

GRAIL: Gradient-Reward Aligned Initialization for LoRA-based GRPO

Liwen Shi¹, Zijie Zeng¹, Bowen Wang¹, Yunji¹, Zeyu Chen¹,
Xinyu Shao¹, Xinzheng Sun¹, Xuyang Xiong¹, Haoyang Wu¹, Sheng Zhang^{1,†}

¹Tsinghua Shenzhen International Graduate School

Correspondence: slw24@mails.tsinghua.edu.cn, zhang_sh@mail.tsinghua.edu.cn

Abstract

Group Relative Policy Optimization (GRPO) improves LLM reasoning, but LoRA-based GRPO can underperform when the initialized adapter subspace is poorly aligned with reward-induced gradients. We propose GRAIL, Gradient-Reward Aligned Initialization for LoRA-based GRPO, which estimates reward-aligned gradients from grouped rollouts before policy optimization and uses them to construct LoRA factors. The full GRAIL configuration includes a reference-solution fallback only for all-zero-correctness groups during calibration; after initialization, all models use the standard GRPO objective without extra trainable parameters or loss changes. We also study Degenerate-Group Reflection Resampling (DGRR) as an optional extension that resamples all-zero-correctness groups after self-reflection without exposing reference solutions. Across Qwen2.5, Phi-3.5, and Gemma-2 backbones on GSM8K and MATH500, GRAIL consistently improves LoRA-based GRPO, with particularly strong gains on smaller Qwen and Gemma models. A verifiable code-generation check further improves MBPP and matches LoRA-GRPO on HumanEval. Analyses attribute these gains to stronger GRPO-gradient capture, smaller principal angles, and better handling of correctness-degenerate groups.

1 Introduction

Large language models (LLMs) have made strong progress in reasoning and generation (Vaswani et al., 2017; Devlin et al., 2019; Raffel et al., 2020; Brown et al., 2020; OpenAI, 2023; Touvron et al., 2023), especially when optimized with reinforcement learning. Group Relative Policy Optimization (GRPO) is a practical alternative to PPO-style training for reasoning tasks because it normalizes rewards across multiple responses to the same

prompt and avoids a separate value model (Schulman et al., 2017; Shao et al., 2024). Yet GRPO remains expensive when all model parameters are updated.

A natural solution is to combine GRPO with parameter-efficient fine-tuning methods such as Low-Rank Adaptation (LoRA) (Hu et al., 2022). LoRA freezes the pretrained model and trains only low-rank adapter parameters, but direct LoRA-GRPO can still underperform. In our preliminary runs, the initialized adapter subspace preserved little reward-induced gradient energy, and increasing rank alone did not consistently close this gap. This raises a central question: why does an efficient method that works well in supervised fine-tuning become less effective under reward-driven policy optimization?

We argue that the key issue is a mismatch between the initialized LoRA adapter subspace and GRPO-induced gradients. Existing LoRA initialization methods optimize for pretrained-weight structure or supervised gradients, but RL policy optimization induces fundamentally different gradient geometry. We show that this mismatch substantially limits LoRA-based GRPO: if the adapter subspace is poorly aligned with policy gradients induced by sampled responses, rewards, and group-relative advantages, early GRPO may miss important reward-improving directions. Thus, the bottleneck is not only rank, but whether the initial adapter subspace is aligned with reward-induced update directions. Figure 1 illustrates this reward-specific subspace view.

To address this issue, we propose GRAIL, Gradient-Reward Aligned Initialization for LoRA-based GRPO. GRAIL is a reward-aware initialization framework that estimates policy gradients from grouped GRPO rollouts before training. Its goal is not continual subspace tracking, but one-shot extraction of stable early reward geometry while retaining low-rank regular-

[†]Corresponding author.

GRAIL Finds Reward-specific Adapter Subspaces

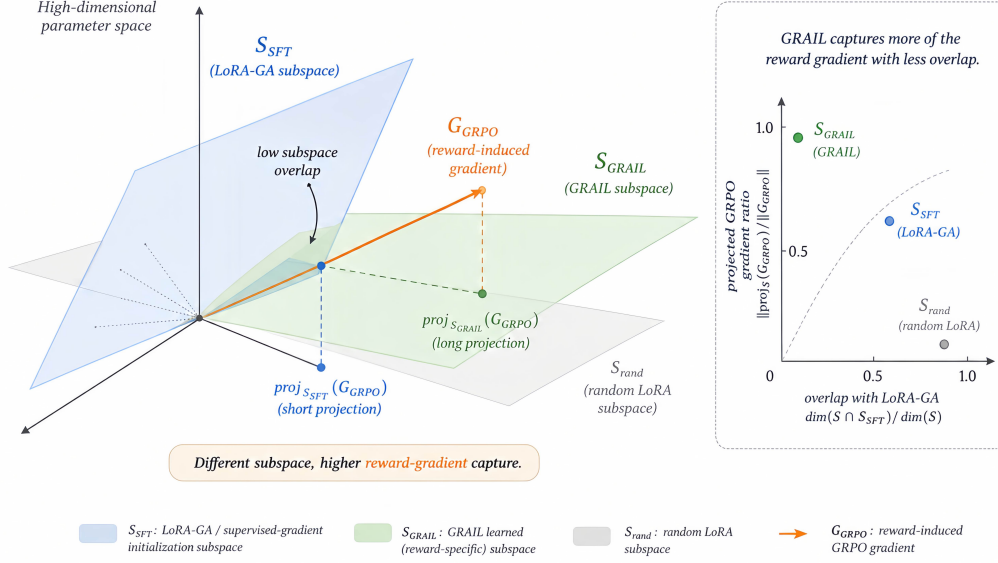


Figure 1: Conceptual illustration of reward-specific adapter subspaces. Compared with random or supervised-gradient initialization, GRAIL seeks an adapter subspace that preserves more of the GRPO-induced reward-gradient direction while remaining distinct from subspaces optimized for supervised adaptation. This figure is intended as a conceptual illustration of the initialization geometry.

ization. Specifically, GRAIL samples multiple responses for each prompt, computes rewards and group-relative advantages, backpropagates the resulting policy-gradient signal to target modules, and constructs LoRA factors from the estimated reward-induced gradients. GRAIL also includes an optional reference-solution fallback for all-zero-correctness groups (Section 3.2). After initialization, the model is trained with the standard GRPO objective and does not require additional trainable parameters or modifications to the training loss.

This one-shot use of full-gradient information separates GRAIL from prolonged full-parameter policy optimization. Full-parameter GRPO can expose useful reward-induced directions early in training, but sustained unconstrained updates may drift away from the stable subspace useful for parameter-efficient adaptation. GRAIL instead uses the early reward geometry as an initialization signal while preserving the stability and implicit regularization of low-rank adaptation.

GRAIL targets the starting geometry of LoRA-based GRPO, while correctness degeneracy can still occur during subsequent policy optimization. We therefore treat Degenerate-Group Reflection Resampling (DGRR) as an optional training-time

extension: when every sampled completion in a group receives zero correctness reward, DGRR skips the uninformative group, asks the current policy to reflect using only binary correctness feedback, and resamples before applying the standard GRPO loss formula. It changes the sampling and update distribution but adds no trainable parameters, exposes no reference solutions, and does not alter the per-group loss formula.

We evaluate GRAIL on mathematical reasoning tasks using Qwen2.5-1.5B-Instruct and Qwen2.5-3B-Instruct (Yang et al., 2024a), and further include Phi-3.5-mini-instruct (Abdin et al., 2024) and Gemma-2-2B-it (Gemma Team, 2024) as non-Qwen model-family checks. Models are trained on GSM8K and evaluated on both GSM8K test and MATH500 (Lightman et al., 2024). We also test Qwen2.5-3B-Instruct on verifiable code generation using execution-based rewards and evaluate on MBPP (Austin et al., 2021) and HumanEval (Chen et al., 2021). Compared with vanilla LoRA, LoRA-GA (Wang et al., 2024b), and PiSSA (Meng et al., 2024), GRAIL obtains the best results in our comparison. Beyond final accuracy, we analyze whether the initialized adapter subspace captures the reward-gradient directions used by GRPO. The analysis links GRAIL’s gains

to projected gradient energy, principal angles, and layer/module-level behavior.

Our contributions are summarized as follows:

- We show that standard LoRA initialization can preserve very little GRPO-induced gradient energy, leading to a subspace-gradient mismatch in LoRA-based GRPO.
- We formulate reward-aware LoRA initialization as a constrained subspace construction problem for policy optimization and propose GRAIL, which estimates GRPO-induced gradients from grouped rollouts and uses the resulting signal to initialize LoRA adapters.
- As an optional extension, we study DGRR for all-zero-correctness GRPO groups through reflection-based resampling while keeping the GRPO loss unchanged and avoiding reference-solution exposure.
- We demonstrate the effectiveness of GRAIL on mathematical reasoning and provide a verifiable code-generation check, together with mechanistic analyses showing improved projected gradient capture across layers and modules.

2 Related Work

Reinforcement learning for reasoning LLMs.

Reinforcement learning from feedback has become a standard way to improve LLM behavior. Early and instruction-following work studies reward modeling and policy optimization from human or model preferences (Christiano et al., 2017; Ziegler et al., 2019; Stiennon et al., 2020; Ouyang et al., 2022; Bai et al., 2022a), while later preference-optimization objectives reduce or replace parts of the RL pipeline (Rafailov et al., 2023; Ethayarajh et al., 2024). Related web and constitutional-feedback systems extend the feedback source and training protocol (Nakano et al., 2021; Bai et al., 2022b). These works mainly study where the reward signal comes from and how the policy objective is optimized; our focus is the trainable subspace through which such objectives update the model.

For mathematical reasoning, prior work has improved performance through prompting and sampling strategies such as chain-of-thought reasoning, zero-shot reasoning, and self-consistency (Wei et al., 2022; Kojima et al.,

2022; Wang et al., 2023); through search or decomposition methods (Zhou et al., 2023; Yao et al., 2023); and through self-improvement, domain-specific training data, or tool-augmented reasoning (Zelikman et al., 2022; Yu et al., 2024; Luo et al., 2025; Gou et al., 2024; Yang et al., 2024b). Process- and outcome-based verifiers provide automatic reward signals for this setting (Uesato et al., 2022; Wang et al., 2024a; Lightman et al., 2024). GRPO (Shao et al., 2024) and recent extensions such as DAPO (Yu et al., 2025) make reward-driven reasoning training practical by using group-relative advantages and scalable sampling strategies. GRAIL is complementary to these recipes: it studies how the LoRA adapter should be initialized before GRPO begins.

Parameter-efficient fine-tuning with LoRA.

Parameter-efficient fine-tuning (PEFT) adapts pre-trained models while updating only a small subset of parameters. Adapter-based methods insert trainable modules into the network (Houlsby et al., 2019; Pfeiffer et al., 2020), prompt- and prefix-tuning methods optimize continuous task prompts (Li and Liang, 2021; Lester et al., 2021; Liu et al., 2022), and bias-only or intrinsic-subspace methods restrict adaptation even further (Ben Zaken et al., 2022; Aghajanyan et al., 2021). Unified and compressed PEFT variants combine these design choices or reduce their parameter cost (He et al., 2022; Mao et al., 2022; Mahabadi et al., 2021). These methods motivate training in a restricted parameter subspace, but they are not designed around the geometry of reward-induced policy gradients.

LoRA (Hu et al., 2022) freezes the pretrained weights and injects trainable low-rank matrices into selected linear layers. Subsequent variants improve LoRA in different ways: QLoRA and LoftQ focus on quantization-aware low-rank adaptation (Dettmers et al., 2023; Li et al., 2024); AdaLoRA and rsLoRA adjust rank allocation or scaling (Zhang et al., 2023; Kalajdziewski, 2023); VeRA and DoRA change the basis or decomposition used for adaptation (Kopiczko et al., 2024; Liu et al., 2024); and related analyses study the effect of LoRA initialization and scaling (Hayou et al., 2024). PiSSA initializes adapters from the principal singular directions of pretrained weights (Meng et al., 2024), while LoRA-GA uses supervised full-gradient information to construct gradient-aware adapters (Wang et al., 2024b).

GRAIL is related to the broader idea that initialization should preserve useful update directions, but it targets a different optimization signal: the group-relative policy gradient induced by GRPO rewards.

LoRA initialization and subspace alignment.

LoRA initialization can affect both convergence and final performance because it determines the update directions available at the start of training. Standard LoRA keeps the initial model output unchanged by initializing one factor randomly and the other as zero, but this choice does not encode information about the downstream optimization signal. Existing initialization methods therefore use either pretrained weight geometry, as in PiSSA, or supervised gradients, as in LoRA-GA. In contrast, LoRA-based GRPO is driven by sampled responses, rewards, and group-relative advantages. We therefore formulate initialization as a reward-conditioned subspace alignment problem for parameter-efficient reinforcement learning, rather than as a supervised warm start transferred unchanged to an RL objective.

3 Method

Figure 2 gives a schematic view of GRAIL as a one-time initialization stage before LoRA-based GRPO training. It estimates initialization gradients from grouped rollouts, constructs LoRA factors from these gradients, and then trains the initialized adapters with the standard GRPO objective.

3.1 Subspace-Gradient Mismatch

LoRA reduces training cost by restricting updates to the subspace induced by the adapter factors. For a target weight matrix W_ℓ , let $G_{\text{GRPO},\ell}$ denote a calibration-batch full-gradient proxy of the GRPO initialization loss with respect to the frozen weight W_ℓ , obtained by temporarily enabling gradients on W_ℓ while keeping the backbone frozen. Ideally, the LoRA subspace should preserve important components of $G_{\text{GRPO},\ell}$ so that parameter-efficient updates can follow reward-improving directions. Standard LoRA initialization does not use this GRPO-gradient geometry, which can create a *subspace-gradient mismatch*.

We quantify this mismatch by projecting $G_{\text{GRPO},\ell}$ onto the linearized LoRA update sub-

space $\mathcal{S}_\ell$ and define

$$\eta(\mathcal{S}_\ell, G_{\text{GRPO},\ell}) = \frac{\|P_{\mathcal{S}_\ell}(G_{\text{GRPO},\ell})\|_F^2}{\|G_{\text{GRPO},\ell}\|_F^2}, \quad (1)$$

where $P_{\mathcal{S}_\ell}$ is the Frobenius-orthogonal projection. For initialized factors B_ℓ and A_ℓ ,

$$\mathcal{S}_\ell = \{\Delta B_\ell A_\ell + B_\ell \Delta A_\ell\}.$$

We compute $P_{\mathcal{S}_\ell}$ using orthonormal bases U_B and V_A for the column space of B_ℓ and row space of A_ℓ :

$$P_{\mathcal{S}_\ell}(G) = U_B U_B^\top G + G V_A V_A^\top - U_B U_B^\top G V_A V_A^\top.$$

A larger η means that the adapter can preserve more reward-gradient energy. It also has a first-order interpretation: for a local reward objective J_R with gradient g_R , the best small update within a LoRA subspace $\mathcal{S}$ satisfies

$$\max_{\Delta \in \mathcal{S}, \|\Delta\| \leq \epsilon} \langle g_R, \Delta \rangle = \epsilon \|P_{\mathcal{S}} g_R\|.$$

Thus, subspace-gradient mismatch is a projection loss on reward-improving directions. Supervised-gradient alignment reduces this loss only when supervised and reward-induced gradients agree, which need not hold under GRPO. We use η in Section 5 as a layer-wise estimate of reachable first-order reward-gradient capacity.

3.2 Reward-Induced Gradient Estimation

GRAIL first estimates reward-aligned initialization gradients before GRPO training. Given a small calibration set of prompts, we perform grouped rollouts using the current model. For each prompt q , we sample G responses, compute their rewards, and obtain group-relative advantages $\hat{A}_i$.

We then construct a policy-gradient loss for initialization:

$$\mathcal{L}_{\text{init}} = -\frac{1}{G} \sum_{i=1}^G \hat{A}_i \frac{1}{|o_i|} \sum_{t=1}^{|o_i|} \log \pi_\theta(o_{i,t} \mid q, o_{i,<t}). \quad (2)$$

The factor $1/|o_i|$ length-normalizes each response-level log-probability term. If all sampled responses receive zero correctness reward, the group has no correctness-based preference signal. GRAIL then ignores format-only differences for that group and replaces the degenerate correctness-gradient component with a length-normalized SFT-style gradient on the reference

4 Experiments

4.1 Experimental Setup

Models and data. We use Qwen2.5-1.5B-Instruct, Qwen2.5-3B-Instruct, Phi-3.5-mini-instruct, and Gemma-2-2B-it (Yang et al., 2024a; Abdin et al., 2024; Gemma Team, 2024). We train on GSM8K (Cobbe et al., 2021) and evaluate on GSM8K test and MATH500 (Lightman et al., 2024). LoRA-based methods freeze the backbone and train adapters on the target linear layers.

Training recipe and baselines. All mathematical-reasoning GRPO experiments use Swift RLHF with rule-based accuracy and format rewards, 8 completions per prompt, temperature 0.9, one training epoch, bf16 precision, and DeepSpeed ZeRO-2 on 8 GPUs; details are in Appendix A. We compare GRAIL with the base model, vanilla LoRA-GRPO, LoRA-GA, and PiSSA under the same target modules, rank, training data, reward functions, rollout configuration, optimizer, and post-initialization GRPO schedule unless explicitly varied. We also report GRAIL+DGRR, which uses the same initialized adapters as GRAIL but applies reflection-based resampling only to all-zero-correctness groups during GRPO. To isolate confounds, we include an SFT-style warm start with the same calibration prompts and gradient-extraction pipeline, a shuffled-reward initialization with the same rollout token budget and low-rank construction, and a longer LoRA-GRPO run with matched extra compute. The code-domain check retains the optimization and rollout settings but replaces the training data and correctness reward with a 10,000-problem subset of the verifiable Python dataset and test-case execution.

Metrics. For trained mathematical-reasoning methods, we report mean accuracy over three seeds with sample standard deviation in the main and control comparisons; the compact hyperparameter sweep reports seed means without standard deviations. Base models are evaluated once with deterministic decoding. The code-domain check reports pass@1 point estimates without across-seed dispersion. GSM8K has 1319 test examples and MATH500 has 500, so single-run accuracies follow the corresponding test-set granularity.

4.2 Main Results

Table 1 shows that vanilla LoRA-GRPO stays close to the base model, whereas GRAIL improves LoRA-based GRPO across all evaluated backbones. The reward-only GRAIL variant without fallback already improves both Qwen backbones on both benchmarks; the full hybrid variant adds a smaller gain by rescuing correctness-degenerate calibration groups. The gains are especially clear on the smaller Qwen and Gemma models and also transfer to MATH500, suggesting that reward-aware initialization helps the adapter subspace capture directions that generalize beyond the GSM8K training distribution. Standard supervised-gradient or weight-SVD initializations are less consistent: LoRA-GA and PiSSA can help in some cases, but they often remain near vanilla LoRA or underperform on the harder transfer benchmark. GRAIL+DGRR usually provides an additional gain, although its effect is not uniformly positive across all model–benchmark pairs.

4.3 Generality to Verifiable Code Generation

We test Qwen2.5-3B-Instruct on a 10,000-problem subset of the [open-r1 verifiable-coding-problems-python dataset](#), using test-case execution as the correctness reward. We retain the math experiments’ optimization and rollout settings while replacing the task-specific data and reward. We evaluate pass@1 on MBPP (Austin et al., 2021) and HumanEval (Chen et al., 2021). Table 2 shows that GRAIL obtains the best MBPP score, improving LoRA-GRPO by 1.55 points, and matches LoRA-GRPO on HumanEval. LoRA-GA falls below the untrained base model on both benchmarks, showing that supervised-gradient initialization is not consistently beneficial in this code setting. This single-model verifier-based experiment supports transfer beyond mathematics, but does not establish generality to learned or subjective rewards.

4.4 Vanilla LoRA-GRPO Hyperparameter Check

Table 3 sweeps learning rate, LoRA rank, and KL coefficient for Qwen2.5-1.5B under the same data, rewards, rollout configuration, optimizer, and schedule. Vanilla LoRA-GRPO remains close to the base model, and its best tuned variant still does not match GRAIL, indicating that GRAIL’s gain is not due to one unfavorable baseline setting.

Model	Method	GSM8K (%)	MATH500 (%)
Qwen2.5-1.5B	Base	70.81	52.2
Qwen2.5-1.5B	LoRA	70.31 ± 0.18	51.6 ± 0.4
Qwen2.5-1.5B	LoRA-GA	67.85 ± 0.30	41.8 ± 0.7
Qwen2.5-1.5B	PiSSA	72.10 ± 0.30	49.0 ± 0.4
Qwen2.5-1.5B	GRAIL w/o fallback	72.16 ± 0.21	54.2 ± 0.5
Qwen2.5-1.5B	GRAIL	72.63 ± 0.27	55.0 ± 0.6
Qwen2.5-1.5B	GRAIL+DGRR	73.09 ± 0.26	55.6 ± 0.4
Qwen2.5-3B	Base	85.14	66.0
Qwen2.5-3B	LoRA	85.06 ± 0.20	65.0 ± 0.5
Qwen2.5-3B	LoRA-GA	85.60 ± 0.26	64.8 ± 0.7
Qwen2.5-3B	PiSSA	85.44 ± 0.23	64.6 ± 0.4
Qwen2.5-3B	GRAIL w/o fallback	87.48 ± 0.17	67.6 ± 0.4
Qwen2.5-3B	GRAIL	88.02 ± 0.15	68.2 ± 0.2
Qwen2.5-3B	GRAIL+DGRR	88.40 ± 0.15	68.8 ± 0.3
Phi-3.5-mini	Base	86.50	49.6
Phi-3.5-mini	LoRA	86.43 ± 0.18	49.1 ± 0.4
Phi-3.5-mini	LoRA-GA	86.91 ± 0.22	48.8 ± 0.5
Phi-3.5-mini	PiSSA	86.76 ± 0.20	48.9 ± 0.4
Phi-3.5-mini	GRAIL	89.69 ± 0.16	51.2 ± 0.3
Phi-3.5-mini	GRAIL+DGRR	89.92 ± 0.14	51.0 ± 0.4
Gemma-2-2B	Base	61.49	24.0
Gemma-2-2B	LoRA	61.79 ± 0.15	24.4 ± 0.2
Gemma-2-2B	LoRA-GA	63.03 ± 0.16	26.0 ± 0.2
Gemma-2-2B	PiSSA	66.31 ± 0.19	29.6 ± 0.2
Gemma-2-2B	GRAIL	66.94 ± 0.23	30.2 ± 0.2
Gemma-2-2B	GRAIL+DGRR	67.83 ± 0.19	31.2 ± 0.2

Table 1: Main GSM8K and MATH500 results. Trained methods are averaged over three seeds; the base model is evaluated once. GRAIL w/o fallback uses reward-induced gradients only. Full GRAIL uses an SFT-style fallback only for all-zero-correctness calibration groups; GRAIL+DGRR additionally applies reflection-based resampling during GRPO.

Method	MBPP	HumanEval
Base	39.30	57.93
LoRA-GRPO	39.69	58.54
LoRA-GA	38.52	56.10
GRAIL	41.24	58.54

Table 2: Verifiable code-generation results (pass@1, %) with Qwen2.5-3B-Instruct.

Sweep	Setting	GSM8K (%)	MATH500 (%)
Default	$\text{lr}=1 \times 10^{-6}, r = 16, \text{KL}=0.05$	70.31	51.6
LR	5×10^{-7}	70.74	51.9
LR	2×10^{-6}	70.48	51.7
LR	5×10^{-6}	69.80	51.0
LR	1×10^{-5}	69.12	50.4
Rank	$r = 8$	70.24	51.4
Rank	$r = 32$	71.02	52.0
Rank	$r = 64$	70.67	51.8
KL	0	69.72	50.9
KL	0.01	70.85	51.9
KL	0.10	70.55	51.7
KL	0.20	70.03	51.3

Table 3: Small hyperparameter sweep for Qwen2.5-1.5B vanilla LoRA-GRPO.

4.5 Initialization and Compute Controls

Table 4 reports Qwen2.5-3B controls under the same protocol. The SFT-style warm start matches GRAIL’s calibration and factor-construction pipeline but uses supervised likelihood gradients, showing that supervised calibration alone is insufficient. Shuffled-reward initialization preserves rollout budget and SVD construction but removes reward-gradient alignment, while longer LoRA-GRPO adds compute without closing the gap.

4.6 Early Reward Geometry in Full-Parameter GRPO

We further examine full-parameter GRPO on Qwen2.5-3B under the same reward functions, rollout configuration, and data source. Figure 6 in Appendix D.3 shows that training reward rises rapidly from about 0.65 to the mid-0.8 range, indicating early reward-improving directions. However, Table 11 shows that the best tuned full-parameter GRPO model obtains only 83.40% GSM8K and 62.40% MATH500, below both the base model and GRAIL-LoRA. This suggests a

Method	GSM8K (%)	MATH500 (%)
SFT-style init	85.65 ± 0.13	64.33 ± 0.42
Shuffled-reward init	80.34 ± 0.26	58.27 ± 0.50
Longer LoRA-GRPO	85.04 ± 0.12	65.07 ± 0.23
GRAIL	88.02 ± 0.15	68.2 ± 0.2

Table 4: Control baselines for Qwen2.5-3B. The SFT-style control matches GRAIL’s calibration prompts and gradient-extraction pipeline but uses supervised likelihood gradients; the shuffled-reward control matches GRAIL’s rollout token budget and low-rank construction but removes reward-gradient alignment by permuting advantages. Results are averaged over three training seeds.

Initialization	DGRR	GSM8K	MATH500
LoRA	No	70.31 ± 0.18	51.6 ± 0.4
LoRA	Yes	70.69 ± 0.15	51.8 ± 0.2
GRAIL	No	72.63 ± 0.27	55.0 ± 0.6
GRAIL	Yes	73.09 ± 0.26	55.6 ± 0.4

Table 5: A 2×2 ablation of initialization and DGRR on Qwen2.5-1.5B.

separation between early reward-geometry discovery and sustained unconstrained policy optimization: GRAIL uses the signal conservatively by extracting reward-aligned directions once and then training a stable low-rank adapter.

4.7 Ablation on the SFT Fallback

The no-fallback rows in Table 1 isolate the reward-only component: even without reference-solution gradients, GRAIL remains clearly better than vanilla LoRA on both Qwen backbones and both benchmarks. Full GRAIL adds a smaller but consistent gain. Appendix B reports that fallback is triggered for only 22.7% and 14.8% of calibration prompts for Qwen2.5-1.5B and Qwen2.5-3B, respectively, and is never used during subsequent GRPO training.

4.8 GRAIL and DGRR Ablation

Table 5 separates the initialization and sampling contributions. DGRR gives a modest gain with vanilla LoRA in this setting, showing that its effect is not specific to GRAIL. GRAIL provides the larger improvement without DGRR, and combining them performs best. Appendix C reports resampling controls and rescue statistics.

4.9 Rank and Calibration Budget Sensitivity

Across the rank and calibration-budget sweep in Figure 3, GRAIL remains stable from rank 8

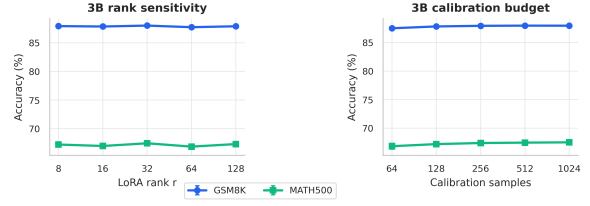


Figure 3: Sensitivity of GRAIL to LoRA rank and calibration samples. GRAIL is stable across ranks and reaches near-best performance with a modest calibration budget.

to 128 and reaches near-best performance with 256–512 calibration prompts. This supports the subspace-alignment account: reward-aligned initialization quality matters more than simply increasing low-rank capacity.

5 Mechanistic Analysis

We use geometry diagnostics to test whether GRAIL changes the early GRPO update subspace. The goal is not a complete causal proof, but to measure reward-gradient capture and test whether orientation, rather than only initialization scale, is associated with final accuracy beyond supervised warm-start controls. Independent-batch stability is reported in Appendix D.

5.1 Initial Reward-gradient Capture

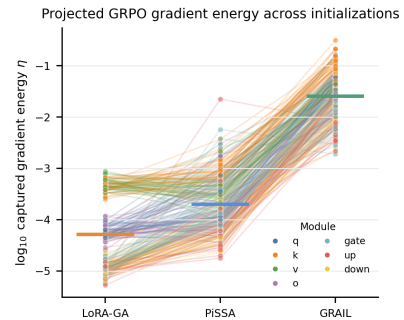


Figure 4: Captured GRPO gradient energy across initializations. Each line follows the same layer-module pair.

Figure 4 evaluates the projected gradient ratio η in Eq. (1). GRAIL captures more GRPO-gradient energy than LoRA-GA and PiSSA across most layer-module pairs, making more reward-induced directions reachable before GRPO starts.

5.2 Subspace Perturbation Controls

The projected-gradient diagnostic is construction-aligned with GRAIL, so we further test whether

the downstream gains come from reward-gradient directions rather than only from initialization scale or singular-value spectra. Appendix D reports perturbation controls that preserve GRAIL’s norm, singular-value spectrum, or factors while disrupting reward-aligned orientation or layer placement. These variants remain below GRAIL without fallback: the drops are 1.65–1.85 points for norm/spectrum-matched random initializations on Qwen2.5-1.5B and 2.11–2.34 points on Qwen2.5-3B, while rotation and layer shuffling also degrade accuracy. Thus, the reward-aligned orientation and layer-specific placement of the initialized subspace both contribute to the downstream improvement.

6 Conclusion

We presented GRAIL, a reward-aware initialization framework for LoRA-based GRPO. GRAIL estimates reward-induced gradients from grouped rollouts and uses them to initialize LoRA adapters before standard GRPO training, reducing subspace-gradient mismatch. We also studied DGRR, an optional training-time extension that skips all-zero-correctness groups and resamples them after self-reflection without exposing reference solutions. Experiments on Qwen2.5, Phi-3.5, and Gemma-2 backbones show improved GSM8K and MATH500 accuracy; a code-generation check improves MBPP and matches LoRA-GRPO on HumanEval. Diagnostics show stronger projected GRPO-gradient capture, independent-batch stability, and early persistence. Overall, PEFT methods for reinforcement learning should account for reward-induced update geometry and correctness-signal degeneracy rather than reuse supervised-finetuning initializers unchanged.

Limitations

Our work has several limitations. First, the evidence is strongest for verifier-based tasks with deterministic correctness signals. The code-generation check extends beyond mathematics but uses one model scale and execution-based rewards; noisy, subjective, or learned rewards for open-ended instruction following remain future work. Second, we evaluate four compact backbones from three model families; larger models and broader task families are still needed to test scalability. Third, GRAIL adds a one-time calibration stage with grouped rollouts and gradient ex-

traction. Although this overhead is small relative to GRPO training, the calibration-budget trade-off deserves further study.

Fourth, GRAIL estimates reward-induced gradients once under the current policy, so later policy drift may reduce alignment. Appendix E shows stable captured-energy advantages over accumulated steps, but online or repeated recalibration remains open. Relatedly, projected-gradient capture is construction-aligned with GRAIL and should be read as a diagnostic of reachable reward-gradient directions, not standalone causal proof. DGRR partly addresses training-time correctness degeneracy, but it adds extra rollout attempts for all-zero-correctness groups and may be less useful when reflection does not recover reward variation.

Finally, the SFT-style fallback assumes reference solutions during calibration, which is natural for GSM8K-like verifiable tasks but less so for open-ended preference optimization. It also makes full GRAIL not strictly reward-only; therefore, we report a no-fallback variant to isolate the reward-gradient component. Without references, GRAIL can use the no-fallback variant or task-specific verifiers. Our reward-aligned SVD factorization is one instantiation, and other mappings from reward gradients to LoRA factors may be explored.

Potential risks. GRAIL is an optimization component, not a safety mechanism. It does not remove biases, unsafe behaviors, or other limitations inherited from the base model, and noisy or learned-reward settings require additional reward validation and robustness evaluation before user-facing use.

References

- Marah Abdin, Jyoti Aneja, Hany Awadalla, Ahmed Awadallah, Ammar Ahmad Awan, Nguyen Bach, Amit Bahree, Arash Bakhtiari, Jianmin Bao, Harkirat Behl, Alon Benhaim, Misha Bilenko, Johan Bjorck, Sébastien Bubeck, et al. 2024. [Phi-3 technical report: A highly capable language model locally on your phone](#). *arXiv preprint arXiv:2404.14219*.
- Armen Aghajanyan, Sonal Gupta, and Luke Zettlemoyer. 2021. [Intrinsic dimensionality explains the effectiveness of language model fine-tuning](#). In *Proceedings of ACL-IJCNLP*.
- Jacob Austin, Augustus Odena, Maxwell Nye, Maarten Bosma, Henryk Michalewski, David Dohan, Ellen Jiang, Carrie Cai, Michael Terry, Quoc Le, and Charles Sutton. 2021. [Program synthesis with large language models](#). *arXiv preprint arXiv:2108.07732*.

- Yuntao Bai, Andy Jones, Kamal Ndousse, Amanda Askell, Anna Chen, Nova DasSarma, Dawn Drain, Stanislav Fort, Deep Ganguli, Tom Henighan, et al. 2022a. [Training a helpful and harmless assistant with reinforcement learning from human feedback](#). *arXiv preprint arXiv:2204.05862*.
- Yuntao Bai, Saurav Kadavath, Sandipan Kundu, Amanda Askell, Jackson Kernion, Andy Jones, Anna Chen, Anna Goldie, Azalia Mirhoseini, Cameron McKinnon, et al. 2022b. [Constitutional AI: Harmlessness from AI feedback](#). *arXiv preprint arXiv:2212.08073*.
- Elad Ben Zaken, Yoav Goldberg, and Shauli Ravfogel. 2022. [BitFit: Simple parameter-efficient fine-tuning for transformer-based masked language-models](#). In *Proceedings of ACL*.
- Tom Brown, Benjamin Mann, Nick Ryder, Melanie Subbiah, Jared D. 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 Ziegler, Jeffrey Wu, Clemens Winter, et al. 2020. [Language models are few-shot learners](#). In *Advances in Neural Information Processing Systems*, volume 33, pages 1877–1901. Curran Associates, Inc.
- Mark Chen, Jerry Tworek, Heewoo Jun, Qiming Yuan, Henrique Ponde de Oliveira Pinto, Jared Kaplan, Harri Edwards, Yuri Burda, Nicholas Joseph, Greg Brockman, Alex Ray, Raul Puri, Gretchen Krueger, Michael Petrov, Heidy Khlaaf, Girish Sastry, Pamela Mishkin, Brooke Chan, Scott Gray, et al. 2021. [Evaluating large language models trained on code](#). *arXiv preprint arXiv:2107.03374*.
- Paul F. Christiano, Jan Leike, Tom Brown, Miljan Martic, Shane Legg, and Dario Amodei. 2017. [Deep reinforcement learning from human preferences](#). In *Advances in Neural Information Processing Systems*, volume 30. Curran Associates, Inc.
- Karl Cobbe, Vineet Kosaraju, Mohammad Bavarian, Mark Chen, Heewoo Jun, Lukasz Kaiser, Matthias Plappert, Jerry Tworek, Jacob Hilton, Reiichiro Nakano, Christopher Hesse, and John Schulman. 2021. [Training verifiers to solve math word problems](#). *arXiv preprint arXiv:2110.14168*.
- Tim Dettmers, Artidoro Pagnoni, Ari Holtzman, and Luke Zettlemoyer. 2023. [QLoRA: Efficient finetuning of quantized LLMs](#). In *Advances in Neural Information Processing Systems*.
- Jacob Devlin, Ming-Wei Chang, Kenton Lee, and Kristina Toutanova. 2019. [BERT: Pre-training of deep bidirectional transformers for language understanding](#). In *Proceedings of the 2019 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies, Volume 1 (Long and Short Papers)*, pages 4171–4186, Minneapolis, Minnesota. Association for Computational Linguistics.
- Kawin Ethayarajh, Winnie Xu, Niklas Muennighoff, Dan Jurafsky, and Douwe Kiela. 2024. [Model alignment as prospect theoretic optimization](#). In *Proceedings of the 41st International Conference on Machine Learning*, volume 235 of *Proceedings of Machine Learning Research*, pages 12634–12651. PMLR.
- Gemma Team. 2024. [Gemma 2: Improving open language models at a practical size](#). *arXiv preprint arXiv:2408.00118*.
- Zhibin Gou, Zhihong Shao, Yeyun Gong, Yelong Shen, Yujiu Yang, Minlie Huang, Nan Duan, and Weizhu Chen. 2024. [ToRA: A tool-integrated reasoning agent for mathematical problem solving](#). In *International Conference on Learning Representations*.
- Soufiane Hayou, Nikhil Ghosh, and Bin Yu. 2024. [LoRA+: Efficient low rank adaptation of large models](#). In *Proceedings of the 41st International Conference on Machine Learning*, volume 235 of *Proceedings of Machine Learning Research*, pages 17783–17806. PMLR.
- Junxian He, Chunting Zhou, Xuezhe Ma, Taylor Berg-Kirkpatrick, and Graham Neubig. 2022. [Towards a unified view of parameter-efficient transfer learning](#). In *International Conference on Learning Representations*.
- Dan Hendrycks, Collin Burns, Saurav Kadavath, Akul Arora, Steven Basart, Eric Tang, Dawn Song, and Jacob Steinhardt. 2021. [Measuring mathematical problem solving with the MATH dataset](#). In *Proceedings of the Neural Information Processing Systems Track on Datasets and Benchmarks*, volume 1.
- Neil Houlsby, Andrei Giurgiu, Stanislaw Jastrzebski, Bruna Morrone, Quentin De Laroussilhe, Andrea Gesmundo, Mona Attariyan, and Sylvain Gelly. 2019. [Parameter-efficient transfer learning for NLP](#). In *International Conference on Machine Learning*.
- Edward J. Hu, Yelong Shen, Phillip Wallis, Zeyuan Allen-Zhu, Yuanzhi Li, Shean Wang, Lu Wang, and Weizhu Chen. 2022. [LoRA: Low-rank adaptation of large language models](#). In *International Conference on Learning Representations*.
- Damjan Kalajdzievski. 2023. [A rank stabilization scaling factor for fine-tuning with LoRA](#). *arXiv preprint arXiv:2312.03732*.
- Takeshi Kojima, Shixiang Shane Gu, Machel Reid, Yutaka Matsuo, and Yusuke Iwasawa. 2022. [Large language models are zero-shot reasoners](#). In *Advances in Neural Information Processing Systems*.
- Dawid J. Kopiczko, Tijmen Blankevoort, and Yuki M. Asano. 2024. [VeRA: Vector-based random matrix adaptation](#). In *International Conference on Learning Representations*.
- Brian Lester, Rami Al-Rfou, and Noah Constant. 2021. [The power of scale for parameter-efficient prompt tuning](#). In *Proceedings of EMNLP*.

- Xiang Lisa Li and Percy Liang. 2021. [Prefix-tuning: Optimizing continuous prompts for generation](#). In *Proceedings of ACL-IJCNLP*.
- Yixiao Li, Yifan Yu, Chen Liang, Pengcheng He, Nikos Karampatziakis, Weizhu Chen, and Tuo Zhao. 2024. [LoftQ: LoRA-fine-tuning-aware quantization for large language models](#). In *International Conference on Learning Representations*.
- Hunter Lightman, Vineet Kosaraju, Yura Burda, Harri Edwards, Bowen Baker, Teddy Lee, Jan Leike, John Schulman, Ilya Sutskever, and Karl Cobbe. 2024. [Let’s verify step by step](#). In *International Conference on Learning Representations*.
- Shih-Yang Liu, Chien-Yi Wang, Hongxu Yin, Pavlo Molchanov, Yu-Chiang Frank Wang, Kwang-Ting Cheng, and Min-Hung Chen. 2024. [DoRA: Weight-decomposed low-rank adaptation](#). In *Proceedings of the 41st International Conference on Machine Learning*, volume 235 of *Proceedings of Machine Learning Research*, pages 32100–32121. PMLR.
- Xiao Liu, Kaixuan Ji, Yicheng Fu, Weng Tam, Zhengxiao Du, Zhilin Yang, and Jie Tang. 2022. [P-Tuning: Prompt tuning can be comparable to fine-tuning across scales and tasks](#). In *Proceedings of the 60th Annual Meeting of the Association for Computational Linguistics (Volume 2: Short Papers)*, pages 61–68. Association for Computational Linguistics.
- Haipeng Luo, Qingfeng Sun, Can Xu, Pu Zhao, Jian-Guang Lou, Chongyang Tao, Xiubo Geng, Qingwei Lin, Shifeng Chen, Yansong Tang, and Dongmei Zhang. 2025. [Wizardmath: Empowering mathematical reasoning for large language models via reinforced evol-instruct](#). In *International Conference on Learning Representations*.
- Rabeeh Karimi Mahabadi, James Henderson, and Sebastian Ruder. 2021. [Compacter: Efficient low-rank hypercomplex adapter layers](#). In *Advances in Neural Information Processing Systems*.
- Yuning Mao, Lambert Mathias, Rui Hou, Amjad Almahairi, Hao Ma, Jiawei Han, Scott Yih, and Madian Khabsa. 2022. [UniPELT: A unified framework for parameter-efficient language model tuning](#). In *Proceedings of the 60th Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, pages 6253–6264. Association for Computational Linguistics.
- Fanxu Meng, Zhaohui Wang, and Muhan Zhang. 2024. [PiSSA: Principal singular values and singular vectors adaptation of large language models](#). In *Advances in Neural Information Processing Systems*, volume 37.
- Reiichiro Nakano, Jacob Hilton, Suchir Balaji, Jeff Wu, Long Ouyang, Christina Kim, Christopher Hesse, Shantanu Jain, Vineet Kosaraju, William Saunders, et al. 2021. [WebGPT: Browser-assisted question-answering with human feedback](#). *arXiv preprint arXiv:2112.09332*.
- OpenAI. 2023. [GPT-4 technical report](#). *arXiv preprint arXiv:2303.08774*.
- Long Ouyang, Jeffrey Wu, Xu Jiang, Diogo Almeida, Carroll L. Wainwright, Pamela Mishkin, Chong Zhang, Sandhini Agarwal, Katarina Slama, Alex Ray, et al. 2022. [Training language models to follow instructions with human feedback](#). In *Advances in Neural Information Processing Systems*, volume 35.
- Jonas Pfeiffer, Andreas Rücklé, Clifton Poth, Aishwarya Kamath, Ivan Vulić, Sebastian Ruder, Kyunghyun Cho, and Iryna Gurevych. 2020. [AdapterHub: A framework for adapting transformers](#). In *Proceedings of EMNLP: System Demonstrations*.
- Rafael Rafailov, Archit Sharma, Eric Mitchell, Stefano Ermon, Christopher D. Manning, and Chelsea Finn. 2023. [Direct preference optimization: Your language model is secretly a reward model](#). In *Advances in Neural Information Processing Systems*.
- Colin Raffel, Noam Shazeer, Adam Roberts, Katherine Lee, Sharan Narang, Michael Matena, Yanqi Zhou, Wei Li, and Peter J. Liu. 2020. [Exploring the limits of transfer learning with a unified text-to-text transformer](#). *Journal of Machine Learning Research*, 21(140):1–67.
- John Schulman, Filip Wolski, Prafulla Dhariwal, Alec Radford, and Oleg Klimov. 2017. [Proximal policy optimization algorithms](#). *arXiv preprint arXiv: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. 2024. [DeepSeekMath: Pushing the limits of mathematical reasoning in open language models](#). *arXiv preprint arXiv:2402.03300*.
- Nisan Stiennon, Long Ouyang, Jeffrey Wu, Daniel Ziegler, Ryan Lowe, Chelsea Voss, Alec Radford, Dario Amodei, and Paul F. Christiano. 2020. [Learning to summarize with human feedback](#). In *Advances in Neural Information Processing Systems*, volume 33, pages 3008–3021. Curran Associates, Inc.
- Hugo Touvron, Thibaut Lavril, Gautier Izacard, Xavier Martinet, Marie-Anne Lachaux, Timothée Lacroix, Baptiste Rozière, Naman Goyal, Eric Hambro, Faisal Azhar, et al. 2023. [LLaMA: Open and efficient foundation language models](#). *arXiv preprint arXiv:2302.13971*.
- Jonathan Uesato, Nate Kushman, Ramana Kumar, Francis Song, Noah Siegel, Lisa Wang, Antonia Creswell, Geoffrey Irving, and Irina Higgins. 2022. [Solving math word problems with process- and outcome-based feedback](#). *arXiv preprint arXiv:2211.14275*.
- Ashish Vaswani, Noam Shazeer, Niki Parmar, Jakob Uszkoreit, Llion Jones, Aidan N. Gomez, Lukasz

- Kaiser, and Illia Polosukhin. 2017. [Attention is all you need](#). In *Advances in Neural Information Processing Systems*.
- Peiyi Wang, Lei Li, Zhihong Shao, Runxin Xu, Damai Dai, Yifei Li, Deli Chen, Yu Wu, and Zhifang Sui. 2024a. [Math-Shepherd: Verify and reinforce LLMs step-by-step without human annotations](#). In *Proceedings of the 62nd Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, pages 9426–9439. Association for Computational Linguistics.
- Shaowen Wang, Linxi Yu, and Jian Li. 2024b. [LoRA-GA: Low-rank adaptation with gradient approximation](#). In *Advances in Neural Information Processing Systems*, volume 37, pages 54905–54931. Curran Associates, Inc.
- Xuezhi Wang, Jason Wei, Dale Schuurmans, Quoc V. Le, Ed H. Chi, Sharan Narang, Aakanksha Chowdhery, and Denny Zhou. 2023. [Self-consistency improves chain of thought reasoning in language models](#). In *International Conference on Learning Representations*.
- Jason Wei, Xuezhi Wang, Dale Schuurmans, Maarten Bosma, Brian Ichter, Fei Xia, Ed H. Chi, Quoc V. Le, and Denny Zhou. 2022. [Chain-of-thought prompting elicits reasoning in large language models](#). In *Advances in Neural Information Processing Systems*.
- An Yang, Baosong Yang, Beichen Zhang, Binyuan Hui, Bo Zheng, Bowen Yu, Chengyuan Li, Dayiheng Liu, Fei Huang, Haoran Wei, Huan Lin, Jian Yang, Jianhong Tu, Jianwei Zhang, Jianxin Yang, Jiaxi Yang, Jingren Zhou, Junyang Lin, Kai Dang, et al. 2024a. [Qwen2.5 technical report](#). *arXiv preprint arXiv:2412.15115*.
- An Yang, Beichen Zhang, Binyuan Hui, Bofei Gao, Bowen Yu, Chengpeng Li, Dayiheng Liu, Jianhong Tu, Jingren Zhou, Junyang Lin, et al. 2024b. [Qwen2.5-Math technical report: Toward mathematical expert model via self-improvement](#). *arXiv preprint arXiv:2409.12122*.
- Shunyu Yao, Dian Yu, Jeffrey Zhao, Izhak Shafran, Thomas L. Griffiths, Yuan Cao, and Karthik Narasimhan. 2023. [Tree of thoughts: Deliberate problem solving with large language models](#). In *Advances in Neural Information Processing Systems*, volume 36, pages 11809–11822. Curran Associates, Inc.
- Longhui Yu, Weisen Jiang, Han Shi, Jincheng Yu, Zhengying Liu, Yu Zhang, James T. Kwok, Zhenguo Li, Adrian Weller, and Weiyang Liu. 2024. [MetaMath: Bootstrap your own mathematical questions for large language models](#). In *International Conference on Learning Representations*.
- Qiyang Yu, Zheng Zhang, Ruofei Zhu, Yufeng Yuan, Xiaochen Zuo, Yu Yue, Weinan Dai, Tiantian Fan, Gaohong Liu, Juncai Liu, Lingjun Liu, Xin Liu, Haibin Lin, Zhiqi Lin, Bole Ma, Guangming Sheng, Yuxuan Tong, Chi Zhang, Mofan Zhang, et al. 2025. [DAPO: An open-source LLM reinforcement learning system at scale](#). In *Advances in Neural Information Processing Systems*.
- Eric Zelikman, Yuhuai Wu, Jesse Mu, and Noah D. Goodman. 2022. [STaR: Bootstrapping reasoning with reasoning](#). In *Advances in Neural Information Processing Systems*.
- Qingru Zhang, Minshuo Chen, Alexander Bukharin, Pengcheng He, Yu Cheng, Weizhu Chen, and Tuo Zhao. 2023. [Adaptive budget allocation for parameter-efficient fine-tuning](#). In *International Conference on Learning Representations*.
- Denny Zhou, Nathanael Schärli, Le Hou, Jason Wei, Nathan Scales, Xuezhi Wang, Dale Schuurmans, Claire Cui, Olivier Bousquet, Quoc Le, and Ed Chi. 2023. [Least-to-most prompting enables complex reasoning in large language models](#). In *International Conference on Learning Representations*.
- Daniel M. Ziegler, Nisan Stiennon, Jeffrey Wu, Tom B. Brown, Alec Radford, Dario Amodei, Paul Christiano, and Geoffrey Irving. 2019. [Fine-tuning language models from human preferences](#). *arXiv preprint arXiv:1909.08593*.

A Training and Initialization Hyperparameters

We include the experimental details needed to interpret the mathematical-reasoning and code-generation results. The datasets are described first, followed by the shared GRPO recipe and the additional one-shot initialization settings used by GRAIL.

A.1 Dataset Descriptions

GSM8K. GSM8K (Cobbe et al., 2021) contains grade-school math word problems with short natural-language solutions and numeric final answers. We use GSM8K as the GRPO training source and also report in-domain test accuracy.

MATH and MATH500. MATH (Hendrycks et al., 2021) is a competition-style mathematical reasoning benchmark covering diverse topics and difficulty levels. MATH500 (Lightman et al., 2024) is a 500-problem subset used for process-supervision and mathematical reasoning evaluation. We use MATH500 as an out-of-domain benchmark to test whether GSM8K-trained policies transfer to harder math problems.

Verifiable code generation. For the code-domain check, we train on a 10,000-problem subset of the public open-r1 verifiable Python dataset using test-case execution as the correctness reward. We evaluate pass@1 on MBPP (Austin et al., 2021) and HumanEval (Chen et al., 2021), two standard English-language Python code-generation benchmarks. This experiment uses Qwen2.5-3B-Instruct and otherwise follows the same GRPO comparison protocol.

A.2 Artifacts, Data, and Responsible Use

We use publicly released research artifacts, including Qwen2.5-Instruct models, Phi-3.5-mini-instruct, Gemma-2-2B-it, GSM8K, MATH/MATH500, a 10,000-problem subset of the open-r1 verifiable coding dataset, MBPP, HumanEval, Swift RLHF, and DeepSpeed, under their respective licenses and terms of use. The Phi-3.5-mini-instruct and Gemma-2-2B-it checkpoints are obtained from their public research releases. These artifacts are used only for research training, evaluation, and analysis. We use the mathematical and code datasets for their intended research evaluation roles and do not redistribute modified datasets or model weights.

The datasets used in this work consist of English mathematical reasoning problems and Python programming tasks. We do not collect new data from human participants and do not introduce additional personally identifying information. We use the public benchmark splits as released and do not perform data collection requiring anonymization.

We use the standard GSM8K split with 7,473 training examples and 1,319 test examples. MATH500 contains 500 evaluation problems. All rows in the main table use the same evaluation prompts and deterministic decoding within each benchmark: GSM8K is evaluated with 4-shot CoT prompts and MATH500 with 0-shot prompts. Under this protocol, the Phi-3.5-mini base model obtains 86.5% on GSM8K and 49.6% on MATH500, and Gemma-2-2B obtains 61.49% on GSM8K and 24.0% on MATH500. For GRAIL calibration, the main setting uses 128 GSM8K training prompts with 8 rollouts per prompt; sensitivity experiments vary the calibration budget over {64, 128, 256, 512, 1024} prompts.

A.3 Compute Budget

All experiments use Qwen2.5-1.5B-Instruct, Qwen2.5-3B-Instruct, Phi-3.5-mini-instruct, and Gemma-2-2B-it as base models. GRPO training is run with bfloat16 precision and DeepSpeed ZeRO-2 on 8 NVIDIA H20 GPUs. Each one-epoch mathematical-reasoning GRPO run takes approximately 2 hours, corresponding to about 16 H20 GPU-hours per run. Trained mathematical-reasoning results are averaged over three independent seeds; the code-domain check reports point estimates without across-seed dispersion. The one-time GRAIL calibration stage uses 128 prompts in the main math setting and is performed before standard GRPO training. With 8 responses per prompt, this corresponds to 1,024 calibration rollouts and takes approximately 12 minutes on a single H20 GPU, or about 0.2 H20 GPU-hours. Thus, the initialization overhead is about 1.25% of a one-epoch GRPO run in our main math setup.

The full hyperparameter settings are given in Tables 6 and 7.

Category	Setting
RL algorithm	GRPO
Implementation	Swift RLHF
Training dataset	GSM8K (math); 10k subset of open-r1 verifiable Python problems (code)
Reward functions	math accuracy + format; test-case execution for code
Generations per prompt	8
Sampling temperature	0.9
Max prompt length	1024
Max completion length	1024
Training epochs	1
Learning rate	1×10^{-6}
Warmup ratio	0.05
Precision	bfloat16
GPUs	8
Distributed training	DS ZeRO-2
Per-device train batch size	64
Per-device eval batch size	16
Gradient accumulation steps	1
LoRA target modules	all linear
LoRA implementation	Swift LoRA

Table 6: Training hyperparameters for GRPO experiments.

Category	Setting
Initialization method	GRAIL
Initialization stage	one-shot, pre-GRPO
Calibration data	GSM8K train
Calibration rewards	math accuracy + format
Rollouts per prompt	8
Sampling temperature	0.9
Max completion length	1024
Initialization loss	reward PG; SFT fallback for all-zero correctness groups
Gradient target	frozen weights, not updated
Main LoRA rank	$r = 16$
Main calibration budget	128 prompts
Sensitivity ranks	{8, 16, 32, 64, 128}
Sensitivity budgets	{64, 128, 256, 512, 1024} prompts
Target modules	all linear
Factor construction	reward-aligned SVD
Post-init training	standard GRPO; optional DGRR for all-zero correctness groups

Table 7: GRAIL initialization hyperparameters.

Model	Calibration prompts	Fallback prompts	Rate
Qwen2.5-1.5B	128	29	22.7%
Qwen2.5-3B	128	19	14.8%

Table 8: Frequency of all-zero-correctness calibration groups where the SFT-style fallback is triggered during initialization. The fallback is not used during subsequent GRPO training.

B Fallback Trigger Details

C DGRR Reflection Prompt

DGRR uses reflection only to resample correctness-degenerate groups during GRPO. The prompt does not include the reference answer, the reference solution, or any gold reasoning. It contains only the original problem, the model’s failed completions, and binary correctness feedback. In our implementation, at most one reflection-resampling attempt is used per degenerate group.

You are solving a math problem. Your previous answers to this problem were marked incorrect by an automatic verifier. The correct answer and reference solution are not provided.

Problem: {problem}

Previous incorrect attempts:
{failed_completions}

Feedback: Each previous attempt was incorrect. Review the attempts, identify likely mistakes, and produce a new solution. Make sure the reasoning is concise and the final answer is clearly stated in the required answer format.

After the policy generates a new group from this prompt, rewards are recomputed using the same

Training-time variant	Qwen2.5-1.5B GSM8K (%)	Qwen2.5-3B GSM8K (%)
GRAIL + random resampling without reflection	72.71 $\pm$ 0.24	88.09 $\pm$ 0.17
GRAIL + resampling with same token budget	72.82 $\pm$ 0.21	88.17 $\pm$ 0.16
GRAIL + DGRR	73.09 $\pm$ 0.26	88.40 $\pm$ 0.15

Table 9: DGRR resampling controls on GSM8K. All rows start from the same full GRAIL initialization and differ only in how all-zero-correctness groups are handled during GRPO. Random resampling re-queries the original problem without failed completions or binary feedback. The token-budget-matched control allows the same additional rollout budget as DGRR but does not use the reflection prompt.

accuracy and format verifiers as standard GRPO. The standard GRPO loss is applied only if the re-generated group contains both correct and incorrect completions, $0 < \sum_i c_i < G$; otherwise, the group is skipped. This design changes the sampling and update distribution while retaining the standard per-group loss formula and adding no supervised target tokens.

Table 9 separates the effect of reflection-conditioned resampling from extra rollout compute. Simple resampling and token-budget-matched resampling improve only modestly over full GRAIL, whereas DGRR gives the largest gain under the same initialization, suggesting that the feedback-conditioned reflection prompt contributes beyond additional sampling alone.

Initialization	Degenerate	Rescued	Rate
LoRA+DGRR	797	217	27.20%
GRAIL+DGRR	766	262	34.20%

Table 10: DGRR process statistics on Qwen2.5-1.5B. A rescued group yields non-degenerate reward variation after reflection-conditioned resampling.

In the recorded process statistics, GRAIL+DGRR encounters fewer degenerate groups and rescues a larger fraction than LoRA+DGRR, with observed rescue rates of 34.20% and 27.20%, respectively. These descriptive counts complement the outcome-level ablation in Table 5; they are not a separate significance test.

D Additional Geometry Diagnostics

The main text reports the central geometry results. The appendix adds checks that vary ranks, calibration budgets, layers, modules, and accumulated gradient steps, so that the same subspace-alignment claim is tested under several perturbations.

D.1 Stability Across Reward Batches

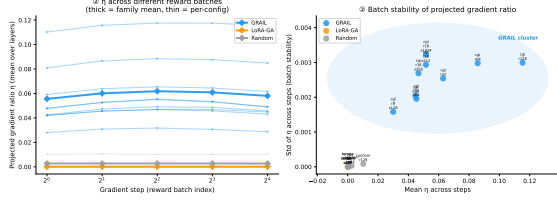


Figure 5: Projected gradient ratio across different reward batches. GRAIL maintains higher reward-gradient capture than LoRA-GA and random initializations across batches, indicating that the aligned subspace is not specific to a single calibration estimate.

To test whether alignment is calibration-specific, we recompute η on independent reward batches. Figure 5 shows that GRAIL maintains higher reward-gradient capture than LoRA-GA and random initialization across batches, ranks, and calibration budgets, indicating a reward-relevant rather than memorized subspace.

D.2 Reward-Aligned SVD Initialization

The reward-aligned SVD operator used by GRAIL is specified below. For each target layer ℓ , GRAIL first estimates the reward-induced initialization gradient

$$g_\ell = \nabla_{W_\ell} \mathcal{L}_{\text{init}}.$$

Since gradient descent follows the negative gradient direction, we define the reward-improving direction as

$$D_\ell = -g_\ell.$$

We then compute a truncated SVD

$$D_\ell = U_\ell \Sigma_\ell V_\ell^\top,$$

and keep the top- r singular components

$$D_{\ell,r} = U_{\ell,r} \Sigma_{\ell,r} V_{\ell,r}^\top.$$

To convert this rank- r reward-improving direction into LoRA factors, we split the singular values symmetrically:

$$B_\ell = \sqrt{\rho_\ell} U_{\ell,r} \Sigma_{\ell,r}^{1/2}, \quad A_\ell = \sqrt{\rho_\ell} \Sigma_{\ell,r}^{1/2} V_{\ell,r}^\top.$$

This yields

$$B_\ell A_\ell = \rho_\ell U_{\ell,r} \Sigma_{\ell,r} V_{\ell,r}^\top = \rho_\ell D_{\ell,r}.$$

The scale coefficient ρ_ℓ controls the magnitude of the initial adapter update. We set

$$\rho_\ell = \tau \frac{\|W_{\ell,0}\|_F}{\|\Sigma_{\ell,r}\|_F + \epsilon},$$

where $W_{\ell,0}$ is the frozen pretrained weight, τ is a small global scale ratio, and ϵ is used for numerical stability. This gives

$$\|\Delta W_\ell\|_F = \|B_\ell A_\ell\|_F \approx \tau \|W_{\ell,0}\|_F,$$

which keeps the initialized adapter close to the original model while aligning the adapter subspace with reward-improving directions.

D.3 Full-Parameter GRPO Reward Curve

Figure 6 shows the training reward trajectory for the full-parameter GRPO control discussed in the main text.

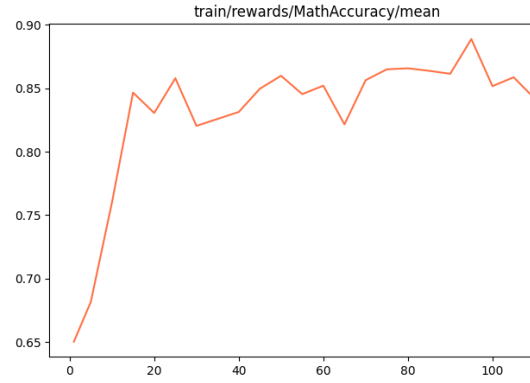


Figure 6: Training reward curve of full-parameter GRPO on Qwen2.5-3B. The reward rises rapidly in the early stage, indicating informative reward-improving directions, but this training reward improvement does not necessarily yield stronger downstream reasoning generalization.

Method	GSM8K (%)	MATH500 (%)
Base	85.14	66.0
Full GRPO	83.40	62.40
GRAIL-LoRA	88.02	68.2

Table 11: Full-parameter GRPO control on Qwen2.5-3B. Full GRPO uses the same reward functions, rollout configuration, and data source, but updates all model parameters. We report the best result among the tuned full-parameter GRPO settings.

D.4 RL-specific Gradient Geometry

The gradient object used by GRAIL differs from supervised likelihood gradients in several ways. GRPO gradients are computed from samples of the current policy, weight responses by group-relative advantages, and contain both positive and negative directions: high-reward completions are reinforced while low-reward completions are suppressed. Under sparse correctness rewards, some

groups also provide no correctness contrast, which motivates the fallback analysis in Section 3.2. These properties make the reward-gradient subspace policy-dependent and contrastive rather than a fixed reference-imitation direction. The projected ratio in Eq. (1) is therefore used as a local measure of how much immediate reward-improving gradient energy a LoRA subspace can express, not as a global convergence guarantee.

D.5 Subspace Perturbation Controls

Model	Initialization	GSM8K (%)	Drop
Qwen2.5-1.5B	LoRA-GRPO	70.31 $\pm$ 0.18	-1.85
Qwen2.5-1.5B	Norm-matched random	70.36 $\pm$ 0.15	-1.80
Qwen2.5-1.5B	Singular-value-matched random	70.51 $\pm$ 0.15	-1.65
Qwen2.5-1.5B	Randomly rotated GRAIL	70.89 $\pm$ 0.23	-1.27
Qwen2.5-1.5B	Layer-shuffled GRAIL	71.42 $\pm$ 0.15	-0.74
Qwen2.5-1.5B	GRAIL w/o fallback	72.16 $\pm$ 0.21	-
Qwen2.5-3B	LoRA-GRPO	85.06 $\pm$ 0.20	-2.42
Qwen2.5-3B	Norm-matched random	85.14 $\pm$ 0.15	-2.34
Qwen2.5-3B	Singular-value-matched random	85.37 $\pm$ 0.15	-2.11
Qwen2.5-3B	Randomly rotated GRAIL	85.70 $\pm$ 0.19	-1.78
Qwen2.5-3B	Layer-shuffled GRAIL	86.66 $\pm$ 0.15	-0.82
Qwen2.5-3B	GRAIL w/o fallback	87.48 $\pm$ 0.17	-

Table 12: GSM8K subspace perturbation controls. Drop is measured against GRAIL without fallback within the same model scale. The consistent degradation of norm-matched, spectrum-matched, rotated, and layer-shuffled controls indicates that GRAIL’s gains depend on reward-aligned directions rather than only update magnitude or singular-value structure.

D.6 Geometry Dashboard

Figure 7 separates the initialization geometry into energy, lift, angle, and calibration-budget views.

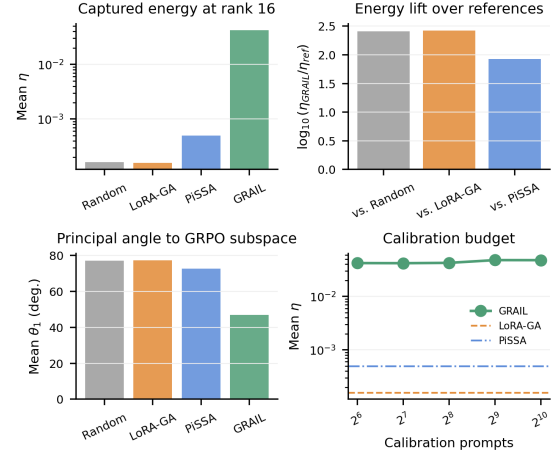


Figure 7: Geometry diagnostics at the initialization point. The top panels compare captured GRPO-gradient energy and energy lift over random, LoRA-GA, and PiSSA references; the bottom panels report principal angles and calibration-budget sensitivity. GRAIL consistently captures more reward-gradient energy and reduces the angle to the GRPO-gradient subspace.

D.7 Principal-Angle Diagnostics

Figure 8 gives a more detailed view of the subspace-angle behavior. GRAIL shifts the smallest principal angle toward lower values than LoRA-GA and random initialization, indicating that the initialized adapter subspace is closer to the GRPO-gradient subspace. The rank sweep further shows that increasing the LoRA rank exposes additional aligned directions.

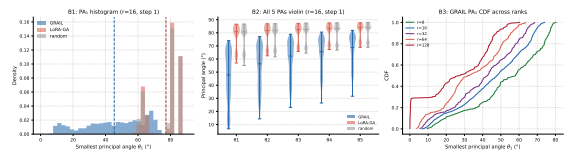


Figure 8: Principal-angle diagnostics at initialization. The histogram and violin plots show that GRAIL produces smaller principal angles to the GRPO-gradient subspace than LoRA-GA and random initialization. The CDF across ranks shows that larger ranks capture more aligned directions.

D.8 Rank and Calibration Design

Rank and calibration budget affect different parts of the initialization. Figure 9 tests whether the alignment advantage depends on the chosen LoRA rank or on the number of calibration prompts. Increasing rank improves the capacity of all adapter subspaces, while GRAIL already obtains useful alignment at small ranks. Increasing the calibra-

tion budget improves the estimate modestly and saturates after a moderate number of prompts, consistent with the main sensitivity results.

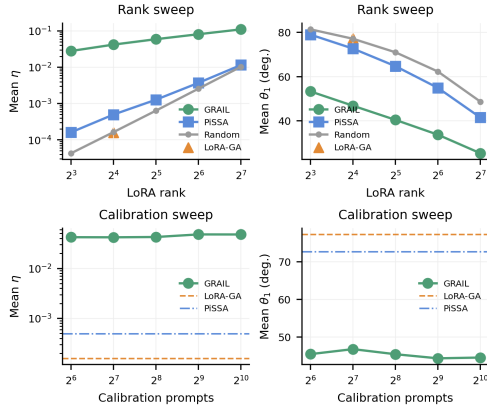


Figure 9: Effect of LoRA rank and calibration budget on subspace alignment. Increasing rank improves the capacity of all adapter subspaces, but GRAIL remains above the random, LoRA-GA, and PiSSA references at matched ranks. The calibration sweep shows that the reward-gradient estimate stabilizes after a moderate number of prompts.

E Persistence and Module-level Behavior

The main-text diagnostics use a single initialization snapshot. Here we check two possible failure modes: the alignment could disappear after accumulating later GRPO gradients, or it could be concentrated in only one module family.

E.1 Temporal Persistence

A single calibration snapshot may overstate alignment if later GRPO gradients quickly rotate away from the initialized subspace. Figure 10 shows the accumulated-gradient view. For the checkpoint diagnostic in Table 13, we compare each method’s initial adapter subspace with reward gradients measured at the indicated Qwen2.5-1.5B GRPO training stage. GRAIL’s captured ratio decreases over training, which is consistent with policy drift, while remaining above random LoRA and LoRA-GA at every reported checkpoint.

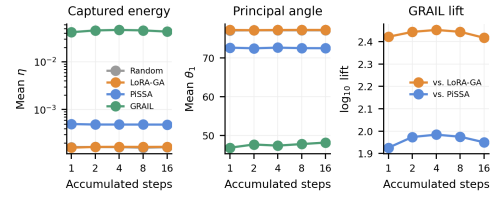


Figure 10: Captured-energy persistence under accumulated GRPO gradients. Across accumulation steps, the GRAIL-initialized subspace continues to capture more gradient energy than LoRA-GA and PiSSA, indicating that the alignment is not restricted to the first calibration snapshot.

Training stage	Random LoRA	LoRA-GA	GRAIL
Step 1	0.008	0.018	0.117
25%	0.006	0.021	0.095
50%	0.006	0.015	0.081
100%	0.004	0.011	0.068

Table 13: Reward-gradient energy captured by the initial adapter subspace at different Qwen2.5-1.5B GRPO training stages.

E.2 Module Signature

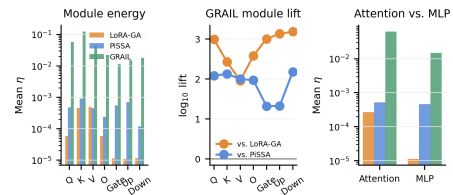


Figure 11: Module-level decomposition of the alignment effect. GRAIL improves captured GRPO-gradient energy in both attention projections and MLP projections, showing that the gain is not driven by a single module family.

Figure 11 breaks the captured-energy comparison down by module family. The gains appear in both attention projections and MLP projections, so the alignment effect is not explained by a single module type dominating the aggregate statistics. This supports the layer-module interpretation and indicates that the geometric advantage is not driven by a particular module family.