S-GRPO: Early Exit via Reinforcement Learning in Reasoning Models

Muzhi Dai[1]*, **Chenxu Yang**[2]*, **Qingyi Si**[1]†,
[1]Huawei Technologies Co., Ltd.
[2]Institute of Information Engineering, Chinese Academy of Sciences, Beijing, China
mzdai666@gmail.com, yangchenxu@iie.ac.cn, siqingyi@huawei.com

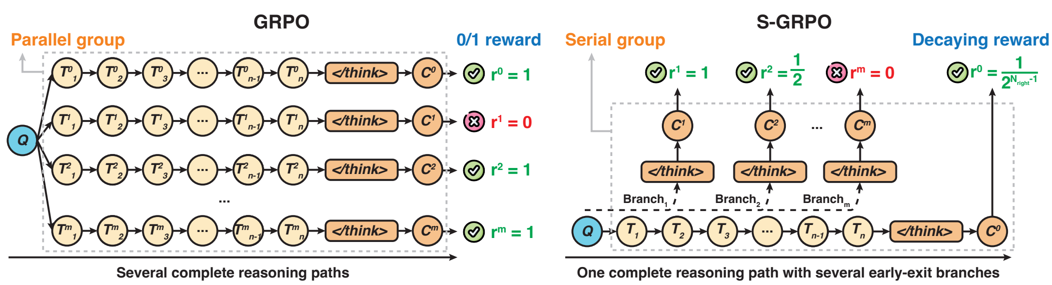

Figure 1: Comparison of parallel-group-relative GRPO and our serial-group-relative S-GRPO.

Abstract

As Test-Time Scaling emerges as an active research focus in the large language model community, advanced post-training methods increasingly emphasize extending chain-of-thought (CoT) generation length, thereby enhancing reasoning capabilities to approach Deepseek R1-like reasoning models. However, recent studies reveal that reasoning models (even Qwen3) consistently exhibit excessive thought redundancy in CoT generation. This overthinking issue arises from the inherent limitations of conventional outcome-reward reinforcement learning, which systematically overlooks the regulation of intermediate reasoning processes. This paper introduces **S**erial-**G**roup Decaying-**R**eward **P**olicy **O**ptimization (S-GRPO), a novel reinforcement learning paradigm that enables models to implicitly evaluate the sufficiency of intermediate reasoning steps, thereby facilitating early exit in CoT generation. Unlike GRPO, which samples multiple possible reasoning paths in parallel (parallel group), S-GRPO only samples one reasoning path and serially selects multiple temporal positions from the path to exit thinking and directly generate answers (serial group). For correct answers within a serial group, rewards gradually decrease based on the exit positions along the reasoning path from front to back. This design encourages the model to produce more accurate and concise thoughts, while also incentivizing early thinking termination when appropriate. Empirical evaluations demonstrate that S-GRPO is compatible with state-of-the-art reasoning models, including Qwen3 and Deepseek-distill. Across diverse benchmarks such as GSM8K, AIME 2024, AMC 2023, MATH-500, and GPQA Diamond, S-GRPO achieves a substantial reduction in sequence length (40.4%∼61.1%) while simultaneously improving accuracy (absolute 0.72%∼3.92%).

* Equal contribution. The work was done when Muzhi and Chenxu were interns at Huawei.
† Corresponding Author.

39th Conference on Neural Information Processing Systems (NeurIPS 2025).

1 Introduction

Test-Time Scaling [1] demonstrates a strong correlation between extended chain-of-thought (CoT) and enhanced reasoning capabilities in Large Language Models, which is more effective than scaling model parameters [2]. The success of DeepSeek-R1 [3] and GPT-o1 [4] has further sparked the research interest in reasoning models [5] within the LLM community. Furthermore, reinforcement learning [6, 7, 8, 9, 10] (RL) in post-training has demonstrated significant potential in stimulating long chain-of-thought generation and strengthening deep-thinking capabilities.

However, recent studies [11, 12] have identified a critical limitation in reasoning models: their tendency to engage in redundant thought processes, a phenomenon referred to as Overthinking [11, 13]. They frequently generate unnecessarily lengthy reasoning sequences [13, 14], including irrelevant information and superfluous reasoning steps. This redundant thought inflates computational overhead and even undermines reasoning accuracy by diverting the model from valid reasoning pathways to incorrect ones [15]. We attribute this issue to the inherent limitations of 0/1 outcome-reward RL (e.g., GRPO [6]), where reliance on final outcome rewards fails to identify when intermediate reasoning steps are sufficient.

As shown in Figure 1 (left), a standard outcome-reward RL, such as GRPO, tends to sample a query multiple times in parallel to obtain a parallel group. All reasoning chains and corresponding conclusions within the parallel group receive 0/1 outcome rewards, which reinforces correct outcomes but overlooks the presence of overthinking or inefficiencies in intermediate reasoning steps. While this approach successfully aggregates pass@k reasoning capability into pass@1, the neglect of regulating intermediate reasoning results in inefficient inference. Conversely, as illustrated in Figure 1 (right), we construct a serial group by sequentially generating multiple completions during a single CoT process. By prioritizing rewards for earlier correct completions, this approach encourages models to demonstrate complete reasoning capabilities in the initial phases, enabling early exit and preventing overthinking.

Motivated by this, we propose **S**erial-**G**roup Decaying-**R**eward **P**olicy **O**ptimization (**S-GRPO**), a simple yet effective modification to address standard outcome-reward RL (GRPO)'s inability to regulate intermediate reasoning processes. During S-GRPO training, we construct a serial group for each query using a two-phase rollout process. In the first phase, a complete reasoning path is generated. In the second phase, subsequent rollouts introduce early-exit interventions at different positions along the reasoning path generated in the first phase, producing intermediate answers. Finally, the serial group is formed by combining the complete reasoning path from the first phase with the intermediate answers appended to its corresponding truncated reasoning path. On this basis, we assign the rewards that decay according to their order of early exit for the correct ones within the serial group. S-GRPO culminates in computing serial-group relative advantages and using their policy gradient to update model parameters.

Our contributions are threefold:

- We pioneer a serial-group RL paradigm that overcomes the critical limitation of outcome-reward RL in regulating intermediate reasoning processes, accompanied by an open-sourced training framework (released once accepted).
- The proposed S-GRPO algorithm enables models to produce higher-quality reasoning paths during the early stages of CoT generation, and implicitly early-exit once sufficiency is achieved. S-GRPO preserves the integrity of the original reasoning process via a two-stage rollout procedure, ensuring that the model's pre-existing reasoning abilities are not compromised, which is well-suited as the final stage of post-training.
- Extensive experiments across GSM8K [16], AIME 2024 [17], AMC 2023 [18], MATH-500 [19], and GPQA [20] benchmarks with Qwen3 [21] and Deepseek-series reasoning models, demonstrate 0.72%$\sim$3.92% absolute accuracy improvement alongside 40.4%$\sim$61.1% average token reduction, establishing an efficiency-accuracy synergistic improvement.

2 Method

The proposed **Serial-Group Decaying-Reward Policy Optimization (S-GRPO)** is a novel reinforcement learning mechanism that innovatively leverages rule-based outcome rewards to regulate

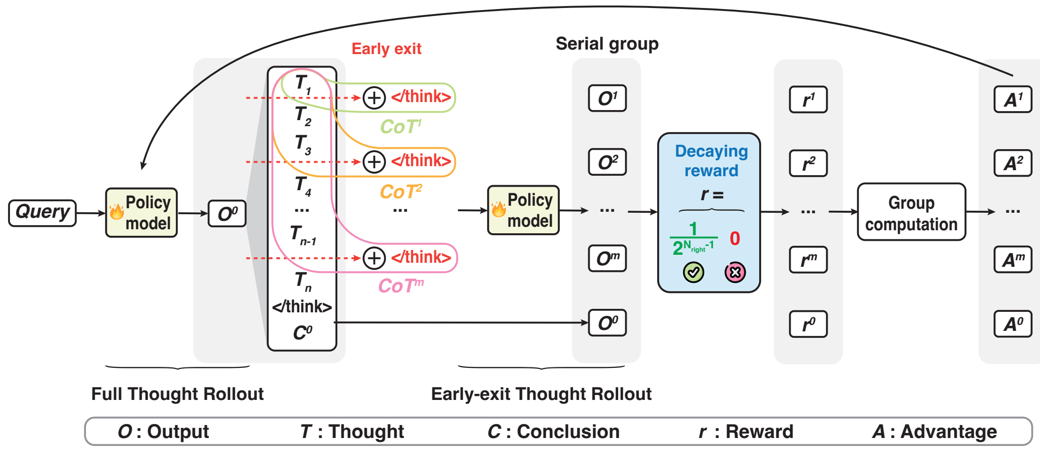

Figure 2: The framework of S-GRPO. The complete answer inducer is omitted in the figure and is represented by </think> instead. The complete answer inducer is `"Time is limited, stop thinking and start answering.\n</think>\n\n"`

intermediate reasoning processes and improve reasoning efficiency. During S-GRPO training, the LLM exits early and generates answers at different positions within a complete CoT, forming a serial group. Rewards for correct answers are assigned based on position, with earlier thinking exits receiving relatively higher rewards. This strategy encourages the model to produce high-quality reasoning earlier and terminate once sufficient reasoning is achieved. In particular, the framework is divided into three stages: Serial-Group Generation, Decaying Reward Strategy, and Advantage Computation and Parameter Update.

2.1 Serial-Group Generation

GRPO is originally designed to enable reasoning models to achieve their pass@k potential under pass@1 settings. To achieve this, GRPO generates multiple CoTs in parallel for each query, forming a parallel group, and rewards only those with correct answers. In contrast, S-GRPO aims to achieve efficient reasoning through sufficient-reasoning early exit. Specifically, we perform early-exit interventions at different positions within a single CoT to construct a serial group, thereby allowing the RL training to compare the thought sufficiency in different positions along the reasoning path. It consists of two stages: Full Thought Rollout and Early-exit Thought Rollout.

2.1.1 Full Thought Rollout

In the Full Thought Rollout stage, the model generates a complete reasoning path sequentially for each query, represented as $\{O^0 = T_1, T_2, \ldots, T_n, < /think >, C_0\}$. To expose the model to diverse reasoning sufficiency scenarios and enhance its ability to handle early exits effectively, we adopt **random-length truncation** during training. Specifically, the reasoning sequence is truncated at m randomly selected temporal positions P_i, where $P_i = T_i \quad and \quad i \sim \text{Uniform}(1, n)$, ensuring that the truncation points are uniformly distributed across the reasoning path. This randomness allows the model to implicitly evaluate reasoning sufficiency across varying depths and prevents overfitting to specific reasoning path lengths.

2.1.2 Early-exit Thought Rollout

In the Early-exit Thought Rollout stage, the policy model extracts several early-exit reasoning paths $\{CoT^1, CoT^2, ..., CoT^m\} = \{O^0[: P_1], O^0[: P_2]..., , O^0[: P_m]\}$, representing truncated segments of the full reasoning path at different positions. For each early-exit path, the model generates corresponding answers $C_1, C_2, \ldots, C_m$.

In particular, we insert the prompt `"Time is limited, stop thinking and start answering. \n</think>\n\n"` at each truncated position P_i. This prompt explicitly instructs the model to halt further reasoning and produce answers C^i. Details of this process are illustrated in Figure 5.

To ensure that the serial group contains early exit samples with correct answers (obtained by Early-exit Thought Rollout), we sample more queries than the training batch required (over-sampling) for data filtering, such as DAPO [22].

2.2 Decaying Reward Strategy

To encourage models to produce adequate and correct reasoning steps at earlier stages of CoT generation for accurate problem-solving, we propose a Decaying Reward Strategy. This mechanism assigns rewards based on the correctness of answers generated during the two-time rollouts ($C_1, C_2, \ldots, C_m, and\ C_0$), while decaying rewards according to their order of early exits. For the answer C^i of each response O^i, the reward r^i is defined as follows:

$$r^i = \begin{cases} \frac{1}{2^{N_{\text{right}}-1}}, & \text{if } C^i \text{ is correct,} \\ 0, & \text{if } C^i \text{ is incorrect.} \end{cases} \tag{1}$$

Where N_{right} refers to the accumulated number of correct answers up to and including the current position. The Decaying Reward Strategy is designed with dual objectives: (1) Exponential decay for correct answers: The strategy applies exponentially diminishing rewards to enhance the quality of earlier reasoning steps, which is overlooked by binary 0/1 outcome rewards. (2) Zero reward for incorrect answers: The strategy enforces a correctness-first optimization strategy, ensuring the model maintains robust core reasoning capabilities.

This dual-objective design strikes a balance between reasoning sufficiency and efficiency, guiding the model to produce reasoning sequences that are both accurate and concise.

2.3 Advantage Computation and Parameter Update

After computing the decaying rewards, S-GRPO calculates the advantage for each response in the serial group. Specifically, the advantage is calculated by subtracting the mean reward of the group from its corresponding reward, as defined by the formula: $\hat{A}_i = r_i - \text{mean}(r_i)$. Here, for training stability, the standard deviation is removed from the advantage computation compared to the GRPO. Subsequently, the computed advantage for each sample is broadcast to all corresponding response tokens. Finally, parameter updates are performed based on the advantage values of each sample (Algorithm 1). The optimization objective is as follows:

$$\mathcal{J}_{\text{S-GRPO}}(\theta) = \mathbb{E}_{[q\sim P(Q),\{o_i\}_{i=1}^{G}\sim\pi_{\theta_{\text{old}}}(O|q)]}$$
$$[\frac{1}{G}\sum_{i=1}^{G}\frac{1}{|o_i|}\sum_{t=1}^{|o_i|}\{\min[\frac{\pi_{\theta}^{i,t}}{\pi_{\theta_{\text{old}}}^{i,t}}\hat{A}_{i,t},\ \text{clip}(\frac{\pi_{\theta}^{i,t}}{\pi_{\theta_{\text{old}}}^{i,t}},\ 1-\epsilon,\ 1+\epsilon)\hat{A}_{i,t}]\}] \tag{2}$$

where $\pi^{i,t} = \pi(o_{i,t} \mid q, o_{i,<t})$ denotes the conditional probability of the token at position t. The models π_{θ_d}, $\pi_{\theta_{d\text{old}}}$ correspond to the training model and sampling model, respectively. q represents the input query, and $\{o_i\}_{i=1}^{G}$ are the full and early-exit thought rollout responses generated by the model. The advantage $\hat{A}_i$ for each response is computed as $\hat{A}_i = r_i - \text{mean}(r_i)$, where r_i is the decaying reward assigned to the response. For token-level advantage $\hat{A}_{i,t}$, it is defined to be equal to the corresponding sequence-level advantage $\hat{A}_i$. The hyperparameter ϵ is used to bound the importance sampling ratio.

3 Experiments

3.1 Experimental Setup

Training datasets. We selected problems from DeepMath-103K [23] to build our training set. This dataset is a large-scale and challenging collection of mathematics problems, focusing on difficulty levels ranging from grade 5 to grade 10. It addresses the lack of sufficient complexity present in existing datasets by offering approximately 103,000 carefully curated problems. The dataset is

Algorithm 1 Serial-Group Decaying-Reward Policy Optimization (S-GRPO)

Require: Query Q, Policy model π_θ, Number of positions to sample m
Ensure: Updated policy parameters θ
Full Thought Rollout:
Generate complete reasoning path $O^0 = (T_1, T_2, \ldots, T_n,$ </think>$, C^0)$ using π_θ for query Q
Sample m temporal positions $P_i \in \{P_1, P_2, \ldots, P_m\}$ where $P_i \sim$ Uniform$(1, n)$
Early-exit Thought Rollout:
$T_{\text{</think>}}$ = tokenizer.encode("Time is limited, stop thinking and start answering. `\n`</think>`\n\n`")
for each position P_i **do**
    Append $T_{\text{</think>}}$ after position P_i to form $CoT^i = (T_1, T_2, \ldots, T_{P_i}, T_{\text{</think>}})$
    Generate answer C^i using policy model π_θ conditioned on CoT^i
    Form output $O^i = (CoT^i, C^i)$
end for
Rule-based Dynamic Decaying-reward:
$N_{\text{right}} \leftarrow 0$ ▷ Counter for correct answers
for each output O^i where $i \in \{1, 2, \ldots, m, 0\}$ in order of position **do**
    if C^i is correct **then**
        $N_{\text{right}} \leftarrow N_{\text{right}} + 1$
        $r^i \leftarrow \frac{1}{2^{N_{\text{right}}-1}}$ ▷ Exponentially decaying reward
    else
        $r^i \leftarrow 0$ ▷ Zero reward for incorrect answers
    end if
end for
Group Computation:
Compute advantages A^i for each output based on rewards r^i
Update policy parameters θ using policy gradient with advantages A^i

constructed through an extensive data cleaning process applied to known benchmarks, ensuring no overlap with widely used benchmarks.

Benchmarks. To comprehensively assess the models' capabilities across a range of reasoning tasks, we select five popular benchmarks that reflect diverse levels of difficulty: **GSM8K** [16] is a well-curated collection of 1,319 problems in elementary mathematics. This benchmark is specifically designed to evaluate multi-step reasoning in foundational math tasks. Problems typically involve two to eight sequential operations, relying primarily on basic arithmetic performed over multiple intermediate steps. **AIME 2024** [17] includes 30 advanced problems selected from the 2024 edition of the American Invitational Mathematics Examination. This highly regarded competition tests participants' ability to reason mathematically across a broad range of topics, including arithmetic, algebra, combinatorics, geometry, number theory, and probability—core components of secondary-level mathematics. **AMC 2023** [18] consists of 40 questions spanning key areas such as algebra, geometry, number theory, and combinatorics. As part of the American Mathematics Competitions organized by the Mathematical Association of America (MAA), AMC aims to foster problem-solving skills and identify promising young mathematicians. **MATH-500** [19] presents a set of challenging problems drawn from high school-level mathematical competitions. To ensure comparability with prior studies, we use the same subset of 500 problems originally compiled by OpenAI for evaluation purposes [24]. In addition to these mathematical evaluations, we also examine performance on scientific reasoning tasks, using the following benchmark: **GPQA** [20] is a rigorously constructed dataset containing graduate-level questions in physics, chemistry, and biology. Notably, even domain experts with PhDs achieve only 69.7% accuracy on this benchmark [25]. For our experiments, we focus on the highest-quality subset, **GPQA Diamond**, which contains 198 carefully chosen questions.

Baselines. We compare S-GRPO with various existing efficient reasoning methods, including the training-free, output-based approach DEER [15], the off-policy optimization method ConCISE [26], and on-policy RL-based approaches such as original GRPO, RL + Length Penalty [27], and ShorterBetter [28]. DEER makes early-exit decisions during the inference phase, guided by the confidence scores of intermediate answers. ConCISE generates more concise chains of thought by

Table 1: Experimental results on four large reasoning models.

Method	GSM8K Acc	GSM8K Tokens	AIME 2024 Acc	AIME 2024 Tokens	AMC 2023 Acc	AMC 2023 Tokens	MATH-500 Acc	MATH-500 Tokens	GPQA Acc	GPQA Tokens	Overall Acc	Overall Tokens
DeepSeek-R1-Distill-Qwen-7B												
Vanilla	92.4	1,833	55.4	13,232	77.2	9,693	85.8	5,590	50.1	15,385	72.18	9,147
DEER	88.8	917	53.3	10,971	87.5	4,142	91.8	2,431	47.5	5,280	73.78 +1.60	4,748 -48.1%
ConCISE$_{SFT}$	92.9	832	52.1	9,751	–	–	92.0	2,244	50.0	5,892	–	–
ConCISE$_{SimPO}$	92.1	715	48.3	7,745	–	–	91.0	1,946	48.0	4,859	–	–
GRPO	93.2	1,767	55.0	13,451	87.5	9,887	93.6	5,317	50.7	15,817	76.00 +3.82	9,248 +1.1%
RL + Length Penalty	92.4	1,062	51.9	7,464	86.9	3,540	92.2	2,451	49.1	3,984	74.50 +2.32	3,700 -59.5%
ShorterBetter	–	–	53.3	5,288	75.9	2,580	–	–	–	–	–	–
S-GRPO	93.8	906	56.0	7,377	87.5	3,494	92.4	2,252	50.8	3,751	**76.10 +3.92**	**3,556 -61.1%**
DeepSeek-R1-Distill-Qwen-14B												
Vanilla	94.2	2,129	64.4	11,099	90.5	5,527	93.5	3,844	59.2	6,034	80.36	5,727
DEER	93.3	982	70.0	10,335	90.0	4,349	91.4	2,753	57.1	4,767	80.36 +0.0	4,637 -19.0%
GRPO	95.3	2,120	65.8	13,504	91.9	6,595	94.0	4,471	58.9	7,354	81.18 +0.82	6,809 +18.9%
RL + Length Penalty	94.7	775	55.0	7,950	88.1	3,396	92.4	1,993	56.0	4,380	77.24 -3.12	3,699 -35.4%
S-GRPO	96.2	724	64.4	6,712	91.9	3,352	93.6	2,146	59.3	3,334	**81.08 +0.72**	**3,254 -43.2%**
Qwen3-8B												
Vanilla	95.4	2,370	74.1	15,326	91.3	9,452	93.4	5,577	55.6	8,741	81.96	8,293
DEER	95.5	981	76.7	11,287	95.0	6,198	93.4	3,208	52.5	3,104	82.62 +0.66	4,956 -40.2%
GRPO	95.8	2,355	72.7	15,154	92.8	8,983	94.4	5,440	55.8	8,819	82.30 +0.34	8,150 -1.7%
RL + Length Penalty	95.4	1,323	73.8	9,666	93.4	5,042	94.2	3,247	56.2	5,293	82.60 +0.64	4,914 -40.7%
S-GRPO	96.1	1,292	77.3	8,810	95.0	5,962	95.2	3,166	57.7	5,271	**84.26 +2.3**	**4,900 -40.9%**
Qwen3-14B												
Vanilla	95.5	1,909	75.4	14,116	96.9	7,576	95.2	5,078	58.8	7,576	84.36	7,251
DEER	95.5	908	76.7	10,333	95.0	5,099	94.8	2,987	57.1	2,435	83.82 -0.54	4,352 -40.0%
GRPO	96.1	1,956	77.7	14,544	98.4	8,000	95.8	5,140	59.3	7,966	85.46 +1.1	7,521 +3.7%
RL + Length Penalty	95.8	1,090	74.8	9,056	96.6	5,059	95.8	2,866	59.4	4,949	84.48 +0.12	4,604 -36.5%
S-GRPO	96.3	952	77.9	8,932	97.8	4,537	96.4	2,652	60.6	4,537	**85.80 +1.44**	**4,322 -40.4%**

integrating specialized prompt tokens and applying early-exit mechanisms during inference, followed by SFT/SimPO to further encourage succinct reasoning. RL + Length Penalty assigns reward values based on the deviation of each correct response length from the mean, penalizing longer correct responses. ShorterBetter promotes shorter yet accurate reasoning paths by assigning higher rewards to compact chains that yield correct answers based on GRPO.

Models. We evaluate S-GRPO on four large reasoning models, including DeepSeek-R1-Distill-Qwen-7B, DeepSeek-R1-Distill-Qwen-14B [3], Qwen3-8B, and Qwen3-14B [21]. Despite achieving state-of-the-art performance on reasoning tasks, these models tend to generate overly verbose reasoning processes that contain excessive redundant information.

Metrics. S-GRPO is designed to improve correctness while minimizing inference length, thereby enabling more efficient reasoning. To evaluate its performance, we adopted two key metrics: *Accuracy* (i.e., pass@1) and *Token Count* (Tokens). Due to the inherent instability of generating long sequences in reasoning models and the limited sample sizes of certain benchmarks, we conducted multiple evaluation runs and reported the averaged results in the tables. Specifically, we performed 16 trials on AIME 2024 and AMC 2023, 8 trials on MATH-500 and GPQA Diamond, and 4 trials on GSM8K.

Training details. For S-GRPO, we use a learning rate of 1×10^{-6} and randomly select 8 temporal positions for each query. Since we adopt an on-policy mode, the generation batch size and training batch size are both set to 128×8. For GRPO, we use the same learning rate and batch size settings. For RL + Length Penalty, we follow the settings described in its original paper [27] and set the scalar parameter α to 0.2. Across all experiments, we employ Adam [29] as the standard optimizer.

3.2 Experimental Results

Main results. The experimental results in Table 1 demonstrate that S-GRPO consistently outperforms existing baselines across five benchmark datasets and four reasoning models, achieving significant improvements. Compared to vanilla reasoning models, S-GRPO achieves an average accuracy (absolute) improvement of 0.72% to 3.92%, while reducing the generated sequence length by 40.4% to 61.1%. S-GRPO achieves notable improvements on in-domain mathematical reasoning benchmarks (e.g., GSM8K, AIME 2024, AMC 2023, MATH-500) and out-of-domain scientific reasoning tasks (e.g., GPQA), demonstrating its effectiveness and robustness. Specifically, when applied to DeepSeek-R1-Distill-Qwen-7B, S-GRPO achieves a 10.3-point increase in accuracy on

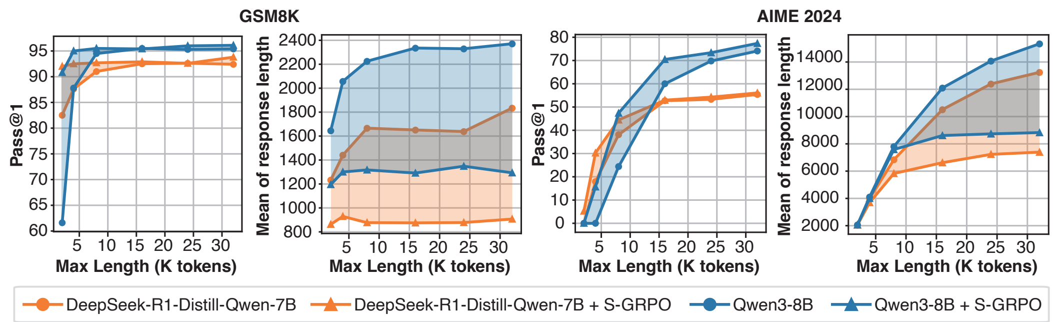

Figure 3: Performance of DeepSeek-R1-Distill-Qwen-7B and Qwen3-8B without or with S-GRPO training on GSM8K and AIME 2024 under different generation-length budgets.

AMC 2023 while utilizing only 36% of the original reasoning budget. Similarly, it obtains a 0.6-point improvement on AIME 2024 with just 56% of the original inference length.

Comparison with SOTAs. We compare S-GRPO with several approaches designed for efficient reasoning. For example, DEER reduces reasoning length by only 7%∼27% on AIME 2024 and sometimes sacrifices accuracy, and its effectiveness is mainly limited to simple tasks. In contrast, S-GRPO consistently reduces reasoning length across both simple and complex tasks while simultaneously improving reasoning accuracy. Compared to the original GRPO, S-GRPO achieves comparable or even better accuracy while significantly shortening the inference trajectory, indicating that the proposed Serial-Group Generation mechanism does not hinder the model's exploration capability in reinforcement learning. Moreover, the reward shaping based on serialized intermediate outputs more effectively guides the model toward efficient reasoning.

Compared with the most recent training-based efficient reasoning methods, S-GRPO achieves the best in both reasoning length reduction and accuracy. This can be explained by the fact that ConCISE's off-policy optimization forces the model to fit into a new data distribution, and ShorterBetter and RL + Length Penalty overly emphasize length reduction, at the expense of generalization and accuracy-driven optimization. In contrast, S-GRPO preserves the integrity of the original reasoning process via a two-stage rollout procedure, ensuring that the model's pre-existing reasoning abilities are not compromised, which is well-suited as the final stage of post-training.

Performance with different token budget. We vary the generation-length budget during inference from short to long, and compare the change in accuracy and actual reasoning length of S-GRPO and vanilla CoT on GSM8K (representing simple problems) and AIME 2024 (representing complex problems). The results in Figure 3 show that, across all budgets tested, S-GRPO consistently achieves higher accuracy while generating shorter sequences compared to vanilla CoT, further highlighting the effectiveness of our method.

Moreover, we observe that under tight length budgets, S-GRPO achieves significantly higher accuracy than vanilla CoT while generating sequences with comparable length. Differently, under larger length budgets, S-GRPO produces significantly shorter reasoning paths with slightly better accuracy, compared with vanilla CoT. The trends in both accuracy and actual generation length of S-GRPO are smoother than vanilla CoT, indicating its greater robustness to variations in length budget. Overall, S-GRPO achieves high accuracy under a low length budget, suggesting that our method is capable of generating concise yet accurate reasoning paths.

Ablation results. To verify the effectiveness of each design component in S-GRPO, we conduct ablation studies under three different settings. *w/o. Decaying (Shortest 1)* denotes the setting where only the shortest correct response in the serial group is assigned a reward of 1, while all other responses receive a reward of 0. *w/o. Decaying (All 1)* refers to the configuration where all correct responses in the serial group are assigned a reward of 1, while incorrect ones receive a reward of 0. *w/o. Serial* indicates that we further remove the Serial-Group Generation mechanism based on the w/o. Decaying (All 1) configuration

Table 2: Ablation results on Qwen3-8B.

Method	GSM8K		AIME 2024		AMC 2023		MATH-500		GPQA		Overall	
	Acc	Tokens	Acc	Tokens	Acc	Tokens	Acc	Tokens	Acc	Tokens	Acc	Tokens
Qwen3-8B												
S-GRPO	96.1	1,292	77.3	8,810	95.0	5,962	95.2	3,166	57.7	5,271	**84.26**	4,900
w/o. Decaying (Shortest 1)	95.9	1,175	69.6	8,721	92.5	4,581	94.8	2,740	55.7	4,734	81.70-2.56	**4,390-10.4%**
w/o. Decaying (All 1)	96.0	2,385	74.4	14,940	94.7	9,000	95.0	5,614	54.9	8,955	83.00-1.26	8,179+66.9%
- w/o. Serial	95.8	2,355	72.7	15,154	92.8	8,983	94.4	5,440	55.8	8,819	82.30-1.96	8,150+66.3%

The results in Table 2 indicate that rewarding only the shortest correct response imposes an overly strict constraint. Although this setting leads to further reductions in reasoning length, it comes at the cost of accuracy. And removing the design that assigns higher rewards to shorter outputs, i.e., *w/o. Decaying (All 1)*, results in lengthy reasoning. This is because long correct answers also receive high rewards, and the model does not shift in favor of generating short CoTs. When the Serial-Group Generation mechanism is removed, our method degenerates to GRPO, achieving performance comparable to *w/o. Decaying (All 1)* in both accuracy and reasoning length. This demonstrates that Serial-Group Generation, as an essential component of S-GRPO, does not compromise the model's exploration capability in reinforcement learning.

Case study. Figure 5 in the Appendix visually illustrates how S-GRPO generates answers at different early-exit positions and also shows the decaying reward assignments. For the first early-exit position, where the model produces an incorrect intermediate answer, we set the reward to 0. For subsequent exits that yield correct answers, we apply a decaying positive reward scheme, where earlier exits are associated with higher reward values. This design incentivizes the model to discover reasoning paths that are both accurate and succinct.

To more intuitively illustrate the effectiveness of our approach, a representative example is shown in Figure 4. The left and right sides of the figure compare the reasoning processes of vanilla CoT and S-GRPO. Although both methods yield the correct final answer, S-GRPO achieves this using less than half of the reasoning budget, showcasing its effectiveness in mitigating the overthinking problem [11]. The central portion of the figure displays an early exit obtained by directly truncating the vanilla CoT's reasoning process using the same token budget as S-GRPO. The reasoning model fails to reach the correct conclusion based on the available partial reasoning. This demonstrates that S-GRPO effectively identifies the correct solution path and guides the model to concise and accurate reasoning, while inherently avoiding the underthinking issue of superficial exploration [30].

4 Related Work

The success of OpenAI o1 [4, 31] highlights the powerful potential of reinforcement learning in post-training to enhance model reasoning capabilities. With the open source of Deepseek-R1 [3] and Qwen3 [21], reasoning models are now widely deployed locally, drawing the LLM community's attention to the efficiency of long chain-of-thought generation.

Reinforcement learning for reasoning. Rule-based outcome reward RL [3, 32, 33, 34, 35, 36, 37] has emerged as the mainstream approach for post-training optimization of reasoning models. This paradigm simplifies reward design by employing binary 0/1 rewards determined through rule-based correctness evaluation, eliminating the need for separate reward models as required in original GRPO [6, 3, 38] implementations, thereby substantially reducing memory and computational overhead during RL training. Recent advances have introduced numerous RL algorithm variants focusing on training efficiency [39, 40, 41], value function optimization [42, 43, 44], and other aspects. However, these developments preserve GRPO's fundamental mechanism that samples several CoTs in parallel for a group, consequently overlooking optimization opportunities during sequential thought generation within a single CoT. Our proposed optimized version of GRPO, S-GRPO, maintains the rule-based outcome reward framework while enabling intermediate reasoning process rewards by sampling multiple early-exit thought chains in serial for one CoT. We allow existing reasoning models to effectively mitigate overthinking and boost inference efficiency.

Efficient reasoning. Approaches for efficient reasoning can be categorized into two groups: training-free and training-based methods. Training-free methods typically improve reasoning effi-

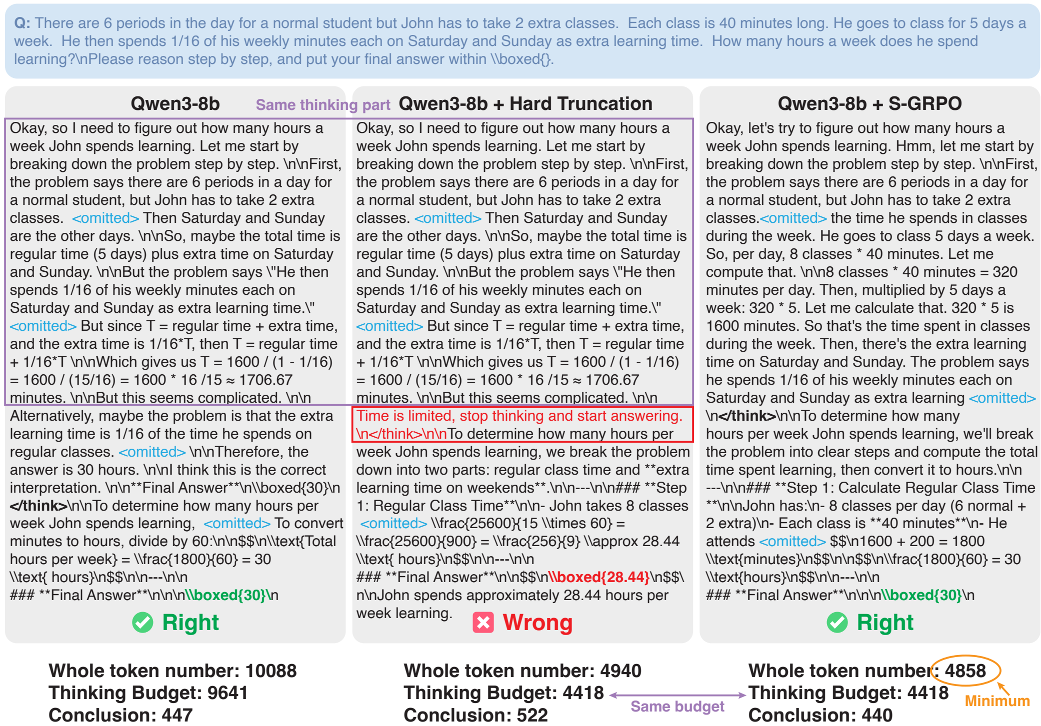

Figure 4: Comparison of a generated content sample on GSM8K.

ciency through dynamic prompting strategies [45, 46, 47, 48, 49, 50, 51, 52] , Best-of-N sampling pruning [53, 54] and optimizations [55, 56, 57] , and early-exit [58, 15] mechanisms during reasoning. DEER [15] and our method employ a similar operation by implementing early exit during CoT generation and producing intermediate answers. The key distinction lies in that DEER directly makes early-exit decisions during inference based on the confidence of intermediate answers, whereas our method leverages RL to reinforce models' behavior of correct early exit, which requires no additional compute overhead during inference.

Training-based methods primarily supervised fine-tuning models with variable-length CoT data [59, 60, 61, 62, 63, 64, 45], or training with length-rewards RL [12, 65, 66, 67, 68, 69, 70, 71, 72]. Recently, ConCISE [26] constructs concise CoT data by inserting prompt tokens and employing early-exit during inference, then enhances the model's reasoning conciseness through SFT/SimPO [73, 74]. [27] and [28] upweight rewards for the shorter CoTs, having correct answers, in parallel-sampled reasoning chains. However, they inherit GRPO's parallel sampling paradigm that neglects the reward function's attention to the intermediate serial reasoning processes.

5 Conclusion

This paper introduces Serial-Group Decaying-Reward Policy Optimization (S-GRPO), which innovatively leverages rule-based outcome rewards to regulate intermediate reasoning processes. By incentivizing the LLM to produce high-quality thoughts earlier and exit promptly when the generated thought is sufficient, S-GRPO enhances reasoning efficiency and maintains accuracy. Empirical evaluations demonstrate that S-GRPO achieves optimal synergy between efficiency and accuracy, significantly outperforming existing efficient reasoning methods. Specifically, S-GRPO demonstrates compatibility with state-of-the-art reasoning models, including Qwen3 and Deepseek-distill, achieving a 40.4%∼61.1% reduction in sequence length while improving accuracy (absolute) by 0.72%∼3.92% across five benchmark datasets: GSM8K, AIME 2024, AMC 2023, MATH-500, and GPQA Diamond. S-GRPO is well-suited for deployment as the final optimization stage in post-training pipelines, offering a practical solution for enhancing reasoning efficiency.

References

[1] Charlie Snell, Jaehoon Lee, Kelvin Xu, and Aviral Kumar. Scaling llm test-time compute optimally can be more effective than scaling model parameters, 2024.

[2] Jared Kaplan, Sam McCandlish, Tom Henighan, Tom B. Brown, Benjamin Chess, Rewon Child, Scott Gray, Alec Radford, Jeffrey Wu, and Dario Amodei. Scaling laws for neural language models, 2020.

[3] DeepSeek-AI, Daya Guo, Dejian Yang, Haowei Zhang, Junxiao Song, Ruoyu Zhang, Runxin Xu, Qihao Zhu, Shirong Ma, Peiyi Wang, Xiao Bi, Xiaokang Zhang, Xingkai Yu, Yu Wu, Z. F. Wu, Zhibin Gou, Zhihong Shao, Zhuoshu Li, Ziyi Gao, Aixin Liu, Bing Xue, Bingxuan Wang, Bochao Wu, Bei Feng, Chengda Lu, Chenggang Zhao, Chengqi Deng, Chenyu Zhang, Chong Ruan, Damai Dai, Deli Chen, Dongjie Ji, Erhang Li, Fangyun Lin, Fucong Dai, Fuli Luo, Guangbo Hao, Guanting Chen, Guowei Li, H. Zhang, Han Bao, Hanwei Xu, Haocheng Wang, Honghui Ding, Huajian Xin, Huazuo Gao, Hui Qu, Hui Li, Jianzhong Guo, Jiashi Li, Jiawei Wang, Jingchang Chen, Jingyang Yuan, Junjie Qiu, Junlong Li, J. L. Cai, Jiaqi Ni, Jian Liang, Jin Chen, Kai Dong, Kai Hu, Kaige Gao, Kang Guan, Kexin Huang, Kuai Yu, Lean Wang, Lecong Zhang, Liang Zhao, Litong Wang, Liyue Zhang, Lei Xu, Leyi Xia, Mingchuan Zhang, Minghua Zhang, Minghui Tang, Meng Li, Miaojun Wang, Mingming Li, Ning Tian, Panpan Huang, Peng Zhang, Qiancheng Wang, Qinyu Chen, Qiushi Du, Ruiqi Ge, Ruisong Zhang, Ruizhe Pan, Runji Wang, R. J. Chen, R. L. Jin, Ruyi Chen, Shanghao Lu, Shangyan Zhou, Shanhuang Chen, Shengfeng Ye, Shiyu Wang, Shuiping Yu, Shunfeng Zhou, Shuting Pan, S. S. Li, Shuang Zhou, Shaoqing Wu, Shengfeng Ye, Tao Yun, Tian Pei, Tianyu Sun, T. Wang, Wangding Zeng, Wanjia Zhao, Wen Liu, Wenfeng Liang, Wenjun Gao, Wenqin Yu, Wentao Zhang, W. L. Xiao, Wei An, Xiaodong Liu, Xiaohan Wang, Xiaokang Chen, Xiaotao Nie, Xin Cheng, Xin Liu, Xin Xie, Xingchao Liu, Xinyu Yang, Xinyuan Li, Xuecheng Su, Xuheng Lin, X. Q. Li, Xiangyue Jin, Xiaojin Shen, Xiaosha Chen, Xiaowen Sun, Xiaoxiang Wang, Xinnan Song, Xinyi Zhou, Xianzu Wang, Xinxia Shan, Y. K. Li, Y. Q. Wang, Y. X. Wei, Yang Zhang, Yanhong Xu, Yao Li, Yao Zhao, Yaofeng Sun, Yaohui Wang, Yi Yu, Yichao Zhang, Yifan Shi, Yiliang Xiong, Ying He, Yishi Piao, Yisong Wang, Yixuan Tan, Yiyang Ma, Yiyuan Liu, Yongqiang Guo, Yuan Ou, Yuduan Wang, Yue Gong, Yuheng Zou, Yujia He, Yunfan Xiong, Yuxiang Luo, Yuxiang You, Yuxuan Liu, Yuyang Zhou, Y. X. Zhu, Yanhong Xu, Yanping Huang, Yaohui Li, Yi Zheng, Yuchen Zhu, Yunxian Ma, Ying Tang, Yukun Zha, Yuting Yan, Z. Z. Ren, Zehui Ren, Zhangli Sha, Zhe Fu, Zhean Xu, Zhenda Xie, Zhengyan Zhang, Zhewen Hao, Zhicheng Ma, Zhigang Yan, Zhiyu Wu, Zihui Gu, Zijia Zhu, Zijun Liu, Zilin Li, Ziwei Xie, Ziyang Song, Zizheng Pan, Zhen Huang, Zhipeng Xu, Zhongyu Zhang, and Zhen Zhang. Deepseek-r1: Incentivizing reasoning capability in llms via reinforcement learning, 2025.

[4] OpenAI. Learning to reason with llms. `https://openai.com/research/learning-to-reason-with-llms`, 2025. Accessed: 15 March 2025.

[5] Fengli Xu, Qianyue Hao, Zefang Zong, Jingwei Wang, Yunke Zhang, Jingyi Wang, Xiaochong Lan, Jiahui Gong, Tianjian Ouyang, Fanjin Meng, Chenyang Shao, Yuwei Yan, Qinglong Yang, Yiwen Song, Sijian Ren, Xinyuan Hu, Yu Li, Jie Feng, Chen Gao, and Yong Li. Towards large reasoning models: A survey of reinforced reasoning with large language models, 2025.

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

[7] Yuntao Bai, Andy Jones, Kamal Ndousse, Amanda Askell, Anna Chen, Nova DasSarma, Dawn Drain, Stanislav Fort, Deep Ganguli, Tom Henighan, et al. Training a helpful and harmless assistant with reinforcement learning from human feedback. *arXiv preprint arXiv:2204.05862*, 2022.

[8] Long Ouyang, Jeffrey Wu, Xu Jiang, Diogo Almeida, Carroll Wainwright, Pamela Mishkin, Chong Zhang, Sandhini Agarwal, Katarina Slama, Alex Ray, et al. Training language models to follow instructions with human feedback. *Advances in neural information processing systems*, 35:27730–27744, 2022.

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

[10] Shyam Sundhar Ramesh, Yifan Hu, Iason Chaimalas, Viraj Mehta, Pier Giuseppe Sessa, Haitham Bou Ammar, and Ilija Bogunovic. Group robust preference optimization in reward-free RLHF. In *The Thirty-eighth Annual Conference on Neural Information Processing Systems*, 2024.

[11] Xingyu Chen, Jiahao Xu, Tian Liang, Zhiwei He, Jianhui Pang, Dian Yu, Linfeng Song, Qiuzhi Liu, Mengfei Zhou, Zhuosheng Zhang, Rui Wang, Zhaopeng Tu, Haitao Mi, and Dong Yu. Do not think that much for 2+3=? on the overthinking of o1-like llms, 2025.

[12] Kimi Team, Angang Du, Bofei Gao, Bowei Xing, Changjiu Jiang, Cheng Chen, Cheng Li, Chenjun Xiao, Chenzhuang Du, Chonghua Liao, et al. Kimi k1. 5: Scaling reinforcement learning with llms. *arXiv preprint arXiv:2501.12599*, 2025.

[13] Alejandro Cuadron, Dacheng Li, Wenjie Ma, Xingyao Wang, Yichuan Wang, Siyuan Zhuang, Shu Liu, Luis Gaspar Schroeder, Tian Xia, Huanzhi Mao, et al. The danger of overthinking: Examining the reasoning-action dilemma in agentic tasks. *arXiv preprint arXiv:2502.08235*, 2025.

[14] Yuyang Wu, Yifei Wang, Tianqi Du, Stefanie Jegelka, and Yisen Wang. When more is less: Understanding chain-of-thought length in llms. *arXiv preprint arXiv:2502.07266*, 2025.

[15] Chenxu Yang, Qingyi Si, Yongjie Duan, Zheliang Zhu, Chenyu Zhu, Zheng Lin, Li Cao, and Weiping Wang. Dynamic early exit in reasoning models, 2025.

[16] Karl Cobbe, Vineet Kosaraju, Mohammad Bavarian, Mark Chen, Heewoo Jun, Lukasz Kaiser, Matthias Plappert, Jerry Tworek, Jacob Hilton, Reiichiro Nakano, Christopher Hesse, and John Schulman. Training verifiers to solve math word problems, 2021.

[17] MAA Committees. Aime problems and solutions. `https://artofproblemsolving.com/wiki/index.php/AIME_Problems_and_Solutions`.

[18] AI-MO. Amc 2023, 2024.

[19] Dan Hendrycks, Collin Burns, Saurav Kadavath, Akul Arora, Steven Basart, Eric Tang, Dawn Song, and Jacob Steinhardt. Measuring mathematical problem solving with the math dataset, 2021.

[20] David Rein, Betty Li Hou, Asa Cooper Stickland, Jackson Petty, Richard Yuanzhe Pang, Julien Dirani, Julian Michael, and Samuel R. Bowman. Gpqa: A graduate-level google-proof q&a benchmark, 2023.

[21] An Yang, Anfeng Li, Baosong Yang, Beichen Zhang, Binyuan Hui, Bo Zheng, Bowen Yu, Chang Gao, Chengen Huang, Chenxu Lv, et al. Qwen3 technical report. *arXiv preprint arXiv:2505.09388*, 2025.

[22] Qiying Yu, Zheng Zhang, Ruofei Zhu, Yufeng Yuan, Xiaochen Zuo, Yu Yue, Tiantian Fan, Gaohong Liu, Lingjun Liu, Xin Liu, et al. Dapo: An open-source llm reinforcement learning system at scale. *arXiv preprint arXiv:2503.14476*, 2025.

[23] Zhiwei He, Tian Liang, Jiahao Xu, Qiuzhi Liu, Xingyu Chen, Yue Wang, Linfeng Song, Dian Yu, Zhenwen Liang, Wenxuan Wang, Zhuosheng Zhang, Rui Wang, Zhaopeng Tu, Haitao Mi, and Dong Yu. Deepmath-103k: A large-scale, challenging, decontaminated, and verifiable mathematical dataset for advancing reasoning, 2025.

[24] Hunter Lightman, Vineet Kosaraju, Yura Burda, Harri Edwards, Bowen Baker, Teddy Lee, Jan Leike, John Schulman, Ilya Sutskever, and Karl Cobbe. Let's verify step by step, 2023.

[25] OpenAI. Learning to reason with llms, September 2024.

[26] Ziqing Qiao, Yongheng Deng, Jiali Zeng, Dong Wang, Lai Wei, Fandong Meng, Jie Zhou, Ju Ren, and Yaoxue Zhang. Concise: Confidence-guided compression in step-by-step efficient reasoning, 2025.

[27] Daman Arora and Andrea Zanette. Training language models to reason efficiently, 2025.

[28] Jingyang Yi and Jiazheng Wang. Shorterbetter: Guiding reasoning models to find optimal inference length for efficient reasoning, 2025.

[29] Diederik P Kingma and Jimmy Ba. Adam: A method for stochastic optimization. *arXiv preprint arXiv:1412.6980*, 2014.

[30] Yue Wang, Qiuzhi Liu, Jiahao Xu, Tian Liang, Xingyu Chen, Zhiwei He, Linfeng Song, Dian Yu, Juntao Li, Zhuosheng Zhang, Rui Wang, Zhaopeng Tu, Haitao Mi, and Dong Yu. Thoughts are all over the place: On the underthinking of o1-like llms, 2025.

[31] Long Ouyang, Jeff Wu, Xu Jiang, Diogo Almeida, Carroll L. Wainwright, Pamela Mishkin, Chong Zhang, Sandhini Agarwal, Katarina Slama, Alex Ray, John Schulman, Jacob Hilton, Fraser Kelton, Luke Miller, Maddie Simens, Amanda Askell, Peter Welinder, Paul Christiano, Jan Leike, and Ryan Lowe. Training language models to follow instructions with human feedback, 2022.

[32] Kimi Team, Angang Du, Bofei Gao, Bowei Xing, Changjiu Jiang, Cheng Chen, Cheng Li, Chenjun Xiao, Chenzhuang Du, Chonghua Liao, Chuning Tang, Congcong Wang, Dehao Zhang, Enming Yuan, Enzhe Lu, Fengxiang Tang, Flood Sung, Guangda Wei, Guokun Lai, Haiqing Guo, Han Zhu, Hao Ding, Hao Hu, Hao Yang, Hao Zhang, Haotian Yao, Haotian Zhao, Haoyu Lu, Haoze Li, Haozhen Yu, Hongcheng Gao, Huabin Zheng, Huan Yuan, Jia Chen, Jianhang Guo, Jianlin Su, Jianzhou Wang, Jie Zhao, Jin Zhang, Jingyuan Liu, Junjie Yan, Junyan Wu, Lidong Shi, Ling Ye, Longhui Yu, Mengnan Dong, Neo Zhang, Ningchen Ma, Qiwei Pan, Qucheng Gong, Shaowei Liu, Shengling Ma, Shupeng Wei, Sihan Cao, Siying Huang, Tao Jiang, Weihao Gao, Weimin Xiong, Weiran He, Weixiao Huang, Wenhao Wu, Wenyang He, Xianghui Wei, Xianqing Jia, Xingzhe Wu, Xinran Xu, Xinxing Zu, Xinyu Zhou, Xuehai Pan, Y. Charles, Yang Li, Yangyang Hu, Yangyang Liu, Yanru Chen, Yejie Wang, Yibo Liu, Yidao Qin, Yifeng Liu, Ying Yang, Yiping Bao, Yulun Du, Yuxin Wu, Yuzhi Wang, Zaida Zhou, Zhaoji Wang, Zhaowei Li, Zhen Zhu, Zheng Zhang, Zhexu Wang, Zhilin Yang, Zhiqi Huang, Zihao Huang, Ziyao Xu, and Zonghan Yang. Kimi k1.5: Scaling reinforcement learning with llms, 2025.

[33] Jiaxuan Gao, Shusheng Xu, Wenjie Ye, Weilin Liu, Chuyi He, Wei Fu, Zhiyu Mei, Guangju Wang, and Yi Wu. On designing effective rl reward at training time for llm reasoning, 2024.

[34] Nathan Lambert, Jacob Morrison, Valentina Pyatkin, Shengyi Huang, Hamish Ivison, Faeze Brahman, Lester James V. Miranda, Alisa Liu, Nouha Dziri, Shane Lyu, Yuling Gu, Saumya Malik, Victoria Graf, Jena D. Hwang, Jiangjiang Yang, Ronan Le Bras, Oyvind Tafjord, Chris Wilhelm, Luca Soldaini, Noah A. Smith, Yizhong Wang, Pradeep Dasigi, and Hannaneh Hajishirzi. Tulu 3: Pushing frontiers in open language model post-training, 2025.

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

[36] Liang Wen, Yunke Cai, Fenrui Xiao, Xin He, Qi An, Zhenyu Duan, Yimin Du, Junchen Liu, Lifu Tang, Xiaowei Lv, Haosheng Zou, Yongchao Deng, Shousheng Jia, and Xiangzheng Zhang. Light-r1: Curriculum sft, dpo and rl for long cot from scratch and beyond, 2025.

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

[38] John Schulman, Filip Wolski, Prafulla Dhariwal, Alec Radford, and Oleg Klimov. Proximal policy optimization algorithms, 2017.

[39] 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, Jinhua Zhu, Jiaze Chen, Jiangjie Chen, Chengyi Wang, Hongli Yu, Weinan Dai, Yuxuan Song, Xiangpeng Wei, Hao Zhou, Jingjing Liu, Wei-Ying Ma, Ya-Qin Zhang, Lin Yan, Mu Qiao, Yonghui Wu, and Mingxuan Wang. Dapo: An open-source llm reinforcement learning system at scale, 2025.

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

[41] Xiaojiang Zhang, Jinghui Wang, Zifei Cheng, Wenhao Zhuang, Zheng Lin, Minglei Zhang, Shaojie Wang, Yinghan Cui, Chao Wang, Junyi Peng, Shimiao Jiang, Shiqi Kuang, Shouyu Yin, Chaohang Wen, Haotian Zhang, Bin Chen, and Bing Yu. Srpo: A cross-domain implementation of large-scale reinforcement learning on llm, 2025.

[42] Amirhossein Kazemnejad, Milad Aghajohari, Eva Portelance, Alessandro Sordoni, Siva Reddy, Aaron Courville, and Nicolas Le Roux. Vineppo: Unlocking rl potential for llm reasoning through refined credit assignment, 2024.

[43] Yufeng Yuan, Yu Yue, Ruofei Zhu, Tiantian Fan, and Lin Yan. What's behind ppo's collapse in long-cot? value optimization holds the secret, 2025.

[44] Yu Yue, Yufeng Yuan, Qiying Yu, Xiaochen Zuo, Ruofei Zhu, Wenyuan Xu, Jiaze Chen, Chengyi Wang, TianTian Fan, Zhengyin Du, Xiangpeng Wei, Xiangyu Yu, Gaohong Liu, Juncai Liu, Lingjun Liu, Haibin Lin, Zhiqi Lin, Bole Ma, Chi Zhang, Mofan Zhang, Wang Zhang, Hang Zhu, Ru Zhang, Xin Liu, Mingxuan Wang, Yonghui Wu, and Lin Yan. Vapo: Efficient and reliable reinforcement learning for advanced reasoning tasks, 2025.

[45] Tingxu Han, Zhenting Wang, Chunrong Fang, Shiyu Zhao, Shiqing Ma, and Zhenyu Chen. Token-budget-aware llm reasoning. *arXiv preprint arXiv:2412.18547*, 2024.

[46] Silei Xu, Wenhao Xie, Lingxiao Zhao, and Pengcheng He. Chain of draft: Thinking faster by writing less. *arXiv preprint arXiv:2502.18600*, 2025.

[47] Ayeong Lee, Ethan Che, and Tianyi Peng. How well do llms compress their own chain-of-thought? a token complexity approach. *arXiv preprint arXiv:2503.01141*, 2025.

[48] Matthew Renze and Erhan Guven. The benefits of a concise chain of thought on problem-solving in large language models. In *2024 2nd International Conference on Foundation and Large Language Models (FLLM)*, pages 476–483. IEEE, 2024.

[49] Qiguang Chen, Libo Qin, Jiaqi Wang, Jingxuan Zhou, and Wanxiang Che. Unlocking the capabilities of thought: A reasoning boundary framework to quantify and optimize chain-of-thought. *Advances in Neural Information Processing Systems*, 37:54872–54904, 2024.

[50] Simon A Aytes, Jinheon Baek, and Sung Ju Hwang. Sketch-of-thought: Efficient llm reasoning with adaptive cognitive-inspired sketching. *arXiv preprint arXiv:2503.05179*, 2025.

[51] Yu-Neng Chuang, Helen Zhou, Prathusha Sarma, Parikshit Gopalan, John Boccio, Sara Bolouki, and Xia Hu. Learning to route llms with confidence tokens. *arXiv preprint arXiv*, 2410, 2024.

[52] Isaac Ong, Amjad Almahairi, Vincent Wu, Wei-Lin Chiang, Tianhao Wu, Joseph E Gonzalez, M Waleed Kadous, and Ion Stoica. Routellm: Learning to route llms with preference data, 2024. *URL https://arxiv. org/abs/2406.18665*.

[53] Yuxi Xie, Kenji Kawaguchi, Yiran Zhao, James Xu Zhao, Min-Yen Kan, Junxian He, and Michael Xie. Self-evaluation guided beam search for reasoning. *Advances in Neural Information Processing Systems*, 36:41618–41650, 2023.

[54] Baohao Liao, Yuhui Xu, Hanze Dong, Junnan Li, Christof Monz, Silvio Savarese, Doyen Sahoo, and Caiming Xiong. Reward-guided speculative decoding for efficient llm reasoning. *arXiv preprint arXiv:2501.19324*, 2025.

[55] Yiwei Li, Peiwen Yuan, Shaoxiong Feng, Boyuan Pan, Xinglin Wang, Bin Sun, Heda Wang, and Kan Li. Escape sky-high cost: Early-stopping self-consistency for multi-step reasoning. *arXiv preprint arXiv:2401.10480*, 2024.

[56] Rohin Manvi, Anikait Singh, and Stefano Ermon. Adaptive inference-time compute: Llms can predict if they can do better, even mid-generation. *arXiv preprint arXiv:2410.02725*, 2024.

[57] Pranjal Aggarwal, Aman Madaan, Yiming Yang, et al. Let's sample step by step: Adaptive-consistency for efficient reasoning and coding with llms. *arXiv preprint arXiv:2305.11860*, 2023.

[58] Wenjie Ma, Jingxuan He, Charlie Snell, Tyler Griggs, Sewon Min, and Matei Zaharia. Reasoning models can be effective without thinking. *arXiv preprint arXiv:2504.09858*, 2025.

[59] Ping Yu, Jing Xu, Jason Weston, and Ilia Kulikov. Distilling system 2 into system 1. *arXiv preprint arXiv:2407.06023*, 2024.

[60] Yu Kang, Xianghui Sun, Liangyu Chen, and Wei Zou. C3ot: Generating shorter chain-of-thought without compromising effectiveness. In *Proceedings of the AAAI Conference on Artificial Intelligence*, volume 39, pages 24312–24320, 2025.

[61] Heming Xia, Yongqi Li, Chak Tou Leong, Wenjie Wang, and Wenjie Li. Tokenskip: Controllable chain-of-thought compression in llms. *arXiv preprint arXiv:2502.12067*, 2025.

[62] Xinyin Ma, Guangnian Wan, Runpeng Yu, Gongfan Fang, and Xinchao Wang. Cot-valve: Length-compressible chain-of-thought tuning. *arXiv preprint arXiv:2502.09601*, 2025.

[63] Tergel Munkhbat, Namgyu Ho, Seo Hyun Kim, Yongjin Yang, Yujin Kim, and Se-Young Yun. Self-training elicits concise reasoning in large language models. *arXiv preprint arXiv:2502.20122*, 2025.

[64] Tengxiao Liu, Qipeng Guo, Xiangkun Hu, Cheng Jiayang, Yue Zhang, Xipeng Qiu, and Zheng Zhang. Can language models learn to skip steps? *arXiv preprint arXiv:2411.01855*, 2024.

[65] Haotian Luo, Li Shen, Haiying He, Yibo Wang, Shiwei Liu, Wei Li, Naiqiang Tan, Xiaochun Cao, and Dacheng Tao. O1-pruner: Length-harmonizing fine-tuning for o1-like reasoning pruning. *arXiv preprint arXiv:2501.12570*, 2025.

[66] Pranjal Aggarwal and Sean Welleck. L1: Controlling how long a reasoning model thinks with reinforcement learning. *arXiv preprint arXiv:2503.04697*, 2025.

[67] Daman Arora and Andrea Zanette. Training language models to reason efficiently. *arXiv preprint arXiv:2502.04463*, 2025.

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

[69] Yi Shen, Jian Zhang, Jieyun Huang, Shuming Shi, Wenjing Zhang, Jiangze Yan, Ning Wang, Kai Wang, and Shiguo Lian. Dast: Difficulty-adaptive slow-thinking for large reasoning models. *arXiv preprint arXiv:2503.04472*, 2025.

[70] Yuxiao Qu, Matthew YR Yang, Amrith Setlur, Lewis Tunstall, Edward Emanuel Beeching, Ruslan Salakhutdinov, and Aviral Kumar. Optimizing test-time compute via meta reinforcement fine-tuning. *arXiv preprint arXiv:2503.07572*, 2025.

[71] Yingqian Cui, Pengfei He, Jingying Zeng, Hui Liu, Xianfeng Tang, Zhenwei Dai, Yan Han, Chen Luo, Jing Huang, Zhen Li, et al. Stepwise perplexity-guided refinement for efficient chain-of-thought reasoning in large language models. *arXiv preprint arXiv:2502.13260*, 2025.

[72] Muzhi Dai, Shixuan Liu, and Qingyi Si. Stable reinforcement learning for efficient reasoning. *arXiv preprint arXiv:2505.18086*, 2025.

[73] Rafael Rafailov, Archit Sharma, Eric Mitchell, Stefano Ermon, Christopher D. Manning, and Chelsea Finn. Direct preference optimization: Your language model is secretly a reward model, 2024.

[74] Yu Meng, Mengzhou Xia, and Danqi Chen. Simpo: Simple preference optimization with a reference-free reward, 2024.

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

[76] Yizeng Han, Gao Huang, Shiji Song, Le Yang, Honghui Wang, and Yulin Wang. Dynamic neural networks: A survey. *IEEE transactions on pattern analysis and machine intelligence*, 44(11):7436–7456, 2021.

[77] Mostafa Elhoushi, Akshat Shrivastava, Diana Liskovich, Basil Hosmer, Bram Wasti, Liangzhen Lai, Anas Mahmoud, Bilge Acun, Saurabh Agarwal, Ahmed Roman, et al. Layerskip: Enabling early exit inference and self-speculative decoding. In *Proceedings of the 62nd Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, pages 12622–12642, 2024.

[78] Wangbo Zhao, Yizeng Han, Jiasheng Tang, Zhikai Li, Yibing Song, Kai Wang, Zhangyang Wang, and Yang You. A stitch in time saves nine: Small vlm is a precise guidance for accelerating large vlms. In *Proceedings of the Computer Vision and Pattern Recognition Conference*, pages 19814–19824, 2025.

NeurIPS Paper Checklist

1. **Claims**

   Question: Do the main claims made in the abstract and introduction accurately reflect the paper's contributions and scope?

   Answer: [Yes]

   Justification: The abstract and introduction have already claimed our contributions.

   Guidelines:

   - The answer NA means that the abstract and introduction do not include the claims made in the paper.
   - The abstract and/or introduction should clearly state the claims made, including the contributions made in the paper and important assumptions and limitations. A No or NA answer to this question will not be perceived well by the reviewers.
   - The claims made should match theoretical and experimental results, and reflect how much the results can be expected to generalize to other settings.
   - It is fine to include aspirational goals as motivation as long as it is clear that these goals are not attained by the paper.

2. **Limitations**

   Question: Does the paper discuss the limitations of the work performed by the authors?

   Answer: [Yes]

   Justification: There is a limitation section in the appendix.

   Guidelines:

   - The answer NA means that the paper has no limitation while the answer No means that the paper has limitations, but those are not discussed in the paper.
   - The authors are encouraged to create a separate "Limitations" section in their paper.
   - The paper should point out any strong assumptions and how robust the results are to violations of these assumptions (e.g., independence assumptions, noiseless settings, model well-specification, asymptotic approximations only holding locally). The authors should reflect on how these assumptions might be violated in practice and what the implications would be.
   - The authors should reflect on the scope of the claims made, e.g., if the approach was only tested on a few datasets or with a few runs. In general, empirical results often depend on implicit assumptions, which should be articulated.
   - The authors should reflect on the factors that influence the performance of the approach. For example, a facial recognition algorithm may perform poorly when image resolution is low or images are taken in low lighting. Or a speech-to-text system might not be used reliably to provide closed captions for online lectures because it fails to handle technical jargon.
   - The authors should discuss the computational efficiency of the proposed algorithms and how they scale with dataset size.
   - If applicable, the authors should discuss possible limitations of their approach to address problems of privacy and fairness.
   - While the authors might fear that complete honesty about limitations might be used by reviewers as grounds for rejection, a worse outcome might be that reviewers discover limitations that aren't acknowledged in the paper. The authors should use their best judgment and recognize that individual actions in favor of transparency play an important role in developing norms that preserve the integrity of the community. Reviewers will be specifically instructed to not penalize honesty concerning limitations.

3. **Theory assumptions and proofs**

   Question: For each theoretical result, does the paper provide the full set of assumptions and a complete (and correct) proof?

   Answer: [NA]

Justification: The paper does not propose theoretical type methods.

Guidelines:

- The answer NA means that the paper does not include theoretical results.
- All the theorems, formulas, and proofs in the paper should be numbered and cross-referenced.
- All assumptions should be clearly stated or referenced in the statement of any theorems.
- The proofs can either appear in the main paper or the supplemental material, but if they appear in the supplemental material, the authors are encouraged to provide a short proof sketch to provide intuition.
- Inversely, any informal proof provided in the core of the paper should be complemented by formal proofs provided in appendix or supplemental material.
- Theorems and Lemmas that the proof relies upon should be properly referenced.

4. **Experimental result reproducibility**

   Question: Does the paper fully disclose all the information needed to reproduce the main experimental results of the paper to the extent that it affects the main claims and/or conclusions of the paper (regardless of whether the code and data are provided or not)?

   Answer: [Yes]

   Justification: Section 3.1 details the base model, data, and parameters used for training.

   Guidelines:

   - The answer NA means that the paper does not include experiments.
   - If the paper includes experiments, a No answer to this question will not be perceived well by the reviewers: Making the paper reproducible is important, regardless of whether the code and data are provided or not.
   - If the contribution is a dataset and/or model, the authors should describe the steps taken to make their results reproducible or verifiable.
   - Depending on the contribution, reproducibility can be accomplished in various ways. For example, if the contribution is a novel architecture, describing the architecture fully might suffice, or if the contribution is a specific model and empirical evaluation, it may be necessary to either make it possible for others to replicate the model with the same dataset, or provide access to the model. In general. releasing code and data is often one good way to accomplish this, but reproducibility can also be provided via detailed instructions for how to replicate the results, access to a hosted model (e.g., in the case of a large language model), releasing of a model checkpoint, or other means that are appropriate to the research performed.
   - While NeurIPS does not require releasing code, the conference does require all submissions to provide some reasonable avenue for reproducibility, which may depend on the nature of the contribution. For example
     (a) If the contribution is primarily a new algorithm, the paper should make it clear how to reproduce that algorithm.
     (b) If the contribution is primarily a new model architecture, the paper should describe the architecture clearly and fully.
     (c) If the contribution is a new model (e.g., a large language model), then there should either be a way to access this model for reproducing the results or a way to reproduce the model (e.g., with an open-source dataset or instructions for how to construct the dataset).
     (d) We recognize that reproducibility may be tricky in some cases, in which case authors are welcome to describe the particular way they provide for reproducibility. In the case of closed-source models, it may be that access to the model is limited in some way (e.g., to registered users), but it should be possible for other researchers to have some path to reproducing or verifying the results.

5. **Open access to data and code**

   Question: Does the paper provide open access to the data and code, with sufficient instructions to faithfully reproduce the main experimental results, as described in supplemental material?

Answer: [No]

Justification: We will release code once accepted.

Guidelines:

- The answer NA means that paper does not include experiments requiring code.
- Please see the NeurIPS code and data submission guidelines (`https://nips.cc/public/guides/CodeSubmissionPolicy`) for more details.
- While we encourage the release of code and data, we understand that this might not be possible, so "No" is an acceptable answer. Papers cannot be rejected simply for not including code, unless this is central to the contribution (e.g., for a new open-source benchmark).
- The instructions should contain the exact command and environment needed to run to reproduce the results. See the NeurIPS code and data submission guidelines (`https://nips.cc/public/guides/CodeSubmissionPolicy`) for more details.
- The authors should provide instructions on data access and preparation, including how to access the raw data, preprocessed data, intermediate data, and generated data, etc.
- The authors should provide scripts to reproduce all experimental results for the new proposed method and baselines. If only a subset of experiments are reproducible, they should state which ones are omitted from the script and why.
- At submission time, to preserve anonymity, the authors should release anonymized versions (if applicable).
- Providing as much information as possible in supplemental material (appended to the paper) is recommended, but including URLs to data and code is permitted.

6. **Experimental setting/details**

Question: Does the paper specify all the training and test details (e.g., data splits, hyperparameters, how they were chosen, type of optimizer, etc.) necessary to understand the results?

Answer: [Yes]

Justification: Section 3.1 details the base model, data, and parameters used for training.

Guidelines:

- The answer NA means that the paper does not include experiments.
- The experimental setting should be presented in the core of the paper to a level of detail that is necessary to appreciate the results and make sense of them.
- The full details can be provided either with the code, in appendix, or as supplemental material.

7. **Experiment statistical significance**

Question: Does the paper report error bars suitably and correctly defined or other appropriate information about the statistical significance of the experiments?

Answer: [Yes]

Justification: We performed multiple samplings during evaluation and took the average as the result (Section 3.1).

Guidelines:

- The answer NA means that the paper does not include experiments.
- The authors should answer "Yes" if the results are accompanied by error bars, confidence intervals, or statistical significance tests, at least for the experiments that support the main claims of the paper.
- The factors of variability that the error bars are capturing should be clearly stated (for example, train/test split, initialization, random drawing of some parameter, or overall run with given experimental conditions).
- The method for calculating the error bars should be explained (closed form formula, call to a library function, bootstrap, etc.)
- The assumptions made should be given (e.g., Normally distributed errors).

- It should be clear whether the error bar is the standard deviation or the standard error of the mean.
- It is OK to report 1-sigma error bars, but one should state it. The authors should preferably report a 2-sigma error bar than state that they have a 96% CI, if the hypothesis of Normality of errors is not verified.
- For asymmetric distributions, the authors should be careful not to show in tables or figures symmetric error bars that would yield results that are out of range (e.g. negative error rates).
- If error bars are reported in tables or plots, The authors should explain in the text how they were calculated and reference the corresponding figures or tables in the text.

8. **Experiments compute resources**

   Question: For each experiment, does the paper provide sufficient information on the computer resources (type of compute workers, memory, time of execution) needed to reproduce the experiments?

   Answer: [Yes]

   Justification: We wrote the details in the appendix.

   Guidelines:

   - The answer NA means that the paper does not include experiments.
   - The paper should indicate the type of compute workers CPU or GPU, internal cluster, or cloud provider, including relevant memory and storage.
   - The paper should provide the amount of compute required for each of the individual experimental runs as well as estimate the total compute.
   - The paper should disclose whether the full research project required more compute than the experiments reported in the paper (e.g., preliminary or failed experiments that didn't make it into the paper).

9. **Code of ethics**

   Question: Does the research conducted in the paper conform, in every respect, with the NeurIPS Code of Ethics `https://neurips.cc/public/EthicsGuidelines`?

   Answer: [Yes]

   Justification: We have followed the NeurIPS Code of Ethics.

   Guidelines:

   - The answer NA means that the authors have not reviewed the NeurIPS Code of Ethics.
   - If the authors answer No, they should explain the special circumstances that require a deviation from the Code of Ethics.
   - The authors should make sure to preserve anonymity (e.g., if there is a special consideration due to laws or regulations in their jurisdiction).

10. **Broader impacts**

    Question: Does the paper discuss both potential positive societal impacts and negative societal impacts of the work performed?

    Answer: [NA]

    Justification: There is no societal impact of the work performed.

    Guidelines:

    - The answer NA means that there is no societal impact of the work performed.
    - If the authors answer NA or No, they should explain why their work has no societal impact or why the paper does not address societal impact.
    - Examples of negative societal impacts include potential malicious or unintended uses (e.g., disinformation, generating fake profiles, surveillance), fairness considerations (e.g., deployment of technologies that could make decisions that unfairly impact specific groups), privacy considerations, and security considerations.

- The conference expects that many papers will be foundational research and not tied to particular applications, let alone deployments. However, if there is a direct path to any negative applications, the authors should point it out. For example, it is legitimate to point out that an improvement in the quality of generative models could be used to generate deepfakes for disinformation. On the other hand, it is not needed to point out that a generic algorithm for optimizing neural networks could enable people to train models that generate Deepfakes faster.
- The authors should consider possible harms that could arise when the technology is being used as intended and functioning correctly, harms that could arise when the technology is being used as intended but gives incorrect results, and harms following from (intentional or unintentional) misuse of the technology.
- If there are negative societal impacts, the authors could also discuss possible mitigation strategies (e.g., gated release of models, providing defenses in addition to attacks, mechanisms for monitoring misuse, mechanisms to monitor how a system learns from feedback over time, improving the efficiency and accessibility of ML).

11. **Safeguards**

    Question: Does the paper describe safeguards that have been put in place for responsible release of data or models that have a high risk for misuse (e.g., pretrained language models, image generators, or scraped datasets)?

    Answer: [NA]

    Justification: The paper poses no such risks.

    Guidelines:

    - The answer NA means that the paper poses no such risks.
    - Released models that have a high risk for misuse or dual-use should be released with necessary safeguards to allow for controlled use of the model, for example by requiring that users adhere to usage guidelines or restrictions to access the model or implementing safety filters.
    - Datasets that have been scraped from the Internet could pose safety risks. The authors should describe how they avoided releasing unsafe images.
    - We recognize that providing effective safeguards is challenging, and many papers do not require this, but we encourage authors to take this into account and make a best faith effort.

12. **Licenses for existing assets**

    Question: Are the creators or original owners of assets (e.g., code, data, models), used in the paper, properly credited and are the license and terms of use explicitly mentioned and properly respected?

    Answer: [Yes]

    Justification: We cited the paper of the assets we used.

    Guidelines:

    - The answer NA means that the paper does not use existing assets.
    - The authors should cite the original paper that produced the code package or dataset.
    - The authors should state which version of the asset is used and, if possible, include a URL.
    - The name of the license (e.g., CC-BY 4.0) should be included for each asset.
    - For scraped data from a particular source (e.g., website), the copyright and terms of service of that source should be provided.
    - If assets are released, the license, copyright information, and terms of use in the package should be provided. For popular datasets, `paperswithcode.com/datasets` has curated licenses for some datasets. Their licensing guide can help determine the license of a dataset.
    - For existing datasets that are re-packaged, both the original license and the license of the derived asset (if it has changed) should be provided.

- If this information is not available online, the authors are encouraged to reach out to the asset's creators.

13. **New assets**

    Question: Are new assets introduced in the paper well documented and is the documentation provided alongside the assets?

    Answer: [NA]

    Justification: The paper does not release new assets.

    Guidelines:

    - The answer NA means that the paper does not release new assets.
    - Researchers should communicate the details of the dataset/code/model as part of their submissions via structured templates. This includes details about training, license, limitations, etc.
    - The paper should discuss whether and how consent was obtained from people whose asset is used.
    - At submission time, remember to anonymize your assets (if applicable). You can either create an anonymized URL or include an anonymized zip file.

14. **Crowdsourcing and research with human subjects**

    Question: For crowdsourcing experiments and research with human subjects, does the paper include the full text of instructions given to participants and screenshots, if applicable, as well as details about compensation (if any)?

    Answer: [NA]

    Justification: The paper does not involve crowdsourcing nor research with human subjects.

    Guidelines:

    - The answer NA means that the paper does not involve crowdsourcing nor research with human subjects.
    - Including this information in the supplemental material is fine, but if the main contribution of the paper involves human subjects, then as much detail as possible should be included in the main paper.
    - According to the NeurIPS Code of Ethics, workers involved in data collection, curation, or other labor should be paid at least the minimum wage in the country of the data collector.

15. **Institutional review board (IRB) approvals or equivalent for research with human subjects**

    Question: Does the paper describe potential risks incurred by study participants, whether such risks were disclosed to the subjects, and whether Institutional Review Board (IRB) approvals (or an equivalent approval/review based on the requirements of your country or institution) were obtained?

    Answer: [NA]

    Justification: The paper does not involve crowdsourcing nor research with human subjects.

    Guidelines:

    - The answer NA means that the paper does not involve crowdsourcing nor research with human subjects.
    - Depending on the country in which research is conducted, IRB approval (or equivalent) may be required for any human subjects research. If you obtained IRB approval, you should clearly state this in the paper.
    - We recognize that the procedures for this may vary significantly between institutions and locations, and we expect authors to adhere to the NeurIPS Code of Ethics and the guidelines for their institution.
    - For initial submissions, do not include any information that would break anonymity (if applicable), such as the institution conducting the review.

16. **Declaration of LLM usage**

Question: Does the paper describe the usage of LLMs if it is an important, original, or non-standard component of the core methods in this research? Note that if the LLM is used only for writing, editing, or formatting purposes and does not impact the core methodology, scientific rigorousness, or originality of the research, declaration is not required.

Answer: [Yes]

Justification: Section 3.1 details the base model used for training.

Guidelines:

- The answer NA means that the core method development in this research does not involve LLMs as any important, original, or non-standard components.
- Please refer to our LLM policy (`https://neurips.cc/Conferences/2025/LLM`) for what should or should not be described.

A An example of Training Data

Here is an example that shows the training data truncation and decaying reward assignment.

Q: Find the smallest integer n such that the cyclic group C_{12} is a subgroup of the symmetric group S_n. Provide your answer as a single integer.

Full Thought Rollout:
Okay, so I need to find the smallest integer n such that the cyclic group C_{12} is a subgroup of the symmetric group S_n. Hmm, let me think. \n\nFirst, I remember that every group can be embedded into some symmetric group Thinking Exit !

⊕ Time is limited, stop thinking and start answering.\n</think>\n\n
Truncated Thought Rollout:
.../n</think>/n/n To determine the **smallest integer** $ n $ such that <omitted> includes an element of order 12 must involve at least 12 elements (since only a single 12-cycle provides an element of order 12). Hence, $ C_{12} $ **cannot** be embedded in any $ S_n $ with $ n < 12 $.\n\n---\n\n### Final Answer:\n\n$$\n\\boxed{12}\n$$ **Reward: 0**

by Cayley's theorem, which says that any group is isomorphic to a subgroup of the symmetric group on its own elements. So for C_{12}, that would mean it's a subgroup of S_{12}. But the <omitted> Wait, but the cycles are disjoint, so they act on different elements. So, the total number of elements is 6 + 4 = 10. Thinking Exit !

⊕ Time is limited, stop thinking and start answering.\n</think>\n\n
Truncated Thought Rollout:
.../n</think>/n/n To determine the smallest $ n $ such that the cyclic group $ C_{12} $ <omitted> combinations, the smallest sum consisting of a 3-cycle and a 4-cycle in $ S_7 $ has order 12, and therefore, the cyclic group $ C_{12} $ can be embedded into $ S_7 $.\n\n---\n\n### Final Answer:\n$$\n\\boxed{7}\n$$ **Reward: 1**

Therefore, such permutation exists in S_{10}. \n\nAlternatively, could we do better? Let me think. <omitted> Or 12 = 2 * 6, LCM 6, which is not 12. Or 3 * 4, LCM 12. 3 + 4 = 7. \n\nAlternatively, 4 * 3. Same thing. \n\nAlternatively, 12 as 2 * 2 * 3, which we saw has LCM 6. \n\nAlternatively, 12 as 2 * 3 * 2. Same thing. \n\n Thinking Exit !

⊕ Time is limited, stop thinking and start answering.\n</think>\n\n
Truncated Thought Rollout:
.../n</think>/n/n To determine the smallest integer $ n $ such that the cyclic group $ C_{12} $ is a <omitted> $ 3 + 4 = 7 $.\n\n Hence, the smallest $ n $ for which $ S_n $ contains a permutation of order 12 (and thus contains $ C_{12} $ as a subgroup) is:\n\n $$\n\\boxed{7}\n$$ **Reward: 1/2**

Is there a way to get LCM 12 with smaller sum? For example, <omitted> 3. Same thing. \n\nAlternatively, 12 as 2 * 2 * 3, which we saw has LCM 6. \n\nAlternatively, 12 as 2 * 3 * 2. Same thing. \n\nIs there a way to get LCM 12 with smaller sum? For example, if I have cycles of length 12, that's sum 12. 3 + Thinking Exit !

⊕ Time is limited, stop thinking and start answering.\n</think>\n\n
Truncated Thought Rollout:
.../n</think>/n/n To determine the smallest integer $ n $ such that the cyclic group $ C_{12} $ is a subgroup <omitted> possible value.\n\n---\n\n### Final Answer:\n$$\n\\boxed{7}\n$$ **Reward: 1/4**

4 = 7. Is there a way to get LCM 12 with a total cycle length less than 7?\n\nLet me think. For example, 2 and 6? LCM is 6. Not enough. 3 and 4: LCM 12, sum 7. If I try to use more cycles. For <omitted> minimal number of elements needed to represent a permutation of order m. \n\nAlternatively, for the cyclic group C_m, the minimal n is Thinking Exit !

⊕ Time is limited, stop thinking and start answering.\n</think>\n\n
Truncated Thought Rollout:
.../n</think>/n/n To determine the smallest integer $ n $ such that the cyclic group $ C_{12} $ is a subgroup of the symmetric <omitted> and no smaller symmetric group $ S_n $, for $ n < 7 $, includes such an element, we conclude that the **smallest** $ n $ for which $ C_{12} $ is a subgroup of $ S_n $ is:\n\n$$\n\\boxed{7}\n$$ **Reward: 1/8**

the minimal number such that m is the LCM of some integers whose sum is n. \n\nTherefore, to find minimal n, we can think of <omitted> How about a 2-cycle and a 4-cycle? LCM(2,4)=4. Still not 8. How about a 8-cycle? Thinking Exit !

⊕ Time is limited, stop thinking and start answering.\n</think>\n\n
Truncated Thought Rollout:
.../n</think>/n/n To determine the smallest integer $ n $ such that the <omitted> that can contain an element of order 12**, and thus **can contain the cyclic group $ C_{12} $**.\n\n---\n\n### Final Answer:\n\n$$\n\\boxed{7}\n$$ **Reward: 1/16**

So only possible if n=8. \n\nTherefore, <omitted>sum is indeed 7. \n\nTherefore, the answer is 7. \n\nBut just to be thorough, let me check n=7 and n=8. \n\nIn S_7, we Thinking Exit !

⊕ Time is limited, stop thinking and start answering.\n</think>\n\n
Truncated Thought Rollout:
.../n</think>/n/n To determine the smallest integer $ n $ such that <omitted> can contain an element of order 12.\n\n---\n\n### Final Answer:\n\n$$\n\\boxed{7}\n$$ **Reward: 1/32**

can have a permutation of order 12. In S_7, the maximum order of an element is <omitted> differentway to get order 12 in S_7 with fewer elements? For example, by having a single cycle? But no, because a single cycle in S_7 can have maximum Thinking Exit !

⊕ Time is limited, stop thinking and start answering.\n</think>\n\n
Truncated Thought Rollout:
.../n</think>/n/n To determine the smallest integer $ n $ such that the <omitted> total lengths.\n\n---\n\n### Conclusion:\n\nThe smallest number $ n $ for which $ C_{12} $ is a subgroup of $ S_n $ is:\n\n$$\n\\boxed{7}\n$$ **Reward: 1/64**

length 7, which has order 7, which is less than 12. <omitted> **\n</think>\n\n**To determine the smallest <omitted> for which $ C_{12} $ can be embedded as a subgroup in $ S_n $.\n\n---\n\n### Final Answer\n\n$$\n\\boxed{7}\n$$ **Reward: 1/128**

Figure 5: An example shows the training data truncation and decaying reward assignment.

B Computation Resource

In our experiments, 64×80g memory was used to train the models.

C Robustness to Prompt Template Variations

We evaluated the sensitivity of our thinking path truncation mechanism to different prompt templates. Three variations were tested: (1) an explicit time constraint prompt, (2) a detailed explanation prompt, and (3) a minimal delimiter-only prompt. All variants included the `\n</think>\n\n` delimiter.

Table 3: Performance across different prompt templates for thinking path truncation.

Prompt Template	AIME 2024		MATH-500	
	Acc	Tokens	Acc	Tokens
Qwen3-8B with S-GRPO				
Time is limited, stop thinking and start answering. `\n</think>\n\n`	77.3	8,810	95.2	3,166
Considering the limited time by the user, I have to give the solution based on the thinking directly now. `\n</think>\n\n`	77.1	8,780	95.0	3,202
`\n</think>\n\n`	76.9	8,823	95.3	3,154

The results demonstrate remarkable consistency across all prompt variations. Accuracy differences are negligible, and token usage remains stable. This indicates that the `\n</think>\n\n` delimiter, rather than the specific prompt wording, is the key factor in controlling thinking path truncation. Such robustness to prompt engineering choices enhances the practical applicability of our approach.

D Comparison of Truncation Strategies

We investigated different strategies for truncating thinking paths beyond random truncation. Specifically, we explored semantic-based truncation at specific markers that indicate thought shifts, such as "Wait" and "Alternatively" tokens, following similar approaches in prior work [75].

Table 4: Performance comparison of different truncation strategies.

Truncation Strategy	AIME 2024		MATH-500	
	Acc	Tokens	Acc	Tokens
Qwen3-8B with S-GRPO				
Random selection	77.3	8,810	95.2	3,166
"Wait" or "Alternatively" position	76.7	8,930	95.2	3,378

While semantic-based truncation achieved comparable performance to random truncation, it exhibited critical stability issues during training. As the model improves through training iterations, reasoning chains become increasingly compressed and streamlined, resulting in a significant reduction of transition tokens like "Wait" and "Alternatively". This progressive decrease creates a practical limitation: for certain queries, insufficient semantic markers are available to meet the sampling requirements for the second rollout phase (requiring n samples). Consequently, semantic-based truncation becomes unreliable as training progresses.

In contrast, random truncation maintains consistent applicability throughout training regardless of how compressed the reasoning becomes. The combination of stable training dynamics and comparable performance validates our choice of random truncation as the primary strategy in the implementation of S-GRPO.

E Computational Efficiency of Dual-Rollout Training

We measured the actual time cost of S-GRPO's dual-rollout approach compared to standard GRPO on Qwen3-8B training with rollout n=8.

Despite requiring two inference passes, S-GRPO incurs only 20% additional time overhead rather than the 2× increase. This efficiency comes from two factors: (1) The second rollout only generates the

Table 5: Time cost comparison between GRPO and S-GRPO.

Method	Average Rollout Time (s)	Average Time Increase	Relative Overhead
Qwen3-8B Training			
GRPO (single rollout)	457	-	1.00×
S-GRPO (dual rollout)	550	+93s	1.20×

shorter conclusion portion, not the full response, reducing decoding time proportionally to sequence length; (2) The truncated thinking path uses efficient prefill computation rather than sequential decoding.

F Effectiveness Across Different Domains

Beyond scientific and mathematical reasoning, we extended our evaluation to MMLU-Pro (general knowledge) and LiveCodeBench v5 (code generation, 2024.10-2025.02).

Table 6: Performance of S-GRPO across different task domains.

Model	MMLU-Pro		LiveCodeBench v5	
	Acc	Tokens	Acc	Tokens
DeepSeek-R1-Distill-Qwen-7B				
Vanilla	52.4	3,646	35.8	12,255
S-GRPO	52.4	1,597	35.9	7,692
Qwen3-8B				
Vanilla	75.1	4,422	56.2	14,794
S-GRPO	75.2	2,313	56.2	10,203

The results demonstrate that S-GRPO also maintains accuracy while achieving substantial token reduction across other domains. On MMLU-Pro, token usage decreased by 56.2% and 47.7% for the two models respectively, with accuracy preserved or slightly improved. Similarly, on LiveCodeBench, S-GRPO reduced tokens by 37.2% and 31.0% while maintaining code generation performance.

G Limitations and Future works

In the later stages of training, S-GRPO encounters a bottleneck: the average reward begins to decline as the response length shortens. This is because the randomly truncated early stopping points become increasingly short, and excessively short CoTs are less likely to generate correct answers. In future work, optimizing the truncation strategy to replace the uniform sampling of truncation points could provide improvements.

H Response Length and Accuracy Curve

The Figure 6 below illustrates the response length and accuracy trends on the AIME 2024 dataset across different training steps for models trained with the RL with length penalty method and the S-GRPO method. The base model used is DeepSeek-R1-Distill-Qwen-7B. Both methods demonstrate a similar downward trend in response length. However, the RL with length penalty method exhibits an earlier decline in accuracy, highlighting the superior performance of the S-GRPO approach.

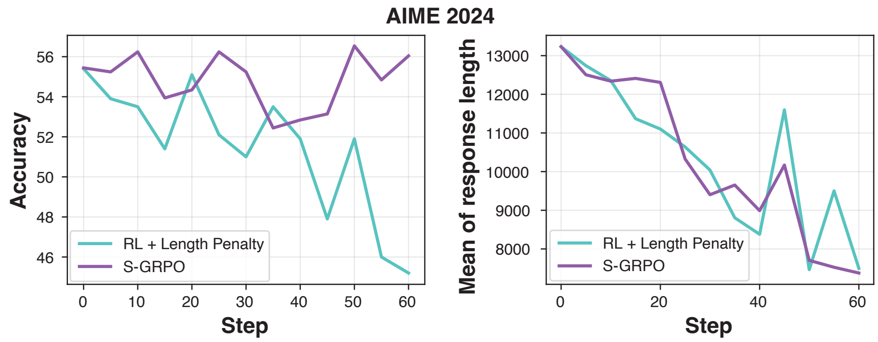

Figure 6: The response length and accuracy trends on the AIME 2024 dataset across different training steps for models trained with the RL with length penalty method and the S-GRPO method.

I More Case Study

To further demonstrate the effectiveness of S-GRPO in comparison to vanilla CoT and its truncated variant, we present two additional examples from the AIME 2024 benchmark in Figure 7. These cases highlight the advantages of S-GRPO in solving complex reasoning tasks with a significantly reduced token budget while maintaining accuracy.

In the first example (top row), all three methods attempt to find the greatest real number satisfying a geometric constraint. The vanilla CoT model produces the correct answer but requires a substantial thinking budget (12,015 tokens). When truncated to match the thinking budget of S-GRPO (6,167 tokens), the vanilla CoT fails to reach the correct conclusion. In contrast, S-GRPO successfully identifies the correct solution path and arrives at the correct answer using only 6,167 thinking budget (7,214 total token budget), demonstrating its ability to achieve efficiency without sacrificing accuracy.

In the second example (bottom row), the task involves calculating the length of a geometric segment based on given constraints. Vanilla CoT again fails when truncated to the same thinking budget as S-GRPO (9,333 tokens), producing an incorrect answer. Meanwhile, S-GRPO reaches the correct solution with a minimal total token budget (10,836 tokens), significantly outperforming vanilla CoT's full reasoning process, which consumes 28,171 tokens without arriving at the correct solution.

These examples further validate that S-GRPO effectively balances reasoning depth and efficiency, avoiding overthinking. By guiding the model to concise and accurate reasoning paths, S-GRPO consistently outperforms baseline methods in challenging reasoning tasks.

J More Related Work

Dynamic and early-exit architectures. There are works exploring dynamic neural networks and early-exit mechanisms to improve inference efficiency. Dynamic neural networks [76] provide a general framework for adaptively adjusting computation based on input complexity, enabling selective activation of subnetworks or layers during inference. Building upon this concept, LayerSkip [77] introduces inter-layer early-exit strategies that allow large language models to skip redundant transformer layers dynamically, achieving faster decoding without substantial performance loss. While LayerSkip focuses on inter-layer early-exit, S-GRPO operates at the reasoning-path level.

More recently, hybrid approaches such as *A Stitch in Time Saves Nine* [78] leverage small vision–language models to guide the early termination of reasoning in larger models, serving as precise external controllers that accelerate computation. These studies collectively demonstrate the potential of adaptive computation and early-exit mechanisms to balance efficiency and accuracy in large-scale models, inspiring further exploration into reasoning-level early stopping and adaptive generation. S-GRPO eliminates the need for small model assistance by internalizing early-exit capabilities directly into the reasoning model through reinforcement learning.

Figure 7: Comparison of two generated content samples on AIME 2024.