

# GRPO-VPS: ENHANCING GROUP RELATIVE POLICY OPTIMIZATION WITH VERIFIABLE PROCESS SUPERVISION FOR EFFECTIVE REASONING

**Anonymous authors**

Paper under double-blind review

## ABSTRACT

Reinforcement Learning with Verifiable Rewards (RLVR) has advanced the reasoning capabilities of Large Language Models (LLMs) by leveraging direct outcome verification instead of learned reward models. Building on this paradigm, Group Relative Policy Optimization (GRPO) eliminates the need for critic models but suffers from indiscriminate credit assignment for intermediate steps, which limits its ability to identify effective reasoning strategies and incurs overthinking. In this work, we introduce a model-free and verifiable process supervision via probing the model’s belief in the correct answer throughout its reasoning trajectory. By segmenting the generation into discrete steps and tracking the conditional probability of the correct answer appended at each segment boundary, we efficiently compute interpretable segment-wise progress measurements to refine GRPO’s trajectory-level feedback. This approach enables more targeted and sample-efficient policy updates, while avoiding the need for intermediate supervision derived from costly Monte Carlo rollouts or auxiliary models. Experiments on mathematical and general-domain benchmarks show consistent gains over GRPO across diverse models: up to 2.6-point accuracy improvements and 13.7% reasoning-length reductions on math tasks, and up to 2.4 points and 4% on general-domain tasks, demonstrating strong generalization.

## 1 INTRODUCTION

Advanced by Reinforcement Learning with Verifiable Rewards (RLVR) (Shao et al., 2024; Yu et al., 2025a), Large Language Models (LLMs) have demonstrated remarkable capabilities in complex reasoning tasks, ranging from mathematical problem solving (OpenAI, 2024; Team et al., 2025; Shao et al., 2024) to multi-hop question answering (Huang et al., 2025; Song et al., 2025). The success of RLVR is largely attributed to computing rewards via direct outcome verification, rather than relying on reward models that complicate the training pipeline and are prone to reward hacking (Yu et al., 2025a). In a similar vein, Group Relative Policy Optimization (GRPO) (Shao et al., 2024) eliminates critic models for token-level advantage estimation, instead uniformly propagating trajectory-level advantages to intermediate steps. While this simplification avoids the challenges of training a critic model and reduces associated overhead, indiscriminate credit assignment hinders sample efficiency and limits the policy model’s ability to learn effective reasoning strategies (Qu et al., 2025).

To address this limitation, we explore enhancing GRPO with model-free, verifiable process supervision derived from the annotated final answer. Our key insight is that the contribution of intermediate reasoning steps can be probed by the probability increment of the reference answer appended at corresponding breakpoints. This is supported by observations in Figure 1: (1) at the macro level, the average probed probability increases as reasoning progresses, with a more pronounced trend for trajectories that ultimately reach the

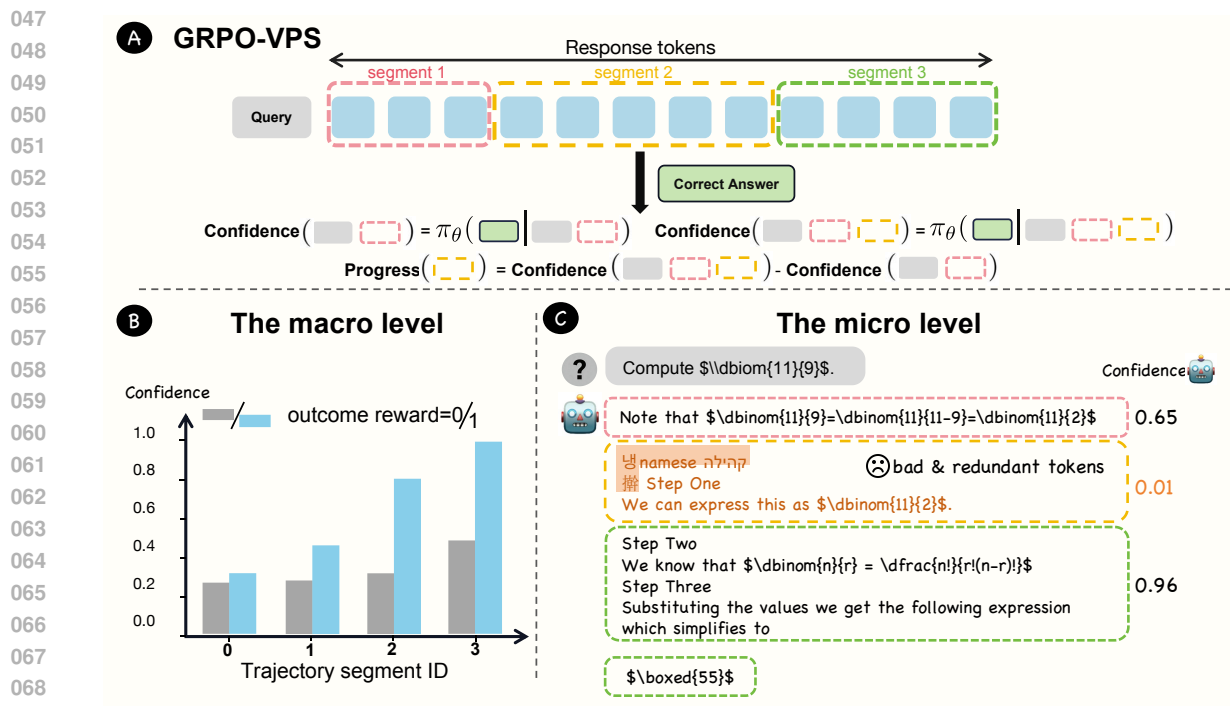


Figure 1: (A) GRPO-VPS supervises intermediate reasoning via a segment-wise process signal computed as the change in the model’s belief in the correct answer across consecutive reasoning segments. (B) At the macro level, we visualize how the probed confidence evolves in the reasoning models. Trajectories that ultimately lead to correct answers exhibit more pronounced upward trends. (C) At the micro level, reasoning chunks with negative confidence increments often contain hallucinations, redundancies, or unhelpful detours.

correct answer; and (2) at the micro level, reasoning segments that reduce the probed probability tend to be of low quality. Based on these observations, our method leverages the model’s own reasoning trace to generate localized supervision signals, enabling more targeted and effective policy updates. Specifically, we segment the model’s response into discrete reasoning segments and strategically concatenate the correct final answer at each segment boundary. By extracting the model’s conditional probability of the correct answer at these positions, we obtain a proxy for its evolving belief state. The differences in these probabilities between adjacent segments serve as segment-wise supervision signals, complementing trajectory-level advantages and quantifying the contribution of each reasoning segment toward the final outcome.

This approach offers two key benefits: (1) it provides dense, interpretable feedback aligned with the model’s internal decision flow and (2) it avoids reliance on auxiliary models (Schulman et al., 2017; Zha et al., 2025; Cui et al., 2025; He et al., 2024b) or Monte Carlo rollouts (Qu et al., 2025; Dai et al., 2025), ensuring high efficiency and scalability and adhering to the design principles established by RLVR and GRPO. Through this fine-grained supervision mechanism, we aim to enhance the sample efficiency of RL training, paving the way for learning more effective and efficient reasoning behaviors.

Our experiments show substantial gains across four math reasoning benchmarks. Compared to GRPO, our method achieves up to +2.6 points Pass@1 on Qwen2.5-Math-1.5B and +1.1 point on Qwen2.5-Math-7B,

094 while concurrently reducing reasoning length by 11.0% to 13.7%. It also consistently outperforms the GRPO  
095 variant (Dai et al., 2025) that relies on costly Monte Carlo rollouts for segment-wise advantage estimation  
096 and auxiliary models (Cui et al., 2025; He et al., 2024b; Schulman et al., 2017). Furthermore, evaluation  
097 on four general-domain reasoning benchmarks confirms strong generalization, with gains of 1.8 points on  
098 MMLUPro and 2.4 points on TheoremQA. These results highlight the effectiveness and scalability of our  
099 verifiable process supervision in delivering more accurate and concise reasoning.

100 In summary, our main contributions are:

- 101
- 102 • We identify and empirically validate that a model’s evolving belief in the correct answer can serve as  
103 a model-free, interpretable signal for reasoning quality of intermediate steps. This enables fine-grained  
104 supervision without auxiliary models or Monte Carlo rollouts.
- 105 • We propose GRPO-VPS, a simple yet effective approach to enhance GRPO with granular, segment-wise  
106 process supervision, avoiding indiscriminate credit assignment and improving sample efficiency.
- 107 • Our empirical results show that the method achieves strong performance on challenging math reasoning  
108 tasks, demonstrating that our method enhances both reasoning effectiveness and efficiency in comparison  
109 with GRPO and its variants.  
110

## 111 2 RELATED WORK

112 **Group Relative Policy Optimization (GRPO).** Reinforcement Learning with Verifiable Rewards (RLVR)  
113 has become a prominent paradigm for fine-tuning LLMs, using definitive signals from rule-based verifiers  
114 to circumvent the need for costly and potentially biased reward models (Shao et al., 2024; Yu et al., 2025a).  
115 Within this paradigm, GRPO (Shao et al., 2024) offers a lightweight and efficient alternative to critic-based  
116 algorithms like PPO (Schulman et al., 2017). By comparing final outcomes across a group of sampled  
117 trajectories, GRPO eliminates the need for a separate value network. However, this simplification comes at  
118 the cost of indiscriminate credit assignment: a single, trajectory-level reward is uniformly propagated to all  
119 intermediate tokens. This can inadvertently reinforce spurious reasoning steps in a successful trajectory or  
120 penalize promising partial logic in a failed one. Our work addresses this limitation by introducing a fine-  
121 grained process supervision mechanism, enhancing its credit assignment capabilities without sacrificing its  
122 lightweight nature.

123 **Process supervision for reasoning.** Recent work has explored injecting fine-grained supervision into the  
124 reasoning process to better guide long-form generation. These efforts can be broadly categorized into model-  
125 based and model-free approaches. Model-based supervision utilizes an auxiliary model to provide fine-  
126 grained feedback. Critic-based methods, often using PPO, train a value network to estimate the expected  
127 return from intermediate states (Yue et al., 2025; Kazemnejad et al., 2024). However, in long-horizon rea-  
128 soning tasks, the critic’s signal can diminish or become unreliable due to the long delay in receiving the final  
129 outcome reward (Shao et al., 2024; Yue et al., 2025). Process Reward Models (PRMs) (Lightman et al., 2023;  
130 Wang et al., 2023) offer an alternative but are typically trained offline, making them vulnerable to reward  
131 hacking and distributional shift. These approaches introduce significant system complexity, requiring an  
132 extra model to be trained, maintained, and served alongside the policy. Model-free process supervision aims  
133 to provide granular feedback without auxiliary models. Recent works have made progress in this direction.  
134 For instance, S-GRPO (Dai et al., 2025) introduces a “serial group” objective with decaying rewards to en-  
135 courage earlier, more efficient reasoning. MRT (Qu et al., 2025) frames the problem as meta reinforcement  
136 learning and computes a dense “progress” reward based on the change in the likelihood of eventual success.  
137 Their reward estimation can require complex, rollout-based procedures or multiple generation branches from  
138 intermediate states, which compromises training efficiency. In contrast, our method simplifies the process  
139 supervision workflow by deriving a high-quality signal from the known ground-truth answer, requiring only  
140

141 a single forward pass per generated trajectory. This makes our approach more efficient while still providing  
 142 the benefits of fine-grained, verifiable process feedback.

### 144 3 METHODOLOGY: PROCESS SUPERVISIONS FROM VERIFIABLE OUTCOMES

145 LLMs trained under GRPO still suffer from indiscriminate credit assignment, where sparse outcome-based  
 146 rewards fail to guide intermediate reasoning steps. To address this, we present a verifiable process supervi-  
 147 sion framework for enhancing GRPO with fine-grained credit assignment. We first introduce a segmentation  
 148 strategy that uses token-level entropy to identify high-uncertainty transitions and partition trajectories into  
 149 semantically meaningful reasoning steps (Section 3.1). We then introduce segment-wise progress estimation  
 150 to quantify the contribution of each reasoning segment based on changes in model confidence (Section 3.2).  
 151 Finally, we incorporate this localized feedback into GRPO’s token-level updates, forming a hybrid advantage  
 152 that fuses outcome-based and process-level signals (Section 3.3).  
 153

#### 154 3.1 REASONING PROCESS SEGMENTATION

155 Recent studies have revealed that performance gains in RLVR are primarily driven by critical decision points  
 156 characterized by high token-level uncertainty (Yang et al., 2025; Wang et al., 2025). Inspired by SPO (Guo  
 157 et al., 2025), we adopt an Adaptive Entropy-based Cutpoint Partition strategy, leveraging token-level entropy  
 158 to robustly identify reasoning ”junctions” where the model’s trajectory is likely to diverge.  
 159

160 Formally, given a response  $o$  of length  $T$ , We identify a set of candidate cutpoints  $\mathcal{U} \subseteq \{1, \dots, T\}$  by  
 161 selecting tokens whose entropy exceeds an adaptive threshold:

$$162 \mathcal{U} = \{t \mid e_t^i \geq \tau\}, \quad (1)$$

163 where  $\tau$  is determined from the entropy distribution of  $o$  (e.g. via a percentile-based rule). To partition  $o$   
 164 into  $M$  reasoning segments  $(z_1, \dots, z_M)$ , we choose boundary indices  $\{t_1, \dots, t_{M+1}\}$  with  $t_1 = 1$  and  
 165  $t_{M+1} = T + 1$ , such that the number of cutpoints in each segment is approximately balanced:

$$166 |\mathcal{U} \cap [t_m, t_{m+1})| \approx \frac{|\mathcal{U}|}{M}, \quad \forall m \in \{1, \dots, M\}. \quad (2)$$

167 This heuristic ensures that each segment contains a comparable number of high-entropy positions, yield-  
 168 ing a balanced and semantically meaningful segmentation of the reasoning trajectory. Our experiments  
 169 demonstrate that this adaptive strategy yields superior performance compared to fixed-token partition (see  
 170 Section 4.3).  
 171

#### 172 3.2 PROGRESS AS PROCESS SUPERVISION

173 Based on the reasoning process segmentation, we propose to leverage segment-wise progress as a form of  
 174 process supervision to address the indiscriminate credit assignment of GRPO. This formulation provides  
 175 a dense, model-free, and scalable supervision signal that quantifies the incremental contribution of each  
 176 reasoning segment toward the correct final answer.

177 Given an input prompt  $x$  and a trajectory  $o$  generated by the policy  $\pi_\theta$ , we compute a segment-wise confi-  
 178 dence score  $C(z_{\leq k})$  representing the model’s conditional probability of the target answer  $y^*$  after generating  
 179 the first  $k$  reasoning steps:

$$180 C(z_{\leq k}) = \pi_\theta(y^* \mid x, z_{\leq k}) \quad (3)$$

181 where  $x$  is the input question and  $z_{\leq k} = (z_1, \dots, z_k)$  denotes partial reasoning trace up to segment  $k$ . The  
 182 initial value, before any reasoning, is  $C_0 = P(y^* \mid x)$ .  
 183

To quantify the contribution of each reasoning segment, we define a segment-wise progress score, denoted as  $\Delta C_k$ , is then computed as the change in this confidence score, effectively isolating the contribution of that specific step:

$$\begin{aligned}\Delta C_k &= C(z_{\leq k}) - C(z_{\leq k-1}) \\ &= \pi_{\theta}(y^* | x, z_{\leq k}) - \pi_{\theta}(y^* | x, z_{\leq k-1})\end{aligned}\tag{4}$$

where  $\Delta C_k \in [-1, 1]$  due to the probabilistic range of confidence scores. This results in a vector  $\Delta C = [\Delta C_1, \dots, \Delta C_m]$  of segment-level supervision signals for each trajectory. It reflects how much each segment improves (or worsens) the model’s belief in the final answer.

### 3.3 GRPO WITH PROCESS SUPERVISION

We design a hybrid advantage signal that fuses sparse outcome-level feedback with dense, segment-level process supervision. For each prompt, we sample a group of  $G$  trajectories  $\{o^1, o^2, \dots, o^G\}$  from the policy  $\pi_{\theta}$ . Each trajectory  $o^i = (z_1^i, \dots, z_m^i, y^i)$  with binary correctness label  $r^i \in \{0, 1\}$ , we compute the group-relative advantage:

$$A^i = r^i - \frac{1}{G} \sum_{j=1}^G r^j,\tag{5}$$

where  $G$  is the number of responses sampled for the same prompt. To complement this global signal, we inject a localized feedback term  $\Delta C_k$ , which quantifies the incremental gain in the model’s belief in the correct answer after each reasoning segment  $z_k^i$ , as defined in Section 3.2. The final hybrid advantage at step  $k$  is:

$$\tilde{A}_k^i = \underbrace{A^i}_{\text{Outcome}} + \underbrace{\alpha \cdot \Delta C_k}_{\text{Process}},\tag{6}$$

where  $\alpha$  is a weighting factor balancing the two components. We empirically set  $\alpha = 1.2$  and found it work well. Sensitivity to  $\alpha$  can be found in Appendix A.4.3.

We then define the final on-policy gradient estimator as:

$$\nabla_{\theta} J(\theta) = \frac{1}{G} \sum_{i=1}^G \sum_{k=1}^M (A^i + \alpha \cdot \Delta C_k) \cdot \nabla_{\theta} \log \pi_{\theta}(z_k^i | x, z_{<k}^i)\tag{7}$$

where the total advantage combines two signals:  $A^i$  provides sparse, trajectory-level feedback based on the final outcome, while  $\alpha \cdot \Delta C_k$  injects dense, segment-level guidance reflecting the progress toward the correct answer. The full algorithm of GRPO-VPS is shown in Algorithm 1.

## 4 EXPERIMENT

### 4.1 SETUP

**Models and baselines.** We conduct experiments on two model families, including Qwen2.5-Math-1.5B, Qwen2.5-Math-7B (Yang et al., 2024) and Gemma-2-2B-it (Team, 2024a). To ensure fair comparison, we include a comprehensive set of baselines categorized by their use of outcome-level vs. process-level supervision:

**Algorithm 1: GRPO WITH VERIFIABLE PROCESS SUPERVISION**

**Input:** Question set  $\mathcal{D}$ ; Base policy  $\pi_{\theta_b}$ ; Entropy percentile  $p$ ; Segment number  $m$ ; Progress reward weight  $\alpha$ ; Training steps  $S$

**Output:** Updated policy parameters  $\theta$

```

1 Initialize policy  $\pi_{\theta} \leftarrow \pi_{\theta_b}$ 
2 for iteration = 1, ..., S do
3   Sample a mini-batch  $\mathcal{D}_b \subset \mathcal{D}$ 
4   for each question  $x \in \mathcal{D}_b$  with target answer  $y^*$  do
5     Generate full trajectory  $o \sim \pi_{\theta}(\cdot|x)$ 
6     Compute token entropies  $e_t$ 
7     Segment  $o = (z_1, \dots, z_m, y)$  using adaptive entropy-based cutpoints
8     Compute initial confidence  $C_0 = \pi_{\theta}(y^* | x)$ 
9     for  $k = 1$  to  $m$  do
10       $C_k = \pi_{\theta}(y^* | x, z_{\leq k})$ 
11       $\Delta C_k = C_k - C_{k-1}$ 
12    Compute hybrid advantages  $\tilde{A}_t$  using Eq. 6
13    Update policy  $\pi_{\theta}$  using policy gradient with  $\tilde{A}_t$ 

```

- **Outcome Supervision Only.** This category includes methods that rely solely on final answer correctness for reward assignment. We consider GRPO and its recent variants, DrGRPO (Liu et al., 2025) and GSPO (Zheng et al., 2025), which enhance group-wise comparison or propagate advantages with entropy-based mechanisms. We also include the BASE models without RL fine-tuning for reference.
- **With Process Supervision.** This group covers methods that incorporate intermediate supervision beyond outcome-level rewards. We evaluate S-GRPO (Dai et al., 2025), which relies on Monte Carlo rollouts with forced early stops to construct sub-trajectories, and assigns segment-level rewards based on their predicted outcomes. We also compare against PRIME-style reward modeling, represented by the public Eurus-2-7B-PRIME (Cui et al., 2025; Yuan et al., 2024), and a controlled variant where we fine-tune Qwen2.5-Math-1.5B and 7B with Skywork-o1-prm using GRPO. These baselines provide strong comparisons for evaluating the effectiveness of verifiable process supervision in our method.

**Training setup.** In line with prior work (Liu et al., 2025), we use MATH (Hendrycks et al., 2021), which contains 7,500 problems. We train the models using the verl framework (Sheng et al., 2024). We sample 8 rollouts per prompt, with a temperature of 1.0 and the maximum response length of 3,072 tokens. The batch size is set to 512, the mini-batch size to 128, and the learning rate to  $1 \times 10^{-6}$ . The training is conducted on a single node with  $8 \times$  H800 GPUs. More hyperparameter settings can be found in Appendix A.1.

**Evaluation setup.** We evaluate on four widely used math reasoning benchmarks, including the test sets of MATH, AIME 2024 (MAA, 2024), AMC23 (MAA, 2023) and OlympiadBench (He et al., 2024a). We set temperature to 1.0, top  $p$  set to 1, and maximum output length set to 3,072 tokens for inference. Due to the high variance of the outputs from reasoning models, we report the average Pass@1 over 4 runs. To ensure accurate evaluation, we utilize Math-Verify<sup>1</sup> to check for answer equivalence.

<sup>1</sup><https://github.com/huggingface/Math-Verify>

Table 1: Experimental results on Qwen-Math Models. BASE denotes the corresponding Qwen or Gemma base model without any fine-tuning.

Model	AMC23		AIME24		MATH		OLYMPIAD		Overall	
	Pass@1	AvgToken	Pass@1	AvgToken	Pass@1	AvgToken	Pass@1	AvgToken	Pass@1	AvgToken
<b>Qwen2.5-Math-1.5B</b>										
<b>Outcome Supervision Only</b>										
BASE	16.3	5534	5.8	5782	22.6	5008	19.0	4624	15.9	5237
GRPO	46.9	3872	18.3	4655	68.8	2749	30.0	3811	41.0 <sub>+25.1</sub>	3772 <sub>-28.0%</sub>
DrGRPO	44.4	3681	<b>20.0</b>	4878	70.5	2757	30.8	3707	41.4 <sub>+25.5</sub>	3756 <sub>-28.3%</sub>
GSPO	45.0	4044	13.3	5146	68.0	2778	28.9	3727	38.8 <sub>+22.9</sub>	3924 <sub>-25.1%</sub>
<b>With Process Supervision</b>										
GRPO w/ Skywork-1.5B	50.0	3527	11.7	4738	71.4	2758	29.8	3403	40.7 <sub>+24.8</sub>	3607 <sub>-31.1%</sub>
S-GRPO	46.3	3535	14.2	4580	67.5	2664	28.3	<b>3313</b>	39.1 <sub>+23.2</sub>	3523 <sub>-32.7%</sub>
<b>GRPO-VPS</b>	<b>55.0</b>	<b>3425</b>	<b>15.0</b>	<b>4278</b>	<b>72.2</b>	<b>2391</b>	<b>32.1</b>	3326	<b>43.6</b> <sub>+27.7</sub>	<b>3355</b> <sub>-35.9%</sub>
<b>Qwen2.5-Math-7B</b>										
<b>Outcome Supervision Only</b>										
BASE	23.8	4485	10.8	5390	31.9	4370	18.8	4699	21.3	4736
GRPO	62.5	3552	30.0	4730	75.4	2671	36.8	3527	51.2 <sub>+29.9</sub>	3620 <sub>-23.6%</sub>
DrGRPO	64.4	3496	28.3	4449	75.5	2700	35.9	3490	51.0 <sub>+29.7</sub>	3534 <sub>-25.4%</sub>
GSPO	60.6	3725	30.8	4191	75.2	2661	36.8	3481	50.9 <sub>+29.6</sub>	3515 <sub>-25.8%</sub>
<b>With Process Supervision</b>										
Eurus-2-7B-PRIME	<b>65.0</b>	4368	15.0	5731	<b>78.3</b>	3024	<b>42.2</b>	4405	50.1 <sub>+28.8</sub>	4382 <sub>-7.5%</sub>
GRPO w/ Skywork-7B	63.8	3990	30.0	4697	75.6	2736	38.8	3740	52.0 <sub>+30.7</sub>	3791 <sub>-20.0%</sub>
S-GRPO	61.9	3235	25.8	<b>3723</b>	74.9	2464	35.8	3123	49.6 <sub>+28.3</sub>	3136 <sub>-33.8%</sub>
<b>GRPO-VPS</b>	<b>64.8</b>	<b>3220</b>	<b>31.7</b>	3829	<b>75.6</b>	<b>2372</b>	<b>37.2</b>	<b>3064</b>	<b>52.3</b> <sub>+31.0</sub>	<b>3121</b> <sub>-34%</sub>
<b>Gemma-2-2B-it</b>										
BASE	3.8	<b>392</b>	0.0	418	21.3	345	2.8	408	6.9	391
GRPO	7.5	3205	0.0	3183	31.0	2938	5.9	3133	11.1 <sub>+4.3</sub>	3115 <sub>+696%</sub>
<b>GRPO-VPS</b>	<b>8.1</b>	2544	<b>0.8</b>	2446	<b>31.6</b>	2212	<b>6.2</b>	2396	<b>11.7</b> <sub>+4.8</sub>	<b>2399</b> <sub>+513%</sub>

## 4.2 MAIN RESULTS

**GRPO-VPS improves mathematical reasoning performance.** As shown in Table 1, our method achieves the highest average accuracy. On Qwen2.5-Math-1.5B, it yields an average gain of 27.7 points. On Qwen2.5-Math-7B, the gain reaches +31.0 points overall. Meanwhile, the average output length is reduced by 35.9% and 34.0% on the 1.5B and 7B models, respectively. Similar trends are observed on Gemma-2-2B-it, where our method improves average accuracy from 6.9 to 11.7 while substantially reducing output length compared to GRPO.

**Comparison with outcome supervised RL.** Compared to GRPO and its recent variants, GRPO-VPS achieves the highest overall accuracy across all benchmarks while substantially shortening the reasoning length. On Qwen2.5-Math-1.5B, it improves Pass@1 by more than 3% over GRPO and sustains a comparable margin on the 7B model, with an average reduction of 10–11% in output length. GRPO-VPS achieves larger gains in accuracy and produces shorter outputs. These results suggest that our method improves both effectiveness and efficiency of policy updates, leading to more focused reasoning traces. On Gemma-2-2B-it, our method further reduces the output length compared to GRPO while achieving higher accuracy. Although the Gemma base model exhibits shorter responses, training with reinforcement learning unlocks its long-chain reasoning behavior, as further evidenced in Appendix A.4.1.

**Comparison with model-based process supervision.** Compared to S-GRPO, which uses multi-rollback early-exit probing to assign segment-level rewards, our method achieves higher accuracy and shorter outputs. To ensure a fair comparison, we adopt the same GRPO training pipeline to finetune both our method and Skywork-o1-prm. Under this controlled setup, GRPO-VPS consistently outperforms the Skywork baseline across both 1.5B and 7B model scales, achieving higher accuracy while generating more concise reasoning traces. We further compare our method with Eurus-2-7B-PRIME, a model trained using PRIME-style reward modeling. Notably, Eurus performs poorly on AIME24 with 15.0%, substantially lower than the 31.7%

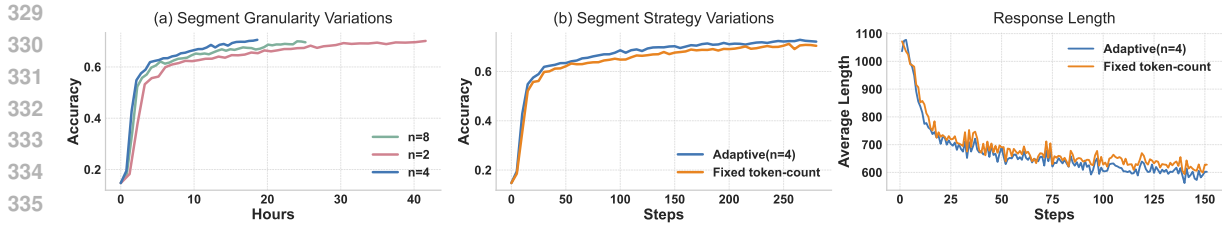


Figure 2: **(a)** Effect of segment granularity by varying the average number of points per segment ( $n$ ), evaluated by validation accuracy under the same wall-clock time. **(b)** Comparison between the proposed adaptive segmentation strategy and a fixed token-count partition baseline. All results are obtained on the MATH Evaluation dataset.

achieved by GRPO-VPS. This discrepancy reveals a lack of generalization in PRM-based methods, which tend to overfit to familiar patterns seen during training. In contrast, our process supervision, grounded in the model’s own belief dynamics, offers more robust and consistent improvements across tasks, while reducing reasoning length by up to 34.0%.

### 4.3 ABLATION STUDY

**Effect of segment granularity.** To investigate the impact of segment granularity on the efficiency and performance of our adaptive strategy, we evaluate the impact of segment granularity by varying the average number of points  $n$  per segment. As shown in Figure 2(a),  $n = 4$  achieves the optimal trade-off between training efficiency and performance under the same wall-clock time. While finer-grained segmentation ( $n = 2$ ) provides more precise local adjustments, it incurs prohibitive computational overhead that slows convergence. This confirms that our segment-level design effectively strikes a balance between computational economy and optimization precision, avoiding the excessive costs associated with fine-grained estimation while ensuring superior learning dynamics.

**Comparison of Different segment strategies.** We compare our adaptive segmentation strategy with a naive fixed token-count partition baseline, where each response is evenly divided into a fixed number of segments. For the fixed baseline, we set the number of segments to 6, exceeding the effective segmentation budget of our adaptive method. As shown in Figure 2(b), the adaptive strategy converges faster, achieves higher accuracy, and maintains shorter responses throughout training. These results demonstrate that placing segment boundaries based on informative decision points is more effective than uniform partitioning, validating the design of our adaptive segmentation strategy.

**Effect of outcome supervision signal.** As shown in Table 2, removing the outcome reward and relying solely on segment-level verifiable supervision results in degradation compared to the full method, confirming its essential role in guiding the model toward globally correct reasoning. While segment-level supervision alone provides fine-grained feedback, it lacks a reliable global anchor. Combining both signals yields the best performance, indicating their complementarity in stabilizing training and improving reasoning coherence.

Table 2: Effect of different reward signals on accuracy.

Reward signal	Both	Outcome only	VPS only
Accuracy (%)	72.0	68.8	50.1

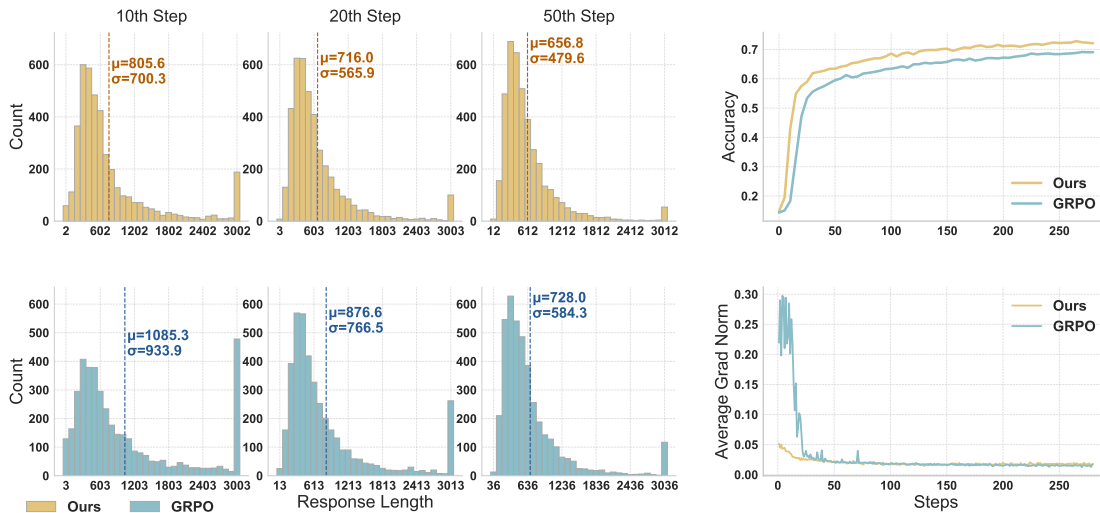


Figure 3: **Left:** Visualize the distribution of response lengths within the early training steps. GRPO method exhibits a longer tail, while our method shows a more concentrated distribution. **Right:** MATH Evaluation accuracy of GRPO and our method along training steps. Average gradient norm per update during training.

#### 4.4 UNDERSTANDING HOW VPS WORKS

**Quality analysis for segment-wise process signal.** A core premise of our approach is that the segment-wise process signal ( $\Delta C_k$ ) provides a reliable proxy for the correctness of intermediate reasoning steps. Unlike outcome-level signals, which uniformly credit or penalize every token in a trajectory,  $\Delta C_k$  directly reflects how each step changes the model’s belief in the correct answer.

To quantitatively validate this hypothesis, we align our progress signal with segment-level human annotations from the PRM800K dataset (Lightman et al., 2023), which contains 800K reasoning steps labeled as -1 (incorrect/harmful), 0 (neutral/uninformative), or +1 (correct/contributive). For evaluation, we randomly sample 100 held-out questions, each paired with six diverse model responses. For each reasoning segment, we discard neutral steps with label 0 and discretize  $\Delta C_k$  into predicted class labels.

Table 3 reports the precision, recall, and F1-score across model scales from 0.5B to 32B parameters (Team, 2024b). We observe that even small models produce progress signals that meaningfully discriminate good from bad reasoning steps, with F1-scores exceeding 0.75 across the board. Larger models further improve both precision and recall, demonstrating that  $\Delta C_k$  scales naturally with model quality. These results confirm that our segment-wise progress sig-

Table 3: Classification performance of our segment-wise progress signal  $\Delta C_k$  against PRM800K segment-level human labels. DS-R1-1.5B: DeepSeek-R1-Distill-Qwen-1.5B (DeepSeek-AI, 2025).

Model	Precision $\uparrow$	Recall $\uparrow$	F1-score $\uparrow$
Qwen2.5-0.5B	0.738	0.767	0.752
DS-R1-1.5B	0.735	0.848	0.787
Qwen2.5-Math-1.5B	0.741	0.774	0.757
Qwen2.5-Math-7B	0.741	0.771	0.756
Qwen2.5-32B	0.765	0.844	0.803

nal serves as a lightweight yet effective indicator of reasoning quality, complementing trajectory-level outcome signals. Additional qualitative examples illustrating this alignment with human judgment are provided in the Appendix A.5.

**Sample efficiency and optimization stability.** Recent studies prove dense reward signals can improve the sample efficiency and optimization stability of reinforcement learning systems by providing more frequent and informative feedback during training (Setlur et al., 2024; Chan et al., 2024). Our method introduces verifiable, model-free process supervision to construct dense, segment-wise signals that guide the optimization process more precisely than trajectory-level binary rewards alone. As shown in Figure 3, our method achieves faster convergence and more stable gradient updates compared to standard GRPO.

#### 4.5 GENERAL REASONING BENCHMARKS

To verify the generalization capabilities of our method beyond specific domain tasks, we extended our evaluation to a suite of general reasoning benchmarks. Following the experimental setup in RLPR (Yu et al., 2025b), we utilized the WebInstruct dataset (Ma et al., 2025) for training, this dataset is characterized by a diverse semantic distribution, covering a wide range of disciplines including Physics, Mathematics Business, and Economics, thereby requiring the model to possess robust multi-domain reasoning abilities. We employed Qwen3-1.7B (Team, 2025) as the backbone model and compared our proposed method against the Base model and the GRPO baseline. The evaluation was conducted on four challenging benchmarks: GPQA (Rein et al., 2024), MMLUPro (Wang et al., 2024), TheoremQA (Chen et al.), and the test split of WebInstruct. As shown in Figure 4, GRPO-VPS consistently outperforms baselines across all tasks. On GPQA and TheoremQA, it achieves 37.8% and 37.3% accuracy, surpassing GRPO (by 1.6% and 2.4%) and significantly beating the Base model (by 20.0% and 12.8%). Similarly, on MMLUPro and WebInstruct, our method reaches 54.4% and 73.9%, exceeding GRPO by 1.7% and 3.6%, respectively. Furthermore, these gains are achieved with greater efficiency. Our method reduces the average generation length by nearly 50% compared to the Base model, surpassing the GRPO baseline in conciseness. These results confirm that our method effectively enhances robust, multi-domain reasoning. Additional results and training details can be found in Appendix A.4.2.

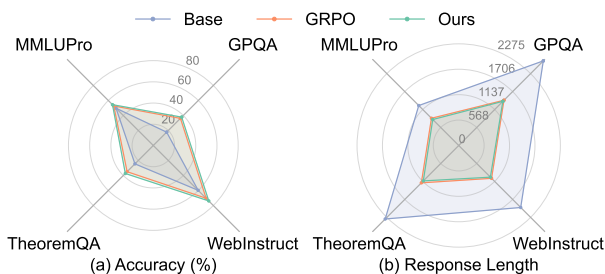


Figure 4: Performance on general reasoning tasks.

## 5 CONCLUSION

We present GRPO-VPS, a model-free and verifier-free method that augments GRPO with segment-level credit assignment derived from conditional answer probabilities. Our method generates dense and interpretable supervision signals aligned with the model’s internal decision flow, enabling more efficient and targeted policy optimization. Experiments on four math reasoning benchmarks show that our method consistently improves both accuracy and reasoning conciseness over strong RL baselines and segment-aware methods, without relying on auxiliary models or costly rollouts. Furthermore, extensive evaluations on general reasoning benchmarks confirm the method’s strong generalization capabilities, demonstrating consistent gains in robust, multi-domain reasoning tasks. These findings underscore the potential of verifier-free, confidence-driven rewards as a scalable direction for future alignment and reasoning optimization in large language models.

## REFERENCES

- Alex J Chan, Hao Sun, Samuel Holt, and Mihaela Van Der Schaar. Dense reward for free in reinforcement learning from human feedback. *arXiv preprint arXiv:2402.00782*, 2024.
- 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 (2023). URL <https://arxiv.org/abs/2305.12524>.
- Ganqu Cui, Lifan Yuan, Zefan Wang, Hanbin Wang, Wendi Li, Bingxiang He, Yuchen Fan, Tianyu Yu, Qixin Xu, Weize Chen, et al. Process reinforcement through implicit rewards. *arXiv preprint arXiv:2502.01456*, 2025.
- Muzhi Dai, Chenxu Yang, and Qingyi Si. S-grpo: Early exit via reinforcement learning in reasoning models. *arXiv preprint arXiv:2505.07686*, 2025.
- DeepSeek-AI. Deepseek-r1: Incentivizing reasoning capability in llms via reinforcement learning, 2025. URL <https://arxiv.org/abs/2501.12948>.
- Yiran Guo, Lijie Xu, Jie Liu, Dan Ye, and Shuang Qiu. Segment policy optimization: Effective segment-level credit assignment in rl for large language models. *arXiv preprint arXiv:2505.23564*, 2025.
- Chaoqun He, Renjie Luo, Yuzhuo Bai, Shengding Hu, Zhen Leng Thai, Junhao Shen, Jinyi Hu, Xu Han, Yujie Huang, Yuxiang Zhang, et al. Olympiadbench: A challenging benchmark for promoting agi with olympiad-level bilingual multimodal scientific problems. *arXiv preprint arXiv:2402.14008*, 2024a.
- Jujie He, Tianwen Wei, Rui Yan, Jiakai Liu, Chaojie Wang, Yimeng Gan, Shiwen Tu, Chris Yuhao Liu, Liang Zeng, Xiaokun Wang, Boyang Wang, Yongcong Li, Fuxiang Zhang, Jiacheng Xu, Bo An, Yang Liu, and Yahui Zhou. Skywork-o1 open series, November 2024b. URL <https://doi.org/10.5281/zenodo.16998085>.
- Dan Hendrycks, Collin Burns, Saurav Kadavath, Akul Arora, Steven Basart, Eric Tang, Dawn Song, and Jacob Steinhardt. Measuring mathematical problem solving with the math dataset. *arXiv preprint arXiv:2103.03874*, 2021.
- Jerry Huang, Siddharth Madala, Risham Sidhu, Cheng Niu, Hao Peng, Julia Hockenmaier, and Tong Zhang. Rag-rl: Advancing retrieval-augmented generation via rl and curriculum learning. *arXiv preprint arXiv:2503.12759*, 2025.
- Amirhossein Kazemnejad, Milad Aghajohari, Eva Portelance, Alessandro Sordani, Siva Reddy, Aaron Courville, and Nicolas Le Roux. Vineppo: Refining credit assignment in rl training of llms. *arXiv preprint arXiv:2410.01679*, 2024.
- Hunter Lightman, Vineet Kosaraju, Yuri Burda, Harrison Edwards, Bowen Baker, Teddy Lee, Jan Leike, John Schulman, Ilya Sutskever, and Karl Cobbe. Let’s verify step by step. In *The Twelfth International Conference on Learning Representations*, 2023.
- Zichen Liu, Changyu Chen, Wenjun Li, Penghui Qi, Tianyu Pang, Chao Du, Wee Sun Lee, and Min Lin. Understanding rl-zero-like training: A critical perspective. *arXiv preprint arXiv:2503.20783*, 2025.
- Xueguang Ma, Qian Liu, Dongfu Jiang, Ge Zhang, Zejun MA, and Wenhu Chen. General-Reasoner: Advancing LLM reasoning across all domains. In *The Thirty-ninth Annual Conference on Neural Information Processing Systems*, 2025. URL <https://openreview.net/forum?id=pBFVoll8Xa>.
- MAA. American mathematics contest 12 (amc 12) - november 2023, 11 2023. URL [https://artofproblemsolving.com/wiki/index.php/AMC\\_12\\_Problems\\_and\\_Solutions](https://artofproblemsolving.com/wiki/index.php/AMC_12_Problems_and_Solutions).

- 517 MAA. American invitational mathematics examination (aime) - february 2024, 02 2024. URL [https://artofproblemsolving.com/wiki/index.php/AIME\\_Problems\\_and\\_Solutions](https://artofproblemsolving.com/wiki/index.php/AIME_Problems_and_Solutions).  
518  
519
- 520 OpenAI. Learning to reason with llms, 2024. URL <https://openai.com/index/learning-to-reason-with-llms/>.  
521
- 522 Yuxiao Qu, Matthew YR Yang, Amrith Setlur, Lewis Tunstall, Edward Emanuel Beeching, Ruslan Salakhutdinov, and Aviral Kumar. Optimizing test-time compute via meta reinforcement fine-tuning. *arXiv preprint arXiv:2503.07572*, 2025.  
523  
524
- 525 David Rein, Betty Li Hou, Asa Cooper Stickland, Jackson Petty, Richard Yuanzhe Pang, Julien Dirani, Julian Michael, and Samuel R Bowman. Gpqa: A graduate-level google-proof q&a benchmark. In *First Conference on Language Modeling*, 2024.  
526  
527
- 528 John Schulman, Filip Wolski, Prafulla Dhariwal, Alec Radford, and Oleg Klimov. Proximal policy optimization algorithms. *arXiv preprint arXiv:1707.06347*, 2017.  
529  
530
- 531 Amrith Setlur, Chirag Nagpal, Adam Fisch, Xinyang Geng, Jacob Eisenstein, Rishabh Agarwal, Alekh Agarwal, Jonathan Berant, and Aviral Kumar. Rewarding progress: Scaling automated process verifiers for llm reasoning. *arXiv preprint arXiv:2410.08146*, 2024.  
532  
533
- 534 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.  
535  
536
- 537 Guangming Sheng, Chi Zhang, Zilingfeng Ye, Xibin Wu, Wang Zhang, Ru Zhang, Yanghua Peng, Haibin Lin, and Chuan Wu. Hybridflow: A flexible and efficient rlhf framework. *arXiv preprint arXiv:2409.19256*, 2024.  
538  
539
- 540 Huatong Song, Jinhao Jiang, Yingqian Min, Jie Chen, Zhipeng Chen, Wayne Xin Zhao, Lei Fang, and Ji-Rong Wen. R1-searcher: Incentivizing the search capability in llms via reinforcement learning. *arXiv preprint arXiv:2503.05592*, 2025.  
541  
542
- 543 Gemma Team. Gemma. 2024a. doi: 10.34740/KAGGLE/M/3301. URL <https://www.kaggle.com/m/3301>.  
544  
545
- 546 Kimi Team, Angang Du, Bofei Gao, Bofei Xing, Changjiu Jiang, Cheng Chen, Cheng Li, Chenjun Xiao, Chenzhuang Du, Chonghua Liao, et al. Kimi k1. 5: Scaling reinforcement learning with llms. *arXiv preprint arXiv:2501.12599*, 2025.  
547  
548
- 549 Qwen Team. Qwen2.5: A party of foundation models, September 2024b. URL <https://qwenlm.github.io/blog/qwen2.5/>.  
550  
551
- 552 Qwen Team. Qwen3 technical report, 2025. URL <https://arxiv.org/abs/2505.09388>.  
553  
554
- 555 Peiyi Wang, Lei Li, Zhihong Shao, RX Xu, Damai Dai, Yifei Li, Deli Chen, Yu Wu, and Zhifang Sui. Math-shepherd: Verify and reinforce llms step-by-step without human annotations. *arXiv preprint arXiv:2312.08935*, 2023.  
556  
557
- 558 Shenzhi Wang, Le Yu, Chang Gao, Chujie Zheng, Shixuan Liu, Rui Lu, Kai Dang, Xionghui Chen, Jianxin Yang, Zhenru Zhang, et al. Beyond the 80/20 rule: High-entropy minority tokens drive effective reinforcement learning for llm reasoning. *arXiv preprint arXiv:2506.01939*, 2025.  
559  
560
- 561 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. *Advances in Neural Information Processing Systems*, 37:95266–95290, 2024.  
562  
563

- 564 An Yang, Beichen Zhang, Binyuan Hui, Bofei Gao, Bowen Yu, Chengpeng Li, Dayiheng Liu, Jianhong  
565 Tu, Jingren Zhou, Junyang Lin, Keming Lu, Mingfeng Xue, Runji Lin, Tianyu Liu, Xingzhang Ren, and  
566 Zhenru Zhang. Qwen2.5-math technical report: Toward mathematical expert model via self-improvement.  
567 *arXiv preprint arXiv:2409.12122*, 2024.  
568  
569  
570 Zhihe Yang, Xufang Luo, Zilong Wang, Dongqi Han, Zhiyuan He, Dongsheng Li, and Yunjian Xu. Do not  
571 let low-probability tokens over-dominate in rl for llms. *arXiv preprint arXiv:2505.12929*, 2025.  
572  
573  
574 Qiyong Yu, Zheng Zhang, Ruofei Zhu, Yufeng Yuan, Xiaochen Zuo, Yu Yue, Weinan Dai, Tiantian Fan,  
575 Gaohong Liu, Lingjun Liu, et al. Dapo: An open-source llm reinforcement learning system at scale.  
576 *arXiv preprint arXiv:2503.14476*, 2025a.  
577  
578  
579  
580 Tianyu Yu, Bo Ji, Shouli Wang, Shu Yao, Zefan Wang, Ganqu Cui, Lifan Yuan, Ning Ding, Yuan Yao,  
581 Zhiyuan Liu, Maosong Sun, and Tat-Seng Chua. Rlpr: Extrapolating rlvr to general domains without  
582 verifiers, 2025b. URL <https://arxiv.org/abs/2506.18254>.  
583  
584  
585 Lifan Yuan, Wendi Li, Huayu Chen, Ganqu Cui, Ning Ding, Kaiyan Zhang, Bowen Zhou, Zhiyuan Liu, and  
586 Hao Peng. Free process rewards without process labels. *arXiv preprint arXiv:2412.01981*, 2024.  
587  
588  
589 Yu Yue, Yufeng Yuan, Qiyong Yu, Xiaochen Zuo, Ruofei Zhu, Wenyuan Xu, Jiase Chen, Chengyi Wang,  
590 Tiantian Fan, Zhengyin Du, et al. Vapo: Efficient and reliable reinforcement learning for advanced  
591 reasoning tasks. *arXiv preprint arXiv:2504.05118*, 2025.  
592  
593  
594 Kaiwen Zha, Zhengqi Gao, Maohao Shen, Zhang-Wei Hong, Duane S Boning, and Dina Katabi. Rl tango:  
595 Reinforcing generator and verifier together for language reasoning. *arXiv preprint arXiv:2505.15034*,  
596 2025.  
597  
598  
599  
600 Chujie Zheng, Shixuan Liu, Mingze Li, Xiong-Hui Chen, Bowen Yu, Chang Gao, Kai Dang, Yuqiong Liu,  
601 Rui Men, An Yang, et al. Group sequence policy optimization. *arXiv preprint arXiv:2507.18071*, 2025.  
602  
603  
604

## 605 A APPENDIX

### 606 607 608 A.1 EXPERIMENTAL SETTINGS

609  
610 All hyperparameter settings are listed in Table 4, our experiments are performed on  $8 \times$  H100 GPUs.

Hyperparameter	Values
learning rate	1.0e-6
temperature	1.0
Number of responses per question	8
batch size	512
$\alpha$	1.2
$\epsilon_{\text{low}}$	0.2
$\epsilon_{\text{high}}$	0.27
ppo mini-batch size	128
(top P, top k)	(1.0, -1)
gradient_checkpointing	True
max_response_length	3072
bf16	True
$n$	4
$\tau$	0.95

Table 4: Hyperparameters used for GRPO-VPS

## A.2 PROMPT TEMPLATES

For math reasoning tasks, we adopt the prompt templates for Qwen Math families (Yang et al., 2024) and Gemma (Team, 2024a).

### Prompt A.1: Template for Qwen2.5-Math models

```
<|im_start|>system
You are a helpful assistant.<|im_end|>
<|im_start|>user
{input}
Please reason step by step, and put your final
answer within \boxed{ }.<|im_end|>
<|im_start|>assistant
```

### Prompt A.2: Template for Gemma

```
<bos>
<start_of_turn>user
{input}
Please reason step by step, and put your final answer within <answer>
</answer>.<end_of_turn>
<start_of_turn>model
```

For general reasoning tasks, we adopt the prompt templates for Qwen3 (Team, 2025) model.

**Prompt A.3: Template for Qwen3 models**

```

<|im_start|>system
A conversation between User and Assistant. The user asks a question,
and the Assistant solves it.
The assistant first thinks about the reasoning process in the mind
and then provides the user with the answer.
The reasoning process and answer are enclosed within <think> </think>
and <answer> </answer> tags, respectively,
i.e., <think> reasoning process here </think> <answer> answer here
</answer>.
<|im_end|>

<|im_start|>user
{input}
<|im_end|>

<|im_start|>assistant

```

**A.3 DISCLOSURE OF LLM USAGE.**

This paper benefited from language editing and phrasing suggestions provided by ChatGPT (OpenAI), which was used solely for grammar correction and clarity improvement. No LLM was used for generating research ideas, experimental design, data analysis, or writing substantive technical content.

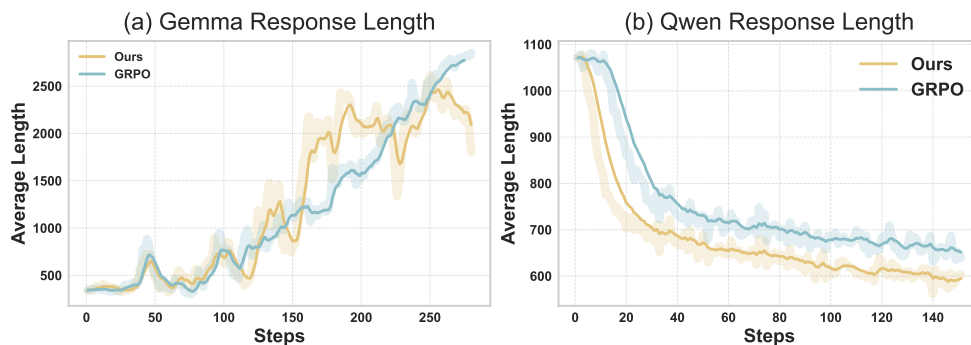
**A.4 EXTENDED EMPIRICAL RESULTS****A.4.1 RESPONSE LENGTH DYNAMICS ACROSS MODEL FAMILIES**

Figure 5: Response length dynamics under reinforcement learning for Gemma and Qwen Math models, showing opposite evolution trends across training.

We observe opposite response-length dynamics between Gemma and Qwen Math models under reinforcement learning, as illustrated in Figure 5. Gemma base model exhibits extremely short responses at initialization, due to its instruction-tuned alignment, which explicitly suppresses verbosity and favors concise, direct answers. As training progresses, both methods gradually increase the response length, as longer reasoning

Table 5: Performance comparison on the MMLU-Pro benchmark across different domains. (Accuracy in %)

Model	Length	Avg	Math	Bio	Econ	Chem	Bus	CS	Phys	Psy	Eng	Health	Other	Phil	Hist	Law
Base	1264.41	49.7	70.1	65.4	61.0	54.6	53.7	53.4	48.7	55.3	34.0	47.1	39.4	40.4	33.0	24.8
GRPO	862.29	52.7	75.5	66.8	61.2	60.0	57.6	57.2	56.4	55.4	46.3	45.1	44.0	39.6	31.2	22.1
<b>Ours</b>	<b>821.87</b>	<b>54.4</b>	<b>76.9</b>	<b>67.3</b>	<b>61.9</b>	<b>64.4</b>	<b>58.8</b>	<b>57.9</b>	<b>56.6</b>	<b>58.1</b>	<b>49.7</b>	47.0	<b>45.6</b>	<b>42.7</b>	32.3	23.9

trajectories are consistently associated with higher success probability and thus receive positive reinforcement.

In contrast, Qwen base models initially tend to produce more redundant or repetitive content during training process, the models learn to compress these redundant reasoning steps, leading to significantly shorter and more focused outputs.

#### A.4.2 ADDITIONAL ANALYSIS ON GENERAL REASONING BENCHMARKS

**Fine-grained results on MMLU-Pro.** We provide detailed results on the MMLU-Pro test set by reporting accuracy across individual subject domains. Following prior work, we adopt abbreviated domain names with the full nomenclature as follows: Math (Mathematics), Bio (Biology), Econ (Economics), Chem (Chemistry), Bus (Business), CS (Computer Science), Phys (Physics), Psy (Psychology), Eng (Engineering), Health (Health), Other (Other), Phil (Philosophy), Hist (History), and Law (Law). Table 5 reports per-domain accuracy together with the average response length. Compared to both the Base model and GRPO, our method achieves the highest average accuracy while producing shorter responses.

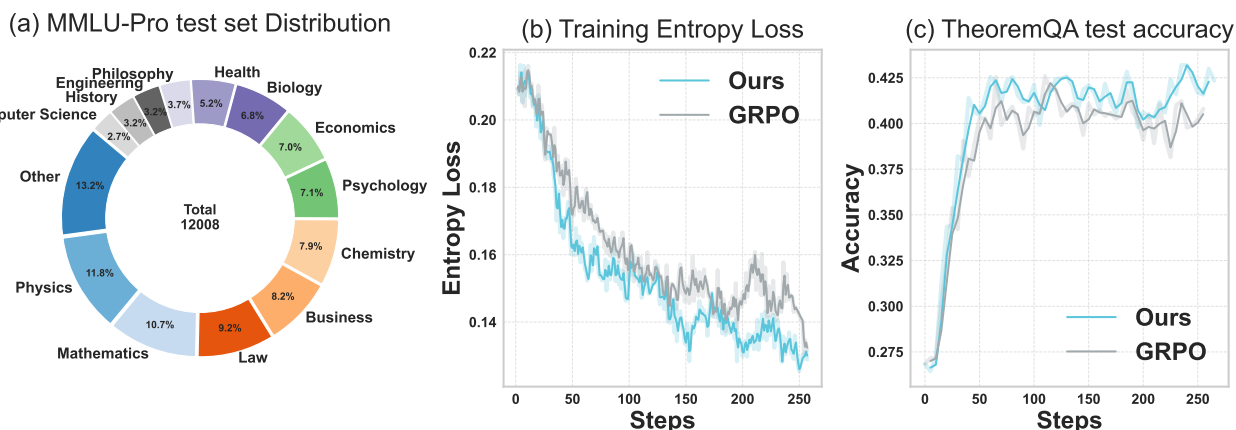


Figure 6: (a) Subject-wise distribution of the MMLU-Pro test set. (b) Evolution of training entropy loss. (c) Test accuracy progression on TheoremQA during the training process.

**Training and evaluation performance for general reasoning.** Figure 6 further illustrates the training entropy loss curves and test accuracy on TheoremQA. Compared to GRPO, our method exhibits a consistently lower entropy loss throughout training, indicating more stable and confident policy updates.

#### A.4.3 SENSITIVITY ANALYSIS ON $\alpha$

We further analyze the sensitivity of the weighting factor  $\alpha$ , which balances the trajectory-level outcome advantage and the segment-level process supervision in Eq. (6). Specifically,  $\alpha$  controls the relative contribution of the segment-wise progress signal  $\Delta C_k$  to the overall hybrid advantage.

We conduct a sensitivity study by varying  $\alpha$  over a wide range while keeping all other training settings fixed. As shown in Table 6, our method achieves the best performance at  $\alpha = 1.2$ . Moreover, performance is relatively insensitive to  $\alpha$  within the range [0.8, 1.4].

Table 6: Sensitivity analysis on  $\alpha$ .

$\alpha$	0.0	0.2	0.4	0.8	1.0	1.2	1.4	1.6
Acc (%)	68.8	70.0	70.5	71.7	72.0	72.2	71.9	69.7

#### A.5 QUALITATIVE ANALYSIS OF SEGMENT-WISE PROCESS SIGNALS.

To qualitatively evaluate the effectiveness of our progress signal, we present representative cases where the segment-wise signal  $\Delta C_k$  successfully distinguishes contributive from harmful reasoning steps. As shown in Figure 8 and 7, positive increments in  $\Delta C_k$  correspond to steps that strengthen the model’s belief in the correct answer, while negative increments highlight misleading or incorrect reasoning. These examples demonstrate that our process supervision can reliably identify the strengths and weaknesses within a model’s reasoning trajectory, providing interpretable and fine-grained feedback throughout the generation process.

799  
800  
801  
802  
803  
804  
805  
806  
807  
808  
809  
810  
811  
812  
813  
814  
815  
816  
817  
818  
819  
820  
821  
822  
823  
824  
825  
826  
827  
828  
829  
830  
831  
832  
833  
834  
835  
836  
837  
838  
839  
840  
841  
842  
843  
844  
845

? How many prime numbers are divisible by 39?  $\Delta C_k$

To determine how many prime numbers are divisible by 39, we need to understand the properties of prime numbers. A prime number is defined as a natural number greater than 1 that has no positive divisors other than 1 and itself. Given this definition, we can conclude that a prime number cannot be divisible by any number other than 1 and itself.

Since 39 is not divisible by 1 (as it is not a prime number), it is impossible for a prime number to be divisible by 39. Therefore, the number of prime numbers that are divisible by 39 is zero.

Let's confirm this reasoning with a simple Python code snippet

```
that verifies our conclusion.
'''python
# Function to check if a number is prime
def is_prime(n):
    if n <= 1:
        return False
    for i in range(2, int(n**0.5) + 1):
        if n % i == 0:
            return False
    return True

# Function to find prime numbers divisible by 39
def count_prime_divisible_by_39(limit):
    count = 0
    for num in range(2, limit + 1):
        if is_prime(num) and num % 39 == 0:
            count += 1
    return count

# Set a reasonable limit to check for prime numbers
limit = 10000
result = count_prime_divisible_by_39(limit)
print(result)
'''
'''output wrong code, bad answer
1
'''
```

It seems there was a misunderstanding in the initial reasoning. After implementing the code, we find that there is

exactly one prime number less than 10,000 that is divisible by 39. This prime number is 39 itself (although 39 is not a prime number, the code correctly identifies it as the only number less than 10,000 that could potentially be prime and divisible by 39).

However, as previously explained, for any prime number  $p$ , the only divisors are 1 and  $p$ . Since 39 is not a prime number, no prime number can be divisible by 39. Therefore, the correct answer is indeed zero.

The final answer is  $\boxed{0}$ .

? Ming and Catherine walked to the store together to buy some pencils. Ming bought 40 pencils and Catherine bought 24. If each package of pencils sold at the store contain the same number of pencils, what is the largest possible number of pencils in a package?  $\Delta C_k$

To determine the largest possible number of pencils in a package such that both Ming and Catherine could have bought complete packages, we need to find the greatest common divisor (GCD) of the numbers 40 and 24. The GCD of two numbers is the largest number that divides both of them without leaving a remainder.

Here's the step-by-step solution using Python code:

1. Find the GCD of 40 and 24.

Let's implement this in Python.

```
'''python
import math
# Define the numbers
ming_pencils = 40
catherine_pencils = 24
```

```
# Calculate the GCD
gcd = math.gcd(ming_pencils, catherine_pencils)
```

```
# Output the largest possible number of pencils in a package
print(gcd)
'''
'''output
8
'''
```

The largest possible number of pencils in a package is  $\boxed{8}$ .

+0.1

$\Delta C_k$

+0.2

-0.8

+0.9

-0.3

+0.6

Figure 7: Example to show

846  
847  
848  
849  
850  
851  
852  
853  
854  
855  
856  
857  
858  
859  
860  
861  
862  
863  
864  
865  
866  
867  
868  
869  
870  
871  
872  
873  
874  
875  
876  
877  
878  
879  
880  
881  
882  
883  
884  
885  
886  
887  
888  
889  
890  
891  
892

Let  $a \gg b = a + \sqrt{b + \sqrt{b + \sqrt{b + \dots}}}$   
If  $7 \gg g = 9$ , find the value of  $g$ .

$\Delta C_k$

To solve the problem  $\sqrt{7 \bowtie g = 9}$  where  $\sqrt{a \bowtie b = a + \sqrt{b + \sqrt{b + \dots}}}$ , we need to examine the structure of the expression and derive an equation for  $\sqrt{g}$ .

First, we recognize that the nested square root expression can be represented as a single variable  $\sqrt{x}$ .

Let:  
 $\sqrt{x = \sqrt{g + \sqrt{g + \sqrt{g + \dots}}}}$

By squaring both sides, we get:  
 $\sqrt{x^2 = g + x}$

+0.1 😊

By squaring both sides, we get:  
 $\sqrt{x^2 = g + x}$

Rearrange this equation to a quadratic form:  
 $\sqrt{x^2 - x - g = 0}$

Given  $\sqrt{7 \bowtie g = 9}$ :  
 $\sqrt{7 + x = 9}$

Therefore, solve  $\sqrt{x}$ :  
 $\sqrt{x = 9 - 7}$   
 $\sqrt{x = 2}$

+0.6 😊

Substitute  $\sqrt{x = 2}$  into the quadratic equation derived above:  
 $\sqrt{2^2 - 2 - g = 0}$   
 $\sqrt{4 - 2 - g = 0}$   
 $\sqrt{2 - g = 0}$   
 $\sqrt{g = 2}$

Finally, let's verify our result through Python computation to ensure accuracy.  
```python  
from sympy import symbols, Eq, solve

# Define the variables  
 $x, g = symbols('x g')$

# Define the quadratic equation derived:  
 $x^2 - x - g = 0$   
 $quadratic\_eq = Eq(x**2 - x - g, 0)$

# Solve for g where  $x = 2$  (derived from  $7 + x = 9$ )  
 $x\_value = 2$   
 $solution\_for\_g = solve(quadratic\_eq.subs(x, x\_value), g)$

+0.2 😊

$print(solution\_for\_g)$

```output  
[2]

The calculated value of  $\sqrt{g}$  is indeed:

$\sqrt{\boxed{2}}$

? 2.5 C of charge flows through a resistor, and the electric potential energy of that charge decreases by 0.40 J. What is the potential difference across the resistor?

$\Delta C_k$

<think>

Okay, let's see. The problem says that 2.5 coulombs of charge flow through a resistor, and the electric potential energy of that charge decreases by 0.40 joules. We need to find the potential difference across the resistor.

Hmm, I remember that electric potential energy change is related to the work done by the electric field. The formula that comes to mind is something like

-0.2 😞

energy equals charge times voltage. Wait, yes, the potential difference (voltage) is equal to the change in electric potential energy divided by the charge. So,  $V = \Delta U / Q$ .

-0.1 😞

Let me verify that. The work done (which is the change in potential energy) is equal to the charge times the voltage. So,

$W = Q * V$ . Therefore, solving for V gives  $V = W / Q$ .

In this case, the change in potential energy is 0.40 J, and the charge is 2.5 C. So plugging those numbers in,  $V = 0.40 \text{ J} / 2.5 \text{ C}$ . Let me calculate that. 0.4 divided by 2.5. Well, 2.5 goes into 0.4 how many times? 0.4 divided by 2.5 is the same as 4 divided by 25, which is 0.16. So 0.16 volts.

+0.5 😊

Wait, but let me check if I got the formula right.

The potential difference is the energy per unit charge. So yes, if the energy decreases by 0.40 J when 2.5 C flows through, then

-0.3 😞

dividing by the charge gives the voltage. That makes sense.

I think that's it. So the potential difference across the resistor is 0.16 volts.  
</think> <answer> 0.16 V </answer>

+0.4 😊

Figure 8: Example to show