G^{2}RPO-A: Guided Group Relative Policy Optimization with Adaptive Guidance

Anonymous ACL submission

Abstract

Reinforcement Learning with Verifiable Rewards (RLVR) has markedly enhanced the reasoning abilities of large language models (LLMs). Its success, however, largely depends on strong base models with rich world knowledge, yielding only modest improvements for small-size language models (SLMs). To address this limitation, we investigate Guided GRPO, which injects ground-truth reasoning steps into roll-out trajectories to compensate for SLMs' inherent weaknesses. Through a comprehensive study of various guidance configurations, we find that naively adding guidance delivers limited gains. These insights motivate G^{2}RPO-A, an adaptive algorithm that automatically adjusts guidance strength in response to the model's evolving training dynamics. Experiments on mathematical reasoning and code-generation benchmarks confirm that G^{2}RPO-A substantially outperforms vanilla GRPO. Our code and models will be publicly available.

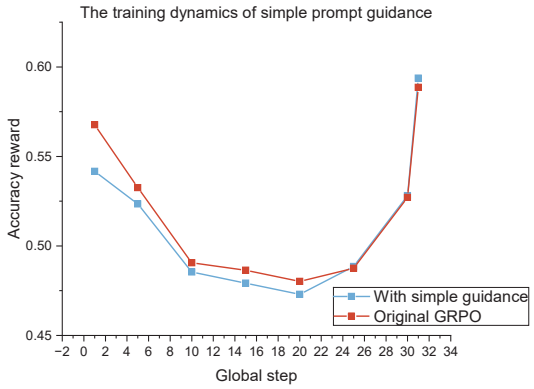


Figure 1: **Naive guidance does not help.** Using Qwen2.5-Math-7B as the base model, we train it on the s1K-1.1 dataset for a single epoch with a simple, fixed-length guidance (naive guidance). The naive guidance method shows a temporary increase in the accuracy reward during the early training stages, but it quickly becomes indistinguishable from the vanilla GRPO curve.

1 Introduction

Recent advancements in reasoning-centric large language models (LLMs), exemplified by DeepSeek-R1 (Guo et al., 2025), OpenAI-o1 (Jaech et al., 2024), and Qwen3 (Yang et al., 2025a), have significantly expanded the performance boundaries of LLMs, showcasing the immense potential of reasoning-enhanced models. Building upon robust base models with comprehensive world knowledge, these reasoning-focused LLMs have achieved breakthrough progress in complex domains such as mathematics (Guan et al., 2025), coding (Souza et al., 2025; HUANG et al., 2025), and other grounding tasks (Li et al., 2025b; Wei et al., 2025). At the core of this success lies Reinforcement Learning with Verifiable Rewards (RLVR) (Shao et al., 2024; Chu et al., 2025; Liu et al., 2025). This innovative approach, which employs reinforcement learning techniques in LLMs using rule-based outcome rewards, has garnered significant attention in the AI community. RLVR has demonstrated remarkable improvement in generalization across a wide spectrum of downstream tasks (Jia et al., 2025; Wu et al., 2025), positioning it as a pivotal advancement in the field of artificial intelligence.

As the de-facto algorithm, Group Relative Policy Optimization (GRPO) (Shao et al., 2024) improves upon Proximal Policy Optimization (PPO) (Schulman et al., 2017) by removing the need for a critic model through inner-group response comparison, thereby speeding up the training.

Capacity of small-size LLMs limit the performance gains of GRPO. Despite GRPO's success with large-scale LLMs, its effectiveness is significantly constrained when applied to smaller LLMs. Recent research (Ye et al., 2025; Muennighoff et al., 2025a) reveals that GRPO's performance gains highly depend on the base model's capacity (Bae et al., 2025; Xu et al., 2025; Zhuang et al., 2025).

Consequently, small-scale LLMs (SLMs) show limited improvement under GRPO (Table 3, 9), exposing a critical scalability challenge in enhancing reasoning capabilities across diverse model sizes. To address this challenge, researchers have explored various approaches: distillation (Guo et al., 2025), multi-stage training (Xu et al., 2025) prior to RLVR, and selective sample filtering (Xiong et al., 2025; Shi et al., 2025). However, these methods precede RLVR or suffer performance degradation in complex problems (Table 8). Consequently, optimizing the RLVR process for efficient learning in SLMs remains an open challenge, representing a critical frontier in AI research.

Adaptive guidance as a solution. We propose incorporating guidance into the roll-out process to facilitate the generation of high-quality, reward-worthy candidates (Figure 2). However, our initial findings revealed that the implementation of simple fixed-length guidance to the prompts (naive guidance) failed to improve overall performance (Figure 1). Through a comprehensive analysis of the guidance mechanism—varying both the proportion of guided roll-outs within GRPO batches and the guidance length over training epochs—we obtained two key findings: (1) Code-generation tasks benefit from a higher guidance ratio than mathematical reasoning tasks, and smaller models likewise require more guidance than larger ones. (2) The optimal guidance length evolves throughout training and is highly context-dependent, rendering simple, predefined schedules ineffective. In response, we introduce the Guided Group Relative Policy Optimization with Adaptive Guidance (G^2RPO-A) algorithm. This innovative approach dynamically adjusts guidance length based on the model's real-time learning state, offering a sophisticated solution to the challenges of enhancing small-size LLMs' performance in RLVR processes. The key contributions of this paper are summarized as follows:

- We enhance GRPO for small-scale LLMs by injecting guidance into the rollout thinking trajectories. Our systematic analysis of key guidance configurations reveals that the optimal strategy is highly context-dependent, while the ideal guidance length evolves dynamically throughout training, rendering simple, predefined schedules ineffective.

- Drawing on the preceding findings, we introduce G^2RPO-A, an adaptive algorithm that automatically adjusts guidance length in response to the evolving training state.

- We evaluate our method on mathematical reasoning and coding tasks with several models–including the Qwen3 series, DeepSeek-Math-7B-Base, and DeepSeek-Coder-6.7B-Base–and observe substantial performance gains over a range of baselines, including vanilla GRPO, SFT, and simple guided approaches.

2 Related Works

The introduction of chain-of-thought (CoT) prompting has markedly improved LLM performance on complex reasoning tasks (Wei et al., 2022; Kojima et al., 2022). Complementing this advance, Reinforcement Learning with Verifiable Rewards (RLVR) has emerged as a powerful training paradigm for reasoning-centric language models (Yue et al., 2025; Lee et al., 2024). The defacto RLVR algorithm, Group Relative Policy Optimization (GRPO) (Guo et al., 2025) delivers strong gains on various benchmarks while remaining training-efficient because it dispenses with the need for a separate critic network (Wen et al., 2025; Shao et al., 2024). Recent efforts to improve GRPO have explored several directions. Some approaches focus on refining the core GRPO objective, either by pruning candidate completions (Lin et al., 2025) or by removing normalization biases (Liu et al., 2025). Separately, another studies aim to enhance the training signal and stability. DAPO (Yu et al., 2025), for instance, introduces dense, step-wise advantage signals and decouples the actor-critic training to mitigate reward sparsity.

However, adapting GRPO–style algorithms to small-scale LLMs remains challenging due to the sparse-reward problem (Lu et al., 2024; Nguyen et al., 2024; Dang and Ngo, 2025). Recent studies have therefore focused on improved reward estimation (Cui et al., 2025). TreeRPO (Yang et al., 2025b) uses a tree-structured sampling procedure to approximate step-wise expected rewards, and Hint-GRPO (Huang et al., 2025) applies several reward-shaping techniques. Other lines of research investigate knowledge distillation (Guo et al., 2025), multi-stage pre-training before RLVR (Xu et al., 2025), and selective sample filtering (Xiong et al., 2025; Shi et al., 2025). In our experiments, however, these filtering or sampling strategies are performed either only before RLVR or do not improve performance on more complex

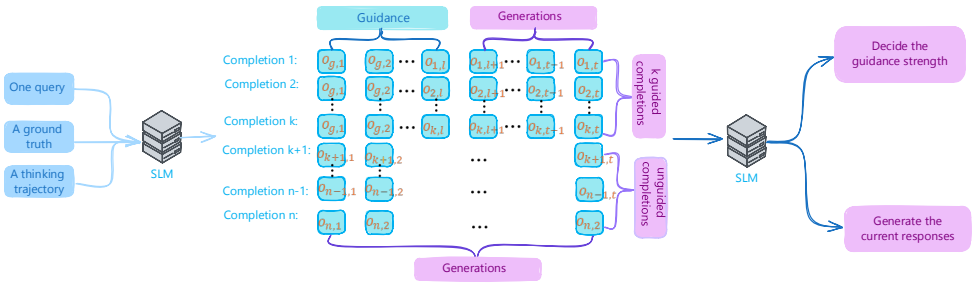


Figure 2: **Overview of G^{2}RPO-A.** Each step we split roll-outs into a guided set and an unguided set. We then compare the current rewards with those from the previous steps; the resulting ratio determines the future guidance length.

tasks. In this paper, we introduce a guidance mechanism that injects ground-truth reasoning steps directly into the model's roll-out trajectories during RL training. Because the guidance is provided online, the proposed method can still learn effectively from difficult examples while mitigating the sparse-reward issue.

The role of guidance in GRPO-style training remains underexplored. Two concurrent studies have addressed related questions (Nath et al., 2025; Park et al., 2025), but both simply append guidance tokens to the input prompt, offering neither a systematic analysis of guidance configurations nor a mechanism that adapts to the changing training state of the model. We show that naive guidance often fails to improve performance because it yields low expected advantage. To remedy this, we provide a comprehensive examination of how guidance length, ratio, and scheduling affect learning, and we introduce G^{2}RPO-A, an adaptive strategy that dynamically adjusts guidance strength throughout training.

3 Preliminary

Group Relative Policy Optimization (GRPO). Given a prompt, GRPO (Shao et al., 2024) samples G completions and computes their rewards $\{r_i\}_{i=1}^{G}$. Define the tth token of the ith completion as $o_{i,t}$. GRPO then assigns a advantage, $\hat{A}_{i,t}$ to it. The optimization objective is defined as:

$$\mathcal{L}_{\text{GRPO}}(\theta) = -\hat{\mathbb{E}}_t\Big[\min\Big(w_t(\theta)\hat{A}_t, \text{clip}\big(w_t(\theta), 1-\epsilon, 1+\epsilon\big)\hat{A}_t\Big) - \beta\mathcal{D}_{\text{KL}}\big(\pi_\theta(\cdot|s_t)\|\pi_{\text{ref}}(\cdot|s_t)\big)\Big] \quad (1)$$

where the importance weight $w_{i,t}$ is given by

$$w_{i,t} = \frac{\pi_\theta(o_{i,t} \mid q, o_{i,<t})}{[\pi_{\theta_{\text{old}}}(o_{i,t} \mid q, o_{i,<t})]}. \quad (2)$$

The clipping threshold ϵ controls update magnitude, $[\cdot]_{\text{old}}$ indicates policy at last step, β is the influence of KL divergence $\mathcal{D}_{\text{KL}}$, whose detailed definition can be found in the section **Detailed equations** in appendix A.2.1.

Limitations of GRPO for small-size LLMs. Despite the success of GRPO in large language models (LLMs), small-size LLMs (SLMs) face significant challenges when confronted with complex problems requiring long chains of thought (Zhang et al., 2025). Due to their inherently limited capacity, SLMs struggle to generate high-quality, reward-worthy candidates for such tasks (Li et al., 2025a; Zheng et al., 2025). As shown in Figure 3a, where Qwen3-1.7B is implemented on a code task and it fails to generate correct answers for most queries. This limitation substantially reduces the probability of sampling high-reward candidates, resulting in advantage signals vanishing (Figure 4b), thereby constraining the potential performance gains achievable through GRPO in SLMs.

4 Methodology

To address the limitations of GRPO on SLMs, we propose a guidance mechanism that **injects partial, high-quality thinking trajectories—such as those generated by more capable LLMs or derived from ground-truth reasoning steps—into the model's rollout process.** This approach steers SLMs toward generating superior completions, thereby facilitating the sampling of high-quality candidates. We then conduct a comprehensive investigation into various design choices for guidance strategies. Finally, we introduce the G^{2}RPO-A algorithm, which integrates our empirical observa-

MATH500	$\alpha = \frac{5}{6}$	$\alpha = \frac{3}{6}$	$\alpha = \frac{1}{6}$	$\alpha = 1$
$\ell = 50$	66.80	66.00	67.20	65.10
$\ell = 100$	65.20	63.00	66.20	64.70
$\ell = 200$	57.60	52.40	62.00	59.30
$\ell = 500$	57.80	62.00	68.20	55.80
$\ell = 0$	62.00			

Table 1: **Empirical study on guidance length ℓ and guidance ratio α.** We use the Qwen2.5-Math-7B as the backbone.

tions and significantly reduces the need for extensive hyperparameter tuning.

4.1 Guided GRPO as a Solution

The Guided GRPO can be formulated as:

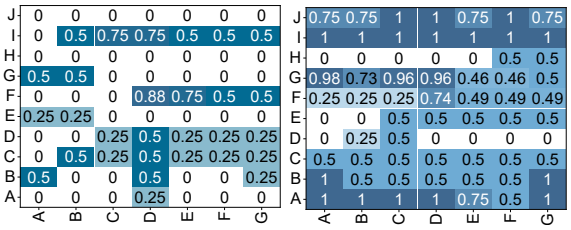
$$\mathcal{L}_{\text{guided}}(\theta) = \mathbb{E}_{(q,a)\sim\mathcal{D},\{g_i\}_{i=1}^{G}\sim\mathcal{G},\{o_i\}_{i=1}^{G}\sim\pi_{\text{ref}}(\cdot|q,g_i)} \Bigg[\frac{1}{G}\sum_{i=1}^{G}\frac{1}{|o_i|+|g_i|}\Bigg(\sum_{t=1}^{|g_i|}\min\big(w_{g,i,t},\text{clip}\big)\,\hat{A}_{i,t} + \sum_{t=1}^{|o_i|}\min\big(w_{o,i,t},\text{clip}\big)\hat{A}_{i,t} - \beta\mathcal{D}_{\text{KL}}(\pi_\theta\|\pi_{\text{ref}})\Bigg)\Bigg]. \quad (3)$$

where $w_{g,i,t}$ and $w_{o,i,t}$ denote the token-level weighting coefficient of guidance g_i and the model outputs o_i, respectively. Note that clip here refers to $\text{clip}(w_{g,i,t}, 1-\epsilon, 1+\epsilon)$ and $\text{clip}(w_{o,i,t}, 1-\epsilon, 1+\epsilon)$. As shown in Figure 3b, this guidance enables SLMs to generate higher-reward candidates, potentially overcoming their inherent limitations.

Naive Guided GRPO fails to boost the final performance. Despite increasing expected rewards (Figure 3b), we found that

simply adding guidance to thinking trajectories of all candidates doesn't enhance final performance and suffers from low advantage.

As shown in Figure 4a and 4b, we train Qwen-3-1.7B-Base on a math dataset sourced from math 220k dataset (Wang et al., 2024), and find that: (1) Guided GRPO's accuracy reward curve almost matches original GRPO. (2) Guided GRPO suffers from low advantage standard deviation, hindering the optimization of the models. As a result, further investigation is needed to leverage Guided GRPO's higher rewards while ensuring effective training, as the naive approach fails to utilize its potential benefits.



(a) GRPO (b) Guided GRPO

Figure 3: **Reward of Guided GRPO.** We fine-tuned Qwen3-1.7B on coding tasks, using 10 roll-outs and generating 280 candidates per batch. The candidates' rewards form a 20x14 matrix. We then applied 2x2 average pooling, reducing it to a 10x7 matrix for clearer visualization. The results demonstrate that when configured with an optimal guidance ratio, G^2RPO-A enables the model to sample candidates that yield a significantly denser reward signal.

4.2 Optimizing Guided GRPO Design

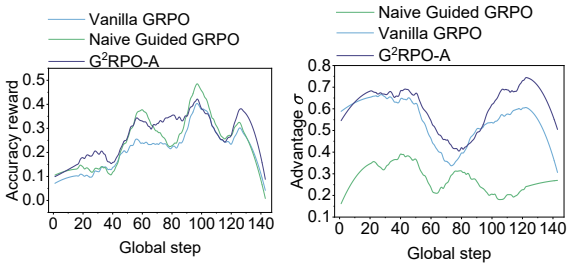
In this section, we thoroughly examine optimal design choices for Guided GRPO, focusing on guidance ratio of GRPO candidate groups and adjusting guidance strength at different training stages. These investigations aim to maximize the effectiveness of the Guided GRPO and overcome the limitations observed in the naive implementation.

Inner-Group Varied Guidance Ratio. The insufficiency of naive guidance suggests that a more nuanced approach is required. We begin by investigating the impact of the guidance ratio α. In each GRPO group of size G, we steer only an α-fraction of the candidates. Let g_i denote the guidance for the i-th candidate (ordered arbitrarily), we have:

$$|g_i| = 0 \quad (i > \alpha G), \qquad |g_i| = l \quad (i \le \alpha G). \quad (4)$$

That is, the first αG candidates have guidance, while the remaining $(1-\alpha)G$ candidates evolve freely. We conduct experiments on the Qwen2.5-Math-7B model (Yang et al., 2024) with a roll-out number $n = 6$, training for one epoch on the s1k-1.1 dataset (Muennighoff et al., 2025b). We set $\alpha \in \{1/6, \ldots, 1\}$ and $l \in \{50, 100, \ldots, 500\}$ tokens, with all accuracies reported on the Math 500 benchmark. The results in Table 1 show that:

- *Partial inner-group guidance improves model performance.* In most settings, Guided GRPO with guidance provided to only a subset of candidates outperforms the vanilla GRPO, confirming the usefulness of the guidance mechanism.



(a) Accuracy Reward (b) Advantage σ

Figure 4: **Pitfalls of naive Guided GRPO.** We trained Qwen3-1.7B-Base on a curriculum-ordered subset of Math-220K (Wang et al., 2024): problems are presented from easy to hard. Because the curriculum continually increases task difficulty, the accuracy reward does not plateau at a high level–an expected outcome of the CL schedule. This training dynamic indicates that the advantage standard deviation is extremely low under the naive guidance condition, a situation that negatively impacts training efficiency for SLMs.

Guidance length	Decay Policies		
Guidance ratio	Step	Linear	Concave
$\ell = 50$, $\alpha = 0.8333$	63.80	58.40	57.60
$\ell = 100$, $\alpha = 0.1667$	60.60	62.00	66.20
$\ell = 200$, $\alpha = 0.8333$	66.60	54.20	58.40
$\ell = 200$, $\alpha = 0.1667$	61.20	64.20	59.60
$\ell = 500$, $\alpha = 0.1667$	59.60	69.80	62.40
$\ell = 0$	62.00		

Table 2: **Performance of Guided GRPO under different guidance-length adjustment policies.** We train Qwen2.5-Math-7B and evaluate it on the MATH 500 benchmark. For each guidance-length schedule, we report the results obtained with the guidance ratio that achieves the highest score in Table 1.

- *For Qwen2.5-Math-7B on the Math500 benchmark, the lowest guidance ratio α combined with a longest guidance window ℓ yields the best results.* This suggests that Qwen2.5-Math-7B benefits from infrequent but heavy-weight guidance.

In summary, selective guidance–directing only few candidates by a long guidance–strikes the best balance between exploration and control, thereby improving model performance. Moreover, the optimal guidance ratio varies with both the task domain and model capacity. As Table 9, 10 shows, smaller models and coding tasks benefit from stronger intervention, whereas larger models and math tasks achieve better results with lighter guidance.

Time Varied Guidance Length. Apart from the guidance ratio, Table 1 shows that performance also depends on the guidance length ℓ. To investigate this further, we evaluate guided GRPO by varying the guidance length during training under three strategies. Those are:

$$\begin{aligned} \text{Concave decay:} \quad & \ell_t = \ell_0 \left(1 - \tfrac{t}{T}\right)^{\beta} \\ \text{Linear decay:} \quad & \ell_t = \ell_0 \left(1 - \tfrac{t}{T}\right) \\ \text{Stepwise decay:} \quad & \ell_t = \ell_0 \, \gamma^{\lfloor t/s \rfloor} , \end{aligned} \tag{5}$$

where T is the total training steps, and ℓ_0 is the initial guidance length. The parameter $\beta \in (1, \infty]$ controls the concavity, and $\gamma \in (0, 1)$ sets the decay rate, and s specifies the decay interval.

We use the same experiment setting as in Table 1, and choose the guidance ratio that performs the best. The results are reported in Table 2. The results indicate that (1) model quality is highly sensitive to the chosen guidance length ℓ_t, and (2) no single schedule consistently outperforms the others. *This highlights the need for more effective methods of controlling guidance length.*

4.3 G²RPO-A: Sampling Difficulty Motivated Adaptive Guidance

In this section, we propose an adaptive algorithm that automatically selects the guidance strength ℓ at every optimization step. Our approach is inspired by recent work on data filtering and sampling (Bae et al., 2025; Xiong et al., 2025; Shi et al., 2025) , which removes examples that yield uniformly low or uniformly high rewards. Such "uninformative" samples–being either too easy or too hard–contribute little to learning and can even destabilize training. The pseudo-code can be found in appendix A.1.

Guidance length adjustment. Our key idea is to control the difficulty of training samples by dynamically adjusting the guidance length, taking into account the ongoing training states. At each training step k, the guidance ℓ_{k+1} is determined by the following equation:

$$\ell_{k+1} = \ell_k \cdot \frac{\min(\mathcal{T}, k) r_k}{\sum_{\tau=1}^{\min(\mathcal{T},k)} r_{k-\tau}}, \tag{6}$$

where r_k is the average reward of the k-th training step, $\mathcal{T}$ is a hyperparameter that controls the number of history steps we considered, and we found that setting $\mathcal{T} = 2$ is already sufficient for noticeably improving Guided GRPO performance (Table 11, 12).

Equation 6 implies the following dynamics:

Base Model	α	Benchmark	Base	GRPO	SFT	G^{2}RPO-A
Qwen3-0.6B-Base	0.75	MATH500	40.18	**54.26**	50.53	51.77
		Minerva	11.43	9.57	10.40	**12.29**
		gpqa	25.49	24.51	25.49	**30.39**
Qwen3-1.7B-Base	0.25	MATH500	50.96	63.74	62.11	**67.21**
		Minerva	13.84	16.19	**18.89**	15.10
		gpqa	27.45	29.41	24.51	**32.35**
Qwen3-8B-Base	0.14	MATH500	71.32	79.49	80.29	**82.08**
		Minerva	33.24	**37.51**	36.60	36.42
		gpqa	43.17	44.13	42.85	**49.72**

Table 3: **Performance of G^{2}RPO-A on Math Tasks.** We report accuracy (%) on various benchmarks. Models are trained for 5 epochs, and guidance ratios are selected based on the best settings obtained from Table 10.

Base Model	α	Benchmark	Base	GRPO	G^{2}RPO-A
Qwen3-0.6B	0.75	MATH500	76.20	85.37	**87.15**
		Minerva	12.32	20.59	**21.57**
		gpqa	24.51	25.45	**26.43**
		AIME24	**10.00**	6.67	**10.00**
		AIME25	13.33	20.00	**23.33**
Qwen3-1.7B	0.25	MATH500	92.71	**94.52**	91.69
		Minerva	33.16	35.38	**38.26**
		gpqa	48.23	51.68	**55.27**
		AIME24	46.67	56.67	**63.33**
		AIME25	36.67	50.00	**53.33**

Table 4: **Performance of G^{2}RPO-A on Math Tasks.** The experiment settings are the same with Table 3. However, we use extra benchmarks like AIME24 and AIME25 here due to the stronger model performances.

	Random		CL	
	GRPO	G^{2}RPO-A	GRPO	G^{2}RPO-A
Qwen3-1.7-Base				
MATH500	53.81	57.67	52.05	58.94
Minerva	12.41	15.12	14.98	16.69
gpqa	24.79	23.53	27.45	25.49
Qwen3-0.6B-Base				
MATH500	43.25	50.72	48.16	53.59
Minerva	11.04	11.21	9.66	10.08
gpqa	23.1	25.49	24.51	32.35

Table 5: **Comparison of training with random order and curriculum learning (CL) order** across different models and benchmarks.

- When recent rewards rise, ℓ_k is reduced, making the next batch of examples harder.
- When recent rewards fall, ℓ_k is increased, making the next batch easier.

Thus, the training difficulty is automatically and continuously adjusted to match the model's current competence.

Curriculum learning for further improvements. Equation 6 shows that the adaptive guidance-length controller updates ℓ by comparing the current reward with rewards from previous steps. When consecutive batches differ markedly in difficulty, these reward variations no longer reflect the model's true learning progress, which in turn degrades G^{2}RPO-A 's performance.

To eliminate this mismatch, we embed a curriculum-learning (CL) strategy (Parashar et al., 2025; Shi et al., 2025; Zhou et al., 2025). Concretely, we sort the samples by difficulty. Using math task as an example, we rank examples by source, yielding five ascending difficulty tiers: cn_contest, aops_forum, amc_aime, olympiads, and olympiads_ref. We also tested ADARFT (Shi et al., 2025), which orders samples by success rate, but its buckets proved uninformative in our cases—most questions were either trivial or impossible (see appendix A.4.1 Figure 6)—so it failed to separate difficulty levels effectively. Table 5 shows that both the performance of vanilla GRPO and G^{2}RPO-A boosted by CL.

Compare G^{2}RPO-A to sample-filtering methods. Earlier work argues that policy-gradient training benefits most from mid-level queries. Bae et al. (2025) keep only moderate-difficulty batches via an online filter, and Reinforce-Rej (Xiong et al., 2025) discards both the easiest and hardest examples to preserve batch quality. Our experiments show that this exclusion is counter-productive: removing hard problems deprives the model of vital learning signals and lowers accuracy on challenging tasks. Table 8 confirms that either dropping hard items or substituting them with moderate ones reduces Level 4 and 5 test accuracy. G^{2}RPO-A avoids this pitfall by retaining tough examples and attaching adaptive guidance to them, thus exploit-

Base Model	Guidance Ratio	Benchmark	Base Perf.	GRPO	SFT	G^2RPO-A
Qwen3-0.6B	0.75	humaneval	32.32	38.89	40.33	**44.96**
		Live Code bench	17.07	22.22	13.58	**23.14**
Qwen3-1.7B	1	humaneval	46.08	67.65	63.34	**75.93**
		Live Code bench	34.31	53.14	**56.33**	51.96
Qwen3-8B	0.57	humaneval	64.36	**81.48**	77.42	80.33
		Live Code bench	60.58	77.12	63.82	**79.71**

Table 6: **Performance of G^2RPO-A on Code Tasks.** We report accuracy (%) on various benchmarks. Models are trained for 5 epochs, and guidance ratios are selected based on the best settings obtained from Table 9.

Qwen3-1.7B					
Benchmark	**Base Perf.**	**GRPO**	**CH**	**SFT**	**G^2RPO-A**
Code-Avg	40.20	60.40	60.19	59.84	**63.95**

Table 7: **Extra baseline comparison on Qwen3-1.7B on coding tasks.** The Code-Avg in table indicates taking the average of Live Code Bench and humaneval benchmark. CH means Clip-Higher algorithm.

Setting	Level 1	Level 2	Level 3	Level 4	Level 5
Remove	76.74	71.11	50.00	35.00	24.00
Replace	**88.37**	75.56	54.00	37.00	18.00
Original	86.05	**77.77**	**60.00**	**43.00**	**28.00**

Table 8: **Performance of GRPO with different sample-filtering methods.** We train Qwen3-1.7B model using G^2RPO-A, with $\alpha = 0.25$. In the REMOVE setting all hard samples are excluded from the original dataset, whereas in the REPLACE setting each hard sample is substituted with a sample of moderate difficulty.

	$\alpha = 0$	$\alpha = 0.25$	$\alpha = 0.5$	$\alpha = 0.75$	$\alpha = 1$
Qwen3-1.7B					
humaneval	68.52	59.88	64.81	72.22	70.81
LCB	28.43	19.61	23.53	30.39	35.72
Qwen3-0.6B					
humaneval	41.98	32.10	27.72	38.89	49.38
LCB	12.75	11.76	9.80	18.63	12.75

Table 9: **Ablation studies on guidance ratio α for Code Tasks.** The group size is set to 12. The initial guidance length for G^2RPO-A is set to 3072. The LCB indicates Live Code Bench.

	$\alpha = 0$	$\alpha = \frac{1}{4}$	$\alpha = \frac{2}{4}$	$\alpha = \frac{3}{4}$	$\alpha = 1$
Qwen3-1.7B-Base					
MATH500	52.05	58.71	53.09	55.53	45.95
Minerva	14.98	16.69	16.25	18.21	16.11
gpqa	27.45	25.49	30.39	25.49	22.55
Qwen3-0.6B-Base					
MATH500	48.16	49.59	50.94	53.50	38.42
Minerva	9.66	9.10	8.96	10.08	15.69
gpqa	24.51	19.61	31.37	32.35	25.49

Table 10: **Ablation studies on guidance ratio α for Math Tasks.** The group size is set to 12. The initial guidance length for G^2RPO-A is set to 3072.

ing the full difficulty spectrum without sacrificing performance.

Experiments

4.4 Experiment Settings

In this section, we outline the experiment settings we used, and more details about dataset filtering methods and evaluation on more models can be found in Appendix.

Datasets and models. We conduct experiments on math and code tasks. In detail,

- **Mathematical reasoning tasks.** We construct a clean subset of the Open-R1 math-220k corpus (Wang et al., 2024). Problems are kept only if their solution trajectories are (i) complete, (ii) factually correct, and (iii) syntactically parsable.
- **Code generation.** For programming experiments we adopt the Verifiable-Coding-Problems-Python benchmark from Open-R1. For every problem we automatically generate a chain-of-thought with QWQ-32B-preview (Team, 2024b). These traces are later used as guidance by our proposed G^2RPO-A training procedure.

We use Qwen3 series (Yang et al., 2025a) for both tasks. Results of DeepSeek-Math-7B-Base (Shao et al., 2024) for math and DeepSeek-Coder-6.7B-Base (Guo et al., 2024) for code also included in appendix A.5. Unless specifically mentioned, CL is used for all experiments for fair comparison, and

	GRPO	Fixed Guidance			RDP	G^{2}RPO-A
		3072	2048	1024		
Qwen3-1.7B-Base						
MATH500	52.05	51.28	60.52	46.78	51.02	**58.71**
Minerva	14.98	14.40	17.16	12.22	17.99	**22.46**
gpqa	**27.45**	25.00	24.51	23.53	22.13	25.49
Qwen3-0.6B-Base						
MATH500	48.16	55.80	54.17	52.69	**55.97**	53.50
Minerva	9.66	13.27	15.26	11.78	14.32	**15.69**
gpqa	24.51	24.00	21.57	22.55	26.00	**32.35**

Table 11: **Guidance-length ablation on Math Tasks.** Each run uses the optimal guidance ratio reported in Table 10. The initial guidance budget for G^{2}RPO-A is fixed at 3,072 tokens. RDP refers to the rule-based decay policy.

	GRPO	Fixed Guidance			RDP	G^{2}RPO-A
		3072	2048	1024		
Qwen3-1.7B						
humaneval	68.52	58.64	58.02	60.49	69.29	**70.81**
LCB	23.53	29.41	28.43	31.37	26.47	**35.72**
Qwen3-0.6B						
humaneval	38.89	43.93	36.54	38.40	42.27	**49.38**
LCB	12.75	**13.73**	10.78	9.80	11.67	12.75

Table 12: **Guidance-length ablation on Code Tasks.** Each run uses the optimal guidance ratio reported in Table 9. The initial guidance budget for G^{2}RPO-A is fixed at 3,072 tokens. RDP refers to the rule-based decay policy.

we also conducted ablation studies in Table 5.

Evaluation protocol. We assess our models mainly on three public mathematical–reasoning benchmarks—MATH500 (Hendrycks et al., 2021), MINERVA-MATH (Lewkowycz et al., 2022), and GPQA (Rein et al., 2024). For the mathematical training of Qwen3-1.7B and Qwen3-0.6B, AIME24 (Jia LI and Polu, 2024) and AIME25 benchmarks are also used. And for code tasks, we use two benchmarks: HUMANEVAL (Chen et al., 2021) and LIVE CODE BENCH (Jain et al., 2024). Decoding hyper-parameters are fixed to: temperature $= 0.6$, top-$p = 0.95$, and top-$k = 20$. Unless otherwise noted, we generate with a batch size of 128 and permit a token budget between 1,024 and 25,000, based on each model's context window.

Training details. Our G^{2}RPO-A algorithm is implemented on top of the fully open-source OPEN-R1 framework (Face, 2025). We use the following hyper-parameters: (i) number of roll-outs per sample set to 12 for 0.6B and 1.7B backbones, and 7 for 7B and 8B backbones; (ii) initial learning rate 1×10^{-6}, decayed with a cosine schedule and a warm-up ratio of 0.1; (iii) a training set of 1,000 problems for 5 epochs. Note that for ablation experiments, only 1 epoch is implemented in our training. (iv) All models are trained on 8 A100 GPUs.

4.5 Numerical Results

Superior performance of G^{2}RPO-A. As reported in Table 3, 4, 6 and 7, (i) our proposed G^{2}RPO-A markedly surpasses vanilla GRPO, SFT on nearly every benchmark. Furthermore, on coding tasks, our proposed G^{2}RPO-A outperforms the novel Clip-Higher (Yu et al., 2025) algorithm when using Qwen3-1.7B as the base model. (ii) all RL-based methods outperform both the frozen base checkpoints and their SFT variants, mirroring trends previously observed in the literature.

Effect of the guidance ratio α. Table 9, 10 show that (1) larger models benefit from weaker guidance—e.g., Qwen3-1.7B peaks at α=0.25/0.5 on Math, whereas the smaller Qwen3-0.6B prefers α=0.75; (2) Code tasks consistently require a higher guidance ratio than Math.

Ablation on guidance–length schedules. Table 11, 12 contrast our adaptive scheme (G^{2}RPO-A) with (i) fixed guidance and (ii) a rule-based decay policy (RDP). (1) G^{2}RPO-A achieves the best score on almost every model–benchmark pair, confirming the benefit of on-the-fly adjustment. (2) For fixed guidance, the optimal value varies across both tasks and model sizes, with no clear global pattern, underscoring the need for an adaptive mechanism such as G^{2}RPO-A.

5 Conclusion

We introduce a method that injects ground-truth guidance into the thinking trajectories produced during GRPO roll-outs, thereby improving the performance of small-scale LLMs. After an extensive study of guidance configurations, we observe that the guidance ratio is significant in guidance mechanism and the optimal guidance length is context-dependent and, based on this, we develop G^{2}RPO-A, an auto-tuned approach. Experiments on mathematical reasoning and code generation demonstrate that G^{2}RPO-A consistently boosts accuracy.

6 Limitations

The scope of this study presents two primary limitations. First, while the efficacy of G^{2}RPO-A has been validated on mathematical and code-generation tasks, its generalizability to other domains, such as multi-modal reasoning, remains to be investigated. Second, the selection of the hyper-parameter α currently relies on empirical tuning via ablation studies, as we have not yet identified a discernible principle governing its optimal value. Future work should be directed towards a more systematic investigation to establish a principled approach for its determination.

References

Sanghwan Bae, Jiwoo Hong, Min Young Lee, Hanbyul Kim, JeongYeon Nam, and Donghyun Kwak. 2025. Online difficulty filtering for reasoning oriented reinforcement learning. *Preprint*, arXiv:2504.03380.

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, and 39 others. 2021. Evaluating large language models trained on code. *Preprint*, arXiv:2107.03374.

Tianzhe Chu, Yuexiang Zhai, Jihan Yang, Shengbang Tong, Saining Xie, Sergey Levine, and Yi Ma. 2025. SFT memorizes, RL generalizes: A comparative study of foundation model post-training. In *The Second Conference on Parsimony and Learning (Recent Spotlight Track)*.

Ganqu Cui, Lifan Yuan, Zefan Wang, Hanbin Wang, Wendi Li, Bingxiang He, Yuchen Fan, Tianyu Yu, Qixin Xu, Weize Chen, and 1 others. 2025. Process reinforcement through implicit rewards. *CoRR*.

Quy-Anh Dang and Chris Ngo. 2025. Reinforcement learning for reasoning in small llms: What works and what doesn't. *Preprint*, arXiv:2503.16219.

E2B-Dev. 2024. e2b: Ai agent sandbox.

Hugging Face. 2025. Open r1: A fully open reproduction of deepseek-r1.

Xinyu Guan, Li Lyna Zhang, Yifei Liu, Ning Shang, Youran Sun, Yi Zhu, Fan Yang, and Mao Yang. 2025. rstar-math: Small LLMs can master math reasoning with self-evolved deep thinking. In *Forty-second International Conference on Machine Learning*.

Daya Guo, Dejian Yang, Haowei Zhang, Junxiao Song, Ruoyu Zhang, Runxin Xu, Qihao Zhu, Shirong Ma, Peiyi Wang, Xiao Bi, and 1 others. 2025. Deepseek-r1: Incentivizing reasoning capability in llms via reinforcement learning. *arXiv preprint arXiv:2501.12948*.

Daya Guo, Qihao Zhu, Dejian Yang, Zhenda Xie, Kai Dong, Wentao Zhang, Guanting Chen, Xiao Bi, Y Wu, YK Li, and 1 others. 2024. Deepseek-coder: When the large language model meets programming-the rise of code intelligence. *CoRR*.

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 *Thirty-fifth Conference on Neural Information Processing Systems Datasets and Benchmarks Track*.

Dong HUANG, Guangtao Zeng, Jianbo Dai, Meng Luo, Han Weng, Yuhao QING, Heming Cui, Zhijiang Guo, and Jie Zhang. 2025. Efficoder: Enhancing code generation in large language models through efficiency-aware fine-tuning. In *Forty-second International Conference on Machine Learning*.

Qihan Huang, Weilong Dai, Jinlong Liu, Wanggui He, Hao Jiang, Mingli Song, Jingyuan Chen, Chang Yao, and Jie Song. 2025. Boosting mllm reasoning with text-debiased hint-grpo. *Preprint*, arXiv:2503.23905.

Aaron Jaech, Adam Kalai, Adam Lerer, Adam Richardson, Ahmed El-Kishky, Aiden Low, Alec Helyar, Aleksander Madry, Alex Beutel, Alex Carney, and 1 others. 2024. Openai o1 system card. *CoRR*.

Naman Jain, King Han, Alex Gu, Wen-Ding Li, Fanjia Yan, Tianjun Zhang, Sida Wang, Armando Solar-Lezama, Koushik Sen, and Ion Stoica. 2024. Livecodebench: Holistic and contamination free evaluation of large language models for code. *CoRR*.

Ruipeng Jia, Yunyi Yang, Yongbo Gai, Kai Luo, Shihao Huang, Jianhe Lin, Xiaoxi Jiang, and Guanjun Jiang. 2025. Writing-zero: Bridge the gap between non-verifiable tasks and verifiable rewards. *Preprint*, arXiv:2506.00103.

Lewis Tunstall Ben Lipkin Roman Soletskyi Shengyi Costa Huang Kashif Rasul Longhui Yu Albert Jiang Ziju Shen Zihan Qin Bin Dong Li Zhou Yann Fleureau Guillaume Lample Jia LI, Edward Beeching and Stanislas Polu. 2024. Numina-math. [https://github.com/project-numina/aimo-progress-prize](https://github.com/project-numina/aimo-progress-prize/blob/main/report/numina_dataset.pdf).

Takeshi Kojima, Shixiang Shane Gu, Machel Reid, Yutaka Matsuo, and Yusuke Iwasawa. 2022. Large language models are zero-shot reasoners. *Advances in neural information processing systems*, 35:22199–22213.

Jung Hyun Lee, June Yong Yang, Byeongho Heo, Dongyoon Han, and Kang Min Yoo. 2024. Token-supervised value models for enhancing mathematical reasoning capabilities of large language models. *CoRR*.

Aitor Lewkowycz, Anders Johan Andreassen, David Dohan, Ethan Dyer, Henryk Michalewski, Vinay Venkatesh Ramasesh, Ambrose Slone, Cem Anil, Imanol Schlag, Theo Gutman-Solo, Yuhuai Wu, Behnam Neyshabur, Guy Gur-Ari, and Vedant Misra. 2022. Solving quantitative reasoning problems with language models. In *Advances in Neural Information Processing Systems*.

Chen Li, Nazhou Liu, and Kai Yang. 2025a. Adaptive group policy optimization: Towards stable training and token-efficient reasoning. *arXiv preprint arXiv:2503.15952*.

Xuefeng Li, Haoyang Zou, and Pengfei Liu. 2025b. Torl: Scaling tool-integrated rl. *arXiv preprint arXiv:2503.23383*.

Zhihang Lin, Mingbao Lin, Yuan Xie, and Rongrong Ji. 2025. Cppo: Accelerating the training of group relative policy optimization-based reasoning models. *Preprint*, arXiv:2503.22342.

Zichen Liu, Changyu Chen, Wenjun Li, Penghui Qi, Tianyu Pang, Chao Du, Wee Sun Lee, and Min Lin. 2025. Understanding r1-zero-like training: A critical perspective. *Preprint*, arXiv:2503.20783.

Zhenyan Lu, Xiang Li, Dongqi Cai, Rongjie Yi, Fangming Liu, Xiwen Zhang, Nicholas D Lane, and Mengwei Xu. 2024. Small language models: Survey, measurements, and insights. *CoRR*.

Niklas Muennighoff, Zitong Yang, Weijia Shi, Xiang Lisa Li, Li Fei-Fei, Hannaneh Hajishirzi, Luke Zettlemoyer, Percy Liang, Emmanuel Candes, and Tatsunori Hashimoto. 2025a. s1: Simple test-time scaling. In *Workshop on Reasoning and Planning for Large Language Models*.

Niklas Muennighoff, Zitong Yang, Weijia Shi, Xiang Lisa Li, Li Fei-Fei, Hannaneh Hajishirzi, Luke Zettlemoyer, Percy Liang, Emmanuel Candès, and Tatsunori Hashimoto. 2025b. s1: Simple test-time scaling. *Preprint*, arXiv:2501.19393.

Vaskar Nath, Elaine Lau, Anisha Gunjal, Manasi Sharma, Nikhil Baharte, and Sean Hendryx. 2025. Adaptive guidance accelerates reinforcement learning of reasoning models. *Preprint*, arXiv:2506.13923.

Chien Van Nguyen, Xuan Shen, Ryan Aponte, Yu Xia, Samyadeep Basu, Zhengmian Hu, Jian Chen, Mihir Parmar, Sasidhar Kunapuli, Joe Barrow, Junda Wu, Ashish Singh, Yu Wang, Jiuxiang Gu, Franck Dernoncourt, Nesreen K. Ahmed, Nedim Lipka, Ruiyi Zhang, Xiang Chen, and 9 others. 2024. A survey of small language models. *Preprint*, arXiv:2410.20011.

Shubham Parashar, Shurui Gui, Xiner Li, Hongyi Ling, Sushil Vemuri, Blake Olson, Eric Li, Yu Zhang, James Caverlee, Dileep Kalathil, and Shuiwang Ji. 2025. Curriculum reinforcement learning from easy to hard tasks improves llm reasoning. *Preprint*, arXiv:2506.06632.

Jinyoung Park, Jeehye Na, Jinyoung Kim, and Hyunwoo J. Kim. 2025. Deepvideo-r1: Video reinforcement fine-tuning via difficulty-aware regressive grpo. *Preprint*, arXiv:2506.07464.

David Rein, Betty Li Hou, Asa Cooper Stickland, Jackson Petty, Richard Yuanzhe Pang, Julien Dirani, Julian Michael, and Samuel R Bowman. 2024. Gpqa: A graduate-level google-proof q&a benchmark. In *First Conference on Language Modeling*.

John Schulman, Filip Wolski, Prafulla Dhariwal, Alec Radford, and Oleg Klimov. 2017. Proximal policy optimization algorithms. *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. *Preprint*, arXiv:2402.03300.

Taiwei Shi, Yiyang Wu, Linxin Song, Tianyi Zhou, and Jieyu Zhao. 2025. Efficient reinforcement fine-tuning via adaptive curriculum learning. *Preprint*, arXiv:2504.05520.

Débora Souza, Rohit Gheyi, Lucas Albuquerque, Gustavo Soares, and Márcio Ribeiro. 2025. Code generation with small language models: A deep evaluation on codeforces. *Preprint*, arXiv:2504.07343.

Qwen Team. 2024a. Qwen2.5: A party of foundation models.

Qwen Team. 2024b. Qwq: Reflect deeply on the boundaries of the unknown. *Hugging Face*.

Jun Wang, Meng Fang, Ziyu Wan, Muning Wen, Jiachen Zhu, Anjie Liu, Ziqin Gong, Yan Song, Lei Chen, Lionel M. Ni, Linyi Yang, Ying Wen, and Weinan Zhang. 2024. Openr: An open source framework for advanced reasoning with large language models. *Preprint*, arXiv:2410.09671.

Jason Wei, Xuezhi Wang, Dale Schuurmans, Maarten Bosma, brian ichter, Fei Xia, Ed 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*, volume 35, pages 24824–24837. Curran Associates, Inc.

Yuxiang Wei, Olivier Duchenne, Jade Copet, Quentin Carbonneaux, Lingming Zhang, Daniel Fried, Gabriel Synnaeve, Rishabh Singh, and Sida I Wang. 2025. Swe-rl: Advancing llm reasoning via reinforcement learning on open software evolution. *arXiv preprint arXiv:2502.18449*.

Xumeng Wen, Zihan Liu, Shun Zheng, Zhijian Xu, Shengyu Ye, Zhirong Wu, Xiao Liang, Yang Wang, Junjie Li, Ziming Miao, Jiang Bian, and Mao Yang. 2025. Reinforcement learning with verifiable rewards implicitly incentivizes correct reasoning in base llms. *Preprint*, arXiv:2506.14245.

Jialong Wu, Shaofeng Yin, Ningya Feng, and Mingsheng Long. 2025. Rlvr-world: Training world models with reinforcement learning. *Preprint*, arXiv:2505.13934.

Wei Xiong, Jiarui Yao, Yuhui Xu, Bo Pang, Lei Wang, Doyen Sahoo, Junnan Li, Nan Jiang, Tong Zhang, Caiming Xiong, and Hanze Dong. 2025. A minimalist approach to llm reasoning: from rejection sampling to reinforce. *Preprint*, arXiv:2504.11343.

Haoran Xu, Baolin Peng, Hany Awadalla, Dongdong Chen, Yen-Chun Chen, Mei Gao, Young Jin Kim, Yunsheng Li, Liliang Ren, Yelong Shen, Shuohang Wang, Weijian Xu, Jianfeng Gao, and Weizhu Chen. 2025. Phi-4-mini-reasoning: Exploring the limits of small reasoning language models in math. *Preprint*, arXiv:2504.21233.

An Yang, Anfeng Li, Baosong Yang, Beichen Zhang, Binyuan Hui, Bo Zheng, Bowen Yu, Chang Gao, Chengen Huang, Chenxu Lv, and 1 others. 2025a. Qwen3 technical report. *arXiv preprint arXiv:2505.09388*.

An Yang, Beichen Zhang, Binyuan Hui, Bofei Gao, Bowen Yu, Chengpeng Li, Dayiheng Liu, Jianhong Tu, Jingren Zhou, Junyang Lin, and 1 others. 2024. Qwen2. 5-math technical report: Toward mathematical expert model via self-improvement. *CoRR*.

Zhicheng Yang, Zhijiang Guo, Yinya Huang, Xiaodan Liang, Yiwei Wang, and Jing Tang. 2025b. Treerpo: Tree relative policy optimization. *Preprint*, arXiv:2506.05183.

Yixin Ye, Zhen Huang, Yang Xiao, Ethan Chern, Shijie Xia, and Pengfei Liu. 2025. Limo: Less is more for reasoning. *Preprint*, arXiv:2502.03387.

Qiying Yu, Zheng Zhang, Ruofei Zhu, Yufeng Yuan, Xiaochen Zuo, Yu Yue, Tiantian Fan, Gaohong Liu, Lingjun Liu, Xin Liu, and 1 others. 2025. Dapo: An open-source llm reinforcement learning system at scale. *CoRR*.

Yang Yue, Zhiqi Chen, Rui Lu, Andrew Zhao, Zhaokai Wang, Yang Yue, Shiji Song, and Gao Huang. 2025. Does reinforcement learning really incentivize reasoning capacity in llms beyond the base model? *Preprint*, arXiv:2504.13837.

Jingyi Zhang, Jiaxing Huang, Huanjin Yao, Shunyu Liu, Xikun Zhang, Shijian Lu, and Dacheng Tao. 2025. R1-vl: Learning to reason with multimodal large language models via step-wise group relative policy optimization. *CoRR*.

Haizhong Zheng, Yang Zhou, Brian R. Bartoldson, Bhavya Kailkhura, Fan Lai, Jiawei Zhao, and Beidi Chen. 2025. Act only when it pays: Efficient reinforcement learning for llm reasoning via selective rollouts. *Preprint*, arXiv:2506.02177.

Yuhang Zhou, Jing Zhu, Shengyi Qian, Zhuokai Zhao, Xiyao Wang, Xiaoyu Liu, Ming Li, Paiheng Xu, Wei Ai, and Furong Huang. 2025. Disco balances the scales: Adaptive domain- and difficulty-aware reinforcement learning on imbalanced data. *Preprint*, arXiv:2505.15074.

Xialie Zhuang, Peixian Ma, Zhikai Jia, Shiwei Liu, and Zheng Cao. 2025. A technical study into 0.5b reasoning language models. *Preprint*, arXiv:2506.13404.

A Appendix

A.1 Pseudo-code for G^{2}RPO-A

The integral process of our proposed G^{2}RPO-A is shown in Algorithm 1:

Algorithm 1 G^{2}RPO-A

Require: Dataset D with guided reasoning traces $\mathcal{O}_g$, initial guidance length l, guidance ratio k/n

Ensure: Learned model parameters θ

```
θ ← θ_0
H ← []
while not converged do
    if |H| = 2 then
        r_old ← mean(H)
        ρ ← scaleFactor(r_old)
        l ← ρ × l
    end if
    Generate k completions under guidance length l
    for each completion o do
        Compute success indicator r(o)
        Append r(o) to buffer H
    end for
    Update θ via GRPO
end while
return θ
```

A.2 Detailed equations

A.2.1 Group Relative Policy Optimization (GRPO)

The loss function for GRPO is:

$$\mathcal{L}_{\text{GRPO}}(\theta) = -\hat{\mathbb{E}}_t \Big[\min\Big(w_t(\theta)\hat{A}_t, \\ \text{clip}\big(w_t(\theta), 1-\epsilon, 1+\epsilon\big)\hat{A}_t \Big) \\ - \beta \mathcal{D}_{\text{KL}}\big(\pi_\theta(\cdot|s_t) \| \pi_{\text{ref}}(\cdot|s_t)\big) \Big]. \quad (7)$$

where the importance weight $w_{i,t}$ is given by

$$w_t(\theta) = \frac{\pi_\theta(a_t|s_t)}{\pi_{\text{ref}}(a_t|s_t)}. \tag{8}$$

The advantages used in GRPO is a normalized, intra-group relative advantage, which can be calculated by:

$$\hat{A}_{i,t} = \frac{r_i - \text{mean}(\mathbf{r})}{\text{std}(\mathbf{r})}. \tag{9}$$

and the KL divergence is given by:

$$\mathbb{D}_{\text{KL}}[\pi_\theta \| \pi_{\text{ref}}] = \frac{\pi_{\text{ref}}(o_{i,t} \mid q, o_{i,<t})}{\pi_\theta(o_{i,t} \mid q, o_{i,<t})} - \log \frac{\pi_{\text{ref}}(o_{i,t} \mid q, o_{i,<t})}{\pi_\theta(o_{i,t} \mid q, o_{i,<t})} - 1, \tag{10}$$

The function $\text{clip}(x, \min, \max)$ is a bounding function that constrains a value x to a specified interval $[\min, \max]$. Mathematically, it is defined as:

$$\text{clip}(x, a, b) = \begin{cases} a & \text{if } x < a \\ x & \text{if } a \le x \le b \\ b & \text{if } x > b \end{cases}$$

In the context of policy optimization, the expression $\text{clip}(w_t(\theta), 1-\epsilon, 1+\epsilon)$ thus constrains the probability ratio $w_t(\theta)$ to the interval $[1-\epsilon, 1+\epsilon]$. Its primary function is to prevent excessively large policy updates, which is critical for ensuring stable and robust training.

A.2.2 Guided Group Relative Policy Optimization (G^{2}RPO)

The loss function for G^{2}RPO can be described as:

$$\begin{aligned}\mathcal{L}_{\text{guided}}(\theta) =& \mathbb{E}_{(q,a)\sim\mathcal{D},\{g_i\}_{i=1}^G\sim\mathcal{G},\{o_i\}_{i=1}^G\sim\pi_{\text{ref}}(\cdot|q,g_i)} \\ & \Bigg[\frac{1}{G}\sum_{i=1}^{G}\frac{1}{|o_i|+|g_i|} \\ & \Bigg(\sum_{t=1}^{|g_i|}\min\big(w_{g,i,t}\hat{A}_{i,t}, \text{clip}\left(w_{g,i,t}, 1-\epsilon, 1+\epsilon\right)\big)\hat{A}_{i,t} \\ & + \sum_{t=1}^{|o_i|}\min\big(w_{o,i,t}\hat{A}_{i,t}, \text{clip}\left(w_{o,i,t}, 1-\epsilon, 1+\epsilon\right)\big)\hat{A}_{i,t} \\ & - \beta\mathcal{D}_{\text{KL}}(\pi_\theta\|\pi_{\text{ref}})\Bigg)\Bigg],\end{aligned} \tag{11}$$

where $w_{g,i,t}$ and $w_{o,i,t}$ denotes the token-level weighting coefficient of guidance g_i and the model outputs o_i:

$$\begin{aligned} w_{g,i,t} &= \frac{\pi_\theta(o_{g,i,t} \mid q, o_{g,i,<t})}{[\pi_{\theta_{\text{old}}}(o_{g,i,t} \mid q, o_{g,i,<t})]}, \\ w_{o,i,t} &= \frac{\pi_\theta(o_{o,i,t} \mid q, o_{o,i,<t})}{[\pi_{\theta_{\text{old}}}(o_{o,i,t} \mid q, o_{o,i,<t})]}. \end{aligned} \tag{12}$$

A.3 Reward Function Definitions

Each completion c is evaluated by up to five base rewards $R_m(c)$, $m \in \{\text{tag}, \text{fmt}, \text{code_fmt}, \text{code}, \text{acc}\}$, combined via nonnegative weights $\{\lambda_m\}$:

$$R_{\text{total}}(c) = \sum_m \lambda_m \, R_m(c).$$

A.3.1 Tag-Count Reward (R_{tag}).

Let $\mathcal{T} = \{\langle\text{think}\rangle, \langle/\text{think}\rangle, \langle\text{answer}\rangle, \langle/\text{answer}\rangle\}$. Then

$$R_{\text{tag}}(c) = \tfrac{1}{4}\sum_{t\in\mathcal{T}} \mathbf{1}\{c \text{ contains } t \text{ exactly once}\}, \quad R_{\text{tag}} \in [0,1].$$

A.3.2 Format Reward (R_{fmt}).

Checks that c matches $\langle\text{think}\rangle \cdots \langle/\text{think}\rangle\langle\text{answer}\rangle \cdots \langle/\text{answer}\rangle$:

$$R_{\text{fmt}}(c) = \begin{cases} 1, & \text{if format is exact,} \\ 0, & \text{otherwise.} \end{cases}$$

A.3.3 Code-Format Reward ($R_{\text{code_fmt}}$).

Using language-specific fenced blocks inside the answer section, together with the required <think>...</think>| and <answer>...</answer>| tags. Formally, let $\mathcal{L}$ be the set of allowed language labels and $\text{match}(c; \mathcal{P}_\ell) \in \{0,1\}$ be a pattern-matching predicate against the code-format pattern for label ℓ. We define

$$R_{\text{code_fmt}}(c) = \mathbf{1}\{\exists\, \ell \in \mathcal{L} : \ \text{match}(c; \mathcal{P}_\ell) = 1\}.$$

Here, $\mathcal{P}_\ell$ requires the tag sequence <think>...</think>| followed by <answer>...| containing a fenced code block labeled ℓ, and closing with </answer>|. The reward returns 1 on an exact match and 0 otherwise.

A.3.4 Code-Execution Reward (R_{code}).

Given M test cases $\{(x_m, y_m)\}$ and execution indicator $\text{exec}(c, x_m) \in \{0,1\}$:

$$R_{\text{code}}(c) = \frac{1}{M}\sum_{m=1}^{M} \text{exec}(c, x_m), \quad R_{\text{code}} \in [0,1].$$

A.3.5 Accuracy Reward (R_{acc}).

Parsing LaTeX and verifying semantic equivalence yields

$$R_{\text{acc}}(c) = \begin{cases} 1, & \text{if output matches,} \\ & \text{gold} \\ 0, & \text{otherwise.} \end{cases} \tag{13}$$

A.3.6 Group-Level Aggregation.

For G completions $\{c_i\}$, form vector $\mathbf{r} = (R_{\text{total}}(c_i))_{i=1}^{G}$, compute mean μ_r and standard deviation σ_r, and normalize:

$$\hat{A}_i = \frac{r_i - \mu_r}{\sigma_r},$$

to obtain the per-token advantage estimates used in the GRPO update.

A.4 Experiment Details

A.4.1 Dataset filtration

We curated our dataset by applying a two-stage filtration process to the math-220k dataset from open-r1 (Wang et al., 2024) to select for samples containing complete, correct, and technically parsable thinking trajectories.

Sourcing from the math-220k dataset offered by open-r1 (Wang et al., 2024), we first perform two filtrations on the math dataset we use:

1. The first filtration ensures the remaining samples are with:
   (a) Integrated thinking trajectories.
   (b) Correct final answers given by thinking trajectories verified by ground truths.
2. The second filtration ensures the remaining samples are with:
   (a) Parsable by the python package "math verify".

Inspired by ADARFT(Shi et al., 2025), we order our dataset by the difficult level of each sample. In detail, we use model Qwen2.5-7B(Team, 2024a) to solve each problem for 128 times, taking the ratio of the number of failures to the total number of attempts as the difficulty score for them. However, the difficulty scores assigned to each sample in our dataset via the scoring methodology proposed in ADARFT exhibited little variance, making them largely indistinguishable from one another. The distribution of difficult score is shown in Figure 5:

As a result, we directly choose samples from five difficulty levels containing cn_contest, aops_forum, amc_aime, olympiads and olympiads_ref ordering those chosen samples from easy to hard.

The dataset we use to train on code tasks is obtained from verifiable-coding-problems-python which is also released by open-r1 (Wang et al., 2024). We also We use a much stonger model qwq-32b-preview (Team, 2024b) to generate the step by step chain of thought for each problem, which will be used as the guidances in the G^{2}RPO-A. After the generation of those thinking trajectories, we also run the them through two filtrations:

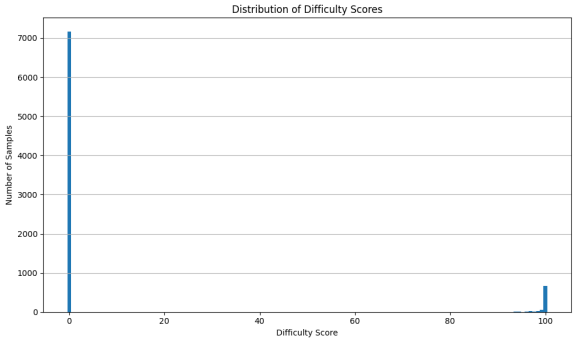


Figure 5: To approximate a difficulty score for each query, we adopt the methodology from ADARFT (Shi et al., 2025). In this approach, the Qwen2.5-1.5B-Instruct model is deployed to solve each query 128 times. The difficulty score is then calculated based on the number of failed attempts.

1. The first filtration ensures the thinking trajectories we generate are:
   (a) Complete.
   (b) The total number of tokens does not exceed the max guidance length we set.
2. The second filtration ensures the thinking trajectories we generate are:
   (a) With code reward being equal to 1 returned by the reward computation function.

We also do a filtration on the query part of the dataset to make sure the numbers of tokens in queries won't surpass 512 because the descriptions of coding tasks are always longer than those of math tasks.

A.4.2 Sandbox for code tasks

For the verification of model-generated code snippets, we employ the E2B sandboxed execution environment(E2B-Dev, 2024), an open-source cloud sandbox environment designed for securely executing AI-generated code.

A.4.3 Evaluation

We assess our models on three public mathematical–reasoning benchmarks—MATH500 (Hendrycks et al., 2021), MINERVA-MATH (Lewkowycz et al., 2022), and GPQA (Rein

Base Model	α	Benchmark	Base	GRPO	G^{2}RPO-A
DeepSeek-Coder -6.7b-Base	0.71	humaneval	51.27	66.71	**70.11**
		LCB	12.38	**25.15**	23.52
DeepSeek-MATH -7B-Base	0.29	MATH500	31.37	49.78	**55.17**
		Minerva	5.61	18.19	**21.63**
		gpqa	17.28	**22.14**	21.52

Table 13: **Performance of G^{2}RPO-A on DeepSeek series base models.** We report accuracy (%) on various benchmarks. Models are trained for 5 epochs.

et al., 2024). For the mathematical trainig of Qwen3-1.7B and Qwen3-0.6B, AIME24 (Jia LI and Polu, 2024) and AIME25 benchmarks are also used. And for code tasks, we use two benchmarks: HUMANEVAL (Chen et al., 2021) and LIVE CODE BENCH(Jain et al., 2024). For Math 500 task, We first present five accuracies on five difficulty levels and then directly take the average of them as our final test results on Math 500. And the evaluation settings are exactly the same as those presented in its official tutorial: all tasks are evaluated in a zero-shot setting without any prompting techniques. We set the evaluation batch size to 128 and allocate a budget from 4096 to 25000 tokens according to the models' max model lengths. The generation configurations are: temperature set to 0.6, top_p set to 0.95 and top_k set to 20.

A.4.4 Training details

Our G^{2}RPO-A algorithm is implemented on top of the fully open-source OPEN-R1 framework (Face, 2025). We use the following hyper-parameters: (i) batch size of 12 per GPU; (ii) number of rollouts per sample set to 12 for 0.6B and 1.7B backbones, and 7 for 7B and 8B backbones; (iii) initial learning rate 1×10^{-6}, decayed with a cosine schedule and a warm-up ratio of 0.1; (iv) a training set of $1{,}000$ problems for 5 epochs. For ablation studies, we only implement 1 training epoch for fast verification (which is also the default training epoch number in OPEN-R1). The maximum sequence length is limited to $4{,}608$ tokens. Thus, the max completion length and max prompt length could be automatically adjusted according to the guidance length provided at every training step.

A.5 Extra results

We also conduct experiments on DeepSeek series (Guo et al., 2024; Shao et al., 2024) to verify the effectiveness of our proposed method. Results are shown in Table 13.