Crowded in B-Space: Calibrating Shared Directions for LoRA Merging

Yixuan Tang
The Hong Kong University of Science and Technology
ytangch@connect.ust.hk

Yi Yang
The Hong Kong University of Science and Technology
imyiyang@ust.hk

Abstract

Merging separately trained LoRA adapters is a practical alternative to joint multi-task training, but it often hurts performance. Existing methods usually treat the LoRA update $\Delta W = BA$ as a single object and do not distinguish the two LoRA matrices. We show that the main source of LoRA merge interference comes from the output-side matrix B. Across tasks, B repeatedly uses a small set of shared directions, while A remains much more task-specific. As a result, the merged adapter overemphasizes these shared directions, and task-specific information is lost. We propose **Pico** (**P**re-merge **i**nterference **c**alibration in **o**utput-space), a data-free method that calibrates B before merge by downscaling over-shared directions and then rescaling the merged update. Pico plugs directly into existing merging methods such as Task Arithmetic, TIES, and TSV-M. Across eight different benchmarks from math, coding, finance, and medical domains, Pico improves average accuracy by 3.4–8.3 points over the corresponding base method and achieves the best overall average performance. Pico also enables merged adapters to outperform the LoRA trained with all task data. These results show that LoRA merging works better when the two LoRA matrices are treated separately.

1 Introduction

Fine-tuning pretrained models for different tasks, domains, or data distributions has become standard practice, producing many specialized checkpoints from the same base model. However, leveraging these diverse checkpoints effectively remains challenging. Model merging offers a promising solution by combining them into a single model without joint retraining [12]. It is now widely used, from composing skills across tasks [12, 25] to averaging training checkpoints for improved robustness [29, 34, 16].

Low-Rank Adaptation (LoRA) [11] parametrizes each task-specific update as a small low-rank adapter. Because adapters are compact and easy to distribute, thousands of them are now publicly available on platforms such as Hugging Face [23, 35]. Merging these adapters is a natural way to combine their capabilities. Yet merging independently trained adapters often degrades their individual performance. To mitigate this issue, existing methods attempt to reduce merge interference by resolving sign conflicts [30], pruning small weights [32, 7], or aligning updates in a shared basis [25]. However, these methods all treat the LoRA update $\Delta W = BA$ as one object. They do not distinguish the two LoRA matrices that produce it, i.e., A and B.

This matters because the two LoRA matrices play different roles. In $\Delta W = BA$, A maps input features into a low-rank space, while B maps that representation back to the output space. Prior

40th Conference on Neural Information Processing Systems (NeurIPS 2026).

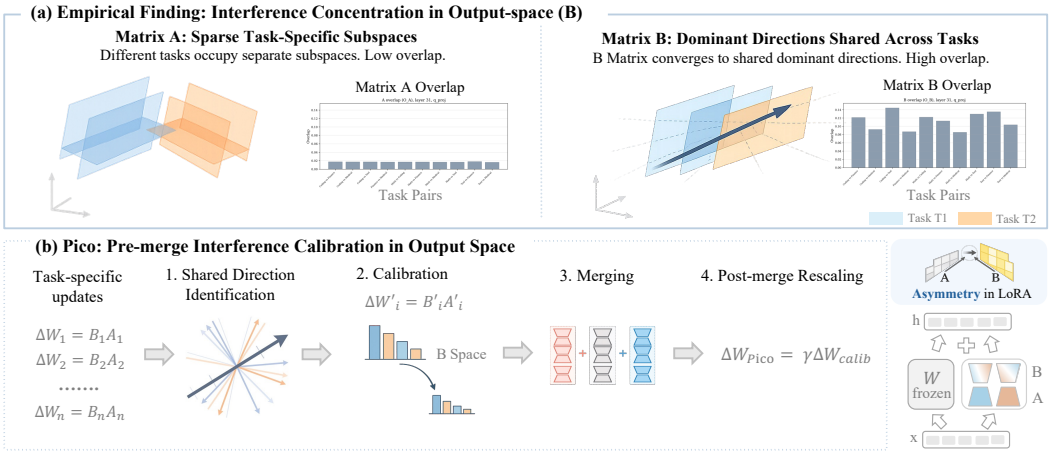

Figure 1: **Overview of Pico.** (a) Merging interference in LoRA is asymmetric: task-specific A matrices remain relatively separated, while B matrices align along shared dominant directions across tasks. (b) Pico identifies and downscales these shared directions in B before merging, then rescales the merged update to preserve its magnitude.

work already suggests that finetuning is asymmetric across the two matrices. Zhu et al. [36] show that, in single-task finetuning, a larger share of the update change is carried by B. Practical methods exploit the same asymmetry. LoRA-FA [33] freezes A after initialization to reduce memory, and HydraLoRA [27] shares a single A across task-specific B heads. These designs treat A as the easier factor to freeze or share, while leaving more task-specific change in B.

This creates a natural question for merging. Existing merge methods still operate on the full update $\Delta W = BA$, even though the two factors do not contribute equally during finetuning. If B carries more of the adapted change, it may also carry more of the merge interference when independently trained adapters are combined. We therefore ask: *does merge interference come from the full update, or mainly from the output-side matrix B?*

Our answer is the output-side matrix B. Across independently trained LoRAs for math, coding, finance, and medical reasoning, cross-task overlap is consistently much larger in B than in A, and this gap grows with rank. At the same time, the effective rank of B stays low even at higher ranks, so different tasks keep reusing a small set of output-space directions. Together, these two patterns point to a simple failure mode: the main problem is not uniform conflict across the full update, but repeated accumulation of a few shared directions in B.

This suggests that the fix should also focus on B. We propose **Pico** (**P**re-merge **i**nterference **c**alibration in **o**utput-space), a data-free calibration step applied to task-specific B matrices before merging. Pico identifies shared directions in B, downscales them according to how strongly they align across tasks, and then rescales the merged update to preserve its overall magnitude. Pico is not a replacement merger. Instead, it plugs directly into methods such as Task Arithmetic [12], TIES [30], and TSV-M [8]. Figure 1 gives an overview of the problem and the method.

Experiments across eight datasets from four domains show that this simple calibration consistently improves downstream merging. Pico improves over the corresponding uncalibrated baselines by 3.4 points for Task Arithmetic, 4.7 points for TIES, and 8.3 points for TSV-M. It also outperforms DARE [32], DELLA [7], KnOTS [25], and Core Space [22] in overall average performance, and enables merged adapters to outperform a LoRA adapter trained jointly on all task data. Overall, this paper makes three contributions. First, it reveals that LoRA merge interference is structurally concentrated in the output-side matrix B rather than uniformly distributed across the full update ΔW. Second, it traces this failure mode to repeated accumulation of a few shared directions in B across tasks. Third, it introduces a simple plug-in calibration step that corrects this over-accumulation and improves a range of existing LoRA merging methods[1].

[1]Github: https://github.com/yixuantt/Pico

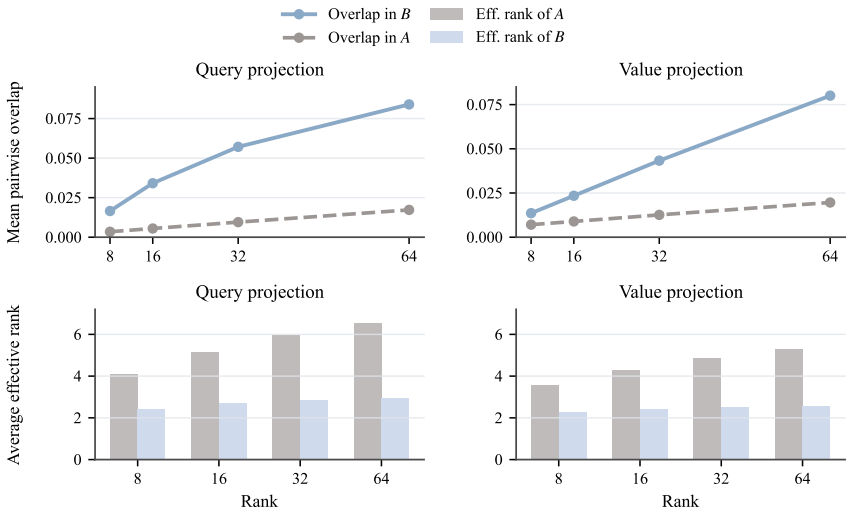

Figure 2: **Quantitative motivation for Pico.** Top row: mean pairwise overlap in A and B across four domain-specific LoRA adapters, shown for the query and value projection matrices. Bottom row: average effective rank of A and B across LoRA ranks. The exact values are reported in Appendix A.

2 Background and Problem Setup

LoRA adapter merging. We use t to index task-specific adapters. For adapter t, the update at a given layer is $\Delta W_t = B_t A_t$, where $A_t \in \mathbb{R}^{r \times d_{\text{in}}}$, $B_t \in \mathbb{R}^{d_{\text{out}} \times r}$, and r is the LoRA rank. For simplicity, we absorb the usual LoRA scaling factor into B_t throughout the paper. Given T independently trained adapters with the same dimension, LoRA merging aims to construct a single merged update

$$\Delta W_{\text{merge}} = \mathcal{M}(\Delta W_1, \ldots, \Delta W_T)$$

that combines their capabilities without retraining on the union of task data.

Merge interference. Under a linear merge, components shared across adapters retain their full magnitude, while task-specific components are diluted. If several adapters contain large aligned directions, those directions can dominate the merged update and make task-specific directions relatively weaker. We call this effect merge interference. Appendix A gives a toy derivation showing that, under simple averaging, the shared-to-specific ratio grows by a factor of T, so task-specific capabilities can be lost even when each source adapter learns them well.

Overlap and concentration. We use two measurements to quantify merge interference. Overlap across a pair of adapters is measured by a normalized subspace overlap score $O_B(i,j), O_A(i,j) \in [0,1]$, where larger values mean that the two adapters share more directions in the output- or input-side matrix, respectively. Let Q_i^B, Q_j^B be orthonormal bases of the column spaces of B_i, B_j and let Q_i^A, Q_j^A be orthonormal bases of the row spaces of A_i, A_j; then

$$O_B(i,j) = \tfrac{1}{r}\left\|(Q_i^B)^\top Q_j^B\right\|_F^2, \qquad O_A(i,j) = \tfrac{1}{r}\left\|(Q_i^A)^\top Q_j^A\right\|_F^2.$$

Spectral concentration is measured by effective rank, where lower values mean that fewer directions contribute substantially. Full definitions (including effective rank and component energy) are given in Appendix A. These measurements do not assume that every shared direction is harmful. Instead, they test whether shared directions accumulate unevenly across A and B.

3 Why Does LoRA Merging Fail?

Setup. We analyze four domain-specific LoRA adapters trained on Llama-3.1-8B [26] for math, coding, finance, and medical reasoning. To keep the comparison controlled, we sample 50k training

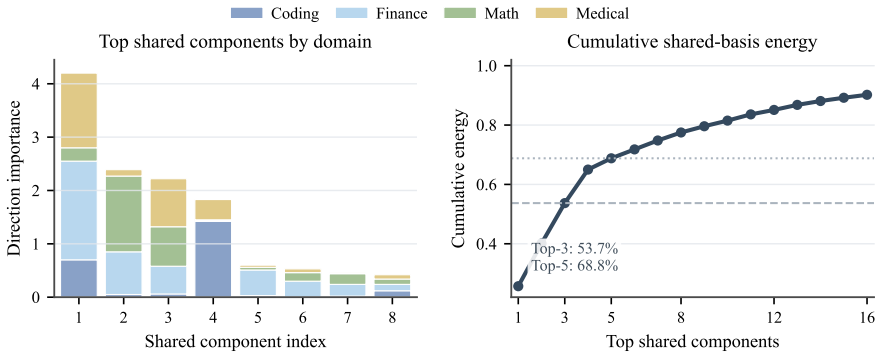

Figure 3: **Representative dominance pattern in the shared B spectrum** (query projection, layer 16, rank 16). Left: contribution of each task to the leading shared components. Right: cumulative energy in the same shared basis. The first five components account for 68.8% of the total energy.

examples per domain and fix LoRA alpha at 16. In our setup, LoRA is applied to the query and value projection matrices in attention, so we focus this analysis on those two target modules. We study LoRA ranks 8, 16, 32, and 64. Appendix A reports the exact datasets and full numeric results.

Overlap is much larger in B. The first observation is that overlap across tasks is consistently much larger in B than in A. At rank 8, $O_B > O_A$ already holds for 99.5% of query-projection layer pairs and 95.3% of value-projection layer pairs. At ranks 16, 32, and 64, it holds for every measured layer pair in both modules. The gap also grows with LoRA rank. For example, in the query projection, the mean overlap in B rises from 0.0166 to 0.0839, while the overlap in A grows only from 0.0035 to 0.0172. The same pattern appears in the value projection, where B rises from 0.0136 to 0.0801 but A increases only from 0.0071 to 0.0196. Although these scores are normalized and averaged, the asymmetry is highly consistent. Appendix A reports the full curves and exact values. Appendix E further provides random-subspace references and an initialization control. Together, these results show that merge interference is concentrated much more strongly in B.

B stays concentrated as rank grows. The second observation is that increasing the LoRA rank does not make B much more diverse. Instead, B keeps using only a small number of directions, while A keeps expanding. At rank 64, the average effective rank in the query projection is 2.94 for B versus 6.55 for A; in the value projection, it is 2.59 for B versus 5.30 for A. Together with the overlap results, this means that independently trained adapters keep writing into the same small set of directions in B, while the corresponding A matrices remain much more spread out. This gives a concrete picture of why naive merge degrades performance. The main problem is not uniform conflict across the full update $\Delta W = BA$. Instead, a few shared directions in B are counted repeatedly during merge and end up dominating the merged adapter. This is exactly the effect that Pico targets in Figure 1(b).

A few shared B components can dominate before merge. Figure 3 adds one more detail to the same diagnosis. For the query projection at layer 16, we stack the four task-specific B matrices, compute a joint SVD, and inspect the leading shared components. In this example, the first three components already account for 53.7% of the shared-basis energy, and the first five account for 68.8%. Finance, math, and medical contribute more strongly to these dominant shared components than coding. This shows that the dominant shared B components are not only concentrated, but do not represent all tasks equally before merging. When merge is dominated by a few shared B components, tasks that contribute less to those components can be harder to preserve.

4 Pico: Pre-merge interference calibration in output-space

Pico addresses the failure mode identified in Section 3. For each target module in each layer, it adds one calibration step before a standard LoRA merge rule: it reduces output-side directions in B that

are reused too heavily across tasks, then merges the calibrated updates and rescales the overall update magnitude. The whole procedure is data-free and plug-in: it uses only the source adapters and can be paired with different downstream merge rules.

4.1 Shared-Direction Calibration

Shared-direction calibration is the key step in Pico. As merge interference is much stronger in the output-side LoRA matrices than in the input-side matrices, Pico calibrates B before applying any downstream merge rule. For one layer, we start from task-specific updates $\{\Delta W_t = B_t A_t\}_{t=1}^{T}$ and produce calibrated updates $\{\widetilde{\Delta W}_t\}_{t=1}^{T}$ that are used in the next merge stage.

Shared Basis. Pico first builds a shared basis for the output space. It stacks the output-side matrices from all tasks and computes a joint singular value decomposition,

$$B_{\text{all}} = [B_1, B_2, \ldots, B_T] = U \Sigma V^\top.$$

The columns of U define shared output-space basis vectors for this layer, and σ_j denotes the j-th singular value in Σ. This gives Pico a common basis for checking how strongly different tasks use each basis component.

Sharing Score. Pico then assigns each basis component a score that measures how strongly it is shared. For the j-th basis component, we define a sharing score and a corresponding scaling coefficient,

$$s_j = \frac{\sigma_j^2}{\sum_k \sigma_k^2}, \qquad \alpha_j = \frac{1}{1 + (T-1)s_j}.$$

Here s_j measures how much of the joint B-space energy lies in component j, so a larger value means it is shared more heavily across tasks. The coefficient α_j controls how much of that component is kept during calibration. Lightly shared components should stay near 1, while a component that dominates the joint spectrum should be reduced toward $1/T$.

Calibration Operator. Pico uses these coefficients to build a layerwise calibration operator. Let $m = \min(d_{\text{out}}, Tr)$, let $\alpha = [\alpha_1, \ldots, \alpha_m]$ collect the scaling coefficients, and let I denote the identity matrix on the output space:

$$S = I + U \,\text{diag}(\alpha - 1)\, U^\top.$$

Pico then applies this operator directly to the output-side LoRA matrix:

$$\widetilde{B}_t = S B_t.$$

The calibrated update is then $\widetilde{\Delta W}_t = \widetilde{B}_t A_t$. In this way, Pico changes only the output-side matrix B_t, where merge interference is strongest, while leaving the more task-specific input-side matrix A_t untouched. Computationally, each layer requires one SVD of the stacked B matrices plus one left multiplication for each task-specific B_t.

4.2 Merge and Rescaling

After calibration, the calibrated updates are merged with a standard LoRA merge rule:

$$\Delta W_{\text{calib}} = \mathcal{M}(\widetilde{\Delta W}_1, \ldots, \widetilde{\Delta W}_T),$$

where $\mathcal{M}$ can be Task Arithmetic, TIES, TSV-M, or another downstream merger.

Magnitude Rescaling. Calibration improves the merged directions but can reduce the Frobenius norm of the merged update, leaving the adapter too weak. Pico therefore rescales the merged update to match the average magnitude of the source adapters:

$$\Delta W_{\text{Pico}} = \gamma \Delta W_{\text{calib}} \quad \text{and} \quad \gamma = \frac{1}{\|\Delta W_{\text{calib}}\|_F} \cdot \frac{1}{T} \sum_{t=1}^{T} \|\Delta W_t\|_F.$$

This scalar rescaling preserves the calibrated direction of the merged update while restoring its overall magnitude.

Table 1: **Main results across four domains.** Each domain column reports the average over its two benchmarks, and the last column reports the average over all eight benchmarks. Bold marks the best overall average in each merger block, and underlining marks the second best. Full benchmark-level results are given in Appendix C.

Merger	Method	Math Avg. ↑	Coding Avg. ↑	Finance Avg. ↑	Medical Avg. ↑	Overall Avg. ↑
Task Arithmetic	No Calibration	0.2666	0.2823	0.5044	0.5839	0.4093
	DARE	0.2757	0.2931	0.5206	0.5851	<u>0.4186</u>
	DELLA	0.2651	0.2662	0.5187	0.5857	0.4089
	KnOTS	0.2727	0.2714	0.5127	0.5784	0.4088
	Core Space	0.2620	0.2773	0.4987	0.5880	0.4065
	Pico	0.3065	0.3372	0.5431	0.5854	**0.4430**
TIES	No Calibration	0.2808	0.2117	0.5271	0.5241	0.3859
	DARE	0.2601	0.1936	0.5130	0.4974	0.3660
	DELLA	0.1928	0.1709	0.4750	0.4025	0.3103
	KnOTS	0.2876	0.3012	0.5277	0.5661	<u>0.4207</u>
	Core Space	0.2413	0.2490	0.4230	0.4936	0.3517
	Pico	0.3061	0.3036	0.5096	0.6116	**0.4328**
TSV-M	No Calibration	0.2636	0.1570	0.4487	0.5197	0.3473
	DARE	0.2926	0.1620	0.4481	0.5859	<u>0.3722</u>
	DELLA	0.2778	0.1598	0.4662	0.5268	0.3577
	KnOTS	0.2666	0.1582	0.4625	0.5218	0.3523
	Core Space	0.2385	0.1415	0.3568	0.4413	0.2945
	Pico	0.2883	0.2942	0.5257	0.6139	**0.4305**

5 Experiment

We evaluate Pico in a controlled four-domain setting. We focus on two outcomes: whether it consistently improves strong LoRA merging baselines, and whether the best merged adapter remains competitive with a jointly trained multi-task LoRA.

5.1 Experimental Setup

Training Data. We train four domain-specific LoRA adapters on Llama-3.1-8B [26], one for each target domain. The training sets are MetaMathQA [31] for math, Magicoder-Evol-Instruct-110K [28] for coding, ODA-Fin-SFT-318k [3] for finance, and Medical-Reasoning-SFT-Trinity-Mini [21] for medical reasoning. To keep the comparison controlled, we sample 50k training examples per domain and keep the LoRA setting fixed across domains. For the main experiments in this section, we use LoRA rank 16. In all experiments, LoRA is applied to the query and value projection matrices in attention.

Benchmarks. We evaluate merged adapters on eight benchmarks. The math domain uses GSM8K [6] and MATH [18], coding uses HumanEval [4] and MBPP [2], finance uses FinanceBench [13] and ConvFinQA [5], and medical uses PubMedQA [15] and MedQA-USMLE. [14] We use the average score across all eight benchmarks as the main summary metric.

Baselines. Our main comparisons use three downstream merge rules: Task Arithmetic [12], TIES [30], and TSV-M [8]. We compare Pico against four strong baselines that reduce merge interference in different ways. DARE [32] is a sparsification-based plug-in that drops and rescales delta parameters before parameter fusion. KnOTS [25] is an alignment-based method that maps LoRA updates into a shared basis before merging. DELLA [7] is a magnitude-based merging technique, and Core Space [22] is a low-rank merging framework that also operates in a shared basis. All methods start from the same four source adapters; only the merge strategy changes. The main experiment reports domain averages. This keeps each table focused on the comparison we care about. Appendix C gives the full benchmark-level results and Appendix B lists the training hyperparameters.

Table 2: **Comparison with single-domain and jointly trained LoRA adapters.** Each domain column reports the average over its two benchmarks. The final row reports the best merged adapter from Table 1. Full benchmark-level values are given in Appendix C.

Adapter	Math Avg. ↑	Coding Avg. ↑	Finance Avg. ↑	Medical Avg. ↑	Overall Avg. ↑
Math LoRA	**0.2830**	0.1090	0.4403	0.3665	0.2997
Coding LoRA	0.2505	**0.3598**	0.4812	0.5516	0.4108
Finance LoRA	0.2477	0.1440	**0.5484**	0.4976	0.3594
Medical LoRA	0.2438	0.3190	0.4848	**0.5950**	0.4106
Joint Multi-Task LoRA	0.2544	0.3151	0.4655	0.4405	0.3688
Pico w/ Task Arithmetic	0.3065	0.3372	0.5431	0.5854	**0.4430**

5.2 Main Results

Main finding. Table 1 gives the main result. Pico improves every downstream merger we test. Relative to the corresponding no-calibration baseline, it raises the average score from 0.4093 to 0.4430 for Task Arithmetic, from 0.3859 to 0.4328 for TIES, and from 0.3473 to 0.4305 for TSV-M. These are absolute gains of 3.4, 4.7, and 8.3 points, respectively. Appendix I further compares Pico with Iso-C, Iso-CTS, and Subspace-Boosted [20, 24] on all eight benchmarks.

Consistency Across Mergers. The gains are not tied to one downstream rule. This matters because Pico is meant to correct an upstream problem in the source adapters rather than replace the merger itself. Task Arithmetic, TIES, and TSV-M make different downstream decisions, but they all start from the same source adapters. Once the over-shared directions in B are calibrated before merge, all three improve.

Where the gains appear. The largest gains appear on top of TIES and TSV-M, where shared components in B interact most directly with the downstream merge logic. In TIES, over-shared components survive magnitude trimming more easily and dominate the elected sign vector; in TSV-M, they concentrate energy in the low-rank singular-vector space that the merger explicitly operates on. Pico calibrates these components before either merger acts, which is why the two are complementary. Task Arithmetic also benefits, but its coordinate-level averaging is less sensitive to spectral concentration, so the gain is smaller. Overall, the same pattern repeats across all three mergers: calibrating B before merge produces a more balanced adapter across domains.

5.3 Comparison with Single-Task and Jointly Trained LoRA

Single-Domain Specialists. Table 2 compares the best merged adapter from Table 1 with four single-domain LoRAs and with a jointly trained multi-task LoRA. The single-domain adapters are highly specialized: each one performs best or near-best on its own source domain, but its average score drops on the other domains. This is the trade-off that makes adapter merging attractive in the first place.

Joint Training vs. Merging. The jointly trained multi-task LoRA is more balanced than the single-domain specialists, but its overall average is still only 0.3688. The best merged adapter, Pico with Task Arithmetic, reaches 0.4430 in Table 2, exceeding joint training by 7.4 points. In this setup, Pico-based merging is therefore not just competitive with joint training; it is stronger on the overall average. In this setup, calibrating the over-shared components in B before merge produces a stronger multi-domain adapter than joint training on all task data.

5.4 Ablations and Analysis

We use ablations and robustness checks to answer two questions. First, is Pico's gain really tied to calibrating the output-side matrix B? Second, does the same advantage remain under other settings beyond the main experiment?

Calibration space matters. Table 3 answers two design questions. First, where should calibration be applied? Calibrating only the output-side matrix B gives the best result, raising the overall average

Table 3: **Calibration-space and restoration ablation on top of Task Arithmetic at rank 16.** Each domain column reports the average over its two benchmarks. Calibrating only B gives the best overall result, while calibrating A, calibrating the full update ΔW, or removing restoration lowers the overall average.

Calibration Space	Math Avg. ↑	Coding Avg. ↑	Finance Avg. ↑	Medical Avg. ↑	Overall Avg. ↑
Task Arithmetic only	0.2666	0.2823	0.5044	0.5839	0.4093
+ A-space	0.2485	0.2656	0.5139	0.5384	0.3916
+ BA-space	0.2281	0.2378	0.5505	0.4810	0.3743
+ B-space w/o restoration	0.3049	0.2146	0.5646	0.4790	0.3908
+ **B-space (Pico)**	**0.3065**	**0.3372**	**0.5431**	**0.5854**	**0.4430**

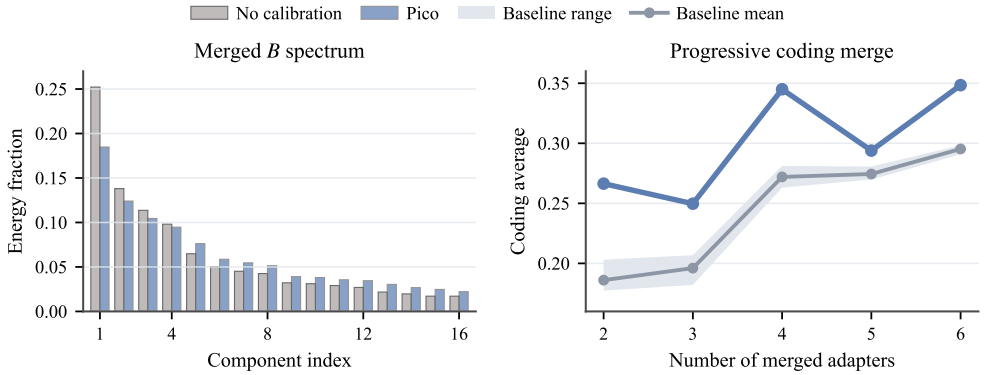

Figure 4: **Spectrum calibration and progressive merge robustness.** Left: representative merged B spectrum for the query projection at layer 16 with Task Arithmetic at rank 16. Pico reduces the dominance of the leading shared components. Right: coding average as the merge pool grows from two to six adapters. The band summarizes the range of non-Pico baselines; Pico remains stronger throughout the progressive merge.Appendix D gives the full spectral metric definitions and exact statistics; Appendix J gives the full progressive-merge comparison.

from 0.4093 to 0.4430 on top of Task Arithmetic. In contrast, calibrating A or the full update $\Delta W = BA$ lowers performance. This matches the motivation analysis: the main merge interference is concentrated in B, while A remains more task-specific. Variants that modify A or the full update act on components that are not the main source of interference and can weaken useful task-specific structure.

The same table also shows that the final magnitude-restoration step is necessary after B-space calibration. Removing restoration drops the overall average to 0.3908 even though the calibrated directions are unchanged. The domain-level effect is not uniform: finance increases from 0.5431 to 0.5646 without restoration, but coding and medical drop sharply, and the overall result is clearly worse. In other words, direction calibration alone is not enough; the merged update also needs its magnitude to be restored. Appendix F provides a complementary min-sharing ablation based on per-task participation.

The merged B space becomes less skewed. Figure 4 shows two complementary views. The left panel is a representative layer-level spectrum of the merged output-side matrix B for Task Arithmetic at rank 16. After Pico, the leading shared-basis components carry less energy, which means that the merged B space is less concentrated in a few over-shared components. This is the same qualitative pattern captured by the spectral diagnostics used in Appendix D: lower $o_{\max}$ means less dominance by a single component, while higher effective rank and stable rank mean that energy is spread more evenly across the shared basis. The right panel shows a harder progressive merge setting on coding benchmarks: as more adapters are added to the merge pool, Pico stays above the baseline range and keeps the coding adapter more stable.

Robustness beyond the main setting. Our main experiments focus on rank 16 on Llama-3.1-8B, but Pico's advantage remains consistent across a broader range of settings. Appendix G reports full results for ranks 8, 16, 32, and 64 across Task Arithmetic, TIES, and TSV-M, where Pico remains the strongest method in every setting. This again matches the motivation analysis: as the LoRA rank grows, overlap in B also grows, so the merged adapter continues to face the same type of over-shared structure before merging. The effect is especially visible for TSV-M, whose no-calibration baseline drops from 0.3473 at rank 16 to 0.3206 at rank 32 and 0.2926 at rank 64, while Pico stays clearly stronger at all three ranks. Appendix H further repeats the Task Arithmetic comparison on Qwen3-4B-Base, where Pico again gives the best overall average, improving the no-calibration baseline from 0.4881 to 0.5557.

6 Related Work

We position Pico against three closely related lines of work: LoRA adaptation, model and LoRA merging, and merge methods that reduce interference in shared subspaces.

LoRA adaptation and asymmetry. LoRA [11] makes parameter-efficient finetuning practical by writing each task update as a low-rank product $\Delta W = BA$. A growing line of work shows that these two matrices do not play symmetric roles. Zhu et al. [36] analyzes this asymmetry in single-task finetuning, and HydraLoRA [27] turns the same idea into an architectural design by sharing one A across multiple task-specific B heads. Other methods also treat the two factors differently in more specific ways: LoRA-FA [33] and LoRA+ [10] assign them different optimization roles, DoRA [19] separates magnitude from direction in the adapted weights, and FedSA-LoRA [9] splits shared and client-specific updates in federated learning. Our work builds on this line of evidence, but studies a different problem: not how asymmetry affects single-task finetuning, but how it affects interference when independently trained LoRA adapters are merged.

Model and LoRA merging. Task Arithmetic [12] shows that finetuned models can often be composed directly in parameter space. More recent work focuses on reducing interference during merge. TIES [30] trims small updates and resolves sign conflicts, DARE [32] sparsifies and rescales delta parameters, and DELLA [7] uses magnitude-based sampling. LoRA Soups [23] further frames LoRA merging as a practical tool for skill composition, while LoRA-LEGO [35] pushes the granularity further by decomposing each adapter into rank-wise semantic units that can be recombined across tasks. These methods reduce merge interference in different ways, but they still operate on task updates or their coordinates rather than asking where the interference comes from inside $\Delta W = BA$.

Shared-basis and low-rank merge methods. Pico is closest to methods that explicitly work in a shared basis or low-rank subspace. KnOTS [25] aligns updates in a shared singular-vector basis, TSV-M [8] merges updates in a low-rank singular-vector space, and Core Space [22] performs low-rank merging in a shared subspace. Li et al. [17] studies a related failure mode: aligned singular directions can be over-accumulated during linear merge, which inflates the merged singular values. Their SVC method corrects the merged spectrum afterward by rescaling those singular values. Pico differs in two ways. It traces the interference back to the LoRA decomposition itself, and it calibrates the output-side matrix B before any downstream merge rule is applied.

7 Conclusion

LoRA merging is a practical alternative to joint multi-task training, but existing methods usually treat the low-rank update $\Delta W = BA$ as a single object. In this paper, we show that the main source of merge interference comes from the output-side matrix B: across tasks, a few shared B components are repeatedly accumulated during merge, while A remains more task-specific. This leads the merged adapter to overemphasize those shared components and makes some task capabilities harder to preserve. Pico follows directly from this diagnosis. It calibrates only B before merging, plugs into existing merge rules, and consistently improves strong LoRA merging baselines across four domains and three downstream mergers. It also yields a stronger multi-domain adapter than a jointly trained multi-task LoRA in our setting. More broadly, these results suggest that adapter merging works better when interference is diagnosed inside the factorization rather than only at the level of the full update.

Limitations

Pico uses spectral energy in the stacked B matrices as a proxy for cross-task sharing, but high energy alone does not establish that a direction causes harmful interference. Since Pico is data-free, it does not directly assess whether downscaling a direction removes useful task information, and calibration may therefore weaken beneficial features in some task combinations. Moreover, improvements in overall average performance do not guarantee gains on every task or full preservation of specialist capabilities. Task-level evaluation remains necessary for applications that prioritize particular capabilities.

Acknowledgments

This work was supported by the HKUST Strategic Fund for Futuristic Business Research under grant SBMSFFBR26BM02.

References

[1] Alignment-Lab-AI (2026). Lawyer-instruct.

[2] Austin, J., Odena, A., Nye, M. I., Bosma, M., Michalewski, H., Dohan, D., Jiang, E., Cai, C. J., Terry, M., Le, Q. V., and Sutton, C. (2021). Program synthesis with large language models. *CoRR*, abs/2108.07732.

[3] Cao, C., Lin, H., Zhong, Z., Gao, X., Cai, M., He, C., Han, S., and Wu, L. (2026). Unlocking data value in finance: A study on distillation and difficulty-aware training.

[4] Chen, M., Tworek, J., Jun, H., Yuan, Q., de Oliveira Pinto, H. P., Kaplan, J., Edwards, H., Burda, Y., Joseph, N., Brockman, G., Ray, A., Puri, R., Krueger, G., Petrov, M., Khlaaf, H., Sastry, G., Mishkin, P., Chan, B., Gray, S., Ryder, N., Pavlov, M., Power, A., Kaiser, L., Bavarian, M., Winter, C., Tillet, P., Such, F. P., Cummings, D., Plappert, M., Chantzis, F., Barnes, E., Herbert-Voss, A., Guss, W. H., Nichol, A., Paino, A., Tezak, N., Tang, J., Babuschkin, I., Balaji, S., Jain, S., Saunders, W., Hesse, C., Carr, A. N., Leike, J., Achiam, J., Misra, V., Morikawa, E., Radford, A., Knight, M., Brundage, M., Murati, M., Mayer, K., Welinder, P., McGrew, B., Amodei, D., McCandlish, S., Sutskever, I., and Zaremba, W. (2021). Evaluating large language models trained on code. *CoRR*, abs/2107.03374.

[5] Chen, Z., Li, S., Smiley, C., Ma, Z., Shah, S., and Wang, W. Y. (2022). Convfinqa: Exploring the chain of numerical reasoning in conversational finance question answering. In *Proceedings of the 2022 Conference on Empirical Methods in Natural Language Processing, EMNLP 2022, Abu Dhabi, United Arab Emirates, December 7-11, 2022*, pages 6279–6292. Association for Computational Linguistics.

[6] Cobbe, K., Kosaraju, V., Bavarian, M., Chen, M., Jun, H., Kaiser, L., Plappert, M., Tworek, J., Hilton, J., Nakano, R., Hesse, C., and Schulman, J. (2021). Training verifiers to solve math word problems. *CoRR*, abs/2110.14168.

[7] Deep, P. T., Bhardwaj, R., and Poria, S. (2024). Della-merging: Reducing interference in model merging through magnitude-based sampling.

[8] Gargiulo, A. A., Crisostomi, D., Bucarelli, M. S., Scardapane, S., Silvestri, F., and Rodolà, E. (2025). Task singular vectors: Reducing task interference in model merging. In *IEEE/CVF Conference on Computer Vision and Pattern Recognition, CVPR 2025, Nashville, TN, USA, June 11-15, 2025*, pages 18695–18705. Computer Vision Foundation / IEEE.

[9] Guo, P., Zeng, S., Wang, Y., Fan, H., Wang, F., and Qu, L. (2025). Selective aggregation for low-rank adaptation in federated learning. In *The Thirteenth International Conference on Learning Representations, ICLR 2025, Singapore, April 24-28, 2025*. OpenReview.net.

[10] Hayou, S., Ghosh, N., and Yu, B. (2024). Lora+: Efficient low rank adaptation of large models. In *Forty-first International Conference on Machine Learning, ICML 2024, Vienna, Austria, July 21-27, 2024*, Proceedings of Machine Learning Research, pages 17783–17806. PMLR / OpenReview.net.

[11] Hu, E. J., yelong shen, Wallis, P., Allen-Zhu, Z., Li, Y., Wang, S., Wang, L., and Chen, W. (2022). LoRA: Low-rank adaptation of large language models. In *International Conference on Learning Representations*.

[12] Ilharco, G., Ribeiro, M. T., Wortsman, M., Schmidt, L., Hajishirzi, H., and Farhadi, A. (2023). Editing models with task arithmetic. In *The Eleventh International Conference on Learning Representations*.

[13] Islam, P., Kannappan, A., Kiela, D., Qian, R., Scherrer, N., and Vidgen, B. (2023). Financebench: A new benchmark for financial question answering. *CoRR*, abs/2311.11944.

[14] Jin, D., Pan, E., Oufattole, N., Weng, W., Fang, H., and Szolovits, P. (2020). What disease does this patient have? A large-scale open domain question answering dataset from medical exams. *CoRR*, abs/2009.13081.

[15] Jin, Q., Dhingra, B., Liu, Z., Cohen, W. W., and Lu, X. (2019). Pubmedqa: A dataset for biomedical research question answering. In *Proceedings of the 2019 Conference on Empirical Methods in Natural Language Processing and the 9th International Joint Conference on Natural Language Processing, EMNLP-IJCNLP 2019, Hong Kong, China, November 3-7, 2019*, pages 2567–2577. Association for Computational Linguistics.

[16] Lee, J., Chen, F., Dua, S., Cer, D., Shanbhogue, M., Naim, I., Ábrego, G. H., Li, Z., Chen, K., Vera, H. S., Ren, X., Zhang, S., Salz, D., Boratko, M., Han, J., Chen, B., Huang, S., Rao, V., Suganthan, P., Han, F., Doumanoglou, A., Gupta, N., Moiseev, F., Yip, C., Jain, A., Baumgartner, S., Shahi, S., Gomez, F. P., Mariserla, S., Choi, M., Shah, P., Goenka, S., Chen, K., Xia, Y., Chen, K., Duddu, S. M. K., Chen, Y., Walker, T., Zhou, W., Ghiya, R., Gleicher, Z., Gill, K., Dong, Z., Seyedhosseini, M., Sung, Y., Hoffmann, R., and Duerig, T. (2025). Gemini embedding: Generalizable embeddings from gemini. *CoRR*, abs/2503.07891.

[17] Li, Y., Peng, Z., Zhang, J., Guo, J., Duan, Y., and Shi, Y. (2026). When shared knowledge hurts: Spectral over-accumulation in model merging.

[18] Lightman, H., Kosaraju, V., Burda, Y., Edwards, H., Baker, B., Lee, T., Leike, J., Schulman, J., Sutskever, I., and Cobbe, K. (2024). Let's verify step by step. In *The Twelfth International Conference on Learning Representations, ICLR 2024, Vienna, Austria, May 7-11, 2024*. OpenReview.net.

[19] Liu, S., Wang, C., Yin, H., Molchanov, P., Wang, Y. F., Cheng, K., and Chen, M. (2024). Dora: Weight-decomposed low-rank adaptation. In *Forty-first International Conference on Machine Learning, ICML 2024, Vienna, Austria, July 21-27, 2024*, Proceedings of Machine Learning Research, pages 32100–32121. PMLR / OpenReview.net.

[20] Marczak, D., Magistri, S., Cygert, S., Twardowski, B., Bagdanov, A. D., and van de Weijer, J. (2025). No task left behind: Isotropic model merging with common and task-specific subspaces. In Singh, A., Fazel, M., Hsu, D., Lacoste-Julien, S., Berkenkamp, F., Maharaj, T., Wagstaff, K., and Zhu, J., editors, *Forty-second International Conference on Machine Learning, ICML 2025, Vancouver, BC, Canada, July 13-19, 2025*, volume 267 of *Proceedings of Machine Learning Research*. PMLR / OpenReview.net.

[21] OpenMed (2025). Medical-reasoning-sft-trinity-mini.

[22] Panariello, A., Marczak, D., Magistri, S., Porrello, A., Twardowski, B., Bagdanov, A. D., Calderara, S., and van de Weijer, J. (2025). Accurate and efficient low-rank model merging in core space. In *The Thirty-ninth Annual Conference on Neural Information Processing Systems*.

[23] Prabhakar, A., Li, Y., Narasimhan, K., Kakade, S., Malach, E., and Jelassi, S. (2025). Lora soups: Merging loras for practical skill composition tasks. In *Proceedings of the 31st International Conference on Computational Linguistics: Industry Track*, pages 644–655.

[24] Skorobogat, R., Roth, K., Georgescu, M., and Akata, Z. (2025). Subspace-boosted model merging. *CoRR*, abs/2506.16506.

[25] Stoica, G., Ramesh, P., Ecsedi, B., Choshen, L., and Hoffman, J. (2025). Model merging with SVD to tie the knots. In *The Thirteenth International Conference on Learning Representations*.

[26] Team, L. (2024). The llama 3 herd of models. *CoRR*, abs/2407.21783.

[27] Tian, C., Shi, Z., Guo, Z., Li, L., and zhong Xu, C. (2024). HydraloRA: An asymmetric loRA architecture for efficient fine-tuning. In *The Thirty-eighth Annual Conference on Neural Information Processing Systems*.

[28] Wei, Y., Wang, Z., Liu, J., Ding, Y., and Zhang, L. (2024). Magicoder: Empowering code generation with oss-instruct. In *Forty-first International Conference on Machine Learning, ICML 2024, Vienna, Austria, July 21-27, 2024*, Proceedings of Machine Learning Research, pages 52632–52657. PMLR / OpenReview.net.

[29] Wortsman, M., Ilharco, G., Gadre, S. Y., Roelofs, R., Lopes, R. G., Morcos, A. S., Namkoong, H., Farhadi, A., Carmon, Y., Kornblith, S., and Schmidt, L. (2022). Model soups: averaging weights of multiple fine-tuned models improves accuracy without increasing inference time. In Chaudhuri, K., Jegelka, S., Song, L., Szepesvári, C., Niu, G., and Sabato, S., editors, *International Conference on Machine Learning, ICML 2022, 17-23 July 2022, Baltimore, Maryland, USA*, Proceedings of Machine Learning Research, pages 23965–23998. PMLR.

[30] Yadav, P., Tam, D., Choshen, L., Raffel, C., and Bansal, M. (2023). TIES-merging: Resolving interference when merging models. In *Thirty-seventh Conference on Neural Information Processing Systems*.

[31] Yu, L., Jiang, W., Shi, H., Yu, J., Liu, Z., Zhang, Y., Kwok, J. T., Li, Z., Weller, A., and Liu, W. (2024a). Metamath: Bootstrap your own mathematical questions for large language models. In *The Twelfth International Conference on Learning Representations, ICLR 2024, Vienna, Austria, May 7-11, 2024*. OpenReview.net.

[32] Yu, L., Yu, B., Yu, H., Huang, F., and Li, Y. (2024b). Language models are super mario: absorbing abilities from homologous models as a free lunch. In *Proceedings of the 41st International Conference on Machine Learning*, ICML'24. JMLR.org.

[33] Zhang, L., Zhang, L., Shi, S., Chu, X., and Li, B. (2023). Lora-fa: Memory-efficient low-rank adaptation for large language models fine-tuning. *CoRR*, abs/2308.03303.

[34] Zhang, Y., Li, M., Long, D., Zhang, X., Lin, H., Yang, B., Xie, P., Yang, A., Liu, D., Lin, J., Huang, F., and Zhou, J. (2025). Qwen3 embedding: Advancing text embedding and reranking through foundation models. *CoRR*, abs/2506.05176.

[35] Zhao, Z., Shen, T., Zhu, D., Li, Z., Su, J., Wang, X., and Wu, F. (2025). Merging loras like playing LEGO: pushing the modularity of lora to extremes through rank-wise clustering. In *The Thirteenth International Conference on Learning Representations, ICLR 2025, Singapore, April 24-28, 2025*. OpenReview.net.

[36] Zhu, J., Greenewald, K., Nadjahi, K., Sáez De Ocáriz Borde, H., Gabrielsson, R. B., Choshen, L., Ghassemi, M., Yurochkin, M., and Solomon, J. (2024). Asymmetry in low-rank adapters of foundation models. In *Proceedings of the 41st International Conference on Machine Learning*, volume 235 of *Proceedings of Machine Learning Research*, pages 62369–62385. PMLR.

Appendix Contents

A Additional Motivation Analysis **14**

B Additional Training Details **15**

C Benchmark-Level Results **15**

D Definitions and Full Spectral Statistics of the Merged B Space **16**

E Random-Subspace and Initialization Controls **16**

F Sharing-Gated Calibration **17**

G Additional Rank-Robustness Results **17**

H Additional Results on Qwen3-4B-Base **18**

I Comparison with Spectrum-Adjusting Merging Methods **18**

J Progressive LoRA Merging on Coding **19**

A Additional Motivation Analysis

This appendix supports the motivation analysis in Section 3. We first give a toy derivation showing how shared directions can dominate a linear merge, then define the diagnostics used in the main text, and finally report the exact statistics behind Figure 2. Unless otherwise noted, all measurements are computed from the four domain-specific LoRAs used in the main text: Math, Coding, Finance, and Medical. The corresponding training datasets are MetaMathQA [31], Magicoder [28], ODA-Fin-SFT-318k [3], and Medical-Reasoning-SFT-Trinity-Mini [21].

Toy derivation for repeated counting. Consider a linear average of T adapters. Suppose each update contains one shared unit direction u_0 with coefficient a, and one task-specific unit direction u_t with coefficient b, where $u_0, u_1, \ldots, u_T$ are orthonormal:

$$\Delta W_t = a u_0 + b u_t.$$

The averaged update is

$$\Delta W_{\text{avg}} = \frac{1}{T}\sum_{t=1}^{T} \Delta W_t = a u_0 + \frac{b}{T}\sum_{t=1}^{T} u_t.$$

The shared component retains its full coefficient a, while each task-specific component is reduced to b/T. The shared-to-specific ratio therefore grows by a factor of T. The overall norm also drops from $\sqrt{a^2 + b^2}$ per source adapter to $\sqrt{a^2 + b^2/T}$ for the merged update, which motivates the magnitude rescaling in Pico (Section 4.2). This derivation does not imply that every shared direction is harmful; it only shows why directions reused by many adapters can dominate a linear merge.

Subspace overlap. For a pair of adapters i and j, let Q_i^B and Q_j^B be orthonormal bases for the column spaces of B_i and B_j, and let Q_i^A and Q_j^A be orthonormal bases for the row spaces of A_i and A_j. The normalized overlap scores are

$$O_B(i,j) = \frac{1}{r}\left\|(Q_i^B)^\top Q_j^B\right\|_F^2, \qquad O_A(i,j) = \frac{1}{r}\left\|(Q_i^A)^\top Q_j^A\right\|_F^2.$$

Both scores lie in $[0, 1]$, and larger values mean that the two adapters share more directions. We report both the layerwise values and their averages across domain pairs.

Effective rank. Given singular values $\{\sigma_k\}$ of a matrix, the effective rank is $\mathrm{EffRank} = \exp\big(-\sum_k p_k \log p_k\big)$ with $p_k = \sigma_k / \sum_\ell \sigma_\ell$. A lower effective rank means that most of the singular-value mass is concentrated in fewer directions.

Component energy. For a fixed decomposition with component coefficients $\{c_j\}$, we define the energy of component j as the normalized squared coefficient

$$e_j = \frac{c_j^2}{\sum_k c_k^2}.$$

For singular components, $c_j = \sigma_j$. Intuitively, component energy is the fraction of the total squared magnitude carried by one component; high energy means that the decomposition is dominated by that component.

Table 4 gives the exact values behind the two trends in Figure 2. The gap between O_B and O_A grows steadily with LoRA rank, and the effective rank of B stays much lower than that of A throughout. The asymmetry is already almost universal at rank 8 and becomes universal from rank 16 onward.

Figure 5 complements the averaged curves in the main text with pair-level evidence. At rank 64, every domain pair satisfies $O_B > O_A$ in both the query and value projection matrices. This shows that the asymmetry is not driven by one unusually similar pair of tasks, but is shared across the full set of domain combinations.

Tables 5 and 6 provide the corresponding exact values. They make it easy to verify that the same pattern holds in both modules and across all LoRA ranks, not only at the averaged level reported in the main text.

Table 4: **Exact statistics underlying Figure 2.** $\mathrm{Frac}[O_B > O_A]$ denotes the fraction of layer pairs for which overlap on the output-side matrix exceeds overlap on the input-side matrix.

Rank	Module	Mean O_B	Mean O_A	Gap	$\mathrm{Frac}[O_B > O_A]$	EffRank(B)	EffRank(A)
8	Query projection	0.0166	0.0035	0.0131	0.995	2.431	4.112
8	Value projection	0.0136	0.0071	0.0065	0.953	2.274	3.564
16	Query projection	0.0341	0.0055	0.0286	1.000	2.715	5.155
16	Value projection	0.0234	0.0089	0.0145	1.000	2.413	4.290
32	Query projection	0.0571	0.0095	0.0476	1.000	2.859	5.987
32	Value projection	0.0433	0.0126	0.0307	1.000	2.529	4.845
64	Query projection	0.0839	0.0172	0.0667	1.000	2.944	6.552
64	Value projection	0.0801	0.0196	0.0604	1.000	2.586	5.302

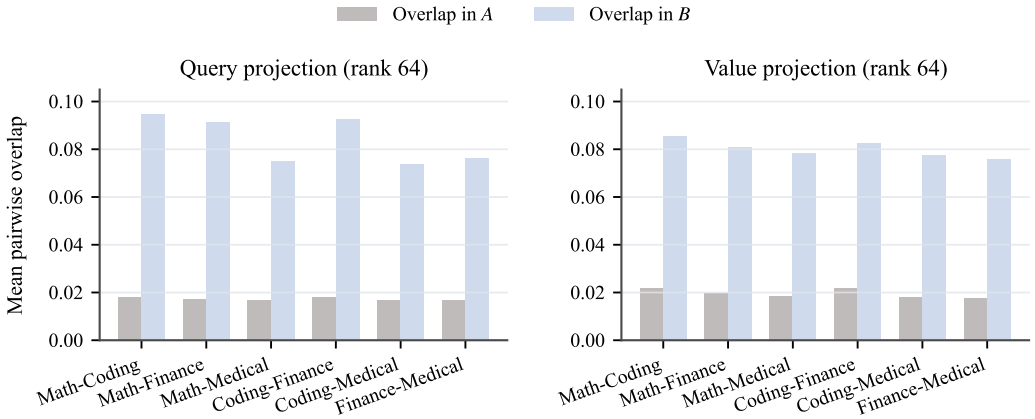

Figure 5: **Pairwise overlap at rank 64.** The asymmetry between B and A holds for every domain pair, showing that the effect is not driven by averaging across tasks.

B Additional Training Details

All four domain-specific adapters are trained with the same distributed setup on 4×H800 GPUs. We use a per-device batch size of 2, gradient accumulation of 8, and train for 3 epochs with AdamW optimization, bf16, gradient checkpointing, and a cosine learning-rate schedule. The learning rate is 2×10^{-4} with a warmup ratio of 0.03. We set the maximum sequence length to 8192, disable packing, apply LoRA to `q_proj` and `v_proj`, and use LoRA dropout 0.05. The LoRA rank varies by experiment, while LoRA alpha is fixed at 16 throughout the paper.

C Benchmark-Level Results

The main text reports domain averages to keep the comparison focused on the main message. This appendix gives the corresponding benchmark-level values for the two main experiment tables.

Table 7 shows that the gains in the main text are not driven by a single benchmark. Pico improves the strongest downstream mergers across multiple benchmarks within each domain, with especially clear gains for Task Arithmetic on MATH and MBPP, for TIES on MATH, MBPP, and MedQA-USMLE, and for TSV-M on MATH, HumanEval, MBPP, FinanceBench, ConvFinQA, and PubMedQA. The benchmark-level view also makes the different baseline behaviors easier to see: TIES and TSV-M are particularly uneven without calibration, while Pico produces a more stable pattern across domains.

Table 8 complements this comparison by showing why merging is needed in the first place. Single-domain LoRAs remain strongest on or near their source domains, but they do not provide the best overall multi-domain performance. The best merged adapter, Pico + Task Arithmetic, gives the highest overall average among the adapters compared here, while also remaining competitive on several individual benchmarks outside the source domain of any single specialist.

Table 5: **Exact pair-level overlap statistics for the query projection.** Each entry reports the mean overlap for one domain pair at a fixed LoRA rank.

	$r = 8$			$r = 16$			$r = 32$			$r = 64$		
Domain pair	O_B	O_A	Gap	O_B	O_A	Gap	O_B	O_A	Gap	O_B	O_A	Gap
Math–Coding	0.0197	0.0040	0.0157	0.0413	0.0061	0.0353	0.0663	0.0102	0.0561	0.0947	0.0179	0.0768
Math–Finance	0.0209	0.0034	0.0175	0.0403	0.0056	0.0347	0.0656	0.0095	0.0561	0.0913	0.0173	0.0740
Math–Medical	0.0131	0.0035	0.0097	0.0278	0.0053	0.0225	0.0479	0.0093	0.0385	0.0750	0.0168	0.0582
Coding–Finance	0.0201	0.0040	0.0161	0.0413	0.0060	0.0352	0.0667	0.0102	0.0565	0.0928	0.0180	0.0747
Coding–Medical	0.0119	0.0029	0.0090	0.0260	0.0049	0.0211	0.0472	0.0089	0.0383	0.0738	0.0167	0.0571
Finance–Medical	0.0135	0.0029	0.0107	0.0280	0.0051	0.0230	0.0491	0.0090	0.0401	0.0760	0.0167	0.0593

Table 6: **Exact pair-level overlap statistics for the value projection.** Each entry reports the mean overlap for one domain pair at a fixed LoRA rank.

	$r = 8$			$r = 16$			$r = 32$			$r = 64$		
Domain pair	O_B	O_A	Gap	O_B	O_A	Gap	O_B	O_A	Gap	O_B	O_A	Gap
Math–Coding	0.0158	0.0106	0.0052	0.0261	0.0122	0.0139	0.0476	0.0155	0.0321	0.0853	0.0219	0.0634
Math–Finance	0.0135	0.0071	0.0064	0.0233	0.0087	0.0146	0.0444	0.0127	0.0317	0.0803	0.0198	0.0604
Math–Medical	0.0135	0.0060	0.0075	0.0236	0.0076	0.0159	0.0452	0.0113	0.0309	0.0784	0.0184	0.0601
Coding–Finance	0.0144	0.0088	0.0056	0.0250	0.0112	0.0138	0.0451	0.0151	0.0300	0.0824	0.0218	0.0606
Coding–Medical	0.0118	0.0054	0.0064	0.0210	0.0074	0.0137	0.0406	0.0108	0.0298	0.0774	0.0181	0.0593
Finance–Medical	0.0125	0.0046	0.0079	0.0216	0.0064	0.0152	0.0399	0.0102	0.0297	0.0759	0.0176	0.0583

D Definitions and Full Spectral Statistics of the Merged B Space

This appendix defines the spectral diagnostics used in Figure 4 and reports their exact values. These diagnostics let us check whether the merged output-side matrix B is dominated by a few over-shared components or spreads its energy more evenly across the shared basis.

We use four diagnostics to summarize the spectrum of a merged B matrix. The first is $o_{\max}$, the largest normalized energy share among the singular components. A lower $o_{\max}$ means that no single shared component dominates the spectrum. The second is effective rank, which measures how evenly the singular-value mass is distributed. The third is stable rank, which plays a similar role: higher effective rank and higher stable rank both indicate that energy is spread across more components instead of concentrating in only a few. We also report the condition number as an additional measure of spectral skewness, where lower values are better. Finally, Frobenius norm is included only as a reference magnitude and does not have a preferred direction.

For completeness, if $\{\sigma_j\}$ are the singular values of the merged B matrix, then the normalized energy share of component j is $o_j = \sigma_j^2 / \sum_k \sigma_k^2$, and $o_{\max}$ is the largest such value. Effective rank is computed from the normalized singular values, and stable rank is $\|B\|_F^2 / \|B\|_2^2$.

E Random-Subspace and Initialization Controls

The normalized subspace overlap has a simple random reference. For two independent random rank-r subspaces in a d-dimensional ambient space, the expected overlap is r/d. Table 10 compares the measured query-projection overlaps with this reference, using $d = 4096$ for both A and B. Across ranks, O_B exceeds its random expectation by 5.4–8.7 times, whereas O_A remains within 1.1–1.8 times its reference.

We next test whether initialization affects which factor carries the excess overlap. We train four domain adapters with the standard initialization and four with the reversed initialization, using independent 1000-step runs with the same data and training budget. The standard setting initializes A with Kaiming initialization and B to zero, while the reversed setting swaps these choices. Table 11 reports the excess overlap above the random reference at rank 16. Under standard initialization, the excess is concentrated in B. Reversing the initialization instead places the excess in A. Thus, initialization determines which factor carries the shared structure, and Pico targets B under the standard LoRA initialization used in our experiments.

Table 7: **Benchmark-level main results across four domains.** This is the detailed version of Table 1.

Method	Merger Used	Math		Coding		Finance		Medical		Average
		GSM8K	MATH	HumanEval	MBPP	FinanceBench	ConvFinQA	PubMedQA	MedQA-USMLE	
No Calibration	Task Arithmetic	0.1626	0.3707	0.2805	0.2840	0.5067	0.5020	0.5850	0.5829	0.4093
DARE	Task Arithmetic	0.1746	0.3768	0.2866	0.2996	0.5467	0.4946	0.5850	0.5852	0.4186
DELLA	Task Arithmetic	0.1732	0.3571	0.2561	0.2763	0.5333	0.5040	0.5800	0.5915	0.4089
KnOTS	Task Arithmetic	0.1656	0.3798	0.2744	0.2685	0.5200	0.5054	0.5780	0.5789	0.4088
Core Space	Task Arithmetic	0.1600	0.3639	0.2744	0.2802	0.5000	0.4973	0.5900	0.5860	0.4065
Pico	Task Arithmetic	0.1672	0.4458	0.3049	0.3696	0.5733	0.5128	0.5650	0.6057	**0.4430**
No Calibration	TIES	0.1658	0.3958	0.2988	0.1245	0.5467	0.5074	0.4810	0.5672	0.3859
DARE	TIES	0.1638	0.3563	0.2744	0.1128	0.5400	0.4859	0.4300	0.5648	0.3660
DELLA	TIES	0.1240	0.2616	0.2134	0.1284	0.5267	0.4234	0.2520	0.5530	0.3103
KnOTS	TIES	0.1704	0.4049	0.2988	0.3035	0.5400	0.5155	0.5800	0.5522	0.4207
Core Space	TIES	0.1580	0.3245	0.1829	0.3152	0.5200	0.3259	0.3540	0.6332	0.3517
Pico	TIES	0.1786	0.4337	0.2805	0.3268	0.5133	0.5060	0.5610	0.6622	**0.4328**
No Calibration	TSV-M	0.1754	0.3518	0.1585	0.1556	0.4667	0.4308	0.2500	0.7895	0.3473
DARE	TSV-M	0.2076	0.3776	0.1646	0.1595	0.4467	0.4496	0.3800	0.7918	0.3722
DELLA	TSV-M	0.1872	0.3685	0.1524	0.1673	0.4733	0.4590	0.2610	0.7926	0.3577
KnOTS	TSV-M	0.1846	0.3487	0.1646	0.1518	0.4733	0.4516	0.2550	0.7887	0.3523
Core Space	TSV-M	0.1836	0.2934	0.1585	0.1245	0.4400	0.2735	0.1190	0.7636	0.2945
Pico	TSV-M	0.1740	0.4026	0.2927	0.2957	0.5400	0.5114	0.5720	0.6559	**0.4305**

Table 8: **Benchmark-level comparison with single-domain and jointly trained LoRA adapters.** This is the detailed version of Table 2.

Adapter	Math		Coding		Finance		Medical		Average
	GSM8K	MATH	HumanEval	MBPP	FinanceBench	ConvFinQA	PubMedQA	MedQA-USMLE	
Math LoRA	0.1892	0.3768	0.1402	0.0778	0.4600	0.4207	0.2930	0.4399	0.2997
Coding LoRA	0.1728	0.3283	0.3110	0.4086	0.4867	0.4758	0.5510	0.5522	0.4108
Finance LoRA	0.1640	0.3313	0.1829	0.1051	0.5667	0.5302	0.2420	0.7533	0.3594
Medical LoRA	0.1660	0.3215	0.2683	0.3696	0.5333	0.4362	0.4170	0.7730	0.4106
Joint Multi-Task LoRA	0.1774	0.3313	0.3110	0.3191	0.6333	0.2977	0.2650	0.6159	0.3688
Pico w/ Task Arithmetic	0.1672	0.4458	0.3049	0.3696	0.5733	0.5128	0.5650	0.6057	**0.4430**

F Sharing-Gated Calibration

The sharing score s_j is based on joint spectral energy. We therefore add an ablation that directly restricts calibration according to per-task participation. For joint direction u_j, define the energy contributed by task t and its normalized share as

$$c_{tj} = \|u_j^\top B_t\|_2^2, \qquad p_{tj} = \frac{c_{tj}}{\sum_{s=1}^{T} c_{sj}}.$$

For a gate of size m, we set $\tau = 1/(2m)$ and apply the Pico coefficient α_j only when at least m tasks satisfy $p_{tj} \geq \tau$. All other directions retain $\alpha_j = 1$, and the merge and rescaling steps remain unchanged.

Table 12 reports the resulting performance on top of Task Arithmetic at rank 16. The $m = 2$ variant retains 2.9 of the 3.4 points gained by Pico over Task Arithmetic. Even the strictest $m = 4$ gate, which requires substantial contributions from every task, retains 2.5 points. Most of the improvement therefore remains when calibration is restricted to directions with verified cross-task participation.

G Additional Rank-Robustness Results

This appendix gives the full rank-sweep results that supplement the short discussion in Section 5.4. The goal is to check whether the advantage of calibrating B before merge remains visible when the LoRA rank changes.

Table 13 shows that the pattern in the main text continues across all four ranks. Pico remains the strongest method for Task Arithmetic, TIES, and TSV-M at ranks 8, 16, 32, and 64. The gap is especially clear for TSV-M, whose uncalibrated performance drops sharply at higher ranks, while Pico stays clearly stronger throughout the full rank sweep.

Table 9: **Full spectral statistics of the merged B space.** Frobenius norm is reported as a reference magnitude. Lower $o_{\max}$ and condition number are better; higher effective rank and stable rank are better.

Adapter	Frobenius Norm (ref.)	$o_{\max} \downarrow$	Effective Rank ↑	Stable Rank ↑	Condition Number ↓
Merged adapters					
Pico	2.4547	0.2192	11.4733	4.7581	4.0219
No Calibration	1.7220	0.3268	9.2866	3.2293	5.8863
DARE	1.7247	0.3249	9.3698	3.2438	5.5169
KnOTS	1.7220	0.3268	9.2868	3.2292	5.8683
Source-domain adapters					
Math LoRA	1.1537	0.7699	2.3481	1.3302	33.7237
Finance LoRA	1.5313	0.5859	4.2613	1.8803	20.4689
Medical LoRA	1.2375	0.7263	2.6762	1.4430	29.4735
Coding LoRA	1.1778	0.7477	2.5098	1.4018	31.4788

Table 10: **Query-projection overlap against the random-subspace reference.** The random expectation is $r/4096$. The final column repeats the fraction of cross-task layer comparisons for which $O_B > O_A$.

Rank	Random overlap	O_B	O_B/Random	O_A	O_A/Random	$\mathrm{Frac}[O_B > O_A]$
8	0.0020	0.0166	8.5	0.0035	1.8	0.995
16	0.0039	0.0341	8.7	0.0055	1.4	1.000
32	0.0078	0.0571	7.3	0.0095	1.2	1.000
64	0.0156	0.0839	5.4	0.0172	1.1	1.000

H Additional Results on Qwen3-4B-Base

This appendix reports the base-model transfer result, summarized briefly in Section 5.4. Table 14 repeats the Task Arithmetic comparison on Qwen3-4B-Base under the same four-domain setup and shows that Pico again gives the best overall average.

Table 14 shows that the overall pattern from the main text survives the backbone change. Pico again gives the best overall average, with especially large gains in math and medical, while coding remains strong instead of dropping sharply.

I Comparison with Spectrum-Adjusting Merging Methods

Iso-C and Iso-CTS [20] adjust the spectrum of the reconstructed full update during merging. Iso-C assigns the mean singular value to all directions, while Iso-CTS retains common and task-specific subspaces before applying a shared isotropic scale. Subspace-Boosted [24] instead raises small singular values of the merged update to a cutoff after merging. Pico differs in both the calibration space and the intervention stage, since it downscales directions in the joint B basis before merging.

We evaluate all three methods on the eight-benchmark suite using the default configurations of their official implementations. Iso-C and Iso-CTS are standalone mergers, while Subspace-Boosted and Pico are applied on top of Task Arithmetic at rank 16. Table 15 gives the full benchmark-level comparison.

Under these configurations, none of the three comparison methods improves the overall average of Task Arithmetic. Pico raises it from 0.4093 to 0.4430. Applying Subspace-Boosted after Pico gives a further improvement to 0.4659. This combination separates two compatible effects: Pico provides pre-merge calibration of the source adapters, while Subspace-Boosted can provide additional post-merge spectrum adjustment. The combined result therefore extends, rather than contradicts, the main conclusion that Pico improves existing merging methods.

Table 11: **Initialization control at rank 16.** Entries report excess overlap above the random-subspace reference for independently trained 1000-step control adapters.

Initialization	A init	B init	Query O_B	Query O_A	Value O_B	Value O_A
Standard	Kaiming	Zero	0.0630	0.0115	0.0724	0.0209
Reversed	Zero	Kaiming	0.0001	0.0135	0.0005	0.0080

Table 12: **Min-sharing gated variants on the eight-benchmark suite.** All methods use Task Arithmetic at rank 16. A gated variant applies calibration only to directions with at least m task shares satisfying $p_{tj} \geq 1/(2m)$.

Method	MATH	GSM8K	HumanEval	MBPP	FinanceBench	ConvFinQA	PubMedQA	MedQA-USMLE	Average
Task Arithmetic	0.3707	0.1626	0.2805	0.2840	0.5067	0.5020	0.5850	0.5829	0.4093
Pico	0.4458	0.1672	0.3049	0.3696	0.5733	0.5128	0.5650	0.6057	0.4430
Pico, min-sharing ($m = 2$)	0.4428	0.1670	0.2805	0.3580	0.5760	0.5155	0.5640	0.6017	0.4382
Pico, min-sharing ($m = 4$)	0.4329	0.1680	0.2988	0.3074	0.5820	0.5208	0.5640	0.6009	0.4344

J Progressive LoRA Merging on Coding

This appendix studies a progressive merge protocol in which the adapter pool grows over time. We start from the coding adapter and then add the finance, medical, math, and law adapters one at a time, evaluating the coding average after each stage. The law adapter is trained on Lawyer-Instruct [1]. Figure 6 reports the resulting coding average as the number of merged adapters increases from two to six.

Figure 6 shows that the coding curve is not monotonic. Adding another adapter can hurt coding, but it can also help. A new adapter may introduce extra interference, and it may also bring in structure that transfers well to coding. That is why some baselines improve when the merge grows from three adapters to four instead of dropping at every step.

The main pattern is that the baseline methods become less stable as more independently trained updates are merged, while Pico remains strongest at every stage. This matches the main story of the paper. With more adapters in the merge, the same shared B directions are more likely to be counted repeatedly across tasks. Pico calibrates those directions before merge, which makes coding performance more stable in this harder setting.

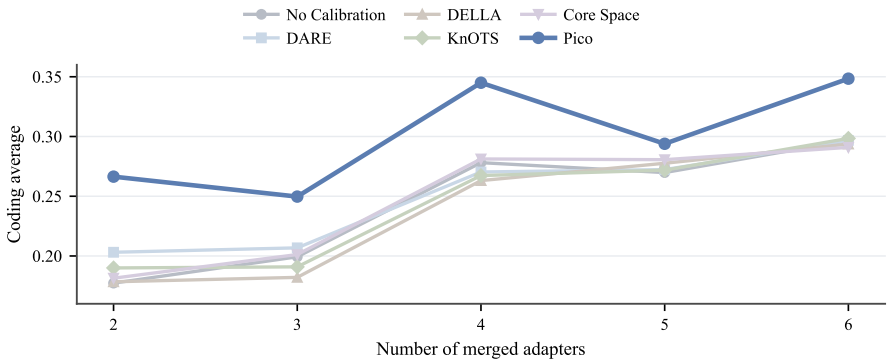

Figure 6: **Progressive LoRA merging on coding benchmarks.** The x-axis shows the number of merged adapters. Starting from the coding adapter, we progressively add finance, medical, math, and law adapters and report the coding average after each merge. Pico remains the strongest method at every stage.

Table 13: **Robustness across LoRA ranks.** Each entry reports the overall average across all eight benchmarks. Pico remains the strongest method for all three downstream mergers at ranks 8, 16, 32, and 64.

Rank	Merge Method	No Calibration	DARE	DELLA	KnOTS	Core Space	Pico
8	Task Arithmetic	0.4285	0.4404	0.4473	0.4412	0.4662	**0.4890**
	TIES	0.4816	0.4857	0.4353	0.4997	0.4395	**0.5008**
	TSV-M	0.3105	0.2811	0.2790	0.2957	0.2807	**0.4847**
16	Task Arithmetic	0.4093	0.4186	0.4089	0.4088	0.4065	**0.4430**
	TIES	0.3859	0.3660	0.3103	0.4207	0.3517	**0.4328**
	TSV-M	0.3473	0.3722	0.3577	0.3523	0.2945	**0.4305**
32	Task Arithmetic	0.4754	0.4679	0.4505	0.4823	0.4802	**0.5279**
	TIES	0.5005	0.4827	0.4748	0.5029	0.4801	**0.5043**
	TSV-M	0.3206	0.3031	0.3096	0.3173	0.2977	**0.4621**
64	Task Arithmetic	0.4257	0.4504	0.4400	0.4270	0.4633	**0.5221**
	TIES	0.4858	0.4878	0.4640	0.4870	0.4749	**0.4941**
	TSV-M	0.2926	0.3052	0.2919	0.2909	0.3011	**0.3976**

Table 14: **Transfer to Qwen3-4B-Base.** Domain averages are computed from the same eight benchmarks as in the main experiment. Pico again gives the best overall average.

Method	Math Avg. ↑	Coding Avg. ↑	Finance Avg. ↑	Medical Avg. ↑	Overall Avg. ↑
No Calibration	0.1952	0.5798	0.6144	0.5632	0.4881
DARE	0.2607	0.5718	0.6167	0.5539	0.5008
DELLA	0.2326	0.5737	0.5996	0.5574	0.4908
KnOTS	0.2622	0.5593	0.6113	0.5661	0.4997
Core-TA	0.1264	0.5563	0.5515	0.5187	0.4382
Pico	**0.3265**	**0.5825**	**0.6545**	**0.6594**	**0.5557**

Table 15: **Comparison with spectrum-adjusting merging methods.** All methods use the same four Llama-3.1-8B LoRA adapters at rank 16. The final row evaluates the complementarity between pre-merge Pico calibration and post-merge Subspace-Boosted adjustment.

Method	MATH	GSM8K	HumanEval	MBPP	FinanceBench	ConvFinQA	PubMedQA	MedQA-USMLE	Average
Task Arithmetic	0.3707	0.1626	0.2805	0.2840	0.5067	0.5020	0.5850	0.5829	0.4093
Iso-C	0.2616	0.1818	0.1159	0.0661	0.5867	0.1915	0.2500	0.7416	0.2994
Iso-CTS	0.2646	0.1834	0.1220	0.0545	0.6067	0.1922	0.1450	0.7439	0.2890
TA + Subspace-Boosted	0.3169	0.2106	0.0183	0.0934	0.5733	0.2762	0.2070	0.7824	0.3098
TA + Pico	0.4458	0.1672	0.3049	0.3696	0.5733	0.5128	0.5650	0.6057	0.4430
TA + Pico + Subspace-Boosted	0.4458	0.1830	0.3232	0.3230	0.7200	0.5141	0.5860	0.6324	0.4659

NeurIPS Paper Checklist

1. **Claims**

   Question: Do the main claims made in the abstract and introduction accurately reflect the paper's contributions and scope?

   Answer: [Yes]

   Justification: The three contributions stated in the abstract and introduction are directly supported by Section 3 (diagnosis on B), Section 4.1–4.2 (Pico method), and Section 5.2 (consistent gains over baselines).

   Guidelines:

   - The answer [N/A] means that the abstract and introduction do not include the claims made in the paper.
   - The abstract and/or introduction should clearly state the claims made, including the contributions made in the paper and important assumptions and limitations. A [No] or [N/A] answer to this question will not be perceived well by the reviewers.
   - The claims made should match theoretical and experimental results, and reflect how much the results can be expected to generalize to other settings.
   - It is fine to include aspirational goals as motivation as long as it is clear that these goals are not attained by the paper.

2. **Limitations**

   Question: Does the paper discuss the limitations of the work performed by the authors?

   Answer: [Yes]

   Justification: The scope of the claims and the role of each design choice are discussed in Section 5.4 and the corresponding appendices (rank sweep, backbone transfer, and progressive-merge robustness).

   Guidelines:

   - The answer [N/A] means that the paper has no limitation while the answer [No] means that the paper has limitations, but those are not discussed in the paper.
   - The authors are encouraged to create a separate "Limitations" section in their paper.
   - The paper should point out any strong assumptions and how robust the results are to violations of these assumptions (e.g., independence assumptions, noiseless settings, model well-specification, asymptotic approximations only holding locally). The authors should reflect on how these assumptions might be violated in practice and what the implications would be.
   - The authors should reflect on the scope of the claims made, e.g., if the approach was only tested on a few datasets or with a few runs. In general, empirical results often depend on implicit assumptions, which should be articulated.
   - The authors should reflect on the factors that influence the performance of the approach. For example, a facial recognition algorithm may perform poorly when image resolution is low or images are taken in low lighting. Or a speech-to-text system might not be used reliably to provide closed captions for online lectures because it fails to handle technical jargon.
   - The authors should discuss the computational efficiency of the proposed algorithms and how they scale with dataset size.
   - If applicable, the authors should discuss possible limitations of their approach to address problems of privacy and fairness.
   - While the authors might fear that complete honesty about limitations might be used by reviewers as grounds for rejection, a worse outcome might be that reviewers discover limitations that aren't acknowledged in the paper. The authors should use their best judgment and recognize that individual actions in favor of transparency play an important role in developing norms that preserve the integrity of the community. Reviewers will be specifically instructed to not penalize honesty concerning limitations.

3. **Theory assumptions and proofs**

Question: For each theoretical result, does the paper provide the full set of assumptions and a complete (and correct) proof?

Answer: [Yes]

Justification: The assumptions are illustrated in Appendix A.

Guidelines:

- The answer [N/A] means that the paper does not include theoretical results.
- All the theorems, formulas, and proofs in the paper should be numbered and cross-referenced.
- All assumptions should be clearly stated or referenced in the statement of any theorems.
- The proofs can either appear in the main paper or the supplemental material, but if they appear in the supplemental material, the authors are encouraged to provide a short proof sketch to provide intuition.
- Inversely, any informal proof provided in the core of the paper should be complemented by formal proofs provided in appendix or supplemental material.
- Theorems and Lemmas that the proof relies upon should be properly referenced.

4. **Experimental result reproducibility**

Question: Does the paper fully disclose all the information needed to reproduce the main experimental results of the paper to the extent that it affects the main claims and/or conclusions of the paper (regardless of whether the code and data are provided or not)?

Answer: [Yes]

Justification: All base models and datasets are public, the experimental setup and hyperparameters are given in Section 5.1 and Appendix B.

Guidelines:

- The answer [N/A] means that the paper does not include experiments.
- If the paper includes experiments, a [No] answer to this question will not be perceived well by the reviewers: Making the paper reproducible is important, regardless of whether the code and data are provided or not.
- If the contribution is a dataset and/or model, the authors should describe the steps taken to make their results reproducible or verifiable.
- Depending on the contribution, reproducibility can be accomplished in various ways. For example, if the contribution is a novel architecture, describing the architecture fully might suffice, or if the contribution is a specific model and empirical evaluation, it may be necessary to either make it possible for others to replicate the model with the same dataset, or provide access to the model. In general. releasing code and data is often one good way to accomplish this, but reproducibility can also be provided via detailed instructions for how to replicate the results, access to a hosted model (e.g., in the case of a large language model), releasing of a model checkpoint, or other means that are appropriate to the research performed.
- While NeurIPS does not require releasing code, the conference does require all submissions to provide some reasonable avenue for reproducibility, which may depend on the nature of the contribution. For example
  (a) If the contribution is primarily a new algorithm, the paper should make it clear how to reproduce that algorithm.
  (b) If the contribution is primarily a new model architecture, the paper should describe the architecture clearly and fully.
  (c) If the contribution is a new model (e.g., a large language model), then there should either be a way to access this model for reproducing the results or a way to reproduce the model (e.g., with an open-source dataset or instructions for how to construct the dataset).
  (d) We recognize that reproducibility may be tricky in some cases, in which case authors are welcome to describe the particular way they provide for reproducibility. In the case of closed-source models, it may be that access to the model is limited in some way (e.g., to registered users), but it should be possible for other researchers to have some path to reproducing or verifying the results.

5. **Open access to data and code**

   Question: Does the paper provide open access to the data and code, with sufficient instructions to faithfully reproduce the main experimental results, as described in supplemental material?

   Answer: [Yes]

   Justification: The main code is given in the supplemental material.

   Guidelines:

   - The answer [N/A] means that paper does not include experiments requiring code.
   - Please see the NeurIPS code and data submission guidelines (`https://neurips.cc/public/guides/CodeSubmissionPolicy`) for more details.
   - While we encourage the release of code and data, we understand that this might not be possible, so [No] is an acceptable answer. Papers cannot be rejected simply for not including code, unless this is central to the contribution (e.g., for a new open-source benchmark).
   - The instructions should contain the exact command and environment needed to run to reproduce the results. See the NeurIPS code and data submission guidelines (`https://neurips.cc/public/guides/CodeSubmissionPolicy`) for more details.
   - The authors should provide instructions on data access and preparation, including how to access the raw data, preprocessed data, intermediate data, and generated data, etc.
   - The authors should provide scripts to reproduce all experimental results for the new proposed method and baselines. If only a subset of experiments are reproducible, they should state which ones are omitted from the script and why.
   - At submission time, to preserve anonymity, the authors should release anonymized versions (if applicable).
   - Providing as much information as possible in supplemental material (appended to the paper) is recommended, but including URLs to data and code is permitted.

6. **Experimental setting/details**

   Question: Does the paper specify all the training and test details (e.g., data splits, hyperparameters, how they were chosen, type of optimizer) necessary to understand the results?

   Answer: [Yes]

   Justification: Section 5.1 describes the experimental setup, and Appendix B lists the full training hyperparameters; the rank sweep is reported in Appendix G.

   Guidelines:

   - The answer [N/A] means that the paper does not include experiments.
   - The experimental setting should be presented in the core of the paper to a level of detail that is necessary to appreciate the results and make sense of them.
   - The full details can be provided either with the code, in appendix, or as supplemental material.

7. **Experiment statistical significance**

   Question: Does the paper report error bars suitably and correctly defined or other appropriate information about the statistical significance of the experiments?

   Answer: [No]

   Justification: We do not report error bars over random seeds, as Pico is a deterministic post-hoc calibration and randomness only enters through LoRA adapter training. We instead demonstrate robustness of the main claim by showing consistent improvements across three downstream mergers, four LoRA ranks, two backbones, and a progressive-merge protocol (Tables 1, 13, 14, Figure 6).

   Guidelines:

   - The answer [N/A] means that the paper does not include experiments.
   - The authors should answer [Yes] if the results are accompanied by error bars, confidence intervals, or statistical significance tests, at least for the experiments that support the main claims of the paper.

- The factors of variability that the error bars are capturing should be clearly stated (for example, train/test split, initialization, random drawing of some parameter, or overall run with given experimental conditions).
- The method for calculating the error bars should be explained (closed form formula, call to a library function, bootstrap, etc.)
- The assumptions made should be given (e.g., Normally distributed errors).
- It should be clear whether the error bar is the standard deviation or the standard error of the mean.
- It is OK to report 1-sigma error bars, but one should state it. The authors should preferably report a 2-sigma error bar than state that they have a 96% CI, if the hypothesis of Normality of errors is not verified.
- For asymmetric distributions, the authors should be careful not to show in tables or figures symmetric error bars that would yield results that are out of range (e.g., negative error rates).
- If error bars are reported in tables or plots, the authors should explain in the text how they were calculated and reference the corresponding figures or tables in the text.

8. **Experiments compute resources**

   Question: For each experiment, does the paper provide sufficient information on the computer resources (type of compute workers, memory, time of execution) needed to reproduce the experiments?

   Answer: [Yes]

   Justification: Appendix B reports the training hardware (4×H800 GPUs), and Pico itself only needs one SVD plus a left multiplication per layer at merge time (Section 4.1).

   Guidelines:

   - The answer [N/A] means that the paper does not include experiments.
   - The paper should indicate the type of compute workers CPU or GPU, internal cluster, or cloud provider, including relevant memory and storage.
   - The paper should provide the amount of compute required for each of the individual experimental runs as well as estimate the total compute.
   - The paper should disclose whether the full research project required more compute than the experiments reported in the paper (e.g., preliminary or failed experiments that didn't make it into the paper).

9. **Code of ethics**

   Question: Does the research conducted in the paper conform, in every respect, with the NeurIPS Code of Ethics `https://neurips.cc/public/EthicsGuidelines`?

   Answer: [Yes]

   Justification: The research uses only publicly released models and datasets, does not involve human-subject data collection, and preserves anonymity in this submission.

   Guidelines:

   - The answer [N/A] means that the authors have not reviewed the NeurIPS Code of Ethics.
   - If the authors answer [No], they should explain the special circumstances that require a deviation from the Code of Ethics.
   - The authors should make sure to preserve anonymity (e.g., if there is a special consideration due to laws or regulations in their jurisdiction).

10. **Broader impacts**

    Question: Does the paper discuss both potential positive societal impacts and negative societal impacts of the work performed?

    Answer: [N/A]

    Justification: No relevant societal impacts unique to this paper (based on existing public LLMs).

Guidelines:

- The answer [N/A] means that there is no societal impact of the work performed.
- If the authors answer [N/A] or [No], they should explain why their work has no societal impact or why the paper does not address societal impact.
- Examples of negative societal impacts include potential malicious or unintended uses (e.g., disinformation, generating fake profiles, surveillance), fairness considerations (e.g., deployment of technologies that could make decisions that unfairly impact specific groups), privacy considerations, and security considerations.
- The conference expects that many papers will be foundational research and not tied to particular applications, let alone deployments. However, if there is a direct path to any negative applications, the authors should point it out. For example, it is legitimate to point out that an improvement in the quality of generative models could be used to generate Deepfakes for disinformation. On the other hand, it is not needed to point out that a generic algorithm for optimizing neural networks could enable people to train models that generate Deepfakes faster.
- The authors should consider possible harms that could arise when the technology is being used as intended and functioning correctly, harms that could arise when the technology is being used as intended but gives incorrect results, and harms following from (intentional or unintentional) misuse of the technology.
- If there are negative societal impacts, the authors could also discuss possible mitigation strategies (e.g., gated release of models, providing defenses in addition to attacks, mechanisms for monitoring misuse, mechanisms to monitor how a system learns from feedback over time, improving the efficiency and accessibility of ML).

11. **Safeguards**

    Question: Does the paper describe safeguards that have been put in place for responsible release of data or models that have a high risk for misuse (e.g., pre-trained language models, image generators, or scraped datasets)?

    Answer: [N/A]

    Justification: The paper releases only a post-hoc calibration method for already-trained LoRA adapters, with no new models, scraped data, or generative systems that would require safeguards.

    Guidelines:

    - The answer [N/A] means that the paper poses no such risks.
    - Released models that have a high risk for misuse or dual-use should be released with necessary safeguards to allow for controlled use of the model, for example by requiring that users adhere to usage guidelines or restrictions to access the model or implementing safety filters.
    - Datasets that have been scraped from the Internet could pose safety risks. The authors should describe how they avoided releasing unsafe images.
    - We recognize that providing effective safeguards is challenging, and many papers do not require this, but we encourage authors to take this into account and make a best faith effort.

12. **Licenses for existing assets**

    Question: Are the creators or original owners of assets (e.g., code, data, models), used in the paper, properly credited and are the license and terms of use explicitly mentioned and properly respected?

    Answer: [Yes]

    Justification: All used base models, datasets, benchmarks, and baseline methods are cited in Section 5.1 and Appendix B, and are used in accordance with their public licenses.

    Guidelines:

    - The answer [N/A] means that the paper does not use existing assets.
    - The authors should cite the original paper that produced the code package or dataset.

- The authors should state which version of the asset is used and, if possible, include a URL.
- The name of the license (e.g., CC-BY 4.0) should be included for each asset.
- For scraped data from a particular source (e.g., website), the copyright and terms of service of that source should be provided.
- If assets are released, the license, copyright information, and terms of use in the package should be provided. For popular datasets, `paperswithcode.com/datasets` has curated licenses for some datasets. Their licensing guide can help determine the license of a dataset.
- For existing datasets that are re-packaged, both the original license and the license of the derived asset (if it has changed) should be provided.
- If this information is not available online, the authors are encouraged to reach out to the asset's creators.

13. **New assets**

    Question: Are new assets introduced in the paper well documented and is the documentation provided alongside the assets?

    Answer: [N/A]

    Justification: No new datasets or pre-trained models are released at submission time.

    Guidelines:

    - The answer [N/A] means that the paper does not release new assets.
    - Researchers should communicate the details of the dataset/code/model as part of their submissions via structured templates. This includes details about training, license, limitations, etc.
    - The paper should discuss whether and how consent was obtained from people whose asset is used.
    - At submission time, remember to anonymize your assets (if applicable). You can either create an anonymized URL or include an anonymized zip file.

14. **Crowdsourcing and research with human subjects**

    Question: For crowdsourcing experiments and research with human subjects, does the paper include the full text of instructions given to participants and screenshots, if applicable, as well as details about compensation (if any)?

    Answer: [N/A]

    Justification: The paper does not involve crowdsourcing or research with human subjects.

    Guidelines:

    - The answer [N/A] means that the paper does not involve crowdsourcing nor research with human subjects.
    - Including this information in the supplemental material is fine, but if the main contribution of the paper involves human subjects, then as much detail as possible should be included in the main paper.
    - According to the NeurIPS Code of Ethics, workers involved in data collection, curation, or other labor should be paid at least the minimum wage in the country of the data collector.

15. **Institutional review board (IRB) approvals or equivalent for research with human subjects**

    Question: Does the paper describe potential risks incurred by study participants, whether such risks were disclosed to the subjects, and whether Institutional Review Board (IRB) approvals (or an equivalent approval/review based on the requirements of your country or institution) were obtained?

    Answer: [N/A]

    Justification: The paper does not involve human-subjects research, so IRB approval is not applicable.

    Guidelines:

- The answer [N/A] means that the paper does not involve crowdsourcing nor research with human subjects.
- Depending on the country in which research is conducted, IRB approval (or equivalent) may be required for any human subjects research. If you obtained IRB approval, you should clearly state this in the paper.
- We recognize that the procedures for this may vary significantly between institutions and locations, and we expect authors to adhere to the NeurIPS Code of Ethics and the guidelines for their institution.
- For initial submissions, do not include any information that would break anonymity (if applicable), such as the institution conducting the review.

16. **Declaration of LLM usage**

    Question: Does the paper describe the usage of LLMs if it is an important, original, or non-standard component of the core methods in this research? Note that if the LLM is used only for writing, editing, or formatting purposes and does *not* impact the core methodology, scientific rigor, or originality of the research, declaration is not required.

    Answer: [N/A]

    Justification: LLMs are used only as standard base models for LoRA fine-tuning.

    Guidelines:

    - The answer [N/A] means that the core method development in this research does not involve LLMs as any important, original, or non-standard components.
    - Please refer to our LLM policy in the NeurIPS handbook for what should or should not be described.