ETR: Entropy Trend Reward for Efficient Chain-of-Thought Reasoning

Anonymous ACL submission

Abstract

Chain-of-thought (CoT) reasoning improves large language model performance on complex tasks, but often produces excessively long and inefficient reasoning traces. Existing methods shorten CoTs using length penalties or global entropy reduction, implicitly assuming that low uncertainty is desirable throughout reasoning. We show instead that reasoning efficiency is governed by the trajectory of uncertainty. CoTs with dominant downward entropy trends are substantially shorter. Motivated by this insight, we propose **E**ntropy **T**rend **R**eward (**ETR**), a trajectory-aware objective that encourages progressive uncertainty reduction while allowing limited local exploration. We integrate ETR into Group Relative Policy Optimization (GRPO) and evaluate it across multiple reasoning models and challenging benchmarks. ETR consistently achieves a superior accuracy–efficiency trade-off, improving DeepSeek-R1-Distill-7B by +9.9% accuracy while reducing CoT length by 67% across four benchmarks.

1 Introduction

Large language models (LLMs) increasingly rely on chain-of-thought (CoT) (Wei et al., 2022) reasoning to solve complex tasks by decomposing them into intermediate steps. By producing intermediate reasoning steps, LLMs can improve answer accuracy, interpretability, and robustness across complex tasks (Yue et al., 2025; Qu et al., 2025; Sui et al., 2025). However, CoT often comes with significant practical drawbacks. Models tend to generate unnecessarily long, repetitive, and sometimes self-contradictory reasoning sequences before reaching a conclusion (Ma et al., 2025; **?**). Such "overthinking" significantly increases inference latency.

Existing approaches to CoT reasoning can be broadly divided into training-free and training-based methods. Training-free methods rely on heuristic prompt design (Xu et al., 2025; Lee et al., 2025) or early stopping criteria (Yang et al., 2025) to reduce reasoning length. Training-based methods include supervised fine-tuning (Xia et al., 2025; Ye et al., 2025; Liu et al., 2024) and reinforcement learning (Luo et al., 2025; Aggarwal and Welleck, 2025), among which reinforcement learning–based approaches have attracted growing attention due to their stronger generalization. A representative example is length-based reward design (Luo et al., 2025; Aggarwal and Welleck, 2025), which encourages concise reasoning by penalizing long trajectories. However, such rewards are inherently content-blind, discouraging informative intermediate reasoning steps that may be important for maintaining final accuracy (Huang et al., 2025a).

Recent work has begun to link CoT length to predictive uncertainty, observing that CoTs with higher overall entropy tend to be longer and proposing to reduce global entropy via supervised fine-tuning (Li et al., 2025) or reinforcement learning

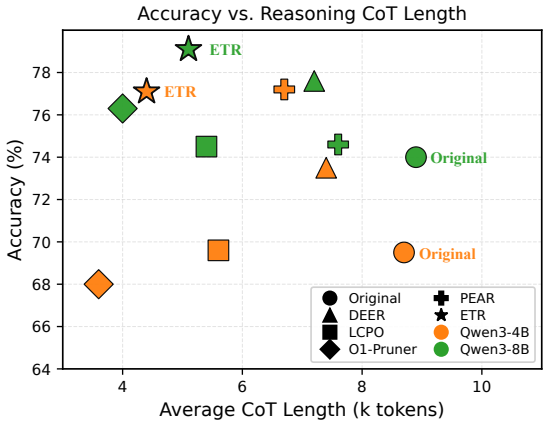

Figure 1: Accuracy versus average chain-of-thought (CoT) length across model sizes. Accuracy is averaged over four representative benchmarks. Compared methods exhibit different accuracy–efficiency trade-offs, with ETR achieving strong accuracy at substantially shorter reasoning lengths.

(Huang et al., 2025a). However, this perspective implicitly assumes that low uncertainty is desirable throughout the entire reasoning process. Such uniform entropy suppression is misaligned with how effective human reasoning unfolds. For example, early steps are naturally exploratory, involving multiple plausible directions, while later stages become increasingly constrained as the solution structure emerges. Consequently, an effective CoT is not one that maintains low entropy everywhere, but one whose uncertainty progressively reduces over time, allowing occasional local backtracking while preserving a coherent global descent. To elaborate, We provide empirical support for this trajectory-centric view by analyzing step-wise entropy patterns in generated CoTs. We find a clear relationship between the directionality of uncertainty evolution and reasoning length: CoTs dominated by downward entropy trends are substantially shorter, whereas those with frequent entropy increases tend to be much longer.

Motivated by this finding, we propose **E**ntropy **T**rend **R**eward (**ETR**) to optimize reasoning efficiency by explicitly shaping the trajectory of uncertainty during CoT generation. Rather than penalizing entropy uniformly, our approach encourages reasoning behaviors that achieve steady, coherent uncertainty reduction over time. This trajectory-aware objective provides dense feedback throughout the reasoning process, enabling the model to distinguish between productive exploration and inefficient uncertainty oscillation. We integrate this trajectory-aware uncertainty shaping into Group Relative Policy Optimization (GRPO) (Shao et al., 2024) that preserves answer correctness as a hard requirement, using efficiency signals only to compare and refine correct reasoning paths. An important consequence is an implicit, instance-adaptive stopping behavior: uncertainty collapses quickly for easy problems, yielding short CoTs, while harder problems are solved through gradual belief refinement without prolonged self-reflection or handcrafted length constraints.

In summary, this paper makes three contributions. First, we identify global uncertainty trends as a key determinant of chain-of-thought length and efficiency, shifting the focus from static entropy measures to trajectory-level reasoning dynamics. Second, we introduce a trajectory-aware uncertainty shaping approach that encourages progressive uncertainty contraction while allowing occasional local exploration. Third, we demonstrate that this approach yields shorter, more decisive reasoning traces while preserving answer quality, offering a principled path toward scalable and efficient chain-of-thought reasoning.

2 Related Work

2.1 Reinforcement Learning for LLM

Reinforcement learning has emerged as a powerful paradigm for unlocking the latent reasoning capabilities of Large Language Models (LLMs). Recent breakthroughs such as DeepSeek-R1 (Guo et al., 2025) and OpenAI-o1 (Jaech et al., 2024) demonstrate the important role of Reinforcement Learning from Verifiable Rewards (RLVR) in encouraging deep thinking and broad explorations. Despite the performance gains, the transition towards long CoT reasoning has introduced a critical challenge: the "overthinking" phenomenon. Recent studies (Yue et al., 2025; Sui et al., 2025) reveals that models often generate excessively verbose reasoning traces, leading to a substantial inference latency and increased computational overhead. Such observation underscores the urgency of developing methods to mitigate redundancy within reasoning paths without compromising accuracy.

2.2 Efficient Reasoning Model

Existing approaches to efficient CoT reasoning can be broadly grouped into three categories. (1) Heuristic-guided methods design specialized prompts (Xu et al., 2025; Lee et al., 2025) or stopping criterion (Yang et al., 2025) to steer models toward shorter reasoning paths without training, but their effectiveness often degrades for smaller models with limited controllability. (2) Variable-length CoT methods (Xia et al., 2025; Ye et al., 2025; Liu et al., 2024) rely on supervised fine-tuning with datasets containing reasoning traces of different lengths. However, they typically exhibit limited generalization beyond the training distribution. (3) length-based reward design (Luo et al., 2025; Aggarwal and Welleck, 2025) incorporates explicit length rewards into reinforcement learning, encouraging concise and correct reasoning. While effective at controlling the number of generated tokens, length-based methods are fundamentally content blind. they treat all tokens equally regardless of whether they contribute useful information.

To move beyond rigid length constraints, recent work dives into entropy (Shannon, 1948) that quantifies the model's uncertainty when generating to-

kens. There are training-free approaches such as CGRS (Huang et al., 2025b) leverages token-level entropy to suppress self-reflection tokens in order to mitigate response length. Other studies (Agarwal et al., 2025; Huang et al., 2025a) investigate the entropy-based rewards as a signal to regulate the model's internal uncertainty during the reasoning process. While existing entropy-based rewards focus on minimizing instantaneous uncertainty to streamline outputs, our entropy trend rewards monitor the evolution of entropy across the reasoning trace and introduce a more nuanced perspective.

3 Preliminary

We briefly review Group Relative Policy Optimization (GRPO) (Shao et al., 2024), which we adopt as the policy optimization algorithm in this work. GRPO is a policy gradient approach that evaluates model outputs based on their relative performance within a group of sampled responses, rather than relying on explicit value function estimation (Schulman et al., 2017). For a given input question q, GRPO samples a set of G candidate responses $\{o_1, o_2, \ldots, o_G\}$ from the current policy. Each response o_i is assigned a scalar return r_i. The return is defined as $r_i \triangleq R(q, o_i)$, where $R(q, o)$ denotes the reward assigned to response o for question q. And a group-normalized advantage is computed as $\hat{A}_i = \frac{r_i - \frac{1}{G}\sum_{j=1}^{G} r_j}{\sqrt{\frac{1}{G}\sum_{j=1}^{G}(r_j - \mu_r)^2}}$. This group-wise normalization emphasizes the relative ranking of candidate responses under the same input and reduces sensitivity to reward scale variations across different questions.

GRPO adopts a PPO-style clipped objective for policy optimization. Let $\pi_{\theta_{\text{old}}}$ denote the policy used to generate the sampled responses, and define the likelihood ratio $\tau_i(\theta) = \pi_\theta(o_i \mid q)/\pi_{\theta_{\text{old}}}(o_i \mid q)$. The optimization objective is given by

$$\mathcal{J}_{\text{GRPO}}(\theta) = \mathbb{E}\left[\frac{1}{G}\sum_{i=1}^{G} \min \mathcal{L}_{i,t}(\theta) - \beta\, \text{KL}(\pi_\theta)\right], \tag{1}$$

where the KL term regularizes policy updates and helps maintain stable training dynamics. $\mathcal{L}_{i,t}(\theta)$ denotes the token-level surrogate loss given by:

$$\mathcal{L}_{i,t}(\theta) = \Big(\tau_i(\theta)\hat{A}_i,\ \text{clip}\Big(\tau_i(\theta),\, 1-\epsilon,\, 1+\epsilon\Big)\hat{A}_i\Big). \tag{2}$$

By leveraging relative advantage estimation within each group, GRPO provides a robust optimization for sequence-level reward learning.

In this work, we focus on the design of the reward function $R(q, o)$ while keeping the GRPO optimization mechanism unchanged. We decompose the reward into a task correctness term and an efficiency-related term based on entropy:

$$R(q, o) = R_{\text{corr}}(q, o) \ \triangleright\ \lambda\, R_{\text{entropy}}(o), \tag{3}$$

where R_{corr} evaluates the correctness of the final answer, and R_{entropy} provides feedback on the reasoning trajectory. $\triangleright$ denotes that the entropy reward is combined with the correctness reward. In the following, we design R_{entropy} to encourage early termination once sufficient information has been acquired, leading to concise yet effective reasoning.

4 Motivation

Recent studies (Huang et al., 2025a; Li et al., 2025) have shown that higher overall entropy in a chain-of-thought (CoT) is strongly correlated with longer reasoning traces, motivating entropy reduction via supervised fine-tuning or reinforcement learning to improve efficiency. However, these approaches implicitly assume that low entropy is desirable at every reasoning step. High-entropy steps typically correspond to self-reflective reasoning, as illustrated in Figure 2 (left). Consequently, directly minimizing overall CoT entropy tends to discourage self-reflection altogether.

This assumption contradicts the nature of human reasoning. Early stages of complex reasoning are often exploratory and uncertain, while later stages become increasingly focused and deterministic. Efficient reasoning is therefore characterized not by uniformly low uncertainty, but by a progressive narrowing of plausible solutions.

To validate this intuition, we analyze step-wise entropy dynamics in generated CoTs. Specifically, we quantify whether uncertainty evolution along a CoT is dominated by entropy descent or ascent using the Spearman rank correlation (ρ) between the reasoning step index and the entropy at each step[1]. A smaller ρ indicates a stronger global entropy descent. Figure 2 (right) reveals a clear monotonic relationship between ρ and reasoning length. As ρ increases, the generated token length grows correspondingly, indicating that weaker global entropy descent is associated with longer reasoning trajectories, whereas CoTs with more pronounced entropy reduction (smaller ρ) tend to terminate earlier. This

[1]We show the detailed experimental setup in Appendix E.

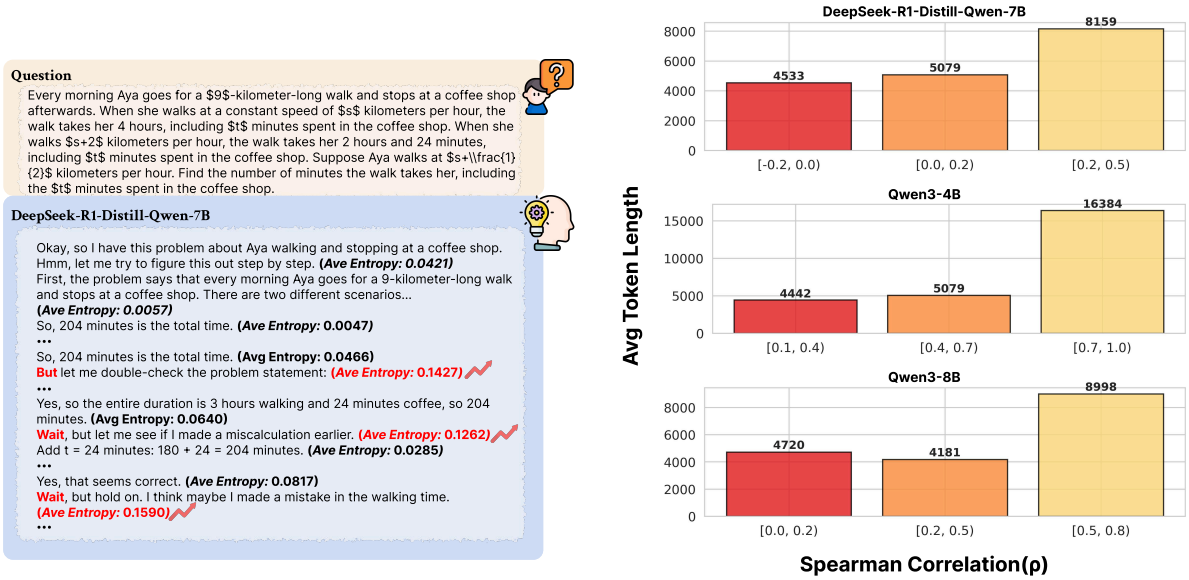

Figure 2: (Left) A sample response illustrating high step-wise entropy during self-reflection, marked by uncertainty cues (e.g., but, wait, etc.). (Right) Relationship between the Spearman rank correlation (ρ) between reasoning step index and step-wise entropy. Across models, higher positive correlations, reflecting persistent entropy growth throughout reasoning, are consistently associated with greater token redundancy.

empirical observation suggests that reasoning efficiency is closely tied to the global entropy trend.

5 Method

In this section, we formalize Entropy Trend Reward (ETR), a trajectory-level reward for efficient chain-of-thought (CoT) reasoning. ETR is designed to encourage progressive uncertainty reduction along a CoT while preserving correctness.

5.1 Entropy Trajectory of Chain-of-Thought

Given an input question q, the model generates a chain-of-thought response

$$o = \{C_1, C_2, \ldots, C_T\},$$

where each C_t corresponds to one sentence-level reasoning step.

For each step t, we define a scalar entropy measure

$$H_t = H(p_\theta(\cdot \mid C_{1:t})), \tag{4}$$

where $H(\cdot)$ denotes the Shannon entropy of the model's next-token predictive distribution. This provides a model-internal estimate of predictive uncertainty at each reasoning step.

We define the step-wise entropy change as:

$$\Delta_t = H_{t-1} - H_t, \qquad t = 2, \ldots, T. \tag{5}$$

A positive Δ_t indicates uncertainty reduction, while a negative value corresponds to increased uncertainty or exploratory divergence.

5.2 Momentum-Based Entropy Trend Reward

A natural baseline objective is to reward total entropy reduction. However, this reward telescopes:

$$R_{\text{naive}}(o) = \sum_{t=2}^{T} \Delta_t = H_1 - H_T, \tag{6}$$

and thus depends only on the initial and final entropy values. As a consequence, fundamentally different entropy trajectories, e.g., smooth monotonic descent versus repeated entropy spikes, receive identical rewards as long as they share the same endpoints.

To address this, we introduce a momentum-based accumulation of entropy changes. We define a latent trend variable S_t recursively:

$$S_t = \gamma S_{t-1} + \Delta_t, \qquad S_1 = 0, \tag{7}$$

where $\gamma \in (0, 1)$ is a momentum coefficient controlling temporal smoothing.

The entropy reward for a CoT is then defined as

$$R_{\text{entropy}}(o) = \sum_{t=2}^{T} S_t. \tag{8}$$

Unrolling the recurrence yields

$$R_{\text{entropy}}(o) = \sum_{t=2}^{T} \alpha_t \, \Delta_t, \tag{9}$$

$$\text{where } \alpha_t = \frac{1 - \gamma^{T-t+1}}{1 - \gamma}. \tag{10}$$

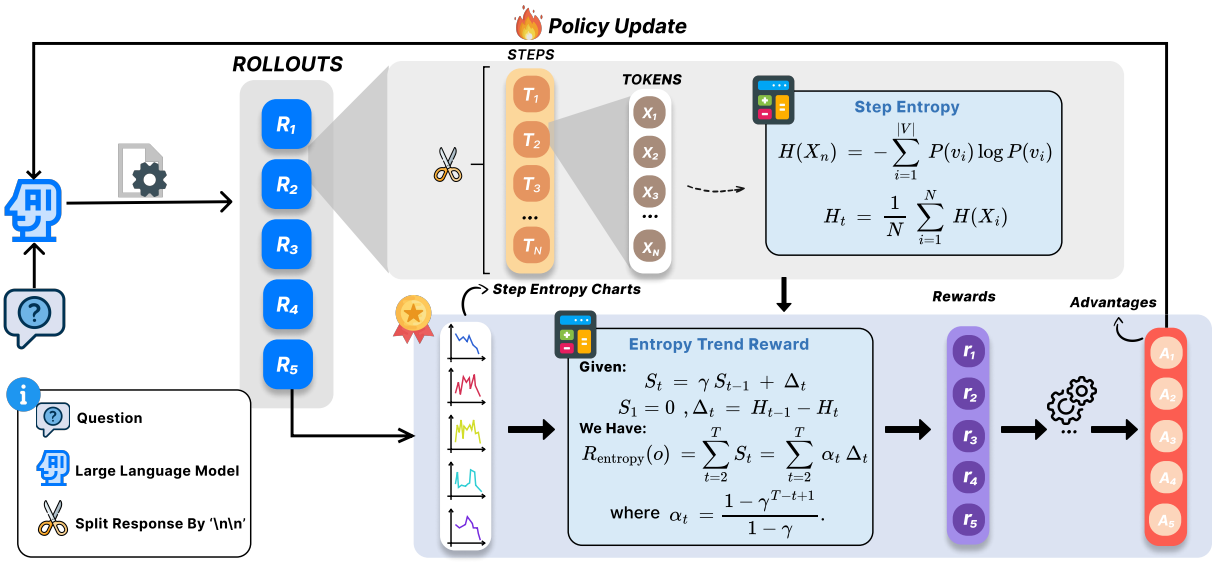

Figure 3: Overview of Entropy Trend Reward (ETR) in RL training. ETR computes step-wise entropy from generated rollouts, aggregates entropy changes with momentum to capture global entropy descent, and provides a reward signal, combined with correctness supervision, to guide policy updates toward more convergent reasoning.

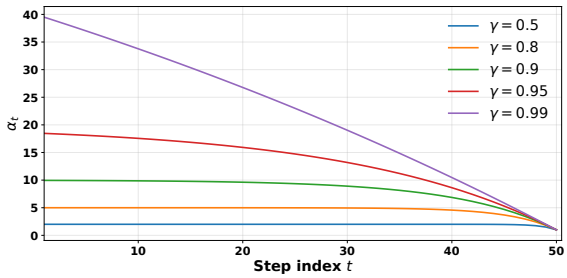

Figure 4: Evolution of the cumulative momentum weights α_t under different momentum coefficients γ.

Unlike the naive entropy trend reward, which ignores intermediate reasoning structure, the momentum-based formulation assigns a gradient signal to *every* step in the trajectory: Thus, each entropy drop (or increase) influences the reward, providing fine-grained feedback that guides the policy toward coherent and efficient reasoning.

A key property of α_t is that it is strictly decreasing in t. To illustrate, we show an example plot of α_t in Figure 4. Because α_t decreases as t increases, reductions in uncertainty that occur earlier in the reasoning trajectory are rewarded more than those occurring later. This encourages the model to converge its belief state rapidly rather than postponing critical deductions to late steps. In our experiments, we set $\gamma = 0.9$, with a detailed analysis of its selection deferred to Appendix F.1.

5.3 Instance-Adaptive Termination via Entropy Trend Shaping

Our reward shaping implicitly encourages instance-adaptive termination without imposing any explicit length constraint. Recall that ETR maintains a momentum-aggregated trend state $S_t = \gamma S_{t-1} + \Delta_t$ with $\Delta_t = H_{t-1} - H_t$. Since the entropy reward accumulates $\sum_{t=2}^{T} S_t$, generating additional steps is beneficial only when the next update keeps the trend state positive, i.e., when the next step continues to reduce uncertainty on average. In contrast, steps that increase entropy ($\Delta_t < 0$) decrease S_t and are repeatedly penalized, which suppresses oscillatory self-reflection loops.

This mechanism yields instance-adaptive behavior. For **easy instances**, entropy typically drops early, leading to large positive Δ_t and a rapid rise of S_t; after the model becomes confident, further steps rarely provide additional entropy descent and are therefore disfavored, resulting in short CoTs. For **hard instances**, entropy may fluctuate; ETR discourages large entropy ascents and favors trajectories with steady, incremental uncertainty reduction, allocating more reasoning steps only when they contribute to convergence. Compared with global entropy penalties (Huang et al., 2025a; Li et al., 2025), ETR is trajectory-aware: it tolerates limited local exploration while enforcing a coherent global descent.

5.4 Integration with GRPO

We incorporate the entropy trend reward into GRPO by decomposing the overall reward as

$$R(q, o) = R_{\text{corr}}(q, o) \triangleright \lambda R_{\text{entropy}}(o), \quad (11)$$

where R_{corr} evaluates the correctness of the final answer. The operator $\triangleright$ now represents that the entropy reward is applied only as a shaping term for correct trajectories. And $\lambda > 0$ controls the influence of entropy-based efficiency shaping.

In practice, we adopt a two-stage reward structure:

$$R(q, o) = \begin{cases} -1, & \text{if incorrect,} \\ 1 + \lambda R_{\text{entropy}}(o), & \text{if correct.} \end{cases} \quad (12)$$

An overview of Entropy Trend Reward (ETR) in RL training is illustrated in Figure 3. And the detailed procedure is summarized in Appendix A, This design ensures that correctness is a hard constraint, while entropy shaping only affects the correct reasoning trajectories. Within each rollout group, GRPO's relative advantage normalization further emphasizes comparative efficiency among valid CoTs generated for the same input.

6 Experiments and Analysis

6.1 Experimental Setup

Dataset. Our training data consists of 7,000 problems randomly sampled from DeepMath-103K (He et al., 2025), covering difficulty levels from 5 to 10 only. We additionally perform a verification step to ensure that the sampled training data has no overlap with the held-out evaluation benchmarks.

Benchmarks. To comprehensively evaluate the reasoning ability of our model, we adopt both mathematical and general reasoning benchmarks. For math-specific evaluation, we select AIME24 (MAA., 2024), AMC23 (MAA., 2023), and MATH500 (Hendrycks et al., 2024), which emphasize high-school to competition-level mathematical problem solving. For broader knowledge-intensive reasoning, we adopt GPQA Diamond (Rein et al., 2024), a subset of the GPQA benchmark that focuses on expert-level question answering.

Reasoning Models. We evaluate our approach using three reasoning models: DeepSeek-R1-Distill-Qwen-7B[2], Qwen3-4B[3], Qwen3-8B[4]. These models generally exhibit strong reasoning abilities but with excessive output tokens.

Baselines. We compare our method with several representative approaches for efficient reasoning, including both training-free and RL–based methods. Training-free methods such as DEER (Yang et al., 2025) and NoThink (Ma et al., 2025) reduce response length through prompt or early stopping criteria design. RL–based approaches, including LCPO (Aggarwal and Welleck, 2025), O1-Pruner (Luo et al., 2025), and PEAR (Huang et al., 2025a), address this problem by designing reward functions that explicitly depend on sequence length or entropy.

Evaluation and Metrics. In this work, we evaluate our method using accuracy (Acc), response length (Len), compression rate (CR), and the Accuracy–Efficiency Trade-off Score (AES) (Luo et al., 2025). CR is defined as the ratio of the average response length to that of the original model, where lower values indicate higher compression. AES measures the trade-off between efficiency gains and accuracy changes relative to a baseline.

$$\text{AES} = \underbrace{\frac{L_{\text{base}} - L_{\text{model}}}{L_{\text{base}}}}_{\text{relative length reduction}} + \varsigma \underbrace{\frac{A_{\text{model}} - A_{\text{base}}}{A_{\text{base}}}}_{\text{relative accuracy change}},$$

where L and A denote response length and accuracy, and ς balances accuracy improvement and efficiency gains. Following Luo et al. (2025), we set $\varsigma = 5$ to prioritize accuracy over length.

Training Details We conduct training using the open-source VeRL framework (Sheng et al., 2025) on 8 NVIDIA H-100 GPUs. To reduce memory footprint and improve training efficiency, we employ Low-Rank Adaptation (LoRA) (Hu et al., 2022). Unless otherwise specified, we use a batch size of 32 and a learning rate of 1×10^{-5}. The maximum response length is set to $16,384$, with 5 rollouts per sample.

6.2 Main Results

Table 1 reports the accuracy–efficiency trade-off of different methods across multiple reasoning benchmarks and model configurations. We highlight two consistent observations.

[2] https://huggingface.co/deepseek-ai/DeepSeek-R1-Distill-Qwen-7B

[3] https://huggingface.co/Qwen/Qwen3-4B

[4] https://huggingface.co/Qwen/Qwen3-8B

Method	AMC23			AIME24			MATH500			GPQA-D			Overall		
	Acc↑	Len↓	CR↓	Acc↑	Len↓	CR↓	Acc↑	Len↓	CR↓	Acc↑	Len↓	CR↓	Acc↑	Len↓	AES↑
DeepSeek-R1-Distill-7B															
Original	80.0	6.6k	100%	43.3	11.8k	100%	85.0	4.2k	100%	24.2	11.3k	100%	58.1	8.5k	0.00
DEER	85.0	4.9k	74.2%	50.0	9.9k	83.9%	85.8	2.3k	54.8%	22.7	7.6k	67.3%	60.9	6.2k	0.51
NoThink	77.5	3.8k	57.6%	56.7	8.3k	70.3%	81.4	1.7k	40.5%	22.2	2.0k	17.7%	59.5	4.0k	0.65
LCPO	87.5	3.5k	53.0%	50.0	6.8k	57.8%	85.4	2.2k	52.6%	11.61	2.8k	24.6%	58.6	3.8k	0.60
O1-Pruner	92.5	3.2k	48.4%	46.7	7.8k	66.1%	89.0	2.1k	50.0%	39.4	6.2k	54.9%	66.9	4.8k	1.18
PEAR	92.5	4.4k	66.04%	60.0	8.5k	72.2%	90.2	2.5k	59.9%	36.9	5.2k	45.9%	69.8	5.1k	1.41
ETR	**87.5**	**2.4k**	**36.4%**	**56.7**	**4.6k**	**39.0%**	**90.6**	**1.5k**	**35.7%**	**37.3**	**2.5k**	**22.1%**	**68.0**	**2.8k**	**1.53**
Qwen3-4B															
Original	90.0	7.6k	100%	53.3	11.7k	100%	90.6	5.0k	100%	43.9	10.4k	100%	69.5	8.7k	0.00
DEER	92.5	6.1k	80.3%	56.7	11.5k	98.3%	92.2	3.7k	74.0%	52.5	8.2k	78.8%	73.5	7.4k	0.44
NoThink	82.5	2.4k	31.6%	33.3	8.0k	68.4%	85.6	1.7k	34.0%	23.2	4.7k	45.2%	56.2	4.2k	-0.44
LCPO	90.0	6.0k	78.9%	50.0	8.2k	70.6%	90.8	4.5k	89.2%	47.5	3.8k	36.3%	69.6	5.6k	0.36
O1-Pruner	85.0	2.6k	34.2%	50.0	6.9k	59.0%	90.6	1.8k	36%	50.0	3.3k	31.7%	68.0	3.6k	0.54
PEAR	92.5	6.0k	34.2%	70.0	9.9k	59.0%	91.8	3.8k	36%	54.54	7.1k	31.7%	77.2	6.7k	0.79
ETR	**90.0**	**4.0k**	**52.6%**	**73.3**	**7.5k**	**64.1%**	**91.4**	**2.1k**	**42.0%**	**53.5**	**4.1k**	**39.4%**	**77.1**	**4.4k**	**1.03**
Qwen3-8B															
Original	90.0	8.0k	100%	63.3	12.2k	100%	90.6	5.4k	100%	52.0	9.9k	100%	74.0	8.9k	0.00
DEER	92.5	6.4k	80.0%	63.3	10.3k	84.4%	93.0	3.1k	57.4%	61.6	9.0k	90.9%	77.6	7.2k	0.43
NoThink	67.5	3.4k	41.3%	40.0	7.0k	57.4%	86.4	1.5k	27.8%	26.8	4.4k	44.4%	55.2	4.1k	-0.73
LCPO	90.0	5.7k	41.3%	60.0	7.0k	57.4%	93.6	4.8k	27.8%	54.6	4.0k	44.4%	74.5	5.4k	0.43
O1-Pruner	90.0	3.3k	41.3%	66.7	6.6k	54.1%	92.2	1.9k	35.2%	56.1	4.1k	41.4%	76.3	4.0k	0.70
PEAR	87.5	6.8k	84.7%	63.3	11.0k	90.1%	92.4	4.5k	83.5%	55.1	8.2k	83.3%	74.6	7.6k	0.18
ETR	**92.5**	**4.2k**	**53.8%**	**73.3**	**8.6k**	**75.4%**	**93.6**	**2.4k**	**44.4%**	**57.1**	**5.2k**	**52.5%**	**79.1**	**5.1k**	**0.77**

Table 1: Evaluation results on four benchmarks using greedy decoding (pass@1). **Acc** denotes accuracy, **Len** denotes token length, and **CR** denotes compression rate. ↑ and ↓ indicate higher- and lower-is-better, respectively. **AES** is a unified score that jointly accounts for accuracy and efficiency.

First, ETR performs robustly across different model families and sizes. Across both DeepSeek-R1-Distill and Qwen3 model families, spanning sizes from 4B to 8B, our method consistently maintains strong accuracy while substantially reducing reasoning length, achieving the highest AES scores in all settings. This indicates that ETR does not rely on a specific model architecture or scale, but generalizes well across comparable model sizes and across different model families.

Second, and more importantly, ETR achieves the best balance between accuracy and reasoning length among all compared methods. Existing training-free approaches (e.g., DEER and NoThink) struggle to jointly optimize these two objectives. DEER largely preserves accuracy but fails to substantially reduce output length, while NoThink aggressively truncates reasoning traces (e.g., on Qwen3-4B, reducing output from 8.7k to 4.2k tokens), but at the cost of severe accuracy degradation (69.5% → 56.2%), resulting in negative AES scores. Methods based on explicit length penalties (e.g., LCPO and O1-Pruner) shorten outputs more effectively than training-free approaches, but this is often accompanied by significant accuracy loss. Entropy-based reinforcement learning methods (e.g., PEAR) tend to preserve accuracy, yet still produce long reasoning chains due to their reliance on global entropy signals.

In contrast, ETR consistently yields the highest AES across all evaluated settings, demonstrating a more principled control of the reasoning process that avoids both underthinking and overthinking.

6.3 Empirical Analysis

Unless otherwise specified, we use DeepSeek-R1-Distill-Qwen-7B as the base model for all empirical analyses in this section.

6.3.1 Analysis on Entropy Rewards.

We conduct an ablation study on entropy-based reward designs to isolate the effects of entropy magnitude, temporal aggregation, and correctness supervision. Results are summarized in Table 2.

Absolute entropy objectives. We first examine two extreme strategies that directly optimize the average predictive entropy over all CoT tokens. Entropy minimization (Min. H), as adopted in prior work (Li et al., 2025; Huang et al., 2025a), yields short CoTs but does not improve accuracy and achieves a lower AES score than our method (1.06 vs. 1.53). In contrast, entropy maximization (Max. H) collapses reasoning, producing maximal-length generations with near-zero accuracy. These results show that optimizing absolute entropy magnitude alone is suboptimal for efficient reasoning.

Effect of momentum. Removing temporal aggregation (No γ) corresponds to the naive variant discussed at the beginning of Section 5.2, which relies only on entropy differences between first and last sentences. As a result, it fails to capture

Reward	AMC23		AIME24		MATH500		GPQA-D		AES↑
	Acc ↑	Len ↓	Acc ↑	Len ↓	Acc ↑	Len ↓	Acc ↑	Len ↓	
Original	80.0	6.6k	43.3	11.8k	85.0	4.2k	24.2	11.3k	0.00
Min. H	80.0	2.1k	43.3	5.1k	88.2	1.3k	38.3	2.1k	1.06
Max. H	10.0	15.1k	0.0	16.4k	9.0	15.3k	1.5	16.0k	-5.4
No γ	87.5	4.9k	46.67	10.0k	87.8	3.6k	31.8	10.0k	0.61
No R_{corr}	65.0	1.2k	23.3	1.4k	78.6	0.7k	29.8	0.7k	0.11
Ours	**87.5**	**2.4k**	**56.7**	**4.6k**	**90.6**	**1.5k**	**37.4**	**2.5k**	**1.53**

Table 2: Ablation Study on Entropy-based Reward Designs. We compare our momentum-based reward against extreme constraints and simplified variants across four benchmarks. Our method demonstrates a superior balance between accuracy and efficiency.

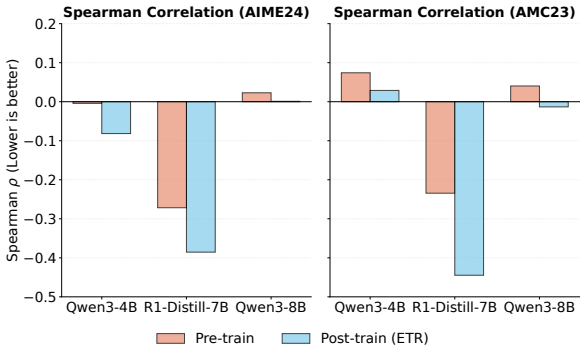

Figure 5: Spearman's rank correlation coefficient between reasoning steps and step entropy. The shift toward negative values indicates a more convergent reasoning trajectory after ETR training.

the overall entropy trend along the CoT. While accuracy remains reasonable because of correctness reward, it cannot reduce CoT length, highlighting the necessity of momentum-based aggregation.

Effect of correctness supervision. Removing the correctness reward (No R_{corr}) leads to short CoTs but causes a substantial accuracy drop (e.g., 65.0% on AMC23), indicating that entropy trends alone are insufficient for ensuring correctness. Overall, combining entropy trends with momentum-based entropy trend reward and correctness supervision yields the best accuracy–efficiency trade-off across all benchmarks.

6.3.3 Analysis on Entropy Trend.

To assess whether Entropy Trend Reward (ETR) promotes convergent reasoning, we analyze entropy dynamics along the Chain-of-Thought (CoT). We compute the Spearman rank correlation (ρ) between the reasoning step index and the entropy at each step, where a more negative ρ indicates a stronger global entropy descent.

As shown in Figure 5, across model scales, ETR consistently shifts the correlation from positive or near-zero to negative values. This sign reversal

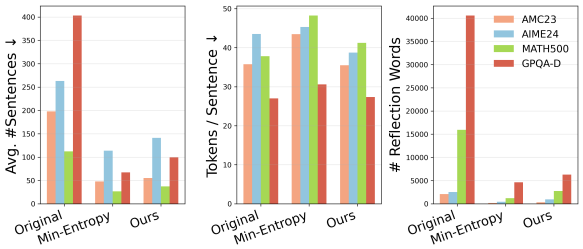

Figure 6: ETR reduces CoT length primarily by maintaining low per-step verbosity, rather than by aggressively suppressing reasoning steps or reflective processes, in contrast to entropy minimization.

provides empirical evidence that ETR effectively regularizes the reasoning path, ensuring that each step contributes to narrowing the solution space rather than accumulating logical ambiguity.

6.3.5 Behavioral View of CoT Reduction.

Figure 6 decomposes CoT length into (i) the number of sentence-level steps, (ii) tokens per step, and (iii) the number of self-reflection markers[5]. Direct entropy minimization shortens CoTs primarily by suppressing exploration, yielding fewer steps and dramatically fewer reflection markers, but at the cost of more verbose reasoning steps (higher tokens per sentence). In contrast, ETR preserves a moderate degree of self-verification and multi-step reasoning while maintaining low per-step verbosity, aligning with our objective of permitting limited local exploration under a globally convergent uncertainty trajectory. This difference explains why ETR achieves a superior accuracy–efficiency trade-off: it discourages oscillatory overthinking without enforcing uniformly low uncertainty at every step.

7 Limitation.

Owing to computational constraints, our experiments are limited to models up to 8B parameters with LoRA-based training. We plan to scale our method to larger models in future work.

8 Conclusion

In this paper, we show that chain-of-thought efficiency is governed by the global trend of predictive uncertainty rather than its absolute magnitude. Based on this insight, we propose Entropy Trend Reward (ETR), which encourages progressive uncertainty reduction while enforcing correctness as a hard constraint. Extensive experiments demonstrate that ETR consistently improves the accuracy–efficiency trade-off.

[5]Self-reflection words: *wait, alternatively, hmm, but, however, alternative, another, check, double-check, oh, maybe*.

References

Shivam Agarwal, Zimin Zhang, Lifan Yuan, Jiawei Han, and Hao Peng. 2025. The unreasonable effectiveness of entropy minimization in llm reasoning. *arXiv preprint arXiv:2505.15134*.

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

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

Zhiwei He, Tian Liang, Jiahao Xu, Qiuzhi Liu, Xingyu Chen, Yue Wang, Linfeng Song, Dian Yu, Zhenwen Liang, Wenxuan Wang, and 1 others. 2025. Deepmath-103k: A large-scale, challenging, decontaminated, and verifiable mathematical dataset for advancing reasoning. *arXiv preprint arXiv:2504.11456*.

Dan Hendrycks, Collin Burns, Saurav Kadavath, Akul Arora, Steven Basart, Eric Tang, Dawn Song, and Jacob Steinhardt. 2024. Measuring mathematical problem solving with the math dataset, 2021. *URL https://arxiv. org/abs/2103.03874*, 2.

Edward J Hu, Phillip Wallis, Zeyuan Allen-Zhu, Yuanzhi Li, Shean Wang, Lu Wang, Weizhu Chen, and 1 others. 2022. Lora: Low-rank adaptation of large language models. In *International Conference on Learning Representations*.

Chen Huang, Wei Lu, and Wenxuan Zhang. 2025a. Pear: Phase entropy aware reward for efficient reasoning. *arXiv preprint arXiv:2510.08026*.

Jiameng Huang, Baijiong Lin, Guhao Feng, Jierun Chen, Di He, and Lu Hou. 2025b. Efficient reasoning for large reasoning language models via certainty-guided reflection suppression. *arXiv preprint arXiv:2508.05337*.

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

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

Zeju Li, Jianyuan Zhong, Ziyang Zheng, Xiangyu Wen, Zhijian Xu, Yingying Cheng, Fan Zhang, and Qiang Xu. 2025. Compressing chain-of-thought in llms via step entropy. *arXiv preprint arXiv:2508.03346*.

Hunter Lightman, Vineet Kosaraju, Yuri Burda, Harrison Edwards, Bowen Baker, Teddy Lee, Jan Leike, John Schulman, Ilya Sutskever, and Karl Cobbe. 2023. Let's verify step by step. In *The Twelfth International Conference on Learning Representations*.

Tengxiao Liu, Qipeng Guo, Xiangkun Hu, Cheng Jiayang, Yue Zhang, Xipeng Qiu, and Zheng Zhang. 2024. Can language models learn to skip steps? *Advances in Neural Information Processing Systems*, 37:45359–45385.

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

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

MAA. 2023. American mathematics competitions.

MAA. 2024. American invitational mathematics examination - aime.

Xiaoye Qu, Yafu Li, Zhaochen Su, Weigao Sun, Jianhao Yan, Dongrui Liu, Ganqu Cui, Daizong Liu, Shuxian Liang, Junxian He, and 1 others. 2025. A survey of efficient reasoning for large reasoning models: Language, multimodality, and beyond. *arXiv preprint arXiv:2503.21614*.

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

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

Claude E Shannon. 1948. A mathematical theory of communication. *The Bell system technical journal*, 27(3):379–423.

Zhihong Shao, Peiyi Wang, Qihao Zhu, Runxin Xu, Junxiao Song, Xiao Bi, Haowei Zhang, Mingchuan Zhang, YK Li, Yang Wu, and 1 others. 2024. Deepseekmath: Pushing the limits of mathematical reasoning in open language models. *arXiv preprint arXiv:2402.03300*.

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

Yang Sui, Yu-Neng Chuang, Guanchu Wang, Jiamu Zhang, Tianyi Zhang, Jiayi Yuan, Hongyi Liu, Andrew Wen, Shaochen Zhong, Na Zou, and 1 others. 2025. Stop overthinking: A survey on efficient reasoning for large language models. *arXiv preprint arXiv:2503.16419*.

Jason Wei, Xuezhi Wang, Dale Schuurmans, Maarten Bosma, Fei Xia, Ed Chi, Quoc V Le, Denny Zhou, and 1 others. 2022. Chain-of-thought prompting elicits reasoning in large language models. *Advances in neural information processing systems*, 35:24824–24837.

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

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

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

Yixin Ye, Zhen Huang, Yang Xiao, Ethan Chern, Shijie Xia, and Pengfei Liu. 2025. Limo: Less is more for reasoning. *arXiv preprint arXiv:2502.03387*.

Linan Yue, Yichao Du, Yizhi Wang, Weibo Gao, Fangzhou Yao, Li Wang, Ye Liu, Ziyu Xu, Qi Liu, Shimin Di, and 1 others. 2025. Don't overthink it: A survey of efficient r1-style large reasoning models. *arXiv preprint arXiv:2508.02120*.

Lianmin Zheng, Liangsheng Yin, Zhiqiang Xie, Chuyue Livia Sun, Jeff Huang, Cody Hao Yu, Shiyi Cao, Christos Kozyrakis, Ion Stoica, Joseph E Gonzalez, and 1 others. 2024. Sglang: Efficient execution of structured language model programs. *Advances in neural information processing systems*, 37:62557–62583.

A Algorithmic Details of Entropy Trend Reward

Algorithm 1 presents the detailed computation of Entropy Trend Reward (ETR) for a generated chain-of-thought. Given a sentence-level CoT response, the algorithm first enforces correctness as a hard constraint by assigning a negative reward to incorrect outputs. For correct responses, it computes step-wise predictive entropy, aggregates entropy changes using a momentum-based recurrence, and accumulates a trajectory-level entropy reward. The final reward combines correctness supervision with entropy-based efficiency shaping, providing dense feedback that encourages progressive uncertainty reduction during reasoning.

Algorithm 1: Entropy Trend Reward (ETR)

Input: CoT response $o = \{C_t\}_{t=1}^{T}$; ground-truth y^*; momentum γ; weight λ.
Output: Reward $R(q, o)$.

```
ŷ ← ExtractAnswer(o)
if ŷ ≠ y* then
    return −1
H_1 ← H(p_θ(· | C_{1:1}))
S ← 0; R_entropy ← 0
for t ← 2 to T do
    H_t ← H(p_θ(· | C_{1:t}))
    Δ_t ← H_{t−1} − H_t
    S ← γS + Δ_t
    R_entropy ← R_entropy + S
return 1 + λR_entropy
```

B Derivation of Equation 8.

Recall that the step-wise entropy change is

$$\Delta_t = H_{t-1} - H_t, \qquad t = 2, \dots, T, \tag{13}$$

and the momentum recursion is

$$S_t = \gamma S_{t-1} + \Delta_t, \qquad S_1 = 0, \tag{14}$$

with $\gamma \in (0, 1)$. The entropy reward is defined as

$$R_{\text{entropy}}(o) = \sum_{t=2}^{T} S_t. \tag{15}$$

Step 1: Unroll the recursion. For any $t \geq 2$, repeatedly substituting Eq. (14) gives

$$\begin{aligned} S_t &= \gamma S_{t-1} + \Delta_t \\ &= \gamma(\gamma S_{t-2} + \Delta_{t-1}) + \Delta_t \\ &= \gamma^2 S_{t-2} + \gamma \Delta_{t-1} + \Delta_t \\ &\vdots \\ &= \gamma^{t-2} S_2 + \sum_{k=3}^{t} \gamma^{t-k} \Delta_k + \Delta_t. \end{aligned} \tag{16}$$

Since $S_2 = \gamma S_1 + \Delta_2 = \Delta_2$ and $S_1 = 0$, Eq. (16) simplifies to

$$S_t = \sum_{k=2}^{t} \gamma^{\,t-k}\, \Delta_k. \tag{17}$$

Step 2: Substitute into the reward and swap summations. Plugging Eq. (17) into Eq. (15) yields

$$\begin{aligned} R_{\text{entropy}}(o) &= \sum_{t=2}^{T} \sum_{k=2}^{t} \gamma^{\,t-k}\, \Delta_k \\ &= \sum_{k=2}^{T} \Delta_k \sum_{t=k}^{T} \gamma^{\,t-k}, \end{aligned} \tag{18}$$

where the last line follows by exchanging the order of summation.

Step 3: Evaluate the inner geometric series. Let $m = t - k$, so that $m = 0, \dots, T - k$. Then

$$\sum_{t=k}^{T} \gamma^{\,t-k} = \sum_{m=0}^{T-k} \gamma^{\,m}. \tag{19}$$

For $\gamma \neq 1$, this is a finite geometric series:

$$\sum_{m=0}^{T-k} \gamma^{\,m} = \frac{1 - \gamma^{\,T-k+1}}{1 - \gamma}. \tag{20}$$

Substituting Eq. (20) into Eq. (18) gives

$$R_{\text{entropy}}(o) = \sum_{k=2}^{T} \alpha_k\, \Delta_k, \qquad \alpha_k = \frac{1 - \gamma^{\,T-k+1}}{1 - \gamma}. \tag{21}$$

Renaming the index $k \mapsto t$ yields Eq. (10) in the main text:

$$R_{\text{entropy}}(o) = \sum_{t=2}^{T} \alpha_t\, \Delta_t, \qquad \alpha_t = \frac{1 - \gamma^{\,T-t+1}}{1 - \gamma}. \tag{22}$$

Hyperparameter	LCPO	O1-Pruner	PEAR	ETR (Ours)
LoRA Rank	-	-	-	32
LoRA Alpha	-	-	-	64
Actor learning rate	$1e-6$	$1e-6$	$1e-6$	$1e-5$
train_batch_size	64	32	32	32
mini_batch_size	64	16	16	16
micro_batch_size	-	2	2	2
Training step	1400	207	207	207
Max response length	4096	16384	16384	16384
Num of rollouts	8	5	5	5
Rollout temp (τ)	0.6	1.0	1.0	1.0
KL penalty (β)	$1e-3$	$1e-3$	$1e-3$	$5e-3$
Entropy Coef	-	-	-	$1e-4$
Advantage clip (ϵ)			0.2	

Table 3: Training configs for various methods.

C Training and Evaluation Details for ETR

Hyperparameter	ETR (Ours)
LoRA Rank	32
LoRA Alpha	64
LoRA Target Modules	all-linear
Actor learning rate	$1e-5$
train_batch_size	32
mini_batch_size	16
micro_batch_size	2
param_offload	True
optimizer_offload	True
log_prob_micro_batch_size	8
enable_chunked_prefill	True
Training step	207
Max response length	16384
Num of rollouts	5
Rollout temp (τ)	1.0
KL penalty (β)	$5e-3$
Entropy Coef	$1e-4$
Advantage clip (ϵ)	0.2
epochs	1

Table 4: Detaied Configurations for ETR

During training and evaluation, we insert the below system prompt before the question:

System Prompt

"Please reason step by step, and put your final answer within `\boxed{}`"

Our evaluation methodology utilizes the SGlang framework (Zheng et al., 2024) to conduct high-throughput inference across multiple benchmarks, including MATH500 (Hendrycks et al., 2024), GPQA Diamond (Rein et al., 2024), AIME24 (MAA., 2024), and AMC23 (MAA., 2023). We configured the generation process with greedy decoding, a max response length of 16,384 and a sample size of 1, ensuring the model has sufficient space to develop complex reasoning paths while maintaining deterministic outputs for accuracy tracking. To extract the final answer from the model's response, we leveraged implemented functions in VeRL library (Sheng et al., 2025). This process specifically extracts the final answer in `\boxed{}`.

For the final assessment of correctness, we integrated the grading logic from the OpenAI PRM800K dataset grader (Lightman et al., 2023). We found this approach to be significantly reliable and robust, as it effectively handles differing mathematical formulations and varied formatting styles. Throughout the evaluation, we recorded both the accuracy and the average response length for each dataset, allowing us to analyze the relationship between reasoning efficiency and model performance.

D Training Details For Baseline Methods

We selected five baseline methods for comparison: **DEER** (Yang et al., 2025), **NoThink** (Ma et al., 2025), **LCPO** (Aggarwal and Welleck, 2025), **O1-Pruner** (Luo et al., 2025) and **PEAR** (Huang et al., 2025a). Among these, DEER and NoThink are

training-free approaches, while the remaining methods are trained using the DeepMath-103K dataset (He et al., 2025). The training details can be found below. We report all training hyperparameters in Table 3.

For DEER (Yang et al., 2025), we use the official implementation and scripts provided by the authors[6]. To ensure a fair and consistent comparison, we use the provided scripts and set think ratio to 0.9 and max generated tokens to 16384 to align with our training configurations.

For NoThink (Ma et al., 2025), we followed the paper to bypass the explicit reasoning process through prompting using "Okay, I think I have finished thinking.".

For LCPO (Aggarwal and Welleck, 2025), we use the official implementation provided by the authors[7] and follow their L1-Exact setup. Training is conducted using GRPO with a specific length constraint embedded in the prompt, instructing the model to "Think for N tokens". The actor model is optimized with a learning rate of 1×10^{-6} and a global training batch size of 64. To ensure stability during policy evolution, we set the KL penalty coefficient (β) to 0.001. During the rollout phase, the model generates 8 independent reasoning trajectories for each prompt with a sampling temperature of 0.6. The maximum response length is configured to 4,096 tokens, adhering to the default parameters provided in the official repository

For O1-Pruner, it has a KL penalty coefficient of 0.001 and a learning rate of 1×10^{-6}. Rollout number is fixed at 5 and maximum response length is set to 16384 align with our training configurations.

For PEAR, we followed the official codebase provided by the authors[8]. We set the batch size to 32 and the learning rate to 1×10^{-6}. The KL penalty coefficient is set to 0.001. The maximum response length is set to 16384 to align with our training configurations.

[6] https://github.com/iie-ycx/DEER
[7] https://github.com/cmu-l3/l1
[8] https://github.com/iNLP-Lab/PEAR

E Experimental Details for the Motivation Section

This appendix provides additional details on the analysis of step-wise entropy trajectories used to motivate our method.

Given a generated CoT, we segment it into sentence-level steps $\{C_t\}_{t=1}^{T}$ and compute an entropy value H_t for each step. We get a list of step-wise entropy

$$\mathbf{H} = [H_1, H_2, \ldots, H_T]$$

Let $\mathcal{T}$ denote the list of the reasoning steps indices in ascending order, which corresponds to the entropy vector $\mathbf{H} = [H_1, H_2, \ldots, H_T]$.

$$\mathcal{T} = [1, 2, \ldots, T]$$

Then we can compute the Spearman correlation coefficient, ρ, is defined as the Pearson correlation coefficient between the rank variables of $\mathcal{T}$ and $\mathbf{H}$:

$$\rho = \frac{\mathrm{cov}(\mathrm{rank}(\mathcal{T}), \mathrm{rank}(\mathbf{H}))}{\sigma_{\mathrm{rank}(\mathcal{T})}\sigma_{\mathrm{rank}(\mathbf{H})}} \tag{23}$$

Where:

- $\mathrm{rank}(\mathcal{T})$ and $\mathrm{rank}(\mathbf{H})$ are the vectors containing the ranks of the original observations.
- $\mathrm{cov}(\cdot)$ is the covariance of the rank variables.
- σ represents the standard deviation of the rank variables.

The final result ρ (Spearman's rank correlation coefficient) directly quantify the stability and speed of the model's logical convergence. Smaller ρ indicates that as the model progresses through reasoning steps $\mathcal{T}$, the entropy H_t consistently decreases. As shown in the Figure 2 (right), all three models show that smaller ρ and better downward entropy trends often relates to fewer tokens generated.

F Additional Empirical Analysis

F.1 Analysis on Momentum

To study the sensitivity to the momentum coefficient γ, we conduct ablation experiments with $\gamma \in \{0.2, 0.5, 0.8, 0.9, 0.95, 0.99\}$. Results are reported in Table 5.

Overall, model performance is highly sensitive to the choice of γ. A typical momentum value $\gamma = 0.9$ consistently achieves the best accuracy across challenging benchmarks, indicated

γ	AMC23		AIME24		MATH500		GPQA-D		AES↑
	Acc ↑	Len ↓	Acc ↑	Len ↓	Acc ↑	Len ↓	Acc ↑	Len ↓	
0.5	85.0	2180.1	40.0	6381.6	90.4	1441.6	36.4	2684.5	1.04
0.8	87.5	2288.1	26.7	5622.7	89.0	1441.5	33.8	5622.7	0.66
0.9	**87.5**	**2411.6**	**56.7**	**4639.3**	**90.6**	**1479.2**	**37.4**	**2452.4**	**1.53**
0.95	82.5	3160.9	56.7	5289.3	89.2	1445.7	35.4	3052.2	1.29
0.99	80.0	2371.0	46.7	5070.0	88.4	1489.8	38.9	2621.4	1.12

Table 5: Analysis on the impact of the momentum γ in Equation 7.

by the highest AES score of 1.528. Smaller values ($\gamma \leq 0.8$) lead to degraded reasoning accuracy, particularly on harder tasks such as AIME24. In contrast, overly large momentum ($\gamma \geq 0.95$) results in reduced accuracy and inflated generation length, indicating over-smoothing of step-wise entropy signals. These results suggest that $\gamma = 0.9$ offers the best balance between temporal stability and responsiveness.

F.2 The Influence of Entropy Trend Reward on Final Answer.

In this section, we evaluate the quality and conciseness of the final response produced after the reasoning process. Unlike the internal reasoning trajectory, this segment represents the actual output presented to the user. While traditional reasoning models often carry over the verbosity of their internal chain-of-thought into the final output, our method helps to reduce such problem by presenting a final result that has higher information density and is free from ambiguity.

F.3 Example 1

Final Answer Part

Before(408 Tokens):
To determine the remainder of $54 \mod 6$, we apply the concept of modular arithmetic, which involves finding the remainder when one number is divided by another.

—

Step 1: Understand the Problem

We are asked to compute the remainder when 54 is divided by 6. In mathematical terms, this is:

$54 \mod 6$

This means we want to find the integer r such that:

$54 = 6q + r \quad \text{where } 0 \leq r < 6$

Here, q is the quotient, and r is the remainder.

—

Step 2: Perform the Division

We divide 54 by 6:
$54 \div 6 = 9$ This division is exact, meaning there is no remainder. Therefore, the remainder r is:

$r = 54 - (6 \times 9) = 54 - 54 = 0$

—

Step 3: Confirm with Alternative Methods

- **Using the Division Algorithm**: Since $54 = 6 \times 9 + 0$, the remainder is clearly 0.
- **Using Properties of Divisibility**: 54 is divisible by both 2 and 3 (since it's even and the sum of its digits is 9, which is divisible by 3), so it is divisible by 6.
- **Breaking Down the Number**: $54 = 50 + 4$, and $50 \mod 6 = 2$, $4 \mod 6 = 4$, so $(2 + 4) \mod 6 = 6 \mod 6 = 0$.

All these methods confirm the same result.

—

Final Answer

$$\boxed{0}$$

After(64 Tokens)
To determine the remainder of $54 \mod 6$, we perform the division of 54 by 6. Since $6 \times 9 = 54$, the division is exact with no remainder. Therefore, the remainder is:

$$\boxed{0}$$

F.4 Example 2

Final Answer Part

Before(386 Tokens):
To find the **dot product** of two vectors, we use the formula:

$$\mathbf{u} \cdot \mathbf{v} = u_1v_1 + u_2v_2 + u_3v_3$$

where $\mathbf{u} = \begin{pmatrix} u_1 \\ u_2 \\ u_3 \end{pmatrix}$ and $\mathbf{v} = \begin{pmatrix} v_1 \\ v_2 \\ v_3 \end{pmatrix}$ are two vectors in $\mathbb{R}^3$.

—

Step 1: Identify the components of the vectors

Given:

$$\mathbf{u} = \begin{pmatrix} -2 \\ 0 \\ 7 \end{pmatrix}, \quad \mathbf{v} = \begin{pmatrix} 3 \\ 4 \\ -5 \end{pmatrix}$$

So, the components are:

- $u_1 = -2$, $u_2 = 0$, $u_3 = 7$
- $v_1 = 3$, $v_2 = 4$, $v_3 = -5$

—

Step 2: Compute the dot product

$$\mathbf{u} \cdot \mathbf{v} = (-2)(3) + (0)(4) + (7)(-5)$$

Now compute each term:

- $(-2)(3) = -6$
- $(0)(4) = 0$
- $(7)(-5) = -35$

Add them together:

$$-6 + 0 + (-35) = -41$$

—

Final Answer

$$\boxed{-41}$$

- -

After(127 Tokens):
To find the dot product of the vectors

$$\begin{pmatrix} -2 \\ 0 \\ 7 \end{pmatrix} \quad \text{and} \quad \begin{pmatrix} 3 \\ 4 \\ -5 \end{pmatrix},$$

multiply the corresponding components and then sum the results:

$$(-2)(3) + (0)(4) + (7)(-5) = -6 + 0 - 35 = -41$$

Thus, the dot product is

$$\boxed{-41}$$

G Example Responses of Trained Models

G.1 Qwen3-8B

Question

A regular octagon has the same perimeter as the regular hexagon shown here with side length 16 cm. How long is each side of the octagon?

Info

(Original Model Output Tokens: **5140**)

Reasoning Trajectory (667 Tokens)

<think>
Okay, so I need to find the side length of a regular octagon that has the same perimeter as a regular hexagon with side length 16 cm. Let me start by recalling some basics about regular polygons.

First, a regular hexagon has all sides equal and all internal angles equal. Similarly, a regular

octagon also has all sides equal. The perimeter of a regular polygon is just the number of sides multiplied by the length of one side.

So, the perimeter of the hexagon would be 6 times the side length. Since the side length is given as 16 cm, the perimeter of the hexagon is 6 * 16 cm. Let me calculate that: 6 * 16 is 96 cm. So the perimeter is 96 cm.

Now, the regular octagon has the same perimeter. Let me denote the side length of the octagon as 's'. Since an octagon has 8 sides, its perimeter would be 8 * s. Therefore, setting this equal to the perimeter of the hexagon: 8s = 96 cm. To find 's', I can divide both sides by 8.

Let me do that: 96 divided by 8. Hmm, 8*12 is 96, so s = 12 cm. Wait, is that right? Let me check again. 8 times 12 is 96, yes. So each side of the octagon would be 12 cm.

But wait, let me make sure I didn't mix up the number of sides. Hexagon has 6 sides, octagon has 8. So yes, if the perimeters are equal, then 6*16 = 8*s. So s = (6*16)/8. Which is (96)/8 = 12. Yep, that seems correct.

I think that's all there is to it. The Asymptote code provided probably just draws the hexagon with a label on one of its sides, but since the problem states the hexagon has side length 16 cm, I don't need to worry about the diagram specifics.

Therefore, the answer should be 12 cm.

Final Answer The length of each side of the octagon is $\boxed{12}$ centimeters.
</think>

To determine the side length of a regular octagon that has the same perimeter as a regular hexagon with side length 16 cm, we start by calculating the perimeter of the hexagon.
A regular hexagon has 6 sides, so its perimeter is:

$$6 \times 16 = 96 \text{ cm}$$

A regular octagon has 8 sides. Let the side length of the octagon be s. Its perimeter is:

$$8s$$

Since the perimeters are equal, we set:

$$8s = 96$$

Solving for s:

$$s = \frac{96}{8} = 12$$

Thus, the length of each side of the octagon is $\boxed{12}$ centimeters.

G.2 Qwen3-4B

Question

What is the least positive integer multiple of 30 that can be written with only the digits 0 and 2?

Info

(Original Model Output Tokens: **9170**)

Reasoning Trajectory (1149 Tokens)

<think>
Okay, so I need to find the smallest positive integer multiple of 30 that only has digits 0 and 2. Let me think. First, since it's a multiple of 30, it has to be divisible by both 3 and 10. Divisible by 10 means the number has to end with a 0. So the last digit is 0. Then, for divisibility by 3, the sum of the digits has to be a multiple of 3. Since the digits can only be 0 or 2, the sum will be 2 times the number of 2s in the number. So the number of 2s has to be a multiple of 3. Therefore, the number must have 3, 6, 9, etc., number of 2s. But since we want the smallest such number, probably 3 twos. But also, the number has to end with 0. So maybe something like 2220? Let me check. 2220 divided by 30 is 74, which is an integer. Wait, is 2220 a multiple of 30? Let me check. 2220 divided by 10 is 222, which is divisible by 3 because 2+2+2=6, which is divisible by 3. So yes, 2220 is a multiple of 30. But is there a smaller number? Let me think. The number has to end with 0. So possible numbers are 20, 200, 220, 2000, 2200, 2020, 2220, etc. Let's check each of these. 20: sum of digits is 2, not divisible by 3. 200: sum is 2, same. 220: sum is 2+2=4, not divisible by 3. 2000: sum is 2, no. 2200: sum is 2+2=4, no. 2020: sum is 2+2=4, no. 2220: sum is 2+2+2=6, which is divisible by 3. So 2220 is the first one. Wait, but is there a smaller number? Like, maybe a number with fewer digits? 2220 is four digits. Is there a three-digit number? The three-digit numbers ending with 0 are 200, 220, 200. But their digit sums are 2, 4, 2. None are divisible by 3. So no. How about a four-digit number? 2220 is the first one. So maybe 2220 is the answer. But let me check if there's a smaller four-digit number. For example, 2020, 2002, but those don't end with 0. Wait, the number has to end with 0. So the four-digit numbers ending with 0 are 2000, 2020, 2200, 2220. We checked those. 2000: sum 2. 2020: sum 4. 2200: sum 4. 2220: sum 6. So 2220 is the first. Therefore, the answer is 2220. Wait, but let me check if there's a number with more digits but smaller in value. Wait, no, because 2220 is four digits. Any number with more digits would be larger. So I think 2220 is the answer. But let me confirm by dividing 2220 by 30. 30 times 74 is 2220. Yes. So I think that's it.

Final Answer The least positive integer multiple of 30 that can be written with only the digits 0 and 2 is $\boxed{2220}$.
</think>
To find the least positive integer multiple of 30 that

can be written using only the digits 0 and 2, we must consider the divisibility rules for 30. A number is divisible by 30 if it is divisible by both 3 and 10.

- **Divisibility by 10**: The number must end in 0. - **Divisibility by 3**: The sum of the digits must be a multiple of 3. Since the only digits allowed are 0 and 2, the sum of the digits will be 2 × (number of 2s). Therefore, the number of 2s must be a multiple of 3.

The smallest such number is formed by using three 2s and one 0 at the end, resulting in the number **2220**. Checking: - It ends in 0, so it is divisible by 10. - The sum of its digits is $2 + 2 + 2 + 0 = 6$, which is divisible by 3.

Thus, 2220 is the smallest number meeting the criteria.

$\boxed{2220}$

G.3 DeepSeek-R1-Distill-Qwen-7B

Question

What is the concentration of calcium ions in a solution containing 0.02 M stochiometric Ca-EDTA complex (we assume that the pH is ideal, T = 25 °C). KCa-EDTA = $5x10^10$.

A. $6.3x10^-7M$
B. $1.0x10^-2M$
C. $5.0x10^-3M$
D. $2.0x10^-2M$

Info

(Original Model Output Tokens: **16384**)

Reasoning Trajectory (940 Tokens)

</think>
Okay, so I have this problem here about the concentration of calcium ions in a solution. Let me try to figure it out step by step.

First, the problem says there's a 0.02 M stochiometric Ca-EDTA complex. I think that means the concentration of the complex is 0.02 M. They also mention that the pH is ideal and the temperature is 25°C. The formation constant, Kf, is given as 5×10^{10}.

Hmm, formation constant, Kf, is the equilibrium constant for the formation of the complex between calcium ions and EDTA. The reaction should be something like $Ca^{2+} + EDTA^{4-} \rightleftharpoons [Ca-EDTA]^{2-}$ complex.

So, the formation constant Kf is equal to $[Ca - EDTA]/([Ca^2+][EDTA^{4-}])$. Since it's a stochiometric complex, I assume the concentration of Ca^{2+} and $EDTA^{4-}$ are equal because they react in a 1:1 ratio. Let me denote the concentration of Ca^{2+} as 'c'. Then, the concentration of $EDTA^{4-}$ is also 'c'.

The total concentration of the complex is 0.02 M, which is equal to [Ca-EDTA]. So, $Kf = 0.02/(c * c) = 0.02/c^2$.

Given that Kf is 5×10^{10}, I can set up the equation: $5 \times 10^{10} = 0.02/c^2$.

To solve for c^2, I rearrange the equation: $c^2 = 0.02/(5 \times 10^{10})$. Let me calculate that. 0.02 divided by 5 is 0.004, and then divided by 10^{10} is $4x10^{-13}$. So, $c^2 = 4 \times 10^{-13}$.

Taking the square root of both sides, $c = \sqrt{4 \times 10^{-13}} = 2 \times 10^{-6.5}$. Wait, $10^{-6.5}$ is the same as $10^{-6} \times 10^{-0.5}$, which is approximately 3.16×10^{-7}. So, $2 \times 3.16 \times 10^{-7}$ is about 6.32×10^{-7}.

Looking at the options, A is $6.3 \times 10^{-7}M$, which matches my calculation. So, I think the answer is A.

Final Answer
$\boxed{A}$
</think> The problem involves finding the concentration of calcium ions in a solution containing a 0.02 M stochiometric Ca-EDTA complex. The formation constant K_f is given as 5×10^{10}.

1. The reaction is $Ca^{2+} + EDTA^{4-} \rightleftharpoons$ Ca-EDTA complex.
2. The formation constant K_f is given by $K_f = \frac{[\text{Ca-EDTA}]}{[Ca^{2+}][EDTA^{4-}]}$.
3. Since the complex is stochiometric, the concentrations of Ca^{2+} and $EDTA^{4-}$ are equal, denoted as c.
4. The total concentration of the complex is 0.02 M, so $K_f = \frac{0.02}{c^2}$.
5. Solving for c^2: $c^2 = \frac{0.02}{5 \times 10^{10}} = 4 \times 10^{-13}$.
6. Taking the square root: $c = \sqrt{4 \times 10^{-13}} = 2 \times 10^{-6.5} \approx 6.3 \times 10^{-7}$.

Thus, the concentration of calcium ions is 6.3×10^{-7} M.

$\boxed{A}$