Walk Before You Run!
Group Length Policy Optimization for Efficient LLM Reasoning

Anonymous ACL submission

Abstract

As test-time scaling becomes a pivotal research frontier in the development of Large Language Models (LLMs), contemporary and advanced post-training methodologies increasingly focus on extending the generation length of long Chain-of-Thought (CoT) responses to enhance reasoning capabilities, aiming for DeepSeek R1-like performance. However, recent studies reveal a persistent overthinking phenomenon in state-of-the-art reasoning baselines, manifesting as excessive redundancy or repetitive thinking patterns. To address this issue, in this paper, we propose **G**roup **L**ength **P**olicy **O**ptimization (**GLPO**), a simple yet effective approach, for achieving concise reasoning in LLMs. Specifically, GLPO penalizes CoT outputs in its reward function based on their length relative to the max output length, thereby encouraging the model to generate shorter, higher-quality CoT outputs to solve problems. Meanwhile, GLPO only optimizes the length of CoT outputs once all rollouts of a sample are correct, following the "*walk before you run*" principle. Experimental results show that the model trained with GLPO, which generates more concise CoT outputs, outperforms recent state-of-the-art reasoning models with zero RL paradigm across AIME 2024, MATH-500, AMC 2023, Minerva, and Olympiad benchmarks. The model checkpoints will be publicly released.

1 Introduction

Test-time scaling (Snell et al., 2024; Muennighoff et al., 2025) has demonstrated a robust correlation between extending the generation length of Chain-of-Thought (CoT) (Wei et al., 2022) and improving the reasoning capabilities of Large Language Models (LLMs). The advent of large reasoning models, such as GPT-o1 (Jaech et al., 2024) and DeepSeek-R1 (Guo et al., 2025), represents a significant breakthrough in natural language processing, especially in tackling complex and intricate reasoning tasks. An interesting phenomenon observed during reinforcement learning post-training via Group Relative Policy Optimization (GRPO) (Shao et al., 2024) is the emergence of an "aha moment" (Guo et al., 2025), which refers to a pivotal inflection point at which the model spontaneously initiates self-correction behaviors. These emergent behaviors develop autonomously through the model's exploration of the solution space rather than through explicit programming. Prior research (Yeo et al., 2025; Luo et al., 2025) has found a distinctive pattern following this moment: the response length of the model tends to increase significantly, accompanied by improvements in overall performance. Despite the lack of a clear understanding of why this occurs, this phenomenon has led many researchers to advocate for longer responses, leveraging additional computational resources in the hope of further enhancing accuracy.

However, generating excessively long reasoning responses substantially increases computational overhead in both the model training and deployment phases. Recent studies (Team et al., 2025; Cuadron et al., 2025) have discovered an intrinsic overthinking phenomenon in reasoning models, where these models persistently produce verbose rationalizations. This tendency manifests as the inclusion of irrelevant contextual information and unnecessary reflective behaviors. Such information and behaviors not only inefficiently consume computational resources but also compromise reasoning accuracy by causing models to deviate from valid logical pathways to incorrect conclusions.

To address these issues, recent studies (Luo et al., 2025; Song et al., 2025; Yu et al., 2025; Liu et al., 2025; Fatemi et al., 2025) are researching efficient reasoning methodologies based on the GRPO algorithm for training the model to produce more concise CoT responses, and have discovered a trade-off between the CoT response length and model reasoning capabilities in most cases, i.e., the shorter the

length, the worse the performance. It is understandable that achieving efficient reasoning, improving ability via a more concise CoT, is inherently more challenging. This is in contrast to boosting performance by merely increasing the response length, as the former requires significantly higher model capabilities. Therefore, we highlight the critical importance of the timing for optimizing response length when training using GRPO-based algorithms. Adhering to the "walk before you run" principle, we consider that during training, response length optimization is only enabled when all rollouts for a training sample are correct.

Motivated by these findings, in this paper, we propose GLPO, a simple yet effective reinforcement learning algorithm for concise LLM reasoning. Unlike prior methods that rely on complex multi-stage pipelines or direct constraints on the context window to optimize reasoning quality, GLPO operates as a unified, single-stage framework. Our key innovation lies in utilizing the remaining context length as a primary reward signal. Specifically, rather than explicitly penalizing generation length, we design the reward mechanism to assign higher values to correct reasoning paths that leave a larger portion of the context window unused. This approach naturally incentivizes the model to discover more efficient solutions, treating the residual context capacity as a positive reinforcement for conciseness. Experiments demonstrate that GLPO effectively reduces response length while simultaneously improving accuracy across benchmarks such as AIME 2024, AMC 2023, MATH-500, Minerva, and Olympiad datasets.

2 Preliminary

2.1 Proximal Policy Optimization

Proximal Policy Optimization (PPO) (Schulman et al., 2017) is one of the policy gradient methods that introduces a clipped surrogate objective for policy optimization. By using clipping to constrain policy updates within a proximal region of the previous policy, PPO stabilizes training and improves sample efficiency. Specifically, PPO updates the policy by maximizing the following objective:

$$\begin{aligned}\mathcal{J}_{\text{PPO}}(\theta) = \mathbb{E}_{q\sim\mathcal{D},o_{\le t}\sim\pi_{\theta_{\text{old}}}(\cdot|q)} \\ \min[\frac{\pi_\theta(o_t \mid q,o_{<t})}{\pi_{\theta_{\text{old}}}(o_t \mid q,o_{<t})}\hat{A}_t, \\ \text{clip}(\frac{\pi_\theta(o_t \mid q,o_{<t})}{\pi_{\theta_{\text{old}}}(o_t \mid q,o_{<t})}, 1-\varepsilon, 1+\varepsilon)\hat{A}_t],\end{aligned} \tag{1}$$

where $\pi_{\theta_{\text{old}}}$ is the policy before the update, ε is the clipping hyper-parameter, q indicates the question from the data distribution $\mathcal{D}$, and $\hat{\mathbf{A}}_t$ is an estimator of the advantage function of the t-th token. Here, a standard and traditional way to estimate $\hat{\mathbf{A}}_t$ is to compute the Generalized Advantage Estimation (GAE) (Schulman et al., 2017) with a learned value model. However, in the context of LLM RL scaling, learning the value model is computationally expensive, so methods that estimate $\hat{\mathbf{A}}_t$ without a learned value model are practically preferred.

2.2 Group Relative Policy Optimization

Group Relative Policy Optimization (GRPO) (Shao et al., 2024) introduces a policy gradient framework that eliminates the reliance on explicit value function by utilizing comparative advantage estimation within a group of responses. Specifically, GRPO first samples a group of candidate outputs $\{o_1, o_2, \ldots, o_{\text{G}}\}$ per question and computes their returns $\mathbf{r} = \{r_1, r_2, \ldots, r_{\text{G}}\}$, then calculates the advantage of the i-th output o_i as,

$$\hat{\mathbf{A}}_i = \frac{r_i - \text{mean}(\{r_1, r_2, \cdots, r_{\text{G}}\})}{\text{std}(\{r_1, r_2, \cdots, r_{\text{G}}\})}. \tag{2}$$

Similar to PPO, GRPO uses a clipped objective with KL penalty and optimizes the policy model π_θ by maximizing the following objective:

$$\begin{aligned}\mathcal{J}_{\text{GRPO}}(\theta) = \mathbb{E}_{q\sim\mathcal{D},\{o_i\}_{i=1}^{\text{G}}\sim\pi_{\theta_{\text{old}}}(\cdot|q)} \\ \frac{1}{\text{G}}\sum_{i=1}^{\text{G}}\{\min[\tau_i(\theta)\hat{\mathbf{A}}_i, \text{clip}(\tau_i(\theta), 1-\varepsilon, \\ 1+\varepsilon)\hat{\mathbf{A}}_i] - \beta\mathbb{D}_{\text{KL}}[\pi_\theta||\pi_{\text{ref}}]\},\end{aligned} \tag{3}$$

where

$$\tau_i = \frac{\pi_\theta(o_i|q)}{\pi_{\theta_{\text{old}}}(o_i|q)}. \tag{4}$$

Here, π_{ref} represents the reference model and the term $\mathbb{D}_{\text{KL}}(\pi_\theta||\pi_{\text{ref}})$ indicates a KL penalty term to limit how much the trained model π_θ can deviate from the reference model π_{ref}.

3 Methodology

3.1 Group Length Policy Optimization

Our primary goal is to enable LLM to generate a more concise CoT output without compromising the model's performance. To this end, we propose Group Length Policy Optimization guided by the principle of "*Aim for 100% accuracy first; speed comes with mastery*". Recent studies (Luo et al.,

Table 1: Training Template. **{question}** will be replaced with the question during the training and testing.

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>. Please reason step by step, and put your final answer within \boxed{}. User: **{question}** Assistant:

2025; Song et al., 2025) have found that the reasoning response length is not strongly correlated with the correctness of the answer; that is, a long CoT reasoning response does not necessarily represent a correct result, and a short CoT reasoning response does not necessarily represent an incorrect one. On the contrary, the correct CoT reasoning responses are usually shorter in length, while incorrect reasoning responses tend to be longer.

Based on the above analysis, we reshape the reward function in GRPO. When the model's rollout results for a question are all correct, we further optimize the model's reasoning length for that question by using the remaining maximum response length as a reward (under the specified context length, the more remaining context length, the higher the reward), as calculated below,

$$\mathcal{J}_{\text{GRPO}}(\theta) = \mathbb{E}_{q\sim\mathcal{D},\{o_i\}_{i=1}^{\text{G}}\sim\pi_{\theta_{\text{old}}}(\cdot|q)} \frac{1}{\text{G}}\sum_{i=1}^{\text{G}}\{\min[\tau_i(\theta)\mathbf{A}_{l_i}, \text{clip}(\tau_i(\theta), 1-\varepsilon_l, 1+\varepsilon_h)\mathbf{A}_{l_i}] - \beta\mathbb{D}_{\text{KL}}[\pi_\theta||\pi_{\text{ref}}] + \alpha\mathbb{H}(\pi_\theta)\}, \quad (5)$$

where $\alpha\mathbb{H}(\pi_\theta)$ indicates the Entropy bonus for enhancing the model's exploration capabilities. Here, $\mathbf{A}_{l_i}$ denotes the length-aware advantage value and can be calculated as follows,

$$\mathbf{A}_{l_i} = \frac{r_{l_i} - \text{mean}(\{r_{l_1}, r_{l_2}, \cdots, r_{l_\text{G}}\})}{\text{std}(\{r_{l_1}, r_{l_2}, \cdots, r_{l_\text{G}}\})}. \quad (6)$$

Here, r_{l_i} indicates the reward with length penalty and is computed as,

$$r_{l_i} = r_i + \lambda\hat{\mathcal{L}}_i, \quad (7)$$

$$\hat{\mathcal{L}}_i = \begin{cases} \frac{\mathcal{L}_{\text{Max}}-\mathcal{L}_i}{\mathcal{L}_{\text{Max}}}, & \text{if } \sum_{i=1}^{\text{G}} r_i = \text{G} \\ 0, & \text{if } \sum_{i=1}^{\text{G}} r_i \neq \text{G}. \end{cases} \quad (8)$$

Here, $\mathcal{L}_i$ is the length of the i-th response and $\mathcal{L}_{\text{Max}}$ indicates the max output length. ε_l and ε_h indicate the clipping range.

3.2 Rule-based Reward Model

Using a trained reward model typically introduces the issue of reward hacking. To mitigate this issue, we directly adopt the final accuracy from a verifiable task as the outcome reward, calculated according to the following rule:

$$r_i(o_i, a) = \begin{cases} 1, & \text{if is\_equivalent}(o_i, a) \\ 0, & \text{if not is\_equivalent}(o_i, a) \end{cases} \quad (9)$$

where a indicates the ground-truth answer and o_i contains the predicted answer. Additionally, it is important to note that the trained model must adhere strictly to the training prompt by generating the chain-of-thought within the <think></think> tags and subsequently presenting the answer within the <answer></answer> tags with the boxed tag.

3.3 Training Dataset Curation

To curate high-quality data for RL, we include challenging problems from DeepScaleR (Luo et al., 2025), DAPO-Math-17K (Yu et al., 2025), and MATH (Hendrycks et al., 2021) to enhance problem difficulty and diversity in our data mixture:

- **DeepScaleR**[1], which contains approximately 40K unique mathematics-specific problem-answer pairs collected from AIME (1984-2023), AMC (prior to 2023), Omni-MATH, and Still datasets (Lewkowycz et al., 2022; Gao et al., 2024; Min et al., 2024).
- **DAPO-Math-17K**[2], which contains approximately 17K problem-answer pairs, each paired with an integer as the answer. DAPO-Math-17K was compiled from the Art of Problem Solving (AoPS[3]) website and official

[1] https://huggingface.co/datasets/agentica-org/DeepScaleR-Preview-Dataset

[2] https://huggingface.co/datasets/BytedTsinghua-SIA/DAPO-Math-17k

[3] https://artofproblemsolving.com/

Model	AIME 2024	MATH-500	AMC 2023	Minerva Math	Olympiad Bench	Avg. Score
Qwen2.5-1.5B-Base (Yang et al., 2024a)†	0.0	3.3	2.5	1.8	1.5	1.82
Qwen2.5-Math-1.5B-Base (Yang et al., 2024b)†	11.3	51.7	44.0	11.3	26.0	28.9
Qwen2.5-1.5B-Instruct (Yang et al., 2024a)†	1.3	57.5	26.2	19.4	20.3	24.9
Qwen2.5-Math-1.5B-Instruct (Yang et al., 2024b)†	12.0	74.7	26.7	35.0	37.9	37.3
Oat-Zero-1.5B (Liu et al., 2025)†	16.0	73.5	52.5	26.3	37.2	41.1
Qwen2.5-Math-1.5B-GRPO ($\varepsilon_h = 0.28, \alpha = 10^{-3}$)	20.0	75.4	57.8	23.2	36.5	42.6
GLPO-Zero-1.5B	23.3	73.8	62.8	20.8	37.7	43.7

Table 2: Overall performance on five competition-level reasoning benchmarks. Our models outperform prior state-of-the-art models (1.5B) with zero RL paradigm. † indicate the results from (Hochlehnert et al., 2025).

Model	AIME 2024	MATH-500	AMC 2023	Minerva Math	Olympiad Bench	Avg. Score
Qwen2.5-7B-Base (Yang et al., 2024a)†	3.3	64.6	30.0	25.7	29.0	30.5
Qwen2.5-Math-7B-Base (Yang et al., 2024b)†	20.7	64.3	56.2	17.3	29.0	37.5
Qwen2.5-7B-Instruct (Yang et al., 2024a)†	12.3	77.1	52.8	34.9	38.7	43.2
Qwen2.5-Math-7B-Instruct (Yang et al., 2024b)†	15.7	82.9	67.0	35.0	41.3	48.4
Eurus-2-7B-PRIME (Cui et al., 2025)†	17.8	80.1	63.0	37.5	43.9	48.5
Open-Reasoner-Zero-7B (Hu et al., 2025)†	19.7	83.9	59.5	31.6	47.6	48.5
SimpleRL-Zero-7B (Zeng et al., 2025)†	14.0	77.9	58.0	33.0	39.0	44.4
SimpleRL-Zero-Math-7B (Zeng et al., 2025)†	22.7	76.9	62.2	30.1	39.3	46.2
Oat-Zero-7B (Liu et al., 2025)†	28.0	79.4	66.2	34.4	43.8	50.4
Qwen2.5-Math-7B-GRPO ($\varepsilon_h = 0.28, \alpha = 10^{-3}$)	48.3	83.9	76.0	28.5	45.0	56.3
GLPO-Zero-7B	48.2	84.1	76.3	29.7	45.8	56.8

Table 3: Overall performance on five competition-level reasoning benchmarks. Our models outperform prior state-of-the-art models (7B) with zero RL paradigm. † indicate the results from (Hochlehnert et al., 2025).

competition websites using a combination of web scraping and manual annotation.

- **MATH**[4] (Level 3-5), which contains approximately 8K problem-answer pairs. Each problem has a step-by-step solution which can be used to teach models to generate explanations.

After obtaining the above datasets, we employ Math-Verify[5] to re-extract answers from the provided textual solutions, selecting only those cases where the extracted answer matches the corresponding answer in the dataset. We discard any samples that are empty, incomplete, or duplicates. Finally, we obtain approximately 59K reasoning problems as the training dataset. It should be noted that in the first stage, we use the 59K data to incentivize the model's reasoning ability. Still, in the second stage, we use the MATH (Level 3-5) data as the training set to optimize the model's reasoning length.

4 Experiments

4.1 Training Details

We adopt Qwen2.5-Math-1.5B and Qwen2.5-Math-7B (Yang et al., 2024b) as the base language model and leverage verl (Sheng et al., 2025) to train our models. During training, we utilize the Adam optimizer with a constant learning rate of 1×10^{-6}. We leverage a batch size of 128 with each question generating 32 rollouts, the maximum response length is set to 3,072 tokens, and training is conducted using mini-batches of size 128. As for the Clip-Higher, similar to the prior work (Yu et al., 2025), we set the clipping parameter ε_l to 0.2 and ε_h to 0.28, which effectively balance the trade-off between exploration and exploitation for RL.

4.2 Evaluation Benchmarks

Similar to Liu et al. (2025); Song et al. (2025), the performance of our models is evaluated on a diverse suite of competition-level benchmarks including AIME 2024[6] (comprises 30 challenge problems), AMC 2023[7] (contains 40 mathematical problems, covering algebra, geometry, number theory, and combinatorics), Minerva Math (Lewkowycz et al., 2022), MATH-500 (Hendrycks et al., 2021), and OlympaidBench (He et al., 2024).

[4] https://huggingface.co/datasets/EleutherAI/hendrycks_math

[5] https://github.com/huggingface/Math-Verify

[6] https://huggingface.co/datasets/AI-MO/aimo-validation-aime

[7] https://huggingface.co/datasets/AI-MO/aimo-validation-amc

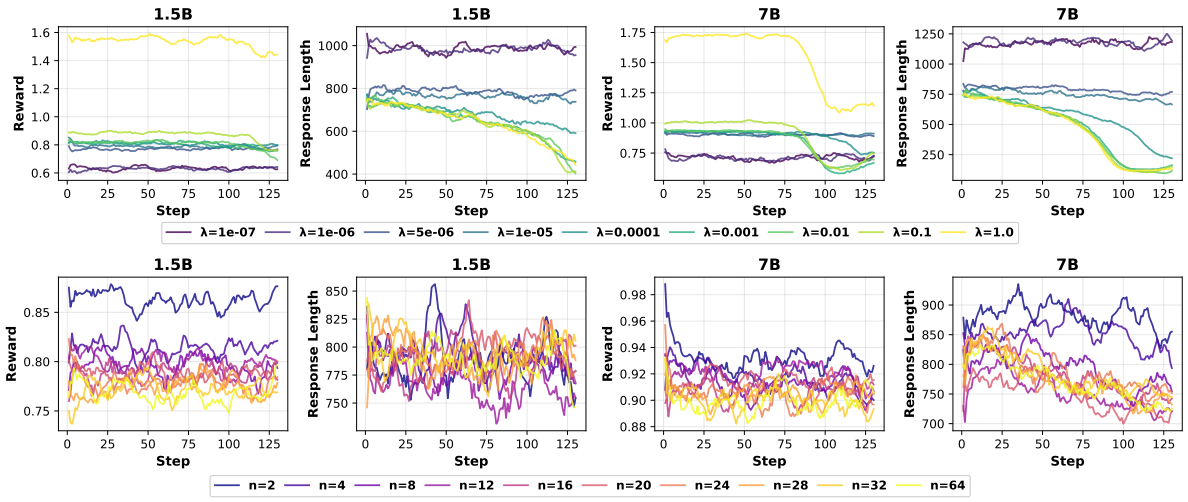

Figure 1: A detailed evaluation of accuracy and response length throughout the training steps.

4.3 Evaluation Setup

We adopt the Pass@k evaluation metric, reporting Pass@1 accuracy computed with a non-zero sampling temperature. Therefore, we set the maximum response length to 3,072 tokens. Specifically, we select a temperature of 0.6 combined with a top-p value of 0.95 to generate multiple responses (typically 32 samples) for each query. The used training template is shown in Table 1.

4.4 Baselines

We conduct evaluations against recent baselines with zero RL paradigm, including Qwen2.5-1.5B and 7B (Yang et al., 2024a), Qwen2.5-Math-1.5B and 7B (Yang et al., 2024b), SimpleRL-Zero (Zeng et al., 2025), Open-Reasoner-Zero-7B (Hu et al., 2025), Eurus-2-7B-PRIME (Cui et al., 2025), and Oat-Zero-1.5B and 7B (Liu et al., 2025).

4.5 Main Results

The experimental results reported in Table 2 and Table 3 demonstrate that our proposed GLPO framework significantly outperforms existing baselines under the zero RL paradigm across five widely recognized competition-level reasoning benchmarks.

Performance on 1.5B Scale. As shown in Table 2, GLPO-Zero-1.5B achieves an average accuracy of 43.7%, surpassing all baseline models including Oat-Zero-1.5B (41.1%) and the vanilla GRPO variant (42.6%). Notably, our model demonstrates substantial improvements on challenging benchmarks such as AIME 2024 (23.3%) and AMC 2023 (62.8%), representing gains of 45.6% and 8.7% relative to Oat-Zero-1.5B, respectively. Compared to the base model Qwen2.5-Math-1.5B-Base, GLPO achieves a remarkable 51.2% relative improvement in average accuracy.

Performance on 7B Scale. Table 3 presents results for 7B-scale models, where GLPO-Zero-7B attains the highest average accuracy of 56.8%, outperforming all prior methods including Oat-Zero-7B (50.4%) and the vanilla GRPO variant (56.3%). Our model achieves particularly strong results on AIME 2024 (48.2%) and AMC 2023 (76.3%), demonstrating superior mathematical reasoning capabilities. Compared to the base model Qwen2.5-Math-7B-Base, GLPO achieves a 51.5% relative improvement in average accuracy.

Response Length Reduction. Table 4 illustrates the training dynamics of GLPO across the five benchmarks. Throughout training, the average accuracy on each benchmark remains stable without noticeable degradation, while the response length consistently decreases. For the 1.5B model, we observe length reductions of 14%, 11%, 22%, 12%, and 0.03% on Olympiad Bench, MATH-500, AMC 2023, Minerva, and AIME 2024, respectively. For the 7B model, the reductions are 6%, 6%, 5%, 5%, and 4% on the corresponding benchmarks. These results demonstrate that our training approach successfully maintains model accuracy while generating more concise and efficient responses.

4.6 Ablation Study

We conduct ablation studies on two key hyperparameters of GLPO: the length penalty coefficient λ and the number of rollouts n. Figure 1 shows the training dynamics of reward and response length

Model	Config	OlyBench	MATH	AIME	AMC	Minerva	Avg Score	Avg Len
1.5B	Step 0 (Baseline)	0.365	0.754	0.200	0.578	0.232	0.426	1001
	λ=1e-4, Step=100	0.377	0.738	0.233	0.628	0.208	0.437	891
	$\triangle$	+0.012	-0.016	+0.033	+0.050	-0.024	+0.011	-110
	λ=1, Step=120	0.319	0.712	0.267	0.506	0.195	0.400	676
	$\triangle$	-0.046	-0.042	+0.067	-0.072	-0.037	-0.026	-325
7B	Step 0 (Baseline)	0.450	0.839	0.495	0.760	0.285	0.566	951
	λ=1e-4, Step=20	0.458	0.841	0.482	0.763	0.297	0.568	903
	$\triangle$	+0.008	+0.002	-0.013	+0.003	+0.012	+0.002	-48
	λ=1e-4, Step=90	0.432	0.823	0.422	0.699	0.286	0.532	623
	$\triangle$	-0.018	-0.016	-0.073	-0.061	+0.001	-0.034	-328

Table 4: GLPO Length Penalty Optimization Results

for both 1.5B and 7B models.

For the length penalty ablation (top row), we observe a clear trade-off between reward maximization and response length reduction. Larger λ values (e.g., $\lambda = 1$) lead to more aggressive length reduction but at the cost of lower rewards, while smaller values (e.g., $\lambda = 1e-6$) maintain higher rewards but with minimal length compression. The optimal balance is achieved at $\lambda = 1e-4$, where both models exhibit substantial length reduction while preserving or even improving task performance. Specifically, the 1.5B model reduces average response length from 1001 to 891 tokens (11% reduction) at step 100 while improving accuracy from 0.426 to 0.437. The 7B model achieves a 5% length reduction (951→903 tokens) at step 20 with a slight accuracy improvement (0.566→0.568). Notably, the training curves show that response length decreases monotonically across all λ values, while reward trajectories diverge significantly depending on the penalty strength.

For the rollout number ablation (bottom row), we find that increasing n generally leads to more stable training and higher final rewards. However, the effect on response length is less pronounced compared to the length penalty. The 7B model with n=28 achieves the best balance, reaching competitive accuracy while maintaining reasonable response lengths. Interestingly, the 1.5B model shows less sensitivity to the rollout number, suggesting that smaller models may require fewer samples for effective policy gradient estimation.

The results demonstrate that GLPO can effectively compress model outputs without significant performance degradation. When prioritizing length reduction, configurations with higher λ values (e.g., $\lambda = 1$ at step 120 for 1.5B) achieve up to 32.5% length reduction with only 6.1% accuracy drop. This favorable trade-off suggests that much of the verbosity in model responses is redundant and can be eliminated through targeted optimization.

4.7 Analysis of Length Penalty Optimization

Table 4 presents an ablation study investigating the impact of the length penalty coefficient (λ) and training steps on both model performance and response conciseness. The results reveal several key insights regarding the balance between computational efficiency and reasoning quality:

For the 1.5B model, applying a moderate length penalty ($\lambda = 10^{-4}$) for 100 steps results in a "win-win" scenario. The model achieves a 1.1% increase in average accuracy (rising from 42.6% to 43.7%) while simultaneously reducing the average response length by 110 tokens. Similarly, the 7B model (at Step 20) maintains superior accuracy (56.8%) while reducing the response length by 48 tokens. This suggests that baseline models often generate redundant or verbose reasoning chains; by incentivizing the model to maximize the remaining context length, GLPO encourages more direct and precise reasoning paths without compromising correctness.

The experiments also demonstrate the boundaries of length optimization. When the penalty is too aggressive (e.g., $\lambda = 1$ for 1.5B) or the optimization is prolonged (e.g., Step 90 for 7B), the model achieves drastic reductions in length (approximately -325 tokens for 1.5B and -328 tokens for 7B). However, this comes at the cost of performance degradation, with average scores dropping by 2.6% and 3.4%, respectively. This indicates that while removing redundancy is beneficial, excessive compression forces the model to skip necessary

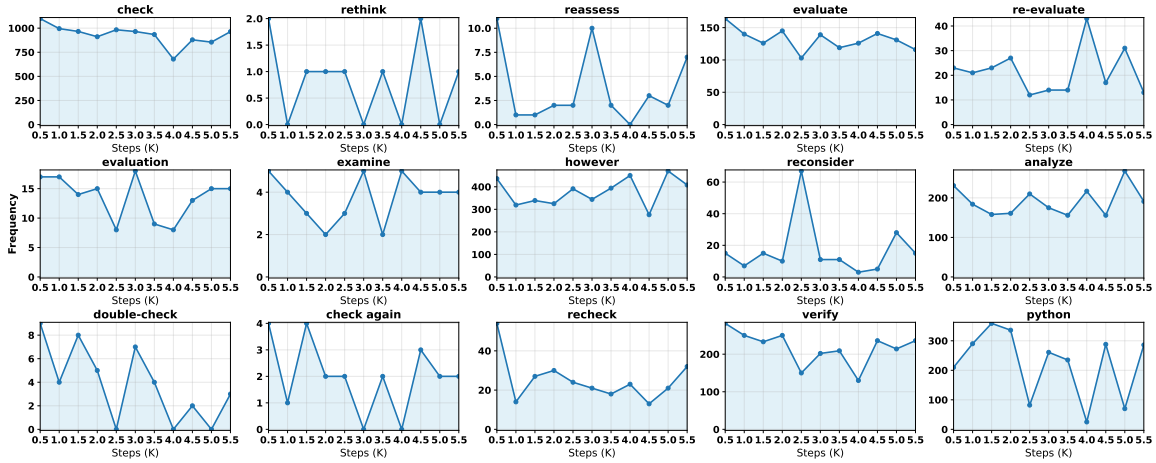

Figure 2: Count of keyword occurrences out of 14,022 responses (1558 questions × 11 test times).

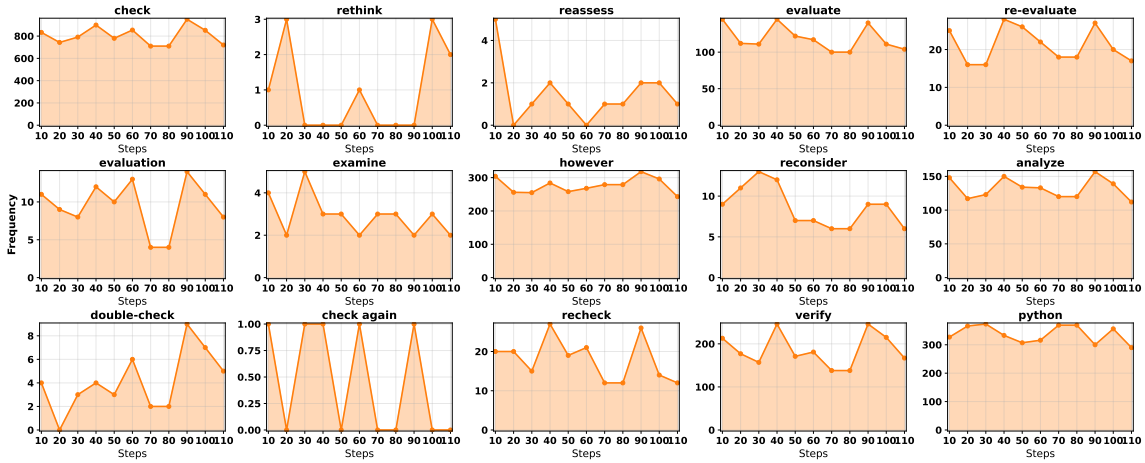

Figure 3: Count of keyword occurrences out of 15,580 responses (1558 questions × 11 test times).

reasoning steps, leading to incorrect answers.

Both 1.5B and 7B models exhibit consistent behavior: a mild length incentive improves or maintains accuracy by pruning unnecessary tokens, while an excessive incentive harms reasoning. Notably, the 1.5B model appears to benefit more significantly from this optimization in terms of accuracy gains, whereas the 7B model primarily maintains its high performance while becoming more efficient. In summary, GLPO effectively allows for the control of response length. By carefully tuning the reward for remaining context, it is possible to obtain a model that is both more accurate and significantly more efficient than the baseline.

4.8 Analysis of Reasoning Patterns

Inspired by prior work that observes the model's reflective behavior by constructing a keyword pool, we have built a carefully selected keyword pool to observe changes in the thinking patterns of the responses during training. In our experiment, the keyword pool is limited to: check, rethink, reassess, evaluate, re-evaluate, evaluation, examine, however, reconsider, analyze, double-check, check again, recheck, verify, and wait. Then, we present the occurrences of various keywords in the responses generated by different training stages and steps in Figure 2 and Figure 3. Interestingly, when comparing the first and second stages, the frequency with which the model uses code to verify results has significantly increased (as reflected in the frequency of the keyword "python"). The model may have discovered that verifying results by writing code is more efficient. Meanwhile, keywords like "re-check" have decreased relatively, and other keywords have remained unchanged.

4.9 Case Study

An interesting observation is that python code is used for verification during mathematical problem

Question (a)

An integer, whose decimal representation reads the same left to right and right to left, is called symmetrical. For example, the number 513151315 is symmetrical, while 513152315 is not. How many nine-digit symmetrical numbers exist such that adding the number 11000 to them leaves them symmetrical?

Response

...

Let's write a Python code to iterate through all possible nine-digit symmetrical numbers and check if adding 11000 results in another symmetrical number.

```python
count = 0
for a1 in range(1, 10):  # a1 ranges from 1 to 9
    for a2 in range(10):  # a2 ranges from 0 to 9
        for a3 in range(10):  # a3 ranges from 0 to 9
            for a4 in range(10):  # a4 ranges from 0 to 9
                for a0 in range(10):  # a0 ranges from 0 to 9
                    N = int(f"{a1}{a2}{a3}{a4}{a3}{a2}{a1}{a0}{a0}")
                    M = N + 11000
                    M_str = str(M)
                    if M_str == M_str[::-1]:
                        count += 1
print(count)
```

```output
8100
```

...

Question (b)

For positive integers $a, b, a \uparrow \uparrow b$ is defined as follows: $a \uparrow \uparrow 1=a$, and $a \uparrow \uparrow b=a^{a \uparrow \uparrow(b-1)}$ if $b>1$. Find the smallest positive integer n for which there exists a positive integer a such that $a \uparrow \uparrow 6 \not \equiv a \uparrow \uparrow 7$ $\bmod n$.

Response

...

Step-by-Step Check

...

Continue this process programmatically for efficiency:

We write a Python script to automate the checking for the smallest \(n \):

```python
import sympy as sp

def tetration_mod(a, b, n):
    if b == 1:
        return a % n
    else:
        exp = tetration_mod(a, b-1, n)
        return pow(a, exp, n)

def find_smallest_n():
    for n in range(2, 100):
        for a in range(1, n):
            if tetration_mod(a, 6, n) != tetration_mod(a, 7, n):
                return n
    return None

smallest_n = find_smallest_n()
print(smallest_n)
```

```output
2
```

...

Question (c)

Given that the sequence starts with 2 and alternates by adding 2 between consecutive terms, find the 30th term of this arithmetic sequence.

Response

<think>a30 = 2 + 29*2 = 60</think>
<answer>\boxed{60}</answer>

Figure 4: Illustration of cases.

solving, e.g., Questions (a) and (b) in Figure 4. Specifically, for Question (a), the model utilizes program code to calculate the answer. For Question (b), the model first presents the solution process through mathematical reasoning and then spontaneously writes program code to verify the correctness of the approach. Such cases illustrate how models employ procedural reasoning to self-correct and engage in subsequent attempts.

5 Related Work

Recent advances in reinforcement learning have significantly enhanced the reasoning capabilities of large language models. A pivotal development in this domain is OpenAI's o1 (Jaech et al., 2024), which uses large-scale RL training to promote CoT reasoning. This approach has resulted in notable improvements in complex mathematics and coding benchmarks. DeepSeek-R1 (Guo et al., 2025) demonstrates that pure RL post-training via GRPO, without the need for supervised warm-up, can directly induce robust reasoning abilities. Remarkably, this kind of method not only achieves performance competitive with o1 but also exhibits emergent behaviors such as self-verification and multi-step planning. This paradigm shift significantly reduces memory and computational overhead compared to earlier GRPO implementations (Hu et al., 2025; Zeng et al., 2025; Face, 2025), all while maintaining competitive performance levels.

Recent algorithmic variants have focused on enhancing training efficiency (Luo et al., 2025; Team et al., 2025; Song et al., 2025; Yu et al., 2025; Liu et al., 2025; Fatemi et al., 2025; Zeng et al., 2025; Wen et al., 2025), yet they preserve GRPO's core methodology of parallel CoT sampling across groups. These advancements collectively contribute to more efficient and robust training methodologies for LLMs, thereby enhancing their reasoning capabilities and performance on complex tasks.

6 Conclusion

In this paper, we propose GLPO, which introduces a simple yet effective reinforcement learning framework. *Importantly, we innovatively propose that during training, response length optimization is only triggered when all rollouts for a given training sample are correct. This embodies the "walk before you run" principle.* Experiments demonstrate that GLPO consistently achieves the best efficiency-accuracy synergistic improvement, significantly outperforming existing efficient reasoning methods across five benchmarks.

7 Limitations

Due to limited resources, this paper verifies the effectiveness of the proposed method, GLPO, only on a 7B language model. Generally, validating its effectiveness on models of varying sizes is a worthwhile direction for future research. Furthermore, in this paper, we investigate the influence of using complexity-aware training data by employing the simplest separation method to validate the efficacy of separating the training data by complexity, and achieves significant results. If more sophisticated separation methods were adopted, achieving even more promising results might be possible.

Training over multiple stages, rather than in a single training stage, involves more than changes in parameters like context length; it also fundamentally alters the reference policy. In a multi-stage training strategy, the KL penalty imposed by the reference policy on the model is gradually relaxed, which allows the trained model to explore a broader range of solutions. Delving into dynamic control of context lengths or implementing a dynamic KL penalty may be valuable directions.

References

Alejandro Cuadron, Dacheng Li, Wenjie Ma, Xingyao Wang, Yichuan Wang, Siyuan Zhuang, Shu Liu, Luis Gaspar Schroeder, Tian Xia, Huanzhi Mao, Nicholas Thumiger, Aditya Desai, Ion Stoica, Ana Klimovic, Graham Neubig, and Joseph E. Gonzalez. 2025. The danger of overthinking: Examining the reasoning-action dilemma in agentic tasks. *Preprint*, arXiv:2502.08235.

Ganqu Cui, Lifan Yuan, Zefan Wang, Hanbin Wang, Wendi Li, Bingxiang He, Yuchen Fan, Tianyu Yu, Qixin Xu, Weize Chen, and 1 others. 2025. Process reinforcement through implicit rewards. *arXiv preprint arXiv:2502.01456*.

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

Mehdi Fatemi, Banafsheh Rafiee, Mingjie Tang, and Kartik Talamadupula. 2025. Concise reasoning via reinforcement learning. *arXiv preprint arXiv:2504.05185*.

Bofei Gao, Feifan Song, Zhe Yang, Zefan Cai, Yibo Miao, Qingxiu Dong, Lei Li, Chenghao Ma, Liang Chen, Runxin Xu, Zhengyang Tang, Benyou Wang, Daoguang Zan, Shanghaoran Quan, Ge Zhang, Lei Sha, Yichang Zhang, Xuancheng Ren, Tianyu Liu, and Baobao Chang. 2024. Omni-math: A universal olympiad level mathematic benchmark for large language models. *CoRR*, abs/2410.07985.

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

Chaoqun He, Renjie Luo, Yuzhuo Bai, Shengding Hu, Zhen Thai, Junhao Shen, Jinyi Hu, Xu Han, Yujie Huang, Yuxiang Zhang, and 1 others. 2024. Olympiadbench: A challenging benchmark for promoting agi with olympiad-level bilingual multimodal scientific problems. In *Proceedings of the 62nd Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, pages 3828–3850.

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*.

Andreas Hochlehnert, Hardik Bhatnagar, Vishaal Udandarao, Samuel Albanie, Ameya Prabhu, and Matthias Bethge. 2025. A sober look at progress in language model reasoning: Pitfalls and paths to reproducibility. *Preprint*, arXiv:2504.07086.

Jingcheng Hu, Yinmin Zhang, Qi Han, Daxin Jiang, Xiangyu Zhang, and Heung-Yeung Shum. 2025. Open-reasoner-zero: An open source approach to scaling up reinforcement learning on the base model. *Preprint*, arXiv:2503.24290.

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*.

Aitor Lewkowycz, Anders Andreassen, David Dohan, Ethan Dyer, Henryk Michalewski, Vinay Ramasesh, Ambrose Slone, Cem Anil, Imanol Schlag, Theo Gutman-Solo, and 1 others. 2022. Solving quantitative reasoning problems with language models. *Advances in Neural Information Processing Systems*, 35:3843–3857.

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

Michael Luo, Sijun Tan, Justin Wong, Xiaoxiang Shi, William Y. Tang, Manan Roongta, Colin Cai, Jeffrey Luo, Tianjun Zhang, Li Erran Li, Raluca Ada Popa, and Ion Stoica. 2025. Deepscaler: Surpassing o1-preview with a 1.5b model by scaling rl. `https://github.com/agentica-project/deepscaler`. Notion Blog.

Yingqian Min, Zhipeng Chen, Jinhao Jiang, Jie Chen, Jia Deng, Yiwen Hu, Yiru Tang, Jiapeng Wang, Xiaoxue Cheng, Huatong Song, Wayne Xin Zhao, Zheng Liu, Zhongyuan Wang, and Ji-Rong Wen.

2024. Imitate, explore, and self-improve: A reproduction report on slow-thinking reasoning systems. *Preprint*, arXiv:2412.09413.

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

John Schulman, Filip Wolski, Prafulla Dhariwal, Alec Radford, and Oleg Klimov. 2017. Proximal policy optimization algorithms. *arXiv preprint arXiv:1707.06347*.

Zhihong Shao, Peiyi Wang, Qihao Zhu, Runxin Xu, Junxiao Song, Xiao Bi, Haowei Zhang, Mingchuan Zhang, Y. K. Li, Y. Wu, and Daya Guo. 2024. Deepseekmath: Pushing the limits of mathematical reasoning in open language models. *Preprint*, arXiv:2402.03300.

Guangming Sheng, Chi Zhang, Zilingfeng Ye, Xibin Wu, Wang Zhang, Ru Zhang, Yanghua Peng, Haibin Lin, and Chuan Wu. 2025. Hybridflow: A flexible and efficient rlhf framework. In *Proceedings of the Twentieth European Conference on Computer Systems*, page 1279–1297. ACM.

Charlie Snell, Jaehoon Lee, Kelvin Xu, and Aviral Kumar. 2024. Scaling llm test-time compute optimally can be more effective than scaling model parameters. *Preprint*, arXiv:2408.03314.

Mingyang Song, Mao Zheng, Zheng Li, Wenjie Yang, Xuan Luo, Yue Pan, and Feng Zhang. 2025. Fastcurl: Curriculum reinforcement learning with progressive context extension for efficient training r1-like reasoning models. *Preprint*, arXiv:2503.17287.

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*.

Jason Wei, Xuezhi Wang, Dale Schuurmans, Maarten Bosma, brian ichter, Fei Xia, Ed H. Chi, Quoc V Le, and Denny Zhou. 2022. Chain of thought prompting elicits reasoning in large language models. In *Advances in Neural Information Processing Systems*.

Liang Wen, Yunke Cai, Fenrui Xiao, Xin He, Qi An, Zhenyu Duan, Yimin Du, Junchen Liu, Lifu Tang, Xiaowei Lv, and 1 others. 2025. Light-r1: Curriculum sft, dpo and rl for long cot from scratch and beyond. *arXiv preprint arXiv:2503.10460*.

An Yang, Baosong Yang, Beichen Zhang, Binyuan Hui, Bo Zheng, Bowen Yu, Chengyuan Li, Dayiheng Liu, Fei Huang, Haoran Wei, Huan Lin, Jian Yang, Jianhong Tu, Jianwei Zhang, Jianxin Yang, Jiaxi Yang, Jingren Zhou, Junyang Lin, Kai Dang, and 22 others. 2024a. Qwen2.5 technical report. *arXiv preprint arXiv:2412.15115*.

An Yang, Beichen Zhang, Binyuan Hui, Bofei Gao, Bowen Yu, Chengpeng Li, Dayiheng Liu, Jianhong Tu, Jingren Zhou, Junyang Lin, Keming Lu, Mingfeng Xue, Runji Lin, Tianyu Liu, Xingzhang Ren, and Zhenru Zhang. 2024b. Qwen2.5-math technical report: Toward mathematical expert model via self-improvement. *Preprint*, arXiv:2409.12122.

Edward Yeo, Yuxuan Tong, Morry Niu, Graham Neubig, and Xiang Yue. 2025. Demystifying long chain-of-thought reasoning in llms. *Preprint*, arXiv:2502.03373.

Qiying Yu, Zheng Zhang, Ruofei Zhu, Yufeng Yuan, Xiaochen Zuo, Yu Yue, Tiantian Fan, Gaohong Liu, Lingjun Liu, Xin Liu, Haibin Lin, Zhiqi Lin, Bole Ma, Guangming Sheng, Yuxuan Tong, Chi Zhang, Mofan Zhang, Wang Zhang, Hang Zhu, and 16 others. 2025. Dapo: An open-source llm reinforcement learning system at scale. *Preprint*, arXiv:2503.14476.

Weihao Zeng, Yuzhen Huang, Qian Liu, Wei Liu, Keqing He, Zejun Ma, and Junxian He. 2025. Simplerl-zoo: Investigating and taming zero reinforcement learning for open base models in the wild. *Preprint*, arXiv:2503.18892.