Stabilizing Efficient Reasoning with Step-Level Advantage Selection

Han Wang[1]*, **Xiaodong Yu**[2], **Jialian Wu**[2], **Jiang Liu**[2],
Ximeng Sun[2], **Mohit Bansal**[1], **Zicheng Liu**[2]
[1]UNC Chapel Hill, [2]Advanced Micro Devices, Inc.
hwang@cs.unc.edu

Abstract

Large language models (LLMs) achieve strong reasoning performance by allocating substantial computation at inference time, often generating long and verbose reasoning traces. While recent work on efficient reasoning reduces this overhead through length-based rewards or pruning, many approaches are post-trained under a much shorter context window than base-model training, a factor whose effect has not been systematically isolated. We first show that short-context post-training alone, using standard GRPO without any length-aware objective, already induces substantial reasoning compression—but at the cost of increasingly unstable training dynamics and accuracy degradation. To address this, we propose Step-level Advantage Selection (SAS), which operates at the reasoning-step level and assigns a zero advantage to low-confidence steps in correct rollouts and to high-confidence steps in verifier-failed rollouts, where failures often arise from truncation or verifier issues rather than incorrect reasoning. Across diverse mathematical and general reasoning benchmarks, SAS reduces average reasoning length by over 30% while improving Pass@1 accuracy by 3.79 points over the strongest length-aware baseline, yielding a better accuracy–efficiency trade-off.[1]

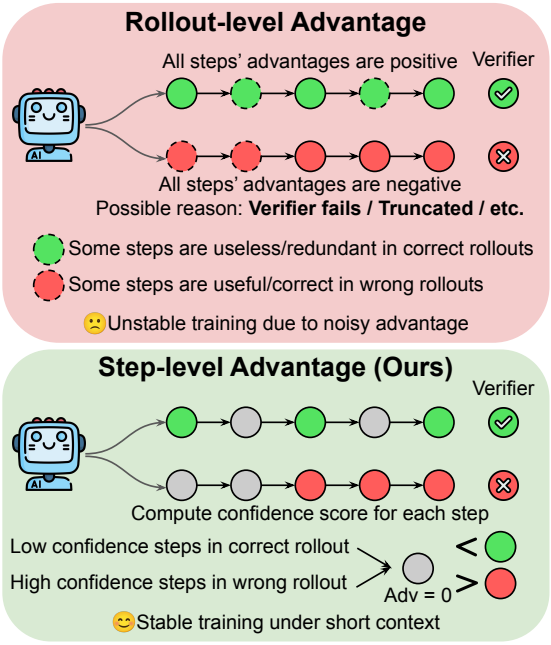


Figure 1: Rollout-level versus step-level advantage selection. Standard GRPO propagates rewards uniformly based on the final verifier outcome: under group-relative normalization, all steps in correct rollouts receive positive advantages and all steps in verifier-failed rollouts receive negative advantages, suppressing useful intermediate reasoning in failed rollouts and over-updating redundant steps in correct rollouts. In contrast, Step-level Advantage Selection (SAS) assigns a zero advantage to low-confidence steps in correct rollouts (below their positive peers) and to high-confidence steps in verifier-failed rollouts (above their negative peers), resulting in more stable training and efficient reasoning compression under short context.

1 Introduction

Large language models (LLMs) have demonstrated strong reasoning capabilities across a wide range of tasks, including mathematical problem solving, logical reasoning, and code generation. Recent progress has shown that allocating more computation at inference time—commonly referred to as test-time scaling—can substantially improve reasoning performance by encouraging models to generate longer chain-of-thought traces or explore multiple reasoning paths (Wei et al., 2022; Kojima et al., 2022; Wang et al., 2023; Yao et al., 2023). However, these gains often come at a significant computational cost, as models tend to produce excessively long and verbose reasoning even for relatively simple problems, leading to increased inference latency and reduced practical efficiency (Chen et al., 2025; Gema et al., 2025; Ghosal et al., 2025).

*Work done during internship at AMD.
[1]Our code is publicly available at https://github.com/HanNight/SAS

Findings of the Association for Computational Linguistics: ACL 2026, pages 26754–26765

To address this issue, a growing body of work has focused on efficient reasoning, which aims to reduce reasoning length while preserving task performance. Many recent approaches adopt reinforcement learning–based post-training strategies that explicitly incorporate length-aware objectives, such as token-budget constraints, length-aware rewards, or pruning mechanisms (Aggarwal and Welleck, 2025; Wu et al., 2025a; Hou et al., 2025; Sui et al., 2025). While effective, these approaches share a critical, yet overlooked, training condition: they typically conduct post-training within a substantially restricted context window (e.g., 4K tokens), a sharp departure from the expansive windows (e.g., 16K–24K tokens) used during base model training. This discrepancy raises a fundamental question: to what extent does the observed reasoning compression stem from explicit length-aware objectives, as opposed to being a natural consequence of short-context post-training itself?

In this work, we systematically isolate this effect by training a long-context reasoning model (Luo et al., 2025b; Guo et al., 2025) using pure GRPO with short context, deliberately excluding any length-aware rewards or pruning techniques. Our findings reveal that short-context post-training alone serves as a strong and sufficient compression signal, achieving reductions in output length comparable to state-of-the-art efficient reasoning methods—a critical variable previously conflated with explicit length-control objectives. However, this compression from short-context post-training comes at a cost: as training progresses, task accuracy fluctuates and degrades, and exploration collapses into brittle policy updates. We hypothesize that this instability stems from truncation within the restricted context window: truncated rollouts receive zero reward despite often containing correct intermediate reasoning, thereby implicitly penalizing correct steps, leading to unstable behavior and degraded task performance. Short-context post-training alone is therefore effective at compression but insufficient for stable, efficient reasoning.

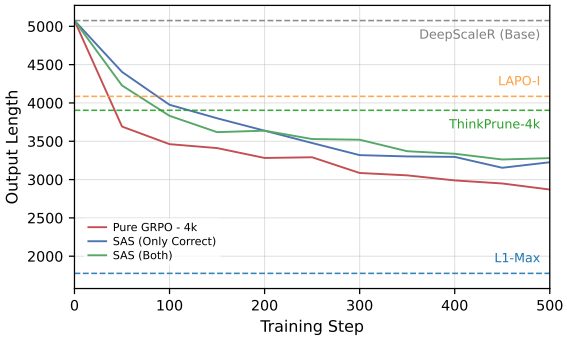
To resolve this tension between aggressive reasoning compression and stable performance preservation, we introduce Step-level Advantage Selection (SAS) (Figure 1), which treats reasoning as a sequence of discrete, evaluable steps. As illustrated in Figure 1, rather than applying a uniform advantage to an entire trace, our method operates at the reasoning-step level to selectively filter out unreliable steps. Specifically, we assign a zero advantage to low-confidence reasoning steps in correct rollouts, and to high-confidence intermediate steps within verifier-failed rollouts–which may fail due to truncation under short context or verifier issues rather than incorrect reasoning. Under GRPO's group-relative advantage normalization, assigning zero advantages has asymmetric effects: it sits below the positive advantages of peers in correct rollouts (suppressing unreliable steps) and above the negative advantages of peers in verifier-failed rollouts (shielding reliable steps from penalization). By focusing the optimization signal on complete, high-confidence reasoning steps, SAS mitigates noisy updates and preserves performance during aggressive reasoning compression under short context. We evaluate SAS on a diverse set of mathematical and general reasoning benchmarks. Experimental results demonstrate that SAS consistently achieves a superior accuracy–efficiency trade-off compared to existing baselines. Specifically, relative to the base DeepScaleR-1.5B-Preview model, our method reduces the average output length by over 30% while simultaneously improving average Pass@1 accuracy by 1.51 points. Compared to length-aware baselines such as LAPO and ThinkPrune, our approach achieves higher accuracy while generating 15% fewer tokens on average. These improvements are reflected in the Accuracy–Efficiency Score, where our method exceeds all baselines by a clear margin, with an absolute gain of over 0.13 AES compared to the strongest competing approaches.

2 Methodology

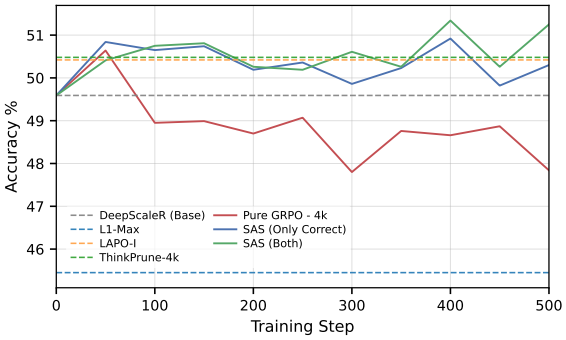
2.1 Preliminary: RL with Length-Aware Reward Design

Recent work on efficient reasoning commonly adopts reinforcement learning (RL) to explicitly control the length of model-generated reasoning. In this setting, a pretrained language model is further optimized using policy optimization, where the training objective combines task-level rewards with regularization terms that constrain deviations from a reference policy. A widely used framework is Group Relative Policy Optimization (GRPO), which optimizes the policy based on relative advantages computed within groups of sampled responses.

Formally, for each prompt $x \sim \mathcal{D}$, GRPO samples a group of G rollouts $\{y_i\}_{i=1}^{G}$ from the current policy π_θ and computes a group-relative advantage



(a) Output length vs. training step.



(b) Accuracy vs. training step.

Figure 2: Training dynamics of average output length and accuracy across six reasoning datasets under short-context (4K) post-training. Note that the dashed horizontal lines just indicate the performance of these baselines and do not represent their training trajectories.

for each rollout by normalizing its reward against the group's reward statistics:

$$\hat{A}_{i,t} = \frac{r(x, y_i) - \text{mean}\big(\{r(x, y_j)\}_{j=1}^{G}\big)}{\text{std}\big(\{r(x, y_j)\}_{j=1}^{G}\big)}, \quad (1)$$

which is shared across all tokens $t \in \{1, \ldots, |y_i|\}$ within the same rollout (i.e., $\hat{A}_{i,t} = \hat{A}_i$) under the outcome-based reward $r(x, y_i) \in \{0, 1\}$. The policy is then optimized with a PPO-style clipped surrogate over these advantages, regularized by a KL penalty against a fixed reference policy π_{ref}. Crucially, this group-relative normalization ensures that correct rollouts (reward $= 1$) receive **positive** advantages and verifier-failed rollouts (reward $= 0$) receive **negative** advantages.

To encourage concise reasoning, many prior methods incorporate length-aware reward design into the RL framework. A generic form of such rewards can be written as:

$$r(x, y) = r_{\text{task}}(x, y) - \lambda \cdot g(|y|), \quad (2)$$

where $r_{\text{task}}(x, y)$ measures task correctness, $|y|$ denotes the output length, and $g(\cdot)$ is a monotonic function that penalizes longer reasoning traces. Different methods instantiate $g(\cdot)$ using target lengths, token budgets, or length-dependent penalties, and integrate this reward into the advantage estimation during policy optimization. Empirically, these approaches show that reasoning length can be directly influenced through RL-based post-training.

However, several empirical observations complicate this formulation. First, prior work reports that output length often exhibits non-monotonic behavior during RL training, where reasoning length initially increases before decreasing later. Second, most length-control methods perform post-training using a substantially shorter context window (e.g., 4K tokens) compared to the long context used during base-model training (e.g., 16–32K). This context mismatch suggests that models trained under shorter contexts may naturally prefer shorter outputs, independent of the specific length-aware reward design. Together, these factors make it unclear to what extent reasoning compression should be attributed to explicit length rewards versus training dynamics and context length.

2.2 Short-Context Post-Training

To isolate the influence of training context length from explicit length-aware reward design, we conduct a controlled study using pure GRPO without any length-dependent reward or constraint. Starting from a reasoning-capable base model pretrained with a long context window, we perform GRPO-based post-training using a fixed 4k context window, which matches the post-training setup commonly adopted in prior efficient reasoning work. Importantly, our reward function is based solely on task correctness, introducing no explicit signal regarding output length.

Despite the absence of length-aware objectives, short-context post-training alone induces a substantial reduction in reasoning length. As shown in Figure 2a, the average output length decreases sharply during the early stages of training and continues to decline steadily thereafter, eventually reaching a level comparable to or even shorter than existing efficient reasoning baselines such as LAPO and ThinkPrune. At the same time, the trained policy achieves task accuracy that is initially comparable to these methods, as shown in Figure 2b. These re-

sults indicate that short-context post-training itself provides a strong and previously underexamined compression signal, independent of explicit length-control mechanisms.

However, a closer inspection of the training dynamics reveals important limitations. While the reasoning length continues to decrease throughout training, task accuracy becomes increasingly volatile, characterized by noticeable fluctuations and a gradual performance decay in later training stages (Figure 2b). We hypothesize that this instability is tightly coupled to how the short context window interacts with rollout-level credit assignment: when rollouts are truncated before reaching the final answer, the verifier assigns zero reward to traces that may contain largely correct intermediate reasoning, producing noisy and misaligned advantage signals. To quantify how often this occurs, we take 8K-length rollouts from the base model, truncate them to 4K tokens, and re-run the same rule-based verifier. Approximately 29% of originally correct responses become verifier-failed after truncation, with the large majority losing only the final boxed answer or the closing steps of an otherwise correct derivation. This indicates that a substantial fraction of zero-reward rollouts under short-context training are not logically flawed but merely incomplete, and that pure GRPO systematically penalizes their intermediate reasoning — a plausible driver of the observed accuracy degradation.

Taken together, these observations highlight a fundamental tension between reasoning compression and performance stability: short-context post-training is effective at reducing output length, but the very mechanism that produces this compression (truncation) also injects noisy credit into standard RL updates. This motivates a training strategy that retains the compression benefits of short-context post-training while explicitly correcting for truncation-induced credit misassignment — which we develop next.

2.3 Step-level Advantage Selection (SAS)

Motivated by the training instability observed during the short-context post-training (Section 2.2), we propose Step-level Advantage Selection (SAS). The method stabilizes the learning process by selectively modulating the influence of individual reasoning steps on policy updates, without requiring explicit length-aware rewards or architectural changes. SAS operates directly at the advantage level, ensuring that only reliable reasoning steps contribute to the optimization signal.

Mask Advantages in Correct Rollouts. Even when a rollout is verified as correct (reward $= 1$), the reasoning trace may contain redundant or low-confidence steps (e.g., self-doubt detours, repetitive verification) that are weakly related to the outcome. Reinforcing these steps can introduce noisy updates and exacerbate policy drift during training.

For each correct rollout y_i, we partition the generated reasoning trace into a sequence of reasoning steps $\{s_j\}_{j=1}^{N}$, where steps are defined as contiguous text segments separated by double newline delimiters (\n\n)[2]. For each step s_j, we compute a step-level confidence score based on the model's token-level log probabilities. Let $\mathcal{T}_j$ denote the set of token positions belonging to step s_j. The confidence score is defined as:

$$c_j = \frac{1}{|\mathcal{T}_j|} \sum_{\tau \in \mathcal{T}_j} \log \pi_\theta\big(y_\tau \mid x, y_{<\tau}\big), \qquad (3)$$

where π_θ denotes the current policy.

Given the confidence scores $\{c_j\}$, we sort the reasoning steps in ascending order of confidence and select a ratio $r \in (0, 1)$ of the lowest-confidence steps. For all token positions belonging to these steps, we set their advantages to zero:

$$\tilde{A}_{i,\tau} = \begin{cases} 0, & \text{if } \tau \in \mathcal{T}_j \text{ for } s_j \in \mathcal{S}^{+}_{\text{mask}}, \\ \hat{A}_{i,\tau}, & \text{otherwise,} \end{cases} \qquad (4)$$

where $\hat{A}_{i,\tau}$ is the original GRPO advantage from Eq. 1, and $\mathcal{S}^{+}_{\text{mask}}$ denotes the subset of low-confidence steps selected from a verifier-approved rollout. This operation suppresses the contribution of unreliable reasoning steps while preserving learning signals from more reliable ones: the zeroed advantages lie strictly below the positive advantages retained by the remaining high-confidence steps in the same rollout.

Shielding Signals in Incorrect Rollouts. We further extend this idea to rollouts that fail the rule-based verification (reward $= 0$). A key observation in short-context training is that many "incorrect" rollouts are not inherently flawed in logic; rather, they are naturally truncated "correct" traces where the model ran out of context before reaching the final answer (Section 2.2). These traces often contain high-quality intermediate reasoning

[2] See Appendix B for a detailed justification and discussion of using \n\n as the step delimiter.

	AIME24		AIME25		MATH		AMC		OlympiadBench		Average		
	Pass@1	#Tok	Pass@1	#Tok	Pass@1	#Tok	Pass@1	#Tok	Pass@1	#Tok	Pass@1	#Tok	AES
DeepScaleR	33.75	6755	26.88	6444	86.36	2809	67.62	4761	47.23	4824	52.37	5118	0.00
GRPO-4K	38.75	5282	25.42	4812	85.09	1976	**71.69**	3395	47.09	3411	53.61	3775	0.33
L1-Max	25.63	2158	21.67	2032	83.60	1480	64.61	1768	44.69	1703	48.04	1828	0.23
LAPO-I	34.58	5627	**27.92**	5290	86.04	2220	69.73	3765	48.18	3735	53.29	4127	0.25
ThinkPrune-4k	36.04	5468	26.67	5177	86.23	2090	70.18	3636	47.62	3651	53.35	4004	0.27
SAS (ours)	**39.79**	4876	26.67	4295	**86.58**	1768	71.46	3090	**48.19**	3008	**54.54**	3407	**0.46**

Table 1: Comparison of methods across five math reasoning benchmarks. SAS achieves a superior accuracy–efficiency trade-off, reducing reasoning length while maintaining or improving accuracy over strong baselines.

	GPQA-Diamond		LSAT		MMLU		Average		
	Pass@1	#Tok	Pass@1	#Tok	Pass@1	#Tok	Pass@1	#Tok	AES
DeepScaleR	35.70	4860	**28.64**	6934	47.99	1454	37.44	4416	0.00
GRPO-4K	32.48	1515	28.45	5025	48.73	950	36.55	2496	0.32
L1-Max	36.05	3878	26.66	1949	48.94	900	37.22	2242	**0.46**
LAPO-I	36.17	3404	28.42	5382	48.71	1207	37.77	3331	0.27
ThinkPrune-4k	35.83	3278	28.13	5131	48.91	973	37.62	3127	0.31
SAS (ours)	**37.18**	2998	28.32	4312	**49.39**	876	**38.30**	2729	0.45

Table 2: Performance comparison of methods on three out-of-domain general reasoning benchmarks. SAS consistently improves Pass@1 accuracy while substantially reducing reasoning length, resulting in a strong and competitive accuracy–efficiency trade-off under short-context post-training.

steps that are identical to those found in successful solutions. Rather than discarding these traces entirely—which would propagate unwarranted negative credit through all of their steps—we shield their reliable segments from penalization.

For each verifier-rejected rollout, we again compute step-level confidence scores $\{c_j\}$ and sort the steps in descending order of confidence. We then select a ratio r of the highest-confidence steps and set their advantages to zero:

$$\tilde{A}_{i,\tau} = \begin{cases} 0, & \text{if } \tau \in \mathcal{T}_j \text{ for } s_j \in \mathcal{S}^{-}_{\text{mask}}, \\ \hat{A}_{i,\tau}, & \text{otherwise,} \end{cases} \quad (5)$$

where $\mathcal{S}^{-}_{\text{mask}}$ denotes the subset of high-confidence steps selected from a verifier-rejected rollout. The remaining steps retain their original negative advantages, so the zeroed steps sit strictly above them, protecting reliable intermediate reasoning from undue penalization. At the same time, zero lies at or below the positive advantages retained by any step in a correct rollout in the same group, so this shielding never over-rewards steps from failed rollouts relative to legitimately correct reasoning. The result is a denser but well-calibrated learning signal, which is critical for maintaining accuracy during aggressive reasoning compression.

Together, these two components form a reward-conditioned, step-level advantage selection mechanism that reuses a single zero-valued operation. Correct rollouts are prevented from reinforcing unreliable reasoning steps, while verifier-rejected rollouts are decomposed to shield useful intermediate reasoning from undue penalization. As shown in our experiments, this symmetric treatment stabilizes training and improves accuracy–efficiency trade-offs under short-context post-training.

3 Experiments

3.1 Experimental Setup

Training Details We use DeepScaleR-Preview-Dataset (Luo et al., 2025b) for training, which includes approximately 40K mathematics problem–answer pairs collected from AIME (1984–2023), AMC (prior to 2023), Omni-MATH (Gao et al., 2025), and Still (Min et al., 2024). Our base model is DeepScaleR-1.5B-Preview (Luo et al., 2025b), a 1.5B-parameter model originally RL fine-tuned from DeepSeek-R1-Distill-Qwen-1.5B (Guo et al., 2025) on the same dataset using three training stages with increasing context lengths (8K $\rightarrow$ 16K $\rightarrow$ 24K). For GRPO training, we adopt the same hyperparameters as in DeepScaleR-1.5B-Preview. In particular, we use a learning rate of 1e-6 and a training batch size of 128. For our method, we set the hyperparameter ratio r to 0.3. For a fair comparison, we set the maximum context length as 4K tokens during training. All experiments are conducted for 500

training steps using the VeRL framework (Sheng et al., 2025), with 8 rollouts sampled per prompt. We use AIME24 as validation set to select the checkpoint with the highest Accuracy–Efficiency Score (AES). All experiments are conducted using 8 AMD MI250 with 64 GB memory.

Evaluation Details We evaluate on five different math reasoning datasets: AIME2024, AIME2025, AMC (MAA, 2024), MATH (Hendrycks et al., 2021b), OlympiadBench (He et al., 2024). In addition, we include GPQA-Diamond (Rein et al., 2024), LSAT (Zhong et al., 2024), MMLU (Hendrycks et al., 2021a), three general reasoning benchmarks to test the ability to generalize to out-of-domain data. For each problem, we sample $k = 16$ responses with a temperature of 0.6, a top-p of 0.95, and a maximum generation length of 8K tokens. Following standard practices (Guo et al., 2025), we report Pass@1 $= \frac{1}{k}\sum_{i=1}^{k} p_i$, where $p_i \in \{0, 1\}$ denotes the correctness of the i-th response[3]. We also report the average number of output tokens to measure reasoning length. Finally, we compute the Accuracy-Efficiency Score (AES, see details in Appendix A; Luo et al., 2025a) to evaluate the trade-off between improving accuracy and reducing computational cost.

Baselines We evaluate our method against the following baselines, all of which are initialized from DeepScaleR-1.5B-Preview post-trained using a maximum context length of 4K tokens:

- **GRPO-4K**: trained with standard GRPO post-training under a 4K training context window, without any additional RL techniques.
- **L1-Max** (Aggarwal and Welleck, 2025): trained with Length Controlled Policy Optimization (LCPO) to generate outputs of varying lengths while respecting a specified maximum length constraint.
- **ThinkPrune-4k** (Hou et al., 2025): An RL–based pruning method that iteratively enforces progressively stricter token limits to remove redundant reasoning steps.
- **LAPO-I** (Wu et al., 2025a): A two-stage RL approach that adaptively controls reasoning length by modeling and leveraging successful solution length distributions.

[3]This is equivalent to generating k outputs with different random seeds (one output per seed) and averaging their correctness, which reduces evaluation variance compared to relying on a single sampled output.

3.2 Experimental Results

Table 1 presents the performance of SAS and baselines across five mathematical reasoning datasets. Our method achieves the best accuracy–efficiency trade-off, outperforming all baselines. Notably, short-context post-training alone already induces substantial reasoning compression: compared to the base DeepScaleR-1.5B-Preview, GRPO-4K substantially reduces the average output length while slightly improving average Pass@1 accuracy, confirming that context length itself provides a strong compression signal. However, this compression comes with limited stability (as shown in Figure 2). Among length-aware baselines, L1-Max achieves the most aggressive compression but at a significant cost to task accuracy. Conversely, ThinkPrune and LAPO prioritize accuracy preservation but achieve only moderate length reduction, resulting in lower overall efficiency gains. In contrast, SAS consistently outperforms all the baselines, improving average Pass@1 accuracy by more than 2 points over the base model while reducing output length by approximately 1,700 tokens. This dual improvement results in the highest AES of 0.46. Notably, our method maintains or exceeds the accuracy of GRPO-4K across all math benchmarks while generating shorter traces, demonstrating a better accuracy–efficiency trade-off.

We extend our evaluation to three general reasoning benchmarks in Table 2. Although these tasks typically require more concise reasoning traces, we observe similar trends. Pure short-context post-training (GRPO-4K) again leads to reduced output length but degrades accuracy relative to the base model, indicating over-compression. In contrast, our method preserves or improves accuracy across all general reasoning datasets with substantially shorter outputs, achieving the best overall efficiency score among stable methods. These results demonstrate that the benefits of our approach are not limited to mathematical reasoning, but extend to broader reasoning settings where controlling training stability is critical.

4 Analysis

4.1 Ablation Study

We evaluate the impact of our design choices on five math reasoning datasets, as shown in Table 3.

	AIME24		AIME25		MATH		AMC		OlympiadBench		Average		
	Pass@1	#Tok	Pass@1	#Tok	Pass@1	#Tok	Pass@1	#Tok	Pass@1	#Tok	Pass@1	#Tok	AES
DeepScaleR	33.75	6755	**26.88**	6444	86.36	2809	67.62	4761	47.23	4824	52.37	5118	0.00
GRPO-4K	38.75	5282	25.42	4812	85.09	1976	71.69	3395	47.09	3411	53.61	3775	0.33
SAS (ours)	**39.79**	4876	26.67	4295	**86.58**	1768	71.46	3090	**48.19**	3008	**54.54**	3407	**0.46**
- Only Correct	36.46	4686	**26.88**	4294	86.18	1801	71.99	3082	47.99	2965	53.90	3366	0.43
- Random Steps	37.29	4892	24.79	4461	86.10	1780	71.16	3086	47.51	3055	53.37	3455	0.38
- Token Level	36.25	4771	25.00	4295	86.11	1912	**72.36**	3148	47.56	3071	53.46	3439	0.39

Table 3: Experimental results for the ablation study.

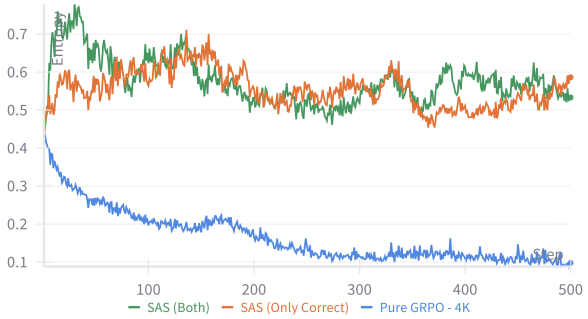


Figure 3: Policy entropy throughout training. SAS maintains higher entropy compared to the rapid entropy collapse observed in pure GRPO-4K, indicating more robust exploration.

Value of Verifier-Failed Rollouts. We first examine the necessity of leveraging verifier-failed rollouts. Restricting SAS to only correct rollouts ("Only Correct") leads to a consistent drop in overall efficiency, reducing the average AES from 0.46 to 0.43 and increasing the average output length. This suggests that shielding high-confidence intermediate steps in verifier-failed rollouts from undue penalization is critical for stabilizing the model during aggressive compression: without this mechanism, truncation-induced verifier failures propagate negative credit to reasoning that is actually correct, destabilizing training. We further analyze the training stability by monitoring the policy entropy, as shown in Figure 3. While the entropy of pure GRPO-4K (blue line) drops rapidly, signaling a collapse in exploration and the adoption of brittle, repetitive reasoning patterns, both SAS (Both) and SAS (Only Correct) maintain a significantly higher and more stable entropy level throughout training. This suggests that by neutralizing noisy advantage signals—rather than propagating them through all steps of a rollout—SAS prevents the model from converging prematurely to suboptimal reasoning patterns, thereby preserving the diversity and robustness of the reasoning process.

Confidence-Aware Step Selection. Replacing our confidence-based selection with random step selection ("Random Steps") results in a clear degradation in efficiency (AES 0.38) and longer reasoning traces. This indicates that the gains of our method do not arise merely from sparsifying the advantage signal, but critically depend on selectively filtering low-confidence reasoning steps.

Step-Level vs. Token-Level Granularity. The token-level variant ("Token Level") underperforms our step-level design, achieving lower average accuracy and AES (0.39 vs. 0.46), while producing longer outputs. This confirms that aggregating tokens into semantically meaningful reasoning steps yields more stable and effective advantage selection than fine-grained token-level selection.

Overall, these ablations demonstrate that all components of our method—leveraging both correct and incorrect rollouts, step-level granularity, and confidence-based selection—jointly contribute to the strongest accuracy–efficiency trade-off under short-context post-training.

4.2 Effect of the Selection Ratio

We study the effect of the selection ratio r in Step-level Advantage Selection (SAS), which controls the fraction of reasoning steps selected for advantage assignment during training. We vary r from 0.1 to 0.9 while keeping all other training settings fixed, and compare against both the base DeepScaleR-1.5B-Preview model and pure GRPO-4K. As shown in Table 4, SAS consistently outperforms the base DeepScaleR-1.5B-Preview model across all tested ratios, yielding higher accuracy and substantially shorter reasoning traces. The best overall performance is achieved at $r = 0.3$, which attains the highest Pass@1 accuracy (54.54), the strongest accuracy–efficiency trade-off (AES = 0.46), and a large reduction in output length compared to the base model. However, performance differences across ratios are relatively small:

Selection Ratio (r)	Pass@1	#Tok	AES
DeepScaleR	52.73	5118	0.00
GRPO-4K	53.61	3775	0.33
0.1	53.52	3259	0.43
0.3	**54.54**	3407	**0.46**
0.5	53.39	3407	0.39
0.7	53.22	3412	0.38
0.9	53.06	3482	0.36

Table 4: Average performance of SAS under different selection ratio r on five math reasoning datasets.

even extreme values $r = 0.9$ remain competitive, with AES scores above 0.36 and stable accuracy. One possible explanation is that, even when most steps are retained, SAS still induces a highly non-uniform learning signal in which only a small subset of informative reasoning steps meaningfully influences policy updates. This aligns with recent findings that reinforcement learning for LLM reasoning is primarily driven by a minority of high-entropy or decision-critical steps, while most tokens in long reasoning traces are effectively redundant from an optimization perspective (Wang et al., 2025b). Overall, these results show that SAS is practically robust and easy to deploy. While moderate ratios offer the best accuracy–efficiency trade-off, the method performs reliably across a wide range of settings. This reinforces the central conclusion that *which* reasoning steps are selected is more important than *how many*, highlighting step-level advantage selection as the key driver of stable and effective reasoning compression.

4.3 Validating Step Confidence

A natural question is whether the policy's own token log-probabilities faithfully reflect step quality. To validate this, we sample 16 responses per question on MATH500 from DeepScaleR-1.5B-Preview (8,000 responses total), segment each response into reasoning steps using double newlines (\n\n), and score every step independently with (i) the mean of token-level log probabilities within that step, and (ii) an external Process Reward Model, Qwen2.5-Math-PRM-7B (Zhang et al., 2025). We then measure the ranking correlation between these two scores using nDCG@k. The resulting correlation is 0.9022, indicating strong agreement between our confidence-based ranking and the PRM-based ranking. This suggests that token-level log probabilities provide a reliable signal for identifying high-quality reasoning steps. Moreover, introducing a PRM can lead to reward hacking (Gao et al., 2023) and significantly increase computational overhead during large-scale RL training (Guo et al., 2025). In contrast, SAS does not rely on an auxiliary reward model and leverages intrinsic confidence signals already available from the policy, avoiding additional training complexity and cost. Taken together, our empirical correlation analysis and the practical considerations support the use of token-level log probabilities as an effective and scalable proxy for step-level confidence in SAS.

4.4 Computational Overhead of SAS

We measured the computational overhead of SAS under the same training configuration (training batch size = 128) as standard GRPO. In this setting, the average training time per step is 279.08 seconds for standard GRPO and 327.15 seconds for SAS. This corresponds to an approximately 17% increase in per-step wall-clock time. The additional cost mainly comes from step segmentation and step-level advantage computation, which introduce lightweight post-processing over generated tokens but do not require additional forward passes, auxiliary models, or extra rollouts. Importantly, SAS does not modify the model architecture or the sampling procedure, and the memory footprint remains unchanged. Given that SAS consistently improves the accuracy–efficiency trade-off while introducing only moderate computational overhead, we believe this cost is practical and acceptable.

5 Related Work

Test-time Scaling in LLMs Test-time scaling has been shown to improve the performance of large language models on various complex reasoning tasks, such as mathematical problem-solving and code generation (Wei et al., 2022; Kojima et al., 2022; Wu et al., 2025b). Such performance gains can often be achieved by allocating more inference-time computation, either through generating longer chain-of-thought reasoning traces (Madaan et al., 2023) or by exploring a larger number of reasoning paths (Wang et al., 2023; Yao et al., 2023; Wang et al., 2024). More recently, reinforcement learning has been used to directly induce extended reasoning behaviors, producing models that generate substantially longer and more elaborate reasoning traces at inference time, such as OpenAI-o1 (Jaech et al., 2024) and DeepSeek-R1 (Guo et al., 2025). While these approaches have demonstrated strong

gains on challenging benchmarks, they often rely on generating significantly longer reasoning traces, which can be several times longer than those produced by short chain-of-thought models, leading to increased inference cost and latency. Subsequent analyses have found that such extended reasoning frequently contains redundant or unnecessary steps, including excessive verification or repetition, even on relatively simple problems, resulting in an "overthinking" phenomenon that degrades efficiency without proportional accuracy gains (Chen et al., 2025; Ghosal et al., 2025; Gema et al., 2025). As a result, despite their effectiveness, existing test-time scaling methods generally lack precise and dynamic control over the length of generated reasoning, motivating growing interest in approaches that retain the benefits of increased inference-time computation while enabling more efficient and controllable reasoning.

Efficient Reasoning for LLMs Motivated by the high computational cost and inefficiency introduced by test-time scaling, a growing body of work has focused on *efficient reasoning*, which aims to reduce reasoning overhead while preserving task performance. Rather than allocating additional computation at inference time, these approaches seek to shorten reasoning traces through training-time optimization, reasoning structure compression, or inference control (Sui et al., 2025). A prominent class of methods focuses on model-based efficient reasoning, particularly through leveraging traditional RL optimization techniques combined with explicit length-aware reward to control the length of CoT reasoning (Team et al., 2025; Arora and Zanette, 2025). Representative approaches include L1 (Aggarwal and Welleck, 2025), which introduces explicit length constraints during policy optimization, length-adaptive policy optimization methods such as LAPO (Wu et al., 2025a), and pruning-based strategies such as ThinkPrune (Hou et al., 2025), which iteratively remove redundant reasoning steps while maintaining correctness. Despite their effectiveness, most existing methods entangle explicit length-aware objectives with short-context post-training, making it difficult to isolate the role of training context and learning signal design. In contrast, our work revisits efficient reasoning from the perspective of credit assignment, showing that stable reasoning compression can be achieved without explicit length rewards by selectively assigning advantages at the level of reasoning steps.

Confidence-based and Entropy-based RL. A parallel line of work leverages confidence or entropy signals during RL: Prabhudesai et al. (2025) replaces the verifier-based reward with a negative-entropy reward to directly optimize for high-confidence rollouts, while Wang et al. (2025b) updates only the highest-entropy tokens and reports *longer* resulting responses. SAS neither modifies the reward nor introduces entropy regularization; confidence serves only as a criterion for selecting which steps receive a nonzero advantage. Furthermore, unlike Wang et al. (2025b), SAS *filters* low-confidence steps to mitigate overthinking under short-context post-training, rather than amplifying high-entropy ones.

6 Discussion and Conclusion

We revisit efficient reasoning through the lens of post-training context length and advantage selection. We show that short-context post-training alone is sufficient to induce substantial reasoning compression in long-context reasoning models, but our analysis also reveals a previously overlooked limitation of rollout-level reinforcement learning: truncated or verifier-failed rollouts introduce noisy and misaligned learning signals that undermine training stability. These findings highlight that reasoning efficiency is not only shaped by reward design, but also by the granularity at which credit is assigned within a rollout. To address this, we propose Step-level Advantage Selection (SAS), which refines advantage assignment through a single zero-valued operation with asymmetric effects: suppressing low-confidence steps in correct rollouts and shielding high-confidence intermediate reasoning in verifier-failed rollouts from undue penalization. SAS stabilizes training and achieves a stronger accuracy–efficiency trade-off than existing length-aware methods, demonstrating that stable and effective reasoning compression can be achieved with minimal modifications to standard RL pipelines.

Limitations

Our proposed method Step-level Advantage Selection(SAS) demonstrates strong empirical results on both in-domain and out-of-domain tasks. However, our experiments focus on a single base model, and it remains to be seen how well SAS generalizes

to models with different sizes, pretraining or post-training paradigms. In addition, all experiments are conducted under a fixed short-context post-training setting, and the behavior of SAS across varying training context length has not been systematically studied. Further, while our ablation results suggest that the selective advantage strategy is broadly effective, we leave a deeper theoretical understanding of advantage selection in reasoning language models to future work. We do not foresee any particular risks associated with the application of our method.

Acknowledgement

We thank the anonymous reviewers for their valuable feedback.

References

Pranjal Aggarwal and Sean Welleck. 2025. L1: Controlling how long a reasoning model thinks with reinforcement learning. In *Second Conference on Language Modeling*.

Daman Arora and Andrea Zanette. 2025. Training language models to reason efficiently. In *The Thirty-ninth Annual Conference on Neural Information Processing Systems*.

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. 2025. Do NOT think that much for 2+3=? on the overthinking of long reasoning models. In *Forty-second International Conference on Machine Learning*.

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. 2025. Omni-MATH: A universal olympiad level mathematic benchmark for large language models. In *The Thirteenth International Conference on Learning Representations*.

Leo Gao, John Schulman, and Jacob Hilton. 2023. Scaling laws for reward model overoptimization. In *Proceedings of the 40th International Conference on Machine Learning*, volume 202 of *Proceedings of Machine Learning Research*, pages 10835–10866. PMLR.

Aryo Pradipta Gema, Alexander Hägele, Runjin Chen, Andy Arditi, Jacob Goldman-Wetzler, Kit Fraser-Taliente, Henry Sleight, Linda Petrini, Julian Michael, Beatrice Alex, Pasquale Minervini, Yanda Chen, Joe Benton, and Ethan Perez. 2025. Inverse scaling in test-time compute. *Transactions on Machine Learning Research*. Featured Certification, J2C Certification.

Soumya Suvra Ghosal, Souradip Chakraborty, Avinash Reddy, Yifu Lu, Mengdi Wang, Dinesh Manocha, Furong Huang, Mohammad Ghavamzadeh, and Amrit Singh Bedi. 2025. Does thinking more always help? mirage of test-time scaling in reasoning models. In *The Thirty-ninth Annual Conference on Neural Information Processing Systems*.

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, Jie Liu, Lei Qi, Zhiyuan Liu, and Maosong Sun. 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, Bangkok, Thailand. Association for Computational Linguistics.

Dan Hendrycks, Collin Burns, Steven Basart, Andy Zou, Mantas Mazeika, Dawn Song, and Jacob Steinhardt. 2021a. Measuring massive multitask language understanding. In *International Conference on Learning Representations*.

Dan Hendrycks, Collin Burns, Saurav Kadavath, Akul Arora, Steven Basart, Eric Tang, Dawn Song, and Jacob Steinhardt. 2021b. Measuring mathematical problem solving with the math dataset. In *Proceedings of the Neural Information Processing Systems Track on Datasets and Benchmarks*, volume 1.

Bairu Hou, Yang Zhang, Jiabao Ji, Yujian Liu, Kaizhi Qian, Jacob Andreas, and Shiyu Chang. 2025. Thinkprune: Pruning long chain-of-thought of llms via reinforcement learning. *arXiv preprint arXiv:2504.01296*.

Aaron Jaech, Adam Kalai, Adam Lerer, Adam Richardson, Ahmed El-Kishky, Aiden Low, Alec Helyar, Aleksander Madry, Alex Beutel, Alex Carney, and 1 others. 2024. Openai o1 system card. *arXiv preprint arXiv:2412.16720*.

Takeshi Kojima, Shixiang (Shane) Gu, Machel Reid, Yutaka Matsuo, and Yusuke Iwasawa. 2022. Large language models are zero-shot reasoners. In *Advances in Neural Information Processing Systems*, volume 35, pages 22199–22213. Curran Associates, Inc.

Haotian Luo, Li Shen, Haiying He, Yibo Wang, Shiwei Liu, Wei Li, Naiqiang Tan, Xiaochun Cao, and Dacheng Tao. 2025a. O1-pruner: Length-harmonizing fine-tuning for o1-like reasoning pruning. In *2nd AI for Math Workshop @ ICML 2025*.

Michael Luo, Sijun Tan, Justin Wong, Xiaoxiang Shi, William Y. Tang, Manan Roongta, Colin Cai, Jeffrey Luo, Li Erran Li, Raluca Ada Popa, and Ion Stoica.

2025b. DeepscaleR: Effective RL scaling of reasoning models via iterative context lengthening. Notion Blog.

Mathematical Association of America MAA. 2024. American mathematics competitions (amc). https://maa.org/student-programs/amc/.

Aman Madaan, Niket Tandon, Prakhar Gupta, Skyler Hallinan, Luyu Gao, Sarah Wiegreffe, Uri Alon, Nouha Dziri, Shrimai Prabhumoye, Yiming Yang, Shashank Gupta, Bodhisattwa Prasad Majumder, Katherine Hermann, Sean Welleck, Amir Yazdanbakhsh, and Peter Clark. 2023. Self-refine: Iterative refinement with self-feedback. In *Advances in Neural Information Processing Systems*, volume 36, pages 46534–46594. Curran Associates, Inc.

Yingqian Min, Zhipeng Chen, Jinhao Jiang, Jie Chen, Jia Deng, Yiwen Hu, Yiru Tang, Jiapeng Wang, Xiaoxue Cheng, Huatong Song, and 1 others. 2024. Imitate, explore, and self-improve: A reproduction report on slow-thinking reasoning systems. *arXiv preprint arXiv:2412.09413*.

Mihir Prabhudesai, Lili Chen, Alex Ippoliti, Katerina Fragkiadaki, Hao Liu, and Deepak Pathak. 2025. Maximizing confidence alone improves reasoning. *arXiv preprint arXiv:2505.22660*.

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

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*, EuroSys '25, page 1279–1297, New York, NY, USA. Association for Computing Machinery.

Yang Sui, Yu-Neng Chuang, Guanchu Wang, Jiamu Zhang, Tianyi Zhang, Jiayi Yuan, Hongyi Liu, Andrew Wen, Shaochen Zhong, Na Zou, Hanjie Chen, and Xia Hu. 2025. Stop overthinking: A survey on efficient reasoning for large language models. *Transactions on Machine Learning Research*.

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*.

Han Wang, Archiki Prasad, Elias Stengel-Eskin, and Mohit Bansal. 2024. Soft self-consistency improves language models agents. In *Proceedings of the 62nd Annual Meeting of the Association for Computational Linguistics (Volume 2: Short Papers)*, pages 287–301, Bangkok, Thailand. Association for Computational Linguistics.

Jianing Wang, Jin Jiang, Yang Liu, Mengdi Zhang, and Xunliang Cai. 2025a. Prejudge-before-think: Enhancing large language models at test-time by process prejudge reasoning. In *Findings of the Association for Computational Linguistics: EMNLP 2025*, Suzhou, China. Association for Computational Linguistics.

Shenzhi Wang, Le Yu, Chang Gao, Chujie Zheng, Shixuan Liu, Rui Lu, Kai Dang, Xiong-Hui Chen, Jianxin Yang, Zhenru Zhang, Yuqiong Liu, An Yang, Andrew Zhao, Yang Yue, Shiji Song, Bowen Yu, Gao Huang, and Junyang Lin. 2025b. Beyond the 80/20 rule: High-entropy minority tokens drive effective reinforcement learning for LLM reasoning. In *The Thirty-ninth Annual Conference on Neural Information Processing Systems*.

Xuezhi Wang, Jason Wei, Dale Schuurmans, Quoc V Le, Ed H. Chi, Sharan Narang, Aakanksha Chowdhery, and Denny Zhou. 2023. Self-consistency improves chain of thought reasoning in language models. In *The Eleventh International Conference on Learning Representations*.

Jason Wei, Xuezhi Wang, Dale Schuurmans, Maarten Bosma, brian ichter, Fei Xia, Ed Chi, Quoc V Le, and Denny Zhou. 2022. Chain-of-thought prompting elicits reasoning in large language models. In *Advances in Neural Information Processing Systems*, volume 35, pages 24824–24837.

Xingyu Wu, Yuchen Yan, Shangke Lyu, Linjuan Wu, Yiwen Qiu, Yongliang Shen, Weiming Lu, Jian Shao, Jun Xiao, and Yueting Zhuang. 2025a. Lapo: Internalizing reasoning efficiency via length-adaptive policy optimization. *arXiv preprint arXiv:2507.15758*.

Yangzhen Wu, Zhiqing Sun, Shanda Li, Sean Welleck, and Yiming Yang. 2025b. Inference scaling laws: An empirical analysis of compute-optimal inference for LLM problem-solving. In *The Thirteenth International Conference on Learning Representations*.

Siheng Xiong, Ali Payani, and Faramarz Fekri. 2025. Enhancing long chain-of-thought reasoning through multi-path plan aggregation. *arXiv preprint arXiv:2510.11620*.

Shunyu Yao, Dian Yu, Jeffrey Zhao, Izhak Shafran, Tom Griffiths, Yuan Cao, and Karthik Narasimhan. 2023. Tree of thoughts: Deliberate problem solving with large language models. In *Advances in Neural Information Processing Systems*, volume 36, pages 11809–11822. Curran Associates, Inc.

Zhenru Zhang, Chujie Zheng, Yangzhen Wu, Beichen Zhang, Runji Lin, Bowen Yu, Dayiheng Liu, Jingren Zhou, and Junyang Lin. 2025. The lessons of developing process reward models in mathematical reasoning. In *Findings of the Association for Computational Linguistics: ACL 2025*, Vienna, Austria. Association for Computational Linguistics.

Wanjun Zhong, Ruixiang Cui, Yiduo Guo, Yaobo Liang, Shuai Lu, Yanlin Wang, Amin Saied, Weizhu Chen,

and Nan Duan. 2024. AGIEval: A human-centric benchmark for evaluating foundation models. In *Findings of the Association for Computational Linguistics: NAACL 2024*, pages 2299–2314, Mexico City, Mexico. Association for Computational Linguistics.

A Accuracy-Efficiency Score (AES)

The AES score (Luo et al., 2025a) compares a tuned model against its corresponding base model to measure the trade-off between accuracy and computational efficiency. Let L_{base} and $\mathrm{L}_{\text{model}}$ denote the output lengths, and $\mathrm{Acc}_{\text{base}}$ and $\mathrm{Acc}_{\text{model}}$ denote the accuracies of the base and tuned models, respectively. Defining $\Delta\mathrm{L} = \frac{\mathrm{L}_{\text{base}} - \mathrm{L}_{\text{model}}}{\mathrm{L}_{\text{base}}}$ and $\Delta\mathrm{Acc} = \frac{\mathrm{Acc}_{\text{model}} - \mathrm{Acc}_{\text{base}}}{\mathrm{Acc}_{\text{base}}}$, the AES is calculated as:

$$\mathrm{AES} = \begin{cases} \alpha \cdot \Delta\mathrm{L} + \beta \cdot \Delta\mathrm{Acc}, & \text{if } \Delta\mathrm{Acc} \geq 0 \\ \alpha \cdot \Delta\mathrm{L} - \gamma \cdot |\Delta\mathrm{Acc}|, & \text{if } \Delta\mathrm{Acc} < 0 \end{cases}$$

where $\alpha > 0$, $\beta > 0$, and $\gamma > 0$. Following the original work, we set $\alpha = 1$, $\beta = 3$, and $\gamma = 5$, with $\gamma > \beta$ to emphasize the penalization of accuracy degradation.

B Step Segmentation

The use of double newlines (`\n\n`) for step segmentation is grounded in the training data format of the model family we use, rather than being an arbitrary heuristic. From Qwen2.5-Math-1.5B to DeepSeek-R1-Distill-Qwen-1.5B, the SFT stage uses approximately 800K examples (Guo et al., 2025), in which individual reasoning steps are consistently separated by `\n\n`. Subsequently, from DeepSeek-R1-Distill-Qwen-1.5B to DeepScaleR-1.5B-Preview, the post-training dataset (DeepScaleR-Preview-Dataset, which we also use) has the same format of separating reasoning steps with `\n\n`. Therefore, our segmentation strategy aligns with the formatting pattern embedded throughout both the SFT and RL training stages, rather than introducing a new structural assumption. In addition, prior work (Zhang et al., 2025; Wang et al., 2025a; Xiong et al., 2025) also segments reasoning steps using double newlines (`\n\n`) naturally, motivated by pretraining priors and the inherent structure of reasoning tasks.

Importantly, SAS operates at the granularity of complete reasoning steps, including the trailing double newline delimiter (`\n\n`), and assigns advantages to all tokens within each step collectively. As a result, SAS does not alter the step formatting pattern. Empirically, we observe that the model preserves the original step format during training. Moreover, SAS is not intrinsically tied to the `\n\n` delimiter: if a different model family adopts another structured reasoning format, step boundaries can be defined accordingly without modifying the core algorithm.

C Licenses

Datasets are released under the following licenses:

- DeepScaleR-Preview-Dataset: MIT license
- AIME24: Apache-2.0 license
- AIME25: Apache-2.0 license
- MATH: MIT license
- AMC: Apache-2.0 license
- OlympiadBench: MIT license
- GPQA: MIT license
- LSAT: MIT license
- MMLU: MIT license

The models we use have the following licenses:

- DeepScaleR-1.5B-Preview: MIT license
- L1-Max: MIT license
- LAPO-I: Apache-2.0 license
- ThinkPrune-4k:Apache-2.0 license