

Step-GRPO: Internalizing Dynamic Early Exit for Efficient Reasoning

Benteng Chen^{1,2,*}, Weida Wang^{1,2,3,*,&}, Shufei Zhang^{2,†}, Mingbao Lin⁴, Min Zhang^{1,†,‡}
¹East China Normal University, ²Shanghai AI Laboratory, ³Fudan University, ⁴Rakuten Singapore

*Equal contribution &Student project leader †Project leader

‡Corresponding authors: mzhang@cs.ecnu.edu.cn, zhangshufei@pjlab.org.cn

Abstract

Large reasoning models that use long chain-of-thought excel at problem-solving yet waste compute on redundant checks. Curbing this overthinking is hard: training-time length penalties can cripple ability, while inference-time early-exit adds system overhead. To bridge this gap, we propose *Step-GRPO*, a novel post-training framework that internalizes dynamic early-exit capabilities directly into the model. *Step-GRPO* shifts the optimization objective from raw tokens to semantic steps by utilizing linguistic markers to structure reasoning. We introduce a Dynamic Truncated Rollout mechanism that exposes the model to concise high-confidence trajectories during exploration, synergized with a Step-Aware Relative Reward that dynamically penalizes redundancy based on group-level baselines. Extensive experiments across three model sizes on diverse benchmarks demonstrate that *Step-GRPO* achieves a superior accuracy-efficiency trade-off. On Qwen3-8B, our method reduces token consumption by 32.0% compared to the vanilla model while avoiding the accuracy degradation observed in traditional length-penalty methods.

1 Introduction

Large reasoning models (LRMs), like DeepSeek-R1 (Guo et al., 2025) and Qwen3 (Yang et al., 2025a), have well solved complex problems using long chain-of-thought (CoT). However, this reasoning ability comes at a high computational cost. A common problem is “overthinking” (Yang et al., 2025b; Dai et al., 2025a), models often generate unnecessary verification steps or circular explanations even after finding the correct solution. This happens because standard reinforcement learning methods, like GRPO (Shao et al., 2024), primarily reward correct outcomes. Since longer reasoning chains statistically increase the chance of getting the right answer, the model naturally learns to be

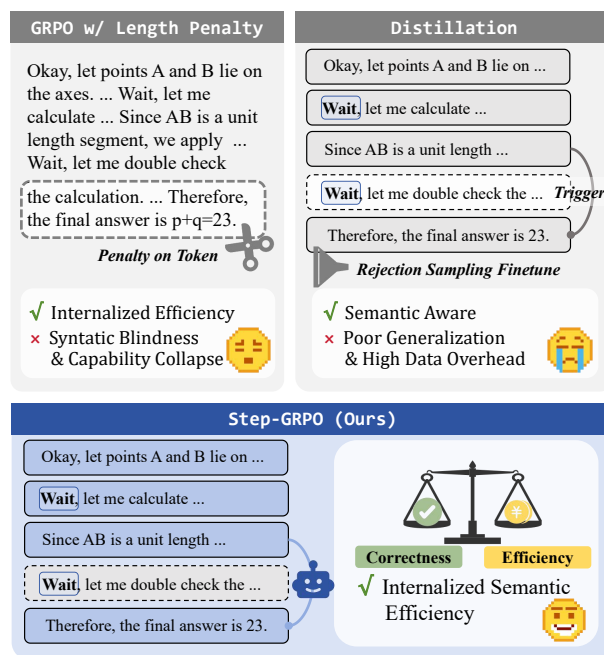


Figure 1: Advantages of our Step-GRPO.

verbose, leading to wasted computation and higher latency.

To address this inefficiency, recent research has focused on constraining the generation length during post-training (Yu et al., 2025; Dai et al., 2025a; Team et al., 2025). Methods like GRPO with length penalty (Team et al., 2025; Yu et al., 2025) or dynamic adjustments like GRPO- λ (Dai et al., 2025a) explicitly add penalty terms to the reward function.

However, as illustrated in Figure 1 (Top Left), these approaches suffer from “syntactic blindness”. Since they rely on indiscriminate token counting, models cannot distinguish between redundancy and necessary reasoning. Consequently, penalizing tokens often forces the model to cut essential verification steps, leading to *capability collapse* where brevity is achieved at the cost of accuracy (Gao et al., 2023; Su and Cardie, 2025).

Another direction attempts to internalize efficient inference strategies (Wang et al., 2025; Yang et al., 2025b), which rely on lexical cues (Vanho-

weghen et al., 2025) or entropy (Agarwal et al., 2025) to optimize generation, via distillation. Recent works apply supervised fine-tuning (SFT) on compressed trajectories (Qiao et al., 2025; Zhang et al., 2025) derived from rejection sampling. While *semantic-aware*, this approach faces severe limitations shown in Figure 1 (Top Right): constructing valid concise samples incurs *high data overhead* due to expensive rejection sampling. Critically, SFT yields *poor generalization*, the model superficially mimics the concise style of the training data without learning the underlying decision-making policy, often failing on complex unseen tasks.

To bridge this gap, we propose Step-GRPO, a novel post-training framework that *internalizes the efficient reasoning capability* directly into the model itself (Figure 1, Bottom). Unlike previous methods that focus on tokens, Step-GRPO addresses the problem through *semantic steps*. First, we solve the syntactic blindness of previous training methods by moving from tokens to *semantic steps*. We use linguistic markers to segment reasoning and apply a *step-aware reward*. Second, we simulate the decision-making process of inference interventions during training rollouts. By mixing natural trajectories with teacher-guided truncated paths, we force the model to learn that fewer reasoning steps yield higher rewards. This effectively transfers the external stopping capability into the model itself, achieving semantically efficient reasoning with zero inference overhead. Experimental results show that Step-GRPO achieves a superior trade-off between accuracy and efficiency, significantly reducing tokens while avoiding the training collapse in length-penalty methods.

2 Related Work

2.1 Inference-Time Efficiency Interventions

With the utilization of test-time scaling (Snell et al., 2024) to enhance complex tasks, LRMs (Kojima et al., 2022; Jaech et al., 2024; Guo et al., 2025) generate long CoT (Lightman et al., 2023) sequences, leading to increased computational load and inference latency. Research indicates an intrinsic “overthinking” phenomenon in LRMs, where models persistently generate verbose reasoning sequences with redundant steps, wasting resources and potentially degrading accuracy (Chen et al.; Saito et al., 2023). To address this, existing work is two-fold: *prompt-based* and *output-based*. Prompt-

based methods apply external constraints, such as NoThinking strategy (Ma et al., 2025a) that skips reasoning, Chain-of-Draft (CoD) (Xu et al., 2025) which limits word counts, and Token-Conditional Control (TCC) (Muennighoff et al., 2025) that sets token budgets. However, these static constraints often compromise reasoning capabilities or are ignored in complex tasks. Conversely, output-based methods aim for dynamic early exit. For example, Dynasor-CoT (Fu et al., 2025) periodically checks answer consistency but often suffers from late termination. While methods like DEER (Yang et al., 2025b) achieve precise truncation, strategies requiring parallel decoding or branch evaluation often increase overhead, potentially negating efficiency gains.

2.2 Training-Time Alignment for Conciseness

To mitigate deployment latency during inference-time, research has internalized concise reasoning capabilities directly into model weights. SFT on compressed CoT data is a primary strategy (Li et al., 2025; Ma et al., 2025b; Cui et al., 2025; Xia et al., 2025). While approaches like ConCISE (Ma et al., 2025b; Qiao et al., 2025; Xia et al., 2025; Cui et al., 2025) improve efficiency via mixed training or confidence-guided data filtration, they face significant bottlenecks in constructing high-quality, logically complete yet concise samples. Alternatively, Reinforcement Learning (RL) targets efficiency via length penalties, yet often risks “corner cutting” and capability collapse (Dai et al., 2025a). While recent innovations like GRPO- λ (Dai et al., 2025a), entropy-guided compression (Zhu et al., 2025), and S-GRPO (Dai et al., 2025b) improve stability through dynamic adjustments or decaying rewards, they remain reliant on statistical heuristics (e.g., entropy, sequence length) that ignore reasoning semantics. Distinct from these scalar-based approaches, we integrate semantic stopping directly into the GRPO loop. By treating linguistic markers (e.g., “Wait”) as decision anchors, we reward the model for logical closure at valid transition points, ensuring compression aligns with the natural structure of reasoning rather than blind truncation.

3 Method

In this section, we propose Step-GRPO, a post-training framework designed to internalize the dynamic early-exit capability into the reasoning model itself. As illustrated in Figure 2, Step-GRPO

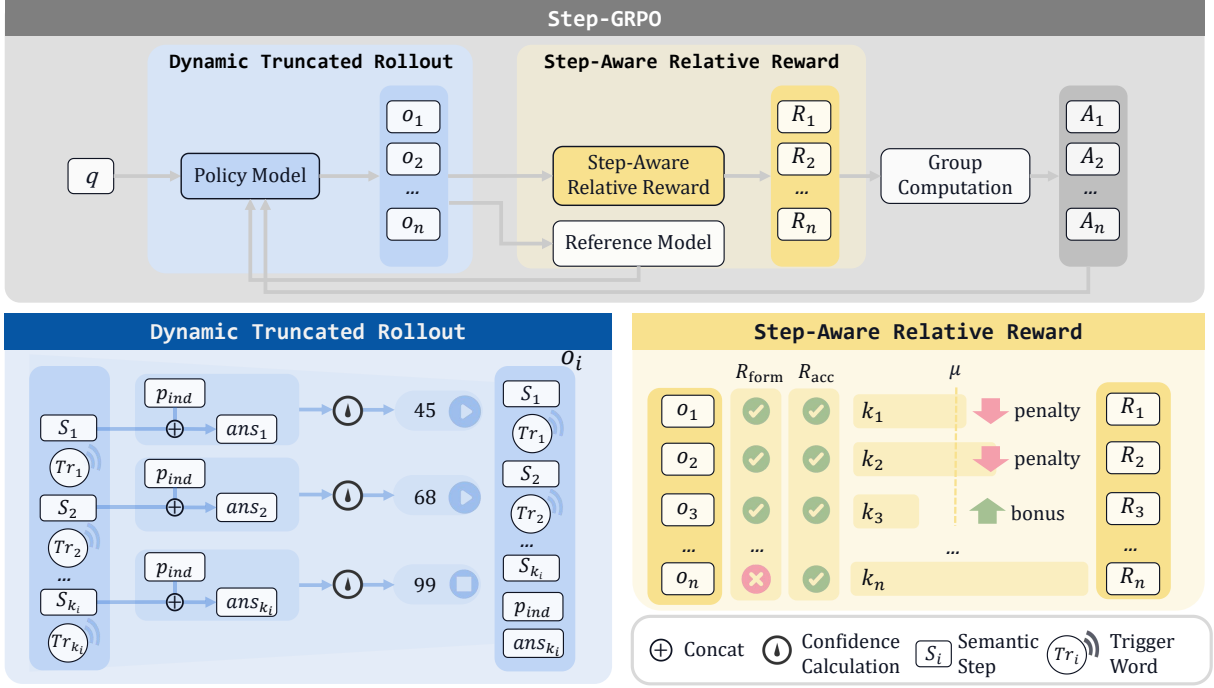


Figure 2: The overall pipeline of Step-GRPO.

consists of three integral components: *Dynamic Truncated Rollout* during exploration (Section 3.2), *Semantic Step Quantification* (Section 3.3), and *Step-Aware Relative Reward* (Section 3.4) for policy optimization.

3.1 Preliminary

We consider a reasoning task where the input is a question q and the ground truth is y^* . The policy model π_θ , parameterized by θ , generates a completion o_i , which consists of both the CoT reasoning path and the final answer. Following the GRPO framework (Shao et al., 2024), for each question q , we sample a group of G outputs $\mathcal{G} = \{o_1, o_2, \dots, o_G\}$ from the old policy $\pi_{\theta_{\text{old}}}$. The objective is to maximize a reward $J(\theta)$:

$$J(\theta) = \mathbb{E}_{q \sim \mathcal{D}, o \sim \pi_\theta(\cdot|q)} [R(o, q)], \quad (1)$$

where $R(o, q)$ is a reward function balancing correctness and reasoning efficiency.

3.2 Dynamic Truncated Rollout

To enable the model to learn efficient reasoning paths, we must expose it to *short yet correct* trajectories during training. Instead of standard autoregressive generation, we enforce a *Dynamic Truncated Rollout* mechanism for all samples in \mathcal{G} , inspired by inference-time early-exit strategies (Yang et al., 2025b; Wang et al., 2025). For each completion o_i , we utilize a set of transition *trigger words*

$\mathcal{W}_{\text{trig}}$ (e.g., “Wait”, “Alternatively”). Following Qiao et al. (2025), we define these triggers as signals for boundaries between *semantic reasoning steps*. The generation proceeds iteratively as:

1) Semantic Step Detection We monitor the generation process continuously. Once a trigger word $Tr \in \mathcal{W}_{\text{trig}}$ is detected at the end of the current sequence, we define the content generated since the last trigger as a semantic step S . We then pause the standard generation to evaluate the necessity of further reasoning.

2) Answer Induction We construct a temporary input context by appending an answer-inducing prompt p_{ind} (e.g., “</think> The final answer is”) to the current history. The model then generates a tentative answer ans .

3) Confidence Calculation We evaluate the confidence $\text{conf}(ans)$ (we use $c(a)$ for short) of the tentative answer, defined as the average log-probability of its tokens:

$$c(a) = \frac{1}{|a|} \sum_{j=1}^{|a|} \log \pi_{\theta_{\text{old}}}(a_j | q, o_{i,<t}, p_{\text{ind}}, a_{<j}). \quad (2)$$

4) Truncation Decision We compare the confidence against a threshold δ .

- If $c(ans) > \delta$, the model is deemed confident. We terminate the reasoning process and define the final completion o_i as the concatenation of the current path (including the current trigger Tr) and the induced answer ans .
- If $c(ans) \leq \delta$, the tentative answer ans is discarded. The model resumes generating the next semantic step until the next trigger word or the end of generation.

This process ensures that the sampled group \mathcal{G} contains diverse reasoning paths that are potentially truncated at the moment of sufficient confidence.

3.3 Semantic Step Quantification

Traditional efficiency metrics relying on raw token counts are often sensitive to phrasing verbosity. We instead evaluate computational cost through *Semantic Steps*.

We quantify the reasoning complexity k_i for the i -th completion o_i by tallying the occurrences of trigger words. Formally, let $N_{\text{trig}}(o_i)$ denote the total count of any trigger word $Tr \in \mathcal{W}_{\text{trig}}$ detected within the completion o_i . The semantic step count is defined as:

$$k_i = 1 + N_{\text{trig}}(o_i), \quad (3)$$

where the initial term 1 accounts for the final reasoning segment (containing the answer) that typically follows the last trigger. This quantification aligns with the discrete decision points in the dynamic rollout, providing a robust, structure-aware metric for reasoning complexity.

3.4 Step-Aware Relative Reward

We propose a *Step-Aware Relative Reward* to guide the policy optimization. Unlike static length penalties, our reward mechanism employs a dynamic reference derived from the group’s performance.

For each completion $o_i \in \mathcal{G}$, the total reward R_i is computed based on its correctness and its relative step efficiency.

Dynamic Step Baseline We first calculate the dynamic average μ , defined as the mean step count of all *correct* completions in the current group $\mathcal{G}_{\text{correct}}$:

$$\mu = \frac{1}{|\mathcal{G}_{\text{correct}}|} \sum_{j: o_j \in \mathcal{G}_{\text{correct}}} k_j. \quad (4)$$

We exclude incorrect samples to prevent baseline skewing, as they often exhibit extreme step counts

(e.g., premature guessing or circular hallucinations). Including such outliers would distort μ , leading to misaligned efficiency incentives. If the group contains no correct answers, we omit efficiency term.

Final Reward Function The total reward R_i serves as a composite objective, balancing solution correctness, reasoning efficiency, and structural compliance. It combines an accuracy indicator $R_{\text{acc}}^{(i)}$, a step-aware efficiency term, and a format consistency reward $R_{\text{form}}^{(i)}$:

$$R_i = \alpha \cdot R_{\text{acc}}^{(i)} \cdot \left[1 - \beta \cdot \tanh \left(\frac{k_i - \mu}{\mu} \right) \right] + (1 - \alpha) \cdot R_{\text{form}}^{(i)}, \quad (5)$$

where $\alpha \in [0, 1]$ balances accuracy against formatting constraints, and $\beta > 0$ controls the penalty strength. The term $\left(\frac{k_i - \mu}{\mu}\right)$ represents the relative deviation from the group mean. By applying the hyperbolic tangent function (\tanh), we bound the *step efficiency incentive* to the range $(-\beta, \beta)$. This formulation prevents extreme variance in rewards while encouraging the model to seek the *minimal sufficient reasoning path*. Specifically, if a correct response uses fewer steps than average ($k_i < \mu$), the penalty term becomes a bonus (positive), increasing R_i above α ; otherwise, it acts as a penalty.

3.5 Policy Optimization

Following the GRPO framework (Shao et al., 2024), we optimize the policy θ by maximizing the expected advantage over the generated tokens. First, we compute the advantage A_i for each completion o_i by standardizing the rewards within the group:

$$A_i = \frac{R_i - \text{mean}(\{R_1, \dots, R_G\})}{\text{std}(\{R_1, \dots, R_G\})}. \quad (6)$$

The final objective function is defined as the average per-token importance-weighted advantage, constrained by a KL divergence term. To maintain readability, we use $\pi(\cdot)$ to denote the conditional dependence $\pi(o_{i,t}|q, o_{i,<t})$:

$$J_{\text{GRPO}}(\theta) = \mathbb{E} \left[\frac{1}{G} \sum_{i=1}^G \sum_{t=1}^{|o_i|} \min \left(\frac{\pi_{\theta}(\cdot)}{\pi_{\theta_{\text{old}}}(\cdot)} A_i, \text{clip} \left(\frac{\pi_{\theta}(\cdot)}{\pi_{\theta_{\text{old}}}(\cdot)}, 1 - \epsilon, 1 + \epsilon \right) A_i \right) - \beta_{\text{KL}} D_{\text{KL}}(\pi_{\theta} \parallel \pi_{\text{ref}}) \right]. \quad (7)$$

where ϵ is the clipping parameter used in Proximal Policy Optimization (PPO) to limit policy updates (Schulman et al., 2017).

4 Experiment

4.1 Experimental Setup

Benchmarks To comprehensively evaluate the accuracy-efficiency trade-off, we conduct experiments across diverse benchmarks spanning varying difficulty levels. We utilize GSM8K (Cobbe et al., 2021) and MATH-500 (Hendrycks et al., 2021) for standard multi-step reasoning tasks. To assess performance on complex problems requiring extensive reasoning chains, we include AMC 2023 (AI-MO, 2024), AIME 2024 (Zhang and Math-AI, 2024), and AIME 2025 (Zhang and Math-AI, 2025). Additionally, we test domain-specific expert reasoning using GPQA (Rein et al., 2024) (Diamond subset).

Baselines We compare our method against a comprehensive set of baselines to evaluate both reasoning accuracy and generation efficiency. *Vanilla* denotes the original base model without additional reinforcement learning. For standard RL comparisons, *GRPO* represents the original Group Relative Policy Optimization algorithm trained purely on correctness rewards. To benchmark efficiency-oriented strategies, we include *GRPO-8k*, which imposes a hard truncation at 8,192 tokens during training to simulate resource-constrained generation. We further compare against three state-of-the-art penalty-based methods: *GRPO+LP*, which adopts the length penalty mechanism implemented in Kimi 1.5 (Team et al., 2025); *GRPO+SOP*, which utilizes the Soft Overlong Punishment introduced in DAPO (Yu et al., 2025); and *GRPO- λ* (Dai et al., 2025a), which dynamically applies length penalties based on the correctness ratio of the rollout group. Finally, to assess the necessity of reinforcement learning over supervised distillation, we include *DEER+SFT*, where the model is finetuned on concise and correct reasoning chains collected via DEER (Yang et al., 2025b)-guided rejection sampling.

Implementation Details Our implementation is built upon the EasyR1 training framework (Yaowei Zheng, 2025). For training, we utilize data selected from the DAPO-Math-17k dataset (Yu et al., 2025). We define the transition trigger tokens \mathcal{W}_{trig} following Qiao et al. (2025), including terms such as “Wait” and “Alternatively”.

Regarding hyperparameters, we set the reward balancing coefficient $\alpha = 0.1$ step penalty strength $\beta = 0.5$ and the rollout group size $G = 5$. During the dynamic truncated rollout, the confidence threshold δ is set to 0.95. For all RL experiments, we set the global batch size to 512, use a constant learning rate of $1e - 6$ and a KL penalty coefficient $\beta_{KL} = 0.01$. The maximum generation length is set to 16,384 tokens for all models (except *GRPO-8k*) to allow sufficient reasoning exploration. All experiments are conducted on $8 \times$ H100 GPUs.

Evaluation Metrics In addition to accuracy, we evaluate reasoning efficiency using the Compression Rate (CR). To faithfully reflect the model’s balanced performance across tasks of different difficulty, we compute CR per task first, then take the arithmetic mean:

$$CR_{\text{overall}} = \frac{1}{|\mathcal{D}|} \sum_{i \in \mathcal{D}} \frac{\text{Avg_Tok}_{\text{Model}}^{(i)}}{\text{Avg_Tok}_{\text{Vanilla}}^{(i)}} \quad (8)$$

where \mathcal{D} denotes the set of evaluated benchmark datasets. Lower values indicate better compression.

4.2 Main Results

Table 1 presents the performance of Step-GRPO compared to baselines across three model scales. Our empirical results suggest that Step-GRPO significantly improves the trade-off boundary between accuracy and reasoning efficiency, outperforming baselines in this region.

Superior Accuracy-Efficiency Trade-off On Qwen3-8B, Step-GRPO achieves 82.1% overall accuracy, surpassing the vanilla model while reducing token usage by 32.0%. Although the aggressive GRPO+LP baseline yields higher compression, it severely harms reasoning performance. Notably, on the hard AIME 2025 benchmark, GRPO+LP accuracy drops to 60.0%, showing that static token penalties force the model to cut essential reasoning steps. In contrast, Step-GRPO maintains 73.3% accuracy on the same task. This proves that our step-aware reward correctly distinguishes between redundancy and necessary logical complexity, preserving the model’s ability to reason deeply.

Necessity of Reinforcement Learning The results expose the limitations of supervised distillation. Creating the DEER+SFT dataset requires costly rejection sampling. Despite this effort, the resulting model generalizes poorly. On smaller

Table 1: Experimental results. The order is arranged as requested. “Acc” denotes accuracy, “Tok” denotes token count, and “CR” denotes compression rate (relative to Vanilla). Intermediate CR columns are omitted for brevity. Best results are in **bold**, and second-best results are underlined.

Method	GSM8K		GPQA		MATH-500		AMC 2023		AIME 2024		AIME 2025		Overall	
	Acc ↑	Tok ↓	Acc ↑	Tok ↓	Acc ↑	Tok ↓	Acc ↑	Tok ↓	Acc ↑	Tok ↓	Acc ↑	Tok ↓	Acc ↑	CR ↓
<i>Qwen3-8B</i>														
Vanilla	93.2	2,362	<u>62.1</u>	7,249	96.8	5,299	97.5	8,648	70.0	15,880	60.0	17,492	79.9	100%
DEER+SFT	90.5	1,733	47.0	8,860	95.0	3,361	90.0	6,149	63.3	10,623	50.0	13,330	72.6	78.9%
GRPO	93.3	2,262	54.6	6,087	96.8	5,092	97.5	8,204	76.7	12,545	<u>66.7</u>	15,407	<u>80.9</u>	89.7%
GRPO-8k	<u>94.2</u>	1,754	55.1	5,085	96.2	3,796	87.5	5,933	66.7	10,422	53.3	12,638	75.5	70.4%
GRPO+LP	<u>94.0</u>	813	55.6	4,820	95.0	2,336	92.5	3,786	<u>73.3</u>	9,899	60.0	<u>11,932</u>	78.4	53.2%
GRPO+SOP	93.0	1,582	62.6	5,330	95.8	3,822	97.5	5,431	66.7	<u>10,173</u>	<u>66.7</u>	11,844	80.4	67.9%
GRPO- λ	92.9	1,193	60.1	5,003	<u>96.4</u>	3,095	90.0	5,432	76.7	<u>10,554</u>	63.3	12,321	79.9	<u>62.9%</u>
Step-GRPO (Ours)	94.9	1,354	56.1	5,560	96.8	3,485	<u>95.0</u>	<u>5,295</u>	76.7	11,657	73.3	12,859	82.1	68.0%
<i>Qwen3-4B</i>														
Vanilla	<u>93.8</u>	2,242	55.1	6,644	96.2	5,346	97.5	8,801	70.0	15,453	63.3	18,049	79.3	100%
DEER+SFT	90.1	1,869	37.4	8,404	93.0	3,610	85.0	5,002	53.3	12,105	46.7	12,503	67.6	80.3%
GRPO	93.0	2,291	51.0	5,380	97.6	4,974	97.5	7,712	<u>73.3</u>	12,126	63.3	14,249	79.3	86.9%
GRPO-8k	<u>93.8</u>	1,753	50.5	4,571	96.2	3,691	<u>95.0</u>	5,506	70.0	10,515	<u>60.0</u>	<u>11,172</u>	77.6	68.1%
GRPO+LP	93.0	680	50.0	4,039	94.2	2,329	<u>95.0</u>	4,712	70.0	10,631	63.3	10,783	77.6	52.8%
GRPO+SOP	93.9	1,656	50.0	4,452	95.6	3,808	<u>95.0</u>	5,793	70.0	9,363	<u>60.0</u>	13,838	77.4	69.2%
GRPO- λ	93.1	1,261	48.0	4,606	96.6	3,232	97.5	4,671	70.0	9,094	<u>60.0</u>	12,814	77.5	<u>61.5%</u>
Step-GRPO (Ours)	93.3	1,298	<u>54.0</u>	5,182	95.6	3,528	<u>95.0</u>	5,456	76.7	11,435	<u>60.0</u>	13,356	<u>79.1</u>	68.7%
<i>Qwen3-1.7B</i>														
Vanilla	<u>89.2</u>	2,326	37.4	6,781	<u>90.8</u>	5,777	80.0	10,713	43.3	18,552	26.7	17,766	61.2	100%
DEER+SFT	84.2	2,811	22.2	14,165	83.0	4,615	67.5	5,712	30.0	<u>12,758</u>	20.0	<u>12,367</u>	51.2	100.2%
GRPO	90.1	2,100	39.4	6,037	92.4	5,140	85.0	9,425	<u>50.0</u>	<u>15,039</u>	<u>40.0</u>	<u>16,427</u>	66.1	88.3%
GRPO-8k	88.9	1,688	<u>38.9</u>	5,123	90.6	3,931	80.0	7,021	43.3	13,600	33.3	12,362	62.5	70.8%
GRPO+LP	90.1	951	<u>38.9</u>	4,691	89.4	2,718	72.5	<u>6,605</u>	53.3	11,508	20.0	11,412	60.7	57.5%
GRPO+SOP	<u>89.2</u>	1,778	36.9	5,188	90.2	4,419	<u>82.5</u>	7,704	40.0	13,934	30.0	14,216	61.5	76.1%
GRPO- λ	88.7	1,431	34.3	4,899	90.2	3,708	75.0	7,050	40.0	13,572	43.3	12,994	61.9	<u>68.3%</u>
Step-GRPO (Ours)	88.5	1,418	34.9	5,356	90.2	3,812	<u>82.5</u>	6,856	43.3	13,728	<u>40.0</u>	13,146	<u>63.2</u>	69.7%

models (4B and 1.7B), DEER+SFT suffers from severe instability on out-of-distribution (OOD) tasks. For instance, on the scientific reasoning benchmark GPQA, it exhibits negative compression (CR > 120-200%), generating hallucinations significantly longer than Vanilla while suffering drastic accuracy drops. This suggests SFT only mimics surface-level conciseness within the training distribution rather than internalizing the generalizable logic of when to stop. While other RL baselines like GRPO+SOP and GRPO- λ improve over static penalties, they still lag behind Step-GRPO on complex tasks, confirming the advantage of our semantic step quantification.

Consistency Across Model Scales Experiments on Qwen3-4B and 1.7B confirm our method’s robustness. Smaller models generally possess fragile reasoning capabilities. While GRPO+LP causes accuracy degradation (e.g., AIME 2025 drops to 20.0% on 1.7B), Step-GRPO maintains a competitive accuracy profile (e.g., 40.0% on AIME 2025) comparable to or better than standard baselines, while achieving over 30% compression. This robustness stems from our dynamic group-relative baseline, which adapts the difficulty to the model’s

current capability. Furthermore, Step-GRPO effectively targets the repetitive loops common in smaller models, achieving high compression on simple tasks without the catastrophic collapse seen in static penalty methods.

5 Discussion

5.1 Ablation Study

To validate the contribution of each component in Step-GRPO, we conduct ablation studies on Qwen3-8B using three representative benchmarks. The results are summarized in Table 2.

Table 2: Ablation studies on Qwen3-8B. We report Accuracy (%) and Average Token Count. The values demonstrate that removing the step reward leads to verbosity, while using a noisy all-sample baseline causes the worst performance degradation.

Method	GPQA		MATH-500		AIME 2024		Average	
	Acc	Tok	Acc	Tok	Acc	Tok	Acc	Tok
Step-GRPO (Ours)	56.1	5560	96.8	3485	76.7	11657	76.5	6901
w/o Step Reward	56.1	6300	96.4	4737	73.3	12787	75.3	7941
w/o Dynamic Rollout	56.6	5319	95.8	3822	70.0	10173	74.1	6438
w/ All-Sample Mean	54.6	5620	96.6	3994	66.7	12250	72.6	7288

Impact of Step-Aware Relative Reward To empirically verify whether Step-GRPO overcomes the

syntactic blindness of traditional length penalties, we designed a fine-grained structural analysis using GPT-4o to dissect the reasoning traces of 100 random samples. We explicitly categorized steps into *Forward Reasoning*, *Verification*, and *Redundancy*. This allows us to distinguish between the *essential verification* required for complex tasks and the *harmful overthinking* that inflates computation. As shown in Table 2, removing this component (w/o Step Reward) leads to a significant rebound in token usage (from 6901 to 7941), confirming that without explicit semantic regularization, the model inevitably drifts back to the verbose generation patterns inherent in the original policy.

Impact of Dynamic Truncated Rollout The variant without dynamic rollout (Row 3 in Table 2) shows a reduction in token usage due to the presence of the step penalty, but suffers a drop in accuracy (Avg 74.1%). Without exposure to “short yet correct” trajectories during training, the model *struggles to internalize the correct stopping logic*, forcing brevity at the cost of reasoning depth on complex tasks like AIME. This performance gap highlights a critical distribution mismatch: without the dynamic rollout acting as a rehearsal mechanism, the model treats early exit signals as out-of-distribution events, leading to hesitant and incomplete reasoning chains during inference.

Importance of Robust Baseline Calculation Calculating the baseline μ using all samples (Row 4 in Table 2) yields the lowest overall accuracy (72.6%). Including incorrect responses, which are often extremely short (give-up) or long (circular), introduces severe noise into the reward signal. For instance, short but incorrect “give-up” responses artificially lower the group mean μ , which inadvertently causes the reward function to penalize necessary, high-quality long reasoning steps as “inefficient”. This *confused reward baseline* prevents the model from effectively distinguishing between efficient reasoning and failure, leading to suboptimal convergence.

5.2 Case Study

To visualize the behavioral shift induced by our method, Figure 3 presents a qualitative comparison on a complex number theory problem requiring strict constraint satisfaction. The Vanilla model, while correct, exhibits a typical trial-and-error pattern characterized by *redundant hesitation* and backtracking, significantly inflating the computa-

tional cost. In contrast, optimization baselines like DEER+SFT and GRPO- λ succumb to *reasoning collapse*; they achieve brevity through a superficial mimicry of concise forms but fail to execute the critical checks required for this specific task, resulting in logical errors. Crucially, Step-GRPO demonstrates a successful *internalization* of the decisive reasoning policy. It effectively prunes the self-doubting loops observed in the Vanilla model but, unlike the aggressive baselines, preserves the *essential verification steps* necessary to identify the “least positive” integer, thereby achieving the correct solution with superior structural efficiency.

5.3 Why Semantic Steps Matter?

To empirically verify whether Step-GRPO overcomes the *syntactic blindness* of traditional length penalties, we designed a fine-grained structural analysis using GPT-4o to dissect the reasoning traces of 200 random samples. We explicitly categorized steps into *Forward Reasoning*, *Verification*, and *Redundancy*. This allows us to distinguish between the *essential verification* required for complex tasks and the *harmful overthinking* that inflates computation.

Selective Pruning of Redundancy. Our analysis reveals a critical divergence in how models achieve brevity. As shown in Figure 4(a), while token-based penalties (e.g., GRPO- λ) compress reasoning by indiscriminately suppressing all step types, Step-GRPO exhibits a selective pruning behavior. It reduces *redundant* steps (Grey bars) to the lowest level among all models (10.9%), yet retains a higher proportion of *verification* steps (22.9%, Yellow bars) compared to GRPO- λ (21.7%). This confirms that our method decouples reasoning efficiency from raw length, internalizing a policy that cuts *syntactic fat* (verbosity) while preserving *cognitive muscle* (verification).

Stabilizing Information Density. Beyond composition, we analyze the semantic density in Figure 4(b). This metric reflects the information payload per reasoning unit, quantified as:

$$\text{Semantic Density}_i = \frac{\text{Semantic Step Count } k_i}{\text{Total Tokens in } o_i} \quad (9)$$

The Vanilla model (Orange box) exhibits extreme variance with a “tall” distribution, indicating unpredictable “overthinking” loops where the model generates excessive tokens with low information

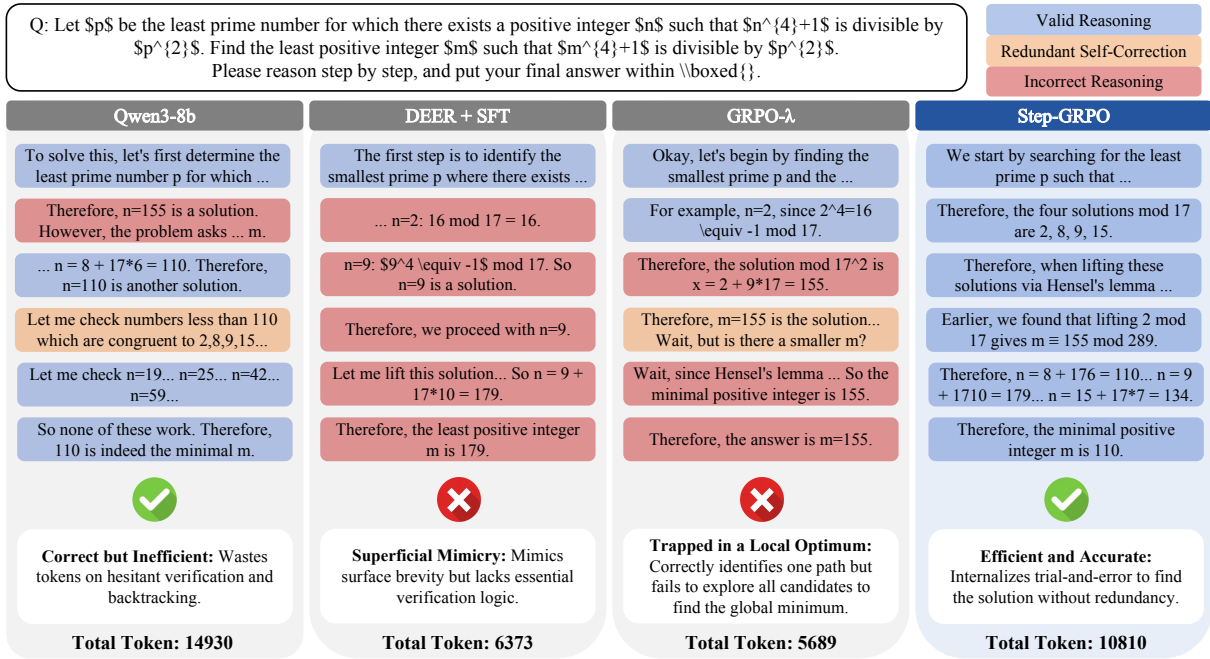


Figure 3: Qualitative comparison of reasoning chains on a number theory problem from AIME 2024.

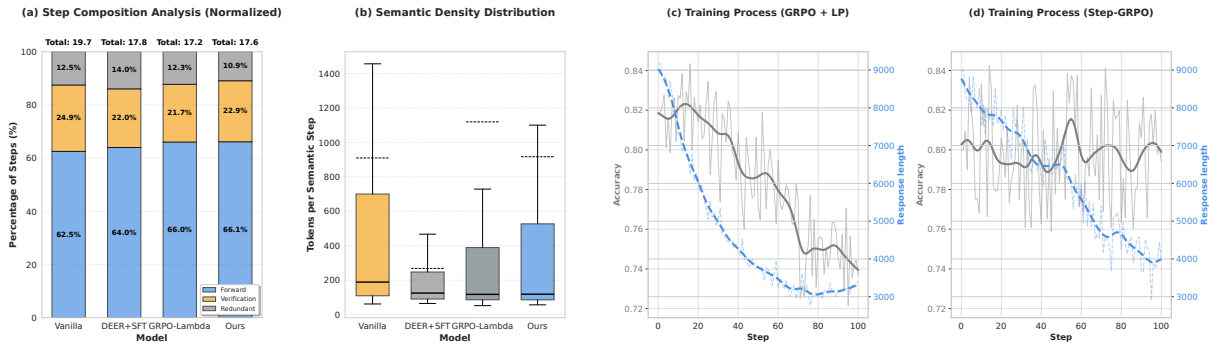


Figure 4: **Structural and Training Dynamics Analysis.** (a) **Step Composition Analysis.** Proportions of step types with average step counts annotated on top. (b) **Semantic Density Distribution.** Tokens per step (outliers excluded); dashed lines denote means. (c)(d) **Training Dynamics.** Evolution of accuracy (Grey) and length (Blue) for GRPO+LP and Step-GRPO.

gain. In contrast, Step-GRPO (Blue box) significantly compresses this variance. Unlike GRPO- λ which often forces premature truncation (yielding a lower median density), our method maintains a compact and consistent density distribution. This suggests that Step-GRPO stabilizes the reasoning process, avoiding the fragmentation observed in static length-penalty methods by effectively filtering out “low-density” redundant steps.

5.4 Training Stability

To investigate the robustness of our method against the “training collapse” phenomenon, we visualize the evolution of response length and accuracy on the validate dataset in Figure 4(c)(d). The results reveal a critical failure mode in static token penal-

ties (Left): *GRPO+LP* shows a strong correlation between the rapid decrease in length (Blue dashed line) and a *simultaneous, precipitous drop in accuracy* (Gray solid line, from ~ 0.82 to ~ 0.74). This indicates that the model is forced to indiscriminately abandon essential reasoning steps to satisfy the fixed penalty. Conversely, *Step-GRPO* (Right) exhibits a *decoupled trajectory*: while the reasoning length is significantly compressed (from $\sim 9k$ to $\sim 4k$ tokens), the accuracy remains robustly stable around 0.82. This demonstrates that our dynamic, group-relative baseline effectively acts as an *adaptive curriculum*, penalizing only relative redundancy while *protecting the necessary cognitive depth* required for complex problem-solving.

6 Conclusion

In this paper, we introduced *Step-GRPO*, a novel post-training framework that enables Large Language Models to internalize efficient reasoning strategies. By synergizing dynamic truncated rollouts with a semantic step-aware relative reward, our method shifts the optimization objective from syntactic token minimization to semantic logic condensation. Extensive experiments across diverse benchmarks demonstrate that Step-GRPO achieves a superior accuracy-efficiency trade-off, significantly reducing computational costs while maintaining robustness on complex reasoning tasks. Crucially, we show that our dynamic, group-relative mechanism effectively resolves the “training collapse” issue plaguing traditional length-penalty methods. Ultimately, this work offers a scalable path toward efficient reasoning: transforming the explicit “early-exit” decision-making process into the model’s intrinsic intuition, yielding decisive and precise reasoning chains without the need for external inference-time interventions.

Limitation

While Step-GRPO successfully internalizes efficient reasoning, the *Dynamic Truncated Rollout* mechanism introduces a marginal increase in training latency during the generation phase, primarily due to the additional forward passes for confidence estimation. However, this overhead is partially mitigated by the accelerated parameter update phase, which benefits from significantly shorter sequence lengths. We consider this a justifiable “training-time investment” as it yields a zero-overhead model for deployment. Additionally, our current semantic step quantification relies on explicit linguistic markers, which may limit applicability in domains lacking such structures. Future work will focus on developing domain-agnostic step segmentation methods and exploring iterative self-training to reduce reliance on predefined triggers.

Ethics Statement

This research complies with the ARR Ethics Policy. The datasets used in this study (e.g., GSM8K, MATH-500, AIME) are established, publicly available benchmarks for mathematical and logical reasoning, ensuring no violation of privacy or copyright. No human subjects or crowdworkers were employed in this research. We believe this work presents no significant risk of harm and offers a

positive societal impact by democratizing efficient reasoning.

Acknowledgements

This work was supported by the Key Laboratory of Cognitive Intelligence and Content Security, Ministry of Education (Grant No.10120251107, Harbin Institute of Technology). The National Natural Science Foundation of China (Grant No. 62477012), the AI for Science Program of the Shanghai Municipal Commission of Economy and Informatization, China (Grant No. 2025-GZL-RGZN-BTBX-01014) and the robotic AI-Scientist platform of Chinese Academy of Sciences.

References

- Shivam Agarwal, Zimin Zhang, Lifan Yuan, Jiawei Han, and Hao Peng. 2025. The unreasonable effectiveness of entropy minimization in llm reasoning. *arXiv preprint arXiv:2505.15134*.
- AI-MO. 2024. Amc 2023. <https://huggingface.co/datasets/AI-MO/aimo-validation-amc>. Accessed: 2026-01-04.
- Xingyu Chen, Jiahao Xu, Tian Liang, Zhiwei He, Jianhui Pang, Dian Yu, Linfeng Song, Qiuqi Liu, Mengfei Zhou, Zhuosheng Zhang, and 1 others. Do not think that much for $2+3=?$ on the overthinking of long reasoning models. In *Forty-second International Conference on Machine Learning*.
- Karl Cobbe, Vineet Kosaraju, Mohammad Bavarian, Mark Chen, Heewoo Jun, Lukasz Kaiser, Matthias Plappert, Jerry Tworek, Jacob Hilton, Reiichiro Nakano, and 1 others. 2021. Training verifiers to solve math word problems. *arXiv preprint arXiv:2110.14168*.
- Yingqian Cui, Pengfei He, Jingying Zeng, Hui Liu, Xianfeng Tang, Zhenwei Dai, Yan Han, Chen Luo, Jing Huang, Zhen Li, and 1 others. 2025. Stepwise perplexity-guided refinement for efficient chain-of-thought reasoning in large language models. *arXiv preprint arXiv:2502.13260*.
- Muzhi Dai, Shixuan Liu, and Qingyi Si. 2025a. Stable reinforcement learning for efficient reasoning. *arXiv preprint arXiv:2505.18086*.
- Muzhi Dai, Chenxu Yang, and Qingyi Si. 2025b. S-grpo: Early exit via reinforcement learning in reasoning models. *arXiv preprint arXiv:2505.07686*.
- Yichao Fu, Junda Chen, Yonghao Zhuang, Zheyu Fu, Ion Stoica, and Hao Zhang. 2025. Reasoning without self-doubt: More efficient chain-of-thought through certainty probing. In *ICLR 2025 Workshop on Foundation Models in the Wild*.

- Leo Gao, John Schulman, and Jacob Hilton. 2023. Scaling laws for reward model overoptimization. In *International Conference on Machine Learning*, pages 10835–10866. PMLR.
- Daya Guo, Dejian Yang, Haowei Zhang, Junxiao Song, Ruoyu Zhang, Runxin Xu, Qihao Zhu, Shitong Ma, Peiyi Wang, Xiao Bi, and 1 others. 2025. Deepseek-r1: Incentivizing reasoning capability in llms via reinforcement learning. *arXiv preprint arXiv:2501.12948*.
- 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. *arXiv preprint arXiv:2103.03874*.
- 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. *arXiv preprint arXiv:2412.16720*.
- 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.
- Zeju Li, Jianyuan Zhong, Ziyang Zheng, Xiangyu Wen, Zhijian Xu, Yingying Cheng, Fan Zhang, and Qiang Xu. 2025. Compressing chain-of-thought in llms via step entropy. *arXiv preprint arXiv:2508.03346*.
- Hunter Lightman, Vineet Kosaraju, Yuri Burda, Harrison Edwards, Bowen Baker, Teddy Lee, Jan Leike, John Schulman, Ilya Sutskever, and Karl Cobbe. 2023. Let’s verify step by step. In *The Twelfth International Conference on Learning Representations*.
- Wenjie Ma, Jingxuan He, Charlie Snell, Tyler Griggs, Sewon Min, and Matei Zaharia. 2025a. Reasoning models can be effective without thinking. *arXiv preprint arXiv:2504.09858*.
- Xinyin Ma, Guangnian Wan, Runpeng Yu, Gongfan Fang, and Xinchao Wang. 2025b. Cot-valve: Length-compressible chain-of-thought tuning. *arXiv preprint arXiv:2502.09601*.
- Niklas Muennighoff, Zitong Yang, Weijia Shi, Xiang Lisa Li, Li Fei-Fei, Hannaneh Hajishirzi, Luke Zettlemoyer, Percy Liang, Emmanuel Candès, and Tatsunori B Hashimoto. 2025. s1: Simple test-time scaling. In *Proceedings of the 2025 Conference on Empirical Methods in Natural Language Processing*, pages 20286–20332.
- Ziqing Qiao, Yongheng Deng, Jiali Zeng, Dong Wang, Lai Wei, Guanbo Wang, Fandong Meng, Jie Zhou, Ju Ren, and Yaoyue Zhang. 2025. Concise: Confidence-guided compression in step-by-step efficient reasoning. In *Proceedings of the 2025 Conference on Empirical Methods in Natural Language Processing*, pages 8021–8040.
- 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*.
- Keita Saito, Akifumi Wachi, Koki Wataoka, and Youhei Akimoto. 2023. Verbosity bias in preference labeling by large language models. *arXiv preprint arXiv:2310.10076*.
- John Schulman, Filip Wolski, Prafulla Dhariwal, Alec Radford, and Oleg Klimov. 2017. Proximal policy optimization algorithms. *arXiv preprint arXiv:1707.06347*.
- Zhihong Shao, Peiyi Wang, Qihao Zhu, Runxin Xu, Junxiao Song, Xiao Bi, Haowei Zhang, Mingchuan Zhang, YK Li, Yang Wu, and 1 others. 2024. Deepseekmath: Pushing the limits of mathematical reasoning in open language models. *arXiv preprint arXiv:2402.03300*.
- Charlie Snell, Jaehoon Lee, Kelvin Xu, and Aviral Kumar. 2024. Scaling llm test-time compute optimally can be more effective than scaling model parameters. *arXiv preprint arXiv:2408.03314*.
- Jinyan Su and Claire Cardie. 2025. Thinking fast and right: Balancing accuracy and reasoning length with adaptive rewards. *arXiv preprint arXiv:2505.18298*.
- Mirac Suzgun, Nathan Scales, Nathanael Schärli, Sebastian Gehrmann, Yi Tay, Hyung Won Chung, Aakanksha Chowdhery, Quoc Le, Ed Chi, Denny Zhou, and 1 others. 2023. Challenging big-bench tasks and whether chain-of-thought can solve them. In *Findings of the Association for Computational Linguistics: ACL 2023*, pages 13003–13051.
- Kimi Team, Angang Du, Bofei Gao, Bowei Xing, Changjiu Jiang, Cheng Chen, Cheng Li, Chenjun Xiao, Chenzhuang Du, Chonghua Liao, and 1 others. 2025. Kimi k1. 5: Scaling reinforcement learning with llms. *arXiv preprint arXiv:2501.12599*.
- Arne Vanhoyweghen, Brecht Verbeken, Andres Al-gaba, and Vincent Ginis. 2025. Lexical hints of accuracy in llm reasoning chains. *arXiv preprint arXiv:2508.15842*.
- Shenzhi Wang, Le Yu, Chang Gao, Chujie Zheng, Shixuan Liu, Rui Lu, Kai Dang, Xionghui Chen, Jianxin Yang, Zhenru Zhang, and 1 others. 2025. Beyond the 80/20 rule: High-entropy minority tokens drive effective reinforcement learning for llm reasoning. *arXiv preprint arXiv:2506.01939*.
- Heming Xia, Chak Tou Leong, Wenjie Wang, Yongqi Li, and Wenjie Li. 2025. Tokenskip: Controllable chain-of-thought compression in llms. *arXiv preprint arXiv:2502.12067*.
- Silei Xu, Wenhao Xie, Lingxiao Zhao, and Pengcheng He. 2025. Chain of draft: Thinking faster by writing less. *arXiv preprint arXiv:2502.18600*.

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*.

Chenxu Yang, Qingyi Si, Yongjie Duan, Zheliang Zhu, Chenyu Zhu, Qiaowei Li, Minghui Chen, Zheng Lin, and Weiping Wang. 2025b. Dynamic early exit in reasoning models. *arXiv preprint arXiv:2504.15895*.

Shenzhi Wang Zhangchi Feng Dongdong Kuang Yuwen Xiong Yaowei Zheng, Junting Lu. 2025. Easyr1: An efficient, scalable, multi-modality rl training framework. <https://github.com/hiyouga/EasyR1>.

Qiyang Yu, Zheng Zhang, Ruofei Zhu, Yufeng Yuan, Xiaochen Zuo, Yu Yue, Weinan Dai, Tiantian Fan, Gaohong Liu, Lingjun Liu, and 1 others. 2025. Dapo: An open-source llm reinforcement learning system at scale. *arXiv preprint arXiv:2503.14476*.

Di Zhang, Weida Wang, Junxian Li, Xunzhi Wang, Jiatong Li, Jianbo Wu, Jingdi Lei, Haonan He, Peng Ye, Shufei Zhang, and 1 others. 2025. Control-r: Towards controllable test-time scaling. *arXiv preprint arXiv:2506.00189*.

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

Yifan Zhang and Team Math-AI. 2025. American invitational mathematics examination (aime) 2025.

Yaowei Zheng, Richong Zhang, Junhao Zhang, Yanhan Ye, Zheyang Luo, Zhangchi Feng, and Yongqiang Ma. 2024. [Llamafactory: Unified efficient fine-tuning of 100+ language models](#). In *Proceedings of the 62nd Annual Meeting of the Association for Computational Linguistics (Volume 3: System Demonstrations)*, Bangkok, Thailand. Association for Computational Linguistics.

Hourun Zhu, Yang Gao, Wenlong Fei, Jiawei Li, and Huashan Sun. 2025. Entropy-guided reasoning compression. *arXiv preprint arXiv:2511.14258*.

A Prompts

Inference Prompt Template

{Input Question} + Please reason step by step, and put your final answer within `\boxed{ }`.

CoT Analysis Prompt Template

System: You are an expert in analyzing the reasoning processes of Large Language Models. Your task is to segment a raw "Chain of Thought" (CoT) into distinct semantic steps and classify each step.

Input Data

You will receive:

1. The Math Problem.
2. The raw reasoning text (content within `<think>` tags).
3. Whether the problem was solved correctly.

Task

1. **Segmentation:** Break the reasoning text into a list of chronological steps. A "step" is a coherent unit of thought, calculation, or reflection.
2. **Classification:** Assign one of the following labels to each step:
 - **Forward:** Constructive reasoning that moves closer to the solution (e.g., proposing a method, performing a calculation, deriving a sub-result).
 - **Verification:** Self-correction, double-checking, or validating a previous step (e.g., "Wait, let me check", "Re-calculating this", "Since 17^2 is 289..."). *Note: This is considered a VALID and useful step.*
 - **Redundant:** Unnecessary repetition, getting stuck in a loop, stalled thinking, or restating the obvious without progress (e.g., repeating the question multiple times, "Hmm... let me think", circular logic that adds no value).

Output Format (JSON Only) {

```
"total_steps": 15,
"breakdown": {
  "forward_count": 10,
  "verification_count": 3,
  "redundant_count": 2
},
"steps": [
  {"content": "...", "label": "Forward"},
  {"content": "Wait, let me double check...", "label": "Verification"},
```

```

{"content": "So the answer is... wait no...", "label": "Redundant"}
]
}

```

B Hyper-parameters

In this section, we list the hyper-parameters used in different phases of training and inference. We used Llama-Factory (Zheng et al., 2024) to conduct SFT training (for DEER+SFT after rejection sampling), and EasyR1 (Yaowei Zheng, 2025) for GRPO training. Table 3 provides the values for the hyper-parameters. The table includes settings such as the number of GPUs, learning rates, batch sizes, and the number of epochs for each phase.

Hyper-parameter	Value
<i>Supervised Fine-Tuning (SFT)</i>	
Number of GPUs (H200)	8
Per Device Train Batch Size	1
Gradient Accumulation Steps	2
Learning Rate	1.0×10^{-5}
LR Scheduler	Cosine
Warmup Ratio	0.1
Epochs	3
<i>GRPO Training</i>	
Number of GPUs (H200)	8
Learning Rate	1.0×10^{-6}
Weight Decay	1.0×10^{-2}
KL Coefficient (β_{KL})	0.01
Group Size (G)	5
Rollout Temperature	1.0
Global Batch Size	128
Rollout Batch Size	512
Total Training Steps	100
<i>Inference</i>	
Temperature	0.6
Top-p	0.9
Max Tokens	32768

Table 3: Hyper-parameters for SFT, GRPO training, and Inference.

C Full Reasoning Trajectories

Demo of Step-GRPO Output

User Input:

Every morning, Aya does a 9 kilometer walk, and then finishes at the coffee shop. One day, she walks at s kilometers per hour, and the walk takes 4 hours, including t minutes at the coffee shop. Another morning, she walks at $s + 2$ kilometers per hour, and

the walk takes 2 hours and 24 minutes, including t minutes at the coffee shop. This morning, if she walks at $s + \frac{1}{2}$ kilometers per hour, how many minutes will the walk take, including the t minutes at the coffee shop? Please reason step by step, and put your final answer within `\boxed{ }`.

Model Prediction:

<think> Okay, so I need to figure out how long Aya's walk will take this morning, including the time she spends at the coffee shop. Let me start by understanding the problem step by step.

First, the problem says that every morning Aya does a 9 kilometer walk and then finishes at the coffee shop. So, the total distance she walks each morning is 9 km, right? Then, on one day, she walks at a speed of s km/h, and the walk takes 4 hours, including t minutes at the coffee shop. Another morning, she walks at $s + 2$ km/h, and the walk takes 2 hours and 24 minutes, including t minutes at the coffee shop. Then, this morning, she walks at $s + 1/2$ km/h, and we need to find how many minutes the walk takes, including t minutes at the coffee shop.

Hmm, okay. Let me parse this again. The key here is that the total time includes both the walking time and the time spent at the coffee shop. So, when they say "the walk takes 4 hours, including t minutes at the coffee shop," that means the total time from starting to finish at the coffee shop is 4 hours, which includes both the walking time and the t minutes at the coffee shop. Similarly for the other day.

So, let me denote: For the first day: Total time = walking time + t minutes = 4 hours. But walking time is distance divided by speed. The distance is 9 km, speed is s km/h. So walking time is $9/s$ hours. Then, total time is $9/s + t/60$ hours = 4 hours.

Similarly, for the second day: Total time is 2 hours and 24 minutes, which is $2 + 24/60 = 2.4$ hours. So, same as before, walking time is $9/(s + 2)$ hours, and total time is $9/(s + 2) + t/60 = 2.4$ hours.

So, we have two equations: 1) $9/s + t/60 = 4$ 2) $9/(s + 2) + t/60 = 2.4$

We need to solve these equations to find s and t , then use them to find the total time for the third day, which is walking time at $s + 1/2$ km/h plus t minutes.

So, first, let me write equations 1 and 2. Equation 1: $9/s + t/60 = 4$. Equation 2: $9/(s + 2) + t/60 = 2.4$. So, if I subtract equation 2 from equation 1, the $t/60$ terms will cancel out. Let me try that. Equation 1 - Equation 2: $[9/s - 9/(s + 2)] + [t/60 - t/60] = 4 - 2.4$. Simplify: $9/s - 9/(s + 2) = 1.6$.

So, compute $9/s - 9/(s + 2) = 1.6$. Let me compute the left side: $9[1/s - 1/(s + 2)] = 9[(s + 2 - s)/[s(s + 2)]] = 9[2/(s(s + 2))] = 18/[s(s + 2)]$. So, $18/[s(s + 2)] = 1.6$. Let me write 1.6 as $8/5$, so: $18/[s(s + 2)] = 8/5$.

Cross multiplying: $18 \times 5 = 8 \times s(s + 2) \Rightarrow 90 = 8s^2 + 16s$. Bring all terms to one side: $8s^2 + 16s - 90 = 0$. Divide both sides by 2: $4s^2 + 8s - 45 = 0$. Now, solving quadratic equation $4s^2 + 8s - 45 = 0$. Using quadratic formula: $s = [-b \pm \sqrt{b^2 - 4ac}]/2a$. Here, $a = 4, b = 8, c = -45$. Discriminant $D = 8^2 - 4 \times 4 \times (-45) = 64 + 720 = 784$. $\sqrt{784} = 28$. So, $s = (-8 \pm 28)/8$. We have two solutions: $s = (-8 + 28)/8 = 20/8 = 2.5$ and $s = (-8 - 28)/8 =$

$-36/8 = -4.5$. But speed can't be negative, so $s = 2.5$ km/h.

Now, let me find t from equation 1: $9/s + t/60 = 4$. So, $9/2.5 + t/60 = 4$. Compute $9/2.5$: 9 divided by 2.5 is same as $9 \times 2/5 = 18/5 = 3.6$ hours. So, $3.6 + t/60 = 4 \Rightarrow t/60 = 4 - 3.6 = 0.4$ hours. Convert 0.4 hours to minutes: $0.4 \times 60 = 24$ minutes. So, $t = 24$ minutes.

So, now, the problem says: This morning, if she walks at $s + 1/2$ km/h, how many minutes will the walk take, including the t minutes at the coffee shop? So, s is 2.5, so $s + 1/2$ is $2.5 + 0.5 = 3$ km/h. So, her speed is 3 km/h. The distance is still 9 km. So, walking time is $9/3 = 3$ hours. Then, total time is walking time + t minutes = 3 hours + 24 minutes. Convert 3 hours to minutes: $3 \times 60 = 180$ minutes. Then add 24 minutes: $180 + 24 = 204$ minutes.

Therefore, the total time is 204 minutes. Let me check if this is consistent with the second day's data. On the second day, she walked at $s + 2 = 2.5 + 2 = 4.5$ km/h. Walking time is $9/4.5 = 2$ hours. Then total time is 2 hours + 24 minutes = 2 hours 24 minutes, which is 2.4 hours, which matches the given data. So that's correct.

So, the answer should be 204 minutes. </think>

To determine how long Aya's walk takes this morning, including the time spent at the coffee shop, we need to analyze the relationship between her walking speed, the distance, and the total time (including the fixed time at the coffee shop).

Step 1: Define the Variables

Let s be Aya's walking speed in km/h and t be the fixed time in minutes at the coffee shop. We are given:

- Day 1: Speed s , Total time 4h.

$$\frac{9}{s} + \frac{t}{60} = 4 \quad (1)$$

- Day 2: Speed $s + 2$, Total time 2.4h.

$$\frac{9}{s+2} + \frac{t}{60} = 2.4 \quad (2)$$

Step 2: Eliminate t to Solve for s

Subtract equation (2) from equation (1):

$$\begin{aligned} \left(\frac{9}{s} - \frac{9}{s+2} \right) &= 4 - 2.4 = 1.6 \\ 9 \left(\frac{s+2-s}{s(s+2)} \right) &= 1.6 \\ \frac{18}{s(s+2)} &= \frac{8}{5} \\ 90 &= 8s^2 + 16s \\ 4s^2 + 8s - 45 &= 0 \end{aligned}$$

Use the quadratic formula:

$$\begin{aligned} s &= \frac{-8 \pm \sqrt{8^2 - 4(4)(-45)}}{2(4)} \\ &= \frac{-8 \pm \sqrt{64 + 720}}{8} \\ &= \frac{-8 \pm 28}{8} \end{aligned}$$

We get $s = 2.5$ (discarding $s = -4.5$).

Step 3: Solve for t

Plug $s = 2.5$ into equation (1):

$$\begin{aligned} \frac{9}{2.5} + \frac{t}{60} &= 4 \\ 3.6 + \frac{t}{60} &= 4 \Rightarrow t = 24 \text{ minutes} \end{aligned}$$

Step 4: Compute Total Time This Morning

This morning, Aya walks at $s + 0.5 = 3$ km/h.

$$\begin{aligned} \text{Total Time} &= \frac{9}{3} \text{ hours} + 24 \text{ minutes} \\ &= 3 \times 60 + 24 \\ &= 180 + 24 = 204 \text{ minutes} \end{aligned}$$

Final Answer

204

D Generalization Capabilities

To verify the broad universality of Step-GRPO and rule out the possibility that our method overfits to specific models or domains, we evaluate its performance across two crucial dimensions: differing model architectures and non-mathematical reasoning domains.

Generalization to Other Architectures We conducted supplementary experiments substituting the base model with the DeepSeek-R1-Distill-Llama-8B architecture. The experimental results show that Step-GRPO also demonstrates robust performance on the Llama architecture. Specifically, Step-GRPO achieved the best overall token compression rate (CR = 66.79%) compared to Vanilla (100.00%), GRPO (109.03%), and GRPO- λ (81.83%) while maintaining a highly competitive overall accuracy of 65.40%.

On highly challenging tasks like AIME and relatively difficult tasks such as GPQA, Step-GRPO

Table 4: Generalization results on DeepSeek-R1-Distill-Llama-8B architecture.

Method	GSM8K		GPQA Diamond		MATH-500		AMC 23		AIME 2024		AIME 2025		Overall	
	Acc	Tok	Acc	Tok	Acc	Tok	Acc	Tok	Acc	Tok	Acc	Tok	Acc	CR
Vanilla	87.95	1673.61	46.97	8919.74	89.80	4237.33	87.50	6151.73	46.67	15542.40	23.33	14925.97	63.70	100.00
GRPO	89.23	2228.22	50.51	9084.33	93.20	4476.27	90.00	7792.75	53.33	13407.43	26.67	15017.83	67.20	109.03
GRPO- λ	86.96	987.39	52.02	8082.42	90.40	3138.21	87.50	5633.93	46.67	12711.67	30.00	14019.73	65.60	81.83
Step-GRPO (Ours)	87.49	976.08	51.01	6984.80	88.60	2710.77	85.00	5060.15	50.00	9105.30	30.00	8848.23	65.40	66.79

achieves significant token compression while maintaining or even improving accuracy. This indicates that our formulation—optimizing relative semantic efficiency—effectively adapts to reasoning models with diverse, idiosyncratic output characteristics without relying on model-specific features.

Generalization to Non-Mathematical Domains

To further demonstrate that our approach transfers beyond domain-specific math problems, we evaluated Step-GRPO on BIG-Bench Hard (BBH) (Suzgun et al., 2023), a benchmark emphasizing symbolic and abstract logical reasoning. As shown in Table 5, Step-GRPO achieves the highest accuracy (86.51%) across all methods while maintaining a superior compression rate compared to standard GRPO and SFT baselines. This confirms that the internalized cognitive state transitions captured by our linguistic triggers apply universally across diverse reasoning paradigms.

Table 5: Performance on the BIG-Bench Hard (BBH) benchmark.

Method	Acc	Tokens	CR
Vanilla	86.14	1337	100.0
GRPO	85.96	1247	93.2
DEER-SFT	84.89	1520	113.6
GRPO- λ	86.30	989	74.0
GRPO-LP	84.81	811	60.7
GRPO-SOP	86.46	1084	81.1
Step-GRPO (Ours)	86.51	1097	82.1

E Ablation on Confidence Threshold

The dynamic truncated rollout relies on the threshold δ to decide when to pause generation. A potential concern is whether confidence estimates correlate perfectly with correctness, as models can be “confidently wrong” on harder instances. However, Step-GRPO does not rely solely on the pre-trained priors; the RL process itself serves as a continuous calibration mechanism. During training, trajectories that are “short, high-confidence,

but factually incorrect” receive severe negative rewards. This dense error-signal forces the model to gradually align its confidence probability with actual correctness. To demonstrate the robustness of our dynamic exit criterion, we performed an ablation on the threshold parameter δ . The overall results in Table 6 show that while lower thresholds ($\delta = 0.80$, $\delta = 0.90$) induce more aggressive truncation leading to faster performance drops, our chosen threshold ($\delta = 0.95$) effectively balances accuracy and token usage. Notably, the method comfortably outperforms baselines across a broad, reasonable range of confidence thresholds, reinforcing the validity of the stopping criterion.

F Empirical Stability Evaluation

To further address variance and confirm stability, we conducted independent runs across all benchmarks on Qwen3-8B. Table 7 reports the Accuracy Mean \pm Standard Deviation across three random seeds. Step-GRPO achieves the highest overall stability, consistently minimizing outcome variance effectively. In contrast, static length-penalty methods demonstrate significant instability, especially on highly demanding tasks such as AIME.

G Training Overhead

To clearly demonstrate the trade-off in computational cost introduced by the Dynamic Truncated Rollout, we compare the average per-step time overhead (in seconds) of standard GRPO and Step-GRPO on the Qwen3-8B model. As shown in Table 8, we evaluate both the initial phase (first 50 steps) and the convergence phase (last 50 steps).

While the *Generation* phase naturally incurs an initial overhead due to the dynamic confidence estimation and answer induction during rollout, this is effectively compensated for by significant time savings in the *Update*, *Old Policy*, and *Reference Model* phases. This reduction occurs because as the model rapidly learns to generate shorter, more efficient sequences, the computational burden for backward passes and likelihood calculations drops dramatically. Notably, as the model

Table 6: Ablation on confidence threshold δ . We report Accuracy (%) across six benchmarks and Overall Compression Rate (CR).

Model / Threshold	GSM8K	GPQA	MATH-500	AMC 23	AIME 24	AIME 25	Overall (Acc)	Overall (CR)
Vanilla	93.2	62.1	96.8	97.5	70.0	60.0	79.9	100.0
$\delta = 0.80$	93.3	55.6	94.0	92.5	66.7	46.7	74.8	62.0
$\delta = 0.90$	93.4	56.1	95.4	90.0	70.0	56.7	76.9	64.0
Step-GRPO ($\delta = 0.95$)	94.9	56.1	96.8	95.0	76.7	73.3	82.1	68.0

Table 7: Accuracy (Mean \pm Std) across 3 independent runs on Qwen3-8B.

Method	GSM8K	MATH-500	GPQA	AMC 23	AIME 24	AIME 25	Overall
Vanilla	93.1 \pm 0.4	96.5 \pm 0.3	57.1 \pm 0.0	95.0 \pm 0.0	67.8 \pm 5.1	62.2 \pm 7.7	78.6 \pm 2.3
DEER+SFT	90.8 \pm 0.3	94.4 \pm 0.7	46.8 \pm 0.6	87.5 \pm 2.5	57.8 \pm 3.9	51.1 \pm 3.9	71.4 \pm 2.0
GRPO	93.3 \pm 0.3	96.4 \pm 0.2	57.7 \pm 1.5	93.3 \pm 1.4	76.7 \pm 3.3	58.9 \pm 6.9	79.4 \pm 2.3
GRPO- λ	92.9 \pm 0.2	96.1 \pm 0.3	57.1 \pm 2.2	92.5 \pm 0.0	73.3 \pm 3.3	54.5 \pm 6.9	77.7 \pm 2.2
GRPO+LP	93.7 \pm 0.3	95.1 \pm 0.2	58.3 \pm 1.8	93.3 \pm 2.9	68.9 \pm 3.9	50.0 \pm 5.8	76.5 \pm 2.5
GRPO+SOP	92.5 \pm 0.2	96.5 \pm 0.1	57.2 \pm 1.9	92.5 \pm 0.0	75.6 \pm 1.9	62.2 \pm 5.1	79.4 \pm 1.6
Step-GRPO	94.8 \pm 0.0	96.5 \pm 0.2	56.6 \pm 0.9	95.8 \pm 1.4	77.8 \pm 5.1	68.9 \pm 3.9	81.7 \pm 1.9

Table 8: Average per-step time overhead (seconds) on Qwen3-8B.

Method	Generation	Update	Old Policy	Ref Model	Reward
GRPO (0-50 steps)	1184.8	928.5	183.4	187.1	3.8
Step-GRPO (0-50 steps)	1678.1	659.2	101.3	101.1	3.8
GRPO (50-100 steps)	1111.6	904.1	179.4	183.1	3.6
Step-GRPO (50-100 steps)	1392.7	551.3	85.8	85.6	3.8

converges (Steps 50-100), Step-GRPO actually becomes faster overall due to these escalating efficiency gains, demonstrating that the initial rollout calculation is a highly worthwhile investment for an efficient training loop.

H Robustness of Heuristic Triggers

A natural concern regarding our methodology is the sensitivity of the heuristic trigger words (*e.g.*, “Wait”, “Alternatively”) used to segment semantic steps. While exploring parameter-free or learned segmentation models represents a promising future direction, we argue that our current approach provides sufficient robustness for two primary reasons. First, *Action Transition Points*: recent literature (Yang et al., 2025b; Qiao et al., 2025) validates these markers as Action Transition Points that naturally signal a high-entropy cognitive shift from linear deduction to self-verification or branch correction. They serve as semantic anchors that effectively delineate valid reasoning steps from redundant loops, demonstrating that they are fundamental to multi-step reasoning processes rather than artifacts of a specific writing style or language.

Second, *RL Adaptability and Invariance*: while heuristic, these triggers serve as initial anchors for an adaptive RL process. Step-GRPO fundamentally relies on the group-relative baseline (Equation 4). Even if the chosen trigger set is imperfect—such as missing occasional boundary markers in a specific domain—the resulting “step count bias” is uniformly applied across all completions within the same sample group \mathcal{G} , ensuring the *relative ranking* of generation efficiency remains consistent and valid. Furthermore, the model learns to align its internal confidence with these structural boundaries to maximize the reward. This structural resilience allows the RL gradient to correctly favor shorter valid paths, making the system inherently robust to minor linguistic variations and the absolute granularity defining a single “step”.