arXiv:2605.22869v1 [cs.LG] 19 May 2026

FuRA: Full-Rank Parameter-Efficient Fine-Tuning with Spectral Preconditioning

Yequan Zhao[1] **Ruijie Zhang**[1] **Liyan Tan**[1] **Niall Moran**[2] **Tong Qin**[2] **Zheng Zhang**[1]
[1]University of California at Santa Barbara [2]Amazon Lab126
{yequan_zhao, ruijiezhang, liyan_tan}@ucsb.edu
{nialmora, tqinmath}@amazon.com
zhengzhang@ece.ucsb.edu

Abstract

Both full fine-tuning (Full FT) and parameter-efficient methods like LoRA add weight updates without regard to the spectral structure that pretraining has established. This allows noisy gradients from a small fine-tuning distribution to freely perturb the robust features learned through pretraining. We first identify *spectral preconditioning* as the key missing ingredient: reparameterizing each weight $\mathbf{W}$ through its full-rank SVD and freezing one singular basis confines every update to the pretrained column space, yielding a preconditioned optimizer that outperforms unconstrained Full FT at the same parameter count. To make this insight practical, we propose FuRA (**Fu**ll-**R**ank **A**daptation), which factorizes $\mathbf{W}$ via a block tensor-train decomposition $\mathbf{W} = \mathbf{L}\,\mathbf{S}\,\mathbf{R}$: the large core $\mathbf{L}$ is frozen at the pretrained block-wise SVD basis while only the small core $\mathbf{R}$ and per-block singular values $\mathbf{S}$ are trained. This single design choice simultaneously delivers full-rank spectral preconditioning, full-rank update capacity, and parameter, step time, memory efficiency on par with LoRA. FuRA outperforms Full FT on LLM fine-tuning ($+1.37$ on LLaMA-3-8B commonsense reasoning), LLM math reinforcement learning, and VLM visual instruction tuning. The 4-bit quantized version QFuRA also outperforms QLoRA. Code is available at `https://github.com/olokevin/FuRA-NIPS`.

1 Introduction

Large language models [10, 36] (LLM) acquire rich, transferable representations during pretraining on web-scale corpora. Fine-tuning these models on a small, task-specific dataset, whether through supervised finetuning (SFT) or reinforcement learning with verifiable rewards (RLVR) [40, 52], can unlock strong downstream performance. Full fine-tuning (Full FT) updates every parameter and remains the performance ceiling on most benchmarks, but incurs prohibitive memory and compute costs. Parameter-efficient methods such as LoRA [14] circumvent this cost by constraining the update $\Delta\mathbf{W}$ to a low-rank subspace, yet a persistent accuracy gap relative to Full FT [39] motivates continued research into more expressive alternatives including [25, 31, 55, 28].

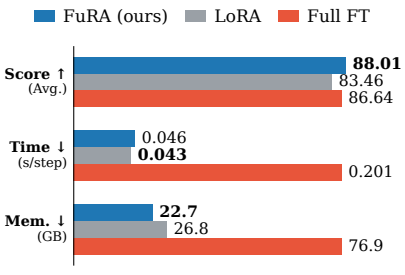

Figure 1: FuRA delivers higher accuracy than Full FT with LoRA-level runtime and the lowest GPU memory.

We argue that both Full FT and LoRA share a deeper limitation: **neither respects the spectral structure of the pretrained weight.** The update is unconstrained in direction and can push the model along axes orthogonal to the feature manifold that pretraining established. Specifically, the update

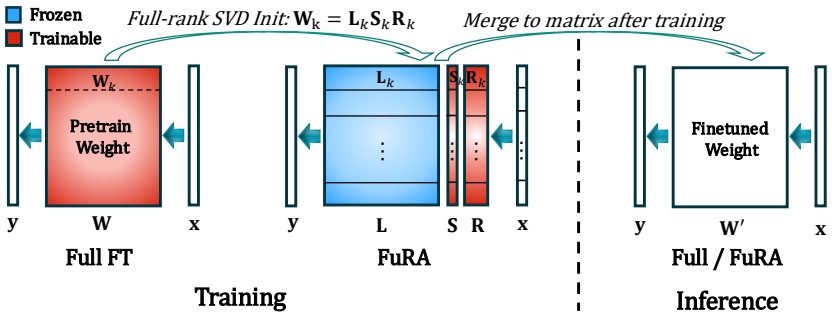

Figure 2: FuRA replaces each pretrained linear layer $\mathbf{W}$ with a lossless block tensor-train factorization $\mathbf{W} = \mathbf{L}\,\mathbf{S}\,\mathbf{R}$. Tensors are flattened to matrix for better demonstration. Each slice is initialized by the full-rank SVD of weight block $\mathbf{W}_k$, and performs spectral preconditioned update.

$\Delta\mathbf{W}$, whether computed directly from gradients (Full FT) or accumulated in a low-rank adapter $\mathbf{BA}$ (LoRA), is added to $\mathbf{W}$ without regard to its learned spectral geometry. Because the fine-tuning data is orders of magnitude smaller and narrower in distribution, its gradients are noisier estimates of the true task-relevant directions. Allowing these noisy updates to freely perturb robust pretrained features risks degrading generalization, a failure mode closely related to catastrophic forgetting [19]. This observation suggests a different design principle: rather than adding an unconstrained perturbation to $\mathbf{W}$, fine-tuning should **re-align the pretrained features** in a way that is guided by, and confined to, the subspace the model has already learned. If the pretrained weight encodes a coordinate system for useful representations, adaptation should re-orient within that coordinate system rather than demolish and rebuild it.

Through a controlled study of SVD FT (§3.2), we further validate that confining weight updates to the pretrained column space can produce superior performance in LLM fine tuning. However, directly doing so can introduce a large number of training variables like full FT. This motivates us to develop FuRA (**Fu**ll-**R**ank **A**daptation, Figure 2), a parameter-efficient method that simultaneously delivers 1) full-rank update capacity, 2) high accuracy through full-rank spectral preconditioned update, and 3) LoRA-level parameter efficiency.

FuRA outperforms other PEFT methods and Full FT without sacrificing inference efficiency: $+2.87$ over DoRA [25] and $+1.37$ over Full FT on LLaMA-3-8B Commonsense SFT, outperforms Full FT on Qwen-1.7B/7B math RLVR, $+1.1$ over Full FT on VLM visual instruction tuning, all with similar memory and wall-clock step time of LoRA. The 4-bit quantized version QFuRA also surpasses QLoRA [7] by $+2.51$ on LLaMA-3-70B math fine-tuning.

Contributions. Our research contributions are summarized below.

- We study the spectral properties of LLM fine-tuning and identify spectral preconditioning as a key missing ingredient. We show that both full fine-tuning and LoRA ignore the pretrained spectral structure, while controlled SVD-based fine-tuning demonstrates that aligning updates with this structure yields better performance than full fine-tuning.

- Based on these observations, we propose FuRA, a block tensor-train factorization that makes spectral preconditioning a practical PEFT method. A single architectural choice simultaneously achieves full-rank spectral preconditioning, full-rank update capacity, and LoRA-level parameter and computing efficiency.

- We prove two key properties of FuRA: although it trains only a small fraction of parameters, it can realize full-rank updates; and its effective update acts as a column-space projection combined with singular-value preconditioning.

- We demonstrate that FuRA outperforms Full FT on both SFT and RLVR tasks with only $< 2\%$ trainable parameters and no task-specific rank tuning, establishing a new state of the art for parameter-efficient LLM adaptation.

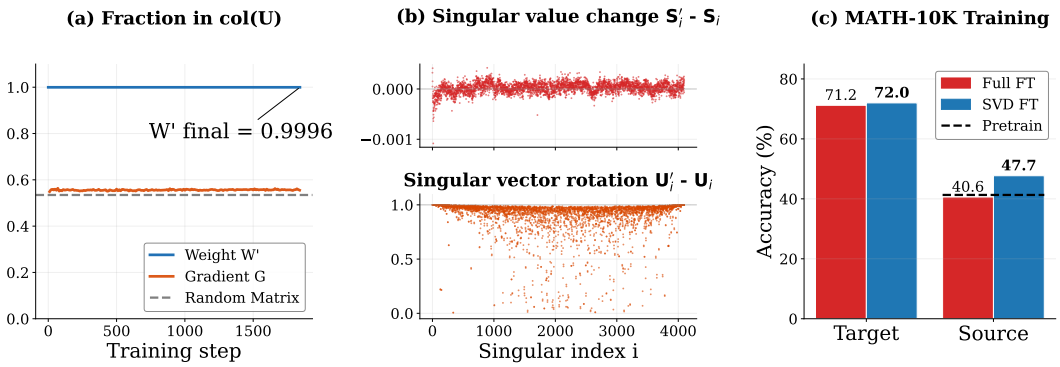

Figure 3: **(a)** Gradient lives outside singular basis of pretrained weight, but weight keep stable. **(b)** Singular values shift selectively, while singular vectors rotate broadly. (a) (b) demonstrates layer 15 q_proj result, the full sweeps are in Figures 6 and 7 (Appendix). **(c)** SVD FT outperforms Full FT on both target domain and source domain.

2 Background and Related Work

PEFT methods for LLMs. LoRA [14] constrains the weight update to $\Delta\mathbf{W} = \mathbf{B}\mathbf{A}^\top$ with rank $r \ll \min(d_{\text{in}}, d_{\text{out}})$, training only $\mathcal{O}(r(d_{\text{in}} + d_{\text{out}}))$ parameters per layer. This low-rank constraint primarily serves parameter efficiency rather than reflecting task-specific inductive bias. Subsequent work improves LoRA along several directions: QLoRA [7] adds weight quantization, DoRA [25] decouples magnitude and direction, AdaLoRA [55] adapts rank across layers, and VeRA [20] shares fixed random bases while training only diagonal scales. Tensor-train decomposition [34, 33, 35] offer another route; adapters such as LoRETTA [49] and LoRTA [15] train all tensor cores but leave $\Delta\mathbf{W}$ low-rank. Despite these advances, a performance gap with Full FT remains.

PEFT with higher or full rank capacity. A natural remedy is to lift the rank cap. MoRA [18] reshapes the adapter into a square matrix to allow higher-rank updates; RandLoRA [1] uses fixed random bases with trainable scales; BOFT [26] applies a butterfly-factored orthogonal transform, and QuanTA [3], FourierFT [8], etc. S^2FT [48] and LIFT [28] train structured- or unstructured-sparse subsets of weights, achieving full-rank updates but requiring gather/scatter operations that are less tensor-parallel friendly. These methods achieve nominally full-rank $\Delta\mathbf{W}$ at the LoRA budget, but have not consistently closed the gap to Full FT on standard benchmarks. We argue that rank alone is insufficient; the *subspace* in which the update operates is equally important.

Spectrum-informed adapters. A related line of work leverages the pretrained weight's SVD to guide adaptation. PiSSA [31] initializes LoRA from top-r singular directions, MiLoRA [47] uses bottom-r, Spectral Adapter [54] parameterizes updates in the spectral basis, and SVFT [23] trains a dense middle matrix between frozen SVD factors. While these approaches incorporate spectral information, they face two limitations. First, the low-rank constraint forces a choice of spectral slice, yet the optimal subspace is task-dependent: SFT emphasizes principal directions, whereas RLVR favors off-principal ones [56]. Second, SVD-based initialization is only a soft constraint, allowing updates to drift away from the pretrained subspace during training [50].

3 Spectral Preconditioning Improves LLM Fine-tuning

Throughout this paper, $\mathbf{W} \in \mathbb{R}^{d_{\text{out}} \times d_{\text{in}}}$ denotes a pretrained linear layer with forward pass $\mathbf{y} = \mathbf{W}\mathbf{x}$. And we use $\mathbf{W}'$ to denote the fine-tuned weight.

3.1 Observations from Full FT Training Dynamics

Both Full FT and existing PEFT methods update weights without considering the spectral structure learned during pretraining, risking disruption of the pretrained feature geometry. We hypothesize that fine-tuning should instead preserve and exploit this structure. To investigate this, we analyze the training dynamics of Full FT on LLaMA-3-8B fine-tuned on Math-10K [16] (Figure 3).

Gradients are spectrally diffuse, while weight spectrum remain stable. Let $\mathbf{W} = \mathbf{U}\mathbf{\Sigma}\mathbf{V}^\top$ denote the SVD of a pretrained weight. We quantify how much a matrix $\mathbf{M}$ lies in the pretrained column space $\mathrm{col}(\mathbf{U})$ using the *column-space energy ratio*:

$$\rho(\mathbf{M};\, \mathbf{U}) \;=\; \frac{\|\mathbf{U}^\top \mathbf{M}\|_F^2}{\|\mathbf{M}\|_F^2}\,. \tag{1}$$

We have $\rho \in [0, 1]$ and $\rho = 1$ indicates full alignment with $\mathrm{span}(\mathbf{U})$. Gaussian random matrices has expected energy ratio $\mathrm{rank}(\mathbf{U})/d_{\text{out}}$ (see Appendix E.2). Tracking the energy ratio of the gradient $\mathbf{G}_\mathbf{W}$ during Full FT (Figure 3a), we observe that $\rho(\mathbf{G}_\mathbf{W}, \mathbf{U})$ stays near the random baseline, showing gradients are updating all directions and are not particularly aligned with pretrained directions. In contrast, $\rho(\mathbf{W}', \mathbf{U}) \approx 1$ throughout training, indicating that the Full FT implicitly preserves the weight's spectral structure despite diffusive gradient updates.

Singular values shift selectively; singular vectors rotate broadly. Comparing the SVDs of $\mathbf{W}'$ and $\mathbf{W}$ (Figure 3b), we find that only a few singular values change significantly, while most remain close to their pretrained values. In contrast, the singular vectors exhibit substantial rotation: cosine similarities between columns of $\mathbf{U}'$ and $\mathbf{U}$ vary widely, with no clear bias toward principal or off-principal directions. The above findings suggest that Full FT, despite being unconstrained, implicitly preserves the pretrained column space, concentrating spectral changes on a few singular values while allowing broad directional rotation. This raises a natural question: *what if we explicitly enforce these patterns as architectural constraints?*

3.2 Insights from SVD Fine-Tuning Experiment

We study a controlled SVD Fine-tuning (SVD FT) to validate the above hypothesis. We decompose each pretrained weight via SVD, $\mathbf{W} = \mathbf{U}\mathbf{\Sigma}\mathbf{V}^\top$, freeze the left-singular basis $\mathbf{U}$, and fine-tune only $\mathbf{\Sigma}$ and $\mathbf{V}^\top$. This SVD FT has identical parameter count and initialization to full FT, differing only in parameterization. It induces a *spectral preconditioning*, where the effective update from $\mathbf{V}^\top$ is

$$\Delta\mathbf{W}\big|_\mathbf{V} \;=\; -\eta\, \mathbf{U}\mathbf{\Sigma}^2\mathbf{U}^\top\, \mathbf{G}_\mathbf{W}\,, \tag{2}$$

with $\mathbf{G}_\mathbf{W} = \partial\mathcal{L}/\partial\mathbf{W}$. The projection $\mathbf{U}\mathbf{U}^\top$ restricts updates to $\mathrm{col}(\mathbf{U})$, formalizing the column-space preservation observed earlier, while $\mathbf{\Sigma}$ separates magnitude from direction, aligning with the distinct spectral dynamics.

We evaluate LLaMA-3-8B trained with Full FT and SVD FT on Math-10K, measuring both target domain math reasoning (GSM8K [6]) and source domain commonsense reasoning performance (Figure 3c). We observe that: (1) SVD FT improves target-domain performance, indicating that gradients within $\mathrm{col}(\mathbf{U})$ suffice for effective learning; (2) SVD FT improves source-domain generalization, whereas Full FT degrades relative to the pretrained model.

This experiment suggests that **spectral preconditioning better preserves pretrained structure**, mitigating forgetting and promoting generalizable representations.

4 The FuRA Framework

SVD FT demonstrates the benefits of spectral preconditioning but is impractical: it retains a full-size trainable factor and doubles forward-pass cost. We propose FuRA, which addresses both issues using a Block Tensor-Train (BTT) parameterization.

FuRA reparameterizes each linear layer with a 2-core Block Tensor-Train (BTT) decomposition [35], which can be interpreted as block-wise SVD. The full-rank 2-core BTT can be used to represent arbitrary matrix. By carefully selecting a special block size, we can perform spectral preconditioned update as for the SVD FT, while with a cost on par with LoRA.

Block tensor-train (BTT) decomposition. We focus on 2-core decomposition case. Given $\mathbf{W} \in \mathbb{R}^{d_{\text{out}} \times d_{\text{in}}}$, choose an $m \times n$ grid such that $d_{\text{out}} = ma$ and $d_{\text{in}} = nb$. Each block $\mathbf{W}_{ij} \in \mathbb{R}^{a\times b}$ is factorized into $\mathbf{L}_{ij} \in \mathbb{R}^{a\times r}$ and $\mathbf{R}_{ij} \in \mathbb{R}^{r\times b}$, forming cores $\boldsymbol{\mathcal{L}} \in \mathbb{R}^{m\times n\times a\times r}$ and $\boldsymbol{\mathcal{R}} \in \mathbb{R}^{m\times n\times r\times b}$. The matrix-vector product $\mathbf{y} = \mathbf{W}\mathbf{x}$ becomes

$$y_{\alpha\beta} = \sum_{\gamma\sigma} \boldsymbol{\mathcal{L}}_{\alpha\beta\gamma\sigma} \sum_{\delta} \boldsymbol{\mathcal{R}}_{\sigma\beta\gamma\delta}\, \mathbf{x}_{\gamma\delta}. \tag{3}$$

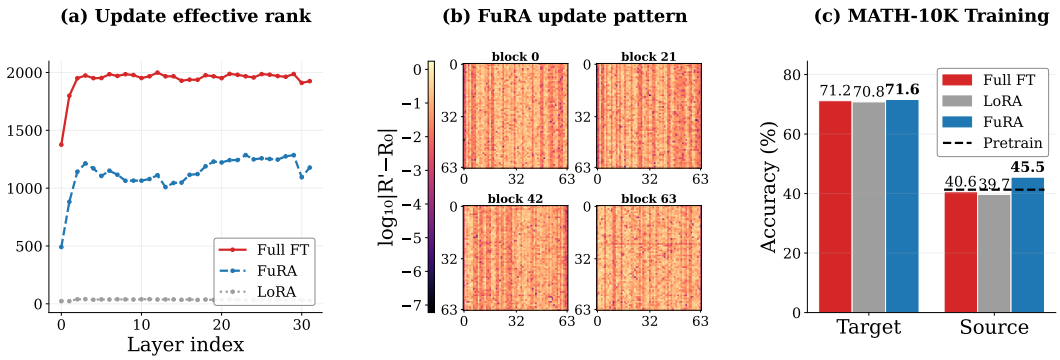

Figure 4: **(a)** Effective rank of $\Delta\mathbf{W}$; full sweep is in Figure 8 (Appendix). **(b)** Singular-direction heatmaps show sharper active/inactive separation, reflecting spectral preconditioning from frozen $\mathbf{L}$. **(c)** FuRA better preserves pretrained capabilities.

Table 1: System overhead comparison. Avg is the average accuracy on LLaMA-3-8B commonsense SFT Detail setups and analysis are in Appendix B.1.

Method	Train. (%)	Extra (%)	Full-rank $\Delta\mathbf{W}$?	Step (s)	Memory (GB)	Avg (%)
Full FT	100.00	0	Yes	0.201	76.9	86.64
LoRA (r=64)	1.39	1.39	No	**0.043**	26.8	83.46
DoRA (r=64)	1.42	1.42	No	0.045	26.8	85.04
RandLoRA (r=64)	0.33	2.57	No (random)	0.059	25.7	54.96
LIFT (r=32)	6.23	0	Yes (sparse)	0.144	56.0	87.88
FuRA (ours)	1.46	1.46	**Yes (dense)**	0.046	**22.7**	**88.01**

At full rank $r = \min(a, b)$, this BTT is applying full-rank SVD for each individual block and hence the representation is equivalent to the original dense matrix. For a square, full-rank decomposition ($d_{\text{out}} = d_{\text{in}} = d$, $m = n = a = b = r = \sqrt{d}$), the total parameter count and MVM FLOPs are both $2d^2$ [35], same as SVD.

The FuRA parameterization. We decompose each pretrained weight using a parameter-efficient BTT configuration and keep the diagonal singular values as separate trainable parameters. Setting $m = 1$ yields blocks with a *large* core $\mathbf{L}_k \in \mathbb{R}^{d_{\text{out}} \times r}$ and a *small* core $\mathbf{R}_k \in \mathbb{R}^{r \times b}$ with $b \ll d_{\text{in}}$. At full rank $r = b$, this decomposition is lossless while only introduces $d_{\text{in}}b$ additional parameters. For each block, full-rank SVD yields $\mathbf{W}_k = \mathbf{U}_k\mathbf{\Sigma}_k\mathbf{V}_k^\top$, and we set $\mathbf{L}_k = \mathbf{U}_k \in \mathbb{R}^{d_{\text{out}} \times b}$, $\mathbf{S}_k = \text{diag}(\mathbf{\Sigma}_k) \in \mathbb{R}^b$, $\mathbf{R}_k = \mathbf{V}_k^\top \in \mathbb{R}^{b \times b}$. Stacking blocks, we got BTT cores $\mathbf{L} = [\mathbf{L}_1, \ldots, \mathbf{L}_n]$, $\mathbf{S} = [\mathbf{S}_1, \ldots, \mathbf{S}_n]$, $\mathbf{R} = [\mathbf{R}_1, \ldots, \mathbf{R}_n]$. FuRA **freezes** $\mathbf{L}$ and trains only $\mathbf{S}$ and $\mathbf{R}$. The fine-tuned weight is assembled column-block-wise:

$$\mathbf{W}'_k = \mathbf{L}_k \,\text{diag}(\mathbf{S}'_k)\,\mathbf{R}'_k, \qquad \mathbf{W}' = \left[\,\mathbf{W}'_1 \,\middle|\, \mathbf{W}'_2 \,\middle|\, \cdots \,\middle|\, \mathbf{W}'_n\,\right], \tag{4}$$

where $[\cdot|\cdot]$ denotes horizontal (column-block) concatenation. Algorithm 1 in Appendix summarizes the procedure.

Complexity analysis. The block-SVD initialization is a **one-time** cost (e.g. $\sim$ 1 minute for 8B model). During training, FuRA matches LoRA in both memory and step time (Table 1). The trainable count is $|\mathbf{R}| + |\mathbf{S}| = nrb + nr \approx d^{3/2}$, roughly 2% of pretrained weight size at $d = 4096$; the $\mathbf{L}$-stage reduces to a single dense MVM and the $\mathbf{R}$-stage to a batched BMM, both well supported in deep learning frameworks; after training, the cores merge back into a single dense matrix so there is no additional inference overhead. We defer the full breakdown to Appendix B.

Advantages of FuRA. This single design choice delivers the following properties

- ***Full-rank capacity.*** Each block's update lies in $\text{col}(\mathbf{L}_k)$, and across blocks these subspaces span the full output space, leaving $\Delta\mathbf{W}$ unconstrained in rank. The optimizer, rather than a preset rank, controls layer capacity. As shown in Fig. 4 (a), the effective rank of FuRA update is significantly higher than LoRA. The lower effective rank compared to Full FT suggests that the spectral preconditioner focuses capacity on task-relevant directions rather than distributing it

uniformly. Importantly, the full-rank block-wise singular basis is preserved, avoiding commitment to specific spectral subspaces. We formalize this in Section 5, Claim 1.

- ***Spectral preconditioning.*** The spectral preconditioning from SVD FT (Section 3.2, Eq. (2)) extends naturally to FuRA's block structure. The trainable $\mathbf{S}$ decouples magnitude and direction, enabling finer spectral control, similar in spirit to DoRA [25]. As shown in Figure 4 (b), the optimizer selectively updates important singular directions while preserving others. The full preconditioner is derived in Section 5, Claim 2.
- ***Better generalization.*** FuRA outperforms Full FT and LoRA on target-domain math reasoning using only $\sim 2\%$ trainable parameters, while better preserving (improving) source-domain commonsense reasoning accuracy [Fig. 4 (c)]. By restricting updates to the pretrained column space, it favors reweighting existing features over introducing arbitrary directions that may overfit to the fine-tuning distribution.

Remark 4.1 (Design space). The FuRA design has two axes: **(1)** preserve the output subspace ($m = 1$, the default) or input subspace ($n = 1$); and **(2)** keep $\mathbf{S}$ as a separate trainable vector (the default) or merged into the trainable / frozen core. We characterize each choice's effective preconditioner in Section 5 (Claim 2, Table 2) and ablate them in Section 6.5.

5 Theoretical Insights

In this section, we present two theoretical claims regarding the theoretical properties of FuRA. We support each claim with a short derivation and provide the full proofs to Appendix E.2.

Claim 1 (Full-rank update capacity). Although FuRA trains only $\mathbf{R}$ and $\mathbf{S}$, the update $\Delta\mathbf{W}$ has full-rank capacity: it can reach any rank up to $\min(d_{\text{in}}, d_{\text{out}})$ whenever the pretrained weight $\mathbf{W}$ is full rank, in contrast to LoRA-style adapters that hard-cap $\operatorname{rank}(\Delta\mathbf{W}) \leq r$.

Proposition 5.1 (No explicit rank cap)**.** *Under the* FuRA *parameterization in Eq.* (4)*, with* $\mathbf{L}$ *frozen and* $\mathbf{S}, \mathbf{R}$ *trainable, the update* $\Delta\mathbf{W} = \mathbf{W}' - \mathbf{W}$ *is not rank-constrained (Lemma E.2).*

Sketch of Proof. Per block, $\Delta\mathbf{W}_k$ lies in $\operatorname{col}(\mathbf{L}_k)$ (Propositions E.3 and E.4, appendix); across the n blocks, these subspaces collectively span $\operatorname{col}(\mathbf{W})$.

Remark 5.2. While Claim 1 establishes full-rank reachability, the rank measure alone does not capture the geometry of the reachable updates. Two further structural properties follow from the per-block construction (Appendix E.2): (i) the default $m{=}1$ orientation processes the input as n disjoint block slices, with no cross-block information transfer at the BTT layer; and (ii) each per-block update is confined to $\operatorname{col}(\mathbf{L}_k)$, so $\Delta\mathbf{W}$ lies in a product of pretrained subspaces. The cross-block locality (i) is the primary expressivity cost, and could be relaxed at extra parameters by prepending an $n{\times}n$ input-mixing matrix before the BTT contraction; we leave this extension to future work.

Claim 2 (Spectral preconditioning). The effective update under FuRA is not standard SGD on $\mathbf{W}$: it applies *column-space projection* and *singular-value preconditioning*, both induced by the frozen block-wise SVD basis. We provide the explanations below.

Consider a single block with SVD $\mathbf{W}_k = \mathbf{U}_k\mathbf{\Sigma}_k\mathbf{V}_k^\top$ and the k-th block input $\mathbf{x}_k$. Let $\mathbf{g} = \partial\mathcal{L}/\partial\mathbf{y}$ and $\mathbf{G}_k = \mathbf{g}\mathbf{x}_k^\top$ denote the unconstrained gradient. Under the default FuRA setting (train $\mathbf{R}$ and $\mathbf{S}$, freeze $\mathbf{L} = \mathbf{U}$), the forward pass is $\mathbf{y}_k = \mathbf{U}_k \operatorname{diag}(\mathbf{S}_k)\,\mathbf{R}_k\,\mathbf{x}_k$. A single gradient step yields

$$\Delta\mathbf{W}_k \;=\; \underbrace{-\eta\,\mathbf{U}_k \operatorname{diag}(\mathbf{S}_k)^2\,\mathbf{U}_k^\top\,\mathbf{G}_k}_{\text{from }\mathbf{R}\text{ update}} \;+\; \underbrace{\mathbf{U}_k \operatorname{diag}(\Delta\mathbf{S}_k)\,\mathbf{V}_k^\top}_{\text{from }\mathbf{S}\text{ update}}, \tag{5}$$

where $\operatorname{diag}(\mathbf{S}_k)^2$ arises because $\operatorname{diag}(\mathbf{S}_k)$ appears twice in the chain rule: in $\partial\mathcal{L}/\partial\mathbf{R}_k$ and in $\mathbf{U}_k \operatorname{diag}(\mathbf{S}_k)\,\delta\mathbf{R}_k$. The preconditioner $\mathbf{U}_k \operatorname{diag}(\mathbf{S}_k)^2\,\mathbf{U}_k^\top$ in the first term combines two effects:

- **Column-space projection.** Since $\mathbf{U}_k$ has orthonormal columns ($\mathbf{U}_k^\top\mathbf{U}_k = \mathbf{I}$, but $\mathbf{U}_k\mathbf{U}_k^\top \neq \mathbf{I}$ for rectangular blocks), the update $\Delta\mathbf{W}_k$ is restricted to $\operatorname{col}(\mathbf{U}_k)$, the pretrained left-singular subspace. In FuRA, $\mathbf{U}_k$ is always rectangular, making this projection non-trivial.
- **Singular-value preconditioning.** Within $\operatorname{col}(\mathbf{U}_k)$, gradients are scaled by σ_i^2 along each singular direction, amplifying high-signal components and suppressing low-signal ones. This induces an adaptive, spectrum-dependent learning rate.

Table 2: Effective gradient-induced $\Delta\mathbf{W}_k$ per design corner of FuRA (up to the learning rate $-\eta$). Principal-biased (PiSSA-like) and off-principal-biased (MiLoRA-like) updates arise as two corners of the same framework. The **default** configuration used in the main experiments is the last row.

Trainable side	$\mathbf{S}$ placement	Effective $\Delta\mathbf{W}_k$
Train $\mathbf{L}$ (output), freeze $\mathbf{R} = \mathbf{V}_k^\top$	$\mathbf{S}$ in trainable $\mathbf{L}$	$\mathbf{g}\mathbf{x}_k^\top\, \mathbf{V}_k\mathbf{V}_k^\top$
Train $\mathbf{L}$ (output), freeze $\mathbf{R} = \mathrm{diag}(\mathbf{S}_k)\mathbf{V}_k^\top$	$\mathbf{S}$ in frozen $\mathbf{R}$	$\mathbf{g}\mathbf{x}_k^\top\, \mathbf{V}_k\,\mathrm{diag}(\mathbf{S}_k)^2\,\mathbf{V}_k^\top$
Train $\mathbf{R}$ (input), freeze $\mathbf{L} = \mathbf{U}_k$	$\mathbf{S}$ in trainable $\mathbf{R}$	$\mathbf{U}_k\mathbf{U}_k^\top\,\mathbf{g}\mathbf{x}_k^\top$
Train $\mathbf{R}$ (input), freeze $\mathbf{L} = \mathbf{U}_k\,\mathrm{diag}(\mathbf{S}_k)$	$\mathbf{S}$ in frozen $\mathbf{L}$	$\mathbf{U}_k\,\mathrm{diag}(\mathbf{S}_k)^2\,\mathbf{U}_k^\top\,\mathbf{g}\mathbf{x}_k^\top$
Train R, keep S separate trainable (default)	separate, trainable	$\mathbf{U}_k\,\mathrm{diag}(\mathbf{S}_k)^2\,\mathbf{U}_k^\top\,\mathbf{g}\mathbf{x}_k^\top + \mathbf{U}_k\,\mathrm{diag}(\Delta\mathbf{S}_k)\,\mathbf{V}_k^\top$

Design choices. FuRA exposes two design axes; Table 2 enumerates all corners.

(1) Subspace preservation. Choosing orientation $m{=}1$ with $\mathbf{L}$ frozen (the default) preserves the per-block pretrained *output* subspace: every block update satisfies $\mathrm{col}(\Delta\mathbf{W}_k) \subseteq \mathrm{col}(\mathbf{U}_k)$. The symmetric choice $n{=}1$ with $\mathbf{R}$ frozen instead preserves the per-block pretrained *input* subspace, $\mathrm{row}(\Delta\mathbf{W}_k) \subseteq \mathrm{row}(\mathbf{V}_k^\top)$. Both orientations share the same trainable parameter count and full-rank update capacity (Claim 1), but bias the optimizer toward different pretrained directions.

(2) Placement of S. Keeping $\mathbf{S}$ as a separate trainable vector (default) yields the $\mathrm{diag}(\mathbf{S}_k)^2$ preconditioner of Eq. (5) plus the magnitude term $\mathbf{U}_k\,\mathrm{diag}(\Delta\mathbf{S}_k)\,\mathbf{V}_k^\top$. Merging $\mathbf{S}$ into the trainable core removes spectral reweighting ("fair" projector $\mathbf{U}_k\mathbf{U}_k^\top$, MiLoRA-like); merging it into the frozen core produces a principal-biased $\mathrm{diag}(\mathbf{S}_k)^2$-weighted projector (PiSSA-like). Derivations for all four placements: Appendix E.3.

6 Experiments

Setup. Tasks. We evaluate FuRA on two settings of major interest to the LLM community: 1) *Commonsense reasoning SFT*, in which we fine-tune on Commonsense-170K [16] and evaluate on the standard eight-task suite (BoolQ, PIQA, SIQA, HellaSwag, WinoGrande, ARC-e, ARC-c, OBQA) [4, 2, 38, 53, 37, 5, 32]; 2) *Math reinforcement learning with verifiable rewards*, in which we run GRPO [40] on a math prompt set and evaluate on four held-out benchmarks: MATH-500, AMC23, AIME-24, and AIME-25 [13, 42, 43, 44]. **Models.** For commonsense SFT we use LLaMA-2-7B [46] and LLaMA-3-8B [10]; for math RLVR we use Qwen3-1.7B [36] and Qwen2.5-7B [45]. We additionally evaluate 4-bit quantized fine-tuning on LLaMA-3-8B and LLaMA-3-70B. **Baseline methods.** We compare against Full FT and four families of parameter-efficient methods covering the leading LoRA variants: (i) *low-rank adapters* LoRA [14] and DoRA [25]; (ii) *SVD-initialized adapters* PiSSA [31] and MiLoRA [47]; and (iii) *full-rank capacity* RandLoRA [1];. For all baselines we use the rank and other hyperparameters reported as best in the original references. Full hyperparameters and sweeps are in Appendix C.

6.1 Commonsense reasoning SFT (LLaMA-2-7B and LLaMA-3-8B)

Table 3 reports per-task accuracy on the eight-task commonsense suite for LLaMA-2-7B and LLaMA-3-8B after training on Commonsense-170K. FuRA achieves 88.01% average accuracy on LLaMA-3-8B, outperforming LoRA by $+4.45$, DoRA by $+2.87$, and Full FT by $+1.27$. Methods that initialize LoRA with principal (PiSSA, $+2.44$ over LoRA) or off-principal (MiLoRA, $+1.50$) singular directions of the pretrained weight improve over LoRA; full-rank-capacity RandLoRA is essentially at LoRA parity (83.28 vs. 83.46); while none outperforms Full FT. In contrast, FuRA combines full-rank capacity, preservation of the full-rank pretrained spectral structure, and spectral preconditioning in a unified design, enabling it to surpass Full FT while tuning only 1.46% of parameters.

6.2 Math Reinforcement Learning

RL setup. We run GRPO [40] for 50 policy-gradient steps with 32 prompts per step and 8 rollouts per prompt. The reward is $\{0, 1\}$ based on symbolic equivalence of the extracted answer. Rollouts are generated with vLLM [21]; optimization uses AdamW [29]. AIME-24/25 are reported as avg@8

Table 3: Commonsense reasoning, fine-tuned on Commonsense-170K.

Method	Trainable %	BoolQ	PIQA	SIQA	HellaSwag	Wino	ARC-e	ARC-c	OBQA	Avg
LLaMA-2-7B										
Full FT	100.00	73.8	84.2	81.0	94.7	85.2	88.9	75.6	84.8	83.53
LoRA (r=64)	3.33	70.8	82.8	79.4	92.9	83.4	86.3	71.6	82.8	81.25
DoRA (r=64)	3.34	71.3	83.4	80.1	92.3	84.0	86.1	71.4	**85.8**	81.80
PiSSA (r=128)	3.33	72.5	85.3	80.8	87.2	86.1	87.1	74.3	85.6	82.36
FuRA (ours)	1.53	**74.6**	**86.7**	**82.5**	**95.2**	**86.3**	**89.0**	**75.6**	85.4	**84.41**
LLaMA-3-8B										
Full FT	100.00	75.4	88.0	81.8	96.5	89.3	93.1	83.0	86.0	86.64
LoRA (r=64)	1.39	71.8	85.3	80.9	93.4	84.5	90.0	77.0	84.8	83.46
DoRA (r=64)	1.42	74.6	87.4	81.2	94.7	87.1	89.4	79.5	86.4	85.04
PiSSA (r=128)	1.39	74.9	87.7	82.3	95.3	87.7	92.9	81.4	87.0	85.90
MiLoRA (r=64)	1.39	73.7	87.3	80.7	94.3	87.2	91.2	78.9	86.4	84.96
RandLoRA (r=64)	0.33	73.1	85.5	79.4	93.1	85.9	87.9	75.9	85.4	83.28
FuRA (ours)	1.46	**76.5**	**90.4**	**83.0**	**96.8**	**89.9**	**93.7**	**84.5**	**89.3**	**88.01**

Table 4: Math RL with GRPO on Qwen3-1.7B and Qwen2.5-7B, 50 policy-gradient steps.

Model	Method	MATH-500	AMC23	AIME-24	AIME-25
Qwen3-1.7B	Base	52.2	30.0	5.8	7.5
	Full FT	61.5	46.7	13.2	14.6
	LoRA [14]	59.6	48.3	9.2	11.1
	DoRA [25]	62.3	45.0	13.6	14.7
	PiSSA [31]	49.1	30.8	2.2	4.7
	MiLoRA [47]	58.5	41.7	7.6	11.7
	RandLoRA [1]	62.2	55.0	13.8	**18.3**
	FuRA (ours)	**62.5**	**55.8**	**13.8**	13.8
Qwen2.5-7B	Base	48.4	40.0	5.0	2.9
	Full FT	58.5	**49.2**	11.0	4.9
	LoRA [14]	58.5	**49.2**	11.5	4.3
	DoRA [25]	59.3	47.5	**12.3**	7.5
	FuRA (ours)	**59.7**	49.0	11.8	**7.5**

at temperature T=0.6; MATH-500 and AMC23 use greedy decoding (T=0). All runs are on a single NVIDIA H100 (94GB) for training.

Results. Beyond SFT, FuRA also outperforms other PEFT methods and matches Full FT in reinforcement learning with verifiable rewards (RLVR). Table 4 reports results on four held-out mathbenchmarks for Qwen3-1.7B and Qwen2.5-7B, all results are averaged over 3 independent seeds (mean±std are in Appendix D.3). On *Qwen3-1.7B*, FuRA outperforms Full FT and other PEFT methods on MATH-500, AMC23, AIME-24. PiSSA and MiLoRA underperform in RLVR due to their low-rank commitment to principal/off-principal spaces, which does not align with RLVR training dynamics [56, 50]. FuRA preserves full-rank spectrum at a similar parameter budget, thus generalizes better in RLVR. On *Qwen2.5-7B*, FuRA also matches or outperforms Full FT.

6.3 VLM Visual Instruction Tuning (LLaVA-1.5-7B)

We follow the DoRA [25] setup to perform visual instruction tuning for LLaVA-1.5-7B [24]. Table 5 reports the 7-task average over VQAv2 [9], GQA [17], VisWiz [11], ScienceQA [30], TextVQA [41], POPE [22], and MMBench [27]. FuRA exceeds Full FT by $+1.1$ and matches DoRA (67.6) with $\mathbf{3.4}\times$ fewer trainable parameters (1.37% vs. 4.63%). Per-task evaluation results are in Appendix D.5.

6.4 QFuRA: 4-bit quantized fine-tuning

Following QLoRA [7], we construct QFuRA by quantizing the frozen large core $\mathbf{L}$ in 4-bit NormalFloat (NF4) while keeping the trainable $\mathbf{R}$ and $\mathbf{S}$ in bf16. Tables 6 and 7 report the 8-task Commonsense-170K average on LLaMA-3-8B and GSM8K [6] accuracy after fine-tuning on MetaMathQA-100K [51] for LLaMA-3-70B. On LLaMA-3-8B, QFuRA reaches 87.30%, surpassing

Table 5: Visual instruction tuning of LLaVA-1.5-7B.

Method	Params	Avg.
Full FT	100%	66.5
LoRA [14]	4.61%	66.9
DoRA [25]	4.63%	67.6
FuRA	1.37%	**67.6**

Table 6: 4-bit quantized fine-tuning on LLaMA-3-8B.

Method	Params	Avg.
QLoRA [7]	1.39%	83.89
QDoRA [25]	1.42%	86.34
QFuRA	1.46%	**87.30**

Table 7: 4-bit quantized fine-tuning on LLaMA-3-70B.

Method	Params	GSM8K
QLoRA [7]	1.17%	81.27
QDoRA [25]	1.18%	81.80
QFuRA	1.45%	**83.78**

Table 8: Ablation study over FuRA design corners. Color code: **blue** = frozen, **red** = trainable.

Group	Notation	Train. %	SFT Avg	MATH-500	AMC23
Full	Full FT $\boldsymbol{W}$	100	86.64	63.6	47.5
	FuRA Full $\boldsymbol{L}\,\boldsymbol{S}\,\boldsymbol{R}$	100	**88.04**	**64.2**	**57.5**
PEFT (m=1)	$\boldsymbol{L}\,\boldsymbol{S}\,\boldsymbol{R}$ (default)	1.46	**87.91**	**63.6**	**57.5**
	$(\boldsymbol{L}\boldsymbol{S})\,\boldsymbol{R}$	1.46	87.12	61.4	55.0
	$\boldsymbol{L}\,(\boldsymbol{S}\boldsymbol{R})$	1.46	84.09	62.8	47.5
PEFT (n=1)	$\boldsymbol{L}\,\boldsymbol{S}\,\boldsymbol{R}$	1.46	86.79	61.2	52.5
	$(\boldsymbol{L}\boldsymbol{S})\,\boldsymbol{R}$	1.46	83.30	61.4	42.5
	$\boldsymbol{L}\,(\boldsymbol{S}\boldsymbol{R})$	1.46	83.75	62.2	42.5

QLoRA [7] by $+3.41$ and QDoRA [25] by $+0.96$ at 1.46% trainable parameters; on LLaMA-3-70B, QFuRA scores 83.78 on GSM8K, beating QLoRA and QDoRA. The results validate that FuRA is robust to quantizing the frozen large core. Per-task scores are in Appendix D.4.

6.5 Ablations: BlockTT design corners

We ablate FuRA design corners on LLaMA-3-8B Commonsense-170K SFT and Qwen3-1.7B GRPO math RL. Table 8 compares FuRA Full (all BlockTT factors trainable) and the six FuRA PEFT corners spanning the two design axes of Section 5 (Claim 2): block orientation (m=1 vs. n=1) and placement of $\mathbf{S}$ (separate trainable, merged into the frozen core, or merged into the trainable core).

Training every factor (FuRA Full) is the best, but the PEFT variant closes nearly the entire gap (87.91 vs. 88.04 SFT; matching RL), confirming that the gain comes from full-rank spectral preconditioning rather than additional trainable capacity. Across the six PEFT corners the default preserving output subspace ($m = 1$) and keeping singular values as separate parameters ($\boldsymbol{L}\,\boldsymbol{S}\,\boldsymbol{R}$) wins on both SFT and RL. Intuitively, freezing $\mathbf{L}$ retains the pretrained output feature space while letting the trainable $\mathbf{R}$ remix based on new input patterns in the fine-tuning data; freezing $\mathbf{R}$ instead pins the input subspace and could filter out task-specific input features that the fine-tuning data carry. We additionally ablate the BlockTT shape factorization (n, b) of the input dimension and show that balanced factorization achieves better result; results are in Appendix D.6.

7 Conclusion, Limitation, and Broader Impact

We have identified spectral preconditioning as a missing ingredient in current fine-tuning, and have proposed FuRA, which delivers full-rank update capacity, spectral preconditioning, and LoRA-level parameter efficiency from a single architectural choice. We have demonstrated that in both SFT and RLVR FuRA matches or surpasses Full FT with level step time and GPU memory on par with LoRA. Several directions remain open: a deeper theoretical account of *why* spectral-preconditioned updates improve generalization; initializations from decompositions beyond SVD; custom kernels to further accelerate the block tensor-train contraction; and a stronger QFuRA that exploits the orthonormal rows of the frozen core $\mathbf{L}$ for more aggressive quantization. We hope these directions inspire further research on the spectral structure of pretrained models and the design of parameter-efficient adaptation. FuRA could reduce computation and energy cost to access high-quality model adaptation and avoid excessive energy spent on tuning LoRA ranks. It inherits the general societal risks of LLM fine-tuning but introduces no new generative capabilities beyond those of the base checkpoints.

References

[1] P. Albert, F. Z. Zhang, H. S. Rodriguez, C. Pham, E. Abbasnejad, and A. van den Hengel. RandLoRA: Full-rank parameter-efficient fine-tuning of large models. In *International Conference on Learning Representations*, 2025.

[2] Y. Bisk, R. Zellers, R. Le Bras, J. Gao, and Y. Choi. PIQA: Reasoning about physical commonsense in natural language. In *Proceedings of the AAAI Conference on Artificial Intelligence*, 2020.

[3] Z. Chen, R. R. Yang, S. Singh, and M. Soljacic. QuanTA: Efficient high-rank fine-tuning of LLMs with quantum-informed tensor adaptation. In *Advances in Neural Information Processing Systems*, 2024.

[4] C. Clark, K. Lee, M.-W. Chang, T. Kwiatkowski, M. Collins, and K. Toutanova. BoolQ: Exploring the surprising difficulty of natural yes/no questions. In *Proceedings of the Conference of the North American Chapter of the Association for Computational Linguistics*, 2019.

[5] P. Clark, I. Cowhey, O. Etzioni, T. Khot, A. Sabharwal, C. Schoenick, and O. Tafjord. Think you have solved question answering? Try ARC, the AI2 reasoning challenge. *arXiv preprint arXiv:1803.05457*, 2018.

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

[7] T. Dettmers, A. Pagnoni, A. Holtzman, and L. Zettlemoyer. QLoRA: Efficient finetuning of quantized LLMs. In *Advances in Neural Information Processing Systems*, 2023.

[8] Z. Gao, Q. Wang, A. Chen, Z. Liu, B. Wu, L. Chen, and J. Li. Parameter-efficient fine-tuning with discrete Fourier transform. In *International Conference on Machine Learning*, 2024.

[9] Y. Goyal, T. Khot, D. Summers-Stay, D. Batra, and D. Parikh. Making the V in VQA matter: Elevating the role of image understanding in Visual Question Answering. In *IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, 2017.

[10] A. Grattafiori, A. Dubey, A. Jauhri, A. Pandey, A. Kadian, et al. The Llama 3 herd of models. *arXiv preprint arXiv:2407.21783*, 2024.

[11] D. Gurari, Q. Li, A. J. Stangl, A. Guo, C. Lin, K. Grauman, J. Luo, and J. P. Bigham. VizWiz grand challenge: Answering visual questions from blind people. In *IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, 2018.

[12] N. Halko, P.-G. Martinsson, and J. A. Tropp. Finding structure with randomness: Probabilistic algorithms for constructing approximate matrix decompositions. *SIAM review*, 53(2):217–288, 2011.

[13] D. Hendrycks, C. Burns, S. Kadavath, A. Arora, S. Basart, E. Tang, D. Song, and J. Steinhardt. Measuring mathematical problem solving with the MATH dataset. In *Advances in Neural Information Processing Systems Datasets and Benchmarks Track*, 2021.

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

[15] I. Hu, R. Cong, A. Zhang, A. Shetty, V. Lingam, W.-L. Chou, A. G. Dimakis, and S. Sanghavi. LoRTA: Low rank tensor adaptation of large language models. *arXiv preprint arXiv:2410.04060*, 2024.

[16] Z. Hu, L. Wang, Y. Lan, W. Xu, E.-P. Lim, L. Bing, X. Xu, S. Poria, and R. K.-W. Lee. LLM-Adapters: An adapter family for parameter-efficient fine-tuning of large language models. *arXiv preprint arXiv:2304.01933*, 2023.

[17] D. A. Hudson and C. D. Manning. GQA: A new dataset for real-world visual reasoning and compositional question answering. In *IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, 2019.

[18] T. Jiang, S. Huang, S. Luo, Z. Zhang, H. Huang, F. Wei, W. Deng, F. Sun, Q. Zhang, D. Wang, and F. Zhuang. MoRA: High-rank updating for parameter-efficient fine-tuning. In *International Conference on Learning Representations*, 2025.

[19] J. Kirkpatrick, R. Pascanu, N. Rabinowitz, J. Veness, G. Desjardins, A. A. Rusu, K. Milan, J. Quan, T. Ramalho, A. Grabska-Barwinska, et al. Overcoming catastrophic forgetting in neural networks. *Proceedings of the national academy of sciences*, 114(13):3521–3526, 2017.

[20] D. J. Kopiczko, T. Blankevoort, and Y. M. Asano. VeRA: Vector-based random matrix adaptation. In *International Conference on Learning Representations*, 2024.

[21] W. Kwon, Z. Li, S. Zhuang, Y. Sheng, L. Zheng, C. H. Yu, J. E. Gonzalez, H. Zhang, and I. Stoica. Efficient memory management for large language model serving with PagedAttention. In *Proceedings of the ACM Symposium on Operating Systems Principles*, 2023.

[22] Y. Li, Y. Du, K. Zhou, J. Wang, W. X. Zhao, and J.-R. Wen. Evaluating object hallucination in large vision-language models. In *Conference on Empirical Methods in Natural Language Processing (EMNLP)*, 2023.

[23] V. Lingam, A. Tejaswi, A. Vavre, A. Shetty, G. K. Gudur, J. Ghosh, A. Dimakis, E. Choi, A. Bojchevski, and S. Sanghavi. SVFT: Parameter-efficient fine-tuning with singular vectors. In *Advances in Neural Information Processing Systems*, 2024.

[24] H. Liu, C. Li, Q. Wu, and Y. J. Lee. Visual instruction tuning. In *Advances in Neural Information Processing Systems*, 2023.

[25] S.-Y. Liu, C.-Y. Wang, H. Yin, P. Molchanov, Y.-C. F. Wang, K.-T. Cheng, and M.-H. Chen. DoRA: Weight-decomposed low-rank adaptation. In *International Conference on Machine Learning*, 2024.

[26] W. Liu, Z. Qiu, Y. Feng, Y. Xiu, Y. Xue, L. Yu, H. Feng, Z. Liu, J. Heo, S. Peng, Y. Wen, M. J. Black, A. Weller, and B. Schölkopf. Parameter-efficient orthogonal finetuning via butterfly factorization. In *International Conference on Learning Representations*, 2024.

[27] Y. Liu, H. Duan, Y. Zhang, B. Li, S. Zhang, W. Zhao, Y. Yuan, J. Wang, C. He, Z. Liu, K. Chen, and D. Lin. MMBench: Is your multi-modal model an all-around player? *arXiv preprint arXiv:2307.06281*, 2023.

[28] Z. Liu, T. Pang, O. Balabanov, C. Yang, T. Huang, L. Yin, Y. Yang, and S. Liu. LIFT the veil for the truth: Principal weights emerge after rank reduction for reasoning-focused supervised fine-tuning. In *International Conference on Machine Learning*, 2025.

[29] I. Loshchilov and F. Hutter. Decoupled weight decay regularization. In *International Conference on Learning Representations*, 2019.

[30] P. Lu, S. Mishra, T. Xia, L. Qiu, K.-W. Chang, S.-C. Zhu, O. Tafjord, P. Clark, and A. Kalyan. Learn to explain: Multimodal reasoning via thought chains for science question answering. In *Advances in Neural Information Processing Systems*, 2022.

[31] F. Meng, Z. Wang, and M. Zhang. PiSSA: Principal singular values and singular vectors adaptation of large language models. In *Advances in Neural Information Processing Systems*, 2024.

[32] T. Mihaylov, P. Clark, T. Khot, and A. Sabharwal. Can a suit of armor conduct electricity? A new dataset for open book question answering. In *Proceedings of the Conference on Empirical Methods in Natural Language Processing*, 2018.

[33] A. Novikov, D. Podoprikhin, A. Osokin, and D. Vetrov. Tensorizing neural networks. In *Advances in Neural Information Processing Systems*, 2015.

[34] I. V. Oseledets. Tensor-train decomposition. *SIAM Journal on Scientific Computing*, 33(5):2295–2317, 2011.

[35] S. Qiu, A. Potapczynski, M. Finzi, M. Goldblum, and A. G. Wilson. Compute better spent: Replacing dense layers with structured matrices. In *International Conference on Machine Learning*, 2024.

[36] Qwen Team. Qwen3 technical report. *arXiv preprint arXiv:2505.09388*, 2025.

[37] K. Sakaguchi, R. Le Bras, C. Bhagavatula, and Y. Choi. WinoGrande: An adversarial Winograd schema challenge at scale. *Communications of the ACM*, 64(9):99–106, 2020.

[38] M. Sap, H. Rashkin, D. Chen, R. Le Bras, and Y. Choi. Social IQa: Commonsense reasoning about social interactions. In *Proceedings of the Conference on Empirical Methods in Natural Language Processing*, 2019.

[39] J. Schulman and Thinking Machines Lab. LoRA without regret. `https://thinkingmachines.ai/blog/lora/`, 2025. Blog post.

[40] Z. Shao, P. Wang, Q. Zhu, R. Xu, J. Song, X. Bi, H. Zhang, M. Zhang, Y. K. Li, Y. Wu, and D. Guo. DeepSeekMath: Pushing the limits of mathematical reasoning in open language models. *arXiv preprint arXiv:2402.03300*, 2024.

[41] A. Singh, V. Natarajan, M. Shah, Y. Jiang, X. Chen, D. Batra, D. Parikh, and M. Rohrbach. Towards VQA models that can read. In *IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, 2019.

[42] M.-A. Team. Amc 2023 dataset, 2023.

[43] M.-A. Team. American invitational mathematics examination (aime) 2024, 2024.

[44] M.-A. Team. American invitational mathematics examination (aime) 2024, 2025.

[45] Q. Team, A. Yang, B. Yang, B. Zhang, B. Hui, B. Zheng, B. Yu, C. Li, D. Liu, F. Huang, et al. Qwen2.5 technical report. *arXiv preprint arXiv:2412.15115*, 2025.

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

[47] H. Wang, Y. Li, S. Wang, G. Chen, and Y. Chen. MiLoRA: Harnessing minor singular components for parameter-efficient LLM finetuning. *arXiv preprint arXiv:2406.09044*, 2024.

[48] X. Yang, J. Leng, G. Guo, J. Zhao, R. Nakada, L. Zhang, H. Yao, and B. Chen. S^2FT: Efficient, scalable and generalizable LLM fine-tuning by structured sparsity. In *Advances in Neural Information Processing Systems*, 2024.

[49] Y. Yang, K. Zhen, E. Banijamali, A. Mouchtaris, and Z. Zhang. LoRETTA: Low-rank economic tensor-train adaptation for ultra-low-parameter fine-tuning of large language models. In *Proceedings of the Conference of the North American Chapter of the Association for Computational Linguistics*, 2024.

[50] Q. Yin, Y. Wu, Z. Shen, S. Li, Z. Wang, Y. Li, C. T. Leong, J. Kang, and J. Gu. Evaluating parameter efficient methods for rlvr. *arXiv preprint arXiv:2512.23165*, 2025.

[51] L. Yu, W. Jiang, H. Shi, J. Yu, Z. Liu, Y. Zhang, J. T. Kwok, Z. Li, A. Weller, and W. Liu. Metamath: Bootstrap your own mathematical questions for large language models. *arXiv preprint arXiv:2309.12284*, 2023.

[52] Q. Yu, Z. Zhang, R. Zhu, Y. Yuan, X. Zuo, Y. Yue, et al. DAPO: An open-source LLM reinforcement learning system at scale. *arXiv preprint arXiv:2503.14476*, 2025.

[53] R. Zellers, A. Holtzman, Y. Bisk, A. Farhadi, and Y. Choi. HellaSwag: Can a machine really finish your sentence? In *Proceedings of the Annual Meeting of the Association for Computational Linguistics*, 2019.

[54] F. Zhang and M. Pilanci. Spectral adapter: Fine-tuning in spectral space. In *Advances in Neural Information Processing Systems*, 2024.

[55] Q. Zhang, M. Chen, A. Bukharin, N. Karampatziakis, P. He, Y. Cheng, W. Chen, and T. Zhao. AdaLoRA: Adaptive budget allocation for parameter-efficient fine-tuning. In *International Conference on Learning Representations*, 2023.

[56] H. Zhu, Z. Zhang, H. Huang, D. Su, Z. Liu, J. Zhao, I. Fedorov, H. Pirsiavash, Z. Sha, J. Lee, D. Z. Pan, Z. Wang, Y. Tian, and K. S. Tai. The path not taken: RLVR provably learns off the principals. *arXiv preprint arXiv:2511.08567*, 2025. NeurIPS 2025 Workshop on Efficient Reasoning (spotlight).

A FuRA Algorithm

Algorithm 1 FuRA: Full-Rank Adaptation

Require: Pretrained weight $\mathbf{W} \in \mathbb{R}^{d_{\text{out}} \times d_{\text{in}}}$, block count n, block width $b = d_{\text{in}}/n$
Reshape $\mathbf{W}$ into blocks: $\boldsymbol{\mathcal{W}} \in \mathbb{R}^{n \times d_{\text{out}} \times b}$
for $k = 1, \ldots, n$ **do**
  $\mathbf{U}_k, \mathbf{\Sigma}_k, \mathbf{V}_k \leftarrow \text{SVD}(\boldsymbol{\mathcal{W}}_k)$ ▷ lossless, $r = b$
end for
$\mathbf{L}_k \leftarrow \mathbf{U}_k, \quad \mathbf{S}_k \leftarrow \text{diag}(\mathbf{\Sigma}_k), \quad \mathbf{R}_k \leftarrow \mathbf{V}_k^\top$ ▷ assign cores
Freeze $\mathbf{L}$; set $\mathbf{R}$ and $\mathbf{S}$ as trainable
while training **do**
  Reshape input: $\boldsymbol{\mathcal{X}} \in \mathbb{R}^{n \times b}$
  Forward: $\mathbf{y} = \sum_{k=1}^{n} \mathbf{L}_k \,\text{diag}(\mathbf{S}_k)\, \mathbf{R}_k \,\boldsymbol{\mathcal{X}}_k$ ▷ two batched GEMMs
  Update $\mathbf{R}$ and $\mathbf{S}$ via optimizer (gradient flows through $\mathbf{L}$ but $\mathbf{L}$ is not updated)
end while
Deploy: merge cores into $\mathbf{W}' \in \mathbb{R}^{d_{\text{out}} \times d_{\text{in}}}$ via Eq. (4) ▷ no serving overhead

B FuRA Complexity analysis

This section expands the brief complexity discussion in Section 4 (and the headline numbers in Table 1).

B.1 System cost measurement protocol

This section documents the protocol used for the step-time and peak-memory numbers in Table 1.

Setup. LLaMA-3-8B Commonsense-170K SFT, 300 optimizer steps, single H100 NVL (95.8 GB), seed 43. Per-device batch 8, gradient accumulation 2 (effective tokens/step 32,768), sequence length 2048, bf16 mixed precision, gradient checkpointing on, AdamW with η=2×10^{-4}, linear LR schedule with 3% warmup. Identical for every method; only the adapter recipe varies.

Reported metrics. *Step (s)* is the median wall-clock per optimizer step over the final 200 steps (after a 100-step warmup that excludes adapter-attach and SVD initialization). *Memory (GB)* is `torch.cuda.max_memory_allocated` over the full run, including the warmup window..

B.2 Results and Analysis

FuRA matches the per-step time of LoRA/DoRA at comparable budgets, achieves the *lowest* peak GPU memory, and uniquely enables full-rank updates. We detail this below.

Trainable parameters. The trainable count is $|\mathbf{R}| + |\mathbf{S}| = nrb + nr$. For a square $d \times d$ layer with $n = b = r = \sqrt{d}$, this is $\approx d^{3/2}$, i.e., $\mathcal{O}(1/\sqrt{d})$ of d^2. At $d = 4096$, $\mathbf{R}$ has $\sim$0.26M parameters, and the overall trainable fraction on LLaMA-3-8B is 1.46%, comparable to LoRA/DoRA ($\sim$1.4%) and far below LIFT's $\sim$6.2%.

Forward pass. For a FuRA layer with $d_{\text{in}} = n \cdot b$, and rank $r = b$, the cores have shapes $\mathbf{R} \in \mathbb{R}^{n \times r \times b}$, $\mathbf{L} \in \mathbb{R}^{n \times d_{\text{out}} \times r}$, and $\mathbf{S} = [\mathbf{S}_1, \ldots, \mathbf{S}_n] \in \mathbb{R}^{n \times r}$ with each $\mathbf{S}_k \in \mathbb{R}^r$ stored as a vector. The forward pass applies, for input $\mathbf{x} \in \mathbb{R}^{d_{\text{in}}}$ reshaped as $\boldsymbol{\mathcal{X}} \in \mathbb{R}^{n \times b}$,

$$\boldsymbol{\mathcal{Z}}_k = \mathbf{R}_k \boldsymbol{\mathcal{X}}_k \qquad (k = 1, \ldots, n), \quad \boldsymbol{\mathcal{Z}} \in \mathbb{R}^{n \times r},$$
$$\widetilde{\boldsymbol{\mathcal{Z}}}_k = \text{diag}(\mathbf{S}_k) \boldsymbol{\mathcal{Z}}_k,$$
$$\mathbf{y} = \sum_{k=1}^{n} \mathbf{L}_k \widetilde{\boldsymbol{\mathcal{Z}}}_k, \quad \mathbf{y} \in \mathbb{R}^{d_{\text{out}}}.$$

Table 1 reports per-step wall-clock, throughput, and peak GPU memory for 300-step commonsense SFT on LLaMA-3-8B. FuRA runs at 0.046 s/step, matching LoRA/DoRA and $4.3\times$ faster than Full FT. Peak memory is 22.7 GB, lower than LoRA/DoRA and $3.4\times$ below Full FT. Efficiency comes from a sequential design: the $\mathbf{L}$-stage reduces to a standard dense MVM, while the $\mathbf{R}$-stage uses fused small MVMs via `torch.bmm`. With $b = \sqrt{d} = 64$, these align well with tensor cores, improving utilization over LoRA's tall-skinny GEMMs.

Memory. FuRA achieves lower peak memory than LoRA because its sequential factorization $\mathbf{L}\,\mathbf{S}\,\mathbf{R}\,\mathbf{x}$ produces a single intermediate activation per layer of size $\mathbb{R}^{n\times r}$, whereas LoRA's parallel residual $\mathbf{W}\mathbf{x} + \mathbf{B}\mathbf{A}\mathbf{x}$ stashes *two* additional activations: the down-projection $\mathbf{A}\mathbf{x} \in \mathbb{R}^{T\times r}$ and the input $\mathbf{x} \in \mathbb{R}^{T\times d}$ feeding $\mathbf{B}\mathbf{A}\mathbf{x}$. Compared to sparse fine-tuning method LIFT [28], FuRA uses dramatically less memory because LIFT must additionally keep the full bf16 backbone gradient buffer ($\sim$16 GB on 8 B params) plus a per-weight bool mask, whereas FuRA produces its full-rank update entirely through the small factored cores $\mathbf{R}, \mathbf{S}$ and never materializes a backbone-sized gradient.

Deployment. After training, the three cores $\mathbf{L}, \mathbf{S}, \mathbf{R}$ merge into a single dense matrix $\mathbf{W}' \in \mathbb{R}^{d_{\text{out}}\times d_{\text{in}}}$ via Eq. (4), incurring no serving latency relative to the base model.

Initialization cost. Block-SVD initialization is a one-time cost: 52.9 s on LLaMA-3-8B (160 target linear layers), versus $\sim$1 s for LoRA/DoRA. This is $< 0.2\%$ of full SFT runtime (10 hours for Commonsense-170K finetuning). The cost can be further reduced by Fast SVD methods like [12].

C Hyperparameters

FuRA hyperparameters for Tables 1, 3, 4, 6, and 7. Tables 9, 10, 11, and 13 list the FuRA training configuration used for the four task suites. All FuRA runs use the project default corner: $m = 1$ and $\mathbf{S}$ kept as a separate trainable vector. The LLaMA-3-8B QFuRA row of Table 6 uses the same settings as the LLaMA-3-8B column of Table 9, with the frozen large core $\mathbf{L}$ NF4-quantized via `bitsandbytes` and the optimizer swapped to `PagedAdamW8bit`. The LLaMA-3-70B QFuRA row of Table 7 uses Table 13.

Table 9: FuRA hyperparameter configuration for the commonsense reasoning tasks (Table 3).

Hyperparameters (FuRA)	LLaMA-2-7B	LLaMA-3-8B
Dropout	0.0	0.0
Optimizer	AdamW	AdamW
LR	3×10^{-4}	2×10^{-4}
LR Scheduler	Linear	Linear
Weight decay	0	0
Batch size (per-device $\times$ grad-accum)	$8 \times 2 = 16$	$8 \times 2 = 16$
Warmup ratio	0.03	0.03
Epochs	3	3
Max sequence length	2048	2048
Mixed precision	bf16	bf16
Gradient checkpointing	on	on
Where	Q, K, V, O, Up, Down, Gate	

Table 10: FuRA hyperparameter configuration for math GRPO RL (Table 4). Reward $\in \{0, 1\}$ via `math_utils.is_equiv`; vLLM rollout at `gpu_memory_utilization`$= 0.25$; GRPO clip ratio 0.2, KL coefficient 0, no entropy bonus.

Hyperparameters (FuRA)	Qwen3-1.7B	Qwen2.5-7B
Optimizer	AdamW	AdamW
LR	1×10^{-4}	1×10^{-4}
LR Scheduler	none (constant)	none (constant)
Warmup ratio	0.0	0.0
Weight decay	0	0
GRPO steps	50	50
Prompts per step	32	32
Group size (rollouts / prompt)	8	8
Epochs per step	1	1
Max model length (vLLM)	2048	2048
Eval max tokens	2048	2048
Train rollout temperature	1.0	1.0
Eval temperature	0.6 (AIME-24/25), 0.0 (MATH-500/AMC23)	
Seed	42	42
Where	All linear projections (`trainable_type=all`)	

Table 11: FuRA hyperparameter configuration for visual instruction tuning of LLaVA-1.5-7B on `llava_v1_5_mix665k` (5,197 steps $= 1$ epoch). Same recipe as the LLaVA-1.5 DoRA fine-tune of [25] with FuRA substituted for DoRA; best LR selected from $\{2, 3, 4\}\times10^{-4}$.

Hyperparameters (FuRA)	LLaVA-1.5-7B
Dropout	0.05
Optimizer	AdamW
LR	3×10^{-4}
LR Scheduler	Cosine decay
Weight decay	0
Batch size (per-device $\times$ grad-accum)	$4 \times 4 = 16$
Warmup ratio	0.03
Epochs	1
Model max length	2048
Mixed precision	bf16
Gradient checkpointing	on
Where	Q, K, V, O, Up, Down, Gate

Table 12: Math-10K SFT hyperparameter configuration for the LR / batch-size sweep on LLaMA-3-8B (§D.1). All three methods share the same training schedule (3 epochs on Math-10K, AdamW, linear LR decay, 0.03 warmup ratio, max sequence length 2048, bf16); only LR and effective batch size vary across the sweep. Best LRs found per (method, effective batch size) are reported in §D.1.

Hyperparameters	Full FT	LoRA	FuRA
Optimizer	AdamW		
LR Scheduler	Linear		
Weight decay	0		
Warmup ratio	0.03		
Epochs	3		
Max sequence length	2048		
Mixed precision	bf16		
Adapter rank / α	—	r=64, α=128	r=full, m=1
Effective batch size (sweep)	$\{16, 64, 256\}$	$\{16, 64, 256\}$	$\{16, 64, 256\}$
LR sweep (bsz $= 16$)	$\{8, 10, 20, 30\}\times10^{-6}$	$\{3, 6, 10, 20\}\times10^{-5}$	$\{2, 3, 4, 6\}\times10^{-4}$
Seed	43		
Where	Q, K, V, O, Up, Down, Gate		

Table 13: QFuRA hyperparameter configuration for math fine-tuning of LLaMA-3-70B on MetaMathQA-100K (Table 7). Mirrors the QPiSSA recipe ($\eta = 2\times10^{-5}$, batch 1×128, sequence length 512) with the rank-r SVD adapter replaced by QFuRA's quantized BlockTT factorization. The frozen large core $\mathbf{L}$ is 4-bit-quantized via `bitsandbytes` storage layout; the trainable cores $\mathbf{R}$ and $\mathbf{S}$ stay in bf16.

Hyperparameters (QFuRA)	LLaMA-3-70B
Optimizer	PagedAdamW 8-bit
LR	2×10^{-5}
LR Scheduler	Cosine decay
Weight decay	0
Batch size (per-device $\times$ grad-accum)	$1 \times 128 = 128$
Warmup ratio	0.03
Steps	100
Max sequence length	512
Mixed precision	bf16
Trainable param dtype	bf16
Seed	42
Where	Q, K, V, O, Up, Down, Gate

D Extended experimental results

This section collects the per-task scores behind the headline averages reported in the body. Subsection D.3 expands Table 4 (Math RL with GRPO on Qwen3-1.7B and Qwen2.5-7B) into per-seed mean$\pm$std across 3 seeds. Subsection D.4 expands Table 6 (QFuRA on LLaMA-3-8B Commonsense-170K) to all eight tasks, Subsection D.5 expands Table 5 (FuRA on LLaVA-1.5-7B visual instruction tuning) to all seven vision-language benchmarks, and Subsection D.1 reports the LR and effective-batch-size sweep for LLaMA-3-8B SFT on Math-10K.

D.1 LLaMA-3-8B Math-10K SFT: LR and batch-size sweep

We additionally ran an LR and effective-batch-size sweep on LLaMA-3-8B SFT on Math-10K [28], evaluated on GSM8K [6]. Hyperparameters shared across all runs are in Table 12. We sweep the learning rate at the default effective batch size of 16 for each method: Full FT over $\{8\times10^{-6}, 1\times10^{-5}, 2\times10^{-5}, 3\times10^{-5}\}$, LoRA over $\{3\times10^{-5}, 6\times10^{-5}, 1\times10^{-4}, 2\times10^{-4}\}$, and FuRA over $\{2, 3, 4, 6\}\times10^{-4}$. We then sweep the effective batch size in $\{16, 64, 256\}$, picking the best LR per (method, batch size) by the standard square-root-rule rescaling around the small-batch optimum. Best LRs we found are $\eta = 1\times10^{-5}$ (Full FT, bsz 16), $\eta = 2\times10^{-5}$ (Full FT, bsz 64 and 256); $\eta = 6\times10^{-5}$ (LoRA, bsz 16), $\eta = 2\times10^{-4}$ (LoRA, bsz 64), $\eta = 6\times10^{-4}$ (LoRA, bsz 256); $\eta = 3\times10^{-4}$ (FuRA, bsz 16), $\eta = 6\times10^{-4}$ (FuRA, bsz 64), $\eta = 8\times10^{-4}$ (FuRA, bsz 256). Figure 5 plots the resulting GSM8K accuracy across the LR sweep at bsz $= 16$ (left) and across the batch-size sweep with best per-batch LR (right).

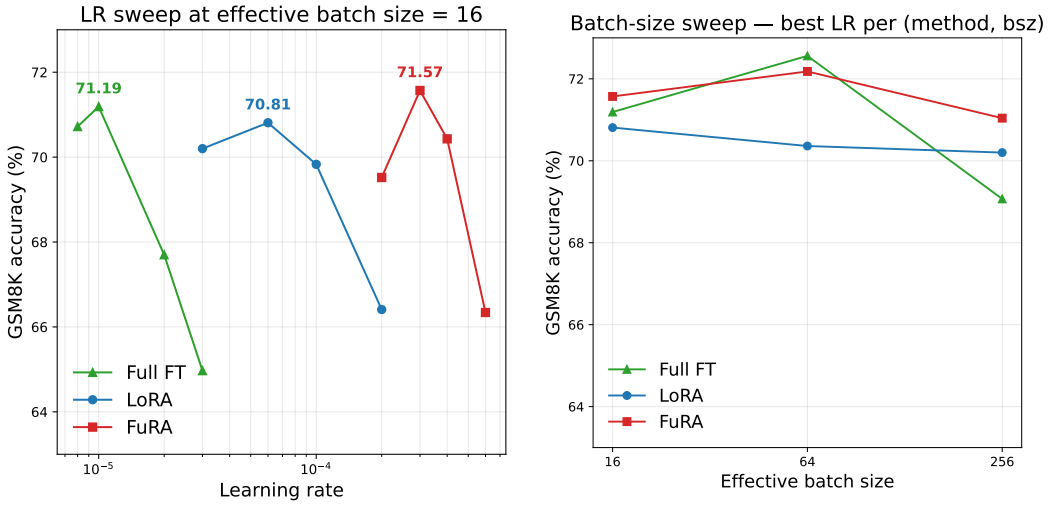

Figure 5: LLaMA-3-8B Math-10K SFT, GSM8K accuracy. **Left**: LR sweep at effective batch size $= 16$. **Right**: batch-size sweep. Peak per method is annotated on the left panel.

D.2 LLaMA-2-7B and LLaMA-3-8B Commonsense-170K Results

Source of the baseline numbers. For Tables 3, the FuRA result is the averaged result across 3 random seeds, with specified numbers per seed in Table 14. FuRA result has low variance across different random seeds. The LLaMA-2-7B rows and the LLaMA-3-8B Full FT / LoRA / DoRA rows, together with the LIFT comparison numbers are reproduced from the hyperparameter setting in LIFT paper [28], which itself uses the LLM-Adapters [16] recipe of r=64 adapters on five linear modules `q_proj k_proj v_proj up_proj down_proj`, best learning rate for each method, and others aligned with Tables 9. Tuning all 7 modules downgrades the LoRA performance [16]. We additionally ran the *RandLoRA, MiLoRA, and PiSSA* baselines on LLaMA-3-8B in-house under the same recipe and 8-task evaluation harness, sweeping the learning rate per method. RandLoRA and MiLoRA peak at $\eta = 1\times10^{-4}$; PiSSA peaks at $\eta = 2\times10^{-5}$.

Table 14: FuRA (PEFT) seed sweep on LLaMA-3-8B fine-tuned on Commonsense-170K (3 epochs).

Seed	BoolQ	PIQA	SIQA	HellaSwag	Wino	ARC-e	ARC-c	OBQA	Avg
42	76.30	90.60	83.00	96.80	90.10	93.80	85.40	89.00	88.13
43	76.60	89.90	83.20	96.80	89.70	93.60	84.10	89.40	87.91
44	76.50	90.80	82.80	96.90	89.80	93.60	84.00	89.60	88.00
Mean	76.47	90.43	83.00	96.83	89.87	93.67	84.50	89.33	88.01
Std	0.12	0.39	0.16	0.05	0.17	0.09	0.64	0.25	0.09

D.3 Math RL: per-seed mean±std across 3 seeds

This subsection expands Table 4 into per-seed mean±std across three independent seeds (42, 43, 44) for each (model, method) cell. Sample std is taken across n=3 seeds, so the SEM $\approx 0.6\times$ std. AIME-24/25 are avg@8 at T=0.6; MATH-500 and AMC23 are greedy@1 at T=0. The Paper Table (Table 4) reports the per-seed mean from the same set of runs.

Table 15: Math RL with GRPO: per-seed mean±std across 3 seeds (42, 43, 44). The Paper Table (Table 4) reports the per-seed mean of the same runs.

Model	Method	MATH-500	AMC23	AIME-24	AIME-25
Qwen3-1.7B	Full FT	61.53 ± 2.10	46.67 ± 3.82	13.21 ± 0.66	14.58 ± 1.10
	LoRA	59.60 ± 0.92	48.33 ± 7.64	9.15 ± 2.90	11.10 ± 0.63
	DoRA	62.33 ± 2.58	45.00 ± 10.90	13.60 ± 4.42	14.73 ± 0.24
	PiSSA	49.07 ± 5.56	30.83 ± 10.10	2.22 ± 1.27	4.71 ± 4.34
	MiLoRA	58.53 ± 0.50	41.67 ± 3.82	7.63 ± 0.86	11.67 ± 1.81
	RandLoRA	62.23 ± 0.58	55.00 ± 2.50	13.83 ± 1.03	18.30 ± 1.46
	FuRA	62.47 ± 1.33	55.83 ± 1.44	13.75 ± 1.10	13.75 ± 1.50
Qwen2.5-7B	Full FT	58.47 ± 2.53	49.17 ± 5.77	10.97 ± 2.69	4.87 ± 1.96
	LoRA	58.47 ± 0.23	49.17 ± 10.10	11.52 ± 1.22	4.30 ± 1.47
	DoRA	59.30 ± 0.42	47.50 ± 3.54	12.29 ± 2.67	7.49 ± 1.16
	FuRA	59.73 ± 0.99	49.00 ± 2.50	11.81 ± 1.20	7.50 ± 2.54

D.4 QFuRA: per-task scores on Commonsense-170K (LLaMA-3-8B)

Table 16: 4-bit QFuRA vs. QLoRA, QDoRA, and bf16 full fine-tuning on LLaMA-3-8B Commonsense-170K. **Bold** = column-best across the 4-bit quantized methods.

Method	Trainable %	BoolQ	PIQA	SIQA	HellaSwag	Wino	ARC-e	ARC-c	OBQA	Avg
Full FT (bf16)	100.00	75.4	88.0	81.8	96.5	89.3	93.1	83.0	86.0	86.64
4-bit QLoRA (r=64)	2.05	72.7	86.6	80.8	93.7	85.4	89.9	76.2	85.8	83.89
4-bit QDoRA (r=64)	2.06	**75.5**	88.0	81.1	95.9	88.4	91.8	82.0	88.0	86.34
4-bit QFuRA (ours)	**1.46**	73.0	**89.9**	**82.7**	**96.6**	**89.1**	**93.1**	**83.4**	**90.6**	**87.30**

Table 6 (full results Table 16) are all our own runs as specified in Tables 9. QFuRA wins seven of eight per-task columns among the 4-bit methods (all but BoolQ, where QDoRA leads by 2.5pp), and the gain over QDoRA is concentrated on ARC-c (+1.4), ARC-e (+1.3), OBQA (+2.6), and PIQA (+1.9). QFuRA's training and evaluation hyperparameters are documented in Table 9 (LLaMA-3-8B column) with the only delta being NF4 quantization of $\mathbf{L}$ via `bitsandbytes` and the `PagedAdamW8bit` optimizer.

D.5 VLM visual instruction tuning: per-task scores (LLaVA-1.5-7B)

We follow the exact LLaVA-1.5 [24] + DoRA [25] fine-tuning recipe: 1 epoch on `llava_v1_5_mix665k` (5,197 optimizer steps), per-device batch 4 with gradient accumulation 4 (effective batch 16), sequence length 2048, bf16, gradient checkpointing on, AdamW, cosine LR schedule with 3% warmup, and `mm_projector_lr`= 2×10^{-5}. FuRA replaces DoRA on all seven linear projections of the language tower (Q, K, V, O, Gate, Up, Down) under the project default corner (`output_one_block` / small core trainable / `s_merged_to=keep_trainable` / full rank), with the CLIP-L/336 vision encoder and the MLP projector kept in their LLaVA-1.5 recipe. We swept the FuRA LR over $\{2, 3, 4\}\times10^{-4}$; LR= 3×10^{-4} wins the 7-task average and is the row reported below.

Evaluation protocol. We follow the DoRA paper's Table 12 conventions: VQAv2 = test-dev2015 Overall (server eval); GQA = testdev_balanced; VisWiz = test Overall (server eval); SQA = full Acc on all 21K questions; VQA^{T} = TextVQA val Acc; POPE = mean F1 across {random, popular, adversarial}$\times100$; MMBench = dev split, CircularEval. Full FT, LoRA, and DoRA columns are quoted from [25] Table 12.

Table 17: Per-task LLaVA-1.5-7B visual instruction tuning results on the 7 vision-language benchmarks of [25], Table 12. Full FT / LoRA / DoRA quoted from the DoRA paper; FuRA is our run at LR$= 3\times10^{-4}$ (best of $\{2,3,4\}\times10^{-4}$). **Bold** = column-best.

Method	# Params (%)	VQAv2	GQA	VisWiz	SQA	VQA^{T}	POPE	MMBench	Avg.
Full FT	100	78.5	61.9	50.0	66.8	58.2	85.9	64.3	66.5
LoRA [14]	4.61	**79.1**	62.9	47.8	68.4	58.2	86.4	**66.1**	66.9
DoRA [25]	4.63	78.6	**62.9**	52.2	**69.9**	57.0	**87.2**	**66.1**	**67.6**
FuRA (ours)	**1.37**	78.7	62.7	**54.4**	67.3	**58.1**	86.6	64.5	**67.6**

Takeaways. FuRA matches DoRA's 7-task average while training **3.4**$\times$ fewer parameters (1.37% vs. 4.63% of 7.06B model parameters). FuRA wins VisWiz (+2.2) and TextVQA (+1.1), ties POPE within 0.6, and trails on the reasoning-heavy SQA (-2.6) and MMBench (-1.6) splits. The wall-clock cost is also lower: at the same effective batch and step count, FuRA trains in 21h11m on a single H100 vs. DoRA's 35h46m (**41**% faster).

Param count derivation (1.37%)**.** For each linear with `in_features` $= n \cdot b$, the trainable count under the default FuRA corner is `btt_r` + `btt_s` $= nb^2 + nb =$ `in_features` $\cdot\,(b{+}1)$, while the dense `btt_l` $\in \mathbb{R}^{\texttt{in\_features}\times\texttt{out\_features}}$ core stays frozen. With 32 LLaMA-2-7B blocks $\times$ 7 projections per block, `in_features` $\in \{4096, 11008\}$ factor as $(64, 64)$ and $(86, 128)$ respectively, giving $32 \cdot (6 \cdot 4096 \cdot 65 + 11008 \cdot 129) \approx 96.6$M trainable parameters out of LLaVA-1.5-7B's 7.06B total ($\approx$ 95.4M from `btt_r` plus 1.1M from `btt_s`, with embeddings, lm_head, norms, the CLIP vision tower, and the MLP projector all kept frozen): $96.6/7060 = 1.37\%$.

D.6 Ablation details

The body of Section 6.5 reports only the SFT 8-task average and the MATH-500/AMC23 RL columns. Table 19 below give the corresponding per-task SFT scores and Table 19 adds the AIME-24/AIME-25 RL columns for both ablation slices. Color code matches the body: **blue** = frozen, **red+underbar** = trainable.

Table 18: FuRA PEFT design corners, full SFT per-task results (LLaMA-3-8B Commonsense-170K).

Group	Notation	Train. %	BoolQ	PIQA	SIQA	HSwag	Wino	ARCe	ARCc	OBQA	Avg
Full	Full FT W	100	75.40	88.00	81.80	96.50	89.30	93.10	83.00	86.00	86.64
	FuRA $L\,S\,R$	100	76.10	**90.80**	82.90	**96.80**	89.30	**94.40**	**85.20**	88.80	**88.04**
PEFT (m=1)	$L\,S\,R$ (default)	1.46	**76.6**	89.9	**83.2**	**96.8**	**89.7**	**93.6**	**84.1**	**89.4**	**87.91**
	$(LS)\,R$	1.46	74.0	**90.8**	81.9	95.9	89.4	93.4	84.0	87.6	87.12
	$L\,(SR)$	1.46	74.8	76.6	82.2	95.4	86.4	90.9	81.6	84.8	84.09
PEFT (n=1)	$L\,S\,R$	1.46	75.0	89.5	81.9	96.2	88.1	93.2	83.0	87.4	86.79
	$(LS)\,R$	1.46	67.0	85.6	80.6	93.5	86.3	90.2	78.4	84.8	83.30
	$L\,(SR)$	1.46	66.3	85.6	79.5	94.7	86.1	90.5	80.5	86.8	83.75

Table 19: FuRA PEFT design corners, full RL benchmarks (Qwen3-1.7B GRPO math).

Group	Notation	Train. %	MATH-500	AMC23	AIME-24	AIME-25
Full	Full FT W	100	**63.6**	47.5	13.8	15.4
	FuRA $L\,S\,R$	100	62.4	**50.0**	**17.1**	**18.3**
PEFT (m=1)	$L\,S\,R$ (default)	1.46	**63.6**	**57.5**	**15.0**	**17.5**
	$(LS)\,R$	1.46	61.4	55.0	10.0	12.9
	$L\,(SR)$	1.46	62.8	47.5	12.5	16.7
PEFT (n=1)	$L\,S\,R$	1.46	61.2	52.5	9.6	13.8
	$(LS)\,R$	1.46	61.4	42.5	12.9	13.8
	$L\,(SR)$	1.46	62.2	42.5	9.6	8.3

Shape factorization ablation

Holding the s-placement recipe fixed at the headline default ***L S R***, this sub-ablation varies how the input dimension d_{in} is split into block sizes (n, b) with $n \cdot b = d_{\text{in}}$. We evaluate three regimes on LLaMA-3-8B Commonsense-170K (1 epoch, $\eta = 2{\times}10^{-4}$, seed 43) covering the full balanced-to-extreme axis.

Table 20: Shape factorization ablation: how the input dimension $d_{\text{in}} = n \cdot b$ is split into BlockTT block sizes. LLaMA-3-8B, Commonsense-170K, 1 epoch, $\eta = 2{\times}10^{-4}$, seed 43, default s-placement ***L S R***. **Bold** = column-best.

Shape variant	Train. %	BoolQ	PIQA	SIQA	HSwag	Wino	ARCe	ARCc	OBQA	Avg
Default ($b{\approx}\sqrt{d_{\text{in}}}$)	1.46	**75.70**	**90.80**	**83.80**	**96.80**	**88.70**	**93.70**	**84.60**	**90.40**	**88.06**
Unbalanced (b=8)	0.03	68.00	86.20	74.10	91.20	75.70	90.20	77.10	77.40	79.99
Extreme (b=1, n=d_{in})	0.14	35.80	47.70	32.90	24.50	47.40	25.30	24.20	27.60	33.18

Concretely on LLaMA-3-8B: Default uses $(n{=}32, b{=}128)$ for $Q/K/V/O$ (head-aligned, n=num_heads, b=head_dim), $(n{=}64, b{=}64)$ for Gate/Up, and $(n{=}112, b{=}128)$ for Down (closest factor pair). Unbalanced uses $(n{=}512, b{=}8)$ for $Q/K/V/O$ and Gate/Up, and $(n{=}1792, b{=}8)$ for Down. Extreme uses $b{=}1, n{=}d_{\text{in}}$ on every module.

A more balanced shape factorization is preferred: pushing b toward 1 (large n) hurts model quality: Avg drops from 88.06 (balanced) to 79.99 (b=8) to 33.18 (b=1, near chance level on most tasks). Under ***L S R*** the trainable count is dominated by $|\mathbf{R}| = r \cdot d_{\text{in}}$ with effective rank $r = \min(a, b)$; shrinking b shrinks both the achievable rank and the small-core size, so very small b is a double penalty on representational capacity *and* trainable budget. The Extreme row ends up with more trainable parameters than the Unbalanced row (0.14% vs. 0.03%) because the separately-held ***S*** vector dominates the parameter count once $r{=}1$, yet model quality is worse.

E Extended proof

E.1 Properties of the energy ratio (1)

In the following lemma, we summarize the properties of the energy ratio (1) to support our observations on the spectral analysis for the Full FT.

Lemma E.1 (Properties of the Energy Ratio)**.** *Let $\mathbf{U} \in \mathbb{R}^{d_{\text{out}} \times k}$ have orthonormal columns. Then:*

(i) (Boundedness and geometric interpretation). *For any $\mathbf{M} \in \mathbb{R}^{d_{\text{out}} \times d_{\text{in}}}$ with $\|\mathbf{M}\|_F > 0$,*

$$0 \;\le\; \rho(\mathbf{M};\, \mathbf{U}) \;\le\; 1\,.$$

Moreover:

- *$\rho(\mathbf{M}; \mathbf{U}) = 1$ if and only if $\mathrm{col}(\mathbf{M}) \subseteq \mathrm{col}(\mathbf{U})$, i.e., every column of $\mathbf{M}$ lies in the column space of $\mathbf{U}$.*
- *$\rho(\mathbf{M}; \mathbf{U}) = 0$ if and only if $\mathrm{col}(\mathbf{M}) \perp \mathrm{col}(\mathbf{U})$, i.e., every column of $\mathbf{M}$ is orthogonal to the column space of $\mathbf{U}$.*
- *$\rho(\mathbf{M}; \mathbf{U}) \in (0, 1)$ if and only if $\mathbf{M}$ spreads its energy between $\mathrm{col}(\mathbf{U})$ and its orthogonal complement.*

(ii) (Gaussian baseline). *If $\mathbf{G} \in \mathbb{R}^{d_{\text{out}} \times d_{\text{in}}}$ has i.i.d. entries with zero mean and variance $\sigma^2 > 0$, then*

$$\mathbb{E}\big[\rho(\mathbf{G};\, \mathbf{U})\big] \;=\; \frac{k}{d_{\text{out}}}\,. \tag{6}$$

Proof. **Part (i).**

Write $\mathbf{M} = [\mathbf{m}_1, \ldots, \mathbf{m}_{d_{\text{in}}}]$ column-wise. Since $\mathbf{U}$ has orthonormal columns, $\mathbf{P} = \mathbf{U}\mathbf{U}^\top$ is the orthogonal projector onto $\mathrm{col}(\mathbf{U})$. Note that $\|\mathbf{U}^\top \mathbf{m}_j\|^2 = \|\mathbf{P}\mathbf{m}_j\|^2$ for each column. We decompose each column as

$$\mathbf{m}_j = \mathbf{P}\mathbf{m}_j + (\mathbf{I} - \mathbf{P})\mathbf{m}_j\,,$$

where the two components are orthogonal. By the Pythagorean theorem:

$$\|\mathbf{m}_j\|^2 = \|\mathbf{P}\mathbf{m}_j\|^2 + \|(\mathbf{I}-\mathbf{P})\mathbf{m}_j\|^2 .$$

Summing over columns:

$$\|\mathbf{M}\|_F^2 = \|\mathbf{U}^\top\mathbf{M}\|_F^2 + \|(\mathbf{I}-\mathbf{P})\mathbf{M}\|_F^2 .$$

Since both terms are non-negative and their sum equals $\|\mathbf{M}\|_F^2 > 0$, we have $0 \le \|\mathbf{U}^\top\mathbf{M}\|_F^2 \le \|\mathbf{M}\|_F^2$, giving $\rho(\mathbf{M};\mathbf{U}) \in [0,1]$.

Case $\rho = 1$: $\rho(\mathbf{M};\mathbf{U}) = 1$ iff $\|(\mathbf{I}-\mathbf{P})\mathbf{M}\|_F^2 = 0$ iff $(\mathbf{I}-\mathbf{P})\mathbf{m}_j = \mathbf{0}$ for all j iff $\mathbf{m}_j \in \mathrm{col}(\mathbf{U})$ for all j iff $\mathrm{col}(\mathbf{M}) \subseteq \mathrm{col}(\mathbf{U})$.

Case $\rho = 0$: $\rho(\mathbf{M};\mathbf{U}) = 0$ iff $\|\mathbf{U}^\top\mathbf{M}\|_F^2 = 0$ iff $\mathbf{U}^\top\mathbf{m}_j = \mathbf{0}$ for all j iff $\mathbf{m}_j \perp \mathrm{col}(\mathbf{U})$ for all j iff $\mathrm{col}(\mathbf{M}) \perp \mathrm{col}(\mathbf{U})$.

Case $\rho \in (0,1)$: This is the complement of the above two cases: $\mathbf{M}$ has nonzero projection onto both $\mathrm{col}(\mathbf{U})$ and $\mathrm{col}(\mathbf{U})^\perp$, meaning its energy is distributed between the two subspaces.

Part (ii).

Let $\mathbf{G}$ have i.i.d. entries with mean zero and variance σ^2. Denote the columns of $\mathbf{U}$ by $\mathbf{u}_1, \dots, \mathbf{u}_k$.

Numerator:

$$\mathbb{E}\big[\|\mathbf{U}^\top\mathbf{G}\|_F^2\big] = \sum_{j=1}^{d_{\text{in}}}\sum_{i=1}^{k}\mathbb{E}\big[(\mathbf{u}_i^\top\mathbf{g}_j)^2\big] . \tag{7}$$

Each $\mathbf{u}_i^\top\mathbf{g}_j = \sum_{\ell=1}^{d_{\text{out}}}(\mathbf{u}_i)_\ell\, G_{\ell j}$ is a linear combination of independent zero-mean entries, so

$$\mathbb{E}[(\mathbf{u}_i^\top\mathbf{g}_j)^2] = \mathrm{Var}(\mathbf{u}_i^\top\mathbf{g}_j) = \sigma^2\sum_{\ell=1}^{d_{\text{out}}}(\mathbf{u}_i)_\ell^2 = \sigma^2\|\mathbf{u}_i\|^2 = \sigma^2 .$$

Therefore $\mathbb{E}[\|\mathbf{U}^\top\mathbf{G}\|_F^2] = k\cdot d_{\text{in}}\cdot\sigma^2$.

Denominator:

$$\mathbb{E}\big[\|\mathbf{G}\|_F^2\big] = d_{\text{out}}\cdot d_{\text{in}}\cdot\sigma^2 .$$

Computing $\mathbb{E}[\rho(\mathbf{G};\mathbf{U})]$: Let $\mathbf{P} = \mathbf{U}\mathbf{U}^\top$ as in Part (i). By the Pythagorean decomposition established there:

$$\|\mathbf{G}\|_F^2 = \|\mathbf{U}^\top\mathbf{G}\|_F^2 + \|(\mathbf{I}-\mathbf{P})\mathbf{G}\|_F^2 .$$

Therefore

$$\rho(\mathbf{G};\mathbf{U}) = \frac{\|\mathbf{U}^\top\mathbf{G}\|_F^2}{\|\mathbf{U}^\top\mathbf{G}\|_F^2 + \|(\mathbf{I}-\mathbf{P})\mathbf{G}\|_F^2} .$$

Extend $\mathbf{U}$ to a full orthonormal basis $\bar{\mathbf{U}} = [\mathbf{U} \mid \mathbf{U}_\perp] \in \mathbb{R}^{d_{\text{out}}\times d_{\text{out}}}$, where $\mathbf{U}_\perp \in \mathbb{R}^{d_{\text{out}}\times(d_{\text{out}}-k)}$. Define $\mathbf{H} = \bar{\mathbf{U}}^\top\mathbf{G}$. Since $\bar{\mathbf{U}}$ is orthogonal and $\mathbf{G}$ has i.i.d. $\mathcal{N}(0,\sigma^2)$ entries, $\mathbf{H}$ also has i.i.d. $\mathcal{N}(0,\sigma^2)$ entries (orthogonal transformations preserve the i.i.d. Gaussian distribution). The first k rows of $\mathbf{H}$ equal $\mathbf{U}^\top\mathbf{G}$ and the remaining $d_{\text{out}}-k$ rows equal $\mathbf{U}_\perp^\top\mathbf{G} = (\mathbf{I}-\mathbf{P})\mathbf{G}$ expressed in the $\mathbf{U}_\perp$ basis. Thus:

$$\|\mathbf{U}^\top\mathbf{G}\|_F^2 = \sum_{i=1}^{k}\sum_{j=1}^{d_{\text{in}}}H_{ij}^2 , \qquad \|\mathbf{G}\|_F^2 = \sum_{i=1}^{d_{\text{out}}}\sum_{j=1}^{d_{\text{in}}}H_{ij}^2 .$$

Let $X_\ell = H_\ell^2/\sigma^2$ for each entry ℓ. All X_ℓ are i.i.d. $\chi^2(1)$. Then:

$$\rho(\mathbf{G};\mathbf{U}) = \frac{\sum_{\ell=1}^{k\cdot d_{\text{in}}}X_\ell}{\sum_{\ell=1}^{d_{\text{out}}\cdot d_{\text{in}}}X_\ell} .$$

The numerator is a partial sum of the denominator, and all terms are i.i.d. By symmetry, each term contributes equally in expectation to the total sum. Therefore:

$$\mathbb{E}\left[\frac{\sum_{\ell=1}^{k\cdot d_{\text{in}}}X_\ell}{\sum_{\ell=1}^{d_{\text{out}}\cdot d_{\text{in}}}X_\ell}\right] = \frac{k\cdot d_{\text{in}}}{d_{\text{out}}\cdot d_{\text{in}}} = \frac{k}{d_{\text{out}}} ,$$

where the first equality follows from the exchangeability of the $\{X_\ell\}$: for i.i.d. positive random variables $X_1,\dots,X_N$, the expectation $\mathbb{E}[X_1/(X_1+\cdots+X_N)] = 1/N$ by symmetry, so $\mathbb{E}[(\sum_{\ell=1}^{n}X_\ell)/(\sum_{\ell=1}^{N}X_\ell)] = n/N$. □

E.2 Claim 1: Full-rank Update Capacity

We follow [54] to define the rank capacity of an weight update $\Delta W = f_\theta(W)$ as

$$\mathcal{R}(f_\theta; W) := \max_\theta \operatorname{rank}(f_\theta(W)) \; - \; \min_\theta \operatorname{rank}(f_\theta(W)) \tag{8}$$

which describes the range of matrix ranks the weight update can achieve.

Lemma E.2 (Rank capacity of slice-wise spectral-core adaptation)**.** *Let $\mathbf{W} \in \mathbb{R}^{m\times n}$ be a full-rank pretrained weight matrix with $\operatorname{rank}(\mathbf{W}) = \min(m, n)$.*

(Column-sliced form). *Suppose $n = n_0 n_1$, and write*

$$\mathbf{W} = [\mathbf{W}_1, \ldots, \mathbf{W}_{n_1}], \quad \mathbf{W}_i \in \mathbb{R}^{m\times n_0}.$$

For each slice, consider a full-rank factorization

$$\mathbf{W}_i = \mathbf{P}_i \mathbf{Q}_i, \quad \mathbf{P}_i \in \mathbb{R}^{m\times r_c}, \; \mathbf{Q}_i \in \mathbb{R}^{r_c\times n_0}, \quad r_c = \min(m, n_0).$$

Fix $\mathbf{P}_i$ and tune $\{\mathbf{Q}_i\}$. Define

$$f_\theta(\mathbf{W}) = [\mathbf{P}_1 \widetilde{\mathbf{Q}}_1, \ldots, \mathbf{P}_{n_1} \widetilde{\mathbf{Q}}_{n_1}].$$

Then

$$\mathcal{R}(f_\theta; \mathbf{W}) = \operatorname{rank}([\mathbf{P}_1, \ldots, \mathbf{P}_{n_1}]).$$

In particular, if $m \le n$, then $\mathcal{R}(f_\theta; \mathbf{W}) = m$.

(Row-sliced form). *Suppose $m = m_0 m_1$, and write*

$$\mathbf{W} = \begin{bmatrix} \mathbf{W}_1 \\ \vdots \\ \mathbf{W}_{m_1} \end{bmatrix}, \quad \mathbf{W}_j \in \mathbb{R}^{m_0\times n}.$$

For each slice, consider a full-rank factorization

$$\mathbf{W}_j = \mathbf{U}_j \mathbf{V}_j, \quad \mathbf{U}_j \in \mathbb{R}^{m_0\times r_r}, \; \mathbf{V}_j \in \mathbb{R}^{r_r\times n}, \quad r_r = \min(m_0, n).$$

Fix $\mathbf{V}_j$ and tune $\{\mathbf{U}_j\}$. Define

$$g_\phi(\mathbf{W}) = \begin{bmatrix} \widetilde{\mathbf{U}}_1 \mathbf{V}_1 \\ \vdots \\ \widetilde{\mathbf{U}}_{m_1} \mathbf{V}_{m_1} \end{bmatrix}.$$

Then

$$\mathcal{R}(g_\phi; \mathbf{W}) = \operatorname{rank}\big(\begin{bmatrix} \mathbf{V}_1^\top & \cdots & \mathbf{V}_{m_1}^\top \end{bmatrix}\big).$$

In particular, if $n \le m$, then $\mathcal{R}(g_\phi; \mathbf{W}) = n$.

Proof. We first prove the column-sliced case. By construction,

$$f_\theta(\mathbf{W}) = [\mathbf{P}_1 \widetilde{\mathbf{Q}}_1, \ldots, \mathbf{P}_{n_1} \widetilde{\mathbf{Q}}_{n_1}].$$

For each i, every column of $\mathbf{P}_i \widetilde{\mathbf{Q}}_i$ lies in $\operatorname{col}(\mathbf{P}_i)$. Hence

$$\operatorname{col}(f_\theta(\mathbf{W})) \subseteq \operatorname{col}([\mathbf{P}_1, \ldots, \mathbf{P}_{n_1}]),$$

which implies

$$\operatorname{rank}(f_\theta(\mathbf{W})) \le \operatorname{rank}([\mathbf{P}_1, \ldots, \mathbf{P}_{n_1}]).$$

Thus

$$\max_\theta \operatorname{rank}(f_\theta(\mathbf{W})) \le \operatorname{rank}([\mathbf{P}_1, \ldots, \mathbf{P}_{n_1}]).$$

On the other hand, since each $\widetilde{\mathbf{Q}}_i \in \mathbb{R}^{r_c\times n_0}$ is unconstrained, we can choose $\widetilde{\mathbf{Q}}_i$ so that the columns of $\mathbf{P}_i \widetilde{\mathbf{Q}}_i$ span any subspace contained in $\operatorname{col}(\mathbf{P}_i)$. Aggregating across all slices, we can realize any column set in

$$\operatorname{col}([\mathbf{P}_1, \ldots, \mathbf{P}_{n_1}]).$$

Therefore,

$$\max_{\theta} \operatorname{rank}(f_\theta(\mathbf{W})) = \operatorname{rank}([\mathbf{P}_1, \ldots, \mathbf{P}_{n_1}]).$$

Moreover, by setting $\widetilde{\mathbf{Q}}_i = \mathbf{0}$ for all i, we obtain $f_\theta(\mathbf{W}) = \mathbf{0}$, hence

$$\min_{\theta} \operatorname{rank}(f_\theta(\mathbf{W})) = 0.$$

Thus,

$$\mathcal{R}(f_\theta; \mathbf{W}) = \operatorname{rank}([\mathbf{P}_1, \ldots, \mathbf{P}_{n_1}]).$$

Now assume $m \leq n$ and $\operatorname{rank}(\mathbf{W}) = m$. Since

$$\mathbf{W} = [\mathbf{P}_1\mathbf{Q}_1, \ldots, \mathbf{P}_{n_1}\mathbf{Q}_{n_1}],$$

all columns of $\mathbf{W}$ lie in $\operatorname{col}([\mathbf{P}_1, \ldots, \mathbf{P}_{n_1}])$, so

$$m = \operatorname{rank}(\mathbf{W}) \leq \operatorname{rank}([\mathbf{P}_1, \ldots, \mathbf{P}_{n_1}]).$$

Since the latter is at most m, we conclude

$$\operatorname{rank}([\mathbf{P}_1, \ldots, \mathbf{P}_{n_1}]) = m,$$

and hence $\mathcal{R}(f_\theta; \mathbf{W}) = m$.

The row-sliced case follows symmetrically. For

$$g_\phi(\mathbf{W}) = \begin{bmatrix} \widetilde{\mathbf{U}}_1\mathbf{V}_1 \\ \vdots \\ \widetilde{\mathbf{U}}_{m_1}\mathbf{V}_{m_1} \end{bmatrix},$$

each row lies in $\operatorname{row}(\mathbf{V}_j)$. Hence

$$\operatorname{row}(g_\phi(\mathbf{W})) \subseteq \operatorname{span}\{\operatorname{row}(\mathbf{V}_j)\}_{j=1}^{m_1},$$

which implies

$$\max_{\phi} \operatorname{rank}(g_\phi(\mathbf{W})) = \operatorname{rank}\left(\begin{bmatrix} \mathbf{V}_1^\top & \cdots & \mathbf{V}_{m_1}^\top \end{bmatrix}\right).$$

Again the minimum rank is 0. If $n \leq m$ and $\operatorname{rank}(\mathbf{W}) = n$, then the row space must have dimension n, yielding

$$\mathcal{R}(g_\phi; \mathbf{W}) = n.$$

□

Constraints of block-wise fixed-subspace adaptation. Although the proposed slice-wise parameterization attains maximal rank capacity, it imposes strong structural constraints on the adapted weight. We characterize these constraints below.

Proposition E.3 (Block-wise fixed-subspace constraint)**.** *Consider the column-sliced parameterization*

$$f_\theta(\mathbf{W}) = [\mathbf{P}_1\widetilde{\mathbf{Q}}_1, \ldots, \mathbf{P}_{n_1}\widetilde{\mathbf{Q}}_{n_1}],$$

with fixed $\{\mathbf{P}_i\}$*. Then for any* θ*:*

$$\operatorname{col}\big(f_\theta(\mathbf{W})_{(:,\mathcal{I}_i)}\big) \subseteq \operatorname{col}(\mathbf{P}_i),$$

where $\mathcal{I}_i$ *indexes columns belonging to slice* i*. Equivalently,*

$$f_\theta(\mathbf{W}) \in \mathcal{S}_{col} := \{[\mathbf{X}_1, \ldots, \mathbf{X}_{n_1}] \mid \operatorname{col}(\mathbf{X}_i) \subseteq \operatorname{col}(\mathbf{P}_i)\}.$$

Proposition E.4 (Row-space constraint (dual form))**.** *For the row-sliced parameterization*

$$g_\phi(\mathbf{W}) = \begin{bmatrix} \widetilde{\mathbf{U}}_1\mathbf{V}_1 \\ \vdots \\ \widetilde{\mathbf{U}}_{m_1}\mathbf{V}_{m_1} \end{bmatrix},$$

we have

$$\operatorname{row}\big(g_\phi(\mathbf{W})_{(\mathcal{J}_j,:)}\big) \subseteq \operatorname{row}(\mathbf{V}_j),$$

and hence

$$g_\phi(\mathbf{W}) \in \mathcal{S}_{row} := \left\{ \begin{bmatrix} \mathbf{Y}_1 \\ \vdots \\ \mathbf{Y}_{m_1} \end{bmatrix} \;\middle|\; \operatorname{row}(\mathbf{Y}_j) \subseteq \operatorname{row}(\mathbf{V}_j) \right\}.$$

Implications. The above propositions show that adaptation is restricted to a *product of subspaces* across slices. This leads to several nontrivial consequences:

(1) No cross-block subspace transfer. Columns in slice i cannot utilize directions in $\mathrm{col}(\mathbf{P}_j)$ for $j \neq i$. Hence global redistribution of information across slices is prohibited.

(2) Restricted perturbation geometry. Let $\Delta\mathbf{W} = f_\theta(\mathbf{W}) - \mathbf{W}$. Then

$$\Delta\mathbf{W} = [\mathbf{P}_1\Delta\mathbf{Q}_1, \ldots, \mathbf{P}_{n_1}\Delta\mathbf{Q}_{n_1}],$$

so each block update lies in a fixed linear subspace. In particular,

$$\Delta\mathbf{W} \in \mathcal{T} := \{[\mathbf{P}_1\mathbf{Z}_1, \ldots, \mathbf{P}_{n_1}\mathbf{Z}_{n_1}]\},$$

which is a strict subset of $\mathbb{R}^{m\times n}$ unless all $\mathrm{col}(\mathbf{P}_i)$ span $\mathbb{R}^m$ individually.

(3) Coupled global rank vs local expressivity. Although the global rank can reach $\mathrm{rank}([\mathbf{P}_1, \ldots, \mathbf{P}_{n_1}])$, each slice individually satisfies

$$\mathrm{rank}(\mathbf{P}_i\widetilde{\mathbf{Q}}_i) \leq r_c.$$

Hence expressivity is locally bottlenecked even when global rank is maximal.

(4) Subspace preservation bias. The adaptation preserves the pretrained column subspaces $\{\mathrm{col}(\mathbf{P}_i)\}$. Therefore, learning is biased toward *reweighting and recombining existing features* rather than creating new directions.

Comparison to LoRA and spectral adapters. Unlike LoRA, which introduces a global low-rank update $\Delta\mathbf{W} = \mathbf{A}\mathbf{B}^\top$ spanning all columns, the above parameterization enforces block-wise independence of subspaces. Compared to spectral adapters that directly modify singular directions, this approach fixes the singular subspaces and only adapts coefficients within them.

When is this favorable? This constraint is beneficial when pretrained subspaces are already well-aligned with downstream tasks, as it: (i) preserves stable directions, (ii) reduces destructive interference, and (iii) improves conditioning by restricting updates to structured subspaces.

E.3 Preconditioner derivations

Let $\mathbf{W}_k = \mathbf{U}_k\mathbf{\Sigma}_k\mathbf{V}_k^\top$ be the SVD of one block. Let $\mathcal{L}$ be the loss, $\mathbf{y} = \mathbf{W}\mathbf{x}$, $\mathbf{g} = \partial\mathcal{L}/\partial\mathbf{y}$. Standard identities give the unconstrained per-block gradient $\partial\mathcal{L}/\partial\mathbf{W}_k = \mathbf{g}\mathbf{x}_k^\top$. We specialize to the four FuRA variants.

Corner 1: output-side training, S merged to frozen R. The parameterization is $\mathbf{W}_k = \mathbf{L}_k\mathbf{R}_k$ with $\mathbf{L}_k$ trainable and $\mathbf{R}_k = \mathrm{diag}(\mathbf{S}_k)\,\mathbf{V}_k^\top$ frozen. The gradient on $\mathbf{L}_k$ is

$$\partial\mathcal{L}/\partial\mathbf{L}_k \;=\; (\mathbf{g}\mathbf{x}_k^\top)\mathbf{R}_k^\top \;=\; \mathbf{g}\mathbf{x}_k^\top\,\mathbf{V}_k\,\mathrm{diag}(\mathbf{S}_k).$$

A GD step $\mathbf{L}_k \leftarrow \mathbf{L}_k - \eta\,\mathbf{g}\mathbf{x}_k^\top\,\mathbf{V}_k\,\mathrm{diag}(\mathbf{S}_k)$ yields

$$\Delta\mathbf{W}_k \;=\; \Delta\mathbf{L}_k\cdot\mathbf{R}_k \;=\; -\eta\,\mathbf{g}\mathbf{x}_k^\top\,\mathbf{V}_k\,\mathrm{diag}(\mathbf{S}_k)^2\,\mathbf{V}_k^\top,$$

the "$\mathrm{diag}(\mathbf{S})^2$-weighted" row of Table 2: $\mathrm{diag}(\mathbf{S}_k)$ enters once via $\mathbf{R}_k^\top$ in the gradient and once via $\mathbf{R}_k$ in the reconstruction.

Corner 2: output-side training, S merged to trainable L. The parameterization is $\mathbf{W}_k = \mathbf{L}_k\mathbf{R}_k$ with $\mathbf{L}_k = \mathbf{U}_k\,\mathrm{diag}(\mathbf{S}_k)$ trainable and $\mathbf{R}_k = \mathbf{V}_k^\top$ frozen. The gradient on $\mathbf{L}_k$ is

$$\partial\mathcal{L}/\partial\mathbf{L}_k \;=\; (\mathbf{g}\mathbf{x}_k^\top)\mathbf{R}_k^\top \;=\; \mathbf{g}\mathbf{x}_k^\top\,\mathbf{V}_k.$$

A GD step yields $\Delta\mathbf{L}_k = -\eta\,\mathbf{g}\mathbf{x}_k^\top\,\mathbf{V}_k$, hence

$$\Delta\mathbf{W}_k \;=\; \Delta\mathbf{L}_k\cdot\mathbf{R}_k \;=\; -\eta\,\mathbf{g}\mathbf{x}_k^\top\,\mathbf{V}_k\mathbf{V}_k^\top,$$

the "fair" projector $\mathbf{V}_k\mathbf{V}_k^\top$ row of Table 2: when $\mathbf{S}$ is absorbed into the trainable side, its scaling is freely re-tuned by the optimizer and drops out of the effective preconditioner, leaving an unweighted projection onto $\mathrm{row}(\mathbf{V}_k^\top)$.

Corner 3 (default): input-side training, S kept separate trainable. Parameterize $\mathbf{W}_k = \mathbf{L}_k \operatorname{diag}(\mathbf{S}_k)\, \mathbf{R}_k$ with $\mathbf{L}_k = \mathbf{U}_k$ frozen and $\mathbf{R}_k = \mathbf{V}_k^\top$, $\mathbf{S}_k$ trainable. The gradient on $\mathbf{R}_k$ is $\partial\mathcal{L}/\partial\mathbf{R}_k = \operatorname{diag}(\mathbf{S}_k)\, \mathbf{U}_k^\top\, \mathbf{g}\mathbf{x}_k^\top$; the gradient on $\mathbf{S}_k$ is the diagonal of $\mathbf{U}_k^\top\, \mathbf{g}\mathbf{x}_k^\top\, \mathbf{V}_k$. A GD step on $\mathbf{R}_k$ gives $\Delta\mathbf{R}_k = -\eta \operatorname{diag}(\mathbf{S}_k)\, \mathbf{U}_k^\top\, \mathbf{g}\mathbf{x}_k^\top$, so the effective weight update from $\mathbf{R}$ is

$$\Delta\mathbf{W}_k\big|_{\mathbf{R}} \;=\; \mathbf{U}_k \operatorname{diag}(\mathbf{S}_k)\, \Delta\mathbf{R}_k \;=\; -\eta\, \mathbf{U}_k \operatorname{diag}(\mathbf{S}_k)^2\, \mathbf{U}_k^\top\, \mathbf{g}\mathbf{x}_k^\top,$$

where $\operatorname{diag}(\mathbf{S}_k)$ enters twice: once from $\mathbf{L}_k \operatorname{diag}(\mathbf{S}_k)$ multiplying $\Delta\mathbf{R}_k$, and once inside $\Delta\mathbf{R}_k$ itself. Combined with the $\mathbf{S}$ update:

$$\Delta\mathbf{W}_k \;=\; -\eta\, \mathbf{U}_k \operatorname{diag}(\mathbf{S}_k)^2\, \mathbf{U}_k^\top\, \mathbf{g}\mathbf{x}_k^\top \;+\; \mathbf{U}_k \operatorname{diag}(\Delta\mathbf{S}_k)\, \mathbf{V}_k^\top,$$

the default preconditioner of Equation (5). The first term combines column-space projection ($\mathbf{U}_k\mathbf{U}_k^\top$, non-trivial since $\mathbf{U}_k$ is rectangular) with singular-value preconditioning ($\operatorname{diag}(\mathbf{S}_k)^2$ scaling).

Corner 4: input-side training, S merged to frozen L. The parameterization is $\mathbf{W}_k = \mathbf{L}_k\mathbf{R}_k$ with $\mathbf{L}_k = \mathbf{U}_k \operatorname{diag}(\mathbf{S}_k)$ frozen and $\mathbf{R}_k = \mathbf{V}_k^\top$ trainable. The gradient on $\mathbf{R}_k$ is

$$\partial\mathcal{L}/\partial\mathbf{R}_k \;=\; \mathbf{L}_k^\top(\mathbf{g}\mathbf{x}_k^\top) \;=\; \operatorname{diag}(\mathbf{S}_k)\, \mathbf{U}_k^\top\, \mathbf{g}\mathbf{x}_k^\top.$$

A GD step yields $\Delta\mathbf{R}_k = -\eta \operatorname{diag}(\mathbf{S}_k)\, \mathbf{U}_k^\top\, \mathbf{g}\mathbf{x}_k^\top$, hence

$$\Delta\mathbf{W}_k \;=\; \mathbf{L}_k \cdot \Delta\mathbf{R}_k \;=\; -\eta\, \mathbf{U}_k \operatorname{diag}(\mathbf{S}_k)^2\, \mathbf{U}_k^\top\, \mathbf{g}\mathbf{x}_k^\top,$$

the principal-biased $\mathbf{U}_k \operatorname{diag}(\mathbf{S}_k)^2\, \mathbf{U}_k^\top$ row of Table 2 (PiSSA-like): $\operatorname{diag}(\mathbf{S}_k)$ enters once via $\mathbf{L}_k^\top$ in the gradient and once via $\mathbf{L}_k$ in the reconstruction, mirroring Corner 1 with the input/output sides swapped. The remaining row of Table 2 (input-side training with $\mathbf{S}$ merged into the trainable $\mathbf{R}$) follows by the same calculation as Corner 2 with the input/output sides swapped, yielding the unweighted projector $\mathbf{U}_k\mathbf{U}_k^\top$.

F Full per-layer sweeps for the motivating figures

This section provides the full per-layer / per-module versions of the panels shown for a single representative weight matrix in Figures 3 and 4 of the body. The body uses layer 15 `q_proj` as a representative slice; the figures here verify that the trends hold across all transformer layers and across the matched set of linear modules.

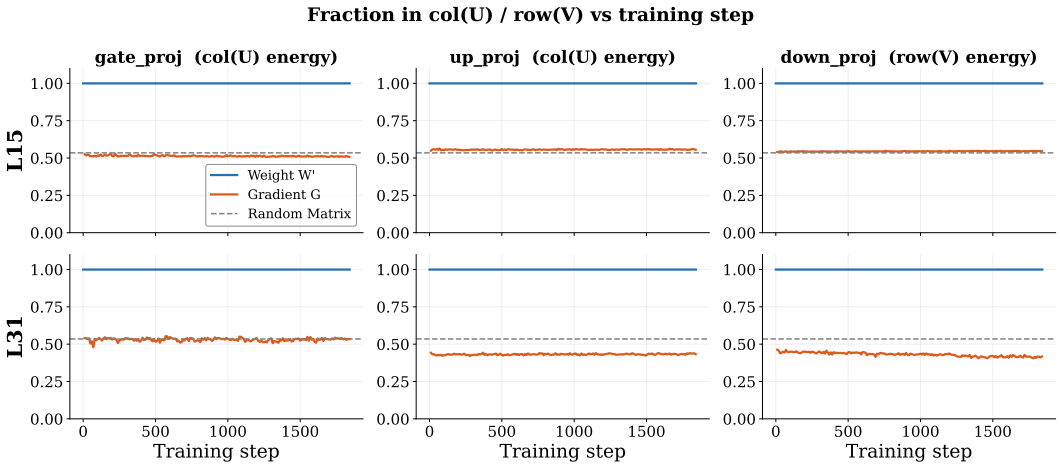

Figure 6: Full per-layer / per-module sweep for Figure 3**(a)**: energy-ratio $\rho(\mathbf{G}_\mathbf{W}; \mathbf{U})$ and $\rho(\mathbf{W}'; \mathbf{U})$ tracked through Full FT training on LLaMA-3-8B / Math-10K. The body figure shows layer 15 `q_proj`; the same pattern (gradient near the random baseline, weight near 1.0) holds across all layers and modules.

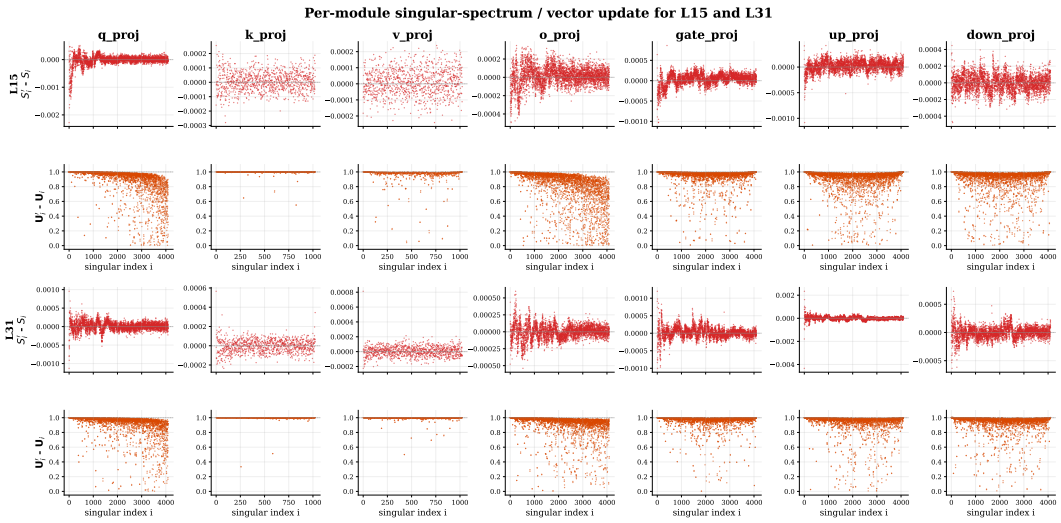

Figure 7: Full per-layer / per-module sweep for Figure 3**(b)**: comparison of singular values and singular vectors of $\mathbf{W}$ versus $\mathbf{W}'$ after Full FT on LLaMA-3-8B / Math-10K. The body figure shows layer 15 `q_proj`; across all layers and modules only a few singular values change significantly while most remain close to their pretrained values.

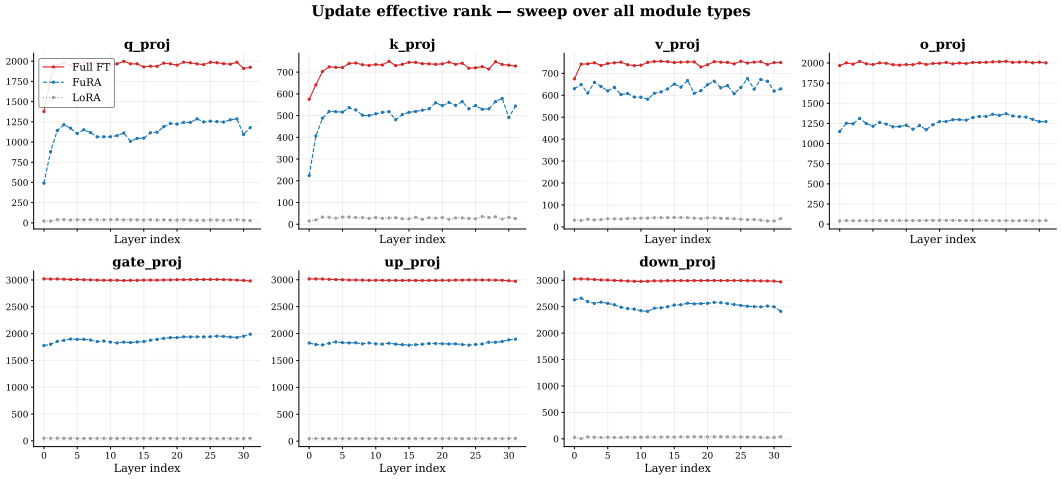

Figure 8: Full per-layer / per-module sweep for Figure 4**(a)**: effective rank of $\Delta\mathbf{W}$ on the Qwen3-1.7B GRPO math-RL task. The body figure shows a representative module; here we report all modules in every transformer layer, confirming that FuRA's effective rank closely tracks Full FT layer-by-layer while remaining well above LoRA.