Geometry-Preserving Orthonormal Initialization for Low-Rank Adaptation in Reinforcement Learning

Anonymous Author(s)
Anonymous Institution
anonymous@email.com

May 15, 2026

Abstract

Low-Rank Adaptation (LoRA) and its variants enable parameter-efficient fine-tuning of large language models under the supervised fine-tuning (SFT) paradigm. However, their efficacy and behavior under Reinforcement Learning with Verifiable Rewards (RLVR) are less well understood. In particular, two structurally initialized LoRA variants, PiSSA and MiLoRA, which outperform standard LoRA under SFT, can underperform standard LoRA under RLVR and may even exhibit training instability. These observations suggest that how to initialize the low-rank matrices in RLVR remains unclear. In this work, we develop a theoretical analysis of LoRA in RLVR, showing that orthonormal initialization achieves the minimal gap between LoRA's outcome and that of full fine-tuning. Guided by this insight, we propose geometry-preserving orthonormal initialization for low-rank adaptation in RLVR, leading to two new variants, LoRA-RLPO and LoRA-RLMO. Experiments on mathematical reasoning benchmarks show that our orthonormal initialization stabilizes RLVR training and outperforms standard LoRA, contrasting with PiSSA and MiLoRA. Finally, our unified analysis also explains why PiSSA and MiLoRA can underperform in RLVR, which may be of independent interest.

1 Introduction

Large language models (LLMs) [Brown et al., 2020, Ouyang et al., 2022, Touvron et al., 2023] are typically pretrained on large-scale dataset via next-token prediction [Brown et al., 2020] and then fine-tuned on relatively smaller datasets to specialize for downstream applications. This paradigm has achieved remarkable success across diverse domains, including mathematical reasoning [Luo et al., 2025a, Azerbayev et al., 2024], code generation [Rozière et al., 2024, Luo et al., 2025b], healthcare [Singhal et al., 2022, Chen et al., 2023], and finance [Wu et al., 2023, Yang et al., 2025]. Because fine-tuning is far more accessible than pretraining a new LLM, it has attracted substantial interest in the community. While fine-tuning all parameters in an LLM ("full fine-tuning") is natural, it is practically highly memory-intensive: fully fine-tuning even a 7B model can require over 100GB of GPU memory [Dettmers et al., 2023]. This high resource demand limits accessibility for practitioners and motivates parameter-efficient fine-tuning (PEFT) methods, which update only a small subset of parameters while keeping the base model frozen [Houlsby et al., 2019, Li and Liang, 2021, Lester et al., 2021]. Among them, Low-Rank Adaptation (LoRA) [Hu et al., 2021] is widely used due to its efficiency and ease of implementation. For any weight matrix $W_0 \in \mathbb{R}^{m\times n}$ in a pretrained model, LoRA parameterizes the update as $\Delta W^{\text{lora}} = BA$ with $B \in \mathbb{R}^{m\times r}$ and $A \in \mathbb{R}^{r\times n}$, where $r \ll \min(m, n)$. This significantly reduces the number of trainable parameters, while still yielding a dense matrix $W_0 + \Delta W^{\text{lora}}$ at inference time.

Meanwhile, supervised fine-tuning (SFT) is a widely used LLM fine-tuning paradigm that trains on high-quality instruction–response pairs. Beyond SFT, reinforcement learning with

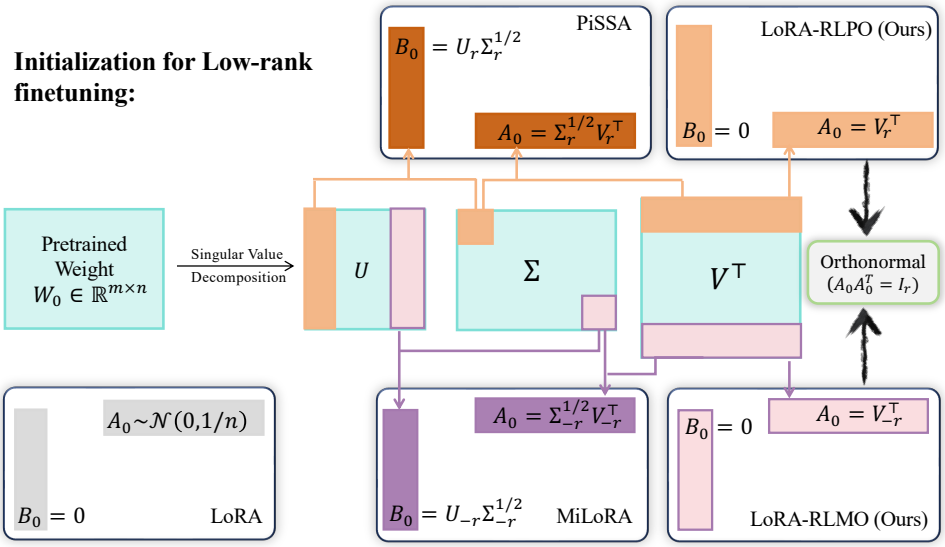

Figure 1: Comparison of LoRA initialization strategies. **LoRA** uses random Gaussian A_0 with $B_0 = 0$. **PiSSA** and **MiLoRA** initialize both adapters from the principal and minor singular components of W_0, respectively, with $B_0 \neq 0$ and singular value scaling. Our proposed methods, LoRA-RLPO and LoRA-RLMO, initialize orthonormal A_0 from the principal and minor right singular vectors with $B_0 = 0$.

verifiable rewards (RLVR) has recently become a pivotal approach and shown effective for widespread tasks such as mathematical reasoning and coding [Guo et al., 2025, Shao et al., 2024]. RLVR uses rule-based feedback (e.g., answer correctness) instead of learned reward models. GRPO [Shao et al., 2024] is a popular optimization method for RLVR. It incorporates a KL penalty between the learned policy and the pretrained reference model [Ziegler et al., 2020], and estimates advantages from a group of samples without requiring a separate critic network.

Compared with SFT, RLVR typically incurs higher memory costs, making LoRA and its memory-efficient variants particularly attractive for RLVR. This overhead stems from the need to keep a reference model in memory to compute the KL divergence [Ziegler et al., 2020, Zhou et al., 2024] and to store multiple sequential responses per prompt for group-based advantage estimation [Shao et al., 2024]. Moreover, LoRA has already shown strong potential for RLVR, matching full fine-tuning in certain cases [Schulman and Lab, 2025]. Despite this progress, the behavior of LoRA and its structural variants under RLVR remains less understood than under SFT, limiting further advances in low-rank adaptation–style fine-tuning for RL.

Particularly, how to initialize the low-rank matrices (B and A) in RLVR starts to become unclear in light of recent observations. Notably, PiSSA and MiLoRA, two LoRA variants that use initializations differing from standard LoRA improve performance and speed up convergence in SFT, can underperform standard LoRA under RLVR and may even exhibit training instability [Yin et al., 2025]. Specifically, both methods initialize the adapter matrices B and A using the singular value decomposition (SVD) of pretrained weights: PiSSA [Meng et al., 2025] initializes along the top-r principal singular directions, whereas MiLoRA [Wang et al., 2025] targets the bottom-r tail directions. This new discrepancy between RLVR and SFT likely reflects their different optimization dynamics. Prior work suggests that, due to the KL constraint, RLVR updates are encouraged to stay close to the reference policy [Wu et al., 2026, Shenfeld et al., 2025], and that RLVR may also prefer off-policy subspaces that differ from those favored by SFT [Zhu et al., 2025]. Consequently, it is no longer guaranteed that LoRA design principles developed for SFT transfer to RLVR, leaving the appropriate initialization and the choice of subspace in RLVR an open question. In this work, we focus on:

What initialization is effective for low-rank adaptation in RLVR fine-tuning?

To this end, we provide a rigorous analysis demonstrating that by initializing $B = 0$ in accordance with standard LoRA, orthonormal initialization for A is potentially optimal and yields superior performance in practice. Our primary contributions are as follows:

- **Orthonormal initialization towards optimal.** To deeply understand behavior of LoRA, we provide a theoretical analysis for LoRA optimization dynamics. The results (see Theorem 5.3) demonstrate that applying orthonormal initialization of A and $B = 0$ minimizes the gap between the results of LoRA and full fine-tuning, making LoRA's outcome closer to that of full fine-tuning.
- **Geometry-preserving orthonormal initialization.** Motivated by the benefits of orthonormal initialization, we propose LoRA-RLPO and LoRA-RLMO, SVD-based initializations for A that not only keep orthonormal, but also preserve geometry information from pretrained weights. We conduct experiments and show that both methods achieve superior performance over standard LoRA in RLVR.
- **Insights into LoRA variants' failures in RLVR.** Focusing on SVD-based methods such as PiSSA and MiLoRA, our analysis framework provides a unified explanation for their instability in RLVR. We reveal that their failures stem from two coupled factors: **subspace geometry**, which accelerates parameter updates along specific directions, and **singular value scaling**, which heavily amplifies update magnitudes. Together, these mechanisms induce aggressive optimization trajectories that rapidly violate the implicit KL constraints, destabilizing training regardless of whether the principal or minor singular directions are targeted.

2 Related Works

Low-Rank Adaptation (LoRA) and its Variants. Among parameter-efficient fine-tuning (PEFT) methods, LoRA [Hu et al., 2021] and its variants have become a popular family, parameterizing weight updates as a product of two low-rank matrices while keeping the base model frozen. Numerous variants have sought to improve upon standard LoRA. AdaLoRA [Zhang et al., 2023] adaptively allocates rank across layers based on importance scores. LoRA+ [Hayou et al., 2024] uses different learning rates for the A and B matrices. DoRA [Liu et al., 2024a] decomposes updates into magnitude and direction components. rsLoRA [Kalajdzievski, 2023] adjusts the scaling factor to stabilize training at higher ranks. VeRA [Kopiczko et al., 2024] shares frozen random matrices across layers to further reduce parameters. Another line of work focuses on improving LoRA initialization beyond the default random scheme [Hu et al., 2021, Hayou et al., 2024]. Among these, SVD-based methods have gained great attention: PiSSA [Meng et al., 2025] initializes adapters using principal singular components, while MiLoRA [Wang et al., 2025] uses minor components. These methods achieve faster convergence and improved performance in supervised fine-tuning. However, recent evaluations show that they underperform standard LoRA and exhibit instability in RLVR [Yin et al., 2025]. Our work addresses this gap through geometry-preserving orthonormal initialization.

Orthonormality in LoRA. Zhu et al. [2024] investigates the asymmetry between LoRA matrices in supervised fine-tuning, showing that A extracts features from inputs while B maps these features to outputs. They find that fixing A to a random orthonormal matrix while training only B achieves better performance than standard LoRA. OLoRA [Büyükakyüz, 2024] uses QR decomposition to initialize both LoRA matrices with orthonormal bases derived from pretrained weights, achieving faster convergence in SFT tasks. OFT [Qiu et al., 2024] and BOFT [Liu et al., 2024b] take a different approach, enforcing weight updates to remain orthogonal throughout training rather than only at initialization. These works are primarily empirical and focus on SFT settings, where the learning dynamics differ from reinforcement learning. Our work provides

the first theoretical explanation for why orthonormal initialization improves LoRA in RLVR: orthonormal initialization enables LoRA to better track full fine-tuning.

LoRA in Reinforcement Learning. While LoRA has been extensively studied in supervised fine-tuning [Hu et al., 2021, Liu et al., 2024a, Kalajdzievski, 2023, Hayou et al., 2024], its behavior in reinforcement learning remains less understood. In practice, LoRA has been widely adopted for memory-efficient RL training [Santacroce et al., 2023, Guo et al., 2025, Shao et al., 2024], enabling PPO and GRPO fine-tuning on consumer hardware. Zhu et al. [2025] provide theoretical analysis showing that RLVR updates favor off-principal directions, in contrast to SFT which targets principal components. This off-principal learning dynamic suggests that methods designed for SFT may not transfer directly to RLVR. Yin et al. [2025] conduct a systematic evaluation of PEFT methods in RLVR, finding that SVD-based initializations (PiSSA, MiLoRA) underperform standard LoRA and exhibit training instability. While their work provides valuable empirical insights, a rigorous theoretical analysis of why these methods underperform remains lacking. Our work provides a unified theoretical framework explaining both failure modes and proposes geometry-preserving orthonormal initialization as a principled solution.

3 Background

In this section, we formalize the fine-tuning problem for LLMs. Consider an LLM parameterized by $\theta = \{W\}$, where W denotes the collection of weight matrices across layers (e.g., fully connected and attention layers). Without loss of generality, we focus on a single weight matrix from a pretrained model, denoted by $W_0 \in \mathbb{R}^{m\times n}$. Full fine-tuning optimizes θ by updating this matrix as $W = W_0 + \Delta W^{\text{full}}$, where the update is unconstrained, i.e., $\Delta W^{\text{full}} \in \mathbb{R}^{m\times n}$, to minimize a task-specific loss $\mathcal{L}(\{W\})$. Full fine-tuning requires storing full gradients and optimizer states, which can become prohibitive for large-scale models.

3.1 Low-Rank Adaptation (LoRA) and Variants

We first review the LoRA algorithm [Hu et al., 2021]. Consider a pretrained weight matrix $W_0 \in \mathbb{R}^{m\times n}$. LoRA parameterizes the weight update as

$$W = W_0 + \Delta W^{\text{lora}}, \qquad \Delta W^{\text{lora}} = BA, \tag{1}$$

where $B \in \mathbb{R}^{m\times r}$ and $A \in \mathbb{R}^{r\times n}$ are low-rank matrices with $r \ll \min\{m, n\}$. Consequently, the optimization of ΔW^{lora} is restricted to a low-rank subspace of $\mathbb{R}^{m\times n}$. A common initialization sets

$$B_0 = 0_{m\times r}, \quad A_0 \sim \mathcal{N}\left(0, \tfrac{1}{n}\right)^{r\times n}. \tag{2}$$

SVD-Based Initialization Variants. Beyond the initialization in (2), many prior works propose alternative initializations for low-rank fine-tuning. We describe two representative SVD-based variants below. Let $W_0 = U\Sigma V^\top$ be the singular value decomposition, where $U \in \mathbb{R}^{m\times k}$, $\Sigma = \text{diag}(\sigma_1, \ldots, \sigma_k)$ with $\sigma_1 \geq \sigma_2 \geq \cdots \geq \sigma_k > 0$, and $V \in \mathbb{R}^{n\times k}$. Here, $k = \text{rank}(W_0)$ (i.e., the number of positive singular values of W_0).

- **PiSSA** [Meng et al., 2025] initializes with top-r principal components as follows:

$$B_0 = U_r\Sigma_r^{1/2}, \quad A_0 = \Sigma_r^{1/2}V_r^\top$$

where U_r and V_r denote the first r columns of U and V, respectively, and $\Sigma_r = \Sigma_{:r,:r}$ is the $r \times r$ diagonal matrix of the top r singular values.

- **MiLoRA** [Wang et al., 2025] initializes with minor components:

$$B_0 = U_{-r}\Sigma_{-r}^{1/2}, \quad A_0 = \Sigma_{-r}^{1/2}V_{-r}^\top,$$

where U_{-r} and V_{-r} denote the last r columns of U and V, respectively, and $\Sigma_{-r} = \Sigma_{-r:,-r:}$ is the $r \times r$ diagonal matrix of the bottom r singular values.

• **OLoRA** [Büyükakyüz, 2024] initializes with top-r principal components as follows:

$$B_0 = U_r, \quad A_0 = V_r^\top$$

where U_r and V_r denote the first r columns of U and V, respectively.

All methods additionally replace each pretrained matrix W_0 with the residual $(W_0 - B_0A_0)$, keep this residual frozen, and only optimize BA thereafter. Equivalently, the weight matrix is parameterized as $W = (W_0 - B_0A_0) + BA$, which guarantees $W = W_0$ at initialization, regardless of which singular components are used.

3.2 Finetuning Paradigms: SFT vs. RLVR

Besides the parameter-update setting, we now introduce two widely used fine-tuning frameworks and their corresponding loss functions.

Supervised Fine-Tuning (SFT) minimizes cross-entropy against ground-truth labels:

$$\mathcal{L}_{\mathrm{SFT}}(\theta) := -\mathbb{E}_{(x,y^*)\sim\mathcal{D}} \left[\log \pi_\theta(y^*|x)\right]$$

SFT imposes no explicit constraint on weight movement. The model is free to drift arbitrarily far from W_0.

Reinforcement Learning with Verifiable Rewards (RLVR) maximizes expected reward subject to a KL penalty:

$$\mathcal{L}_{\mathrm{RLVR}}(\theta) := \mathbb{E}_{x\sim\mathcal{D},\, y\sim\pi_\theta(\cdot|x)} \left[R(x,y)\right] - \beta \cdot \mathrm{KL}(\pi_\theta \| \pi_{\mathrm{ref}}), \tag{3}$$

where $R(x,y) \in \{0,1\}$ equals 1 if y is a valid solution to x and 0 otherwise, π_{ref} is the reference policy, and $\beta > 0$ is the KL penalty coefficient.

RLVR demands conservative updates. A central result in policy optimization [Kakade and Langford, 2002] establishes that policy improvement can be guaranteed only when each update keeps the new policy close to the current one. Large steps can degrade performance because the training signal (advantages estimated under the current policy) becomes unreliable when the policies diverge.

This conservative-update principle has been instantiated in three different ways by modern RL algorithms:

Trust-region constraint (TRPO). Schulman et al. [2015] directly constrains the KL divergence between consecutive policies:

$$\max_\theta \ \mathcal{L}_{\theta_{\mathrm{old}}}(\theta) \qquad \text{s.t.} \qquad \bar{D}_{\mathrm{KL}}(\theta_{\mathrm{old}}, \theta) \ \le \ \delta, \tag{4}$$

where $\mathcal{L}_{\theta_{\mathrm{old}}}(\theta) = \mathbb{E}_t \left[\frac{\pi_\theta(a_t|s_t)}{\pi_{\theta_{\mathrm{old}}}(a_t|s_t)} A_t\right]$ is the importance-weighted surrogate. Here, t denotes the time step, s_t is the state, a_t is the action, and A_t is the advantage estimate at time t. $\bar{D}_{\mathrm{KL}}$ is the average KL divergence over states. The trust-region radius δ directly limits how far the policy can move.

Clipped surrogate (PPO). Schulman et al. [2017] approximate the trust-region constraint by clipping the probability ratio $r_t(\theta) = \pi_\theta(a_t|s_t)/\pi_{\theta_{\mathrm{old}}}(a_t|s_t)$:

$$\mathcal{L}^{\mathrm{CLIP}}(\theta) \ = \ \mathbb{E}_t\Big[\min\big(r_t(\theta)\, A_t,\ \mathrm{clip}\big(r_t(\theta),\, 1-\epsilon,\, 1+\epsilon\big)\, A_t\big)\Big]. \tag{5}$$

The clipping range $[1-\epsilon,\, 1+\epsilon]$ prevents the ratio from deviating too far from 1, limiting how much the policy can change in a single update.

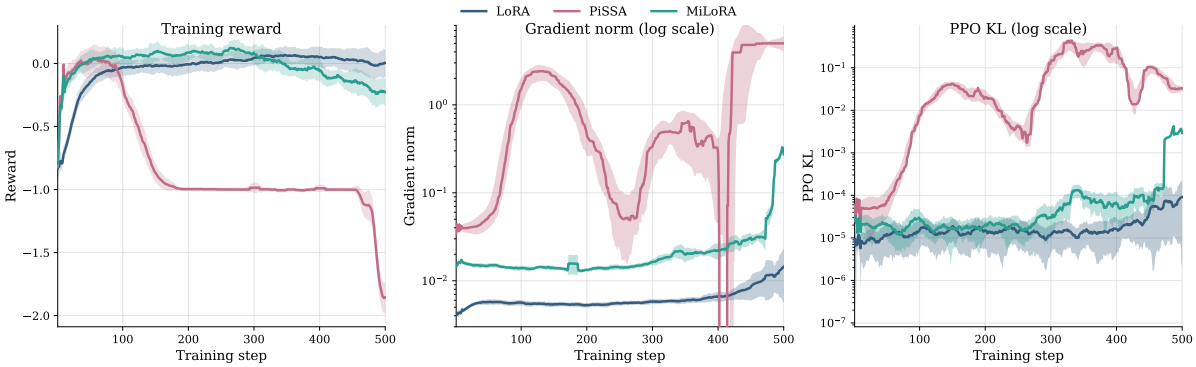

Figure 2: Training dynamics on DAPO-MATH. Left: Training reward comparison. Middle: Gradient norm during training. Right: KL divergence during training. Both PiSSA and MiLoRA exhibit training reward collapse, higher gradient norm and KL divergence than standard LoRA.

KL penalty (GRPO). GRPO [Shao et al., 2024] enforces conservatism through the KL penalty term $\beta \cdot \mathrm{KL}(\pi_\theta \| \pi_{\text{ref}})$ in the objective (3), which penalizes the learned policy for deviating from the reference model.

Despite different implementations, all three mechanisms enforce the same principle: each policy update must remain small enough that the training signal stays reliable. Any initialization that causes updates to exceed this implicit budget risks violating the conservative-update requirement and destabilizing training.

4 Why SVD-Based Initialization Variants Fail

While both PiSSA and MiLoRA outperform standard LoRA in supervised fine-tuning, Figure 2 shows that, in RLVR, both methods underperform standard LoRA and even exhibit obvious collapses. Moreover, in terms of training stability, Figure 2 indicates that both incur substantially higher cumulative KL divergence than standard LoRA throughout training, suggesting markedly unstable behavior.

These observations raise two natural questions: why do these effective LoRA initialization principles (e.g., PiSSA and MiLoRA) break down in RLVR?

KL Constraint and Weight Updates. To understand why SVD-Based Initialization Variants exhibit unstable training, we recall a key result from Zhu et al. [2025] that connects KL divergence to weight update magnitude.

Theorem 4.1 (KL Constraint Implies Weight Bound [Zhu et al., 2025, Gate I])**.** *Assume* $\log \pi_\theta$ *is* C^3*, and let* $F(\theta)$ *denote the Fisher information matrix. Suppose a single-step update* $\theta^+ = \theta + \Delta$ *satisfies* $D_{\mathrm{KL}}(\pi_{\theta^+} \| \pi_\theta) \le K$ *and that, on the update subspace,* $F(\theta) \succeq \mu I$ *for some* $\mu > 0$*. Then, for* K *sufficiently small and any weight-matrix block* $W \subset \theta$*,*

$$\|\Delta W\|_F \;\le\; \sqrt{\frac{2K}{\mu}}\,\big(1 + o(1)\big).$$

This result shows that RLVR training imposes an implicit *KL leash*: the KL penalty constrains each update to remain close to the current policy, which in turn bounds the magnitude of weight changes. When this constraint is violated, training becomes unstable.

Geometry-Informed Initialization Accelerates Updates. By analyzing the post-training energy concentration, we find that initialization intrinsically governs the optimization trajectory rather than merely defining the starting weights. As illustrated in Figure 3, geometry-informed

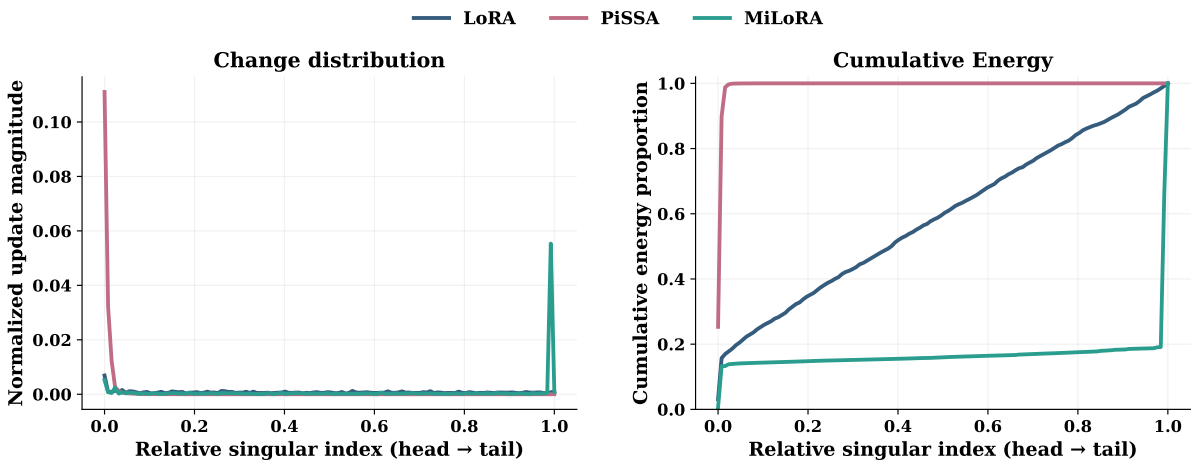

Figure 3: **SVD-based change distribution and cumulative energy proportion for LoRA, PiSSA, and MiLoRA after training.** The left panel shows the normalized distribution of update magnitude across relative singular directions, while the right panel shows the cumulative energy of the update. Results are averaged over `self_attn.q_proj` and `self_attn.o_proj` in Transformer layers 0, 14, and 27. The different patterns show that initialization shapes not only the starting point of optimization, but also the spectral structure of the subsequent updates.

initializations channel updates into highly concentrated singular directions, inducing aggressively amplified weight changes. This intensified dynamic directly drives the rapid KL growth observed under RLVR.

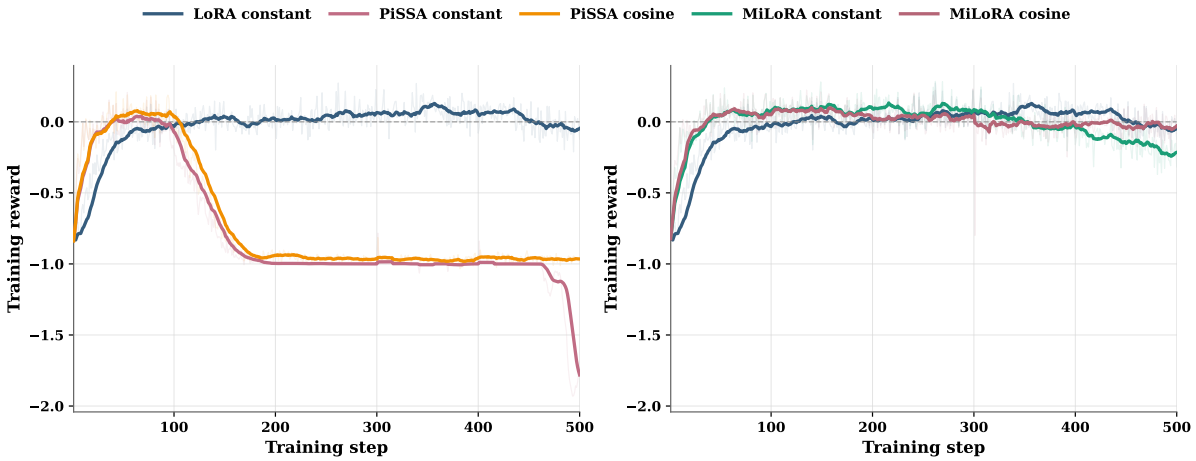

Figure 4: **Ablation on learning-rate decay for PiSSA and MiLoRA under RLVR training.** The left panel compares PiSSA with a constant learning rate and a cosine-decay schedule, together with the LoRA constant-learning-rate baseline; the right panel shows the analogous comparison for MiLoRA. Both methods benefit from slower optimization, confirming that enlarged effective updates are a primary source of instability in RLVR training. Nevertheless, while MiLoRA with learning-rate decay approaches the stable behavior of standard LoRA, PiSSA still exhibits significant late-stage collapse, pointing to an inherently more hazardous optimization direction.

To further empirically verify whether this excessive update magnitude is the primary catalyst for instability, we conduct an ablation study on learning-rate decay, with results detailed in Figure 4. The findings confirm that all methods benefit from slower optimization, underscoring that enlarged effective updates fundamentally destabilize RLVR training. Notably, Figure 4 further reveals that while MiLoRA approaches the stable behavior of standard LoRA under a

reduced learning rate, PiSSA still suffers from severe collapse in the later phases of training. This indicates an inherently hazardous optimization landscape for PiSSA: once optimization is accelerated along these geometry-informed paths, crossing the safe KL boundary becomes almost inevitable.

Singular Value Scaling Destabilizes Training. Beyond the geometric properties of the selected subspace, the scaling of singular values constitutes another critical dimension influencing optimization stability. To disentangle the impact of singular value scaling from the choice of subspace, we utilize **OLoRA** as a controlled baseline. Since both PiSSA and OLoRA target the identical principal subspace of W_0, any divergence in their stability can be attributed specifically to the **singular value scaling** effect inherent in PiSSA. The following theorem formalizes this mechanism by quantifying how PiSSA's initialization leads to excessive update magnitudes.

Theorem 4.2 (PiSSA Gradient Amplification)**.** *The first-step weight updates satisfy:*

$$\frac{\|\Delta W_1^{PiSSA}\|_F}{\|\Delta W_1^{OLoRA}\|_F} \geq \sigma_r, \tag{6}$$

where σ_r is the r-th largest singular value of W_0.

The proof is postponed to Appendix C. For pretrained LLMs, singular values typically exhibit a heavy-tailed distribution where $\sigma_r \gg 1$ for moderate rank r. Consequently, Theorem 4.2 implies that PiSSA inherently scales up weight updates by a substantial factor compared to OLoRA. Combined with our prior analysis of the KL leash (Theorem 4.1), this provides a complete picture of PiSSA's instability: the singular value amplification directly intensifies the update trajectory along already accelerated geometric directions, rapidly violating the implicit KL budget.

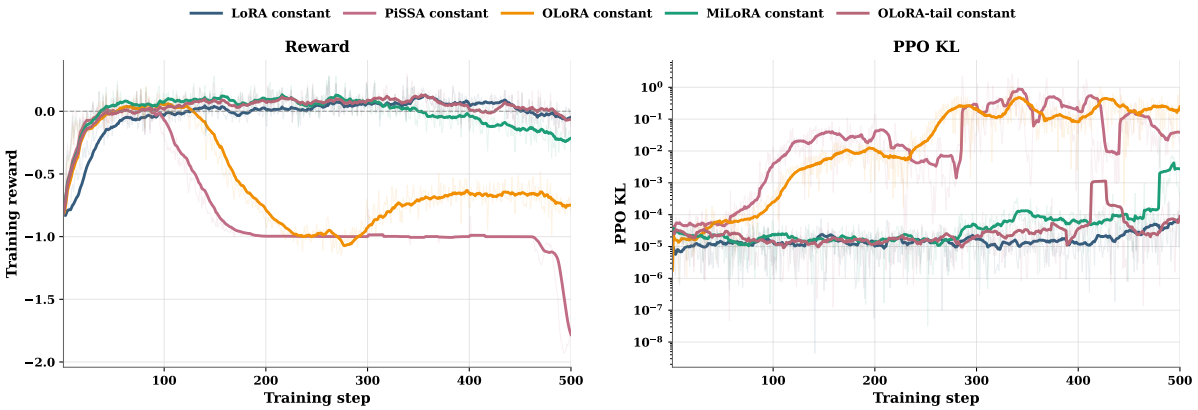

Figure 5: **Singular value scaling exacerbates instability beyond subspace selection.** The left panel shows training reward and the right panel shows PPO KL for LoRA, PiSSA, OLoRA, MiLoRA, and OLoRA-tail under a constant learning rate. Comparing methods within identical subspaces (PiSSA vs. OLoRA for the top; MiLoRA vs. OLoRA-tail for the tail) reveals that removing singular value scaling consistently mitigates collapse and stabilizes training. These controlled comparisons isolate scaling as a distinct destabilizing factor, independent of the chosen geometry.

To rigorously verify that this destabilizing scaling effect is not unique to the principal subspace, we extend our ablation to the minor directions by introducing **OLoRA-tail**. As a counterpart to MiLoRA, OLoRA-tail targets the same tail singular subspace without the singular value scaling:

$$B_0 = U_{-r}, \quad A_0 = V_{-r}^\top,$$

where U_{-r} and V_{-r} denote the last r columns of U and V, respectively. As demonstrated in Figure 5, removing the singular value scaling in OLoRA-tail yields fundamentally more stable training dynamics that closely mirror standard LoRA.

We empirically evaluate the impact of singular value scaling across different singular subspaces in Figure 5. The empirical results reveal a nuanced interaction between subspace geometry and initialization scaling: in the inherently hazardous principal directions, OLoRA substantially delays the onset of reward collapse compared to PiSSA. Meanwhile, in the minor directions, OLoRA-tail effectively mitigates instability, yielding training dynamics much closer to standard LoRA. While certain directions are more prone to exceeding the KL leash, enforcing orthonormality offers a viable path toward stable RLVR training. This further motivates our theoretical research on optimization dynamics of LoRA in Section 5.

5 Optimization Dynamics of LoRA

In this section, we formally analyze the optimization dynamics of LoRA to understand how orthonormal initialization bridges the gap to full fine-tuning while maintaining training stability

The related proof has been deferred to the Appendix B. Following standard LoRA, we set $B_0 = 0$ and consider a more general initialization for $A_0 \in \mathbb{R}^{r\times n}$.

5.1 Orthonormal Initialization Closer to Full Fine-Tuning

Since LoRA constrains updates to a rank-r subspace with $r < n$, it cannot exactly match full fine-tuning. We quantify this approximation gap in Theorem 5.3.

Let T denote the total number of training iterations and $t \in \{0, 1, \ldots, T\}$ the current iteration. Consider a single linear layer with weight matrix $W_t = W_0 + B_t A_t$ and input x. The forward pass computes logits $z_t = W_t x$, and the policy $\pi_\theta(y \mid x) = \mathrm{softmax}(z_t)$ gives the probability of generating output y, where θ denotes the model parameters including W_t. The gradients with respect to the LoRA matrices are

$$\frac{\partial \mathcal{L}}{\partial A} = B^\top G_t, \quad \frac{\partial \mathcal{L}}{\partial B} = G_t A^\top,$$

where $G_t = \nabla_{W_t}\mathcal{L}$ denotes the gradient of the loss with respect to W_t.

Assumption 5.1. The RLVR loss $\mathcal{L}$ from (3) is L-smooth with respect to logits $z = Wx$.

This assumption holds for the RLVR objective with binary rewards. The detailed derivation is provided in Appendix A.1.

Assumption 5.2. The gradient is bounded: $\|G_t\|_F \le M$ for all t.

This assumption is typically enforced via gradient clipping.

Theorem 5.3 (LoRA Approximation Error)**.** *Let W_T^{full} and W_T^{LoRA} be the weights after T steps of full fine-tuning and LoRA under RLVR. Under Assumptions 5.1 and 5.2,*

$$\frac{1}{T}\left\|W_T^{\text{LoRA}} - W_T^{\text{full}}\right\|_F \le \frac{M\eta}{1 - L\eta\|x\|_2^2}\left\|I_n - A_0^\top A_0\right\|_2 + \mathcal{O}(\eta^2). \tag{7}$$

Theorem 5.3 shows that when the learning rate η is small, as is typical in practice, the approximation error between LoRA and full fine-tuning is controlled by $\|I_n - A_0^\top A_0\|_2$. The following corollary shows that orthonormal initialization of A_0 minimizes this term to 1.

Corollary 5.4. *Let $A_0 \in \mathbb{R}^{r\times n}$ with $r < n$. Then*

$$\left\|I_n - A_0^\top A_0\right\|_2 \geq 1,$$

with equality if and only if A_0 has orthonormal rows, i.e., $A_0 A_0^\top = I_r$.

With standard LoRA initialization $B_0 = 0$, Corollary 5.4 implies that orthonormal initialization A_0 minimizes the approximation gap between LoRA and full fine-tuning in RLVR, making LoRA's performance closer to that of full fine-tuning.

5.2 Orthonormal Initialization Stabilizes in RLVR

Proposition 5.5 (Bounded Weight Updates for Orthonormal Initialization)**.** *For LoRA parameterization $\Delta W^{lora} = BA$ with $B_0 = 0$ and orthonormal A_0, the first-step LoRA update satisfies*

$$\left\|\Delta W_1^{lora}\right\|_F = \eta \left\|G_0 A_0^\top A_0\right\|_F \leq \eta \left\|G_0\right\|_F, \tag{8}$$

where $G_0 = \nabla_{W_0}\mathcal{L}$.

Remark 5.6. This result shows that, at the first iteration, orthonormal initialization guarantees that the LoRA weight update $\left\|\Delta W_1^{\text{lora}}\right\|_F$ does not exceed that of full fine-tuning in Frobenius norm. This controlled first-step behavior is likely to persist throughout training by extending the proof of Proposition 5.5. As a result, it potentially mitigates abrupt policy shifts and improves RLVR fine-tuning stability.

6 Geometrical-Preserving Orthonormal Initialization for LoRA in RL

Inspired by our theoretical insights that orthonormal initialization of A_0 is beneficial for low-rank fine-tuning in RLVR (Corollary 5.4 and Proposition 5.5), we propose two initialization schemes that enforce orthonormality of A_0 while setting $B_0 = \mathbf{0}_{m\times r}$. In particular, to preserve the geometric structure of the pretrained weight matrix W_0, we design both schemes using the SVD of W_0.

Principal Orthonormal Initialization: LoRA-RLPO**.** We initialize the adapter A_0 using principal singular vectors:

$$B_0 = \mathbf{0}_{m\times r}, \quad A_0 = V_r^\top,$$

where $V_r \in \mathbb{R}^{n\times r}$ contains the top-r right singular vectors of W_0. This preserves geometric information from the pretrained model W_0 by aligning the adapter with its principal directions. The design is similar in spirit to PiSSA in that both retain the top-r singular directions of W_0, whereas PiSSA additionally incorporates the singular values.

Minor Orthonormal Initialization: LoRA-RLMO**.** Analogously, we define an initialization targeting the minor subspace:

$$B_0 = \mathbf{0}_{m\times r}, \quad A_0 = V_{-r}^\top,$$

where $V_{-r} \in \mathbb{R}^{n\times r}$ consists of the bottom-r right singular vectors of W_0. While MiLoRA also targets the minor subspace, it incorporates singular value scaling and nonzero B_0. In contrast, LoRA-RLMO adopts orthonormal A_0 with $B_0 = 0$.

7 Experiments and Analysis

In this section, we present empirical evaluations to validate our theoretical findings across various benchmarks.

	LoRA	MiLoRA	LoRA-RLMO (Ours)	PiSSA	LoRA-RLPO (Ours)
B_0	$\mathbf{0}$	$U_{-r}\Sigma_{-r}^{1/2}$	$\mathbf{0}$	$U_r\Sigma_r^{1/2}$	$\mathbf{0}$
A_0	$\mathcal{N}(0, \frac{1}{n})$	$\Sigma_{-r}^{1/2}V_{-r}^\top$	$V_{-r}^\top$	$\Sigma_r^{1/2}V_r^\top$	$V_r^\top$
GSM8K$^{@1}$	74.22	74.98	**76.42**	19.11	75.66
MATH500$^{@4}$	86.20	83.20	86.80	1.00	**88.30**
AIME22$^{@32}$	43.33	36.67	42.22	0.00	**48.33**
AIME23$^{@32}$	40.00	**43.33**	41.11	0.00	38.33
AIME24$^{@32}$	70.00	66.67	72.22	0.00	**73.33**
Avg	62.75	60.97	63.75	4.02	**64.79**

Table 1: Comparisons of SVD-based LoRA initialization methods under cosine learning rate decay.

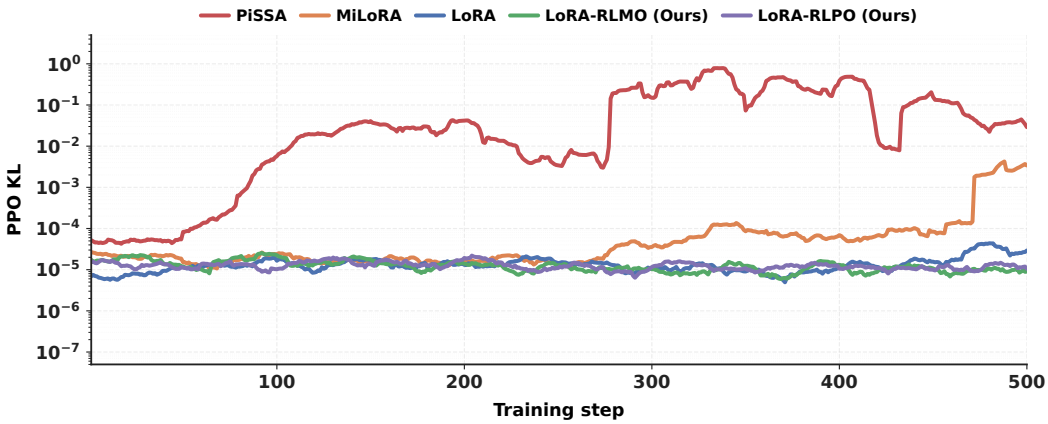

Figure 6: PPO KL during training for different initialization methods. PiSSA shows the highest KL divergence, MiLoRA exhibits intermediate KL growth, and LoRA remains relatively stable. Our proposed methods (LoRA-RLMO and LoRA-RLPO) maintain the lowest KL trajectories overall, indicating improved training stability.

Experimental Setup. We fine-tune DeepSeek-R1-Distill-Qwen-1.5B using DAPO [Yu et al., 2025] on DAPO-Math-17k [Yu et al., 2025] with rank $r = 16$ LoRA applied to all linear layers. We compare standard LoRA, PiSSA, MiLoRA, LoRA-RLPO, and LoRA-RLMO across five mathematical reasoning benchmarks: GSM8K [Cobbe et al., 2021] (1,319 samples), MATH500 [Hendrycks et al., 2021] (500 samples, mean@4), and AIME 2022/2023/2024 (30 samples each, mean@32) [Zhang and Math-AI, 2024]. Full experimental details are provided in Appendix D.

7.1 Evaluation of Proposed Methods

Performance. Table 1 compares SVD-based LoRA initialization methods at step 500 across five mathematical reasoning benchmarks. Overall, LoRA-RLPO achieves the highest average accuracy (64.79%), followed by LoRA-RLMO (63.75%) and LoRA (62.75%). LoRA-RLPO obtains the best performance on MATH500 (88.30%), AIME22 (48.33%), and AIME24 (73.33%), while LoRA-RLMO achieves the strongest GSM8K result (76.42%) and remains competitive on the remaining benchmarks. MiLoRA trails behind standard LoRA despite achieving the best AIME23 result (43.33%). PiSSA performs substantially worse than the other methods in this setting. These results demonstrate that our geometry-preserving orthonormal initializations resolve the underperformance of existing SVD-based methods, successfully adapting SVD-informed LoRA

variants to RLVR.

Training Stability. Figure 6 compares PPO KL during training for different initialization methods. PiSSA shows the largest KL divergence by a wide margin, while MiLoRA exhibits intermediate KL growth. LoRA, LoRA-RLMO, and LoRA-RLPO remain in a low-KL regime throughout training. Among them, our proposed methods are particularly stable, with PPO KL trajectories that are consistently comparable to, and even lower than that of standard LoRA. Together with their stronger final evaluation results, these observations indicate that geometry-preserving initialization supports both stable optimization and improved downstream performance in RLVR.

Preserving Pretrained Geometry. Our theoretical analysis shows that orthonormal A_0 minimizes approximation error to full fine-tuning. To verify that orthonormality alone improves RLVR training, independent of SVD-based geometry information, we introduce two model-agnostic baselines: **DCT-LoRA** and **Wavelet-LoRA**. Both satisfy the orthonormality condition $A_0 A_0^\top = I_r$ but do not use any information from the pretrained weight matrix W_0.

DCT-LoRA initializes $B_0 = 0$ and $A_0 = D^\top_{:r,:}$ where $D \in \mathbb{R}^{n\times n}$ is the Discrete Cosine Transform (DCT) orthonormal matrix [Ahmed et al., 2006], with entries:

$$D_{ij} = \sqrt{\frac{2}{n}} \cos\left(\frac{\pi(2j+1)i}{2n}\right).$$

Wavelet-LoRA initializes $B_0 = 0$ and $A_0 = H^\top_{:r,:}$ where $H \in \mathbb{R}^{n\times n}$ is the Haar wavelet orthonormal matrix [Mallat, 1999].

To analyze how these initialization strategies preserve or distort the geometric structure of pretrained weights, we measure the subspace similarity between the learned adapter A and the singular vectors of W_0. Let V_r and V_{-r} denote the top-r (principal) and bottom-r (minor) right singular vectors of W_0, respectively. For each layer, we compute the similarity as $\|AV\|_F / \|A\|_F$ and report the average across all layers.

Figure 7 shows the results. LoRA-RLPO maintains high subspace similarity with the principal singular vectors throughout training, consistent with its initialization from V_r. Similarly, LoRA-RLMO shows high similarity with the minor singular vectors. In contrast, DCT-LoRA and Wavelet-LoRA exhibit uniformly low similarity across all singular subspaces, confirming that these model-agnostic bases do not align with the pretrained weight geometry.

These results suggest that learning in the principal subspace is not necessarily harmful, but it is considerably more fragile in RLVR. While both LoRA-RLPO and PiSSA are initialized in the principal singular subspace, only LoRA-RLPO remains stable and achieves the best overall performance (Table 1). Therefore, subspace choice alone does not determine the outcome. Rather, our results indicate that an orthonormal initialization together with cosine learning-rate decay is sufficient to make principal-subspace learning stable and effective in this setting.

7.2 Ablation for Orthonormal Initialization

Orthonormality vs. Geometry Information. Our theoretical analysis shows that orthonormal A_0 minimizes approximation error to full fine-tuning. To verify that orthonormality alone improves RLVR training, independent of SVD-based geometry information, we compare standard LoRA with DCT-LoRA and Wavelet-LoRA, which use model-agnostic orthonormal bases.

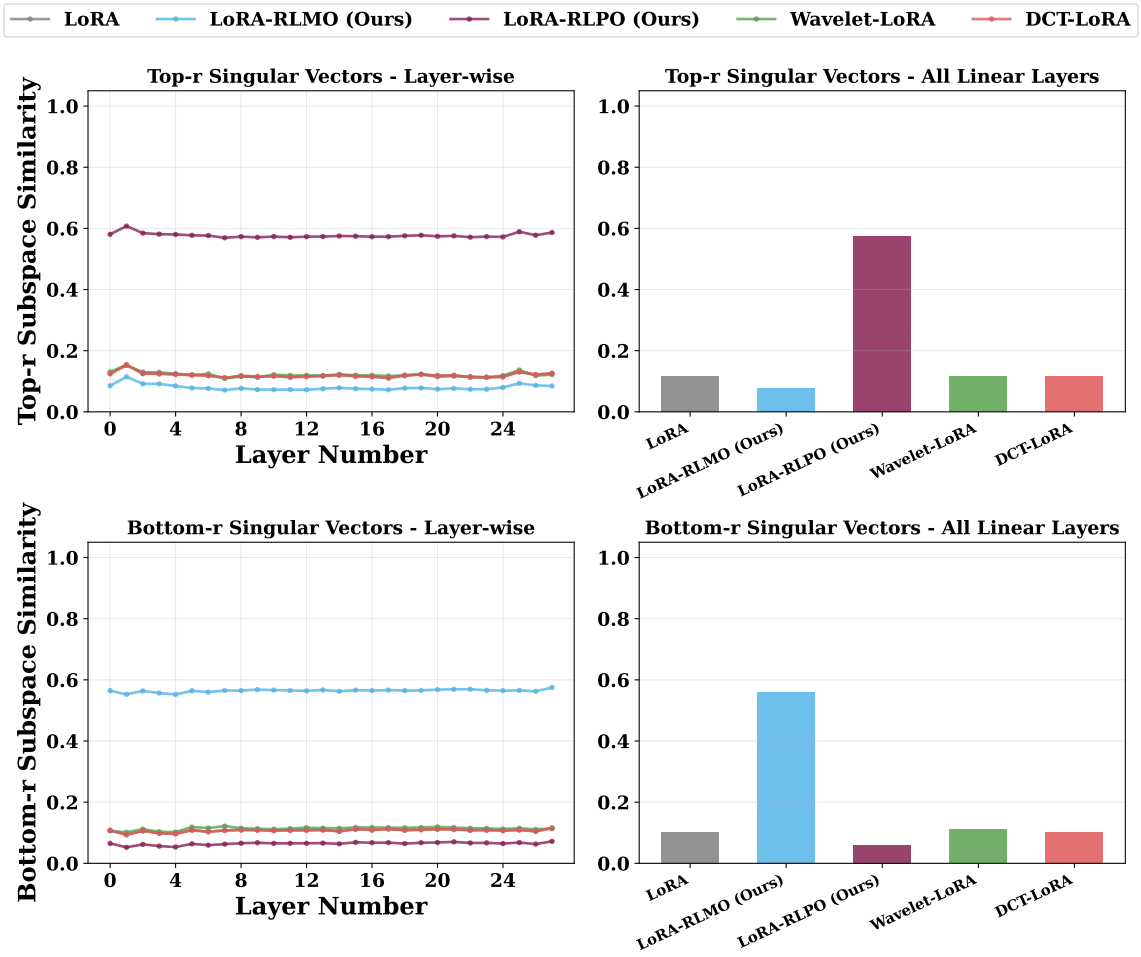

Figure 7: Subspace similarity between learned adapters and singular vectors of pretrained weights W_0. Top row: similarity with principal (top-r) right singular vectors. Bottom row: similarity with minor (bottom-r) right singular vectors. Left column shows per-layer similarity; right columns show averages over all linear layers.

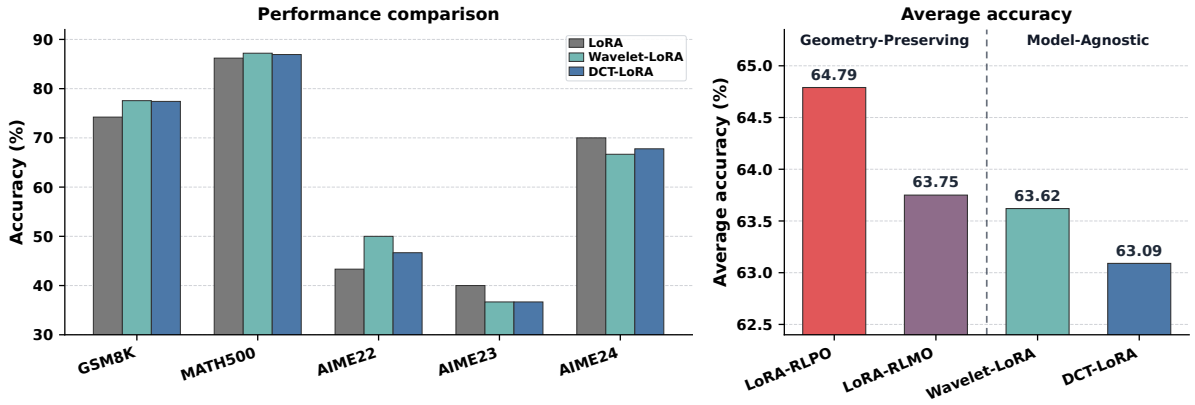

Figure 8: Left: Performance comparison of orthonormal LoRA variants with standard LoRA. Right: Average accuracy comparison between geometry-preserving and model-agnostic methods.

Figure 8 (left) shows that both DCT-LoRA and Wavelet-LoRA outperform standard LoRA across most benchmarks, confirming that orthonormality alone improves RLVR training. Figure 8 (right) shows that LoRA-RLMO achieves comparable performance to model-agnostic methods,

while LoRA-RLPO significantly outperforms all others, suggesting that principal subspace alignment provides substantial additional benefit beyond orthonormality alone.

8 Conclusion

In this work, we investigate the underlying mechanisms that cause geometry-informed LoRA variants to destabilize during RLVR training. We identify two primary factors that govern this instability: (1) **Subspace Geometry**, which fundamentally shapes the optimization trajectory and induces high energy concentration in parameter updates; and (2) **Singular Value Scaling**, which serves as a distinct destabilizing factor by amplifying gradient magnitudes and driving rapid KL divergence.

Our theoretical and empirical analyses demonstrate that while geometry-informed methods like PiSSA and MiLoRA offer potential for faster convergence, their inherent singular value scaling often leads to updates that exceed the implicit KL budget of the RLVR framework. Based on these insights, we propose LoRA-RLPO and LoRA-RLMO, two geometry-preserving orthonormal variants that successfully adapt from SFT to RLVR.

References

Nasir Ahmed, T_ Natarajan, and Kamisetty R Rao. Discrete cosine transform. *IEEE transactions on Computers*, 100(1):90–93, 2006.

Zhangir Azerbayev, Hailey Schoelkopf, Keiran Paster, Marco Dos Santos, Stephen McAleer, Albert Q. Jiang, Jia Deng, Stella Biderman, and Sean Welleck. Llemma: An open language model for mathematics, 2024. URL https://arxiv.org/abs/2310.10631.

Tom B. Brown, Benjamin Mann, Nick Ryder, Melanie Subbiah, Jared Kaplan, Prafulla Dhariwal, Arvind Neelakantan, Pranav Shyam, Girish Sastry, Amanda Askell, Sandhini Agarwal, Ariel Herbert-Voss, Gretchen Krueger, Tom Henighan, Rewon Child, Aditya Ramesh, Daniel M. Ziegler, Jeffrey Wu, Clemens Winter, Christopher Hesse, Mark Chen, Eric Sigler, Mateusz Litwin, Scott Gray, Benjamin Chess, Jack Clark, Christopher Berner, Sam McCandlish, Alec Radford, Ilya Sutskever, and Dario Amodei. Language models are few-shot learners, 2020. URL https://arxiv.org/abs/2005.14165.

Kerim Büyükakyüz. Olora: Orthonormal low-rank adaptation of large language models, 2024. URL https://arxiv.org/abs/2406.01775.

Zeming Chen, Alejandro Hernández Cano, Angelika Romanou, Antoine Bonnet, Kyle Matoba, Francesco Salvi, Matteo Pagliardini, Simin Fan, Andreas Köpf, Amirkeivan Mohtashami, Alexandre Sallinen, Alireza Sakhaeirad, Vinitra Swamy, Igor Krawczuk, Deniz Bayazit, Axel Marmet, Syrielle Montariol, Mary-Anne Hartley, Martin Jaggi, and Antoine Bosselut. Meditron-70b: Scaling medical pretraining for large language models, 2023. URL https://arxiv.org/abs/2311.16079.

Karl Cobbe, Vineet Kosaraju, Mohammad Bavarian, Mark Chen, Heewoo Jun, Lukasz Kaiser, Matthias Plappert, Jerry Tworek, Jacob Hilton, Reiichiro Nakano, et al. Training verifiers to solve math word problems. *arXiv preprint arXiv:2110.14168*, 2021.

Tim Dettmers, Artidoro Pagnoni, Ari Holtzman, and Luke Zettlemoyer. Qlora: Efficient finetuning of quantized llms, 2023. URL https://arxiv.org/abs/2305.14314.

Daya Guo, Dejian Yang, Haowei Zhang, Junxiao Song, Peiyi Wang, et al. Deepseek-r1 incentivizes reasoning in llms through reinforcement learning. *Nature*, 645(8081):633–638, September

2025. ISSN 1476-4687. doi: 10.1038/s41586-025-09422-z. URL http://dx.doi.org/10.1038/s41586-025-09422-z.

Soufiane Hayou, Nikhil Ghosh, and Bin Yu. Lora+: Efficient low rank adaptation of large models, 2024. URL https://arxiv.org/abs/2402.12354.

Dan Hendrycks, Collin Burns, Saurav Kadavath, Akul Arora, Steven Basart, Eric Tang, Dawn Song, and Jacob Steinhardt. Measuring mathematical problem solving with the math dataset, 2021. URL https://arxiv.org/abs/2103.03874.

Neil Houlsby, Andrei Giurgiu, Stanislaw Jastrzebski, Bruna Morrone, Quentin de Laroussilhe, Andrea Gesmundo, Mona Attariyan, and Sylvain Gelly. Parameter-efficient transfer learning for nlp, 2019. URL https://arxiv.org/abs/1902.00751.

Edward J. Hu, Yelong Shen, Phillip Wallis, Zeyuan Allen-Zhu, Yuanzhi Li, Shean Wang, Lu Wang, and Weizhu Chen. Lora: Low-rank adaptation of large language models, 2021. URL https://arxiv.org/abs/2106.09685.

Sham Kakade and John Langford. Approximately optimal approximate reinforcement learning. In *Proceedings of the Nineteenth International Conference on Machine Learning*, ICML '02, page 267–274, San Francisco, CA, USA, 2002. Morgan Kaufmann Publishers Inc. ISBN 1558608737.

Damjan Kalajdzievski. A rank stabilization scaling factor for fine-tuning with lora, 2023. URL https://arxiv.org/abs/2312.03732.

Dawid J. Kopiczko, Tijmen Blankevoort, and Yuki M. Asano. Vera: Vector-based random matrix adaptation, 2024. URL https://arxiv.org/abs/2310.11454.

Brian Lester, Rami Al-Rfou, and Noah Constant. The power of scale for parameter-efficient prompt tuning, 2021. URL https://arxiv.org/abs/2104.08691.

Xiang Lisa Li and Percy Liang. Prefix-tuning: Optimizing continuous prompts for generation, 2021. URL https://arxiv.org/abs/2101.00190.

Shih-Yang Liu, Chien-Yi Wang, Hongxu Yin, Pavlo Molchanov, Yu-Chiang Frank Wang, Kwang-Ting Cheng, and Min-Hung Chen. Dora: Weight-decomposed low-rank adaptation, 2024a. URL https://arxiv.org/abs/2402.09353.

Weiyang Liu, Zeju Qiu, Yao Feng, Yuliang Xiu, Yuxuan Xue, Longhui Yu, Haiwen Feng, Zhen Liu, Juyeon Heo, Songyou Peng, Yandong Wen, Michael J. Black, Adrian Weller, and Bernhard Schölkopf. Parameter-efficient orthogonal finetuning via butterfly factorization, 2024b. URL https://arxiv.org/abs/2311.06243.

Haipeng Luo, Qingfeng Sun, Can Xu, Pu Zhao, Jianguang Lou, Chongyang Tao, Xiubo Geng, Qingwei Lin, Shifeng Chen, Yansong Tang, and Dongmei Zhang. Wizardmath: Empowering mathematical reasoning for large language models via reinforced evol-instruct, 2025a. URL https://arxiv.org/abs/2308.09583.

Ziyang Luo, Can Xu, Pu Zhao, Qingfeng Sun, Xiubo Geng, Wenxiang Hu, Chongyang Tao, Jing Ma, Qingwei Lin, and Daxin Jiang. Wizardcoder: Empowering code large language models with evol-instruct, 2025b. URL https://arxiv.org/abs/2306.08568.

Stephane Mallat. *A wavelet tour of signal processing*. Academic Press, 1999.

Fanxu Meng, Zhaohui Wang, and Muhan Zhang. Pissa: Principal singular values and singular vectors adaptation of large language models, 2025. URL https://arxiv.org/abs/2404.02948.

Long Ouyang, Jeff Wu, Xu Jiang, Diogo Almeida, Carroll L. Wainwright, Pamela Mishkin, Chong Zhang, Sandhini Agarwal, Katarina Slama, Alex Ray, John Schulman, Jacob Hilton, Fraser Kelton, Luke Miller, Maddie Simens, Amanda Askell, Peter Welinder, Paul Christiano, Jan Leike, and Ryan Lowe. Training language models to follow instructions with human feedback, 2022. URL https://arxiv.org/abs/2203.02155.

Zeju Qiu, Weiyang Liu, Haiwen Feng, Yuxuan Xue, Yao Feng, Zhen Liu, Dan Zhang, Adrian Weller, and Bernhard Schölkopf. Controlling text-to-image diffusion by orthogonal finetuning, 2024. URL https://arxiv.org/abs/2306.07280.

Baptiste Rozière, Jonas Gehring, Fabian Gloeckle, Sten Sootla, Itai Gat, Xiaoqing Ellen Tan, Yossi Adi, Jingyu Liu, Romain Sauvestre, Tal Remez, Jérémy Rapin, Artyom Kozhevnikov, Ivan Evtimov, Joanna Bitton, Manish Bhatt, Cristian Canton Ferrer, Aaron Grattafiori, Wenhan Xiong, Alexandre Défossez, Jade Copet, Faisal Azhar, Hugo Touvron, Louis Martin, Nicolas Usunier, Thomas Scialom, and Gabriel Synnaeve. Code llama: Open foundation models for code, 2024. URL https://arxiv.org/abs/2308.12950.

Michael Santacroce, Yadong Lu, Han Yu, Yuanzhi Li, and Yelong Shen. Efficient rlhf: Reducing the memory usage of ppo, 2023. URL https://arxiv.org/abs/2309.00754.

John Schulman and Thinking Machines Lab. Lora without regret. *Thinking Machines Lab: Connectionism*, 2025. doi: 10.64434/tml.20250929. https://thinkingmachines.ai/blog/lora/.

John Schulman, Sergey Levine, Pieter Abbeel, Michael Jordan, and Philipp Moritz. Trust region policy optimization. In Francis Bach and David Blei, editors, *Proceedings of the 32nd International Conference on Machine Learning*, volume 37 of *Proceedings of Machine Learning Research*, pages 1889–1897, Lille, France, 07–09 Jul 2015. PMLR. URL https://proceedings.mlr.press/v37/schulman15.html.

John Schulman, Filip Wolski, Prafulla Dhariwal, Alec Radford, and Oleg Klimov. Proximal policy optimization algorithms, 2017. URL https://arxiv.org/abs/1707.06347.

Zhihong Shao, Peiyi Wang, Qihao Zhu, Runxin Xu, Junxiao Song, Xiao Bi, Haowei Zhang, Mingchuan Zhang, Y. K. Li, Y. Wu, and Daya Guo. Deepseekmath: Pushing the limits of mathematical reasoning in open language models, 2024. URL https://arxiv.org/abs/2402.03300.

Idan Shenfeld, Jyothish Pari, and Pulkit Agrawal. Rl's razor: Why online reinforcement learning forgets less, 2025. URL https://arxiv.org/abs/2509.04259.

Karan Singhal, Shekoofeh Azizi, Tao Tu, S. Sara Mahdavi, Jason Wei, Hyung Won Chung, Nathan Scales, Ajay Tanwani, Heather Cole-Lewis, Stephen Pfohl, Perry Payne, Martin Seneviratne, Paul Gamble, Chris Kelly, Nathaneal Scharli, Aakanksha Chowdhery, Philip Mansfield, Blaise Aguera y Arcas, Dale Webster, Greg S. Corrado, Yossi Matias, Katherine Chou, Juraj Gottweis, Nenad Tomasev, Yun Liu, Alvin Rajkomar, Joelle Barral, Christopher Semturs, Alan Karthikesalingam, and Vivek Natarajan. Large language models encode clinical knowledge, 2022. URL https://arxiv.org/abs/2212.13138.

Hugo Touvron, Thibaut Lavril, Gautier Izacard, Xavier Martinet, Marie-Anne Lachaux, Timothée Lacroix, Baptiste Rozière, Naman Goyal, Eric Hambro, Faisal Azhar, Aurelien Rodriguez, Armand Joulin, Edouard Grave, and Guillaume Lample. Llama: Open and efficient foundation language models, 2023. URL https://arxiv.org/abs/2302.13971.

Hanqing Wang, Yixia Li, Shuo Wang, Guanhua Chen, and Yun Chen. Milora: Harnessing minor singular components for parameter-efficient llm finetuning, 2025. URL https://arxiv.org/abs/2406.09044.

Fang Wu, Weihao Xuan, Ximing Lu, Mingjie Liu, Yi Dong, Zaid Harchaoui, and Yejin Choi. The invisible leash: Why rlvr may or may not escape its origin, 2026. URL https://arxiv.org/abs/2507.14843.

Shijie Wu, Ozan Irsoy, Steven Lu, Vadim Dabravolski, Mark Dredze, Sebastian Gehrmann, Prabhanjan Kambadur, David Rosenberg, and Gideon Mann. Bloomberggpt: A large language model for finance, 2023. URL https://arxiv.org/abs/2303.17564.

Hongyang Yang, Xiao-Yang Liu, and Christina Dan Wang. Fingpt: Open-source financial large language models, 2025. URL https://arxiv.org/abs/2306.06031.

Qingyu Yin, Yulun Wu, Zhennan Shen, Sunbowen Li, Zhilin Wang, Yanshu Li, Chak Tou Leong, Jiale Kang, and Jinjin Gu. Evaluating parameter efficient methods for rlvr, 2025. URL https://arxiv.org/abs/2512.23165.

Qiying Yu, Zheng Zhang, Ruofei Zhu, Yufeng Yuan, Xiaochen Zuo, Yu Yue, Weinan Dai, Tiantian Fan, Gaohong Liu, Lingjun Liu, Xin Liu, Haibin Lin, Zhiqi Lin, Bole Ma, Guangming Sheng, Yuxuan Tong, Chi Zhang, Mofan Zhang, Wang Zhang, Hang Zhu, Jinhua Zhu, Jiaze Chen, Jiangjie Chen, Chengyi Wang, Hongli Yu, Yuxuan Song, Xiangpeng Wei, Hao Zhou, Jingjing Liu, Wei-Ying Ma, Ya-Qin Zhang, Lin Yan, Mu Qiao, Yonghui Wu, and Mingxuan Wang. Dapo: An open-source llm reinforcement learning system at scale, 2025. URL https://arxiv.org/abs/2503.14476.

Qingru Zhang, Minshuo Chen, Alexander Bukharin, Nikos Karampatziakis, Pengcheng He, Yu Cheng, Weizhu Chen, and Tuo Zhao. Adalora: Adaptive budget allocation for parameter-efficient fine-tuning, 2023. URL https://arxiv.org/abs/2303.10512.

Yifan Zhang and Team Math-AI. American invitational mathematics examination (aime) 2024, 2024.

Jin Zhou, Hanmei Yang, Steven, Tang, Mingcan Xiang, Hui Guan, and Tongping Liu. Understanding and alleviating memory consumption in rlhf for llms, 2024. URL https://arxiv.org/abs/2410.15651.

Hanqing Zhu, Zhenyu Zhang, Hanxian Huang, DiJia Su, Zechun Liu, Jiawei Zhao, Igor Fedorov, Hamed Pirsiavash, Zhizhou Sha, Jinwon Lee, David Z. Pan, Zhangyang Wang, Yuandong Tian, and Kai Sheng Tai. The path not taken: Rlvr provably learns off the principals, 2025. URL https://arxiv.org/abs/2511.08567.

Jiacheng Zhu, Kristjan Greenewald, Kimia Nadjahi, Haitz Sáez de Ocáriz Borde, Rickard Brüel Gabrielsson, Leshem Choshen, Marzyeh Ghassemi, Mikhail Yurochkin, and Justin Solomon. Asymmetry in low-rank adapters of foundation models, 2024. URL https://arxiv.org/abs/2402.16842.

Daniel M. Ziegler, Nisan Stiennon, Jeffrey Wu, Tom B. Brown, Alec Radford, Dario Amodei, Paul Christiano, and Geoffrey Irving. Fine-tuning language models from human preferences, 2020. URL https://arxiv.org/abs/1909.08593.

A Additional Theoretical Results

A.1 Smoothness of the Loss Function (Assumption 5.1)

We verify that Assumption 5.1 holds for both supervised fine-tuning (SFT) with cross-entropy loss and reinforcement learning with verifiable rewards (RLVR) with policy gradient loss.

Cross-entropy loss: Let $y = \mathrm{softmax}(z)$. For cross-entropy $\mathcal{L} = -\sum_i t_i \log y_i$ with one-hot target t:

$$\frac{\partial \mathcal{L}}{\partial z} = y - t, \quad \frac{\partial^2 \mathcal{L}}{\partial z^2} = \mathrm{diag}(y) - yy^\top.$$

Let $H = \mathrm{diag}(y) - yy^\top$. For any v with $\|v\|_2 = 1$:

$$v^\top H v = \sum_i y_i v_i^2 - \left(\sum_i y_i v_i\right)^2 \leq \sum_i y_i v_i^2 \leq 1,$$

where the first inequality uses $(\sum_i y_i v_i)^2 \geq 0$, and the second inequality follows from Jensen's inequality. Thus, $\|H\|_2 \leq 1$ for all z, which implies $L \leq 1$.

RLVR loss: The standard policy-gradient surrogate of (3) is:

$$\mathcal{L}_{\mathrm{PG}}(\theta) = -\mathbb{E}_{x\sim\mathcal{X}, y\sim\pi_\theta(\cdot|x)}\left[A^\perp(x,y)\log \pi_\theta(y|x)\right],$$

where $A^\perp$ is an advantage estimate. The Hessian with respect to logits satisfies:

$$\frac{\partial^2 \mathcal{L}_{\mathrm{PG}}}{\partial z^2} = -A^\perp(x,y)\left(\mathrm{diag}(y) - yy^\top\right),$$

with spectral norm bounded by $|A^\perp(x,y)|$. For binary verifiable rewards $R \in \{0,1\}$, we have $|A^\perp(x,y)| \leq 1$, giving $L = 1$.

A.2 PiSSA and MiLoRA Failure Analysis

Table 2: Initialization norms (Frobenius) for LoRA and MiLoRA on DeepSeek-R1-Distill-Qwen-1.5B, averaged across all target linear layers at rank $r = 32$.

	LoRA	MiLoRA
$\|B_0A_0\|_F$	0	5.11

Comparison with prior hypothesis. Yin et al. [2025] attribute MiLoRA's failure to negligible initialization magnitude, arguing that small tail singular values cause the adapter to collapse. Our results on DeepSeek-R1-Distill-Qwen-1.5B do not support this explanation. Table 2 reports initialization norms on the 1.5B model. Notably, MiLoRA has $\|B_0A_0\|_F = 5.11$, which is clearly non-negligible, contradicting the near-zero initialization hypothesis. In contrast, standard LoRA satisfies $\|B_0A_0\|_F = 0$, yet it remains stable during RL training. This shows that MiLoRA's failure cannot be explained solely by initialization magnitude.

Furthermore, the singular-value structure of the 1.5B model does not support the hypothesis that MiLoRA fails because the minor subspace is numerically negligible. As shown in Figure 9, the top singular components are indeed substantially larger than the tail components across all three MLP projections, but the tail singular values remain clearly non-zero. In DeepSeek-R1-Distill-Qwen-1.5B, the mean of the tail 16 singular values is approximately 0.46–1.54 in layer 1, 1.04–1.52 in layer 14, and 1.32–1.73 in layer 28, depending on the projection module. Thus, although the minor spectrum is weaker than the principal spectrum, it still retains substantial signal across representative layers and modules. Taken together, these results suggest that MiLoRA's instability is unlikely to be caused simply by vanishing tail-spectrum magnitude.

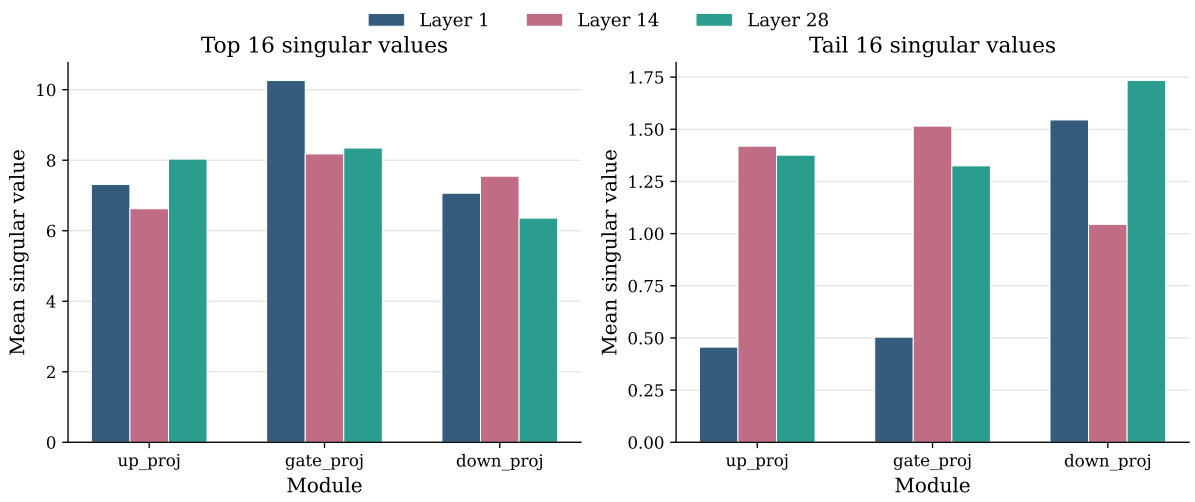

Figure 9: Mean singular values of MLP projection weights in DeepSeek-R1-Distill-Qwen-1.5B across representative layers 1, 14, and 28.

B Proofs for Section 5 (Optimization Dynamics)

B.1 Proof of Lemma B.1 (First-Step LoRA Update)

Lemma B.1 (First-Step LoRA Update)**.** *Let $W_0 \in \mathbb{R}^{m\times n}$ be the pretrained weight and consider the LoRA parameterization $W = W_0 + BA$ with $B_0 = 0_{m\times r}$ and $A_0 \in \mathbb{R}^{r\times n}$. Under gradient descent with step size η, the first-step weight updates satisfy:*

$$\Delta W_1^{\text{LoRA}} = \Delta W_1^{\text{full}} A_0^\top A_0, \tag{9}$$

where $\Delta W_1^{\text{full}} = -\eta \nabla_W \mathcal{L}(W_0)$ is the full fine-tuning update and $\Delta W_1^{\text{LoRA}} = B_1 A_1$ is the LoRA update.

Proof. Let $G_0 := \nabla_W \mathcal{L}(W)|_{W=W_0} \in \mathbb{R}^{m\times n}$ denote the gradient at initialization. By the chain rule for $\Delta W = BA$, the gradients with respect to the LoRA matrices are:

$$\nabla_A \mathcal{L}\big|_0 = B_0^\top G_0 = 0, \qquad \nabla_B \mathcal{L}\big|_0 = G_0 A_0^\top.$$

After one gradient step:

$$A_1 = A_0 - \eta \nabla_A \mathcal{L}\big|_0 = A_0, \qquad B_1 = B_0 - \eta \nabla_B \mathcal{L}\big|_0 = -\eta G_0 A_0^\top.$$

Thus the LoRA weight update is:

$$\Delta W_1^{\text{LoRA}} = B_1 A_1 = -\eta G_0 A_0^\top A_0.$$

For full fine-tuning:

$$\Delta W_1^{\text{full}} = -\eta G_0.$$

Therefore:

$$\Delta W_1^{\text{LoRA}} = \Delta W_1^{\text{full}} A_0^\top A_0,$$

which completes the proof. □

B.2 Proof of Theorem 5.3 (LoRA Approximation Error)

Proof. Let W_t^{LoRA} and W_t^{full} denote the weights after t gradient descent steps with step size η, initialized from the same pretrained weight

$$W_0^{\text{LoRA}} = W_0^{\text{full}} = W_0.$$

For LoRA, write

$$W_t^{\text{LoRA}} = W_0 + \Delta W_t^{\text{LoRA}}, \qquad \Delta W_t^{\text{LoRA}} = B_t A_t,$$

where $B_0 = 0$ and $A_0 \in \mathbb{R}^{r\times n}$. Define the cumulative approximation error $E_t \coloneqq \left\|W_t^{\text{LoRA}} - W_t^{\text{full}}\right\|_F$. Our goal is to upper bound E_T/T.

Let $G_t \coloneqq \nabla_W \mathcal{L}(W)\big|_{W=W_t^{\text{LoRA}}} = g_t x^\top, G_t' \coloneqq \nabla_W \mathcal{L}(W)\big|_{W=W_t^{\text{full}}} = g_t' x^\top$.

By Lemma B.1, the first LoRA update satisfies

$$W_1^{\text{LoRA}} - W_0^{\text{LoRA}} = -\eta G_0 A_0^\top A_0,$$

whereas the corresponding full fine-tuning update is $W_1^{\text{full}} - W_0^{\text{full}} = -\eta G_0$.

Therefore, $E_1 = \left\|W_1^{\text{LoRA}} - W_1^{\text{full}}\right\|_F = \eta \left\|G_0(I_n - A_0^\top A_0)\right\|_F \le \eta\|G_0\|_F \left\|I_n - A_0^\top A_0\right\|_2$.

By Assumption 5.2, $\|G_0\|_F \le M$, and hence $E_1 \le M\eta \left\|I_n - A_0^\top A_0\right\|_2$.

We next control the error recursively. For $t \ge 1$, full fine-tuning performs the update

$$W_{t+1}^{\text{full}} = W_t^{\text{full}} - \eta G_t'.$$

For LoRA, using the gradient updates

$$A_{t+1} = A_t - \eta B_t^\top G_t, \qquad B_{t+1} = B_t - \eta G_t A_t^\top,$$

and the initialization $B_0 = 0$, we have, for small η,

$$A_t = A_0 + \mathcal{O}(\eta^2), \qquad B_{t+1} - B_t = -\eta G_t A_0^\top + \mathcal{O}(\eta^3).$$

Consequently,

$$\begin{aligned} W_{t+1}^{\text{LoRA}} - W_t^{\text{LoRA}} &= B_{t+1}A_{t+1} - B_t A_t \\ &= -\eta G_t A_0^\top A_0 + \mathcal{O}(\eta^3). \end{aligned} \tag{10}$$

Thus,

$$\begin{aligned} W_{t+1}^{\text{LoRA}} - W_{t+1}^{\text{full}} &= W_t^{\text{LoRA}} - W_t^{\text{full}} + \left(W_{t+1}^{\text{LoRA}} - W_t^{\text{LoRA}}\right) - \left(W_{t+1}^{\text{full}} - W_t^{\text{full}}\right) \\ &= W_t^{\text{LoRA}} - W_t^{\text{full}} - \eta G_t A_0^\top A_0 + \eta G_t' + \mathcal{O}(\eta^3) \\ &= W_t^{\text{LoRA}} - W_t^{\text{full}} - \eta G_t(A_0^\top A_0 - I_n) + \eta(G_t' - G_t) + \mathcal{O}(\eta^3). \end{aligned} \tag{11}$$

Taking Frobenius norms and applying the triangle inequality gives

$$\begin{aligned} E_{t+1} &\le E_t + \eta \left\|G_t(I_n - A_0^\top A_0)\right\|_F + \eta \left\|G_t - G_t'\right\|_F + \mathcal{O}(\eta^3) \\ &\le E_t + \eta\|G_t\|_F \left\|I_n - A_0^\top A_0\right\|_2 + \eta\|G_t - G_t'\|_F + \mathcal{O}(\eta^3). \end{aligned} \tag{12}$$

By Assumption 5.2, $\|G_t\|_F \le M$. Moreover, since

$$G_t - G_t' = (g_t - g_t')x^\top,$$

we have

$$\|G_t - G_t'\|_F = \|(g_t - g_t')x^\top\|_F = \|g_t - g_t'\|_2\|x\|_2.$$

By Assumption 5.1, the loss is L-smooth with respect to the logits. Therefore,

$$\begin{aligned}\|g_t - g_t'\|_2 &\le L\|z_t - z_t'\|_2 \\ &= L\|(W_t^{\mathrm{LoRA}} - W_t^{\mathrm{full}})x\|_2 \\ &\le L\|W_t^{\mathrm{LoRA}} - W_t^{\mathrm{full}}\|_F\|x\|_2 = LE_t\|x\|_2. \end{aligned} \tag{13}$$

Substituting this into (12) yields

$$E_{t+1} \le \left(1 + L\eta\|x\|_2^2\right) E_t + M\eta \left\|I_n - A_0^\top A_0\right\|_2 + \mathcal{O}(\eta^3). \tag{14}$$

At this point, directly unrolling (14) would produce a factor depending on T. Since the theorem concerns the average approximation error E_T/T, we equivalently write

$$E_T = \left\| \sum_{t=0}^{T-1} \left[\left(W_{t+1}^{\mathrm{LoRA}} - W_t^{\mathrm{LoRA}}\right) - \left(W_{t+1}^{\mathrm{full}} - W_t^{\mathrm{full}}\right) \right] \right\|_F.$$

Using the triangle inequality and the same step-wise bound above, we obtain

$$\begin{aligned} E_T &\le \sum_{t=0}^{T-1} \left\| \left(W_{t+1}^{\mathrm{LoRA}} - W_t^{\mathrm{LoRA}}\right) - \left(W_{t+1}^{\mathrm{full}} - W_t^{\mathrm{full}}\right) \right\|_F \\ &\le \sum_{t=0}^{T-1} \left[M\eta \left\|I_n - A_0^\top A_0\right\|_2 + L\eta\|x\|_2^2 E_t + \mathcal{O}(\eta^2) \right]. \end{aligned} \tag{15}$$

Since $E_t \le E_T + \mathcal{O}(\eta)$ for $t \le T$ up to higher-order terms under small-step gradient descent, we have

$$\frac{1}{T}\sum_{t=0}^{T-1} E_t \le \frac{1}{T} E_T + \mathcal{O}(\eta).$$

Substituting this into (15) and dividing both sides by T gives

$$\frac{1}{T}E_T \le M\eta \left\|I_n - A_0^\top A_0\right\|_2 + L\eta\|x\|_2^2 \frac{1}{T} E_T + \mathcal{O}(\eta^2).$$

Rearranging terms, provided that $L\eta\|x\|_2^2 < 1$, yields

$$\frac{1}{T}E_T \le \frac{M\eta}{1 - L\eta\|x\|_2^2} \left\|I_n - A_0^\top A_0\right\|_2 + \mathcal{O}(\eta^2).$$

Since $E_T = \|W_T^{\mathrm{LoRA}} - W_T^{\mathrm{full}}\|_F$, we obtain

$$\frac{1}{T}\left\|W_T^{\mathrm{LoRA}} - W_T^{\mathrm{full}}\right\|_F \le \frac{M\eta}{1 - L\eta\|x\|_2^2} \left\|I_n - A_0^\top A_0\right\|_2 + \mathcal{O}(\eta^2),$$

which completes the proof. □

B.3 Proof of Corollary 5.4 (Optimality of Orthonormal Initialization)

Proof. Let $A_0 \in \mathbb{R}^{r\times n}$ with $r < n$ and $A_0^\top A_0 \succeq 0$. Since $\mathrm{rank}(A_0^\top A_0) \le \mathrm{rank}(A_0) \le r < n$, the matrix $A_0^\top A_0$ has at least $n - r$ zero eigenvalues. Consequently, $I_n - A_0^\top A_0$ has at least $n - r$ eigenvalues equal to 1, and therefore

$$\|I_n - A_0^\top A_0\|_2 \ge 1. \tag{16}$$

Now suppose A_0 has orthonormal rows, i.e., $A_0 A_0^\top = I_r$. Then for any $z \in \mathbb{R}^n$ we can decompose

$$z = z_\parallel + z_\perp, \qquad z_\parallel \in \mathrm{row}(A_0), \quad z_\perp \in \mathrm{row}(A_0)^\perp.$$

For $z_\parallel = A_0^\top u \in \mathrm{row}(A_0)$,

$$A_0^\top A_0 z_\parallel = A_0^\top A_0 A_0^\top u = A_0^\top (A_0 A_0^\top) u = A_0^\top u = z_\parallel,$$

while for $z_\perp \perp \mathrm{row}(A_0)$ we have $A_0 z_\perp = 0$ and thus $A_0^\top A_0 z_\perp = 0$. Consequently,

$$(I_n - A_0^\top A_0) z = z_\perp,$$

so

$$\|I_n - A_0^\top A_0\|_2 = \sup_{\|z\|_2=1} \|(I_n - A_0^\top A_0) z\|_2 = \sup_{\|z\|_2=1} \|z_\perp\|_2 = 1,$$

where the last equality holds because choosing $z \in \mathrm{row}(A_0)^\perp$ gives $\|z_\perp\|_2 = 1$. Combining this with the lower bound (16) shows that the minimum possible value of $\|I_n - A_0^\top A_0\|_2$ over all $A_0 \in \mathbb{R}^{r \times n}$ is 1, and it is achieved by any row-orthonormal initialization. □

B.4 Proof of Proposition 5.5 (Bounded Weight Updates)

Proof. Since A_0 has orthonormal rows, $A_0 A_0^\top = I_r$, which implies that $A_0^\top A_0$ is an orthogonal projection matrix onto the row space of A_0. In particular, $\|A_0^\top A_0\|_2 = 1$.

By Lemma B.1, the first-step LoRA update is $\Delta W_1^{\mathrm{lora}} = -\eta G_0 A_0^\top A_0$. To bound its Frobenius norm, we use the mixed-norm submultiplicativity property: for any matrices $A \in \mathbb{R}^{m \times n}$ and $B \in \mathbb{R}^{n \times p}$,

$$\|AB\|_F \le \|A\|_F \|B\|_2.$$

Applying this inequality with $A = G_0$ and $B = A_0^\top A_0$, we obtain

$$\|\Delta W_1^{\mathrm{lora}}\|_F = \eta \|G_0 A_0^\top A_0\|_F \le \eta \|G_0\|_F \|A_0^\top A_0\|_2 = \eta \|G_0\|_F.$$

□

C Proof of Theorem 4.2 (PiSSA Gradient Amplification over OLoRA)

In this section, we prove Theorem 4.2, which explains why PiSSA is more aggressive than OLoRA even when both are initialized on the same principal singular subspace. We first derive the first-step update expansions for PiSSA and OLoRA, and then compare their mode-wise coefficients directly to establish a global norm amplification result.

Lemma C.1 (PiSSA First-Step Update Expansion)**.** *For PiSSA with principal components indexed by $\mathcal{R} = \{1, \dots, r\}$, the first-step weight update satisfies*

$$\Delta W_1^{\mathrm{PiSSA}} = -\eta \sum_{i,j} c_{ij}^{\mathrm{PiSSA}} \alpha_i \beta_j \, u_i v_j^\top + \mathcal{O}(\eta^2),$$

where $c_{ij}^{\mathrm{PiSSA}} = \sigma_i \mathbf{1}_{i \in \mathcal{R}} + \sigma_j \mathbf{1}_{j \in \mathcal{R}}$, and $\alpha_i = u_i^\top g, \beta_j = v_j^\top x$, with $g = \frac{\partial \mathcal{L}}{\partial z}$ and $z = Wx$.

Proof. We expand the gradient $G = g x^\top$ in the SVD basis as

$$G = \sum_{i,j} \alpha_i \beta_j \, u_i v_j^\top, \quad \text{where} \quad \alpha_i = u_i^\top g, \ \beta_j = v_j^\top x.$$

For PiSSA with non-zero initialization on both A_0 and B_0, the chain rule gives the first-step update

$$\Delta W_1^{\text{PiSSA}} = -\eta \left(B_0 B_0^\top G + G A_0^\top A_0 \right) + \mathcal{O}(\eta^2).$$

Substituting the PiSSA initialization $B_0 = U_r \Sigma_r^{1/2}$ and $A_0 = \Sigma_r^{1/2} V_r^\top$ yields

$$B_0 B_0^\top = U_r \Sigma_r U_r^\top, \qquad A_0^\top A_0 = V_r \Sigma_r V_r^\top.$$

Hence,

$$\Delta W_1^{\text{PiSSA}} = -\eta \left(U_r \Sigma_r U_r^\top G + G V_r \Sigma_r V_r^\top \right) + \mathcal{O}(\eta^2).$$

Substituting the SVD expansion of G gives

$$\Delta W_1^{\text{PiSSA}} = -\eta \sum_{i,j} \left(\sigma_i \mathbf{1}_{i \in \mathcal{R}} + \sigma_j \mathbf{1}_{j \in \mathcal{R}} \right) \alpha_i \beta_j \, u_i v_j^\top + \mathcal{O}(\eta^2),$$

which completes the proof. □

Lemma C.2 (OLoRA First-Step Update Expansion)**.** *For OLoRA with principal components indexed by $\mathcal{R} = \{1, \ldots, r\}$, the first-step weight update satisfies*

$$\Delta W_1^{\text{OLoRA}} = -\eta \sum_{i,j} c_{ij}^{\text{OLoRA}} \alpha_i \beta_j \, u_i v_j^\top + \mathcal{O}(\eta^2),$$

where $c_{ij}^{\text{OLoRA}} = \mathbf{1}_{i \in \mathcal{R}} + \mathbf{1}_{j \in \mathcal{R}}$, and $\alpha_i = u_i^\top g, \beta_j = v_j^\top x$, with $g = \frac{\partial \mathcal{L}}{\partial z}$ and $z = Wx$.

Proof. We expand the gradient $G = g x^\top$ in the SVD basis as

$$G = \sum_{i,j} \alpha_i \beta_j \, u_i v_j^\top, \quad \text{where} \quad \alpha_i = u_i^\top g, \ \beta_j = v_j^\top x.$$

For OLoRA with non-zero initialization on both A_0 and B_0, the chain rule gives the first-step update

$$\Delta W_1^{\text{OLoRA}} = -\eta \left(B_0 B_0^\top G + G A_0^\top A_0 \right) + \mathcal{O}(\eta^2).$$

Substituting the OLoRA initialization $B_0 = U_r$ and $A_0 = V_r^\top$ yields

$$B_0 B_0^\top = U_r U_r^\top, \qquad A_0^\top A_0 = V_r V_r^\top.$$

Hence,

$$\Delta W_1^{\text{OLoRA}} = -\eta \left(U_r U_r^\top G + G V_r V_r^\top \right) + \mathcal{O}(\eta^2).$$

Substituting the SVD expansion of G gives

$$\Delta W_1^{\text{OLoRA}} = -\eta \sum_{i,j} \left(\mathbf{1}_{i \in \mathcal{R}} + \mathbf{1}_{j \in \mathcal{R}} \right) \alpha_i \beta_j \, u_i v_j^\top + \mathcal{O}(\eta^2),$$

which completes the proof. □

Lemma C.3 (PiSSA–OLoRA Coefficient Comparison)**.** *For PiSSA and OLoRA initialized on the same principal components indexed by $\mathcal{R} = \{1, \ldots, r\}$, the first-step update coefficients satisfy*

$$c_{ij}^{\text{PiSSA}} \geq \sigma_r \, c_{ij}^{\text{OLoRA}} \qquad \textit{for all } i, j.$$

Proof. By Lemma C.1 and Lemma C.2,

$$c_{ij}^{\mathrm{PiSSA}} = \sigma_i \mathbf{1}_{i\in\mathcal{R}} + \sigma_j \mathbf{1}_{j\in\mathcal{R}}, \qquad c_{ij}^{\mathrm{OLoRA}} = \mathbf{1}_{i\in\mathcal{R}} + \mathbf{1}_{j\in\mathcal{R}}.$$

We consider four cases.

If $i, j \in \mathcal{R}$, then

$$c_{ij}^{\mathrm{PiSSA}} = \sigma_i + \sigma_j \geq 2\sigma_r = \sigma_r(1+1) = \sigma_r c_{ij}^{\mathrm{OLoRA}}.$$

If $i \in \mathcal{R}$ and $j \notin \mathcal{R}$, then

$$c_{ij}^{\mathrm{PiSSA}} = \sigma_i \geq \sigma_r = \sigma_r \cdot 1 = \sigma_r c_{ij}^{\mathrm{OLoRA}}.$$

If $i \notin \mathcal{R}$ and $j \in \mathcal{R}$, then

$$c_{ij}^{\mathrm{PiSSA}} = \sigma_j \geq \sigma_r = \sigma_r \cdot 1 = \sigma_r c_{ij}^{\mathrm{OLoRA}}.$$

If $i, j \notin \mathcal{R}$, then

$$c_{ij}^{\mathrm{PiSSA}} = c_{ij}^{\mathrm{OLoRA}} = 0,$$

so the inequality also holds.

Therefore, for all i, j,

$$c_{ij}^{\mathrm{PiSSA}} \geq \sigma_r \, c_{ij}^{\mathrm{OLoRA}},$$

which completes the proof. □

Proof of Theorem 4.2. By Lemma C.1 and Lemma C.2, we may write

$$\Delta W_1^{\mathrm{PiSSA}} = -\eta \sum_{i,j} c_{ij}^{\mathrm{PiSSA}} \alpha_i \beta_j \, u_i v_j^\top + \mathcal{O}(\eta^2),$$
$$\Delta W_1^{\mathrm{OLoRA}} = -\eta \sum_{i,j} c_{ij}^{\mathrm{OLoRA}} \alpha_i \beta_j \, u_i v_j^\top + \mathcal{O}(\eta^2).$$

Since the matrices $\{u_i v_j^\top\}_{i,j}$ are orthonormal under the Frobenius inner product,

$$\left\|\Delta W_1^{\mathrm{PiSSA}}\right\|_F^2 = \eta^2 \sum_{i,j} \left(c_{ij}^{\mathrm{PiSSA}}\right)^2 \alpha_i^2 \beta_j^2 + \mathcal{O}(\eta^3),$$
$$\left\|\Delta W_1^{\mathrm{OLoRA}}\right\|_F^2 = \eta^2 \sum_{i,j} \left(c_{ij}^{\mathrm{OLoRA}}\right)^2 \alpha_i^2 \beta_j^2 + \mathcal{O}(\eta^3).$$

By Lemma C.3, for every i, j,

$$c_{ij}^{\mathrm{PiSSA}} \geq \sigma_r \, c_{ij}^{\mathrm{OLoRA}}.$$

Therefore,

$$\begin{aligned}\left\|\Delta W_1^{\mathrm{PiSSA}}\right\|_F^2 &\geq \eta^2 \sum_{i,j} \sigma_r^2 \left(c_{ij}^{\mathrm{OLoRA}}\right)^2 \alpha_i^2 \beta_j^2 + \mathcal{O}(\eta^3) \\ &= \sigma_r^2 \left\|\Delta W_1^{\mathrm{OLoRA}}\right\|_F^2 + \mathcal{O}(\eta^3).\end{aligned}$$

Taking square roots in the small-step regime yields

$$\left\|\Delta W_1^{\mathrm{PiSSA}}\right\|_F \geq \sigma_r \left\|\Delta W_1^{\mathrm{OLoRA}}\right\|_F.$$

□

Remark C.4. For pretrained LLMs, $\sigma_r \gg 1$ for moderate r (see Figure 9). This explains why PiSSA is significantly more unstable than OLoRA in RLVR (see Figure 5): even under the same principal spectral prior, PiSSA's first-step update norm exceeds OLoRA's by a factor of at least $\Omega(\sigma_r)$, making it much more likely to violate the implicit KL constraint (Theorem 4.1).

D Experimental Details

D.1 Training Setup

Model and Dataset. We conduct our main DAPO experiments on DeepSeek-R1-Distill-Qwen-1.5B. All methods are trained on the DAPO-Math-17k dataset, which contains 17,000 mathematical reasoning problems with verifiable answers. We apply LoRA adapters to all linear layers of the policy model, including the attention projections and MLP projections. Unless otherwise specified, all methods use the same base model, training data, reward function, and optimization setup.

Data Preprocessing. Training prompts use the `prompt` field from the DAPO-Math-17k parquet file. We use left truncation and set the maximum prompt length to 512 tokens. During rollout generation, the maximum response length is set to 16,384 tokens, matching the long-reasoning setting used by DAPO. We sample 8 candidate responses per prompt using temperature $\tau = 1.0$, top-$p = 1.0$, and top-$k = -1$.

Implementation. All experiments are implemented in the verl framework. We use vLLM for rollout generation and Flash Attention for efficient attention computation. Training is run on 8 NVIDIA A100-SXM4-80GB GPUs with FSDP. The rollout engine uses tensor parallelism of size 2 during training. We enable gradient checkpointing, remove-padding optimization, chunked prefill, dynamic batch sizing, actor parameter offload, actor optimizer offload, and reference parameter offload.

Hyperparameters. We use two training configurations in our DAPO 1.5B experiments. The main configuration uses a constant learning-rate schedule and serves as the primary setting for comparing LoRA, PiSSA, and MiLoRA. We additionally run cosine-decay variants to study the effect of the learning-rate schedule while keeping the remaining hyperparameters matched to the corresponding constant-LR runs. Unless otherwise stated, all methods use GRPO as the advantage estimator, LoRA-style adapters applied to all linear layers, 8 responses per prompt, and the same DAPO-style objective without an explicit KL reward penalty or actor KL loss.

In the constant-LR setting, we train for 500 optimization steps using AdamW with no warmup, a constant learning-rate schedule, weight decay 0.1, and gradient clipping at 1.0. For the standard LoRA baseline and rank-16 variants, we use learning rate 1×10^{-5}, rank $r = 16$, and scaling parameter $\alpha = 32$, corresponding to an effective scaling factor of $\alpha/r = 2$. For the original PiSSA and MiLoRA setting, we follow the commonly used configuration with learning rate 2×10^{-5}, rank $r = 32$, and $\alpha = 32$. LoRA dropout is set to 0.0 for all methods. The clipping range is asymmetric, with lower clip ratio 0.2 and upper clip ratio 0.28. The prompt batch size is 128, with a PPO mini-batch size of 32.

In the cosine-decay setting, we keep the optimizer, warmup, weight decay, batch size, rollout configuration, and adapter configuration matched to the corresponding constant-LR run, but replace the constant schedule with cosine decay. In our 1.5B cosine runs, the initial learning rate is 1×10^{-5}, warmup is 0, and weight decay is 0.1.

Evaluation Protocol. We evaluate on GSM8K, MATH500, and AIME 2022–2025. For GSM8K, we report pass@1 with greedy decoding. For MATH500, we report pass@4 with temperature sampling. For AIME, we report pass@32. Unless otherwise stated, evaluation uses temperature $\tau = 0.6$, top-$p = 0.95$, top-$k = -1$, maximum prompt length 1024, and maximum response length 16,384. GSM8K uses greedy decoding with $\tau = 0$ and maximum response length 4096. MATH500 uses maximum response length 8192. AIME uses maximum response length 16,384.

For answer scoring, we use the DAPO math reward implementation provided in verl. GSM8K is scored with flexible numerical extraction to avoid undercounting correct answers that do not follow the strict `####` format. For MATH500 and AIME, we use the default mathematical answer normalization and exact-match scoring implemented by the DAPO reward function.

D.2 Evaluation Benchmarks

We evaluate all methods on six mathematical reasoning benchmarks:

- **GSM8K** [Cobbe et al., 2021]: Grade-school math word problems requiring multi-step arithmetic reasoning. We use the full test set of 1,319 examples and report pass@1 with greedy decoding.
- **MATH500** [Hendrycks et al., 2021]: A 500-problem subset of the MATH benchmark covering algebra, geometry, counting, probability, number theory, and precalculus. We report pass@4.
- **AIME 2022/2023/2024/2025**: Competition-level mathematical reasoning problems from the American Invitational Mathematics Examination. Each year contains 30 problems. We report pass@32.

Table 4: Evaluation settings for the DAPO 1.5B experiments.

Bendchmark	**Metric**	**Samples**	**Temperature**	**Top-p**	**Max response length**
GSM8K	pass@1	1	0	1.0	4096
MATH500	pass@4	4	0.6	0.95	8192
AIME 2022–2025	pass@32	32	0.6	0.95	16384

E Additional Experiments

E.1 Generalization to Supervised Fine-Tuning (SFT)

To investigate whether the proposed geometry-preserving initializations generalize beyond RL fine-tuning, we conduct Supervised Fine-Tuning (SFT) experiments on Qwen2.5-7B-Instruct. We evaluate the methods across two distinct benchmark categories:

- **GLUE** (CoLA and MRPC): Classification tasks evaluating linguistic acceptability and paraphrase detection.
- **GSM8K**: Grade school math reasoning, evaluated using strict exact match (via the `lm-eval` harness, `gsm8k_cot` task).

As shown in Table 5, both LoRA-RLPO and LoRA-RLMO generalize effectively to the SFT paradigm, consistently outperforming standard LoRA across all three benchmarks.

Table 5: SFT evaluation results on Qwen2.5-7B-Instruct. Results are reported as mean ± standard deviation across 3 random seeds.

Task	**LoRA**	**LoRA-RLPO**	**LoRA-RLMO**
CoLA (acc.)	85.46 ± 0.22	86.42 ± 0.24	$\mathbf{86.48 \pm 0.50}$
MRPC (acc.)	86.52 ± 0.25	$\mathbf{88.48 \pm 0.25}$	87.91 ± 0.51
GSM8K (strict)	23.96 ± 8.89	29.74 ± 0.54	$\mathbf{33.61 \pm 2.99}$

Training Dynamics. Figure 10 illustrates the training loss curves across the three SFT tasks. Both LoRA-RLPO and LoRA-RLMO consistently converge faster and reach a lower final loss compared to standard LoRA. Notably, this performance gap is most pronounced on the GSM8K reasoning task.

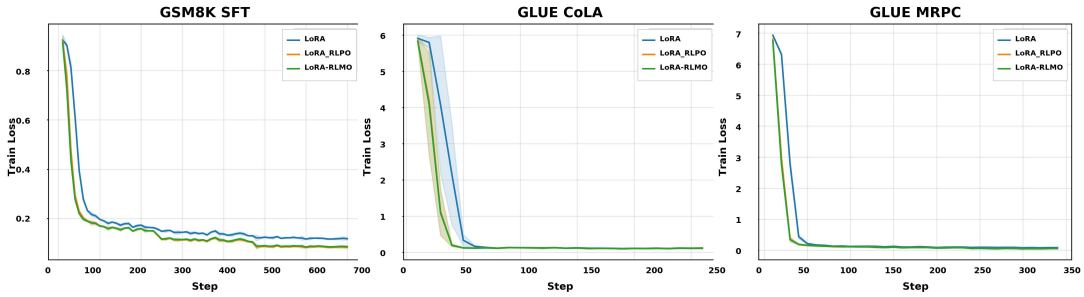

Figure 10: Train loss curves across three SFT tasks (3 seeds, shaded region denotes ±1 standard deviation). Both LoRA-RLPO and LoRA-RLMO demonstrate faster convergence and lower final loss than standard LoRA.

Experimental Setup. For all SFT experiments, we use rank $r = 32$, $\alpha = 64$, and a constant learning rate of 1×10^{-5}. Models are trained for 3 epochs with a global batch size of 32 (4 per device $\times$ 8 gradient accumulation steps). All evaluations are averaged across three random seeds ($\{1, 42, 123\}$).

E.2 SVD Preprocessing Cost

Table 6 reports the SVD preprocessing cost for different model scales. The measurements are conducted with rank $r = 32$ using bfloat16 precision on a single NVIDIA A100 GPU. Note that LoRA-RLPO and LoRA-RLMO have identical SVD costs. This preprocessing is a one-time cost incurred before all subsequent training steps.

Table 6: SVD preprocessing cost evaluated with $r = 32$, bfloat16 precision, on a single A100 GPU.

Model	Parameters	Wall-Clock Time	Peak GPU Memory
Qwen3-4B-Instruct	4B	1.8 min	8.3 GiB
Qwen2.5-7B-Instruct	7B	3.5 min	16.0 GiB
Qwen2.5-14B-Instruct	14B	12.3 min	29.8 GiB

E.3 Task Domain and Model Family Diversity.

We have added experiments on Llama 3.2-3B-Instruct (different family) and Qwen2.5-1.5B-Instruct (different scale). To test domain generality, we also evaluate on code generation: Llama 3.2-3B-Instruct on MBPP-style program synthesis with test-case-based reward. LoRA-RLPO remains stable and effective under this different reward structure.

Table 7: Code generation on Llama 3.2-3B-Instruct (MBPP-style, test-case reward).

LoRA	LoRA-RLPO(Ours)	LoRA-RLMO(Ours)
43.44 ± 3.10	$\mathbf{45.67 \pm 1.41}$	46.11 ± 1.26

Our theory is loss-agnostic and applies to any RLVR objective with KL constraints.

E.4 Larger Model Size

We fine-tune Qwen2.5-7B-Instruct using GRPO [Shao et al., 2024] on DAPO-Math-17k [Yu et al., 2025] with rank $r = 32$ LoRA applied to all linear layers.

Hyperparameters. For the 7B model experiments, we train for 150 optimization steps using the AdamW optimizer with a constant learning rate schedule and no warmup. To ensure a fair comparison, we use a learning rate of 1×10^{-5} and scaling factor $\alpha = 64$ for standard LoRA, LoRA-RLPO, and LoRA-RLMO. For PiSSA and MiLoRA, we adopt a learning rate of 2×10^{-5} and $\alpha = 32$, following their standard tuning practices. We set the effective batch size to 32 (4 prompts per batch with 8 responses sampled per prompt) and use a KL penalty coefficient of $\beta = 0.001$ for the GRPO objective.

As shown in Table 8, our proposed geometry-preserving initializations maintain their empirical advantages at a larger scale. Consistent with our observations on the 1.5B model, the SVD-based variants PiSSA and MiLoRA struggle under the strict KL constraints of RLVR, yielding average performances (24.74 and 26.72) that are substantially inferior to standard LoRA (30.39). In contrast, both LoRA-RLMO and LoRA-RLPO maintain stable optimization dynamics and achieve consistent performance improvements. Notably, LoRA-RLPO achieves the highest average score of **35.96** across all mathematical reasoning benchmarks, with significant gains on GSM8K and AIME evaluations. These results demonstrate that our theoretically motivated initializations scale robustly to 7B-parameter models without requiring additional hyperparameter tuning.

	LoRA	MiLoRA	LoRA-RLMO (Ours)	PiSSA	LoRA-RLPO (Ours)
B_0	$\mathbf{0}$	$U_{-r}\Sigma_{-r}^{1/2}$	$\mathbf{0}$	$U_r\Sigma_r^{1/2}$	$\mathbf{0}$
A_0	$\mathcal{N}(0, \frac{1}{n})$	$\Sigma_{-r}^{1/2}V_{-r}^\top$	$V_{-r}^\top$	$\Sigma_r^{1/2}V_r^\top$	$V_r^\top$
GSM8K$^{@1}$	79.13±4.6	57.27±12.9	74.30±12.3	55.60±44.0	**85.29**±2.4
MATH500$^{@4}$	52.80±2.3	54.13±1.5	**56.73**±1.3	50.33±1.1	55.60±2.6
AIME22$^{@16}$	3.33±2.7	2.22±1.9	7.78±1.6	4.44±1.9	**7.78**±1.6
AIME23$^{@16}$	11.11±4.2	11.11±1.9	10.00±0.0	6.67±3.3	**13.33**±2.7
AIME24$^{@16}$	5.56±1.6	8.89±1.9	13.33±2.7	6.67±3.3	**17.78**±6.8
Avg	30.39	26.72	32.42	24.74	**35.96**

Table 8: Evaluation results of Qwen2.5-7B-Instruct fine-tuned with GRPO on DAPO-Math-17k. Our proposed LoRA-RLPO and LoRA-RLMO initializations outperform standard LoRA, whereas PiSSA and MiLoRA exhibit performance degradation.

E.5 Learning Rate Sensitivity

We conduct a learning rate sweep on Qwen-2.5-7B-Instruct to evaluate the robustness of different initialization methods. Figure 11 shows the average accuracy across learning rates $\{10^{-6}, 10^{-5}, 10^{-4}\}$. LoRA-RLPO and LoRA-RLMO consistently outperform standard LoRA across all learning rates. All methods achieve peak performance at 10^{-5}, which we adopt as the learning rate for all experiments reported in this paper.

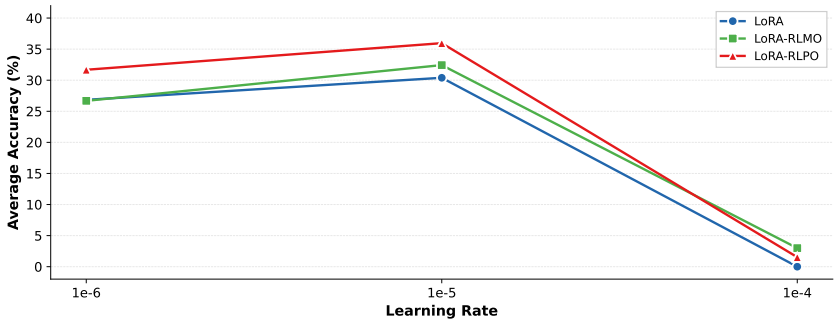

Figure 11: Learning rate sensitivity comparison. LoRA-RLPO and LoRA-RLMO consistently outperform standard LoRA across learning rates, with peak performance at 10^{-5}.

Table 3: Hyperparameters for the DAPO 1.5B experiments.

Hyperparameter	**Constant-LR setting**	**Cosine-decay setting**
Model and Software		
Base model	DeepSeek-R1-Distill-Qwen-1.5B	DeepSeek-R1-Distill-Qwen-1.5B
Training framework	verl 0.7.0.dev	verl 0.7.0.dev
Inference engine	vLLM 0.11.0	vLLM 0.11.0
Flash Attention	2.8.1	2.8.1
PyTorch	2.8.0+cu126	2.8.0+cu126
Hardware	8 × NVIDIA A100-SXM4-80GB	8 × NVIDIA A100-SXM4-80GB
Optimization		
Optimizer	AdamW	AdamW
Learning rate	1×10^{-5}	1×10^{-5}
Learning rate schedule	Constant	Cosine decay
Warmup steps	0	0
Weight decay	0.1	0.1
Gradient clipping	1.0	1.0
Training steps	500	500
Batch Size		
Prompt batch size	128	128
PPO mini-batch size	32	32
Responses per prompt	8	8
GRPO / DAPO		
Advantage estimator	GRPO	GRPO
KL reward coefficient	0.0	0.0
Actor KL loss coefficient	0.0	0.0
Clip ratio lower / upper	0.2/0.28	0.2/0.28
Clip ratio c	10.0	10.0
Loss aggregation	Token mean	Token mean
Entropy coefficient	0	0
Overlong buffer length	4096	4096
Overlong penalty factor	1.0	1.0
Adapter Configuration		
Adapter type	LoRA-style	LoRA-style
Rank (r)	16 or 32	16
Alpha (α)	32	32
Target modules	All linear layers	All linear layers
Dropout	0.0	0.0
Bias	None	None
Training Rollout Generation		
Max prompt length	512	512
Max response length	16384	16384
Temperature	1.0	1.0
Top-p	1.0	1.0
Top-k	-1	-1
Rollout tensor parallel size	2	2
Rollout GPU memory utilization	0.75	0.75
Chunked prefill	Enabled	Enabled