPS)*, 2024.

Lei Wang, Jieming Bian, Letian Zhang, and Jie Xu. Breaking the structural identity: Personalized federated lora fine-tuning under rank heterogeneity. In *Conference on Empirical Methods in Natural Language Processing (EMNLP)*, 2026.

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 (NeurIPS)*, 2024.

Dezhong Yao, Wanning Pan, Yuexin Shi, Michael J. O'Neill, Yutong Dai, Yao Wan, Peilin Zhao, Hai Jin, and Lichao Sun. FedHM: Efficient federated learning for heterogeneous models via low-rank factorization. *Artificial Intelligence*, 344:104333, 2025.

Haoran Zhang, Dongjun Kim, Seohyeon Cha, and Haris Vikalo. FedRot-LoRA: Mitigating rotational misalignment in federated lora. In *International Conference on Machine Learning (ICML)*, 2026.

Jianyi Zhang, Saeed Vahidian, Martin Kuo, Chunyuan Li, Ruiyi Zhang, Guoyin Wang, and Yiran Chen. Towards building the federated GPT: Federated instruction tuning. In *IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP)*, 2024a.

Zikai Zhang, Rui Hu, Ping Liu, and Jiahao Xu. Fed-pilot: Optimizing lora allocation for efficient federated fine-tuning with heterogeneous clients. *arXiv preprint arXiv:2410.10200*, 2024b.

Zikai Zhang, Ping Liu, Jiahao Xu, and Rui Hu. Fed-HeLLo: Efficient federated foundation model fine-tuning with heterogeneous lora allocation. *IEEE Transactions on Neural Networks and Learning Systems (TNNLS)*, 2025.

Lele Zheng, Ruijie Hu, Tao Zhang, Ke Cheng, and Yulong Shen. FedGSA: Geometry-consistent subspace aggregation for differentially private federated lora. *arXiv preprint arXiv:2608.03267*, 2026.

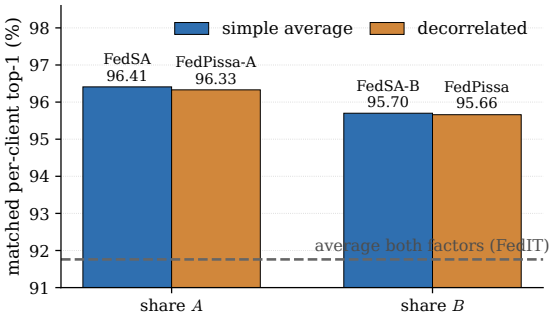

Figure 4: Which factor and which rule (CIFAR-100, α=0.1; setup and metric in Section 5). We compare sharing A (FedSA, FedPissa-A) against sharing B (FedSA-B, FedPissa), each under a simple average or a decorrelated rule. The two rules differ by less than 0.1 percentage point, and sharing A leads sharing B by about 0.7 percentage points. All four sit above averaging both factors (FedIT, dashed). In He et al. (2026)'s own setup the A-versus-B order reverses.

A Additional Results and Details

A.1 The A-versus-B study (Problem 1)

Problem 1 asks which factor to share, and Section 3.1 reads the answer off a 2×2 we lay out in full here. The point is to separate two questions that are usually entangled: whether the chosen *factor* (A or B) or the *aggregation rule* (a simple average or a decorrelated merge) drives the result. Crossing them gives four methods: FedSA and FedSA-B (simple average, sharing A and B respectively) and FedPissa-A and FedPissa (the decorrelated projection-merge of He et al. (2026), sharing A and B). We build the swapped variants FedSA-B and FedPissa-A ourselves so that the factor and the rule move independently. Figure 4 reports all four.

A.2 Full construction of the aggregation rule

Section 4 states the rule in outline. We give the full construction here so it can be reproduced exactly, and collect the whole per-round procedure as pseudocode in Algorithm 1.

Consensus subspace. From the mean left projector Proj_L (equation 4), the shared left basis U_c is its top-k eigenvectors, $U_c \in \mathbb{R}^{d\times k}$. We couple the right basis to U_c by projecting the mean update onto it,

$$\overline{\Delta W} = \tfrac{1}{K}\sum_{i=1}^{K} \Delta W_i, \qquad M = U_c^\top \overline{\Delta W}, \qquad V_c = \text{top-}k \text{ right singular vectors of } M \in \mathbb{R}^{k\times l}. \tag{8}$$

Coupling V_c to U_c, instead of estimating the two sides independently, keeps the same singular components on both sides, so the two-sided projection captures real shared energy.

Split, aggregation, and return. The split and aggregation are equation 5 and equation 6 in the main text, and the return adds each client's residual back, $\widetilde{\Delta W}_i = G + P_i$. The two parts are orthogonal, $G_i \perp P_i$: G_i lies in the shared subspace spanned by (U_c, V_c) and P_i is its complement, so $\langle G_i, P_i\rangle_F = 0$.

Refactoring to each client's rank. The rank-r_i refactor is equation 7 in the main text, obtained from a truncated SVD of $\widetilde{\Delta W}_i$. Client i then uses (A'_i, B'_i). This is the only step that depends on r_i.

A.3 Consensus subspace: intersection vs. averaging

Section 4.1 estimates the consensus subspace by an *intersection* estimate: the top eigenvectors of the mean of the clients' projectors, Proj_L, rather than the main directions of the averaged update.

Algorithm 1 O4A server aggregation (one communication round, per layer)

Require: client factors $\{(A_i, B_i)\}_{i=1}^{K}$, consensus rank k, weights $p_i = 1/K$

$\Delta W_i \leftarrow B_i A_i$ for all i ▷ reconstruct in ambient $d \times l$ space, Eq. 2

$k \leftarrow \min\big(k,\ \min_i r_i\big)$ ▷ per-layer clamp; keeps every residual non-empty

compute the SVD $\Delta W_i = U_i \Sigma_i V_i^\top$ for all i

$\mathrm{Proj}_L \leftarrow \frac{1}{K} \sum_i U_i^{(r)} \big(U_i^{(r)}\big)^\top$ ▷ Eq. 4

$U_c \leftarrow$ top-k eigenvectors of Proj_L; $V_c \leftarrow$ coupled right basis ▷ Eq. 8

for $i = 1, \ldots, K$ **do**

  $G_i \leftarrow$ projection of ΔW_i onto (U_c, V_c), $P_i \leftarrow \Delta W_i - G_i$ ▷ Eq. 5

end for

$G \leftarrow \sum_i p_i G_i$ ▷ aggregate the shared component only, Eq. 6

for $i = 1, \ldots, K$ **do**

  $\widetilde{\Delta W}_i \leftarrow G + P_i$ ▷ consensus plus each client's residual

  $(B_i', A_i') \leftarrow$ rank-r_i truncated SVD of $\widetilde{\Delta W}_i$ ▷ Eckart–Young, Eq. 7

end for

return personalized adapters $\{(A_i', B_i')\}_{i=1}^{K}$

On a synthetic model where the ground-truth shared subspace is known (K clients, mean over 10 seeds), the intersection estimate reconstructs the shared component G with about $6\times$ lower error than averaging the main directions (G-error 0.008 vs. 0.046). A few clients with large update norm dominate the average but not the intersection, which weights a direction by how many clients share it. What this establishes is that the *construction* of the consensus subspace matters, not only that some consensus exists to be shared. The advantage is in the quality of the recovered G. In this synthetic setting the final per-client error is similar for the two constructions (0.018), so the choice matters most for the clients that rely on G heavily, those with little data or low rank, exactly the regimes studied in Sections 5.1–5.2.

A.4 Consensus rank: the ablation behind the per-setting choice

We treat the consensus rank k as the method's single hyperparameter and set it per setting. This appendix gives the ablation behind those choices. Recall k controls how much of each update is shared: a k larger than the structure clients actually have in common forces private, conflicting directions into the shared component (over-sharing), while too small a k discards structure that could be shared.

Table 4 varies k against label skew. At strong skew, little is held in common, so a small k is best and a larger k over-shares and degrades. At milder skew the order reverses and a larger k is best. The best k thus tracks how much clients share, an empirical pattern that holds independent of any closed-form rule.

The near-homogeneous end is more subtle than a simple "$k \to r$ reduces to the single-global model" argument would suggest. Raising k there does *not* recover the single-global accuracy. In fact k barely moves it. At near-IID our rule stays at 90.7–90.8 across $k \in \{2, 4\}$ (90.82 and 90.66), about 1.7 percentage points below the single-global methods. The reason is that the clients' *updates* stay divergent even when their *data* is near-IID, so the shared subspace stays small at every k and a per-client residual persists. Near-IID is therefore the boundary of the map. A single shared model is the right tool there, holding out any residual is a small cost, and existing single-global methods are the better choice.

Per-setting values. The main text uses k=2 for CIFAR-100 at α=0.1 (all α=0.1 rows, including the few-shot and combined settings) and k=4 for CIFAR-100 at α=0.5 and for all DomainNet runs. Under rank heterogeneity the per-layer clamp $k \leftarrow \min(k, \min_i r_i)$ of Section 4.3 reduces k to 2 automatically (since $\min_i r_i$=2).

Table 4: Consensus rank k vs. label skew on CIFAR-100 (full data, homogeneous rank 8): accuracy (%). Smaller α is stronger skew.

Heterogeneity	$\boldsymbol{k = 2}$	$\boldsymbol{k = 4}$
$\alpha = 0.1$ (strong)	$\mathbf{96.45 \pm 0.11}$	78.33 ± 1.35 (degrades)
$\alpha = 0.5$ (mild)	93.38 ± 0.18	93.51 ± 0.14
IID ($\alpha = 100$)	90.82 ± 0.10	90.66 ± 0.18

A.5 The $\sqrt{K}$ denoising of the shared component

Section 4.2 argues that the shared component $G = \sum_i p_i G_i$ acts as a variance reducer: the aligned (shared) part of the G_i adds coherently while the client-specific noise, being roughly independent across clients, averages down by $\sqrt{K}$. We check the scaling directly. Measuring the norm of the cross-client mean of the per-client residual directions against a typical single client's, we find a ratio of about 0.41 for $K = 6$ clients, matching $1/\sqrt{K} = 1/\sqrt{6} \approx 0.41$ expected for independent components. This is the sense in which G denoises: returning the K-averaged consensus in place of a client's own noisy estimate helps most exactly the clients whose estimate is noisiest (little data or small rank), which is the mechanism behind the data-amount behavior of Section 5.2. It is variance reduction, not a transfer of unique knowledge between clients. The per-client residual P_i carries what is genuinely client-specific.

A.6 Which clients benefit from the consensus: a discussion

Section 4.2 argues that the shared component G acts as a denoiser: it helps most the clients whose own estimate is noisiest, and does not transfer unique knowledge between clients. Appendices A.5 and A.3 support this on synthetic data. Here we give a complementary, more exploratory reading on real DomainNet: which clients actually benefit, and what makes a domain "hard." The numbers below are three-seed observations meant to explain the pattern, not a formal claim.

Hard domains are the beneficiaries. We measure each domain's gain from our rule over `local-only` (identical except that `local-only` never aggregates), as a paired difference across seeds. These are the per-domain entries of Table 2 read as Ours $-$ `local-only`.

Domain	**Ours $-$ `local-only` (pp)**	**Difficulty (our acc)**
infograph	$+1.60 \pm 0.37$	58.0 (hardest)
quickdraw	$+1.45 \pm 0.22$	76.2
clipart	$+0.74 \pm 0.63$	86.4
sketch	-0.18 ± 0.23	82.3
real	-0.26 ± 0.11	91.7 (easiest)
painting	-0.35 ± 0.25	83.3

A domain that already does well alone gains little from the consensus, and can lose a little. A domain that struggles alone gains the most. This is the signature of denoising rather than knowledge transfer: G replaces a noisy local estimate of the shared directions with a cleaner one, which matters only where the local estimate is poor. The rank correlation between difficulty and gain is positive but, with six domains, not significant, so we read it as directional.

What "hard" is, and is not. One might read "hard" as "having a unique, private update direction" that the consensus cannot capture. On DomainNet the opposite holds. The hardest high-magnitude domain, quickdraw, has the *largest* absolute consensus energy and the most repeatable update across seeds. Its low alignment *score* is an artifact of a large update norm inflating the denominator, not an inability to learn the shared directions. Difficulty also decouples from data amount and from update magnitude. Quickdraw and real hold almost the same amount of data, yet quickdraw's update is about four times larger. A consistent reading is that a hard domain sits *far from the frozen backbone* and so needs a large update. It needs local rank capacity to hold its own residual P_i, together with a denoised shared head (a small consensus rank k suffices). This also explains the single-domain

rescue of Table 8: the rescue is largest for a deeply starved domain with recoverable signal, and smallest for quickdraw, whose difficulty is intrinsic rather than a matter of too little data.

A.7 Experimental configuration

We list every setting needed to reproduce the numbers, anchoring the choices shared across methods to prior work so that what varies between methods is the aggregation rule and not the tuning. All runs fine-tune a frozen ViT-B/16 backbone with LoRA adapters for 50 communication rounds, with 2 local epochs per round, batch size 128, and learning rate 0.01. Each configuration is run over three seeds ($\{0, 1, 42\}$) and we report the mean. The backbone, batch size, learning rate, local-epoch count, and round count follow the federated-LoRA setup of LoRA-FAIR (Bian et al., 2025), whose rank ablation also covers our LoRA rank $r{=}8$. The per-client rank-heterogeneity assignments below are ours.

Methods and levels. The taxonomy described in Section 2 labels each baseline by its personalization level. L1 (single global, factor space) is FedIT (Zhang et al., 2024a). L2 (single global, rank-aware) covers FlexLoRA (Bai et al., 2024) and HetLoRA (Cho et al., 2024), shown as L2a and L2b where both appear in a plot. L3 (personalized) covers the factor-sharing methods FedPissa (He et al., 2026) (shares B) and FedSA (Guo et al., 2025) (shares A), shown as L3a and L3b, alongside our rule. `local-only` trains each client on its own data with no aggregation.

Why `local-only`, not centralized, is our no-sharing reference. We take `local-only` as the zero-communication floor on both benchmarks. Because each client is evaluated on its own distribution (Section 5), the relevant question is whether sharing beats training alone, which `local-only` measures directly. Centralized training (pooling all clients' data) is a natural ceiling only when clients are near-homogeneous; under strong heterogeneity it converges to a compromise across distributions and need not exceed `local-only`. On DomainNet's feature shift, centralized reaches 77.8 (reported by Bian et al. (2025)), below `local-only` (79.1), so we report `local-only` as the reference.

CIFAR-100. Ten clients over 100 classes. Label skew is a Dirichlet(α) partition of the classes with $\alpha \in \{0.1, 0.5, 100\}$ ($\alpha{=}100$ is near-homogeneous). Homogeneous-rank runs use rank $r{=}8$ for every client (total budget $\sum_i r_i = 80$). The rank-heterogeneous runs assign ranks $(8, 8, 8, 8, 8, 2, 2, 2, 2, 2)$: five clients at rank 8 and five at rank 2, $\sum_i r_i = 50$. Data-amount runs keep a fraction $\{0.1, 0.2, 1.0\}$ of each client's training data. The consensus rank is $k{=}2$ for all $\alpha{=}0.1$ rows and $k{=}4$ otherwise (Appendix A.4), with the per-layer clamp $k \leftarrow \min(k, \min_i r_i)$ described in Section 4.

DomainNet. Six domains mapped to six clients over the first 100 DomainNet categories, following LoRA-FAIR (Bian et al., 2025) (a fixed feature shift). Homogeneous-rank runs use $r{=}8$ for every client ($\sum_i r_i = 48$). The half-budget rank-heterogeneous runs assign ranks $(2, 6, 2, 8, 2, 4)$ ($\sum_i r_i = 24$). The consensus rank is $k{=}4$. The feature-shift panel of Figure 2 restricts each domain to 10% of its data to make the feature-shift effect visible.

A.8 Per-seed numbers

Full per-seed accuracy for every main-text table will be released with the code and configuration files. As the key integrity check, the flagship CIFAR-100 result at $\alpha{=}0.1$ (Table 1) is $\{96.57, 96.43, 96.35\}$ across the three seeds (mean 96.45), and the over-sharing $k{=}4$ case (Table 4) is $\{76.89, 79.56, 78.55\}$ (mean 78.33), so both are stable three-seed means rather than single runs.

A.9 Robustness to dynamic rank truncation

The rank-heterogeneity study (Section 5.1) fixes each client's rank for the whole run. A harder version lets the rank vary *over time*: in each round half the clients upload at half rank ($r{=}8 \to 4$) and the other half at full rank, with the roles alternating round to round. This stresses aggregation in the ambient space, since the shared object must absorb a rank that changes every round. On DomainNet our rule is essentially unaffected (Table 5): it reaches 79.59 ± 0.13, within noise of the 79.64 it

attains with no truncation. The rank-aware L2 baselines also run, while the factor-sharing FedPissa has no defined aggregation step once the per-round ranks differ and produces no model, the same failure mode as static rank heterogeneity, now triggered dynamically. Aggregating the reconstructed update handles time-varying rank at no extra cost, because the ambient update is $d \times l$ whatever the round's rank.

Table 5: Dynamic rank truncation on DomainNet (each round, half the clients upload at half rank): accuracy (%). Our rule stays within noise of its no-truncation value (79.64).

Method	**Level**	**Accuracy**
Ours	L3	$\mathbf{79.59 \pm 0.13}$
FlexLoRA	L2	76.40 ± 0.05
HetLoRA	L2	76.28 ± 0.14
FedPissa (share B): n/a (shape mismatch)		

A.10 CIFAR-100 DATA AMOUNT (FULL TABLE)

These are the per-method numbers behind the data-amount curve of Section 5.2 (Figure 5; CIFAR-100, α=0.1). The last column reads off our lead over the best non-personalized baseline at the scarcest setting, the point the data-amount axis is really about, where a client has least of its own data to learn from.

Table 6: Data amount on CIFAR-100 (α=0.1), top-1 accuracy (%). "frac" is the fraction of each client's training data kept. The right column is our lead over the best non-personalized baseline (`local-only` or L2).

Method	**frac 0.1**	**frac 0.2**	**full**	**lead (frac 0.1)**
Ours	$\mathbf{91.59 \pm 0.16}$	$\mathbf{94.16 \pm 0.10}$	$\mathbf{96.45 \pm 0.11}$	—
FedPissa (L3)	87.62 ± 0.40	90.32 ± 0.47	95.66 ± 0.06	+3.97
local-only	86.68 ± 0.49	91.11 ± 0.27	95.82 ± 0.23	+4.91
FlexLoRA (L2)	86.65 ± 0.31	89.16 ± 0.14	91.69 ± 0.34	+4.94
HetLoRA (L2)	86.70 ± 0.11	89.02 ± 0.17	91.76 ± 0.29	+4.89
FedIT (L1)	86.61 ± 0.15	89.09 ± 0.19	91.76 ± 0.32	+4.98

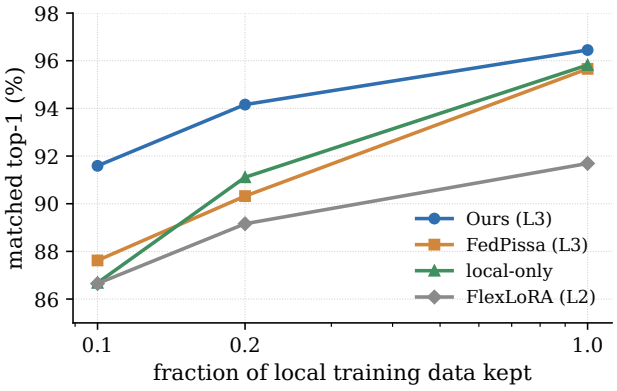

Figure 5: Data amount on CIFAR-100 (α=0.1). All methods are close at full data; as each client keeps less of its own data, our rule stays higher and its lead over training alone (`local-only`) widens. This is the growth of the data-amount axis, read against the "train alone" baseline.

A.11 DOMAINNET

DomainNet has six domains, mapped to six clients, with a fixed feature-shift heterogeneity. Table 7 reports the data-amount axis when every client's data is reduced to a fraction (the counterpart of Table 6 on a natural feature shift). The rank order of the methods matches CIFAR-100: the personalized methods lead the single-global and local-only baselines, and the gap grows as data becomes scarcer. On DomainNet our rule and FedPissa are on par (they differ by less than their standard deviation), while both lead the L2 methods and `local-only`.

Table 7: DomainNet per-domain accuracy (%) with every client's data reduced to fraction (a) 0.1 and (b) 0.2; "Avg" is the six-domain mean. `local-only` collapses on the low-data domains (infograph) while the shared methods hold up. Bold: best per column.

(a) fraction 0.1

Method	**Avg**	clip	info	paint	quick	real	sketch
Ours	**68.94**	**75.20**	45.09	**75.31**	59.33	**88.38**	**70.34**
FedPissa	68.86	74.60	**45.21**	74.87	**60.56**	88.03	69.93
FlexLoRA	67.76	73.93	43.88	74.25	59.28	87.20	67.99
HetLoRA	67.64	74.11	43.75	74.23	59.10	87.20	67.44
local-only	63.04	65.88	31.82	72.47	58.21	88.31	61.56

(b) fraction 0.2

Method	**Avg**	clip	info	paint	quick	real	sketch
Ours	72.77	78.97	48.72	**78.37**	**65.93**	89.55	**75.06**
FedPissa	**72.82**	**79.59**	**48.94**	78.34	65.79	89.53	74.70
FlexLoRA	70.82	77.98	47.32	76.32	64.02	88.01	71.28
HetLoRA	70.82	77.73	47.02	76.35	64.36	88.14	71.34
local-only	69.76	75.27	40.39	77.50	65.33	**89.88**	70.18

Per-seed, heterogeneous rank. For the half-budget DomainNet heterogeneous-rank result (Table 2, $\sum_i r_i$=24), the three-seed runs are: our rule $\{79.84, 79.83, 79.59\}$ (mean 79.75), FlexLoRA $\{73.60, 73.95, 74.05\}$ (mean 73.87), and HetLoRA $\{73.70, 74.06, 73.82\}$ (mean 73.86). The lead is stable across seeds, not an average over a wide spread.

Single-domain rescue. This probe starves one domain to a small fraction of its data while the other five keep full data, then measures how much sharing lifts the starved domain (our rule minus `local-only`, on that domain) and the six-domain average. Table 8 reports all four cells over three seeds. The effect follows the data-amount axis. The rescue is largest where the starved domain still has recoverable signal and is deeply starved (infograph at 5%, real at 2%), and shrinks to a small margin where the domain is barely starved (real at 5%) or is intrinsically hard to learn (quickdraw). The quickdraw cell is the low end of this spectrum, as the map predicts: sharing helps least where a domain has little structure to recover.

Table 8: Single-domain rescue on DomainNet: one domain is starved to the listed fraction while the other five keep full data (three seeds). Δ starved is our rule minus `local-only` on the starved domain; Δ Avg is on the six-domain mean. Values are percentage points.

Starved domain	**Fraction**	**Δ starved (pp)**	**Δ Avg (pp)**
infograph	5%	$+22.32 \pm 1.74$	$+3.86 \pm 0.37$
real	2%	$+7.83 \pm 0.49$	$+1.94 \pm 0.14$
real	5%	$+2.14 \pm 0.23$	$+0.88 \pm 0.03$
quickdraw	5%	$+1.16 \pm 0.76$	$+0.72 \pm 0.10$

A.12 Robustness across heterogeneity strength (the α sweep)

The main text (Figure 3) shows the strong- and mild-skew end. Here we give the full sweep including near-IID (Table 9) and read the boundary, where the order reverses, in detail.

The trend is a clean crossover. At strong skew the personalized and local methods lead. As α grows and clients become alike, the order reverses. Near homogeneity the single-global methods are best, our rule sits just below them, and `local-only` is lowest (Table 9). We read this boundary as expected behavior, not a weakness. When clients are homogeneous there is little client-specific structure to keep, so holding out a per-client residual costs a little and gains little, and the single-global methods are the right tool. This mirrors the combined setting (Section 5.3), where the same rule leads by the largest margin because there *is* the most that is client-specific to keep. Raising the consensus rank does not close the near-homogeneity gap, as Appendix A.4 explains.

Table 9: Full label-skew sweep on CIFAR-100 (full data, homogeneous rank): top-1 accuracy (%) from strong skew (α=0.1) to near-IID. Near IID the single-global methods overtake, the expected behavior where personalization is not needed, not a failure of the rule. Bold: best per column.

Method	$\boldsymbol{\alpha = 0.1}$ (strong)	$\boldsymbol{\alpha = 0.5}$ (mild)	near-IID
Ours	$\mathbf{96.45 \pm 0.11}$	93.51 ± 0.14	90.66 ± 0.18
FedPissa (L3)	95.66 ± 0.06	$\mathbf{93.85 \pm 0.16}$	92.41 ± 0.19
local-only	95.82 ± 0.23	92.25 ± 0.07	88.62 ± 0.34
FlexLoRA (L2)	91.69 ± 0.34	92.27 ± 0.12	92.55 ± 0.22
HetLoRA (L2)	91.76 ± 0.29	92.30 ± 0.20	$\mathbf{92.67 \pm 0.11}$
FedIT (L1)	91.76 ± 0.32	92.28 ± 0.12	92.58 ± 0.11

A.13 Stability of the returned adapters

Our rule returns $G + P_i$ to each client every round. A natural question is whether the per-client residual P_i grows without bound as rounds accumulate. Empirically it does not. On the synthetic model of Appendix A.3 (mean over 20 seeds), the client error converges to a small non-zero floor (0.018 ± 0.003) with no divergence, and the residual norm $\|P_i\|$ stays near 2.1 across rounds. Returning a per-client residual each round therefore does not amplify it. The floor is non-zero because the rank-k consensus does not capture a client's update exactly, the same approximation gap that makes keeping a residual a small, bounded cost near homogeneity (Section 6).