

GradAlign: Gradient-Aligned Data Selection for LLM Reinforcement Learning

Ningyuan Yang^{1*}, Weihua Du^{2*}, Weiwei Sun², Sean Welleck², Yiming Yang²

¹ Institute for Interdisciplinary Information Sciences (IIIS), Tsinghua University

² Language Technologies Institute (LTI), Carnegie Mellon University

yangny23@mails.tsinghua.edu.cn, {weihuad, weiweis, wellecks, yiming}@cs.cmu.edu

Abstract

Reinforcement learning (RL) has become a central post-training paradigm for large language models (LLMs), but its performance is highly sensitive to the quality of training problems. This sensitivity stems from the non-stationarity of RL: rollouts are generated by an evolving policy, and learning is shaped by exploration and reward feedback, unlike supervised fine-tuning (SFT) with fixed trajectories. As a result, prior work often relies on manual curation or simple heuristic filters (e.g., accuracy), which can admit incorrect or low-utility problems. We propose **GradAlign**, a gradient-aligned data selection method for LLM reinforcement learning that uses a small, trusted validation set to prioritize training problems whose policy gradients align with validation gradients, yielding an adaptive curriculum. We evaluate GradAlign across three challenging data regimes: unreliable reward signals, distribution imbalance, and low-utility training corpus, showing that GradAlign consistently outperforms existing baselines, underscoring the importance of directional gradient signals in navigating non-stationary policy optimization and yielding more stable training and improved final performance. We release our implementation at <https://github.com/StigLidu/GradAlign>.

1 Introduction

Reinforcement learning (RL) has become an important training paradigm for enhancing the capabilities of large language models (LLMs) in complex tasks and environments (DeepSeek-AI et al., 2025; Yang et al., 2025; Sun et al., 2025b). However, the current success of LLM RL is critically dependent on high-quality human-labeled training corpora (Hu et al., 2025; He et al., 2025), which are expensive to obtain. To support large-scale RL training, researchers increasingly turn to massive, automatically collected corpora (Yue et al., 2024). These collections inevitably mix high-quality instances with trivial or misleading ones, along with noisy labels. This raises two key challenges: (1) how to identify instances that are suitable for a given downstream task, and (2) how to design mechanisms that prevent low-quality or misleading data from corrupting the RL training signal.

Existing RL data selection strategies for LLM training are relatively simple and primarily rely on rule-based heuristics. For example, Yu et al. (2025) retain problems with intermediate accuracy to alleviate gradient vanishing, whereas Sun et al. (2025c) prioritize samples with accuracy near 50%, analogous to selecting samples near the decision boundary in active learning. Although such partially solved problems can be informative, these methods rely on scalar difficulty signals and do not consider whether the resulting updates move the policy toward improved downstream performance. Other approaches score training problems using gradient-based criteria, often emphasizing large parameter updates (Li et al., 2025b), internal consistency within the training distribution (Li et al., 2025a), or

*Equal contribution.

gradient diversity (Liang et al., 2025). However, these strategies do not explicitly verify alignment with downstream optimization objectives and are typically static, failing to adapt the selection criterion as the policy evolves during RL training.

In this work, we propose **GradAlign**, an automated, gradient-informed algorithm for selecting training problems in LLM reinforcement learning. GradAlign leverages a small, trusted validation set and uses the policy gradient as a first-order surrogate to approximate the expected improvement in validation performance induced by policy updates. For each candidate training problem, we compute an alignment score: the cosine similarity between its policy gradient and the aggregated validation gradient. Training problems are then ranked by this alignment score, favoring updates that are most consistent with improving validation performance.

To address RL non-stationarity, GradAlign adopts an adaptive curriculum that periodically recomputes validation gradients and re-scores candidate problems under the current policy, using the validation set as a directional probe rather than a direct scoring or retrieval signal. GradAlign is particularly effective in challenging data regimes common in large-scale LLM RL, including unreliable reward signals (Skalse et al., 2022), distribution mismatch between training and downstream problems (Kirk et al., 2023), and datasets with low-utility training instances (Dodge et al., 2021). In these settings, scalar accuracy-based heuristics often fail to distinguish useful learning signals from misleading ones, whereas GradAlign consistently prioritizes updates that improve downstream performance.

In summary, our contributions are as follows.

- We propose **GradAlign**, an online RL data-selection method that selects training samples whose updates are aligned with downstream performance improvement.
- We identify three challenging data regimes for RL data selection and conduct targeted experiments showing that GradAlign consistently outperforms prior baselines in these settings.

2 Related Work

2.1 Data Selection for LLMs

Data selection has emerged as a critical lever for enhancing the training efficiency and performance of LLMs (Koh & Liang, 2017). Early approaches largely relied on static heuristics to filter low-quality web text, using metrics such as perplexity thresholds, language identification, deduplication, and rule-based filters (Penedo et al., 2023; Raffel et al., 2019; Abbas et al., 2023; Li et al., 2024; Xie et al., 2023b; Lin et al., 2024). Beyond filtering, optimizing the data-source mix has been standard practice to balance general capabilities with specific domain knowledge (Xie et al., 2023a; Ye et al., 2024). More recent strategies employ diversity-based sampling (Zhang et al., 2024) or proxy models to score data quality (Wettig et al., 2025; Liu et al., 2024). However, these methods predominantly rely on human intuition or general-purpose heuristics rather than a principled evaluation of how specific data points contribute to the model’s learning trajectory, often leading to suboptimal alignment with downstream tasks (Xia et al., 2024).

2.2 Influence Estimation and Gradient Alignment

To transcend heuristic data selection, researchers have adopted Influence Functions (Koh et al., 2019; Grosse et al., 2023) to theoretically quantify the impact of individual training instances on validation loss. In the context of LLMs, these ideas have inspired a series of gradient-based data selection methods for pre-training and supervised fine-tuning (SFT) (Engstrom et al., 2024; Wang et al., 2023). Specifically, gradient alignment techniques—which select training samples whose gradients exhibit high cosine similarity with those of a trusted validation set—have proven effective for SFT (Zhang et al., 2024; Xia et al., 2024; Yu et al., 2024; Sun et al., 2025a). Recent work has further extended such estimators to preference data, for example, by pruning reward-model training sets using influence functions (Fein &

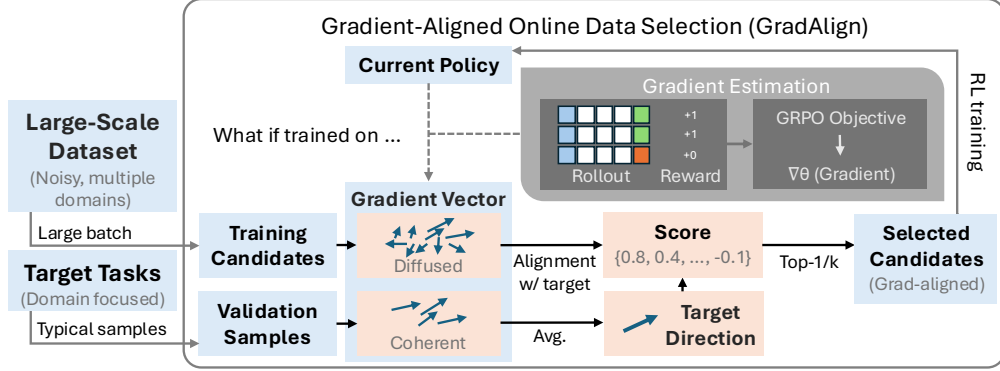


Figure 1: **Overview of GradAlign.** GradAlign uses a small validation set to estimate a target gradient direction and to score large-scale training candidates by gradient alignment, selecting the top-ranked fraction to form an adaptive RL online learning curriculum.

Aranguiz-Dias, 2025). However, these methods remain confined to offline, imitation-style paradigms (SFT and reward-model training), and their effectiveness for online RL policy optimization remains largely unexplored.

2.3 LLM Reinforcement Learning

Reinforcement Learning (RL) is increasingly central in unlocking advanced LLM capabilities, particularly for complex reasoning (DeepSeek-AI et al., 2025; Yu et al., 2025) and autonomous agentic tasks (Du et al., 2025). While standard Proximal Policy Optimization (PPO) (Schulman et al., 2017) remains a cornerstone, recent advances such as Group Relative Policy Optimization (GRPO) (Shao et al., 2024) and the reasoning-focused DeepSeek-R1 (DeepSeek-AI et al., 2025) demonstrate that RL can substantially outperform supervised baselines. However, RL algorithms are notoriously sensitive to data quality and reward noise; for example, training on trivial or unsolvable problems can induce instability or reward hacking (Hu et al., 2025; Gao et al., 2022; Lightman et al., 2023). As a result, although considerable effort has been devoted to selecting valuable data using heuristic signals such as success rates, gradients, or parameter distributions (Yu et al., 2025; Li et al., 2025b;a; Liang et al., 2025), the curriculum of *which* problems to present to LLMs, and *when*, is typically treated as static. This overlooks the potential of dynamic data selection to stabilize and accelerate RL training. In this work, we propose gradient alignment as a principled approach for dynamically selecting high-quality data during the RL process.

3 Preliminary

3.1 Problem Definition

We study an online reinforcement learning (RL) setting where the learner filters a large, noisy candidate pool to maximize downstream task performance. Formally, at each training round r , given the current policy π_θ , we receive M candidate problems $\mathcal{P}_r = \{p_{r,1}, \dots, p_{r,M}\}$. A data selector outputs a subset $\mathcal{S}_r \subseteq \mathcal{P}_r$ of size $|\mathcal{S}_r| = M/q$, where $q > 1$ is the selection ratio. The policy π_θ is then trained in $\mathcal{S}_r$ using binary rewards computed by a judge model against the reference answers.

To guide data selection toward informative problems, algorithms can access a small validation set $\mathcal{P}_v$ representing the downstream task (e.g., few-shot examples or public test cases). The final performance is evaluated on a held-out test set $\mathcal{P}_t$.

3.2 GRPO as the Reinforcement Learning Objective

We use Group Relative Policy Optimization (GRPO) (Shao et al., 2024) as a unified training objective to isolate the effect of data selection. For an input x sampled from dataset $\mathcal{D}$,

the policy π_θ generates k responses $\{y_j\}_{j=1}^k$ with corresponding rewards $\{r_j\}_{j=1}^k$. GRPO computes a normalized advantage: $\hat{A}_j = \frac{r_j - \bar{r}}{\sqrt{\frac{1}{k} \sum_{j=1}^k (r_j - \bar{r})^2 + \epsilon}}$, where $\bar{r} = \frac{1}{k} \sum_{j=1}^k r_j$. The policy is optimized using a clipped objective with KL regularization:

$$\mathcal{L}_{\text{GRPO}}(\theta) = \mathbb{E}_{x \sim \mathcal{D}, y \sim \pi_{\theta_{\text{old}}}} [\min(\rho_j(\theta) \hat{A}_j, \text{clip}(\rho_j(\theta), 1 - \epsilon, 1 + \epsilon) \hat{A}_j)] - \beta_{\text{KL}} \mathcal{D}_{\text{KL}}[\pi_\theta \parallel \pi_{\text{ref}}],$$

where $\rho_j(\theta) = \frac{\pi_\theta(y_j|x)}{\pi_{\theta_{\text{old}}}(y_j|x)}$, and β_{KL} is the coefficient for the KL penalty. For theoretical analysis, we define a surrogate objective $\tilde{\mathcal{L}}_{\text{GRPO}}$ that ignores clipping and KL regularization. Assuming on-policy sampling, where the rollout policy is the same as the current policy (i.e., $\pi_{\theta_{\text{old}}} = \pi_\theta$), this yields:

$$\tilde{\mathcal{L}}_{\text{GRPO}}(\theta) = \mathbb{E}_{x \sim \mathcal{D}, y \sim \pi_{\theta_{\text{old}}}} [\hat{A}_j \log \pi_\theta(y_j|x)].$$

Taking the gradient provides a stable policy-gradient estimator:

$$\nabla_\theta \tilde{\mathcal{L}}_{\text{GRPO}}(\theta) = \mathbb{E}_{x \sim \mathcal{D}, y \sim \pi_{\theta_{\text{old}}}} [\hat{A}_j \nabla_\theta \log \pi_\theta(y_j|x)].$$

While we instantiate our framework using GRPO, it only requires on-policy gradient estimates and is compatible with other RL objectives.

4 GradAlign

We propose **GradAlign**, an online data selection method for RL. While prior work often uses accuracy-based filtering (Yu et al., 2025), accuracy alone cannot determine which problems induce beneficial policy updates. GradAlign addresses this limitation by selecting high-quality samples through gradient alignment.

Our key insight is that a training problem is valuable if updating the policy on it improves performance on a trustworthy validation set $\mathcal{P}_v$. We approximate this improvement using first-order policy gradients. GradAlign treats the average policy gradient on the validation set as a target direction and ranks candidate problems by their alignment with this direction. As illustrated in Figure 1, GradAlign operates iteratively by estimating validation gradients, selecting aligned training problems, and updating the policy. Rollouts are resampled each round to dynamically adapt to the evolving policy.

4.1 Validation Improvement as a Proxy

Unlike supervised fine-tuning, where training data provides detailed solutions, RL only exposes reward signals, making it difficult to directly estimate the utility of individual training problems. We therefore use *validation improvement* as a proxy for problem utility: a training problem is useful if updating the policy on it improves performance on the validation set $\mathcal{P}_v$, which is more likely to transfer to the held-out test set. A direct approach would be to perform RL updates on each candidate problem and measure the resulting change on $\mathcal{P}_v$, but this is computationally prohibitive.

4.2 Validation Improvement via Policy Gradients

Because accuracy change is a discrete signal, ranking candidates precisely by it is expensive. We instead find that the gradient of the on-policy GRPO loss can serve as a surrogate for accuracy change, as formalized in the following theorem:

Theorem 4.1 (Unbiased expected-accuracy gradient estimation). *Under on-policy sampling, binary rewards, unbiased advantage estimation without normalization, and ignoring KL regularization and clipping, the policy-gradient estimator is unbiased for the gradient of expected accuracy.*

The proof is provided in Appendix C.1. We therefore approximate the expected change in validation accuracy using the gradient of the GRPO objective. Based on Theorem 4.1, under

a first-order approximation, the validation improvement induced by a candidate training problem is proportional to the inner product between the policy gradients computed on the validation set and on the candidate. In practice, the advantage estimator $\hat{A}$ is normalized for stable training. As the following theorem states, the normalized policy gradient remains directionally aligned, in expectation, with the expected accuracy gradient:

Theorem 4.2 (Direction preservation under advantage normalization). *Under on-policy sampling, binary rewards, and $k \geq 2$ i.i.d. rollouts per problem, group-wise advantage normalization biases the magnitude of the per-problem GRPO gradient but preserves its expected direction: the expected normalized GRPO gradient is a non-negative scalar multiple of the gradient of the expected accuracy, and the scalar is strictly positive whenever the expected accuracy lies in $(0, 1)$. Consequently, whenever $\nabla_{\theta} p_{\theta}(x) \neq 0$, the two gradients share the same direction.*

The proof is provided in Appendix C.2. Note that the binary-reward assumption is necessary; Appendix C.3 provides a counterexample in the general scalar-reward setting.

Because gradients are typically normalized in practice, their direction is more important than magnitude. Therefore, we use cosine similarity as the alignment score instead of the inner product. This choice is further supported by our ablation studies (see Section 7).

4.3 Algorithm Details

To estimate the validation policy gradient, we first compute the policy gradient for each validation problem. Specifically, for every problem p_i in the validation set $\mathcal{P}_v = \{p_1^v, \dots, p_{|\mathcal{P}_v|}^v\}$, we sample k_v rollouts $o_{i,1}, \dots, o_{i,k_v}$ with corresponding binary rewards $r_{i,1}, \dots, r_{i,k_v} \in \{0, 1\}$. We then compute the normalized advantages $\hat{A}_{i,j}$ following the GRPO surrogate objective described in Section 3.2. The gradient g_i for the problem p_i is given by

$$g_i^v = \nabla_{\theta} \tilde{\mathcal{L}}_{\text{GRPO}}(p_i^v) = \frac{1}{k_v} \sum_{j=1}^{k_v} \hat{A}_{i,j} \nabla_{\theta} \log \pi_{\theta}(o_{i,j} \mid p_i^v).$$

Averaging all gradients, we obtain the validation policy gradient $G_v = \frac{1}{|\mathcal{P}_v|} \sum_{i=1} g_i^v$.

In each training round r , for each candidate training problem $p_{r,i} \in \mathcal{P}_r$, we similarly estimate the GRPO policy gradient $g_{r,i}$ using sampled rollouts of size k_r . Since the training set is substantially larger than the validation set ($|\mathcal{P}_r| \gg |\mathcal{P}_v|$), we use fewer rollouts ($k_r < k_v$) for training candidates, as high-precision gradient estimates are less critical on training candidates. Finally, we rank the candidate training problems according to the cosine similarity between $g_{r,i}$ and G_v , selecting the top- $|S_r|$ candidates.

Our design differs from influence functions, which require fully trained models and second-order information (Koh & Liang, 2017). Second-order effects are unnecessary here, since each candidate is used for only a few updates, for which first-order approximations suffice.

4.4 On-Policy Resampling

Because the policy evolves during RL training, both the expected validation accuracy and the validation GRPO gradients change over time. To mitigate the bias introduced by stale rollouts in advantage estimation, we resample all validation problems and recompute the average validation gradient at the beginning of each data-selection round.

This design differs from Zhu et al. (2025), which estimates gradients using offline rollouts with importance sampling. Although importance sampling can be unbiased, behaviors that emerge as the policy evolves may fall outside the offline rollout distribution. It also exhibits high variance when behavior and target policies diverge, requiring more samples and incurring higher computational cost for stable estimation.

4.5 Regarding KL Regularization

Although we do not add KL regularization in our experiments to isolate the effect of data selection, we could modify the surrogate GRPO loss to incorporate a KL penalty for both

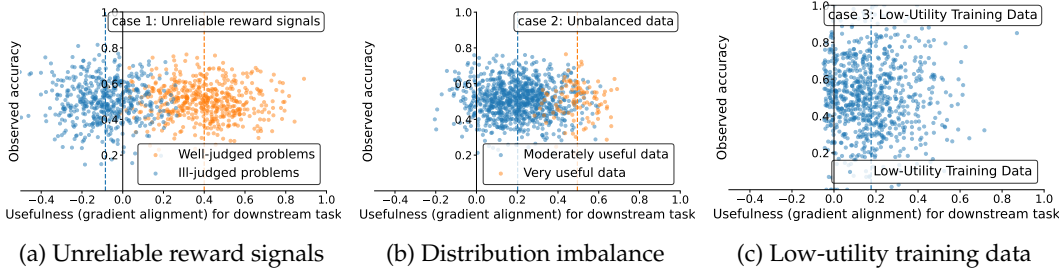


Figure 2: **Illustration of Three Challenging Data-Selection Scenarios.** Each panel shows a failure mode where accuracy-based filtering fails to identify training samples that improve downstream performance.

the training and validation sets. The modified surrogate objective would be formulated as:

$$\tilde{\mathcal{L}}_{\text{GRPO-KL}}(\theta) = \mathbb{E}_{x,y \sim \pi_\theta} [\hat{A}_j \log \pi_\theta(y_j|x)] - \beta_{\text{KL}} \mathcal{D}_{\text{KL}}[\pi_\theta \parallel \pi_{\text{ref}}].$$

In this formulation, we could still compute the gradient alignment by evaluating the gradient of this modified loss, $\nabla_\theta \tilde{\mathcal{L}}_{\text{GRPO-KL}}(\theta)$, for both the validation set to establish the target direction and the training candidates for ranking.

Overall Algorithm. The pseudocode for GradAlign is presented in Algorithm 1.

Algorithm 1 Grad-Aligned Data Selection (GRADALIGN)

- 1: **Input:** initial policy π_θ ; validation set $\mathcal{P}_v$; number of training rounds R ; candidate pools $\{\mathcal{P}_r\}_{r=1}^R$ with $|\mathcal{P}_r| = M$; selection ratio q (select M/q candidates per round); number of rollouts k_v and k_r ; normalization constant ϵ .
 - 2: **Output:** trained policy π_θ
 - 3: **for** $r = 1$ to R **do**
 - 4: **(Resample & compute validation gradient)**
 - 5: Initialize $G_v \leftarrow 0$.
 - 6: **for** each validation problem $p^v \in \mathcal{P}_v$ **do**
 - 7: Roll out k_v samples $\{o_j\}_{j=1}^{k_v} \sim \pi_\theta(\cdot | p^v)$ and obtain rewards $\{r_j\}_{j=1}^{k_v}$.
 - 8: Compute GRPO gradient:

$$g_p^v \leftarrow \frac{1}{k_v} \sum_{j=1}^{k_v} \hat{A}_j \nabla_\theta \log \pi_\theta(o_j | p^v).$$
 - 9: Accumulate $G_v \leftarrow G_v + g_p^v$.
 - 10: **end for**
 - 11: Average $G_v \leftarrow \frac{1}{|\mathcal{P}_v|} G_v$.
 - 12: **(Score candidates by gradient alignment)**
 - 13: **for** each candidate problem $p_{r,i} \in \mathcal{P}_r$ **do**
 - 14: Roll out k_r samples and compute its GRPO gradient estimate $g_{r,i}$ (same procedure as above).
 - 15: Score $s_{r,i} \leftarrow \cos(g_{r,i}, G_v)$.
 - 16: **end for**
 - 17: Select $\mathcal{S}_r \leftarrow$ top- M/q candidates in $\mathcal{P}_r$ by $s_{r,i}$.
 - 18: **(RL update on selected curriculum)**
 - 19: Update θ by GRPO training on $\mathcal{S}_r$.
 - 20: **end for**
-

5 Considerations

5.1 Applicable Scenarios

GradAlign is particularly suitable for settings where the training rewards are noisy or unreliable, as the gradients are more dispersed. We highlight three common scenarios: the **Distinguishability Test with Noisy Rewards**, an **Unbalanced Dataset**, and **Low-Utility Training Data**. Details can be found in Appendix D with visualization in Figure 2.

5.2 Computational Overhead

GradAlign introduces additional computation to estimate gradient alignment, but the relative overhead remains tractable. Let $|\mathcal{P}_r|$ and $|\mathcal{P}_v|$ be the number of training candidates and validation problems, evaluated with k_r and k_v rollouts, respectively. When training a selected subset $|\mathcal{S}_r|$ using n_t rollouts per problem, the relative computational overhead is:

$$\frac{|\mathcal{P}_r|k_r + |\mathcal{P}_v|k_v}{|\mathcal{S}_r|n_t}.$$

In our standard setting ($|\mathcal{P}_r| = 5120$, $|\mathcal{S}_r| = 1280$, $k_r = k_v = 16$, $n_t = 128$, $|\mathcal{P}_v| \leq 200$), GradAlign adds $\approx 65\%$ compute overhead. By comparison, Align (Li et al., 2025a) adds $\approx 50\%$ overhead ($\frac{|\mathcal{P}_r|k_r}{|\mathcal{S}_r|n_t}$), and this doubles if gradients are not projected for storage to avoid recomputation. Accuracy-based greedy filtering (AccGreedy, see Section 6.2) incurs an overhead of $C \frac{|\mathcal{P}_r|k_r}{|\mathcal{S}_r|n_t}$, where C denotes the relative cost of rollouts compared to full updates. This overhead is justified for three reasons: (1) the small validation set makes target gradient computation negligible; (2) candidate gradient computation adds only a constant-factor overhead over accuracy-based filtering (Kaplan et al., 2020); and (3) prioritizing highly informative problems improves sample efficiency, significantly reducing ineffective updates and yielding better final performance than unfiltered training. A detailed per-round FLOP breakdown and a memory analysis are provided in Appendix F, confirming that both the compute and memory overhead are invariant to model scale.

6 Experiments

In this section, we evaluate the effectiveness of GradAlign across diverse scenarios. Through these experiments, we aim to verify GradAlign’s ability to select high-quality training data for specific domains from a large and noisy training dataset.

6.1 Experimental Setup

RL Framework All experiments are conducted using GRPO (Shao et al., 2024). We use a sparse binary reward indicating whether the final answer is correct, and all rewards are provided by Qwen2.5-72B-Instruct as the judge. We train base models in the 1.5B–8B range, and the model used in each setting is stated with the corresponding scenario below. Detailed RL training hyperparameters can be found in Appendix A. Recalling the three scenarios mentioned in Section 5.1, we conduct experiments to inspect each of them.

Scenario 1: Distinguishability Test with Noisy Rewards To rigorously assess the selector’s capacity to discriminate valid learning signals from stochastic noise, we conduct a controlled experiment that systematically perturbs the training set. We train Qwen3-8B-Base on the DAPO training corpus (Yu et al., 2025), introducing controlled corruptions to 50% of the samples, whereby rewards are decoupled from model outputs and instead sampled from a Bernoulli distribution ($p = 0.5$). This procedure generates a subset of data that statistically mimics the difficulty of “complex” problems (50% pass rate) while being devoid of helpful gradient information.

Scenario 2: Unbalanced Dataset To evaluate alignment capabilities under distributional shift, we train Qwen2.5-1.5B-Math on a heterogeneous dataset comprising 4k Countdown Game problems (Gandhi et al., 2024), 20k WebInstruct problems, and 10k DAPO problems (Yu et al., 2025). Given that the target domain (Countdown Game) constitutes only $\approx 12\%$ of the aggregate pool, this setting challenges methods to retrieve domain-specific problems from an imbalanced distribution. We employ a selection ratio of $q = 20$ (selecting the top 5%), with validation and test sets consisting exclusively of Countdown Game problems.

Method	Val	Held-out Test			Ratio of Corrupted ($\downarrow$)	
	AMC22	AMC23	AIME2425	Test Avg.	Step 0	Step 100
GradAlign (Ours)	49.1	68.2	20.5	44.4	17.8%	29.9%
Random	47.5	58.5	15.8	37.2	50.0%*	50.0%*
AccGreedy	18.1	28.5	1.9	15.2	81.8%	87.1%
Align	45.7	<u>61.5</u>	<u>16.6</u>	<u>39.1</u>	<u>33.5%</u>	<u>47.5%</u>

Table 1: **Scenario 1: Controlled Noise Injection (50% Corrupt)**. Downstream performance on AMC/AIME benchmarks and the ratio of corrupted problems in selected training data. Best results are in **bold**; second-best are underlined. All results are reported at step 100. The Test Avg. is computed over the held-out test columns only and excludes the validation (Val) column. *Random selection has an expected corruption ratio of 50.0% by construction.

Scenario 3: Low-Utility Training Data The training set is the WebInstruct-filtered-unverified dataset (Ma et al., 2025), a large-scale collection (808k problems) gathered from internet sources and reflecting typical noisy, uncurated training data. For the two Qwen3-8B-Base settings, we apply a preliminary difficulty filter (pass rates $\in [0.2, 0.8]$) before selection to improve computational efficiency. We consider three target domains.

SuperGPQA & TheoremQA: Using Qwen3-8B-Base, we draw a single *combined* validation set from SuperGPQA (Team et al., 2025) and TheoremQA (Chen et al., 2023) to guide selection, and evaluate both held-out test sets; this probes whether a mixed-domain validation signal can improve multiple target domains at once.

Mathematical Reasoning Task: Using Qwen3-8B-Base, selection is guided by AMC22 (Mathematical Association of America, 2022), with performance evaluated on AMC23 and AIME2425. This is a *partial* validation-test mismatch: the probe is AMC22, but the test set includes the harder AIME2425 from different competitions and years, testing whether GradAlign still transfers when the validation set only partially represents the target.

MMLU-Pro: Using Qwen2.5-1.5B-Math-Instruct and the unfiltered corpus, the target domain is the Math category of MMLU-Pro (Wang et al., 2024), divided into validation and test sets. This evaluates GradAlign’s ability to identify problems beneficial to multiple-choice mathematical reasoning.

6.2 Baselines

We compare GradAlign with three common data selection strategies:

1. **Random Selection:** Uniformly samples training data. Serves as a lower bound.
2. **Accuracy Greedy (AccGreedy)** (Sun et al., 2025c): Prioritizes samples with pass rates closest to 50%, based on the heuristic that problems near the decision boundary provide the strongest signal.
3. **LearnAlign (Align)** (Li et al., 2025a): A method that selects data based on gradient similarity within the training set without using external validation guidance.

6.3 Experimental Results

We report results for each scenario below, focusing on downstream accuracy and, where applicable, the composition of selected training data.

Results on Noisy Rewards Table 1 presents the results of the noise-injection experiment. GradAlign significantly outperforms all baselines on most datasets. We also report the ratio of corrupted problems selected at steps 0 and 100 in Table 1. GradAlign consistently selects the lowest proportion of corrupted data. AccGreedy selects the most corrupted data (over 80%) because the corrupted samples have pass rates near 50%. Figure 5 shows the training accuracy curves, where GradAlign outperforms all baselines.

Method	Performance		Ratio of Countdown ($\uparrow$)	
	Countdown(Val)	Countdown(Test)	Step 0	Step 50
GradAlign (Ours)	33.8	34.0	92.9%	62.1%
Random	14.1	10.4	11.8%*	11.8%*
AccGreedy	21.7	15.2	7.8%	1.5%
Align	<u>31.1</u>	<u>28.2</u>	<u>47.2%</u>	<u>60.6%</u>

Table 2: **Scenario 2: Unbalanced Dataset.** Downstream performance on Countdown Game and the ratio of Countdown problems in selected training data. Best results are in **bold**; second-best are underlined. All results are reported at step 50. *Random selection has an expected Countdown ratio of 11.8% by construction.

Method	SuperGPQA		TheoremQA		AIME/AMC			MMLU-Pro		Test Avg.
	Val	Test	Val	Test	Val	AMC23	AIME2425	Val	Test	
GradAlign	33.5	35.7	55.7	59.3	48.8	64.2	18.3	59.8	57.9	47.1
Random	29.6	<u>35.5</u>	<u>53.5</u>	55.2	<u>47.6</u>	60.0	17.2	<u>57.8</u>	<u>57.1</u>	<u>45.0</u>
AccGreedy	<u>31.7</u>	31.6	53.1	54.9	45.0	<u>62.3</u>	<u>17.8</u>	57.5	55.1	44.3
Align	29.8	32.7	52.8	<u>57.9</u>	46.4	61.5	16.1	57.1	56.6	<u>45.0</u>

Table 3: **Scenario 3: Low-Utility Training Data.** Downstream performance on SuperGPQA, TheoremQA, AMC/AIME, and MMLU-Pro. The training set is WebInstruct. Best results are in **bold**; second-best are underlined. All results are reported at step 100. The Test Avg. covers only the held-out test columns, excluding all Val columns.

Results on Unbalanced Dataset Table 2 shows results for the unbalanced dataset setting. GradAlign achieves the best performance on both validation and test sets, followed by Align. Table 2 shows the ratio of the selected Countdown Game problems. AccGreedy rarely selects these problems because of the low initial accuracy on Countdown. This result demonstrates the importance of gradient-informed data selection for domain-specific filtering.

Performance on Low-Utility Data WebInstruct is a large-scale, automatically constructed corpus in which many problems are on-distribution but overly simple, providing valid rewards yet little learning signal for the target task. On this corpus, GradAlign attains the highest average performance across all settings, while some baselines remain competitive on individual splits (Table 3). It achieves the best results across all datasets, e.g., 59.3% on TheoremQA-test, compared to 54.9% for AccGreedy and 57.9% for Align.

Although GradAlign requires more computation per step, it converges faster and reaches higher final performance than random selection. As shown in Figure 3, in Scenario 3, random selection does not plateau but degrades after step 120, as continued training on noisy data erodes the policy, so additional training would not close the gap.

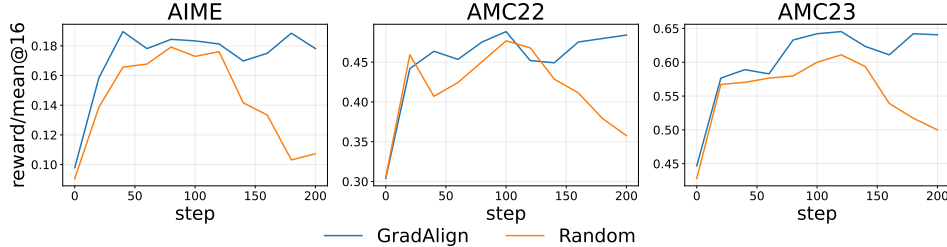


Figure 3: **Training Accuracy Curve on AIME2425, AMC22 and AMC23 (Scenario 3).** GradAlign (ours) achieves the strongest performance.

7 Ablation Study

In this section, we conduct ablation studies on three design choices: the alignment score metric (cosine similarity versus inner product), the sample size required for stable gradient estimation, and training directly on the validation set instead of the training set.

Inner product vs. cosine similarity. We compare cosine similarity and the inner product as alignment-score measures in the noisy-reward setting. As shown in Figure 4 (left), cosine similarity achieves higher performance. Further analysis (see Appendix B) indicates that cosine similarity better separates clean samples from corrupted samples.

Directly training on validation set.

We study the performance of training directly on the validation set (AMC22) instead of selecting data from the training set. Figure 4 (right) shows that while direct training achieves higher performance on AMC22 itself, it fails to generalize to test sets, unlike GradAlign. Intuitively, training directly on the small validation set can lead to memorizing its specific trajectories, whereas GradAlign preserves diversity.

Validation set size To test the sensitivity of our method to the size of the validation set, we conduct an ablation study in the *Unbalanced dataset* setting. As shown in Table 4, GradAlign achieves a stable improvement over baselines even with small validation set sizes. We also measure gradient noise in GRPO, and results are given in Appendix E.

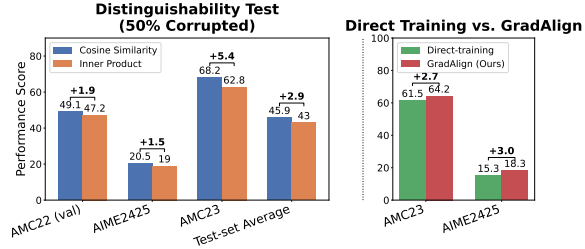


Figure 4: **Performance Analysis of Gradient Alignment and Direct Training.** (Left) Under 50% reward noise, cosine similarity outperforms the inner product. (Right) Direct training on the validation set overfits, whereas GradAlign generalizes. All results are reported at step 100.

	GradAlign (val. set size)			Baselines		
	10	30	100	Random	AccGreedy	Align
Test Set Accuracy	32.7	32.1	34.0	10.4	15.2	28.2

Table 4: Test set performance at step 50 on the Countdown task, varying the GradAlign validation set size (10/30/100 problems). GradAlign stays well above all baselines.

8 Conclusion

We presented GradAlign, a gradient-aligned data selection method for LLM reinforcement learning that uses a small validation set to prioritize training problems whose policy-gradient updates most improve downstream performance. Across noisy rewards, distribution imbalance, and low-utility web data, GradAlign yields more stable training and stronger final results than heuristic and prior baselines. These results suggest that policy-gradient direction is a reliable signal for RL data attribution and curriculum learning, offering a broadly applicable way to improve training efficiency in LLM reinforcement learning.

Limitations GradAlign relies on a validation set representative of the downstream task: it selects training samples aligned with the validation distribution, so a mismatched probe (e.g., validating on MMLU but testing on AIME) would steer selection toward the wrong problems. It thus targets settings with a relevant validation signal rather than task-agnostic selection. It also adds compute overhead ($\approx 65\%$ over unfiltered training; Section 5.2), though this cost is bounded and scale-invariant.

Acknowledgments

This work was supported in part by the National Science Foundation under Grant No. DMS-2502281.

References

- Amro Abbas, Kushal Tirumala, Daniel Simig, Surya Ganguli, and Ari S. Morcos. Semd-edup: Data-efficient learning at web-scale through semantic deduplication. *ArXiv*, abs/2303.09540, 2023.
- Alon Albalak, Duy Phung, Nathan Lile, Rafael Rafailov, Kanishk Gandhi, Louis Castricato, Anikait Singh, Chase Blagden, Violet Xiang, Dakota Mahan, et al. Big-math: A large-scale, high-quality math dataset for reinforcement learning in language models. *arXiv preprint arXiv:2502.17387*, 2025.
- Wenhu Chen, Ming Yin, Max Ku, Pan Lu, Yixin Wan, Xueguang Ma, Jianyu Xu, Xinyi Wang, and Tony Xia. Theoremqa: A theorem-driven question answering dataset. In *The 2023 Conference on Empirical Methods in Natural Language Processing*, 2023.
- DeepSeek-AI, Daya Guo, Dejian Yang, Haowei Zhang, Jun-Mei Song, Ruoyu Zhang, Runxin Xu, Qihao Zhu, Shirong Ma, Peiyi Wang, Xiaoling Bi, Xiaokang Zhang, Xingkai Yu, Yu Wu, Z. F. Wu, Zhibin Gou, Zhihong Shao, Zhuoshu Li, Ziyi Gao, Aixin Liu, Bing Xue, Bing-Li Wang, Bochao Wu, Bei Feng, Chengda Lu, Chenggang Zhao, Chengqi Deng, Chenyu Zhang, Chong Ruan, Damai Dai, Deli Chen, Dong-Li Ji, Erhang Li, Fangyun Lin, Fucong Dai, Fuli Luo, Guangbo Hao, Guanting Chen, Guowei Li, H. Zhang, Han Bao, Hanwei Xu, Haocheng Wang, Honghui Ding, Huajian Xin, Huazuo Gao, Hui Qu, Hui Li, Jianzhong Guo, Jiashi Li, Jiawei Wang, JingChang Chen, Jingyang Yuan, Junjie Qiu, Junlong Li, Jiong Cai, Jiaqi Ni, Jian Liang, Jin Chen, Kai Dong, Kai Hu, Kaige Gao, Kang Guan, Kexin Huang, Kuai Yu, Lean Wang, Lecong Zhang, Liang Zhao, Litong Wang, Liyue Zhang, Lei Xu, Leyi Xia, Mingchuan Zhang, Minghua Zhang, M. Tang, Meng Li, Miaojuan Wang, Mingming Li, Ning Tian, Panpan Huang, Peng Zhang, Qiancheng Wang, Qinyu Chen, Qiushi Du, Ruiqi Ge, Ruisong Zhang, Ruizhe Pan, Runji Wang, R. J. Chen, Ruiqi Jin, Ruyi Chen, Shanghao Lu, Shangyan Zhou, Shanhuang Chen, Shengfeng Ye, Shiyu Wang, Shuiping Yu, Shunfeng Zhou, Shuting Pan, S. S. Li, Shuang Zhou, Shao-Kang Wu, Tao Yun, Tian Pei, Tianyu Sun, T. Wang, Wangding Zeng, Wanjia Zhao, Wen Liu, Wenfeng Liang, Wenjun Gao, Wen-Xia Yu, Wentao Zhang, Wangding Xiao, Wei An, Xiaodong Liu, Xiaohan Wang, Xiaokang Chen, Xiaotao Nie, Xin Cheng, Xin Liu, Xin Xie, Xingchao Liu, Xinyu Yang, Xinyuan Li, Xuecheng Su, Xuheng Lin, X. Q. Li, Xiangyu Jin, Xi-Cheng Shen, Xiaosha Chen, Xiaowen Sun, Xiaoxiang Wang, Xinnan Song, Xinyi Zhou, Xianzu Wang, Xinxia Shan, Y. K. Li, Y. Q. Wang, Y. X. Wei, Yang Zhang, Yanhong Xu, Yao Li, Yao Zhao, Yaofeng Sun, Yaohui Wang, Yi Yu, Yichao Zhang, Yifan Shi, Yi Xiong, Ying He, Yishi Piao, Yisong Wang, Yixuan Tan, Yiyang Ma, Yiyuan Liu, Yongqiang Guo, Yuan Ou, Yuduan Wang, Yue Gong, Yu-Jing Zou, Yujia He, Yunfan Xiong, Yu-Wei Luo, Yu mei You, Yuxuan Liu, Yuyang Zhou, Y. X. Zhu, Yanping Huang, Yao Li, Yi Zheng, Yuchen Zhu, Yunxiang Ma, Ying Tang, Yukun Zha, Yuting Yan, Zehui Ren, Zehui Ren, Zhangli Sha, Zhe Fu, Zhean Xu, Zhenda Xie, Zhen guo Zhang, Zhewen Hao, Zhicheng Ma, Zhigang Yan, Zhiyu Wu, Zihui Gu, Zijia Zhu, Zijun Liu, Zi-An Li, Ziwei Xie, Ziyang Song, Zizheng Pan, Zhen Huang, Zhipeng Xu, Zhongyu Zhang, and Zhen Zhang. Deepseek-r1 incentivizes reasoning in llms through reinforcement learning. *Nature*, 645:633 – 638, 2025.
- Jesse Dodge, Maarten Sap, Ana Marasović, William Agnew, Gabriel Ilharco, Dirk Groeneveld, Margaret Mitchell, and Matt Gardner. Documenting large webtext corpora: A case study on the colossal clean crawled corpus. *arXiv preprint arXiv:2104.08758*, 2021.
- Weihua Du, Hailei Gong, Zhan Ling, Kang Liu, Lingfeng Shen, Xuesong Yao, Yufei Xu, Dingyuan Shi, Yiming Yang, and Jiecao Chen. Generalizable end-to-end tool-use rl with synthetic codegym. *arXiv preprint arXiv:2509.17325*, 2025.
- Logan Engstrom, Axel Feldmann, and Aleksander Madry. Dsdm: Model-aware dataset selection with datamodels. *ArXiv*, abs/2401.12926, 2024.

- Daniel Fein and Gabriel Aranguiz-Dias. Influence functions for preference dataset pruning. In *NeurIPS 2025 Workshop*, 2025.
- Kanishk Gandhi, Denise Lee, Gabriel Grand, Muxin Liu, Winson Cheng, Archit Sharma, and Noah D Goodman. Stream of search (sos): Learning to search in language. *arXiv preprint arXiv:2404.03683*, 2024.
- Leo Gao, John Schulman, and Jacob Hilton. Scaling laws for reward model overoptimization. In *International Conference on Machine Learning*, 2022.
- Roger Baker Grosse, Juhan Bae, Cem Anil, Nelson Elhage, Alex Tamkin, Amirhossein Tajdini, Benoit Steiner, Dustin Li, Esin Durmus, Ethan Perez, Evan Hubinger, Kamil.e Lukovsiut.e, Karina Nguyen, Nicholas Joseph, Sam McCandlish, Jared Kaplan, and Sam Bowman. Studying large language model generalization with influence functions. *ArXiv*, abs/2308.03296, 2023.
- Zhiwei He, Tian Liang, Jiahao Xu, Qiuzhi Liu, Xingyu Chen, Yue Wang, Linfeng Song, Dian Yu, Zhenwen Liang, Wenxuan Wang, et al. Deepmath-103k: A large-scale, challenging, decontaminated, and verifiable mathematical dataset for advancing reasoning. *arXiv preprint arXiv:2504.11456*, 2025.
- Jingcheng Hu, Yinmin Zhang, Qi Han, Daxin Jiang, Xiangyu Zhang, and Heung-Yeung Shum. Open-reasoner-zero: An open source approach to scaling up reinforcement learning on the base model. *arXiv preprint arXiv:2503.24290*, 2025.
- Jared Kaplan, Sam McCandlish, Tom Henighan, Tom B Brown, Benjamin Chess, Rewon Child, Scott Gray, Alec Radford, Jeffrey Wu, and Dario Amodei. Scaling laws for neural language models. *arXiv preprint arXiv:2001.08361*, 2020.
- Robert Kirk, Ishita Mediratta, Christoforos Nalmpantis, Jelena Luketina, Eric Hambro, Edward Grefenstette, and Roberta Raileanu. Understanding the effects of rlhf on llm generalisation and diversity. *arXiv preprint arXiv:2310.06452*, 2023.
- Pang Wei Koh and Percy Liang. Understanding black-box predictions via influence functions. In *International Conference on Machine Learning*, 2017.
- Pang Wei Koh, Kai-Siang Ang, Hubert Hua Kian Teo, and Percy Liang. On the accuracy of influence functions for measuring group effects. In *Neural Information Processing Systems*, 2019.
- Jeffrey Li, Alex Fang, Georgios Smyrnis, Maor Ivgi, Matt Jordan, Samir Yitzhak Gadre, Hritik Bansal, Etash Kumar Guha, Sedrick Scott Keh, Kushal Arora, Saurabh Garg, Rui Xin, Niklas Muennighoff, Reinhard Heckel, Jean-Pierre Mercat, Mayee Chen, Suchin Gururangan, Mitchell Wortsman, Alon Albalak, Yonatan Bitton, Marianna Nezhurina, Amro Abbas, Cheng-Yu Hsieh, Dhruba Ghosh, Josh Gardner, Maciej Kilian, Hanlin Zhang, Rulin Shao, Sarah Pratt, Sunny Sanyal, Gabriel Ilharco, Giannis Daras, Kalyani Marathe, Aaron Gokaslan, Jieyu Zhang, Khyathi Chandu, Thao Nguyen, Igor Vasiljevic, Sham M. Kakade, Shuran Song, Sujay Sanghavi, Fartash Faghri, Sewoong Oh, Luke S. Zettlemoyer, Kyle Lo, Alaaeldin El-Nouby, Hadi Pouransari, Alexander Toshev, Stephanie Wang, Dirk Groeneveld, Luca Soldani, Pang Wei Koh, Jenia Jitsev, Thomas Kollar, Alexandros G. Dimakis, Yair Carmon, Achal Dave, Ludwig Schmidt, and Vaishaal Shankar. Datacomp-lm: In search of the next generation of training sets for language models. *ArXiv*, abs/2406.11794, 2024.
- Shipeng Li, Shikun Li, Zhiqin Yang, Xinghua Zhang, Gaode Chen, Xiaobo Xia, Hengyu Liu, and Zhe Peng. Learnalign: Reasoning data selection for reinforcement learning in large language models based on improved gradient alignment. *arXiv preprint arXiv:2506.11480*, 2025a.
- Xuefeng Li, Haoyang Zou, and Pengfei Liu. Limr: Less is more for rl scaling. *arXiv preprint arXiv:2502.11886*, 2025b.

- Zhenwen Liang, Sidi Lu, Wenhao Yu, Kishan Panaganti, Yujun Zhou, Haitao Mi, and Dong Yu. Can llms guide their own exploration? gradient-guided reinforcement learning for llm reasoning. *arXiv preprint arXiv:2512.15687*, 2025.
- Hunter Lightman, Vineet Kosaraju, Yura Burda, Harrison Edwards, Bowen Baker, Teddy Lee, Jan Leike, John Schulman, Ilya Sutskever, and Karl Cobbe. Let’s verify step by step. *ArXiv*, abs/2305.20050, 2023.
- Zheng-Wen Lin, Zhibin Gou, Yeyun Gong, Xiao Liu, Yelong Shen, Ruochen Xu, Chen Lin, Yujiu Yang, Jian Jiao, Nan Duan, and Weizhu Chen. Rho-1: Not all tokens are what you need. *ArXiv*, abs/2404.07965, 2024.
- Qian Liu, Xiaosen Zheng, Niklas Muennighoff, Guangtao Zeng, Longxu Dou, Tianyu Pang, Jing Jiang, and Min Lin. Regmix: Data mixture as regression for language model pre-training. *ArXiv*, abs/2407.01492, 2024.
- Xueguang Ma, Qian Liu, Dongfu Jiang, Ge Zhang, Zejun Ma, and Wenhui Chen. General-Reasoner: Advancing llm reasoning across all domains. *arXiv:2505.14652*, 2025. URL <https://arxiv.org/abs/2505.14652>.
- Mathematical Association of America. American Mathematics Competitions 2022, 2022. URL <https://maa.org/student-programs/amc>.
- Guilherme Penedo, Quentin Malartic, Daniel Hesslow, Ruxandra-Aimée Cojocaru, Hamza Alobeidli, Alessandro Cappelli, Baptiste Pannier, Ebtesam Almazrouei, and Julien Launay. The refinedweb dataset for falcon llm: Outperforming curated corpora with web data only. In *Neural Information Processing Systems*, 2023.
- Colin Raffel, Noam Shazeer, Adam Roberts, Katherine Lee, Sharan Narang, Michael Matena, Yanqi Zhou, Wei Li, and Peter J. Liu. Exploring the limits of transfer learning with a unified text-to-text transformer. *J. Mach. Learn. Res.*, 21:140:1–140:67, 2019.
- Samyam Rajbhandari, Jeff Rasley, Olatunji Ruwase, and Yuxiong He. ZeRO: Memory optimizations toward training trillion parameter models. In *SC20: International Conference for High Performance Computing, Networking, Storage and Analysis*, pp. 1–16. IEEE, 2020.
- John Schulman, Filip Wolski, Prafulla Dhariwal, Alec Radford, and Oleg Klimov. Proximal policy optimization algorithms. *ArXiv*, abs/1707.06347, 2017.
- Zhihong Shao, Peiyi Wang, Qihao Zhu, Runxin Xu, Junxiao Song, Xiao Bi, Haowei Zhang, Mingchuan Zhang, YK Li, Yang Wu, et al. Deepseekmath: Pushing the limits of mathematical reasoning in open language models. *arXiv preprint arXiv:2402.03300*, 2024.
- Joar Skalse, Nikolaus Howe, Dmitrii Krashenninikov, and David Krueger. Defining and characterizing reward gaming. *Advances in Neural Information Processing Systems*, 35: 9460–9471, 2022.
- Weiwei Sun, Haokun Liu, Nikhil Kandpal, Colin Raffel, and Yiming Yang. Enhancing training data attribution with representational optimization. *ArXiv*, abs/2505.18513, 2025a.
- Weiwei Sun, Xuhui Zhou, Weihua Du, Xingyao Wang, Sean Welleck, Graham Neubig, Maarten Sap, and Yiming Yang. Training proactive and personalized llm agents. *arXiv preprint arXiv:2511.02208*, 2025b.
- Yifan Sun, Jingyan Shen, Yibin Wang, Tianyu Chen, Zhendong Wang, Mingyuan Zhou, and Huan Zhang. Improving data efficiency for llm reinforcement fine-tuning through difficulty-targeted online data selection and rollout replay. *arXiv preprint arXiv:2506.05316*, 2025c.
- Richard S Sutton, David McAllester, Satinder Singh, and Yishay Mansour. Policy gradient methods for reinforcement learning with function approximation. *Advances in neural information processing systems*, 12, 1999.

- M-A-P Team, Xinrun Du, Yifan Yao, Kaijing Ma, Bingli Wang, Tianyu Zheng, Kang Zhu, Minghao Liu, Yiming Liang, Xiaolong Jin, Zhenlin Wei, Chujie Zheng, Kaixin Deng, Shian Jia, Sichao Jiang, Yiyan Liao, Rui Li, Qinrui Li, Sirun Li, Yizhi Li, Yunwen Li, Dehua Ma, Yuansheng Ni, Haoran Que, Qiyao Wang, Zhoufutu Wen, Siwei Wu, Tianshun Xing, Ming Xu, Zhenzhu Yang, Zekun Moore Wang, Juntao Zhou, Yuelin Bai, Xingyuan Bu, Chenglin Cai, Liang Chen, Yifan Chen, Chengtuo Cheng, Tianhao Cheng, Keyi Ding, Siming Huang, Yun Huang, Yaoru Li, Yizhe Li, Zhaoqun Li, Tianhao Liang, Chengdong Lin, Hongquan Lin, Yinghao Ma, Tianyang Pang, Zhongyuan Peng, Zifan Peng, Qige Qi, Shi Qiu, Xingwei Qu, Shanghaoran Quan, Yizhou Tan, Zili Wang, Chenqing Wang, Hao Wang, Yiya Wang, Yubo Wang, Jiajun Xu, Kexin Yang, Ruibin Yuan, Yuanhao Yue, Tianyang Zhan, Chun Zhang, Jinyang Zhang, Xiyue Zhang, Xingjian Zhang, Yue Zhang, Yongchi Zhao, Xiangyu Zheng, Chenghua Zhong, Yang Gao, Zhoujun Li, Dayiheng Liu, Qian Liu, Tianyu Liu, Shiwen Ni, Junran Peng, Yujia Qin, Wenbo Su, Guoyin Wang, Shi Wang, Jian Yang, Min Yang, Meng Cao, Xiang Yue, Zhaoxiang Zhang, Wangchunshu Zhou, Jiaheng Liu, Qunshu Lin, Wenhao Huang, and Ge Zhang. Supergpqa: Scaling llm evaluation across 285 graduate disciplines, 2025. URL <https://arxiv.org/abs/2502.14739>.
- Xiao Wang, Wei Zhou, Qi Zhang, Jie Zhou, Songyang Gao, Junzhe Wang, Menghan Zhang, Xiang Gao, Yunwen Chen, and Tao Gui. Farewell to aimless large-scale pretraining: Influential subset selection for language model. In *Annual Meeting of the Association for Computational Linguistics*, 2023.
- Yubo Wang, Xueguang Ma, Ge Zhang, Yuansheng Ni, Abhranil Chandra, Shiguang Guo, Weiming Ren, Aaran Arulraj, Xuan He, Ziyang Jiang, et al. Mmlu-pro: A more robust and challenging multi-task language understanding benchmark. *arXiv preprint arXiv:2406.01574*, 2024.
- Alexander Wettig, Kyle Lo, Sewon Min, Hanna Hajishirzi, Danqi Chen, and Luca Soldaini. Organize the web: Constructing domains enhances pre-training data curation. *ArXiv*, abs/2502.10341, 2025.
- Mengzhou Xia, Sadhika Malladi, Suchin Gururangan, Sanjeev Arora, and Danqi Chen. Less: Selecting influential data for targeted instruction tuning. *ArXiv*, abs/2402.04333, 2024.
- Sang Michael Xie, Hieu Pham, Xuanyi Dong, Nan Du, Hanxiao Liu, Yifeng Lu, Percy Liang, Quoc V. Le, Tengyu Ma, and Adams Wei Yu. Doremi: Optimizing data mixtures speeds up language model pretraining. *ArXiv*, abs/2305.10429, 2023a.
- Sang Michael Xie, Shibani Santurkar, Tengyu Ma, and Percy Liang. Data selection for language models via importance resampling. *ArXiv*, abs/2302.03169, 2023b.
- John Yang, Kilian Lieret, Carlos E Jimenez, Alexander Wettig, Kabir Khandpur, Yanzhe Zhang, Binyuan Hui, Ofir Press, Ludwig Schmidt, and Diyi Yang. Swe-smith: Scaling data for software engineering agents. *arXiv preprint arXiv:2504.21798*, 2025.
- Jiasheng Ye, Peiju Liu, Tianxiang Sun, Yunhua Zhou, Jun Zhan, and Xipeng Qiu. Data mixing laws: Optimizing data mixtures by predicting language modeling performance. *ArXiv*, abs/2403.16952, 2024.
- Qiyang Yu, Zheng Zhang, Ruofei Zhu, Yufeng Yuan, Xiaochen Zuo, Yu Yue, Weinan Dai, Tiantian Fan, Gaohong Liu, Lingjun Liu, et al. Dapo: An open-source llm reinforcement learning system at scale. *arXiv preprint arXiv:2503.14476*, 2025.
- Zichun Yu, Spandan Das, and Chenyan Xiong. Mates: Model-aware data selection for efficient pretraining with data influence models. *ArXiv*, abs/2406.06046, 2024.
- Xiang Yue, Tianyu Zheng, Ge Zhang, and Wenhao Chen. Mammoth2: Scaling instructions from the web. *Advances in Neural Information Processing Systems*, 37:90629–90660, 2024.
- Chi Zhang, Huaping Zhong, Kuan Zhang, Chengliang Chai, Rui Wang, Xinlin Zhuang, Tianyi Bai, Jiantao Qiu, Lei Cao, Ye Yuan, Guoren Wang, and Conghui He. Harnessing diversity for important data selection in pretraining large language models. *ArXiv*, abs/2409.16986, 2024.

Erle Zhu, Dazhi Jiang, Yuan Wang, Xujun Li, Jiale Cheng, Yuxian Gu, Yilin Niu, Aohan Zeng, Jie Tang, Minlie Huang, et al. Data-efficient rlvr via off-policy influence guidance. *arXiv preprint arXiv:2510.26491*, 2025.

A Detailed Hyperparameters

We detail the hyperparameters used for GRPO training and the GradAlign selection process in Table 5. For all experiments, we utilize the AdamW optimizer with a learning rate of 1×10^{-6} .

Hyperparameter	MMLU-Pro	SuperGPQA & TheoremQA	Math (AIME/AMC)	Countdown	Noisy Rewards (Distinguishability)
<i>Model Configuration</i>					
Base Model	Qwen2.5-1.5B-Math-Instruct	Qwen3-8B-Base	Qwen3-8B-Base	Qwen2.5-1.5B-Math	Qwen3-8B-Base
<i>GRPO Training</i>					
Learning Rate	1×10^{-6}	1×10^{-6}	1×10^{-6}	1×10^{-6}	1×10^{-6}
Optimizer	AdamW	AdamW	AdamW	AdamW	AdamW
Training Problems per Step (n_{train})	32	128	128	128	128
Rollouts per Training Problem	128	128	128	128	128
<i>Data Selection</i>					
Selection Interval (U steps)	10	10	10	10	10
Rollouts for Estimation (k_v)	16	16	16	16	64
Selection Ratio (q)	4	4	4	20	4
Selected Percentage	25%	25%	25%	5%	25%

Table 5: **Detailed Hyperparameters across Experimental Settings.** Note that n_{train} denotes the number of distinct problems per GRPO update step. The selection ratio q determines the fraction of the candidate pool retained ($1/q$).

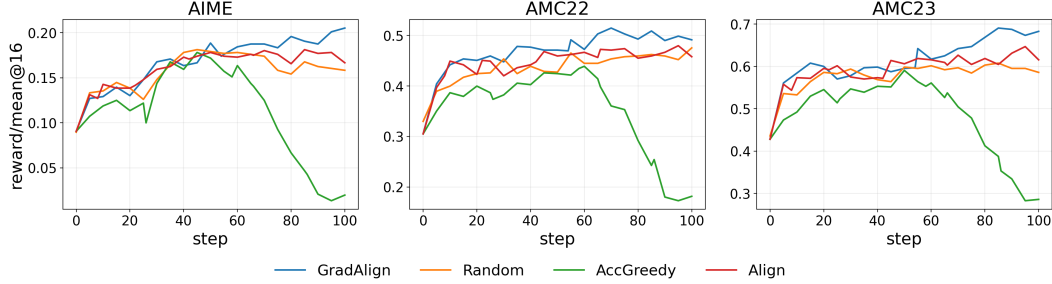


Figure 5: **Training Accuracy Curve on AIME2425, AMC22 and AMC23 (Scenario 1).** GradAlign (ours) achieves the strongest performance.

B Candidate Distribution

We visualize the score distribution for Scenario 1 (Noisy Rewards), in which half of the training candidates are corrupted by random reward signals sampled from a Bernoulli distribution with $p = 0.5$. An effective data selection method should be able to distinguish clean candidates from corrupted ones. As shown in Figure 6, we compare two similarity metrics (cosine similarity and inner product) for computing alignment scores. Cosine similarity provides clearer separation between clean and corrupted candidates across different training stages, demonstrating its superior discriminative capability.

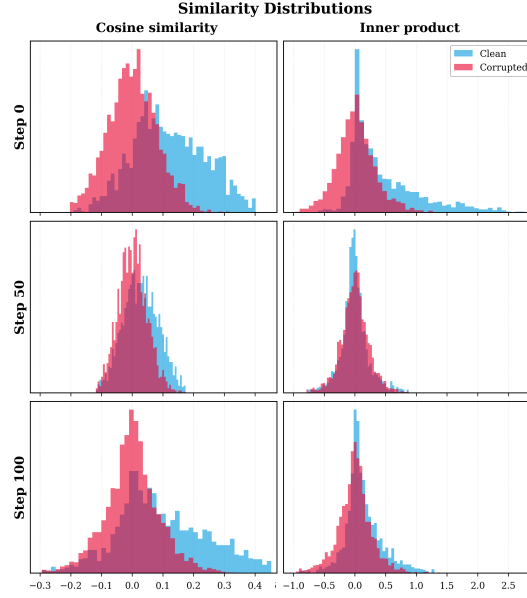


Figure 6: **Distribution of Cosine Similarity and Inner Product Similarity.** Cosine similarity is more indicative than the inner product regarding detecting corrupted instances.

C Proofs of Validation Improvement

C.1 Proof of Theorem 4.1

Proof. Let $J(\theta) = \mathbb{E}_{x \sim \mathcal{D}, y \sim \pi_\theta(\cdot|x)}[R(x, y)]$ be the expected reward, where $R(x, y) = \mathbf{1}[y = y^*(x)]$ is the judge outcome indicating whether y matches the reference answer $y^*(x)$. Hence, $J(\theta)$ equals the expected accuracy. Under on-policy sampling, the policy gradient theorem (Sutton et al., 1999) gives

$$\nabla_\theta J(\theta) = \mathbb{E}_{x, y \sim \pi_\theta}[Q^\pi(x, y) \nabla_\theta \log \pi_\theta(y|x)].$$

Here $Q^\pi(x, y)$ denotes the action-value function, i.e., the expected return obtained by taking action y under policy π given input x . Let $A^\pi(x, y) = Q^\pi(x, y) - b(x)$ denote the advantage function, where $b(x)$ is any baseline independent of y . Then $\mathbb{E}_{y \sim \pi_\theta}[b(x) \nabla_\theta \log \pi_\theta(y|x)] = 0$. Therefore

$$\nabla_\theta J(\theta) = \mathbb{E}_{x, y \sim \pi_\theta}[A^\pi(x, y) \nabla_\theta \log \pi_\theta(y|x)].$$

If the advantage estimator $\hat{A}$ is unbiased, i.e., $\mathbb{E}[\hat{A} | x, y] = A^\pi(x, y)$, then

$$\mathbb{E}_{x, y \sim \pi_\theta}[\hat{A} \nabla_\theta \log \pi_\theta(y|x)] = \nabla_\theta J(\theta),$$

showing that the resulting update is an unbiased estimator of $\nabla_\theta J(\theta)$, which equals the gradient of expected accuracy.

We note that GRPO does not use such an unbiased advantage: it centers rewards by the group mean $\bar{r} = \frac{1}{k} \sum_{\ell=1}^k r_\ell$, which includes the sampled reward and is therefore not a baseline independent of the action. Writing $\bar{r}_{-j} = \frac{1}{k-1} \sum_{\ell \neq j} r_\ell$ for the leave-one-out mean, which is a valid action-independent baseline, one has $r_j - \bar{r} = (1 - \frac{1}{k})(r_j - \bar{r}_{-j})$. Hence group-mean centering (without standard-deviation normalization) equals the unbiased leave-one-out estimator scaled by $1 - \frac{1}{k}$: it is biased in magnitude but preserves the direction of the expected-accuracy gradient. $\square$

C.2 Proof of Theorem 4.2

Proof. Fix a problem x , and define its expected accuracy under the current policy as

$$p_\theta(x) = \mathbb{E}_{y \sim \pi_\theta(\cdot|x)}[R(x, y)],$$

where $R(x, y) \in \{0, 1\}$, and thus $p_\theta(x) \in [0, 1]$.

Let $y_1, \dots, y_k \stackrel{\text{i.i.d.}}{\sim} \pi_\theta(\cdot | x)$, $r_j = R(x, y_j)$, $p = p_\theta(x)$, and define the score vector $s_j = \nabla_\theta \log \pi_\theta(y_j | x)$.

As is standard in policy-gradient methods, the sampled rewards and normalization statistics are treated as stop-gradient quantities when differentiating the GRPO surrogate objective. We define the group mean $\bar{r} = \frac{1}{k} \sum_{j=1}^k r_j$ and, for any normalization constant $\epsilon \geq 0$, the normalized advantage $\hat{A}_j = \frac{r_j - \bar{r}}{\sqrt{\frac{1}{k} \sum_{\ell=1}^k (r_\ell - \bar{r})^2 + \epsilon}}$, with the convention that a zero-variance

group (in which all rewards are equal) contributes zero advantage, $\hat{A}_j = 0$; this covers the case $\epsilon = 0$, for which the denominator would otherwise vanish. The corresponding normalized GRPO gradient estimator per-problem is

$$\hat{g}_{\text{norm}}(x) = \frac{1}{k} \sum_{j=1}^k \hat{A}_j s_j.$$

We first establish two standard score-function identities:

$$\mathbb{E}[s_j] = \mathbb{E}_{y \sim \pi_\theta(\cdot | x)} [\nabla_\theta \log \pi_\theta(y | x)] = 0,$$

$$\mathbb{E}[r_j s_j] = \mathbb{E}_{y \sim \pi_\theta(\cdot | x)} [R(x, y) \nabla_\theta \log \pi_\theta(y | x)] = \nabla_\theta \mathbb{E}_{y \sim \pi_\theta(\cdot | x)} [R(x, y)] = \nabla_\theta p.$$

If $p \in \{0, 1\}$, all sampled rewards are equal, so $\hat{g}_{\text{norm}}(x) = 0$ and the claim holds with $c_\theta(x) = 0$. We therefore assume $p \in (0, 1)$ below. Because the reward is binary, we have $\mathbb{E}[r_j s_j] = p \mathbb{E}[s_j | r_j = 1]$. Therefore, $\mathbb{E}[s_j | r_j = 1] = \frac{\nabla_\theta p}{p}$. Similarly, for $\mathbb{E}[s_j | r_j = 0]$, we have

$$(1 - p) \mathbb{E}[s_j | r_j = 0] = \mathbb{E}[(1 - r_j) s_j] = \mathbb{E}[s_j] - \mathbb{E}[r_j s_j] = -\nabla_\theta p,$$

$$\text{so } \mathbb{E}[s_j | r_j = 0] = -\frac{\nabla_\theta p}{1 - p}.$$

Let $n = \sum_{i=1}^k r_i = \bar{r}k$ denote the number of correct rollouts in this realized group; the group variance is

$$\frac{1}{k} \sum_{\ell=1}^k (r_\ell - \bar{r})^2 = \frac{n}{k} \left(1 - \frac{n}{k}\right)^2 + \frac{k - n}{k} \left(\frac{n}{k}\right)^2 = \frac{n}{k} \left(1 - \frac{n}{k}\right).$$

Define $d_n = \sqrt{\frac{n}{k} \left(1 - \frac{n}{k}\right)} + \epsilon$. For $n \in \{1, \dots, k - 1\}$ this is strictly positive; the boundary cases $n \in \{0, k\}$ are zero-variance groups that contribute zero by the convention above (their numerator $n(k - n)$ also vanishes), so the value of d_n there is immaterial. Condition on an arbitrary reward vector $(r_1, \dots, r_k)$ containing exactly n correct rollouts. Since the rollouts are independent, the conditional expectation of each score vector depends only on its corresponding reward. Therefore,

$$\begin{aligned} \mathbb{E} \left[\sum_{j=1}^k (r_j - \bar{r}) s_j \mid r_1, \dots, r_k \right] &= n \left(1 - \frac{n}{k}\right) \mathbb{E}[s_j | r_j = 1] + (k - n) \left(-\frac{n}{k}\right) \mathbb{E}[s_j | r_j = 0] \\ &= n \left(1 - \frac{n}{k}\right) \frac{\nabla_\theta p}{p} + (k - n) \left(-\frac{n}{k}\right) \left(-\frac{\nabla_\theta p}{1 - p}\right) \\ &= \frac{n(k - n)}{k} \left(\frac{1}{p} + \frac{1}{1 - p}\right) \nabla_\theta p \\ &= \frac{n(k - n)}{kp(1 - p)} \nabla_\theta p. \end{aligned}$$

This expression depends on the reward vector only through n . Let N be the number of correct rollouts. Conditional on $N = n$, the expected normalized gradient is

$$\mathbb{E} [\hat{g}_{\text{norm}}(x) \mid N = n, x] = \frac{1}{kd_n} \mathbb{E} \left[\sum_{j=1}^k (r_j - \bar{r}) s_j \mid N = n, x \right] = \frac{n(k - n)}{k^2 p(1 - p) d_n} \nabla_\theta p.$$

Taking expectation over $N \sim \text{Binomial}(k, p)$ gives

$$\mathbb{E} [\hat{g}_{\text{norm}}(x) \mid x] = \sum_{n=0}^k \Pr(N = n) \frac{n(k-n)}{k^2 p(1-p) d_n} \nabla_{\theta} p = c_{\theta}(x) \nabla_{\theta} p_{\theta}(x),$$

where

$$c_{\theta}(x) = \sum_{n=0}^k \binom{k}{n} p^n (1-p)^{k-n} \frac{n(k-n)}{k^2 p(1-p) d_n}.$$

Here the boundary terms $n \in \{0, k\}$ are zero (their numerator $n(k-n)$ vanishes, and for $\epsilon = 0$ they are zero by the zero-variance convention above). All the remaining terms are non-negative, and when $p \neq 0, 1$ and $n \neq 0, k$ they are strictly positive. Therefore, $c_{\theta}(x)$ is positive when $p \in (0, 1)$ and $k \geq 2$.

It follows that

$$\mathbb{E} [\hat{g}_{\text{norm}}(x) \mid x] = c_{\theta}(x) \nabla_{\theta} p_{\theta}(x), \quad c_{\theta}(x) \geq 0,$$

where $c_{\theta}(x) > 0$ whenever $p_{\theta}(x) \in (0, 1)$ and $k \geq 2$. $\square$

C.3 Counterexample for Non-Binary Rewards

The binary-reward assumption in Theorem 4.2 is essential. We construct a counterexample showing that, for general scalar rewards, group-wise standard-deviation normalization does not necessarily preserve the expected policy-gradient direction.

Consider a softmax policy over three actions parameterized by logits $\theta \in \mathbb{R}^3$,

$$\pi_{\theta}(i) = \frac{\exp(\theta_i)}{\sum_{j=1}^3 \exp(\theta_j)},$$

evaluated at the uniform policy

$$\pi_{\theta}(1) = \pi_{\theta}(2) = \pi_{\theta}(3) = \frac{1}{3}.$$

Let

$$s_i = \nabla_{\theta} \log \pi_{\theta}(i)$$

denote the score function associated with action i . Under the softmax parameterization,

$$s_i = e_i - \pi_{\theta},$$

where e_i is the i -th standard basis vector. Assign deterministic rewards

$$R(1) = 0, \quad R(2) = 1, \quad R(3) = 3.$$

The expected reward is

$$J(\theta) = \mathbb{E}_{i \sim \pi_{\theta}} [R(i)],$$

whose policy gradient is

$$\nabla_{\theta} J(\theta) = \sum_{i=1}^3 \pi_{\theta}(i) R(i) s_i = \frac{1}{3} s_2 + s_3 = \frac{1}{9} \begin{pmatrix} -4 \\ -1 \\ 5 \end{pmatrix}.$$

Now consider group-wise advantage normalization with $k = 2$ independent rollouts in the limit $\epsilon \rightarrow 0^+$. Zero-variance groups have zero gradient because their rewards are equal. For two actions with distinct rewards,

$$\hat{g}_{\text{norm}}(i, j) \longrightarrow \frac{1}{2} \text{sign}(R(i) - R(j))(s_i - s_j).$$

The key here is that standard-deviation normalization removes the magnitude of each pairwise reward difference and retains only its sign. At the uniform policy, each ordered

action pair is sampled with probability $1/9$. Combining the two orderings of each distinct pair gives

$$\begin{aligned}\mathbb{E}[\hat{g}_{\text{norm}}] &= \frac{1}{9} [-(s_1 - s_2) - (s_2 - s_3) - (s_1 - s_3)] \\ &= \frac{1}{9} \begin{pmatrix} -2 \\ 0 \\ 2 \end{pmatrix},\end{aligned}$$

where we use

$$s_i - s_j = e_i - e_j.$$

Since

$$\frac{1}{9} \begin{pmatrix} -2 \\ 0 \\ 2 \end{pmatrix} \not\parallel \frac{1}{9} \begin{pmatrix} -4 \\ -1 \\ 5 \end{pmatrix},$$

the expected normalized policy gradient is not a scalar multiple of $\nabla_{\theta} J(\theta)$. Therefore, group-wise standard-deviation normalization does not, in general, preserve the expected policy-gradient direction for non-binary rewards. By continuity, the same conclusion holds for all sufficiently small $\epsilon > 0$.

D Applicable Scenarios

GradAlign is particularly well-suited for settings where the training rewards are noisy or unreliable, as the gradients are more dispersed. We highlight three common scenarios, with illustrations in Figure 2.

Unreliable Reward Signals In many RL settings, the reward signal can be noisy or unreliable, providing an imperfect proxy for true task performance. For example, in LLM RL, rewards are often obtained from model-based judges, and such rewards can be inaccurate on ambiguous problems or inputs beyond the judge’s capabilities. These cases often exhibit intermediate accuracy and are therefore difficult to remove with accuracy-based filtering, which can waste compute and destabilize training. GradAlign is well-suited to this setting: gradients induced by misjudged problems tend to be weakly aligned with the validation gradient (often near-orthogonal in high-dimensional space), so GradAlign naturally downweights them. In Section 6, we show that GradAlign improves the stability of the training and the final performance under unreliable reward signals.

Distribution Imbalance When the training dataset contains a wide range of problems (e.g., math problems containing different categories (Albalak et al., 2025)) and the target downstream task is more specific (e.g., one specific puzzle game), many training problems may be correct and of moderate difficulty, but uninformative for the target task, causing accuracy-based methods to fail at filtering such problems. GradAlign addresses this mismatch by favoring problems whose gradients align with the validation gradient, while downweighting target-irrelevant ones with low alignment.

Low-Utility Training Data As RL training scales up, practitioners increasingly rely on large web-mined datasets (Yue et al., 2024; Du et al., 2025). Unlike distribution imbalance, where useful data is underrepresented, the challenge here is that many training problems are on-distribution but overly simple, yielding valid rewards while providing limited learning signal for the target task. GradAlign addresses this setting by prioritizing problems whose gradients align with validation performance and downweighting low-utility samples.

E Measuring Gradient Noise

We conduct an ablation study to determine the number of response samples required for a reliable estimation of the cosine similarity between policy gradients. Specifically, we estimate the gradient cosine similarity using $k_v \in \{32, 128, 512\}$ samples per problem,

computing each estimate twice with different random seeds. We then calculate the Pearson correlation between similarity scores obtained from these independent runs using the DAPO training set and the AMC22 validation set.

Table 6 shows that the correlation increases with sample size, indicating that larger k_v produces more stable and accurate estimates. $k_v = 512$ achieves the most accurate estimation with consistent similarity scores but requires high computational cost. $k_v = 32$ reduces the computational cost significantly but produces noisy estimates. Since GradAlign relies on relative ranking rather than exact gradient values, a moderate correlation (e.g., 0.49) already provides a reliable directional signal in high-dimensional gradient space.

Number of Samples k_v	32	128	512
Cosine Similarity	0.30	0.49	0.79

Table 6: **Pearson Correlation of Similarity Scores Calculated Using Responses Sampled Twice.** Pearson correlation between cosine similarity scores computed using two different random seeds, for varying numbers of samples per problem (k_v). Samples and gradients are produced from Qwen2.5-1.5B-Math-Instruct.

F Compute and Memory Analysis

We provide a detailed per-round FLOP breakdown and a memory analysis for the computational overhead summarized in Section 5.2.

FLOP accounting. We estimate per-round FLOPs following Kaplan et al. (2020), where a forward pass costs $\approx 2N$ FLOPs per token and a forward-backward pass $\approx 6N$, with N the model size and T the sequence length. In our standard setting ($|\mathcal{P}_r| = 5120$, $|\mathcal{S}_r| = 1280$, $k_r = k_v = 16$, $n_t = 128$, $|\mathcal{P}_v| = 200$), the per-round cost decomposes as:

- **Candidate rollouts:** $|\mathcal{P}_r| k_r \cdot 2NT \approx 2.6 \times 10^{18}$ FLOPs;
- **Candidate gradients:** $|\mathcal{P}_r| k_r \cdot 6NT \approx 7.9 \times 10^{18}$ FLOPs;
- **Validation gradient:** $|\mathcal{P}_v| k_v \cdot 6NT \approx 3.1 \times 10^{17}$ FLOPs (negligible);
- **RL update:** $|\mathcal{S}_r| n_t \cdot 6NT \approx 1.6 \times 10^{19}$ FLOPs.

The three GradAlign terms (candidate rollouts, candidate gradients, and validation gradient) total $\approx 1.1 \times 10^{19}$ FLOPs against a $\approx 1.6 \times 10^{19}$ -FLOP update, matching the $\approx 65\%$ relative overhead reported in Section 5.2. Since N appears linearly in every term, it cancels in the ratio, so this relative overhead is **invariant to model scale**.

Memory. GradAlign introduces no new memory peak beyond a standard GRPO step. A GRPO update under AdamW must hold model weights, activations, and optimizer states, which together occupy roughly $16N$ at peak (Rajbhandari et al., 2020). During the selection step, GradAlign only stores the gradient of the current candidate and the aggregated validation gradient; combined with the model parameters, this is about $6N$, well below the training peak. Gradient alignment thus reuses the existing training footprint rather than adding to it.