# ROAST: REDUCING OVERTHINKING IN REASONING VIA ADVANTAGE RESHAPING AND STATE-AWARE TERMINATION

**Anonymous authors**
Paper under double-blind review

## ABSTRACT

Large reasoning models trained via Reinforcement Learning with Verifiable Rewards (RLVR) exhibit exceptional capabilities in complex tasks, yet they increasingly suffer from overthinking. This phenomenon traps them in redundant cycles and excessive self-denial. Existing methods that apply scalar penalties to reasoning length or steps struggle to balance efficiency and accuracy. Furthermore, the absence of structural metrics and the sparsity of binary rewards in Group Relative Policy Optimization (GRPO) severely hinder effective policy updates. To address this, we propose **ROAST** (**R**educing **O**verthinking via **A**dvantage reshaping and **S**tate-aware **T**ermination). First, we introduce a structural prior that utilizes a suffix automaton to detect repeated substrings and high local repetition density. This mechanism safely terminates the reasoning process, significantly reducing generation length without compromising accuracy. Using the detected repetition patterns, we reshape the GRPO advantage for precise credit assignment: it rewards the shortest on-policy correct trajectories while applying relative down-weighting exclusively to the truncated redundant suffixes. This guarantees a positive net advantage for correct answers, mitigating gradient noise. Experiments demonstrate that ROAST effectively mitigates endless self-denial and path-switching, slashing reasoning lengths by 46-56% while preserving overall performance.

## 1 INTRODUCTION

While Reinforcement Learning with Verifiable Rewards (RLVR) enables Large Reasoning Models (LRMs), such as OpenAI o1 (Jaech et al., 2024) and DeepSeek-R1 (Guo et al., 2025), to achieve high accuracy through extensive Chains-of-Thought (CoT), outcome-only optimization breeds *overthinking*. Manifesting as redundant loops and excessive self-denial, this pathological behavior traps models in hesitation and undermines both inference efficiency and performance.

To address this efficiency bottleneck, prior works have explored inference-time interventions and training-time internalization. Inference-time methods typically rely on internal signals such as entropy, attention, and confidence, which are often difficult to access efficiently in modern serving engines such as vLLM. Conversely, training-time approaches internalize conciseness into the model weights. While some methods utilize Supervised Fine-Tuning (SFT) on compressed CoTs (Qiao et al., 2025), Reinforcement Learning, particularly Group Relative Policy Optimization (GRPO) (Shao et al., 2024) with length rewards (Team et al., 2025), offers an on-policy paradigm that uses extra reward signals to encourage concise answers. Unfortunately, RL approaches for efficient reasoning rely on length rewards, which are difficult to balance against accuracy objectives. As model accuracy improves, assigning length rewards creates a structural vulnerability in GRPO's normalization: slightly longer correct samples receive rewards below the group mean, yielding *negative* advantages. This inadvertently punishes valid reasoning, causing gradient instability and reward collapse. Although recent studies dynamically adjust reward strengths (Hu et al., 2026), these heuristic length estimations struggle to isolate genuinely redundant segments under sparse verifiable rewards. Furthermore, methods relying on external auxiliary models for post-pruning introduce unacceptable computational overhead (Jiang et al., 2025), and signals like token entropy have proven unreliable, as models naturally drift into low-entropy and low-confidence states during extended reasoning (Feng et al., 2025). Some works (Lee et al., 2026) attempt to assign fine-grained rewards by identifying



Reviewers: please read the Reviewer Guidelines (iclr.cc/Conferences/2027/ReviewerGuidelines) and the AI Policy for Reviewers (iclr.cc/Conferences/2027/AIPolicyForReviewers). If you used AI to expand, edit, or polish your review, please provide the input text to the LLM. Better still, consider skipping the LLM and submitting your original text: we, and the authors, are much more interested in your unedited thoughts than in what an LLM has to say. AI-assisted or not, you are putting your name and reputation behind your review: LLM-generated falsehoods, hallucinations or misrepresentations are subject to disciplinary action, which may include desk-rejecting all papers you have authored.

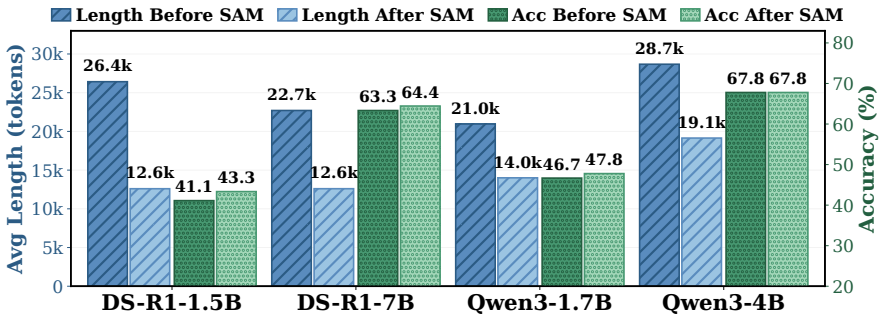



Figure 1: **Motivation.** Zero-shot structural truncation cuts length by 40% while improving accuracy.

harmful overthinking segments that derail correct reasoning, but require strictly formatted intermediate outputs.

We observe that redundant trajectories are often marked by excessive $n$-gram repetition and localized stagnation, where the model repeatedly recombines previously generated substrings. These stagnation windows encapsulate overthinking semantics, such as contradictions, self-denials, and cyclic derivations. Empirically, we find that applying repetition-triggered truncation rules based on repeated substrings can terminate these loops early. As shown in Figure 1, applying our SAM-based truncation rules directly during inference reduces the generation length of truncated samples by 40%, while fully preserving or slightly improving the overall reasoning accuracy.

Building on this, we propose **ROAST**, an RL framework that stabilizes efficiency training without fragile reward-level length penalties. Instead, ROAST intervenes directly at the advantage estimation level. First, it assigns an advantage bonus to the shortest (or fewest-step) correct sample within a group, intrinsically motivating conciseness without forcing longer valid trajectories into negative advantages. Second, during the rollout phase, ROAST uses these repetition-triggered rules to execute non-intrusive truncation. For correct truncated samples, relative down-weighting is localized exclusively to detected redundant suffixes, while assigning positive advantages to all tokens preceding the truncation window. This design guarantees that the overall advantage of correct samples remains strictly positive throughout training, while relative down-weighting only the redundant segments.

## 2 Related Work

### 2.1 Overthinking and Inference-Time Interventions

Overthinking in LLMs produces excessively long reasoning trajectories, exacerbating latency and computational costs. Crucially, prolonged reasoning often degrades performance by introducing harmful behaviors that corrupt correct intermediate states. To mitigate this, inference-time interventions operate at two levels. Input-level methods constrain reasoning budgets via prompt engineering (Han et al., 2025) or step-planning with non-thinking models (Lin et al., 2026). Output-level interventions can be broadly divided into explicit and implicit strategies. Explicit methods modify the decoding distribution to suppress redundant reflection tokens such as "wait" (Zhang et al., 2025; Liu et al., 2025), whereas implicit methods trigger early exit based on signals such as entropy, confidence, or the rank of the `</think>` token (Yang et al., 2025; Wei et al., 2026a); implicit methods extract latent overthinking vectors (Chen et al., 2025) and apply activation steering to push models away from degenerate reasoning states (Li et al., 2026; Huang et al., 2026).

### 2.2 RL for Efficient Reasoning

Inference-time interventions often incur computational overhead and framework incompatibilities. Consequently, recent methods internalize conciseness via Reinforcement Learning (RL) to overcome the excessive verbosity incentivized by traditional correctness-only verifiable rewards. Length-based approaches utilize length penalties but risk training instability and length collapse. ShorterBetter (Yi et al., 2026) penalizes deviations from the shortest correct sample, while LC-R1 (Cheng et al., 2026) explicitly penalizes redundant invalid thinking after deriving correct answers. To stabilize training, dynamic length adjustments are employed (Wang et al., 2026a):

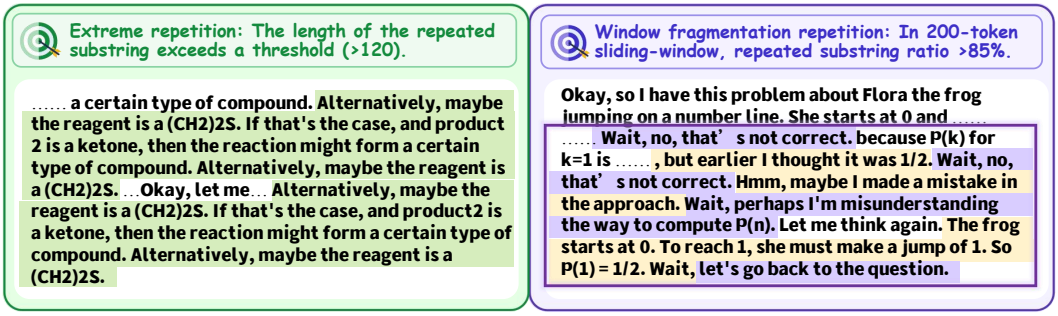



Figure 2: Visualization of the two repetition-triggered truncation rules for identifying overthinking: extreme repetition (left) and window fragmentation repetition (right).

Laser (Liu et al., 2026) introduces step-wise target-length rewards with difficulty awareness, and SmartThinker (Hu et al., 2026) adaptively scales reward coefficients by estimating optimal lengths via Gaussian distributions. Alternatively, early-exit methods evaluate entropy or confidence at reflection tokens to prematurely output answers via injected hints (Dai et al., 2026; Wei et al., 2026b; Chen et al., 2026). Specifically, S-GRPO (Dai et al., 2026) truncates outputs at various CoT positions, DRE (Wei et al., 2026b) locates optimal truncation points on correct trajectories, and JET (Han et al., 2026) combines progressive truncation with quality-controlled length rewards to encourage active termination. Finally, addressing the low entropy often associated with overthinking states, CER (Bin et al., 2025) uses segmented reward-penalty schemes to accelerate convergence to deterministic states, ReSum (Wang et al., 2026b) reduces uncertainty via injected summarization hints, and TRAAC (Singh et al., 2025) penalizes redundant tokens based on low self-attention scores.

# 3 Motivation

## 3.1 GRPO and the Pitfalls of Length Rewards

GRPO eliminates the parameterized value model by estimating the baseline directly from group scores (Shao et al., 2024). Formally, for a given query $q$, GRPO samples a group of $G$ outputs $\{o_1, \ldots, o_G\}$ from the policy $\pi_{\theta_{old}}$. A rule-based verifier evaluates each output to assign a binary correctness reward $r_{acc}(o_i) \in \{0, 1\}$.

To mitigate overthinking, recent methods introduce a length reward $r_{len}(o_i)$. The composite reward $r_i$ is defined as $r_i = r_{acc}(o_i) + r_{len}(o_i)$, where $r_{len}(o_i) = -\lambda \cdot \Delta(|o_i|, L_{opt})$. Here, $\lambda > 0$, and $\Delta$ measures the deviation of the output length $|o_i|$ from the optimal length $L_{opt}$. GRPO calculates the advantage $\hat{A}_i$ by normalizing these rewards within the sampled group: $\hat{A}_i = \frac{r_i - \text{mean}(r)}{\text{std}(r) + \epsilon}$, where $\text{mean}(r)$ and $\text{std}(r)$ are the group reward statistics, and $\epsilon$ ensures numerical stability. The policy $\pi_\theta$ is updated to encourage tokens with positive advantages ($\hat{A}_i > 0$) while suppressing those with negative advantages ($\hat{A}_i < 0$).

While initially encouraging conciseness, this group-relative normalization harbors a critical structural vulnerability. As training progresses and accuracy improves, most sampled outputs become correct ($r_{acc}(o_i) = 1$). Consequently, correctness heavily inflates the group mean $\text{mean}(r)$, leaving the intra-group variance $\text{std}(r)$ driven almost exclusively by the length reward $r_{len}$. Under this dynamic, a correct and logically sound trajectory $o_k$ slightly longer than $L_{opt}$ falls below the inflated $\text{mean}(r)$, yielding a negative advantage ($\hat{A}_k < 0$). The policy gradient therefore suppresses a valid response in the same direction as an incorrect one, conflating reasoning quality with response length.

## 3.2 Identifying Overthinking via Structural Repetition Patterns

To bypass the vulnerabilities of length-penalized rewards, we target overthinking through its structural symptoms: severe textual repetition and localized stagnation. As illustrated in Figure 2, we deploy an online SAM to efficiently execute two structural truncation rules during rollout: (1) **Extreme Repetition**, triggered by any continuous repeated substring exceeding a strict length threshold; and

(2) **Window Fragmentation**, activated when the density of non-trivial repeated substrings (length $> 10$ tokens) in a sliding window exceeds 85%. Preliminary AIME tests show that applying these rules directly at inference time reduces generation length while preserving accuracy. These results suggest that severe structural repetition provides a high-precision proxy for harmful overthinking, motivating its use as a structural prior for fine-grained advantage reshaping.

# 4 Proposed Method: ROAST

Figure 3 illustrates the overall architecture of ROAST, which mitigates pathological overthinking through two complementary mechanisms: on-policy state-aware termination during rollout generation and fine-grained advantage reshaping.

## 4.1 State-aware Termination via SAM-based Repetition Detection

To detect pathological overthinking with minimal overhead, we deploy an online SAM during rollout generation, as shown in Figure 3 (**Left**). Let $y_i = (y_{i,1}, \ldots, y_{i,|y_i|})$ denote the $i$-th generated response, and let $y_{i,\le k} = (y_{i,1}, \ldots, y_{i,k})$ denote its reasoning prefix up to position $k$. The SAM tracks $\ell_{i,k}$, the length of the longest suffix ending at $y_{i,k}$ that has appeared previously in the same sequence:

$$\ell_{i,k} = \max(\{0\} \cup \{d \in \{1, \ldots, k-1\} \mid \exists j \in \{d, \ldots, k-1\},\ y_{i,k-d+1:k} = y_{i,j-d+1:j}\}). \quad (1)$$

Using SAM suffix links, $\ell_{i,k}$ can be updated in amortized $\mathcal{O}(1)$ time. Based on $\ell_{i,k}$, we consider two termination conditions.

**Extreme Repetition.** The condition $\mathcal{C}_{\text{ext}}^{(i,k)}$ is activated when the repeated suffix reaches the threshold $\tau_{\text{ext}}$, which is set to 120:

$$\mathcal{C}_{\text{ext}}^{(i,k)} := \mathbf{1}\left(\ell_{i,k} \ge \tau_{\text{ext}}\right). \quad (2)$$

**Window Fragmentation.** A model may repeatedly alternate among failed reasoning paths without forming an extremely long repeated suffix. We therefore monitor a sliding window of size $W_{\text{win}}$, set to 200. For each position $u$, define

$$z_{i,u} := \mathbf{1}\left(\ell_{i,u} > \tau_{\text{frag}}\right), \quad (3)$$

where $\tau_{\text{frag}} = 10$ identifies non-trivial repeated suffixes. Once a complete window is available, the fragmentation condition is defined as

$$\mathcal{C}_{\text{frag}}^{(i,k)} := \mathbf{1}\left(k \ge W_{\text{win}}\right) \mathbf{1}\left(\frac{1}{W_{\text{win}}} \sum_{u=k-W_{\text{win}}+1}^{k} z_{i,u} \ge \rho\right), \quad (4)$$

where the repetition-density threshold is set to $\rho = 0.85$.

**State-aware Intervention.** If either condition is activated, the reasoning process is considered trapped in a redundant state. We define the SAM-intervened policy as

$$\pi_\theta^{\text{SAM}}\left(a \mid x, y_{i,\le k}\right) = \begin{cases} \mathbf{1}\left(a = \texttt{EoT}\right), & \mathcal{C}_{\text{ext}}^{(i,k)} \vee \mathcal{C}_{\text{frag}}^{(i,k)}, \\ \pi_\theta\left(a \mid x, y_{i,\le k}\right), & \text{otherwise}, \end{cases} \quad (5)$$

where $a$ denotes a candidate token and EoT denotes the end-of-thought token, implemented as the </think> tag. The intervention is applied only during the reasoning phase; after emitting EoT, the model proceeds to generate the final answer normally. This computationally efficient mechanism operates in amortized $\mathcal{O}(1)$ time per token and linear space, terminating redundant reasoning tails while preserving the preceding reasoning prefix for subsequent fine-grained credit assignment.

## 4.2 Advantage Reshaping for Precise Credit Assignment

To train the policy to avoid redundant states without assigning absolute negative advantages to useful reasoning, ROAST reshapes the GRPO advantage at the token level using positive efficiency bonuses and localized suffix penalties. Let $R_i = R(x, y_i)$ denote the outcome reward of response $y_i$, and let

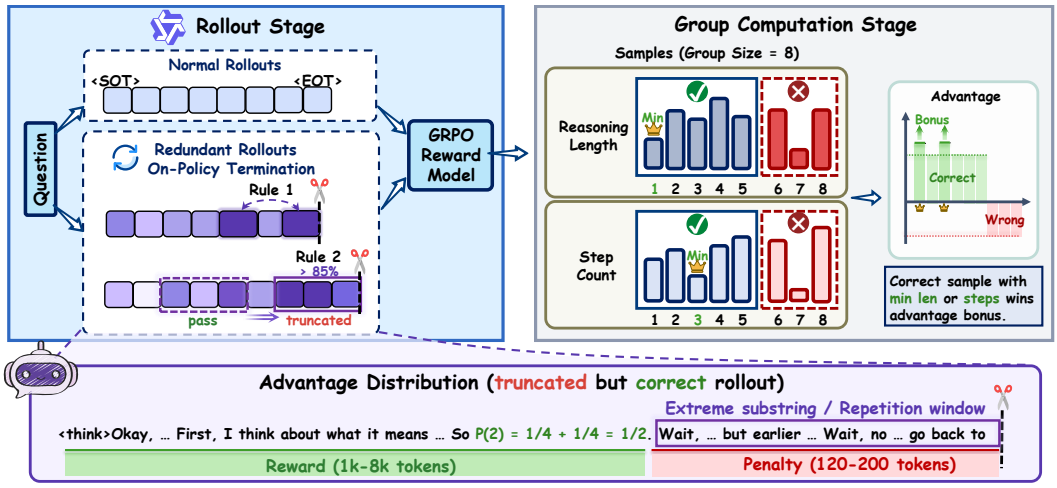



Figure 3: Overview of ROAST. **Left:** During training rollouts, a Suffix Automaton monitors the reasoning sequence and applies state-aware termination when structural repetition patterns are detected. **Right:** After group-level advantage computation, efficiency anchors assign bonuses to the shortest or fewest-step correct responses. **Bottom:** For SAM-terminated correct responses, relative token penalties target only the redundant suffix, protecting the valid prefix.

$c_i := \mathbf{1}(\mathrm{Correct}(x, y_i) = 1)$ denote its correctness indicator. The set of correct responses in group $\mathcal{Y} = \{y_i\}_{i=1}^{G}$ is $\mathcal{Y}_{\mathrm{corr}} = \{y_i \in \mathcal{Y} \mid c_i = 1\}$. The normalized group-relative advantage is defined as

$$\hat{A}_i = \frac{R_i - \mu_{\mathcal{Y}}}{\sigma_{\mathcal{Y}} + \varepsilon_A}, \qquad \mu_{\mathcal{Y}} = \frac{1}{G}\sum_{j=1}^{G} R_j, \tag{6}$$

where $\sigma_{\mathcal{Y}}$ is the standard deviation of group rewards and $\varepsilon_A > 0$ ensures numerical stability.

**Efficiency Anchors.** Let $y_i^{\mathrm{think}}$ denote the reasoning portion of $y_i$, and define its reasoning length and reasoning-step count as

$$L_i^{\mathrm{think}} = \left|y_i^{\mathrm{think}}\right|, \qquad S_i = \mathrm{Steps}\left(y_i^{\mathrm{think}}\right), \tag{7}$$

where $\mathrm{Steps}(\cdot)$ counts reasoning segments partitioned by 14 comman separator phrases, like "wait", adapted from (Qiao et al., 2025) (see the Appendix B for details). When $\mathcal{Y}_{\mathrm{corr}} \neq \varnothing$, the two efficiency anchors are

$$L_{\min} = \min_{y_j \in \mathcal{Y}_{\mathrm{corr}}} L_j^{\mathrm{think}}, \qquad S_{\min} = \min_{y_j \in \mathcal{Y}_{\mathrm{corr}}} S_j. \tag{8}$$

For response $y_i$, the corresponding bonuses are

$$B_{\mathrm{len}}^{(i)} = \alpha c_i \mathbf{1}\left(L_i^{\mathrm{think}} = L_{\min}\right), \qquad B_{\mathrm{step}}^{(i)} = \alpha c_i \mathbf{1}\left(S_i = S_{\min}\right). \tag{9}$$

where $\alpha > 0$ is a shared hyperparameter controlling the magnitude of both bonuses and penalties. The two bonuses are additive and are applied uniformly to the generated tokens:

$$B_{i,t} = B_{\mathrm{len}}^{(i)} + B_{\mathrm{step}}^{(i)}, \qquad t \in \{1, \ldots, |y_i|\}. \tag{10}$$

When $\mathcal{Y}_{\mathrm{corr}} = \varnothing$, we set $B_{i,t} = 0$ for all responses and tokens.

**Targeted Suffix Penalty.** Let $h_i := \mathbf{1}(\text{the } i\text{-th rollout triggers SAM termination})$ indicate whether response $y_i$ is terminated by SAM, and let $k_i^{\mathrm{trig}}$ denote the final reasoning-token position before the forced EoT token. We define the redundant-window length as

$$W_i = \max\left\{\mathcal{C}_{\mathrm{ext}}^{(i,k_i^{\mathrm{trig}})}\ell_{i,k_i^{\mathrm{trig}}},\ \mathcal{C}_{\mathrm{frag}}^{(i,k_i^{\mathrm{trig}})}W_{\mathrm{win}}\right\}. \tag{11}$$

The beginning of the penalized suffix is

$$s_i = \max\left(1, k_i^{\mathrm{trig}} - W_i + 1\right). \tag{12}$$

The token-level suffix penalty is then

$$P_{i,t} = -\alpha c_i h_i \mathbf{1}\left(|\mathcal{Y}_{\text{corr}}| < G\right) \mathbf{1}\left(s_i \le t \le k_i^{\text{trig}}\right). \tag{13}$$

By explicitly disabling the penalty when all responses in the group are correct ($|\mathcal{Y}_{\text{corr}}| = G$), relative down-weighting is applied exclusively to the redundant suffixes of correct SAM-intervened responses in non-uniform groups.

**The Final Reshaped Advantage** is:

$$\tilde{A}_{i,t} = \hat{A}_i + B_{i,t} + P_{i,t}. \tag{14}$$

Because $\hat{A}_i$ is normalized solely on binary correctness rewards prior to reshaping, any correct penalized response under $G = 8$ strictly receives a base advantage $\hat{A}_i \ge \sqrt{1/7} \approx 0.37$. Under our setting, the minimum base advantage of a correct response in a non-uniform group remains above $\alpha = 0.1$. Therefore, even tokens receiving the suffix penalty retain positive reshaped advantages. This mathematically guarantees a positive overall sequence-level advantage, safely localizing relative down-weighting without eroding fundamental problem-solving capabilities.

**Overall Training Objective.** Given a prompt $x \sim \mathcal{D}$, we sample a group of $G$ responses from the old SAM-intervened policy, $y_{1:G} \sim \pi^{\text{SAM}}_{\theta_{\text{old}}}(\cdot \mid x)$. For each generated token, the token-level importance ratio is $r_{i,t}(\theta) = \frac{\pi^{\text{SAM}}_{\theta}(y_{i,t}|x, y_{i,<t})}{\pi^{\text{SAM}}_{\theta_{\text{old}}}(y_{i,t}|x, y_{i,<t})}$. The asymmetrically clipped ratio is $\bar{r}_{i,t}(\theta) = \text{clip}\left(r_{i,t}(\theta), 1-\epsilon_{\text{low}}, 1+\epsilon_{\text{high}}\right)$,, where $\epsilon_{\text{low}}$ and $\epsilon_{\text{high}}$ are the lower and upper clipping bounds. Let $N = \sum_{i=1}^{G} |y_i|$ denote the total number of generated tokens in the group. The ROAST clipped policy-gradient loss is:

$$\mathcal{L}_{\text{ROAST}}(\theta) = -\mathbb{E}_{x\sim\mathcal{D},\, y_i \sim \pi^{\text{SAM}}_{\theta_{\text{old}}}(\cdot|x)} \left[ \frac{1}{N} \sum_{i=1}^{G} \sum_{t=1}^{|y_i|} \min\Big( r_{i,t}(\theta)\tilde{A}_{i,t}, \bar{r}_{i,t}(\theta)\tilde{A}_{i,t} \Big) \right]. \tag{15}$$

In practice, following the standard GRPO implementation, we additionally apply a low-variance KL regularizer against the reference policy with coefficient $\beta = 0.001$. Thus, the actual optimization loss is $\mathcal{L}_{\text{ROAST}} + \beta\mathcal{L}_{\text{KL}}$. Unlike sample-level GRPO, the normalization over $N$ assigns equal unit weight to each generated response token rather than equal aggregate weight to each response. Consequently, the reshaped advantage provides fine-grained supervision: efficiency bonuses reinforce concise correct trajectories, while relative down-weighting is restricted to redundant suffixes.

# 5 Experiments

## 5.1 Experimental Setup

**Implementation Details.** We implement ROAST and all RL baselines using the open-source `verl` framework. Policies are trained on the DAPO-Math-17k dataset for exactly one epoch. The maximum generation length during the training rollout is restricted to 8k tokens. We use $\alpha = 0.1$ to scale the bonus reward and $\beta = 0.001$ for KL regulation. We set the training batch size to 32 and the group size to 8. Comprehensive experimental configurations are detailed in the Appendix A.

**Models & Benchmarks.** We evaluate our method across different model scales using the **Qwen3** (1.7B, 4B) and **DeepSeek-R1-Distill-Qwen** (1.5B, 7B) series. We report zero-shot pass rates and average generation lengths on AIME25, AIME26, AMC23, and MINERVA MATH. All evaluations are performed directly on post-trained weights without inference-time interventions (SAM is disabled during evaluation), utilizing vLLM 0.10.2 with a temperature of 0.6, a top-$p$ of 0.95, and a 16k maximum token limit to accommodate extended reasoning chains.

**Baselines.** We compare ROAST against four categories of baselines: (1) **Standard Baselines:** the base model strictly truncated at 8k tokens (**Truncated-8k**) and standard GRPO optimized under an 8k length budget (**GRPO-8k**); (2) **Length-Reward Methods:** techniques applying scalar penalties, including **ShorterBetter** (Yi et al., 2026), **LC-R1** (Cheng et al., 2026), **LASER** (Liu et al., 2026), and **SmartThinker** (Hu et al., 2026); (3) **Truncation Methods: JET** (Han et al., 2026), which combines progressive truncation with quality-controlled rewards; (4) **State-based Methods:** interventions based on entropy or attention, including **CER** (Bin et al., 2025) and **TRAAC** (Singh et al., 2025).

Table 1: Main results on mathematical reasoning benchmarks. We report zero-shot accuracy (%) and average generation length (#tokens). All reported results are averaged over 4 independent runs. Best and second-best in the Average column are bold and underlined. The base model is separated and excluded from ranking. † indicates that the corresponding method lacks publicly available pre-trained weights; we reproduced it following the data and settings described in the original paper.

| Model | Method | AIME 2025 | | AIME 2026 | | AMC 2023 | | MINERVA | | Average | |
|---|---|---|---|---|---|---|---|---|---|---|---|
| | | Acc | Len | Acc | Len | Acc | Len | Acc | Len | Acc | Len |
| **Qwen3-1.7B** | Baseline | 33.33 | 12977 | 31.67 | 13006 | 76.88 | 7988 | 49.08 | 5934 | 47.74 | 9976 |
| | Truncated-8k | 22.50 | 7448 | 22.50 | 7779 | 56.88 | 5530 | 46.23 | 4763 | 37.03 | 6380 |
| | GRPO-8k | 28.33 | 10336 | 31.67 | 9276 | 71.88 | 4901 | 46.41 | 2911 | 44.57 | 6856 |
| | ShorterBetter† | 27.50 | 9074 | 30.00 | 8346 | 77.50 | 4256 | 45.12 | 2366 | 45.03 | <u>6011</u> |
| | SmartThinker† | 29.17 | 9748 | 32.50 | 9473 | 69.38 | 4917 | 45.86 | 3093 | 44.23 | 6808 |
| | JET† | 35.83 | 10581 | 28.33 | 10632 | 73.75 | 6075 | 48.81 | 4219 | <u>46.68</u> | 7877 |
| | ROAST (Ours) | 39.17 | 7245 | 30.83 | 7407 | 78.13 | 4050 | 47.98 | 2738 | **49.03** | **5360** |
| **Qwen3-4B** | Baseline | 52.50 | 12638 | 54.17 | 11869 | 93.13 | 6478 | 58.09 | 5578 | 64.47 | 9141 |
| | Truncated-8k | 27.50 | 7608 | 31.67 | 7585 | 76.88 | 5275 | 53.30 | 4457 | 47.34 | 6231 |
| | GRPO-8k | 51.67 | 7120 | 50.83 | 6629 | 91.88 | 3447 | 57.35 | 2522 | <u>62.93</u> | 4930 |
| | ShorterBetter† | 38.33 | 4154 | 36.67 | 4006 | 81.88 | 1858 | 53.31 | 834 | 52.55 | **2713** |
| | CER | 50.00 | 9404 | 44.17 | 8818 | 88.75 | 4087 | 55.42 | 2207 | 59.59 | 6129 |
| | TRAAC | 42.50 | 8050 | 46.67 | 6980 | 88.12 | 3284 | 56.24 | 2568 | 58.38 | 5221 |
| | JET† | 48.33 | 10431 | 53.33 | 8401 | 93.13 | 3551 | 54.96 | 2775 | 62.44 | 6290 |
| | ROAST (Ours) | 53.33 | 5995 | 52.50 | 5895 | 92.50 | 2569 | 55.79 | 1611 | **63.53** | <u>4018</u> |
| **DS-R1-1.5B** | Baseline | 23.33 | 11307 | 18.33 | 11913 | 65.63 | 6879 | 34.56 | 4877 | 35.46 | 8744 |
| | Truncated-8k | 18.33 | 7140 | 12.50 | 7361 | 57.50 | 4821 | 32.81 | 3830 | 30.29 | 5788 |
| | LC-R1 | 22.50 | 6580 | 16.67 | 8296 | 68.13 | 3388 | 35.67 | 2047 | 35.74 | 5078 |
| | ShorterBetter | 19.17 | 3440 | 16.67 | 3743 | 63.13 | 1729 | 34.83 | 887 | 33.45 | **2450** |
| | LASER-DE | 23.33 | 7432 | 18.33 | 7483 | 72.50 | 4083 | 38.05 | 2869 | 38.05 | 5467 |
| | SmartThinker | 23.33 | 7457 | 17.50 | 7781 | 69.38 | 3607 | 35.02 | 2135 | 36.31 | 5245 |
| | JET | 24.17 | 8681 | 25.00 | 8599 | 73.75 | 4478 | 38.97 | 3631 | <u>40.47</u> | 6347 |
| | ROAST (Ours) | 26.67 | 5521 | 23.33 | 6237 | 76.25 | 3172 | 35.75 | 2267 | **40.50** | <u>4299</u> |
| **DS-R1-7B** | Baseline | 35.83 | 10576 | 41.67 | 9525 | 86.87 | 4324 | 51.01 | 4025 | 53.84 | 7113 |
| | Truncated-8k | 29.17 | 6745 | 30.00 | 6773 | 83.75 | 3689 | 49.26 | 3550 | 48.05 | 5189 |
| | LC-R1 | 38.33 | 6880 | 40.83 | 6493 | 86.25 | 2735 | 52.76 | 1925 | 54.54 | 4508 |
| | ShorterBetter | 32.50 | 4492 | 36.67 | 4278 | 88.75 | 1773 | 50.92 | 1125 | 52.21 | **2917** |
| | LASER-DE | 34.17 | 5799 | 47.50 | 5597 | 91.25 | 2355 | 53.30 | 2232 | <u>56.55</u> | 3996 |
| | TRAAC | 38.33 | 7763 | 44.17 | 7430 | 85.63 | 3209 | 52.94 | 2867 | 55.27 | 5317 |
| | JET | 42.50 | 8816 | 43.33 | 8256 | 87.50 | 3690 | 51.10 | 3207 | 56.11 | 5992 |
| | ROAST (Ours) | 39.17 | 5537 | 44.17 | 5140 | 91.25 | 2327 | 52.85 | 1713 | **56.86** | <u>3679</u> |

### 5.2 Main Results

Table 1 presents a comprehensive comparison of our proposed ROAST framework against various baselines across four demanding mathematical reasoning benchmarks. Overall, ROAST achieves a highly competitive trade-off between efficiency and accuracy, with its superiority in out-of-distribution reasoning tasks like GPQA further discussed in the Appendix D.

**The Pitfalls of Aggressive Length Penalties.** Length-reward baselines, particularly ShorterBetter, exhibit a critical vulnerability. By applying aggressive scalar length penalties directly to the reward function, ShorterBetter forces the policy into extreme brevity, yielding the shortest average generation lengths across most models. However, this macroscopic suppression comes at an unacceptable cost: it suffers from the most severe performance degradation among the learned length-reduction baselines. For instance, on the Qwen3-4B model, its average accuracy drops by nearly 12 percentage points compared to the base model. This empirical evidence strongly corroborates our theoretical hypothesis: applying blunt scalar penalties to successful trajectories heavily penalizes

valid, necessary logical deductions, thereby inducing severe gradient noise and eroding the model's fundamental reasoning capacity.

**Efficiency–Accuracy Trade-off** When excluding ShorterBetter, which fundamentally sacrifices reasoning capability for superficial conciseness, **ROAST** achieves both the shortest generation lengths and the highest overall accuracy. Unlike entropy-based methods (CER, TRAAC) that struggle to balance the efficiency-accuracy trade-off, or progressive truncation methods (JET) that still leave a considerable margin of redundancy, ROAST exerts precise, structural interventions. By selectively applying token-level penalties only to the trailing redundant states (as matched by the Suffix Automaton) and rewarding efficiency anchors, ROAST robustly maintains positive net advantages for correct solutions. Consequently, across the Qwen3-4B and DS-R1-7B scales, ROAST secures the highest average accuracies (63.53% and 56.86%, respectively) while reducing reasoning length by 46%-56% compared to the base model and further shortening generations by 18%-22% compared to GRPO-8k.

Table 2: Density of overthinking indicators per 1k tokens on correct and incorrect reasoning traces.

| Method | Correct | Incorrect |
|---|---|---|
| Qwen3-4B Base Model | 1.77 | 1.41 |
| ShorterBetter | 1.05 | 1.29 |
| SmartThinker | 1.44 | 1.18 |
| JET | 1.18 | 1.60 |
| TRAAC | 0.96 | 1.48 |
| GRPO | 0.94 | 1.31 |
| + Step Bonus | 0.85 | 1.34 |
| + Len Bonus | 1.09 | 1.66 |
| + Len Bonus & SAM | 0.70 | **0.96** |
| ROAST (Ours) | **0.62** | 1.02 |

### 5.3 Semantic Analysis of Pathological Overthinking

Prior work associates explicit self-doubt, approach abandonment, and self-contradiction with pathological overthinking. We therefore track a predefined set of lexical markers corresponding to these behaviors and analyze their frequency in generated reasoning traces (Qiao et al., 2025; Lee et al., 2026).

Table 2 reports the frequency of these markers across generated traces. While competitive baselines reduce these behaviors compared to the Base model, their marker frequencies in correct traces remain high (0.96–1.44) in correct traces. Similarly, relying solely on reward shaping upon GRPO remains insufficient; applying only the Length bonus leaves a high marker frequency of 1.09. Crucially, **State-aware Termination** drives this semantic purification. Integrating SAM sharply decreases marker density, with **ROAST** achieving the absolute lowest frequency (0.62) in correct traces. This significant reduction suggests that, despite using no explicit linguistic supervision, the structural patterns detected by SAM effectively suppress unnecessary self-doubt and frequent approach switching during otherwise successful reasoning trajectories, serving as a useful structural proxy for these lexical manifestations of overthinking.

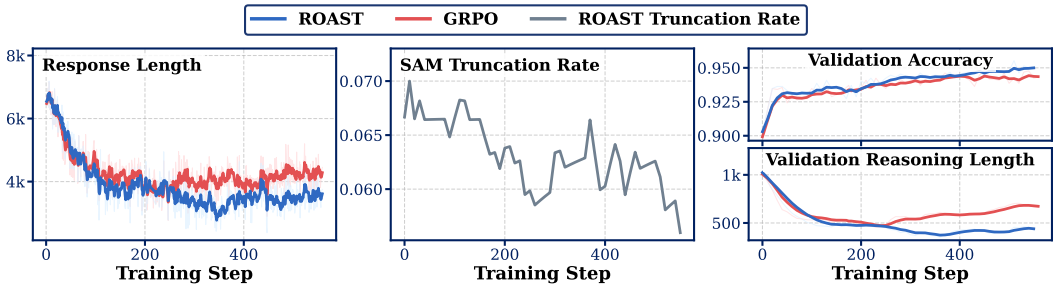



Figure 4: ROAST vs. GRPO training dynamics. Left to right: response length, SAM truncation, and validation accuracy/length. ROAST prevents late-stage length inflation. The declining SAM truncation rate is consistent with the policy progressively internalizing concise reasoning behavior rather than relying solely on external termination.

**Training Dynamics and Intrinsic Efficiency.** Figure 4 shows that, while standard GRPO exhibits late-stage length inflation, ROAST maintains shorter reasoning trajectories alongside high accuracy. The SAM truncation rate also declines over training, indicating that fewer rollouts trigger repetition-

based termination. Together, these trends are consistent with the policy learning to produce concise responses with less frequent SAM intervention.

## 6 Ablation Study

**Synergy of ROAST Components.** To isolate our contributions, we compare ROAST against standard GRPO and four ablated variants in Table 3. Relying solely on length bonuses (Anchor Only) degrades accuracy (61.97%) as the model struggles to distinguish valid deductions from pathological loops. Integrating state-aware termination (Anchor & SAM) physically excises these loops, recovering accuracy (63.25%) and improving the compression rate (55.08%). Conversely, applying SAM without anchors (SAM Only) yields marginal compression (47.02%), demonstrating that positive efficiency anchors are important for achieving stronger intrinsic compression. Ultimately, full ROAST integrates both length and step anchors to prevent reward exploitation, achieving an optimal balance: peak accuracy (63.53%) with robust compression (56.05%). Increasing $\alpha$ to 0.5 further improves compression to 64.23% but reduces average accuracy to 59.48%, indicating that overly aggressive reshaping can harm reasoning quality.

Table 3: Ablation study of ROAST on Qwen3-4B. Results are averaged over AIME25, AIME26, AMC23, and MINERVA.

| Method | Avg Acc | Avg Len | Comp. Rate |
|---|---|---|---|
| GRPO-8k | 62.93 | 4930 | 46.07% |
| SAM Only | 63.41 | 4843 | 47.02% |
| Anchor Only | 61.97 | 4297 | 52.99% |
| Anchor+SAM | 63.25 | 4107 | 55.08% |
| Aggressive $\alpha$ | 59.48 | 3270 | **64.23%** |
| **ROAST** | **63.53** | 4018 | 56.05% |

**Sensitivity of SAM Thresholds.** Table 4 ablates SAM matching granularity ($\tau_{\text{ext}}$, $W_{\text{win}}$). Strict thresholds (60, 100) trigger aggressively (58.1% rate), halting generation early (7.9k tokens) but erroneously truncating 41 correct samples, which severely harms accuracy. Our default setting (120, 200) achieves an optimal balance by excising pathological tails (cutting trapped sample lengths from 24.3k to 14.5k) while reducing false-positive degradations to zero across runs. Additionally, preventing exhaustion occasionally rescues previously failed cases. Further relaxing to (180, 300) yields diminishing returns, as interventions occur too late (16.0k tokens) without accuracy benefits.

Table 4: SAM threshold ablation across 4 models on AIME (360 total samples). "Rescued / Degraded": fixed failures vs. erroneously truncated correct solutions. "Trunc Len" shows avg. length of intervened samples before / after truncation.

| Threshold ($\tau_{\text{ext}}, W_{\text{win}}$) | Trunc (%) | Rescued / Degraded | Trunc Len Before $\rightarrow$ After |
|---|---|---|---|
| (60, 100) | 58.1% | 1 / 41 | 18.9k $\rightarrow$ 7.9k |
| (90, 150) | 32.0% | 8 / 10 | 21.6k $\rightarrow$ 12.4k |
| (120, 200) | 19.5% | 4 / 0 | 24.3k $\rightarrow$ 14.5k |
| (180, 300) | 17.8% | 4 / 1 | 23.9k $\rightarrow$ 16.0k |

## 7 Conclusion

This paper presents ROAST. It is a novel reinforcement learning framework designed to mitigate pathological overthinking in reasoning language models. ROAST employs an online Suffix Automaton to detect and truncate redundant loops in real-time. This structural intervention immediately stops the waste of computational resources. Moreover, ROAST introduces a token-level advantage reshaping mechanism. It restricts the penalty strictly to the redundant suffix. Meanwhile, it rewards the valid reasoning prefix using length and step efficiency anchors. Extensive experiments on mathematical benchmarks confirm the robustness of our approach. ROAST achieves the highest reasoning accuracy. Simultaneously, it significantly reduces the overall generation length. Our framework offers a highly effective approach for balancing efficiency and accuracy in large reasoning models.

## AI USE STATEMENT

Large Language Models were employed solely for the purpose of language editing and grammatical correction to improve the readability and flow of the manuscript. The AI tool was not used for idea generation, data analysis, or writing the core technical content. All suggestions provided by the AI were reviewed and verified by the authors to ensure accuracy and adherence to the original meaning.

## REFERENCES

Yi Bin, Tianyi Jiang, Yujuan Ding, Kainian Zhu, Fei Ma, Jingkuan Song, Yang Yang, and Heng Tao Shen. Explore briefly, then decide: Mitigating llm overthinking via cumulative entropy regulation. *arXiv preprint arXiv:2510.02249*, 2025.

Benteng Chen, Weida Wang, Shufei Zhang, Mingbao Lin, and Min Zhang. Step-grpo: Internalizing dynamic early exit for efficient reasoning. In *Proceedings of the 64th Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, pp. 21710–21724, 2026.

Runjin Chen, Zhenyu Zhang, Junyuan Hong, Souvik Kundu, and Zhangyang Wang. Seal: Steerable reasoning calibration of large language models for free. In *Second Conference on Language Modeling*, 2025.

Zhengxiang Cheng, Dongping Chen, Mingyang Fu, and Tianyi Zhou. Optimizing length compression in large reasoning models. In *Proceedings of the 64th Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, pp. 3232–3250, 2026.

Muzhi Dai, Chenxu Yang, and Qingyi Si. S-grpo: Early exit via reinforcement learning in reasoning models. *Advances in Neural Information Processing Systems*, 38:48178–48204, 2026.

Yunzhen Feng, Julia Kempe, Cheng Zhang, Parag Jain, and Anthony Hartshorn. What characterizes effective reasoning? revisiting length, review, and structure of cot. *arXiv preprint arXiv:2509.19284*, 2025.

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

Jinyi Han, Ying Huang, Ying Liao, Haiquan Zhao, Zishang Jiang, Xinyi Wang, Xikun Lu, Guanghao Zhou, Sihang Jiang, Jiaqing Liang, Weikang Zhou, Zeye Sun, Fei Yu, and Yanghua Xiao. Your models have thought enough: Training large reasoning models to stop overthinking. In *The Fourteenth International Conference on Learning Representations*, 2026. URL https://openreview.net/forum?id=2u5ZRzDyS0.

Tingxu Han, Zhenting Wang, Chunrong Fang, Shiyu Zhao, Shiqing Ma, and Zhenyu Chen. Token-budget-aware llm reasoning. In *Findings of the Association for Computational Linguistics: ACL 2025*, pp. 24842–24855, 2025.

Chenzhi Hu, Qinzhe Hu, Yuhang Xu, Junyi Chen, Ruijie Wang, Shengzhong Liu, Jianxin Li, Fan Wu, and Guihai Chen. Smartthinker: Progressive chain-of-thought length calibration for efficient large language model reasoning. In *Forty-third International Conference on Machine Learning*, 2026.

Yao Huang, Huanran Chen, Shouwei Ruan, Yichi Zhang, Xingxing Wei, and Yinpeng Dong. Mitigating overthinking in large reasoning models via manifold steering. *Advances in Neural Information Processing Systems*, 38:102543–102568, 2026.

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

Shuyang Jiang, Yusheng Liao, Ya Zhang, Yanfeng Wang, and Yu Wang. Overthinking reduction with decoupled rewards and curriculum data scheduling. *arXiv preprint arXiv:2509.25827*, 2025.

Chia-Hsuan Lee, Sihui Dai, Mingyang Zhou, Isha Slavin, Shi-Xiong Zhang, Sambit Sahu, and William Campbell. Know when to stop: Segment-level credit assignment for reducing overthinking. *arXiv preprint arXiv:2607.00482*, 2026.

Yulin Li, Tengyao Tu, Li Ding, Junjie Wang, Hui-Ling Zhen, Yixin Chen, Yong Li, and Zhuotao Tian. Efficient reasoning with balanced thinking. In *The Fourteenth International Conference on Learning Representations*, 2026.

Junhong Lin, Xinyue Zeng, Jie Zhu, Song Wang, Julian Shun, Jun Wu, and Dawei Zhou. Plan and budget: Effective and efficient test-time scaling on reasoning large language models. In *The Fourteenth International Conference on Learning Representations*, 2026.

Wei Liu, Ruochen Zhou, Yiyun Deng, Yuzhen Huang, Junteng Liu, Yuntian Deng, Yizhe Zhang, and Junxian He. Learn to reason efficiently with adaptive length-based reward shaping. In *The Fourteenth International Conference on Learning Representations*, 2026. URL https://openreview.net/forum?id=hj9eKpqxQl.

Xiang Liu, Xuming Hu, Xiaowen Chu, and Eunsol Choi. Diffadapt: Difficulty-adaptive reasoning for token-efficient llm inference. *arXiv preprint arXiv:2510.19669*, 2025.

Ziqing Qiao, Yongheng Deng, Jiali Zeng, Dong Wang, Lai Wei, Guanbo Wang, Fandong Meng, Jie Zhou, Ju Ren, and Yaoxue Zhang. Concise: Confidence-guided compression in step-by-step efficient reasoning. In *Proceedings of the 2025 Conference on Empirical Methods in Natural Language Processing*, pp. 8021–8040, 2025.

Zhihong Shao, Peiyi Wang, Qihao Zhu, Runxin Xu, Junxiao Song, Xiao Bi, Haowei Zhang, Mingchuan Zhang, YK Li, Yang Wu, et al. Deepseekmath: Pushing the limits of mathematical reasoning in open language models. *arXiv preprint arXiv:2402.03300*, 2024.

Joykirat Singh, Justin Chih-Yao Chen, Archiki Prasad, Elias Stengel-Eskin, Akshay Nambi, and Mohit Bansal. Think right: Learning to mitigate under-over thinking via adaptive, attentive compression. *arXiv preprint arXiv:2510.01581*, 2025.

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.

Chen Wang, Hexuan Deng, Yining Zhang, Yuchen Zhang, Jionghao Bai, Zhaochun Li, Ge Lan, and Yue Wang. Implicit compression regularization: Concise reasoning via internal shorter distributions in rl post-training. *arXiv preprint arXiv:2605.07316*, 2026a.

Xucong Wang, Ziyu Ma, Yong Wang, Shidong Yang, Hailang Huang, Renda Li, Pengkun Wang, and Xiangxiang Chu. Resum: Synergizing llm reasoning and summarization with reinforcement learning. *arXiv preprint arXiv:2606.13316*, 2026b.

Zihao Wei, Liang Pang, Jiahao Liu, Wenjie Shi, Jingcheng Deng, Shicheng Xu, Zenghao Duan, Jingang Wang, Fei Sun, Huawei Shen, et al. The evolution of thought: Tracking llm overthinking via reasoning dynamics analysis. In *Proceedings of the 64th Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, pp. 26905–26920, 2026a.

Zihao Wei, Wenjie Shi, Liang Pang, Jingcheng Deng, Shicheng Xu, Shasha Guo, Zenghao Duan, Jiahao Liu, Jingang Wang, Huawei Shen, et al. Dynamic rollout editing for reducing overthinking in rl-trained reasoning models. *arXiv preprint arXiv:2606.17890*, 2026b.

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

Jingyang Yi, Jiazheng Wang, and Sida Li. Shorterbetter: Guiding reasoning models to find optimal inference length for efficient reasoning. *Advances in Neural Information Processing Systems*, 38: 39011–39043, 2026.

Junyu Zhang, Runpei Dong, Han Wang, Xuying Ning, Haoran Geng, Peihao Li, Xialin He, Yutong Bai, Jitendra Malik, Saurabh Gupta, et al. Alphaone: Reasoning models thinking slow and fast at test time. In *Proceedings of the 2025 Conference on Empirical Methods in Natural Language Processing*, pp. 11340–11365, 2025.

# A Training Details

To ensure reproducibility, we detail the complete hardware architecture, software dependencies, dataset configurations, and reinforcement learning hyperparameters applied across our experiments.

## A.1 Hardware, Software, and Dataset Configuration

All reinforcement learning models are trained using the open-source `verl` framework on NVIDIA A800 (80GB) GPUs, utilizing vLLM for high-throughput rollout generation. Specifically, for the 1.5B, 1.7B, and 4B models, we allocate 2 NVIDIA A800 (80GB) GPUs per experiment node. For the 7B models, we scale the hardware allocation to 4 NVIDIA A800 (80GB) GPUs to accommodate larger optimizer states and activation memory.

Our primary software stack is built on Python 3.12.0 and CUDA 13.0, utilizing VeRL 0.8.0.dev0, vLLM v0.10.2, PyTorch 2.8.0, Transformers 4.56.2, and FlashAttention 2.8.3. Table 5 summarizes the specific software dependencies and hardware configurations.

For dataset partitioning, we conduct post-training on the **DAPO-MATH-17K** dataset for exactly one training epoch. During training, validation evaluations are performed on **GSM8K** every 10 steps purely as an indicative metric to monitor reasoning accuracy and generation length dynamics; it is not used for checkpoint selection or hyperparameter tuning. All hyperparameters and the final epoch checkpoint are fixed prior to evaluation. Overlong prompt sequences exceeding 2048 tokens are filtered prior to training.

Table 5: Software dependencies and hardware configurations.

| **Component** | **Specification / Version** |
|---|---|
| *Software Environment* | |
| Python | 3.12.0 |
| CUDA | 13.0 |
| VeRL | 0.8.0.dev0 |
| vLLM | 0.10.2 |
| PyTorch | 2.8.0 |
| Transformers | 4.56.2 |
| FlashAttention | 2.8.3 |
| *Hardware Allocation* | |
| 1.5B / 1.7B / 4B Models | $2 \times$ NVIDIA A800 (80GB) |
| 7B Models | $4 \times$ NVIDIA A800 (80GB) |

## A.2 Optimization and Algorithm Settings

We train all policy models using Group Relative Policy Optimization (GRPO) augmented with our proposed ROAST advantage reshaping. The policy is optimized using AdamW with a default random seed of 42. Following standard GRPO configurations in `verl`, we perform 1 optimization update epoch per rollout generation step.

The importance ratio $r_{i,t}(\theta)$ is clipped symmetrically using clipping bounds $\epsilon_{\text{low}} = 0.2$ and $\epsilon_{\text{high}} = 0.2$, restricting the ratio to the interval $[0.8, 1.2]$. To regulate divergence from the reference model, we apply a low-variance KL divergence penalty with a coefficient of $\beta = 0.001$. The advantage reshaping coefficient controlling the magnitude of efficiency bonuses and suffix penalties is set to $\alpha = 0.1$.

### A.3 Sampling and Context Lengths

During the training rollout phase, we sample a group of $G = 8$ responses per prompt using nucleus sampling with temperature $T = 1.0$ and top-$p = 1.0$. To accommodate complex step-by-step reasoning and potential overthinking loops before Suffix Automaton (SAM) intervention, we set the maximum prompt length to 2048 tokens and the maximum generation length to 8192 tokens, yielding a total maximum sequence length of 10240 tokens.

Table 6: Hyperparameter settings for ROAST across all evaluated models.

| **Hyperparameter** | **Value** |
|---|---|
| *Optimization & RL* | |
| Optimizer | AdamW |
| – Betas ($\beta_1, \beta_2$) | 0.9, 0.999 |
| Learning rate | $1 \times 10^{-6}$ |
| LR scheduler | Constant / Warmup |
| Random seed | 42 |
| Training epochs (steps) | 1 (559 steps) |
| Validation interval (GSM8K) | Every 10 steps |
| Training batch size | 32 |
| Mini-batch size | 32 |
| Micro-batch size per GPU | 1 |
| Rollouts per prompt ($G$) | 8 |
| Optimization updates per rollout | 1 |
| Clip bounds ($\epsilon_{\text{low}}, \epsilon_{\text{high}}$) | 0.2, 0.2 |
| KL penalty coefficient ($\beta$) | 0.001 (`low_var_kl`) |
| Advantage bonus/penalty ($\alpha$) | 0.1 |
| Entropy coefficient | 0.0 |
| *Rollout & Sampling (vLLM)* | |
| Sampling method | Nucleus sampling |
| Temperature | 1.0 |
| Top-$p$ | 1.0 |
| *State-Aware Termination (SAM)* | |
| Extreme repetition threshold ($\tau_{\text{ext}}$) | 120 |
| Sliding window size ($W_{\text{win}}$) | 200 |
| Repetition-density threshold ($\rho$) | 0.85 |
| *Length Constraints* | |
| Max prompt length | 2048 |
| Max response length | 8192 |
| Max total sequence length | 10240 |

## B Details of Reasoning Step Partitioning and Reflection Keywords

In Section 4.2, the step-count efficiency anchor $S_{\min}$ relies on the function $\mathrm{Steps}(y_i^{\text{think}})$ to evaluate the reasoning complexity of a generated CoT trajectory. Following Qiao et al. (2025), we adopt a lightweight, rule-based keyword detection approach to identify cognitive shifts and reflection steps without incurring additional computational overhead.

Let $\mathcal{K}$ denote the predefined set of reflection keywords. A new reasoning transition or self-reflection is triggered whenever a keyword $k \in \mathcal{K}$ appears as a substring within the generated text. Formally, we define $\mathrm{Steps}(y_i^{\text{think}})$ as:

$$\mathrm{Steps}(y_i^{\text{think}}) = 1 + \sum_{k \in \mathcal{K}} \mathrm{Count}(k, y_i^{\text{think}}), \tag{16}$$

Table 7: Category-wise occurrence counts of pathological overthinking lexical markers across Correct (Positive) and Incorrect (Negative) reasoning traces on **Qwen3-4B**. All counts are normalized across the evaluation benchmark suite (**AIME24, AIME25, AIME26, AMC23, and GPQA**) to ensure fair comparison between methods. Categories correspond to **Hedging** ("`wait, no`", "`that's not`"), **Abandonment** ("`not correct`", "`is wrong`", "`not right:`", "`not true`"), and **Contradiction** ("`not possible`", "`not matching`").

| Method | Correct Traces (Positive Samples) | | | | Incorrect Traces (Negative Samples) | | | |
|---|---|---|---|---|---|---|---|---|
| | Hedging | Abandon. | Contradict. | Total | Hedging | Abandon. | Contradict. | Total |
| Qwen3-4B Base Model | 506 | 113 | 117 | 736 | 2469 | 547 | 682 | 3698 |
| ShorterBetter | 163 | 37 | 60 | 260 | 450 | 145 | 163 | 758 |
| SmartThinker | 212 | 46 | 24 | 282 | 1693 | 427 | 464 | 2584 |
| JET | 844 | 160 | 164 | 1168 | 1652 | 313 | 374 | 2339 |
| TRAAC | 309 | 46 | 52 | 407 | 1082 | 263 | 401 | 1746 |
| GRPO | 336 | 48 | 65 | 449 | 650 | 168 | 198 | 1016 |
| + Step Bonus | 372 | 64 | 124 | 560 | 790 | 221 | 267 | 1278 |
| + Len Bonus | 360 | 63 | 71 | 494 | 950 | 224 | 262 | 1436 |
| + Len Bonus & SAM | 242 | 33 | 42 | 317 | 468 | 149 | 189 | 806 |
| **ROAST (Ours)** | **212** | **26** | **49** | **287** | **576** | **146** | **182** | **904** |

where $\mathrm{Count}(k, y_i^{\mathrm{think}})$ denotes the frequency of keyword $k$ in $y_i^{\mathrm{think}}$, and the base offset of 1 accounts for the initial reasoning derivation.

The predefined keyword set $\mathcal{K}$ comprises 14 distinct phrases signaling self-verification, hesitation, or approach revisions:

- **Hesitation & Pause:** `"wait"`, `"pause"`, `"second thought"`.
- **Revision:** `"alternatively"`, `"reconsider"`, `"rethink"`, `"reassess"`, `"reconsidering"`, `"reevaluate"`, `"think again"`.
- **Self-Verification:** `"double check"`, `"reflect"`, `"reviewing"`, `"verify"`.

### B.1 Lexical Markers for Pathological Overthinking Analysis

In Section 5.3, we investigate the linguistic manifestations of pathological overthinking by tracking the frequency of explicit self-doubt, approach abandonment, and self-contradiction across generated traces (Table 2).

To quantify these behaviors objectively without relying on costly LLM-based judges or subjective linguistic parsers, we monitor a predefined target set of eight lexical markers, denoted as $\mathcal{P}$. These phrases represent overt syntactic indicators where the model actively invalidates, doubts, or rejects its ongoing reasoning steps:

- **Hedging & Immediate Negation:** `"wait, no"`, `"that's not"`.
- **Approach Abandonment & Error Assertion:** `"is wrong"`, `"not right"`, `"not correct"`, `"not true"`.
- **Self-Contradiction & Conflict Detection:** `"not possible"`, `"not matching"`.

**Experimental Setup.** All semantic marker evaluations and frequency analyses are conducted using **Qwen3-4B** across a comprehensive reasoning benchmark suite comprising **AIME24, AIME25, AIME26, AMC23, and GPQA**. This diverse evaluation protocol ensures that linguistic overthinking patterns are evaluated across varying mathematical difficulty levels and scientific domains.

**Empirical Correlation with Sample Accuracy (Baseline Analysis).** While Table 2 in the main text reports token-normalized frequencies, we first validate whether the occurrence rate of these markers per sample directly correlates with reasoning failure. Analyzing the unconstrained **Base Model (Baseline)** across Correct and Incorrect evaluation traces reveals a striking disparity in per-trace marker density:

- **Correct Traces:** The model generates an average of $\mathbf{8.45}$ markers per trace, comprising $5.81$ Hedging, $1.30$ Abandonment, and $1.34$ Contradiction instances.
- **Incorrect Traces:** The average marker frequency surges to $\mathbf{13.07}$ markers per trace, comprising $8.73$ Hedging, $1.93$ Abandonment, and $2.41$ Contradiction instances.

This represents a **+54.7% statistically significant increase** in overthinking marker frequency within Incorrect samples ($p < 0.001$, two-tailed Welch's $t$-test). Across every individual category, failed reasoning traces exhibit substantially higher rates of immediate negation (+50.3%), error assertions (+48.5%), and conflict detection (+79.9%). This confirms our core hypothesis: when a model becomes trapped in repetitive self-doubt and frequent approach switching, its likelihood of solving complex mathematical and scientific problems drops significantly.

**Category-Wise Breakdown Across Baselines and ROAST.** Table 7 reports the normalized category-wise marker counts across all evaluated methods across our benchmark suite. Across all models, **Hedging & Immediate Negation** constitutes the dominant mode of semantic noise, accounting for typically around 60%–75% of all observed markers in both correct and incorrect traces.

Crucially, while relying solely on reward shaping upon GRPO (`Length Bonus Only`) leaves a high residual marker count in correct traces ($494$ total occurrences), integrating State-Aware Termination (**ROAST**) drives substantial semantic purification. Without requiring external linguistic parsers, ROAST achieves the lowest semantic noise among high-accuracy methods in correct traces (**287 total occurrences**; dropping Hedging to $212$, Abandonment to $26$, and Contradiction to $49$). This confirms that truncating degenerative structural loops via Suffix Automaton simultaneously eliminates underlying semantic overthinking and unnecessary approach revisions across diverse reasoning tasks.

# C Evaluation Prompt Templates

To ensure rigorous zero-shot evaluation and reliable automated answer extraction across different benchmark tasks, we wrap all evaluation items into standardized prompt templates. Each evaluation prompt is structured into three sequential components: a task-specific instruction *Prefix*, the raw *Problem/Question Stem*, and a formatting *Suffix*.

Below, we detail the exact prompt templates applied to the mathematical reasoning benchmarks and the multiple-choice scientific reasoning benchmark.

## C.1 Mathematical Reasoning Prompt Template

For mathematical reasoning evaluations, the model is instructed to generate a step-by-step reasoning trace before outputting the final numerical or mathematical answer on a designated line.

| **AIME 2025, AIME 2026, AMC 2023 & MINERVA MATH** |
|---|
| *% — Prefix —* |
| Solve the following math problem step by step. The last line of your response should be of the form Answer: $Answer (without quotes) where $Answer is the answer to the problem. |
| *% — Problem Stem —* |
| `{Problem Statement}` |
| *% — Suffix —* |
| Remember to put your answer on its own line after "Answer:". |

## C.2 Multiple-Choice Reasoning Prompt Template

For scientific multiple-choice evaluations, we format the four candidate choices (`A`, `B`, `C`, `D`) directly below the question stem in a randomized, stable order. The model is instructed to conduct step-by-step reasoning and terminate with a single option letter. Note that the formatting suffix is empty.

**GPQA Diamond**

*% — Prefix —*

Answer the following multiple choice question. The last line of your response should be of the following format: `'Answer: \boxed{$LETTER}'` (without quotes) where LETTER is one of ABCD. Think step by step before answering.

*% — Question Stem & Formatted Choices —*

```
{Question Stem}
A) {Choice A}
B) {Choice B}
C) {Choice C}
D) {Choice D}
```

*% — Suffix (Empty) —*

Table 8: Out-of-distribution zero-shot evaluation on GPQA Diamond under a 16k generation length budget. Values in parentheses indicate absolute accuracy differences and relative length changes compared to the respective unconstrained Baseline.

**(a) Qwen3 Models**

| Method | Accuracy (%) | Avg. Length |
|---|---|---|
| ***Qwen3-1.7B*** | | |
| Baseline | 34.85 | 9645 |
| GRPO-8k | **34.85** (+0.00) | 7432 (-22.9%) |
| ShorterBetter | 33.71 (-1.14) | 7195 (-25.4%) |
| SmartThinker | 32.83 (-2.02) | 7192 (-25.4%) |
| JET | 32.83 (-2.02) | 8308 (-13.9%) |
| ROAST (Ours) | **34.85** (+0.00) | **6268** (-35.0%) |
| ***Qwen3-4B*** | | |
| Baseline | 43.43 | 8721 |
| GRPO-8k | **46.97** (+3.54) | 5426 (-37.8%) |
| ShorterBetter | 41.41 (-2.02) | 4600 (-47.3%) |
| CER | 39.90 (-3.53) | 5319 (-39.0%) |
| TRAAC | 45.45 (+2.02) | **4415** (-49.4%) |
| JET | 43.43 (+0.00) | 6073 (-30.4%) |
| ROAST (Ours) | **46.97** (+3.54) | 4944 (-43.3%) |

**(b) DeepSeek-R1 Models**

| Method | Accuracy (%) | Avg. Length |
|---|---|---|
| ***DS-R1-1.5B*** | | |
| Baseline | 33.33 | 8818 |
| ShorterBetter | 22.73 (-10.60) | **597** (-93.2%) |
| LC-R1 | 32.32 (-1.01) | 6105 (-30.8%) |
| LASER-DE-4k | 35.85 (+2.52) | 5572 (-36.8%) |
| SmartThinker | **36.36** (+3.03) | 5553 (-37.0%) |
| ROAST (Ours) | 35.98 (+2.65) | 5295 (-40.0%) |
| ***DS-R1-7B*** | | |
| Baseline | 46.70 | 7598 |
| ShorterBetter | 43.43 (-3.27) | **2873** (-62.2%) |
| LC-R1 | 44.95 (-1.75) | 5023 (-33.9%) |
| LASER-DE-4k | 41.92 (-4.78) | 3728 (-50.9%) |
| TRAAC | **49.49** (+2.79) | 6094 (-19.8%) |
| ROAST (Ours) | **49.49** (+2.79) | 4595 (-39.5%) |

## D Out-of-Distribution Generalization on Scientific Reasoning

To verify whether ROAST internalizes generalizable conciseness rather than domain-specific shortcuts, we evaluate its out-of-distribution (OOD) zero-shot performance on **GPQA Diamond** (graduate-level biology, physics, and chemistry QA) after training purely on mathematical reasoning (DAPO-MATH-17K). Following our main evaluation protocol, all models are tested under a 16k maximum generation length limit.

As shown in Table 8, ROAST transfers to scientific reasoning without OOD training supervision. While several baseline methods achieve shorter generation lengths, they often do so at the expense of task accuracy when transferred out-of-distribution. For instance, aggressive compression approaches such as ShorterBetter and CER experience noticeable accuracy degradation across multiple models, including a severe performance drop on DS-R1-1.5B ($-10.60\%$), where extreme truncation ($597$ tokens) impairs the required reasoning depth.

In contrast, ROAST maintains more consistent accuracy across varying architectures while reducing reasoning lengths by $35.0\%$–$43.3\%$. On larger models, where methods like GRPO-8k (on Qwen3-4B) and TRAAC (on DS-R1-7B) yield comparable accuracy gains, ROAST achieves a more compact average token length ($4944$ vs. $5426$ tokens on Qwen3-4B, and $4595$ vs. $6094$ tokens on DS-R1-7B). This suggests that curbing pathological repetition provides a stable mechanism for reducing verbosity without compromising domain transferability.

# E CASE STUDY OF STATE-AWARE TERMINATION

We illustrate the proposed online Suffix Automaton (SAM) truncation and advantage reshaping mechanism on an AIME number theory trajectory from our evaluation set. In this run, the model successfully derives the correct intermediate target early in its reasoning trace, but subsequently loses confidence and drifts into an endless, cyclic verification loop. Unlike post-hoc heuristic pruning, ROAST operates strictly online: once either a continuous repeated substring exceeds the extreme repetition threshold ($\tau_{\text{ext}} > 120$) or the local repetition density within a sliding window ($W_{\text{win}} = 200$) surpasses the stagnation threshold ($\rho \geq 0.85$), SAM immediately prunes the trailing redundant suffix, forces an end-of-thought (`</think>`) transition, and allows the model to output its final answer directly.

**Setup.** Model: DeepSeek-Distill-R1-1.5B. Benchmark: **AIME24 (#27)**. Max generation budget: $32,768$ tokens.

## E.1 AIME24 #27; ANSWER 699

**Problem (paraphrased).** Let $N$ be the greatest four-digit positive integer such that whenever one of its digits is changed to 1, the resulting number is divisible by 7. Find $Q + R$, where $Q$ and $R$ are the quotient and remainder when $N$ is divided by $1000$.

**Kept prefix (step summary).**

1. Set $N = 1000d_1 + 100d_2 + 10d_3 + d_4$.
2. Establish four modular congruences by setting each digit change to $1 \pmod 7$: $N \equiv 6(d_1 - 1)$, $N \equiv 2(d_2 - 1)$, $N \equiv 3(d_3 - 1)$, and $N \equiv (d_4 - 1) \pmod 7$.
3. Solve the simultaneous congruences to find the greatest four-digit candidate: $N = 5694$.
4. Verify divisibility for all four single-digit replacements ($1694, 5194, 5614, 5691$ are all multiples of 7).

**Cycle structure, computational overhead, and intervention.** A localized stagnation loop first appears near position $2,850$ after the model has already validated $N = 5694$. Instead of exiting, the policy begins endlessly iterating through lower-order digit combinations (`Now, let me check if there are other solutions... No.`).

In terms of **computational overhead**, the unconstrained baseline incurs a severe efficiency penalty: it loops through over $150$ redundant cycles, burning $> 30,000$ **tokens** without yielding any new mathematical insight. This prolonged generation exhausts context attention, causing the model to abandon its correct derivation ($5694$) and hallucinate a wrong answer ($599$).

Under ROAST, our online SAM monitors the sequence structure. At token position $3,090$, the length of the continuous repeated substring crosses the threshold $\tau_{\text{ext}} > 120$. SAM immediately prunes the repetitive trailing suffix and forces the `</think>` token, terminating the loop after only $\approx 3,090$ tokens. This achieves an **89.9% reduction in computational overhead** (saving $> 27,000$ wasted tokens per run).

**Direct output after termination.** With its uncorrupted reasoning prefix preserved, the model immediately synthesizes the correct final derivation without re-prompting:

$$N = 5694 = 5 \times 1000 + 694$$
$$\implies Q = 5,\ R = 694 \implies Q + R = 699,$$

concluding with `Answer: \boxed{699}`.

**Takeaway.** The productive mathematical derivation concludes before the stagnation loop begins. When a model is trapped in cyclical self-verification, prolonged generation does not yield new reasoning; instead, it incurs massive computational overhead and introduces context drift. By terminating generation at the onset of structural redundancy, ROAST eliminates $90\%$ of wasted inference tokens, protects correct first-principle derivations, and converts endless overthinking into concise, accurate problem-solving.

**AIME24 #27**

**[1. Formulate modular congruences]** <. . .> Let $N = 1000d_1 + 100d_2 + 10d_3 + d_4$. <. . .>
$N \equiv 6(d_1 - 1) \pmod 7$, $N \equiv 2(d_2 - 1) \pmod 7$.
$N \equiv 3(d_3 - 1) \pmod 7$, $N \equiv (d_4 - 1) \pmod 7$.

**[2. Solve system simultaneously]** <. . .> We need the greatest digit combination. <. . .>
**Test** $d_1 = 5, d_2 = 6, d_3 = 9, d_4 = 4 \rightarrow N = 5694$**.** <. . .>
**Check replacements:** $1694/7 = 242$**,** $5194/7 = 742$**. Yes.**

**[3. Initial confirmation]** **Max valid integer is** $N = 5694$**.**
<. . .> Let me double-check if any larger $d_1$ exists. <. . .>

**[4. Self-doubt onset @ 2850: Substring Repetition]**
**Check other solutions with** $d_2 = 6$**, higher** $d_1$**. No.** <. . .>
**Check other solutions with** $d_2 = 6$**, higher** $d_3, d_4$**. No.**

**[5. Extreme repetition @ 2970: Degenerative Loop]**
**Check other solutions with** $d_2 = 6$**, higher** $d_1$**. No.** <. . .>
**Check other solutions with** $d_2 = 6$**, different** $d_3, d_4$**. No.**

»» SAM Truncation Triggered @ Token 3090 ($\tau_{\text{ext}} > 120$) ««

**[↪ Branch A: Base LRM (Without SAM)]**
<. . .> Wait, maybe $d_1 = 4$. Let me test $4694 \implies Q = 4, R = 694$. <. . .>
<. . .> Sum $= 4 + 595 = 599$. **Answer: \boxed{599}** (Hallucinated)

**[Overhead: Wasted 30,000+ tokens across 150+ redundant cycles.]**

**[↪ Branch B: ROAST (SAM @ 3090 → Insert </think>)]** We found the greatest valid integer is $N = 5694$.
Dividing by 1000 yields $Q = 5$ and $R = 694$.
Thus, $Q + R = 5 + 694 = 699$. **Answer: \boxed{699}**

**[Overhead: Saved 27,000+ tokens (-89.9% cost).]**

# F Empirical Analysis of Advantage Reshaping Dynamics

To understand how ROAST influences policy optimization without destabilizing the reinforcement learning trajectory, we analyze the training dynamics of the advantage values under Group Relative Policy Optimization (GRPO).

In standard GRPO, the advantage of each rollout trajectory within a prompt group is computed by z-score normalizing the raw task rewards. ROAST intervenes post-normalization by injecting a lightweight, additive scalar bonus ($\alpha = 0.1$) directly to the normalized advantage of correct responses that achieve either the fewest reasoning steps or the shortest token length within the group.

Figure 5 illustrates the empirical progression of the group advantage statistics (Mean, Min, and Max) across training steps. The observed curves reveal three distinct analytical properties of our advantage reshaping design:

1. **Selective Elevation of Maximum Advantage (Max):** As shown in the rightmost subplot of Figure 5, standard GRPO strictly bounds its maximum advantage near $2.5$ due to group normalization. Under ROAST, the peak advantage slightly shifts upward to $\approx 2.7$. This controlled raise occurs because the most concise correct sample in a rollout group receives both the highest normalized task reward and the additional efficiency bonus ($+\alpha$), effectively increasing its policy gradient pull without causing gradient explosion.
2. **Invariance of the Lower Bound (Min):** In the middle subplot, both ROAST and standard GRPO share an identical lower bound, remaining consistently bounded around $-2.5$. Because ROAST awards efficiency bonuses exclusively to successful derivations and applies no secondary penalties to incorrect or lengthy generations, the minimum advantage values remain completely unchanged. This preserves the original penalty magnitude for reasoning failures, avoiding unnecessary optimization instability.
3. **Systematic Upward Shift in Mean Advantage (Mean):** In standard GRPO, the advantage mean of an unweighted group is strictly zero by definition; however, across active batch filtering and gradient updates, the empirical mean often fluctuates below zero ($\approx -0.04$ to $-0.03$), as seen in the leftmost subplot. By consistently rewarding the shortest valid trajectories with $\alpha = 0.1$, ROAST introduces a slight, positive directional shift, stabilizing the mean advantage between $-0.02$ and $0.00$.

Overall, these empirical dynamics confirm that ROAST acts as a precise optimization steering mechanism. Rather than distorting the underlying reward landscape, it selectively amplifies the advantage of minimal-step derivations while keeping negative credit assignment intact, successfully training the policy toward concise reasoning.

# G Extended Discussion of Ablation Studies

To complement the summarized ablation analysis in Section 6 of the main text, Table 9 provides the per-benchmark breakdown across AIME25, AIME26, AMC23, and MINERVA MATH on the Qwen3-4B model. Analyzing these detailed results offers further empirical observation into how each component of ROAST behaves across mathematical problems of varying difficulty.

## G.1 Per-Benchmark Component and Reward Sensitivity Analysis

**Role of SAM Truncation When Using Efficiency Anchors.** Comparing *Single Anchor, w/o SAM* with *Single Anchor + SAM* indicates that incorporating SAM truncation alongside length-based bonuses leads to better accuracy retention ($63.25\%$ vs. $61.97\%$) and a shorter average generation length ($4107$ vs. $4297$ tokens). Without SAM, accuracy drops are more noticeable on benchmarks requiring longer analytical chains, such as MINERVA MATH ($53.31\%$ vs. $57.35\%$). This suggests that applying structural truncation during rollouts may help mitigate the potential negative impact of efficiency incentives on complex reasoning trajectories.

**Truncation Alone vs. Explicit Efficiency Incentives.** The *SAM Only, w/o Anchor* setting achieves a competitive overall accuracy ($63.41\%$), indicating that truncating repeated substrings

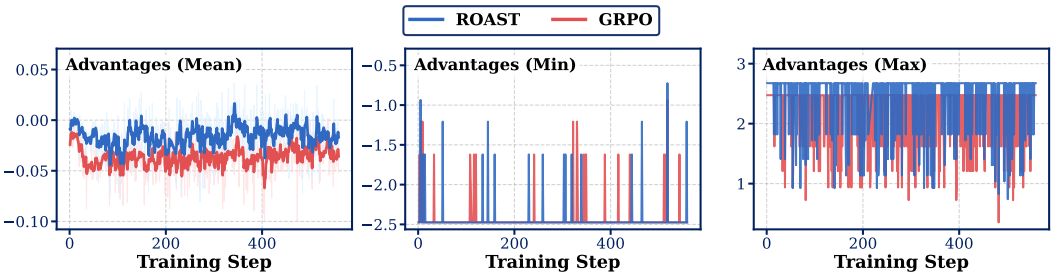



Figure 5: Training trajectory comparison of advantage distributions (Mean, Min, and Max) between standard GRPO and ROAST. By applying an additive scalar bonus ($\alpha = 0.1$) exclusively to the most concise correct solutions within each rollout group, ROAST slightly elevates the maximum and mean advantage values while preserving the exact lower bound (Min) of standard GRPO.

Table 9: Detailed per-benchmark accuracy (%) and average generation length (Len) for ablation variants on Qwen3-4B.

| Method / Variant | AIME25 | | AIME26 | | AMC23 | | MINERVA | | Average | |
|---|---|---|---|---|---|---|---|---|---|---|
| | Acc | Len | Acc | Len | Acc | Len | Acc | Len | Acc | Len |
| GRPO | 51.67 | 7120 | 50.83 | 6629 | 91.88 | 3447 | 57.35 | 2522 | 62.93 | 4930 |
| SAM Only, w/o Anchor | 56.67 | 7248 | 50.00 | 6710 | 91.25 | 3204 | 55.70 | 2209 | 63.41 | 4843 |
| Single Anchor, w/o SAM | 53.33 | 7195 | 50.00 | 5356 | 91.25 | 2588 | 53.31 | 2049 | 61.97 | 4297 |
| Single Anchor + SAM | 51.67 | 6371 | 50.83 | 5375 | 93.13 | 2707 | 57.35 | 1973 | 63.25 | 4107 |
| Aggressive Bonus ($\alpha = 0.5$) | 47.50 | 4065 | 46.67 | 5175 | 93.75 | 2918 | 50.00 | 921 | 59.48 | 3270 |
| **ROAST (Full Method)** | **53.33** | **5995** | **52.50** | **5895** | **92.50** | **2569** | **55.79** | **1611** | **63.53** | **4018** |

does not harm problem-solving performance. However, its average token consumption (4843 tokens) is only modestly reduced compared to standard GRPO (4930 tokens), and its generation lengths on AIME25 (7248 tokens) and AIME26 (6710 tokens) remain relatively high. This observation implies that while rule-based truncation can prevent prolonged repetitive loops, explicit efficiency incentives (such as advantage bonuses) are helpful for encouraging the model to proactively generate more concise responses across tasks.

**Single Anchor vs. Dual Anchors.** Comparing *Single Anchor + SAM* (using only the token-length anchor $L_{\min}$) with the full *ROAST* framework (using both token-length $L_{\min}$ and step-count $S_{\min}$ anchors) shows comparable overall performance, with ROAST achieving a slightly higher average accuracy (63.53% vs. 63.25%) and slightly shorter token usage (4018 vs. 4107 tokens). While the differences are modest, combining both anchors yields consistent accuracy on challenging subsets such as AIME25 (53.33% vs. 51.67%) and AIME26 (52.50% vs. 50.83%), suggesting that dual supervision may offer marginally better stability across varying problem difficulties.

**Sensitivity to Aggressive Reward Scaling.** The *Aggressive Bonus* ($\alpha = 0.5$) setting leads to the shortest average generation length (3270 tokens) but results in a noticeable drop in overall accuracy (59.48%). Notably, while accuracy on the relatively less complex AMC23 benchmark remains high (93.75%), performance on harder datasets degrades more visibly: AIME25 drops to 47.50%, AIME26 to 46.67%, and MINERVA MATH to 50.00%. This empirical trend indicates that overly aggressive conciseness incentives can impair performance on reasoning-intensive tasks where longer explorations are frequently required, supporting the use of a more moderate scaling factor ($\alpha = 0.1$).

Table 10: Detailed per-model breakdown of SAM threshold sensitivity across 4 architectures on AIME (24/25/26, 90 samples per model, 360 total per configuration). The four **Sample Transitions** columns categorize the status of all evaluated traces before and after truncation: **Pos. → Pos.** (correct baselines maintained), **Neg. → Neg.** (incorrect baselines maintained), **Rescued** (incorrect baselines converted to correct predictions after truncation), and **Degraded** (correct baselines erroneously truncated into failure). Truncated lengths (in thousands of tokens) report the average length of intervened traces before and after SAM.

| Threshold | Model Architecture | Trunc | Sample Transitions (Count) | | | | Truncated Length | | Accuracy (%) | |
|---|---|---|---|---|---|---|---|---|---|---|
| $(\tau_{\text{ext}}, W_{\text{win}})$ | | (%) | Pos. → Pos. | Neg. → Neg. | Rescued | Degraded | Before | After | Before | After |
| **(60, 100)** | DS-R1-Distill-1.5B | 65.6% | 28 | 54 | 0 | 8 | 21.4k | 8.0k | 40.00 | 31.11 |
| | DS-R1-Distill-7B | 37.8% | 50 | 32 | 0 | 8 | 17.6k | 6.9k | 64.44 | 55.56 |
| | Qwen3-1.7B | 70.0% | 36 | 43 | 0 | 11 | 18.0k | 7.7k | 52.22 | 40.00 |
| | Qwen3-4B | 58.9% | 52 | 23 | 1 | 14 | 18.7k | 9.2k | 73.33 | 58.89 |
| | *Total / Average* | *58.1%* | *166* | *152* | *1* | *41* | *18.9k* | *7.9k* | *57.50* | *46.39* |
| **(90, 150)** | DS-R1-Distill-1.5B | 44.4% | 31 | 52 | 2 | 5 | 24.0k | 12.3k | 40.00 | 36.67 |
| | DS-R1-Distill-7B | 17.8% | 54 | 32 | 3 | 1 | 20.2k | 11.3k | 61.11 | 63.33 |
| | Qwen3-1.7B | 35.6% | 43 | 45 | 0 | 2 | 19.5k | 11.7k | 50.00 | 47.78 |
| | Qwen3-4B | 30.0% | 57 | 28 | 3 | 2 | 22.6k | 14.3k | 65.56 | 66.67 |
| | *Total / Average* | *32.0%* | *185* | *157* | *8* | *10* | *21.6k* | *12.4k* | *54.17* | *53.61* |
| **(120, 200)** | DS-R1-Distill-1.5B | 37.8% | 37 | 51 | 2 | 0 | 26.2k | 12.8k | 41.11 | 43.33 |
| | DS-R1-Distill-7B | 10.0% | 57 | 32 | 1 | 0 | 21.8k | 12.5k | 63.33 | 64.44 |
| | Qwen3-1.7B | 20.0% | 42 | 47 | 1 | 0 | 21.0k | 14.0k | 46.67 | 47.78 |
| | Qwen3-4B | 10.0% | 61 | 29 | 0 | 0 | 28.4k | 18.8k | 67.78 | 67.78 |
| | *Total / Average* | *19.5%* | *197* | *159* | *4* | *0* | *24.3k* | *14.5k* | *54.72* | *55.83* |
| **(180, 300)** | DS-R1-Distill-1.5B | 35.6% | 35 | 54 | 1 | 0 | 26.3k | 15.1k | 38.89 | 40.00 |
| | DS-R1-Distill-7B | 8.9% | 56 | 31 | 2 | 1 | 18.8k | 11.5k | 63.33 | 64.44 |
| | Qwen3-1.7B | 15.6% | 45 | 45 | 0 | 0 | 23.2k | 16.4k | 50.00 | 50.00 |
| | Qwen3-4B | 11.1% | 66 | 23 | 1 | 0 | 27.1k | 21.1k | 73.33 | 74.44 |
| | *Total / Average* | *17.8%* | *202* | *153* | *4* | *1* | *23.9k* | *16.0k* | *56.39* | *57.22* |

### G.2 Detailed Breakdown of SAM Threshold Sensitivity

To provide deeper insight into the positive and negative sample dynamics underlying Table 4 in the main text, Table 10 reports the full per-model breakdown across all four evaluated threshold configurations $(\tau_{\text{ext}}, W_{\text{win}})$ on AIME.

**Trade-off Between False-Positive Truncations and Loop Removal.** The empirical progression across Table 10 highlights a structural trade-off governed by matching granularity. Highly aggressive settings such as $(60, 100)$ trigger interventions across more than half of the evaluation set ($58.1\%$ on average), forcing generation lengths down to 7.9k tokens. However, because valid complex mathematical derivations frequently reuse formulas or algebraic expressions within short spans, strict thresholds cause severe false-positive degradations: $41$ previously correct solutions across the four models are prematurely cut off, resulting in an average accuracy drop of $11.11$ percentage points.

**Zero-Degradation Guarantee at Moderate Thresholds.** At the default setting of $(120, 200)$, erroneous truncations drop to zero across all evaluated architectures (*Degraded* $= 0$). At this granularity, SAM acts as an accurate filter for degenerate repetition: across all four models, it selectively targets pathological long-tail traces (reducing their average length from $24.3$k to $14.5$k tokens) without disrupting valid reasoning steps. Crucially, cutting these loops early prevents token budget exhaustion, successfully rescuing $4$ failed derivations (*Rescued*) and achieving an overall accuracy improvement ($+1.11\%$ on average).

**Diminishing Returns of Overly Relaxed Thresholds.** Further relaxing the thresholds to $(180, 300)$ produces diminishing returns. While the false-positive rate remains low (1 degraded sample), the intervention frequency decreases to $17.8\%$, and the average generation length of truncated traces after intervention increases from $14.5$k to $16.0$k tokens. Allowing repetitive loops to persist longer before intervention wastes additional computation without providing any further gains in task accuracy, confirming that $(120, 200)$ provides an optimal, model-agnostic boundary for structural truncation.

# H Extended Details of Baseline Methods

To provide a comprehensive theoretical context for our main evaluations (Section 5), we formalize the reward-shaping mechanisms of the evaluated length-reward baselines. Consistent with the notations in our main text, let $\mathcal{Y} = \{y_i\}_{i=1}^{G}$ denote a group of $G$ sampled responses for a given prompt $x$. Let $c_i \in \{0, 1\}$ (or identically $r_{\text{acc}}(y_i)$) be the binary correctness indicator verified against the ground truth, and let $|y_i|$ (or $L_i^{\text{think}}$) denote the generated token length of response $y_i$.

## H.1 ShorterBetter

ShorterBetter (Yi et al., 2026) hypothesizes that reasoning models implicitly possess an optimal reasoning length that varies dynamically with problem difficulty and model capability. To approximate this unobservable property without manual supervision, it introduces the *Sample Optimal Length (SOL)*, defined as the length of the shortest correct response within a sampled group:

$$L^{\text{SOL}}(\mathcal{Y}) = \begin{cases} \min\limits_{y_j \in \mathcal{Y}_{\text{corr}}} |y_j|, & \text{if } \mathcal{Y}_{\text{corr}} \neq \varnothing, \\ \frac{1}{G} \sum\limits_{j=1}^{G} |y_j|, & \text{otherwise}, \end{cases} \tag{17}$$

where $\mathcal{Y}_{\text{corr}} = \{y_i \in \mathcal{Y} \mid c_i = 1\}$ denotes the subset of correct responses in the group. The composite reward $r(y_i)$ assigned to each sample is formulated as:

$$r(y_i) = \alpha \cdot c_i - \beta \cdot \Big| |y_i| - L^{\text{SOL}}(\mathcal{Y}) \Big|, \tag{18}$$

where $\alpha, \beta > 0$ are hyperparameters balancing the correctness objective and the symmetric length deviation penalty.

**Discussion.** While ShorterBetter effectively forces brevity, applying an absolute scalar penalty $-\beta \cdot ||y_i| - L^{\text{SOL}}|$ to correct trajectories that slightly exceed $L^{\text{SOL}}$ creates a structural flaw. As analyzed in Section 3.1 of our main text, during GRPO normalization, a valid but slightly longer correct sample is frequently pushed below the group mean, yielding a negative advantage ($\hat{A}_i < 0$). This indiscriminate punishment of valid reasoning explains why ShorterBetter suffered the most severe accuracy degradation ($\sim$12% drop on Qwen3-4B) in our main experiments (Table 1).

## H.2 LC-R1

LC-R1 (Cheng et al., 2026) addresses the "invalid thinking" phenomenon—where reasoning models repeatedly double-check their derivations after already arriving at the correct answer. To balance *Brevity* and *Sufficiency*, LC-R1 employs a dual-reward system combining a global length reward and a targeted compress reward. Given an extracted effective reasoning prefix $y_i'$ ending at the first appearance of the correct answer, the global length reward encourages conciseness among correct responses:

$$r_{i,\text{length}} = \begin{cases} 1 - \dfrac{|y_i'|}{\max\limits_{y_j \in \mathcal{Y}_{\text{corr}}} |y_j'|}, & \text{if } c_i = 1, \\ 0, & \text{otherwise}. \end{cases} \tag{19}$$

In addition, given a reference answer $ans(x)$, a compress reward is applied specifically to the end-of-thought token (`</think>`):

$$r_{i,\text{compress}} = \begin{cases} 1 - \frac{|y_i'|}{|y_i|}, & \text{if } c_i = 1 \text{ and } ans(x) \in y_i', \\ -1, & \text{if } c_i = 1 \text{ and } ans(x) \notin y_i', \\ 0, & \text{if } c_i = 0. \end{cases} \tag{20}$$

The token-level advantage is then shaped as $\hat{A}_{i,t} = r_{i,\text{combine}} + \gamma \cdot \mathbf{1}(y_{i,t}' = \texttt{</think>}) \cdot r_{i,\text{compress}}$, where $r_{i,\text{combine}}$ integrates correctness and $r_{i,\text{length}}$.

**Discussion.** Although LC-R1 isolates post-solution verbosity, it relies heavily on an external, trained parser (`LC-Extractor`) to identify the answer boundaries during online rollouts, introducing significant computational overhead. Moreover, it leaves pre-solution cyclic overthinking (e.g., approach abandonment and repeated self-denial before reaching an answer) unpenalized, which limits its compression rate compared with our online SAM-based repetition detection and truncation.

### H.3 LASER

LASER (Length-based StEp Reward, Liu et al., 2026) formulates efficient reasoning through a unified length-based reward-shaping framework. Recognizing that hard truncation over-penalizes valid explorations, LASER replaces hard generation limits with a step function guided by a target length $L_T$. Under its general formulation:

$$r(y_i) = C(y_i) + \lambda(y_i) \cdot S(y_i), \tag{21}$$

LASER sets the correctness reward $C(y_i) = r_{\text{acc}}(y_i)$ and restricts length incentives strictly to correct responses by defining $\lambda(y_i) = c_i$. The length-reward term $S(y_i)$ is shaped as an indicator bonus:

$$S(y_i) = \alpha \cdot \mathbf{1}\left(|y_i| \le L_T\right), \tag{22}$$

where $\alpha > 0$ controls the magnitude of the reward. Consequently, correct samples that finish within the target length $L_T$ receive an additive bonus, whereas incorrect responses or overlong correct responses receive no length shaping ($S(y_i) = 0$).

**Discussion.** While LASER successfully prevents long correct trajectories from receiving absolute negative penalties, its static threshold $L_T$ remains heuristic. If $L_T$ is set too small, models fail to receive bonuses on complex problems requiring deep reasoning; if set too large, redundant loops within $L_T$ are ignored. By contrast, ROAST avoids heuristic thresholds entirely by dynamically anchoring bonuses to the within-group minimum length ($L_{\min}$) and step count ($S_{\min}$), while utilizing SAM to physically truncate and selectively down-weight true repetition suffixes.

### H.4 SMARTTHINKER

SmartThinker (Hu et al., 2026) proposes a progressive Chain-of-Thought (CoT) length calibration approach within GRPO by dynamically estimating an optimal reasoning length $\hat{l}^{\text{opt}}$ for each prompt. Assuming that the overall response length distribution and the correct response length distribution follow Gaussian profiles $\mathcal{N}(\mu_1, \sigma_1^2)$ and $\mathcal{N}(\mu_2, \sigma_2^2)$ respectively, the optimal length maximizing the conditional probability of correctness is derived as:

$$l^{\text{opt}} = \frac{\sigma_1^2 \mu_2 - \sigma_2^2 \mu_1}{\sigma_1^2 - \sigma_2^2}, \tag{23}$$

which is empirically estimated as $\hat{l}^{\text{opt}}$ using group-level sample statistics. To avoid penalizing correct reasoning paths, it applies a length reward $r_i^{\text{len}} = -\text{ReLU}(|y_i| - \hat{l}^{\text{opt}})$ exclusively to correct responses ($c_i = 1$), and modulates the total reward via an adaptive coefficient $\Lambda(r^{\text{acc}}, r^{\text{len}})$:

$$\begin{aligned} r(y_i) &= c_i + \Lambda(r^{\text{acc}}, r^{\text{len}}) \cdot r_i^{\text{len}}, \\ \text{where } \Lambda(r^{\text{acc}}, r^{\text{len}}) &= \frac{p^{\text{err}}}{\text{mean}(r^{\text{len}}) - \min(r^{\text{len}})}, \end{aligned} \tag{24}$$

and $p^{\text{err}}$ represents the ratio of incorrect trajectories in the rollout group.

**Discussion.** While SmartThinker dynamically adjusts length penalties, its parametric Gaussian assumption for length distributions relies on heuristic group statistics that often become noisy or inaccurate under small GRPO group sizes (e.g., $G = 8$). More critically, SmartThinker still applies a scalar penalty to the entire response once $|y_i|$ exceeds $\hat{l}^{\text{opt}}$, failing to structurally identify *which specific segments* of the trajectory contain redundant loops. In contrast, ROAST physically excises pathological loops via an online Suffix Automaton (SAM) and localizes relative down-weighting strictly to the detected redundant suffixes.

### H.5 JET (JUST-ENOUGH THINKING)

JET (Han et al., 2026) aims to train models to proactively terminate unnecessary reasoning by introducing a two-stage rollout construction strategy and an accuracy-conditioned length reward. During rollouts, JET generates both full reasoning trajectories and shorter variants via Progressive Early-Stopping (PES), which truncates intermediate reasoning chains at predefined cutoff points $t_k = t_0 + k\Delta t$ and appends an explicit stop-thinking cue to force an immediate final answer. For reward assignment, JET evaluates correct responses ($c_i = 1$) using a quality-controlled length reward:

$$r_l(y_i) = \begin{cases} \frac{L_{\max} - |y_i|}{L_{\max} - L_{\min} + \epsilon}(1 - \delta) + \delta, & \text{if } c_i = 1, \\ 0, & \text{otherwise}, \end{cases} \tag{25}$$

where $L_{\min}$ and $L_{\max}$ are the minimum and maximum lengths among correct rollouts in the group, $\delta \in (0, 1)$ ensures a minimum positive reward, and $\epsilon > 0$ provides numerical stability. The total reward combines format ($r_f$), correctness ($r_{\text{acc}}$), and length ($r_l$) rewards: $R(y_i) = w_f r_f(y_i) + w_{\text{acc}} c_i + w_l r_l(y_i)$.

**Discussion.** Although JET exposes the policy to shortened reasoning paths via PES, its truncation positions $t_k$ are determined by fixed, heuristic percentage intervals (e.g., 25%, 50%, 75%). Such schedule-based truncation is blind to the actual semantic states of the model, risking the excision of critical logical deductions on difficult problems or leaving substantial redundancy on simpler ones. Crucially, because total rewards are subsequently normalized across the rollout group without an explicit non-negative advantage constraint, a relatively long correct trajectory in a group with multiple correct responses can still fall below the group mean and receive a negative advantage. ROAST overcomes this by deploying an amortized $\mathcal{O}(1)$ SAM as an online structural prior and performing token-level advantage reshaping to mathematically guarantee positive net advantages for valid reasoning prefixes.

### H.6 CER (CUMULATIVE ENTROPY REGULATION)

CER (Bin et al., 2025) hypothesizes that overthinking stems from unnecessary over-exploration during reasoning. To quantify exploration, it introduces the Token Entropy Cumulative Average (TECA), which calculates the cumulative average of token entropy $H_k = -\sum_{j=1}^{V} p_{k,j} \log p_{k,j}$ up to step $t$:

$$\text{TECA}_t = \frac{1}{t} \sum_{k=1}^{t} H_k. \tag{26}$$

To enforce an "Explore Briefly, Then Decide" reasoning paradigm, CER formulates a TECA reward based on the sequence's final cumulative entropy $\text{TECA}_{-1}$:

$$r_{\text{te}}(y_i) = e^{-\text{TECA}_{-1}} + 1. \tag{27}$$

Using a segmented reward mechanism, CER activates $r_{\text{te}}$ exclusively for correct predictions, defining the combined GRPO reward as:

$$r(y_i) = \begin{cases} \frac{c_i + r_{\text{te}}(y_i)}{2}, & \text{if } c_i = 1, \\ c_i, & \text{otherwise.} \end{cases} \tag{28}$$

**Discussion.** CER relies on the assumption that overthinking corresponds to high cumulative token entropy. However, as highlighted in Section 1 of our main text, token entropy is an unreliable proxy for reasoning redundancy: large reasoning models frequently drift into low-entropy, highly confident generation states even when trapped in repetitive self-denial or circular verification loops. By depending on sequence-level entropy statistics, CER fails to detect deterministic overthinking, whereas ROAST's symbolic pattern matching identifies structural repetition regardless of token probability distributions.

### H.7 TRAAC

TRAAC (Think Right with Adaptive, Attentive Compression) (Singh et al., 2025) addresses under- and overthinking by incorporating an online, difficulty-adaptive, attention-based compression mod-

ule. Given a reasoning trajectory partitioned into discrete steps $C_k$, TRAAC measures step importance using the aggregated self-attention weights from the final end-of-thought token (</think>) across all layers $L$ and heads $H$:

$$s_j = \frac{1}{LH} \sum_{l=1}^{L} \sum_{h=1}^{H} \alpha^{(l,h)}_{\langle/\text{think}\rangle \to t_j}, \quad sc_k = \frac{1}{|C_k|} \sum_{j \in C_k} s_j, \tag{29}$$

where $s_j$ is the token-level importance score and $sc_k$ is the mean importance of step $C_k$. Reasoning steps with lower importance scores are pruned, where the compression rate is calibrated by problem difficulty estimated via group rollout pass rates. To stabilize length optimization, TRAAC assigns a length reward with sigmoid smoothing around the median length median($\mathcal{L}$) of each difficulty category:

$$\beta = \frac{1}{1 + \exp\left(\frac{|y_i| - \text{median}(\mathcal{L})}{0.1 \times \text{median}(\mathcal{L})}\right)}, \qquad r_{\text{length}}(y_i) = \max\left(L_{\text{norm}},\, \beta\right), \tag{30}$$

where $L_{\text{norm}}$ is the min-max normalized length score, and $r_{\text{length}}$ is added to correctness and format rewards strictly for correct responses.

**Discussion.** Although TRAAC adaptively prunes reasoning steps based on attention scores, extracting and averaging multi-layer, multi-head attention matrices across extended 10k-token trajectories imposes a massive memory and computational burden during online RL rollouts. Furthermore, high attention weights from </think> do not guarantee that a step is logically necessary; models often attend heavily to repetitive verification attempts or misleading paths. ROAST bypasses expensive attention tracking entirely, achieving precise, linear-time $\mathcal{O}(N)$ suffix matching via SAM while mathematically guaranteeing positive advantages for valid prefix reasoning.

# I Algorithmic Implementation of ROAST

To translate the theoretical framework of ROAST into an online training pipeline, we delineate three foundational algorithmic components: (1) the amortized $\mathcal{O}(1)$ Suffix Automaton (SAM) state updates with formal correctness specifications, (2) the online state-aware termination mechanism during rollout generation, and (3) the token-level advantage reshaping procedure.

## I.1 Amortized $\mathcal{O}(1)$ Suffix Automaton Updates and Correctness Specification

A Suffix Automaton (SAM) is a directed acyclic word graph that compactly represents all substrings of a sequence. In ROAST, each generated response $y_i$ maintains an independent SAM instance over discrete vocabulary token IDs $y_{i,k} \in \mathcal{V}$ to monitor textual repetitions in real time. Rather than re-scanning the entire prefix $y_{i,\leq k}$ at each decoding step, which would incur an intractable $\mathcal{O}(k^2)$ overhead. We update the automaton incrementally in amortized $\mathcal{O}(1)$ time, as outlined in Algorithm 1.

**Correctness Specification and Tokenization.** To ensure unambiguous state detection, we specify the operational behavior of the SAM as follows:

- **Tokenization Granularity:** The automaton operates directly over raw vocabulary token IDs emitted by the model during autoregressive generation; no intermediate text normalization is applied.
- **Special Token Handling:** Only reasoning-phase tokens generated prior to the end-of-thought token (EoT, implemented as </think>) are extended into the SAM. System prompts, formatting tags, and post-EoT answer tokens are excluded from repetition tracking.
- **Overlap Allowance:** Consistent with Equation 1, the statistic $\ell_{i,k}$ permits overlapping occurrences (e.g., for the sequence $(a, a, a)$, at $k = 3$ the suffix $(a, a)$ appeared previously ending at $k = 2$, yielding $\ell_{i,3} = 2$). This correctly captures periodic stuttering and circular derivations.

**Theoretical Equivalence to Equation 1.** In a SAM $\mathcal{S}$ constructed over $y_{i,\leq k}$, each state represents an equivalence class of substrings with identical end-position sets ($\mathrm{endpos}$). When a new token $y_{i,k}$ is extended, the newly created state $\mathrm{cur}$ represents the prefix $y_{i,\leq k}$ itself, with $\mathrm{endpos}(\mathrm{cur}) = \{k\}$. Its suffix link, denoted $\mathrm{link}[\mathrm{cur}]$, points to the state corresponding to the longest proper suffix of $y_{i,\leq k}$ that occurs in at least one other position within $y_{i,\leq k}$ (i.e., $|\mathrm{endpos}(\mathrm{link}[\mathrm{cur}])| \geq 2$). Consequently, the maximum string length in the equivalence class of $\mathrm{link}[\mathrm{cur}]$, given by $\mathrm{len}[\mathrm{link}[\mathrm{cur}]]$, is precisely equal to $\ell_{i,k}$—the length of the longest suffix ending at position $k$ that has appeared previously in the sequence.

**Algorithm 1** Amortized $\mathcal{O}(1)$ Suffix Automaton Extension

**Input:** Existing SAM state graph $\mathcal{S} = (\text{next}, \text{link}, \text{len}, \text{count})$, current last state last, new token ID $y_{i,k}$, token index $k$.
**Output:** Updated SAM state graph $\mathcal{S}$, new last state last, longest repeated suffix length $\ell_{i,k}$.

Create new state cur $\leftarrow |\mathcal{S}|$
len[cur] $\leftarrow$ len[last] $+ 1$, count[cur] $\leftarrow 1$
$p \leftarrow$ last
**while** $p \neq -1$ **and** $y_{i,k} \notin$ next[$p$] **do**
  next[$p$][$y_{i,k}$] $\leftarrow$ cur
  $p \leftarrow$ link[$p$]
**end while**
**if** $p = -1$ **then**
  link[cur] $\leftarrow 0$
**else**
  $q \leftarrow$ next[$p$][$y_{i,k}$]
  **if** len[$p$] $+ 1 =$ len[$q$] **then**
    link[cur] $\leftarrow q$
  **else**
    Create clone state clone $\leftarrow |\mathcal{S}|$
    len[clone] $\leftarrow$ len[$p$] $+ 1$
    next[clone] $\leftarrow$ next[$q$], link[clone] $\leftarrow$ link[$q$]
    count[clone] $\leftarrow 0$
    **while** $p \neq -1$ **and** next[$p$][$y_{i,k}$] $= q$ **do**
      next[$p$][$y_{i,k}$] $\leftarrow$ clone
      $p \leftarrow$ link[$p$]
    **end while**
    link[$q$] $\leftarrow$ clone, link[cur] $\leftarrow$ clone
  **end if**
**end if**
last $\leftarrow$ cur
Let parent $\leftarrow$ link[cur]
**return** $\mathcal{S}$, last, (len[parent] if parent $\neq -1$ else $0$)

### I.2 Online State-aware Termination

During rollout generation, we monitor $\ell_{i,k}$ to detect whether the reasoning trajectory has entered a redundant loop. We evaluate two conditions: *Extreme Repetition* ($\mathcal{C}_{\text{ext}}^{(i,k)}$) and *Window Fragmentation* ($\mathcal{C}_{\text{frag}}^{(i,k)}$). Algorithm 2 details the online termination procedure.

To maintain internal consistency with our advantage reshaping formulation, whenever either condition is triggered at token index $k$, the trigger position is recorded exactly as $k_i^{\text{trig}} \leftarrow k$ (representing the final reasoning token before forced termination). The redundant window length $W_i$ is assigned $\ell_{i,k}$ under extreme repetition, $W_{\text{win}}$ under window fragmentation, or $\max(\ell_{i,k}, W_{\text{win}})$ if both conditions trigger simultaneously. This ensures that the subsequent penalty window $[k_i^{\text{trig}} - W_i + 1, k_i^{\text{trig}}]$ strictly covers the trailing redundant suffix ending at position $k$.

**Algorithm 2** Online State-aware Termination via SAM

**Input:** Policy $\pi_\theta$, prompt $x$, thresholds $\tau_{\text{ext}} = 120$, $\tau_{\text{frag}} = 10$, window size $W_{\text{win}} = 200$, density threshold $\rho = 0.85$, max reasoning length $L_{\max}$.

**Output:** Generated response $y_i$, truncation indicator $h_i$, trigger position $k_i^{\text{trig}}$, redundant window length $W_i$.

Initialize empty SAM $\mathcal{S}$, last $\leftarrow 0$, history queue $\mathcal{Q} \leftarrow \varnothing$, active repeat count $C_{\text{frag}} \leftarrow 0$
$h_i \leftarrow 0$, $k_i^{\text{trig}} \leftarrow -1$, $W_i \leftarrow 0$
**for** $k = 1$ **to** $L_{\max}$ **do**
  Sample token $y_{i,k} \sim \pi_\theta(\cdot \mid x, y_{i,<k})$
  $\mathcal{S}$, last, $\ell_{i,k} \leftarrow$ Extension($\mathcal{S}$, last, $y_{i,k}, k$)
  $z_{i,k} \leftarrow \mathbf{1}(\ell_{i,k} > \tau_{\text{frag}})$
  $\mathcal{Q}$.push($z_{i,k}$), $C_{\text{frag}} \leftarrow C_{\text{frag}} + z_{i,k}$
  **if** $|\mathcal{Q}| > W_{\text{win}}$ **then**
    $C_{\text{frag}} \leftarrow C_{\text{frag}} - \mathcal{Q}$.pop()
  **end if**
  $is_ext \leftarrow \mathbf{1}(\ell_{i,k} \geq \tau_{\text{ext}})$
  $is_frag \leftarrow \mathbf{1}\left(k \geq W_{\text{win}} \wedge \frac{C_{\text{frag}}}{W_{\text{win}}} \geq \rho\right)$
  **if** $is_ext = 1 \vee is_frag = 1$ **then**
    $h_i \leftarrow 1$, $k_i^{\text{trig}} \leftarrow k$ {Record exact trigger position}
    **if** $is_ext = 1 \wedge is_frag = 1$ **then**
      $W_i \leftarrow \max(\ell_{i,k}, W_{\text{win}})$
    **else if** $is_ext = 1$ **then**
      $W_i \leftarrow \ell_{i,k}$
    **else**
      $W_i \leftarrow W_{\text{win}}$
    **end if**
    Force $y_{i,k+1} \leftarrow$ E○T **and break**
  **end if**
  **if** $y_{i,k} =$ E○T **then**
    **break**
  **end if**
**end for**
Generate final answer tokens after E○T using $\pi_\theta$
**return** $y_i, h_i, k_i^{\text{trig}}, W_i$

### I.3 Token-level Advantage Reshaping

After sampling a group of $G$ responses $\mathcal{Y} = \{y_1, \ldots, y_G\}$ for prompt $x$, we evaluate binary correctness $c_i \in \{0, 1\}$ and compute the standard GRPO advantage $\hat{A}_i$. Algorithm 3 outlines the token-level advantage reshaping procedure. For correct responses that triggered SAM termination ($h_i = 1$) in non-uniform groups ($|\mathcal{Y}_{\text{corr}}| < G$), relative down-weighting is applied exclusively to

token indices $t \in [s_i, k_i^{\text{trig}}]$, where $s_i = \max(1, k_i^{\text{trig}} - W_i + 1)$. This leaves all prefix reasoning tokens $t < s_i$ fully protected by the positive base advantage and efficiency bonuses.

**Algorithm 3** ROAST Token-level Advantage Reshaping

**Input:** Group responses $\mathcal{Y} = \{y_i\}_{i=1}^G$, correctness indicators $\{c_i\}_{i=1}^G$, base advantages $\{\hat{A}_i\}_{i=1}^G$, truncation indicators $\{h_i\}_{i=1}^G$, trigger positions $\{k_i^{\text{trig}}\}_{i=1}^G$, redundant lengths $\{W_i\}_{i=1}^G$, scaling hyperparameter $\alpha = 0.1$.

**Output:** Token-level reshaped advantages $\tilde{A}_{i,t}$ for all $i \in \{1, \ldots, G\}$ and $t \in \{1, \ldots, |y_i|\}$.

$\mathcal{Y}_{\text{corr}} \leftarrow \{y_i \in \mathcal{Y} \mid c_i = 1\}$
**if** $\mathcal{Y}_{\text{corr}} = \varnothing$ **then**
  **return** $\tilde{A}_{i,t} \leftarrow \hat{A}_i$ for all $i, t$
**end if**
$L_{\min} \leftarrow \min_{y_j \in \mathcal{Y}_{\text{corr}}} L_j^{\text{think}}, \quad S_{\min} \leftarrow \min_{y_j \in \mathcal{Y}_{\text{corr}}} S_j$
**for** $i = 1$ **to** $G$ **do**
  // Compute Sequence-level Efficiency Bonuses
  $B_{\text{len}}^{(i)} \leftarrow \alpha \cdot c_i \cdot \mathbf{1}(L_i^{\text{think}} = L_{\min})$
  $B_{\text{step}}^{(i)} \leftarrow \alpha \cdot c_i \cdot \mathbf{1}(S_i = S_{\min})$
  $B_i \leftarrow B_{\text{len}}^{(i)} + B_{\text{step}}^{(i)}$
  **for** $t = 1$ **to** $|y_i|$ **do**
    $\tilde{A}_{i,t} \leftarrow \hat{A}_i + B_i$
    // Apply Targeted Suffix Penalty exclusively to the trailing redundant tail
    **if** $c_i = 1$ **and** $h_i = 1$ **and** $|\mathcal{Y}_{\text{corr}}| < G$ **then**
      $s_i \leftarrow \max(1, k_i^{\text{trig}} - W_i + 1)$
      **if** $s_i \leq t \leq k_i^{\text{trig}}$ **then**
        $\tilde{A}_{i,t} \leftarrow \tilde{A}_{i,t} - \alpha$
      **end if**
    **end if**
  **end for**
**end for**
**return** $\{\tilde{A}_{i,t}\}_{i=1,t=1}^{G,|y_i|}$