ON-POLICY SUPERVISED FINE-TUNING FOR EFFICIENT REASONING

Anonymous authors
Paper under double-blind review

ABSTRACT

Large reasoning models (LRMs) are commonly trained with reinforcement learning (RL) to explore long chain-of-thought reasoning, achieving strong performance at high computational cost. Recent methods add multi-reward objectives to jointly optimize correctness and brevity, but these complex extensions often destabilize training and yield suboptimal trade-offs. We revisit this objective and challenge the necessity of such complexity. Through principled analysis, we identify fundamental misalignments in this paradigm: KL regularization loses its intended role when correctness and length are directly verifiable, and group-wise normalization becomes ambiguous under multiple reward signals. By removing these two items and simplifying the reward to a truncation-based length penalty, we show that the optimization problem reduces to supervised fine-tuning on self-generated data filtered for both correctness and conciseness. We term this simplified training strategy **on-policy SFT**. Despite its simplicity, on-policy SFT consistently defines the accuracy–efficiency Pareto frontier. It reduces CoT length by up to **80**% while maintaining original accuracy, surpassing more complex RL-based methods across five benchmarks. Furthermore, it significantly enhances training efficiency, reducing GPU memory usage by **50**% and accelerating convergence by **70**%.

1 INTRODUCTION

The emergence of large reasoning models (LRMs), trained with reinforcement learning (RL) methods including Group Relative Policy Optimization (GRPO) Shao et al. (2024) for reasoning-trajectory exploration, has yielded strong performance across many tasks Wei et al. (2023); Xu et al. (2025a); Guo et al. (2025); Yang et al. (2025a). However, their performance relies on long chains of reasoning tokens, incurring substantial computational and memory overhead Chen et al. (2025c); Ding et al. (2026).

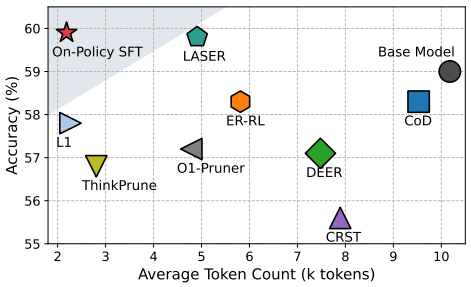

Figure 1: On-Policy SFT achieves a state-of-the-art accuracy–length trade-off on DeepSeek-R1-1.5B, reducing CoT length by ∼80% while slightly improving accuracy.

To address this bottleneck, a growing body of work has explored efficient reasoning methods to shorten CoT while preserving accuracy Chen et al. (2025d); Hou et al. (2025); Feng et al. (2025). A common and intuitive strategy is to inherit the GRPO-based RL framework and reformulate efficient reasoning as a multi-reward optimization problem, in which auxiliary length-related rewards are introduced to encourage concise reasoning (Aggarwal & Welleck, 2025; Liu et al., 2025b). Despite their conceptual appeal, such multi-reward formulations substantially increase reward-shaping complexity. In practice, they often lead to degraded accuracy, training instability, slower convergence, and heightened sensitivity to hyperparameters (Liu et al., 2025a; Li et al., 2025a; Chen et al., 2025b).

In this paper, we examine whether indiscriminately adopting complex objectives like GRPO aligns with the intrinsic structure of efficient reasoning. Through principled analysis, we identify two fundamental misalignments in the current paradigm. First, *KL-divergence regularization becomes*

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.

redundant. While essential in reinforcement learning from human feedback (RLHF) to mitigate reward overoptimization against noisy or learned reward models (Stiennon et al., 2022; Christiano et al., 2023), it is unnecessary for common efficient reasoning tasks, where correctness and length are directly verifiable and free from distributional ambiguity. Second, *group-wise reward normalization becomes ill-suited*. Originally designed to stabilize training with single-reward signals (Shao et al., 2024), this mechanism can introduce optimization bias by amplifying gradients from uninformative samples (Liu et al., 2025c) and obscuring distinctions between candidates in multi-reward scenarios, thereby introducing learning-signal ambiguity Yang et al. (2026). Furthermore, guided by recent findings that optimization objective design outweighs sophisticated reward shaping Liu et al. (2025a); He et al. (2025), we adopt the simplest length penalty truncation, which assigns zero reward to responses exceeding a fixed length limit. Taken together, by *removing KL regularization, group-wise normalization, and adopting truncation-based rewards*, the original policy gradient objective reduces into a reward-free form equivalent to maximum-likelihood supervised fine-tuning (SFT). Unlike standard SFT on fixed datasets, however, this method updates gradients using correctness- and conciseness-filtered on-policy responses. We call this reward-free recipe **on-policy SFT**.

Despite its simplicity and reward-shaping-free design, on-policy SFT achieves state-of-the-art accuracy–length trade-offs against **ten** strong baselines spanning training-free, SFT-based, and RL-based approaches on five widely used math benchmarks. On DeepSeek-R1-Distill-Qwen-1.5B, it attains 59.9% overall accuracy and 73.6% Pass@N, slightly outperforming the original model's 59.0% and 73.5%, while reducing average generation length from 10,178 to 2,186 tokens, an approximate **80**% reduction (see Figure 1). Measured by `Eff`, a quantitative accuracy–efficiency metric, on-policy SFT scores 2.74%, surpassing the strongest RL-based baseline at 2.55%. On DeepSeek-R1-Distill-Qwen-7B, it similarly achieves the highest `Eff` score of 2.97% while reducing generation length by roughly **70**% relative to the original model. Furthermore, under varying generation-length budgets, on-policy SFT consistently lies on the accuracy–efficiency Pareto frontier.

Beyond inference-time efficiency, on-policy SFT also substantially improves training efficiency, reducing GPU memory consumption and wall-clock time per training step by approximately **50**% across rollout configurations, and accelerating convergence by **70**% compared to RL-based methods. Moreover, on-policy SFT demonstrates **stronger length control** than RL-based baselines, as evidenced by lower generation length variance across multiple generations for the same input. To better understand the source of these overall gains, we examine the role of the training data distribution. Specifically, we show that the effectiveness of on-policy SFT is primarily driven by the use of on-policy data. Finally, we further present a set of empirically grounded training guidelines for on-policy SFT, including rollout temperature, the number of rollouts per input, length bias correction, and maximum output length. Together, these guidelines enable stable optimization and strong practical performance, and are accompanied by principled explanations of their underlying mechanisms.

Overall, our results suggest that some complex designs in existing approaches may be misaligned with efficient reasoning, while simpler, principled alternatives already achieve strong accuracy–efficiency Pareto frontiers at substantially lower training cost. These findings suggest efficient reasoning may benefit from prioritizing simplicity and principled design over algorithmic complexity.

2 PRELIMINARY

Given an question q, an LLM generates an output $o = \{S, a\}$, where $S = (s_1, s_2, \ldots, s_T)$ denotes a sequence of reasoning steps and a is the final answer. The objective of efficient reasoning is to learn a policy that minimizes reasoning length while preserving answer accuracy.

GRPO for Efficient Reasoning GRPO has become widely adopted for improving mathematical reasoning, driven by its success in training DeepSeek-R1 Guo et al. (2025). Compared to PPO (Schulman et al., 2017), GRPO removes the explicit value function and estimates advantages group-relatively. Specifically, for a question–answer pair (q, a), the behavior policy $\pi_{\theta_{\text{old}}}$ samples G candidate responses $\{o_i\}_{i=1}^{G}$. We compute advantages by normalizing rewards within each group:

$$\hat{A}_{i,t} = \frac{R_i - \text{mean}(\{R_i\}_{i=1}^{G})}{\text{std}(\{R_i\}_{i=1}^{G})}. \tag{1}$$

Table 1: Comparison of on-policy SFT with baselines on the GSM8K, MATH-500, AMC23, AIME24, and AIME25 datasets. **Acc**, **P@N**, **Tok**, and **CR** denote accuracy, Pass@N, average generated tokens, and compression rate, respectively. Row colors denote method categories: amber for *training-free*, green for *SFT-based*, and lavender for *RL-based* methods.

Method	GSM8K				MATH-500				AMC23				AIME24				AIME25				Overall				
	Acc↑	P@N↑	Tok↓	CR↓	Acc↑	P@N↑	Tok↓	CR↓	Acc↑	P@N↑	Tok↓	CR↓	Acc↑	P@N↑	Tok↓	CR↓	Acc↑	P@N↑	Tok↓	CR↓	Acc↑	P@N↑	Tok↓	CR↓	Eff↑
DeepSeek-R1-Distill-Qwen-1.5B																									
COT	85.0	92.0	2,533	100.0%	86.0	93.4	5,420	100.0%	73.0	92.5	9,171	100.0%	25.3	50.0	17,418	100.0%	26.0	40.0	16,348	100.0%	59.0	73.5	10,178	100.0%	0.58
CoD	84.8	92.4	1,717	67.7%	83.8	91.8	5,044	93.0%	71.5	87.5	8,518	92.8%	28.6	53.3	17,058	97.9%	22.6	33.3	15,297	93.5%	58.3	71.7	9,527	93.6%	0.61
DEER	83.5	92.0	1,091	43.0%	83.9	92.6	3,131	57.7%	71.0	92.5	6,908	75.3%	26.6	46.6	12,964	74.4%	20.6	33.3	13,305	81.3%	57.1	71.4	7,480	73.5%	0.76
CRST	78.7	89.8	1,075	42.4%	83.5	92.7	3,874	71.5%	70.5	92.5	7,410	80.8%	25.3	53.3	13,954	80.1%	20.0	30.0	13,147	80.4%	55.6	71.7	7,892	77.5%	0.70
TokenSkip	52.6	52.6	12,715	501.9%	52.0	52.0	13,685	252.4%	32.5	32.5	17,452	190.3%	3.3	3.3	24,889	142.8%	3.3	3.3	28,148	172.1%	28.7	28.7	19,378	190.4%	0.15
StepEntropy	77.1	87.7	**507**	**20.0%**	64.2	80.6	4,475	82.5%	47.5	77.5	9,237	100.7%	8.6	30.0	15,437	88.6%	6.6	23.3	14,438	88.3%	40.8	59.8	8,819	86.6%	0.46
ThinkPrune	84.7	91.8	710	28.0%	84.2	92.4	1,546	28.5%	72.0	**95.0**	2,672	29.1%	25.3	46.6	4,888	28.1%	18.0	33.3	4,213	25.8%	56.8	71.8	2,806	27.6%	2.02
O1-Pruner	85.2	92.8	916	36.1%	83.0	90.6	2,383	43.9%	72.5	90.0	3,942	42.9%	27.3	53.3	8,164	46.8%	18.0	33.3	8,528	52.1%	57.2	72.0	4,787	47.0%	1.19
L1	86.0	**93.1**	1,607	63.4%	85.3	91.6	1,922	35.4%	**74.5**	90.0	2,170	23.7%	23.3	43.3	**2,881**	**16.5%**	20.0	30.0	**2,762**	**16.9%**	57.8	69.6	2,268	22.3%	2.55
ER-RL	83.1	92.0	721	28.5%	85.3	**93.6**	2,466	45.5%	72.4	92.5	4,913	53.6%	27.3	50.0	11,584	66.5%	**23.3**	33.3	9,373	57.3%	58.3	72.3	5,811	57.1%	1.00
LASER	86.0	92.2	1,178	46.5%	85.3	91.8	2,714	50.0%	72.0	90.0	4,262	46.4%	**33.3**	**66.6**	7,821	44.9%	22.6	33.3	8,569	52.4%	59.8	**74.8**	4,909	48.2%	1.22
On-Policy SFT	**86.1**	92.1	696	27.5%	**85.7**	92.5	**1,354**	**25.0%**	73.5	90.0	**1,977**	**21.6%**	30.7	56.7	3,808	21.8%	**23.3**	**36.7**	3,096	18.9%	**59.9**	73.6	**2,186**	**21.5%**	**2.74**
DeepSeek-R1-Distill-Qwen-7B																									
COT	92.5	95.3	1,701	100.0%	93.0	96.2	4,132	100.0%	88.5	95.0	6,722	100.0%	54.0	80.0	13,960	100.0%	38.6	56.6	14,478	100.0%	73.3	84.6	8,199	100.0%	0.89
CoD	91.4	96.0	605	35.5%	91.6	95.6	2,694	65.2%	**90.5**	95.0	5,018	74.6%	51.3	76.6	13,372	95.7%	38.0	56.6	14,202	98.0%	72.6	84.0	7,178	87.5%	1.01
DEER	91.9	95.8	788	46.3%	90.3	95.8	2,508	60.7%	**90.5**	95.0	4,866	72.3%	52.6	**80.0**	11,294	80.9%	36.6	56.6	11,786	81.4%	72.4	84.6	6,248	76.2%	1.16
CRST	91.6	95.1	1,601	94.1%	91.3	95.6	3,879	93.8%	88.5	92.5	6,337	94.2%	50.0	76.6	13,841	99.1%	36.6	53.3	14,785	102.0%	71.6	82.6	8,089	98.7%	0.89
TokenSkip	86.1	86.1	4,323	254.1%	79.6	79.6	6,584	159.3%	72.5	72.5	9,459	140.7%	23.3	23.3	22,443	160.7%	16.6	16.6	21,383	147.6%	55.6	55.6	12,838	156.6%	0.43
StepEntropy	87.6	93.6	487	28.6%	71.2	85.2	4,914	118.9%	48.5	77.5	9,734	144.8%	10.0	26.6	15,986	114.5%	12.6	20.0	15,618	107.8%	46.0	60.6	9,348	114.0%	0.49
ThinkPrune	92.2	95.7	862	50.7%	91.3	95.2	1,984	48.0%	88.0	95.0	3,397	50.5%	45.9	70.0	7,181	51.4%	31.3	43.3	7,514	51.9%	69.7	79.8	4,188	51.1%	1.67
O1-Pruner	92.0	**96.2**	1,439	84.6%	**92.4**	96.0	3,617	87.5%	**90.5**	95.0	5,379	80.0%	50.0	**80.0**	12,791	91.6%	39.3	**63.3**	12,585	86.9%	**72.8**	**86.1**	7,162	87.4%	1.02
L1	91.7	94.9	1,371	80.6%	91.4	95.3	1,708	41.3%	83.5	92.5	2,191	32.6%	39.3	76.6	**3,390**	**24.3%**	30.6	53.3	**3,122**	**21.6%**	67.3	82.5	**2,356**	**28.7%**	2.86
ER-RL	87.3	94.1	**371**	**21.8%**	90.9	96.1	2,014	48.7%	87.5	95.0	4,221	62.8%	**53.3**	**80.0**	9,858	70.6%	39.3	56.6	10,895	75.3%	71.7	84.4	5,472	66.7%	1.31
LASER	92.1	95.9	930	54.6%	92.0	**96.4**	1,857	44.9%	88.5	95.5	3,137	46.6%	51.3	73.3	6,104	43.7%	**40.0**	60.0	6,270	43.3%	**72.8**	84.2	3,660	44.6%	1.99
On-Policy SFT	**92.6**	96.0	498	29.3%	91.4	95.0	**1,178**	**28.5%**	87.5	**97.5**	**1,961**	**29.2%**	52.6	**80.0**	4,594	32.9%	**40.0**	60.0	4,039	27.9%	**72.8**	85.7	2,454	29.9%	**2.97**

Similar to PPO, GRPO employs a clipped surrogate objective and additionally incorporates an explicit Kullback–Leibler (KL) divergence penalty:

$$J_{\text{GRPO}}(\theta) = \mathbb{E}_{(q,a)\sim\mathcal{D},\,\{o_i\}_{i=1}^{G}\sim\pi_{\theta_{\text{old}}}(\cdot|q)}\left[\frac{1}{G}\sum_{i=1}^{G}\frac{1}{|o_i|}\sum_{t=1}^{|o_i|}\min\Big(r_{i,t}(\theta)\hat{A}_{i,t},\,\text{clip}\left(r_{i,t}(\theta),\,1-\varepsilon,1+\varepsilon\right)\hat{A}_{i,t}\Big)-\beta\,\text{KL}(\pi_\theta\,\|\,\pi_{\text{ref}})\right]. \tag{2}$$

where $r_{i,t}(\theta) = \frac{\pi_\theta(o_{i,t}|q,o_{i,<t})}{\pi_{\theta_{\text{old}}}(o_{i,t}|q,o_{i,<t})}$. In GRPO, although practical implementations may perform multiple gradient updates per training step, the samples are generated by a very recent version of the current policy $\pi_{\theta_{\text{old}}}(\cdot \mid q)$. Consequently, gradients and advantages are estimated using data generated by the current or a very recent policy, and GRPO is therefore categorized as an on-policy algorithm (Mroueh et al., 2025). As a reward-based policy gradient method, GRPO can be naturally extended to tasks that could be well defined by explicit reward functions. Since reasoning efficiency can be readily quantified as a scalar reward (e.g., via length penalties), recent studies (Hou et al., 2025; Aggarwal & Welleck, 2025; Rakotonirina et al., 2026) incorporate length-penalty rewards into GRPO to improve reasoning efficiency.

Reward Shaping for Efficient Reasoning While existing RL-based efficient reasoning methods employ a wide range of reward designs, we show that most can be cast into a unified formalization:

$$R_{\text{Eff}}(o \mid q) = R_{\text{Acc}}(o \mid q) + \gamma(o \mid q)\, R_{\text{Len}}(o \mid q), \tag{3}$$

where $R_{\text{Acc}}(o \mid q)$ denotes the accuracy reward, $R_{\text{Len}}(o \mid q)$ represents a length-related reward that encourages shorter reasoning trajectories, and $\gamma(o \mid q)$ controls the trade-off between correctness and efficiency. The landscape of reward shaping strategies has gradually shifted toward greater complexity, largely driven by increasingly elaborate choices of $R_{\text{Len}}(o \mid q)$ and $\gamma(o \mid q)$. For example, prior work conditions $\gamma(o \mid q)$ on answer correctness or question difficult to selectively activate $R_{\text{Len}}(o \mid q)$, and defines $R_{\text{Len}}(o \mid q)$ using group-level statistics such as the average, median, or maximum length (Arora & Zanette, 2025; Li et al., 2025a; Chen et al., 2025b). See Appendix B.

3 From GRPO to On-Policy SFT

Although existing RL-based efficient reasoning methods substantially reduce CoT length, they often suffer accuracy degradation varying across task complexity (Liu et al., 2025a; Rakotonirina et al., 2026). These suboptimal accuracy–efficiency trade-offs prompt us to examine whether indiscriminately inheriting complex optimization objectives fits efficient reasoning.

3.1 REVISITING THE GRPO OBJECTIVE

We examine GRPO for potential mismatches with two properties of efficient reasoning: *directly verifiable correctness and length penalties* and *its multi-reward nature*.

KL Divergence The KL divergence term is commonly adopted in reinforcement learning from human feedback (RLHF) (Christiano et al., 2023), where the reward is provided by a learned reward model trained on data collected under a reference policy π_{ref}. In this setting, KL regularization plays a critical role in preventing the optimized policy π_θ from deviating excessively from the distribution over which the reward model is reliable (Jaques et al., 2019; Stiennon et al., 2022; Liu et al., 2025c). In contrast, for efficient reasoning tasks, answer accuracy is verifiable, and length penalties can be quantified as scalar rewards, thereby eliminating the need for a learned reward model. As a result, concerns about reward unreliability under distributional shift do not arise. This suggests that *the KL divergence term may be unnecessary and may even overly constrain policy updates during optimization*. Removing this component can simplify the training objective and reduce the memory and computational overhead associated with maintaining the reference policy π_{ref} during training.

Group-wise Reward Normalization GRPO estimates advantages by normalizing rewards within each rollout group, thereby avoiding explicit value function approximation (as in PPO) and stabilizing training (Shao et al., 2024). Despite its benefits for reasoning, group-wise normalization may be ill-suited to efficient reasoning. First, *question-level normalization results in varying weights across different questions.* When rewards within a group vary little (e.g., all responses are correct and similarly long), the resulting small standard deviation in the denominator can amplify gradient magnitudes. In practice, such samples provide limited learning signal yet are overemphasized during optimization, potentially leading to training instability (Liu et al., 2025c; Yang et al., 2026). Second, *GRPO is fundamentally designed to optimize a single scalar reward objective, whereas efficient reasoning operates in a multi-reward regime*. This mismatch can introduce ambiguity in the learning signal when heterogeneous reward components are aggregated and normalized (Liu et al., 2026). For instance, consider two rollouts with reward vectors $(0, 1)$ and $(0, 0)$, which yield scalar rewards of 1 and 0 under summation. After group-wise normalization, the resulting advantages are $(-0.7071,\ 0.7071)$. Notably, the same normalized advantages also arise for reward vectors $(1, 1)$ and $(0, 0)$, despite their fundamentally different reward compositions. This observation motivates us to *forego group-wise reward normalization and optimize rewards directly using a baseline*.

The Simplest Length Penalty: Truncation Recent evidence suggests that principled optimization-objective design matters more than sophisticated reward design (Liu et al., 2025a; He et al., 2025). In particular, Liu et al. (2025a) show that even simple truncation—assigning zero reward to responses exceeding a fixed length threshold—can achieve state-of-the-art accuracy–efficiency trade-offs with an appropriate RL objective. Motivated by this finding, we pair the same minimalist truncation reward with our revised objective to isolate the optimization formulation's role and assess whether strong performance can be attained without intricate reward engineering. Formally, truncation defines a binary reward:

$$R_{\text{trunc}}(o \mid q) = \begin{cases} 1, & \text{if } o \text{ is correct and } |o| \le \tau, \\ 0, & \text{otherwise,} \end{cases} \tag{4}$$

where τ is a predefined length threshold. Unlike the composite reward in Equation 3, truncation removes auxiliary length components and trade-off coefficients, unifying correctness and conciseness.

3.2 ON-POLICY SFT

We analyze the optimization objective resulting from the analyses in subsection 3.1 from a gradient-based perspective. We begin by considering the gradient of the original GRPO objective:

$$\nabla_\theta J_{\text{GRPO}}(\theta) = \mathbb{E}_{(q,a)\sim\mathcal{D},\, \{o_i\}_{i=1}^{G}\sim\pi_{\theta_{\text{old}}}(\cdot|q)} \Bigg[\frac{1}{G}\sum_{i=1}^{G} \frac{1}{|o_i|} \sum_{t=1}^{|o_i|} \Big(\hat{A}_{i,t} + \beta\Big(\frac{\pi_{\text{ref}}(o_{i,t} \mid o_{i,<t})}{\pi_\theta(o_{i,t} \mid o_{i,<t})} - 1 \Big) \Big) \nabla_\theta \log \pi_\theta(o_{i,t} \mid q, o_{i,<t}) \Bigg]. \tag{5}$$

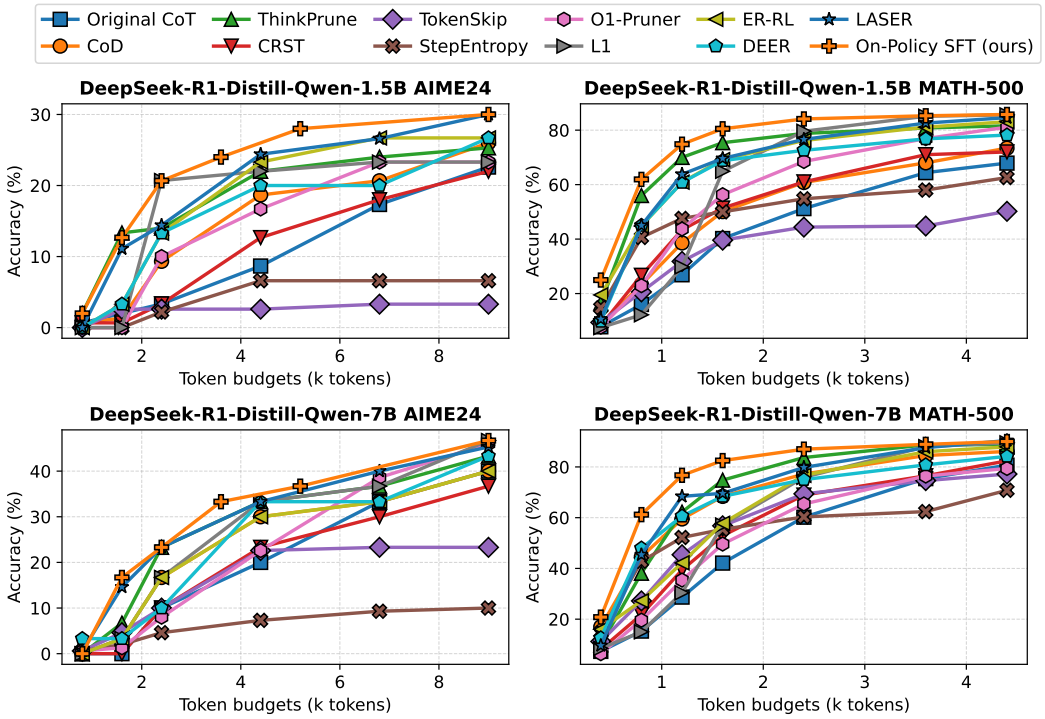

Figure 2: Performance–efficiency trade-offs of on-policy SFT vs. baselines across token budgets.

A detailed derivation is provided in Appendix C. We first simplify the GRPO objective by removing the KL divergence–related term $\beta\left(\frac{\pi_{\text{ref}}(o_{i,t}|o_{i,<t})}{\pi_\theta(o_{i,t}|o_{i,<t})} - 1\right)$ and by eliminating the standard-deviation $\text{std}(\{R_i\}_{i=1}^G)$ in reward normalization. The resulting policy gradient simplifies to:

$$\nabla_\theta J(\theta) = \mathbb{E}_{(q,a)\sim\mathcal{D},\,\{o_i\}_{i=1}^G\sim\pi_{\theta_{\text{old}}}(\cdot|q)}\left[\frac{1}{G}\sum_{i=1}^{G}\frac{1}{|o_i|}\sum_{t=1}^{|o_i|}\hat{R}_{i,t}\,\nabla_\theta \log \pi_\theta(o_{i,t}\mid q, o_{i,<t})\right], \tag{6}$$

where $\hat{R}_{i,t} = R_i - \text{mean}(\{R_i\}_{i=1}^G)$. The reward baseline $\text{mean}(\{R_i\}_{i=1}^G)$ reduces gradient variance and helps stabilize training in challenging regimes that require balancing exploration and exploitation, such as training models from weak to strong reasoning ability (Guo et al., 2025; Wang et al., 2025a). In our setting, however, we focus on models already exhibiting strong reasoning ability, with the objective of favoring shorter solutions among existing reasoning trajectories and emphasizing exploitation over exploration. Accordingly, we set $\hat{R}_{i,t} = R_i$, further simplifying the derivation without affecting intended optimization behavior. Notably, after simplification, GRPO's gradient reduces to a form mathematically equivalent to the classical *REINFORCE* (Williams, 1992) policy gradient, as detailed in Appendix D.

In GRPO, the reward assigned to a rollout is shared across all tokens. Accordingly, for a fixed rollout o_i, the reward term is constant, i.e., $\hat{R}_{i,t} = \hat{R}_i$ for all $t = 1, \ldots, |o_i|$. We therefore drop the token index t and express the reward as depending only on the rollout index i, yielding

$$\nabla_\theta J(\theta) = \mathbb{E}_{(q,a)\sim\mathcal{D},\,\{o_i\}_{i=1}^G\sim\pi_{\theta_{\text{old}}}(\cdot|q)}\left[\frac{1}{G}\sum_{i=1}^{G}\hat{R}_i\,\frac{1}{|o_i|}\sum_{t=1}^{|o_i|}\nabla_\theta \log \pi_\theta(o_{i,t}\mid q, o_{i,<t})\right]. \tag{7}$$

Next, following Sec. 3.1, we instantiate the raw reward as a binary truncation reward. Specifically, $\hat{R}_i \in \{0, 1\}$, where $\hat{R}_i = 1$ if the response is correct within a predefined fixed length limit L, and $\hat{R}_i = 0$ otherwise. We represent this reward using an indicator function $\mathbf{1}\{\hat{R}_i = 1\}$. Let $\mathcal{C}_L$ denote the set of responses for which $\hat{R}_i = 1$. The indicator can then be equivalently expressed as the

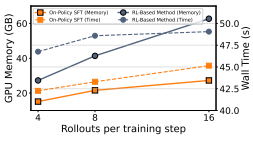

Figure 3: GPU memory and wall-clock time per step of on-policy SFT and ThinkPrune.

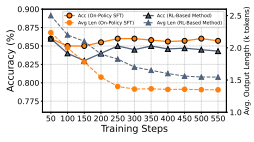

Figure 4: Convergence speed of on-policy SFT and ThinkPrune on MATH-500 dataset.

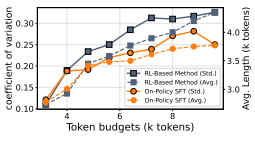

Figure 5: Length control comparison measured by coefficient of variation on AIME24.

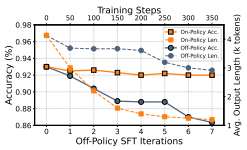

Figure 6: Comparison between on-policy SFT and its off-policy variant on MATH-500.

set-based constraint $\mathbf{1}\{o_i \in \mathcal{C}_L\}$. Substituting this expression yields

$$\nabla_\theta J(\theta) = \mathbb{E}_{(q,a)\sim\mathcal{D},\, \{o_i\}_{i=1}^{G}\sim\pi_{\theta_{\text{old}}}(\cdot|q)} \left[\frac{1}{G} \sum_{i=1}^{G} \mathbf{1}\{o_i \in \mathcal{C}_L\} \frac{1}{|o_i|} \sum_{t=1}^{|o_i|} \nabla_\theta \log \pi_\theta(o_{i,t} \mid q, o_{i,<t}) \right]. \tag{8}$$

Surprisingly, we observe that the *resulting gradient shares the same functional form as the gradient of the maximum likelihood objective* used in standard supervised fine-tuning:

$$\nabla_\theta J_{\text{SFT}}(\theta) = \mathbb{E}_{(q,o)\sim\mathcal{D}_{\text{SFT}}} \left[\frac{1}{|o|} \sum_{t=1}^{|o|} \nabla_\theta \log \pi_\theta(o_t \mid q, o_{<t}) \right]. \tag{9}$$

The only difference lies in the data distribution. Rather than optimizing over a fixed supervised dataset $\mathcal{D}_{\text{SFT}}$, our updates are computed on responses generated on-policy by $\pi_{\theta_{\text{old}}}$, where $\mathbf{1}\{o_i \in \mathcal{C}_L\}$ serves as a selection mechanism enforcing correctness and length constraints.

Finally, we examine the dependence on response length $|o_i|$. Unlike pretraining (Shoeybi et al., 2020), where all tokens are packed into a fixed-length context and the loss is normalized by the context length, self-generated CoT responses vary substantially in length. As a result, the $|o_i|$ term in Equation 8 can induce disproportionately large gradients for shorter, potentially noisy responses, biasing training toward suboptimal solutions. To mitigate this issue, we replace $|o_i|$ with the maximum response length within the batch, ensuring uniform gradient magnitudes across all tokens. Rewriting the corrected gradients as a loss function yields our final optimization objective:

$$\begin{aligned} J(\theta) &= \mathbb{E}_{(q,a)\sim\mathcal{D},\, \{o_i\}_{i=1}^{G}\sim\pi_{\theta_{\text{old}}}(\cdot|q)} \left[\frac{1}{G} \sum_{i=1}^{G} \mathbf{1}\{o_i \in \mathcal{C}_L\} \frac{1}{\max_j |o_j|} \sum_{t=1}^{|o_i|} \log \pi_\theta(o_{i,t} \mid q, o_{i,<t}) \right] \\ &= c_L \underbrace{\mathbb{E}_{o_i\sim\mathcal{D}_L^+} \left[\frac{1}{\max_j |o_j|} \sum_{t=1}^{|o_i|} \log \pi_\theta(o_{i,t} \mid q, o_{i,<t}) \right]}_{J_{\text{SFT}}(\theta)}, \end{aligned} \tag{10}$$

where $\mathcal{D}_L^+ = \{o_i \mid (q,a) \sim \mathcal{D},\, o_i \sim \pi_{\theta_{\text{old}}}(\cdot \mid q),\, o_i \in \mathcal{C}_L\}$, and $c_L = \mathbb{E}_{(q,a)\sim\mathcal{D},\, o_i\sim\pi_{\theta_{\text{old}}}(\cdot|q)}[\mathbf{1}\{o_i \in \mathcal{C}_L\}]$ is a constant independent of θ. We adopt Equation 10 as the optimization objective and refer to the resulting minimalist training approach as **on-policy supervised fine-tuning (SFT)**, reflecting its use of the standard SFT loss under an on-policy data distribution. For completeness, we present the full training procedure of on-policy SFT in Algorithm 1.

4 Experimental Setup

Models and Training Dataset We train representative open-sourced LRMs, including DeepSeek-R1-1.5B and DeepSeek-R1-7B (Guo et al., 2025), on the DeepScaleR dataset (Luo et al., 2025b).

Implementation Details During rollout, we use temperature 1.0, top-$p = 0.95$, and a maximum decoding length of 3,500 tokens; training details are provided in Appendix F.

Evaluation We evaluate on five widely used mathematical reasoning benchmarks—GSM8K (Cobbe et al., 2021), MATH-500 (Lightman et al., 2023), AMC23 (AMC2023, 2023), AIME24

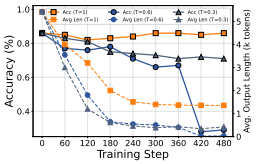

Figure 7: Training at rollout temperatures 0.3, 0.6, and 1.0.

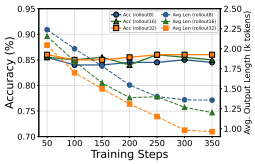

Figure 8: Rollout number ablation at temperature 1.0.

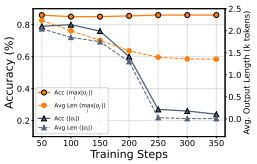

Figure 9: Training with and without length bias correction.

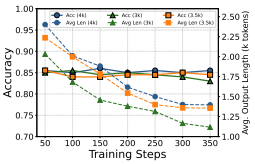

Figure 10: Maximum lengths of 3k, 3.5k, and 4k tokens.

(AIME2024, 2024), and AIME25 (AIME2025, 2025)—using multi-sample evaluation with $N = 64$ outputs per question for AMC23, AIME24, and AIME25, and $N = 16$ for the rest.

Metrics We report five evaluation metrics: accuracy (**Acc**, in %), **Pass@**N (in %), average response length in tokens (**Tok**), compression rate (**CR**, in %), and token efficiency (**Eff**). **Eff** represents the ratio of accuracy to average response length ($\text{Eff} = \text{Acc}/\text{Tok}$, in %), and reflects the trade-off between correctness and efficiency. Appendix F details the remaining metrics.

Baselines The baselines fall into three major categories:

- **Training-free methods** *Chain of Draft (CoD)* (Xu et al., 2025b) prompts concise generation at each step; see Appendix G.2 for the template. *Dynamic Early Exit in Reasoning (DEER)* (Yang et al., 2025b) terminates reasoning when the model exhibits high confidence.
- **SFT-based methods** *Concise Reasoning Self-Training (CRST)* (Munkhbat et al., 2025) fine-tunes models on concise reasoning paths from best-of-N sampling. *TokenSkip* (Xia et al., 2025) constructs compressed training data by pruning unimportant tokens; see Appendix H for details. *StepEntropy* (Li et al., 2025b) replaces the 80% lowest-entropy reasoning steps with a `[skip]` token.
- **RL-based methods** *ThinkPrune* (Hou et al., 2025) assigns a reward of 1 to responses that are correct within a predefined length limit and 0 otherwise. *O1-Pruner* (Luo et al., 2025a) introduces a length-harmonizing reward computed as the ratio between the CoT length of a reference model and that of the current policy. *L1* (Aggarwal & Welleck, 2025) trains models to satisfy user-specified length constraints; implementation details are provided in the Appendix H. *ER-RL* (Arora & Zanette, 2025) assigns rewards only to correct responses, with shorter outputs receiving higher rewards. *LASER* (Liu et al., 2025b) uses a step-function reward based on target length.

5 Experimental Results

5.1 Main Experiment

RL > training-free > SFT for efficient reasoning We summarize several key observations from our experiments. (i) As shown in Table 1, across both the 1.5B and 7B scales, RL-based methods consistently achieve superior accuracy–efficiency trade-offs, as reflected by higher Eff scores. (ii) Somewhat surprisingly, training-free methods consistently outperform SFT-based approaches in terms of efficiency trade-offs, indicating that not all training paradigms equally improve efficient reasoning. (iii) Although training-free methods preserve accuracy well, they struggle to achieve substantial compression on more challenging datasets. For instance, on the 7B model, CoD maintains accuracy on AIME25 but reduces generation length by only 2%, whereas on simpler datasets such as GSM8K it achieves up to a 35.5% reduction. (iv) We observe that methods such as TokenSkip and StepEntropy, which prune tokens or steps from self-generated CoT traces during training, suffer from severe performance degradation. In contrast, CRST, which performs SFT using complete, fluent, and concise reasoning traces, achieves markedly better results. This suggests that training on coherent and well-formed reasoning sentences is essential for effective efficient reasoning.

On-policy SFT exhibits surprisingly strong effectiveness Despite its simplicity and the fact that on-policy SFT does not rely on reward signals, it *achieves a superior performance–efficiency trade-off compared to existing state-of-the-art RL-based efficient reasoning methods* at both the 1.5B and 7B scales. As shown in Table 1, on the 1.5B model, on-policy SFT attains an accuracy of 59.9%,

outperforming all baselines and the original CoT approach, while generating only 2,186 tokens on average—an approximately **80**% reduction compared to the original model's 10,178 tokens. When evaluated using Eff, which captures the trade-off between correctness and reasoning efficiency, our method achieves an Eff score of 2.7%, exceeding the strongest RL-based baseline, L1 (2.55%). A similar trend holds for the 7B model, where on-policy SFT attains the highest Eff score of 2.97% and achieves an approximately **70**% reduction in generation length, again surpassing all RL-based methods. In addition, Figure 2 compares the performance of on-policy SFT with all other baselines on AIME24 and MATH-500 under varying generation length budgets, where we observe that our method consistently lies on the Pareto frontier.

We report *two intriguing findings*: (i) Despite zero truncation reward for responses over 3,500 tokens, our method remains effective when average lengths exceed this threshold. On AIME25, the 7B model achieves 40.0% accuracy at an average length of 4,039 tokens, outperforming all baselines and the original CoT. This suggests strong generalization: reasoning patterns learned from correct, length-constrained samples transfer to longer responses. (ii) Recent work highlights negative samples' role in preventing entropy collapse and improving exploration, often increasing Pass@N (Chen et al., 2025a; Tang et al., 2025). In contrast, our method optimizes only on positive samples yet achieves competitive Pass@N performance. At 1.5B, it scores 73.6%, slightly above the original CoT (73.5%) and trailing only LASER (74.8%); at 7B, it scores 85.7%, exceeding the original CoT (84.6%) and narrowly trailing o1-pruner (86.1%). We conjecture that negative samples may serve different roles in multi-reward optimization and reasoning enhancement alone.

On-Policy SFT improves training efficiency Beyond inference-time efficiency, we examine the training efficiency of on-policy SFT. Specifically, we compare our method with the most competitive RL-based baseline ThinkPrune, which also relies on a truncation reward, in terms of average GPU memory consumption and wall-clock time per training step (including rollout generation and parameter updates) under varying rollout numbers, as shown in Figure 3. All experiments are conducted on a single 94GB H100 GPU without gradient accumulation. Across all rollout settings, on-policy SFT consistently reduces both time and memory costs by approximately **50**% per training step relative to the RL-based method. The improved training efficiency of on-policy SFT stems from two factors. First, it avoids maintaining a reference model for KL-divergence computation, reducing GPU memory usage and computational overhead. Second, training is performed only on filtered on-policy samples. We argue that this filtering is critical to the superior performance of on-policy SFT over ThinkPrune, since truncation-based rewards combined with group-wise normalization can penalize responses that would otherwise be correct when their length slightly exceeds the maximum constraint. Beyond per-step efficiency, on-policy SFT also converges faster under identical training configurations: Figure 4 shows that the average output length stabilizes after approximately 300 optimization steps, compared with around 500 for ThinkPrune to reach comparable stability, a **40**% reduction in training steps. Overall, training requires significantly less memory and wall-clock time, with convergence taking only about **30**% of the RL-based baseline's wall-clock time.

On-policy SFT exhibits superior length control Beyond efficiency–performance trade-offs, strong length control requires stable response lengths across repeated samples. We measure the coefficient of variation, the standard deviation divided by the mean generation length (Everitt & Skrondal, 2010); see Appendix I. Lower values indicate more consistent lengths and stronger length control. For the 1.5B model, Figure 5 compares on-policy SFT with the RL-based baseline ThinkPrune using results averaged over AIME24. On-policy SFT produces shorter average responses and consistently exhibits stronger length control under varying token budgets. We attribute this advantage to removing group-wise reward normalization, which avoids penalizing potentially correct answers and helps preserve the model's confidence during generation.

These results suggest that the field may have introduced unnecessary complexity and that simpler alternatives can achieve strong accuracy–efficiency tradeoffs with markedly fewer training resources.

5.2 The Key Driver of Effectiveness: On-Policy Data

A natural question is why SFT-based baselines underperform on-policy SFT, despite sharing the same reward-free SFT objective. We identify the key distinction as lying in the data generation paradigm: SFT-based methods operate in an off-policy setting with a fixed dataset, whereas on-

policy SFT continually resamples training data from the current policy. This distinction further motivates a closer examination of the specific role played by on-policy data in our approach.

We construct an off-policy variant for comparison. Each iteration uses the same rollout and filtering procedure as on-policy SFT: sampling 8 responses per question and retaining only correct responses within the 3,500-token limit, matching the total number of questions used in 50 on-policy training steps. We train for one epoch on this fixed dataset and repeat for seven iterations, corresponding to 350 on-policy training steps. Figure 6 reports results for the 7B model on MATH-500. On-policy SFT rapidly reduces generation length and converges stably while preserving accuracy, whereas the off-policy variant yields only modest length reduction with substantial performance degradation. These results suggest that on-policy data primarily drives effectiveness in both RL-based efficient reasoning and on-policy SFT, rather than complex optimization objectives or reward shaping, which may even be detrimental. This aligns with recent findings attributing RL's effectiveness to on-policy data rather than sophisticated algorithmic choices, highlighting the importance of continual data refresh (Chen et al., 2025a; Shenfeld et al., 2026; Zhao et al., 2026; Hübotter et al., 2026).

5.3 Practical Training Guidelines

To promote stable training, we provide practical on-policy SFT guidelines with principled rationale. Unless specified otherwise, this subsection uses the 1.5B model evaluated on MATH-500.

Rollout temperature should be set to 1.0 Figure 7 shows training dynamics at rollout temperatures 0.3, 0.6, and 1.0. Temperatures below 1.0 consistently cause excessive length reduction with performance degradation or even collapse. Higher temperatures 1.2 and 1.5 also destabilize training, with monotonically increasing loss that fails to converge (Figure 13). Temperatures other than 1.0 violate the on-policy assumption; Appendix J provides theoretical justification and a detailed proof.

On-policy SFT benefits from scaling the rollout number At rollout temperature 1.0, training remains stable across all tested rollout numbers (Figure 8). Moreover, increasing the rollout number consistently leads to shorter average response lengths while maintaining accuracy. These results indicate that, given sufficient computational resources, scaling the rollout number at temperature 1.0 effectively improves on-policy SFT performance.

Length bias correction is necessary We study the bias-correction scheme described in subsection 3.2, where $|o_i|$ is replaced by the maximum response length within a batch, $\max_j |o_j|$. We visualize the training dynamics under both settings in Figure 9. As shown, directly using $|o_i|$ leads to training collapse: both generation length and accuracy rapidly decay and approach zero after approximately 250 training steps. In contrast, adopting $\max_j |o_j|$ yields stable training dynamics. This behavior arises because using $|o_i|$ assigns larger gradient updates to shorter responses, making training overly sensitive to noise and prone to converging to suboptimal solutions or even collapsing.

Length limit trades off performance and training cost We analyze training dynamics under different maximum generation lengths in Figure 10. A limit of 3k tokens yields the lowest training cost and fastest length reduction, but with slight performance degradation, whereas 4k tokens incur the highest cost and achieve the best performance with slower, more stable length reduction. Our chosen setting of 3.5k tokens strikes a balance, matching 4k performance at moderate training cost.

6 Conclusion

We revisit RL-based efficient reasoning from a minimalist perspective and show that much of the complexity inherited from GRPO, including KL regularization, group-wise normalization, and sophisticated reward shaping, may not be necessary for the efficient reasoning setting. By deliberately removing these components, we derive a simple, reward-free training procedure that is mathematically equivalent to SFT on on-policy data. Despite its simplicity, on-policy SFT achieves strong accuracy–efficiency trade-offs, improves training efficiency, and exhibits robust length control across benchmarks and model scales. Our findings suggest that efficient reasoning may benefit from prioritizing simplicity and principled design over increasing complexity.

AI USE STATEMENT

Generative AI tools were used for language polishing and assistance with code writing and document formatting.

ETHICS STATEMENT

This work studies efficient reasoning using existing models and benchmarks. Appendix Q discusses the scope and limitations of the evaluation, the potential benefits of reduced computational costs, and considerations regarding accuracy, robustness, and failure modes in practical deployments.

REPRODUCIBILITY STATEMENT

Section 4 describes the models, datasets, baselines, and evaluation metrics. Appendix F provides the training and evaluation settings, Appendix G provides the prompt templates, and Appendix H describes the baseline implementation details. Algorithm 1 presents the training procedure, while Appendices C and J provide the mathematical derivations.

REFERENCES

Pranjal Aggarwal and Sean Welleck. L1: Controlling how long a reasoning model thinks with reinforcement learning, 2025. URL https://arxiv.org/abs/2503.04697.

AIME2024. American invitational mathematics examination (aime) 2024-i & ii. https://huggingface.co/datasets/Maxwell-Jia/AIME_2024, 2024. Accessed: 2024-01-18.

AIME2025. American invitational mathematics examination (aime) 2025-i & ii. https://huggingface.co/datasets/opencompass/AIME2025, 2025. Accessed: 2024-01-18.

AMC2023. American mathematics competitions, 2023. URL https://artofproblemsolving.com/wiki/index.php/AMC_Problems_and_Solutions.

Sigrún Andradóttir, Daniel P. Heyman, and Teunis J. Ott. On the choice of alternative measures in importance sampling with markov chains. *Operations Research*, 43(3):509–523, 1995. doi: 10.1287/opre.43.3.509.

Daman Arora and Andrea Zanette. Training language models to reason efficiently, 2025. URL https://arxiv.org/abs/2502.04463.

Howard Chen, Noam Razin, Karthik Narasimhan, and Danqi Chen. Retaining by doing: The role of on-policy data in mitigating forgetting, 2025a. URL https://arxiv.org/abs/2510.18874.

Qiguang Chen, Dengyun Peng, Jinhao Liu, HuiKang Su, Jiannan Guan, Libo Qin, and Wanxiang Che. Aware first, think less: Dynamic boundary self-awareness drives extreme reasoning efficiency in large language models, 2025b. URL https://arxiv.org/abs/2508.11582.

Xinghao Chen, Zhijing Sun, Guo Wenjin, Miaoran Zhang, Yanjun Chen, Yirong Sun, Hui Su, Yijie Pan, Dietrich Klakow, Wenjie Li, et al. Unveiling the key factors for distilling chain-of-thought reasoning. In *Findings of the Association for Computational Linguistics: ACL 2025*, pp. 15094–15119, 2025c.

Xingyu Chen, Jiahao Xu, Tian Liang, Zhiwei He, Jianhui Pang, Dian Yu, Linfeng Song, Qiuzhi Liu, Mengfei Zhou, Zhuosheng Zhang, Rui Wang, Zhaopeng Tu, Haitao Mi, and Dong Yu. Do not think that much for 2+3=? on the overthinking of o1-like llms, 2025d. URL https://arxiv.org/abs/2412.21187.

Paul Christiano, Jan Leike, Tom B. Brown, Miljan Martic, Shane Legg, and Dario Amodei. Deep reinforcement learning from human preferences, 2023. URL https://arxiv.org/abs/1706.03741.

Karl Cobbe, Vineet Kosaraju, Mohammad Bavarian, Mark Chen, Heewoo Jun, Lukasz Kaiser, Matthias Plappert, Jerry Tworek, Jacob Hilton, Reiichiro Nakano, et al. Training verifiers to solve math word problems. *arXiv preprint arXiv:2110.14168*, 2021.

Alexis Conneau, Kartikay Khandelwal, Naman Goyal, Vishrav Chaudhary, Guillaume Wenzek, Francisco Guzmán, Edouard Grave, Myle Ott, Luke Zettlemoyer, and Veselin Stoyanov. Unsupervised cross-lingual representation learning at scale, 2020. URL https://arxiv.org/abs/1911.02116.

Peter Dayan and Geoffrey E. Hinton. Using expectation-maximization for reinforcement learning. *Neural Computation*, 9(2):271–278, 1997. doi: 10.1162/neco.1997.9.2.271.

Longwei Ding, Anhao Zhao, Fanghua Ye, Ziyang Chen, and Xiaoyu Shen. From llms to lrms: Rethinking pruning for reasoning-centric models. *arXiv preprint arXiv:2601.18091*, 2026.

Brian S. Everitt and Anders Skrondal. *The Cambridge Dictionary of Statistics*. Cambridge University Press, Cambridge, UK, 4 edition, 2010.

Sicheng Feng, Gongfan Fang, Xinyin Ma, and Xinchao Wang. Efficient reasoning models: A survey, 2025. URL https://arxiv.org/abs/2504.10903.

Bofei Gao, Feifan Song, Zhe Yang, Zefan Cai, Yibo Miao, Qingxiu Dong, Lei Li, Chenghao Ma, Liang Chen, Runxin Xu, et al. Omni-math: A universal olympiad level mathematic benchmark for large language models. In *The Thirteenth International Conference on Learning Representations*, 2025.

Daya Guo, Dejian Yang, Haowei Zhang, Junxiao Song, Peiyi Wang, Qihao Zhu, Runxin Xu, Ruoyu Zhang, Shirong Ma, Xiao Bi, Xiaokang Zhang, Xingkai Yu, Yu Wu, Z. F. Wu, Zhibin Gou, Zhihong Shao, Zhuoshu Li, Ziyi Gao, Aixin Liu, Bing Xue, Bingxuan Wang, Bochao Wu, Bei Feng, Chengda Lu, Chenggang Zhao, Chengqi Deng, Chong Ruan, Damai Dai, Deli Chen, Dongjie Ji, Erhang Li, Fangyun Lin, Fucong Dai, Fuli Luo, Guangbo Hao, Guanting Chen, Guowei Li, H. Zhang, Hanwei Xu, Honghui Ding, Huazuo Gao, Hui Qu, Hui Li, Jianzhong Guo, Jiashi Li, Jingchang Chen, Jingyang Yuan, Jinhao Tu, Junjie Qiu, Junlong Li, J. L. Cai, Jiaqi Ni, Jian Liang, Jin Chen, Kai Dong, Kai Hu, Kaichao You, Kaige Gao, Kang Guan, Kexin Huang, Kuai Yu, Lean Wang, Lecong Zhang, Liang Zhao, Litong Wang, Liyue Zhang, Lei Xu, Leyi Xia, Mingchuan Zhang, Minghua Zhang, Minghui Tang, Mingxu Zhou, Meng Li, Miaojun Wang, Mingming Li, Ning Tian, Panpan Huang, Peng Zhang, Qiancheng Wang, Qinyu Chen, Qiushi Du, Ruiqi Ge, Ruisong Zhang, Ruizhe Pan, Runji Wang, R. J. Chen, R. L. Jin, Ruyi Chen, Shanghao Lu, Shangyan Zhou, Shanhuang Chen, Shengfeng Ye, Shiyu Wang, Shuiping Yu, Shunfeng Zhou, Shuting Pan, S. S. Li, Shuang Zhou, Shaoqing Wu, Tao Yun, Tian Pei, Tianyu Sun, T. Wang, Wangding Zeng, Wen Liu, Wenfeng Liang, Wenjun Gao, Wenqin Yu, Wentao Zhang, W. L. Xiao, Wei An, Xiaodong Liu, Xiaohan Wang, Xiaokang Chen, Xiaotao Nie, Xin Cheng, Xin Liu, Xin Xie, Xingchao Liu, Xinyu Yang, Xinyuan Li, Xuecheng Su, Xuheng Lin, X. Q. Li, Xiangyue Jin, Xiaojin Shen, Xiaosha Chen, Xiaowen Sun, Xiaoxiang Wang, Xinnan Song, Xinyi Zhou, Xianzu Wang, Xinxia Shan, Y. K. Li, Y. Q. Wang, Y. X. Wei, Yang Zhang, Yanhong Xu, Yao Li, Yao Zhao, Yaofeng Sun, Yaohui Wang, Yi Yu, Yichao Zhang, Yifan Shi, Yiliang Xiong, Ying He, Yishi Piao, Yisong Wang, Yixuan Tan, Yiyang Ma, Yiyuan Liu, Yongqiang Guo, Yuan Ou, Yuduan Wang, Yue Gong, Yuheng Zou, Yujia He, Yunfan Xiong, Yuxiang Luo, Yuxiang You, Yuxuan Liu, Yuyang Zhou, Y. X. Zhu, Yanping Huang, Yaohui Li, Yi Zheng, Yuchen Zhu, Yunxian Ma, Ying Tang, Yukun Zha, Yuting Yan, Z. Z. Ren, Zehui Ren, Zhangli Sha, Zhe Fu, Zhean Xu, Zhenda Xie, Zhengyan Zhang, Zhewen Hao, Zhicheng Ma, Zhigang Yan, Zhiyu Wu, Zihui Gu, Zijia Zhu, Zijun Liu, Zilin Li, Ziwei Xie, Ziyang Song, Zizheng Pan, Zhen Huang, Zhipeng Xu, Zhongyu Zhang, and Zhen Zhang. Deepseek-r1 incentivizes reasoning in llms through reinforcement learning. *Nature*, 645 (8081):633–638, September 2025. ISSN 1476-4687. doi: 10.1038/s41586-025-09422-z. URL http://dx.doi.org/10.1038/s41586-025-09422-z.

Tingxu Han, Zhenting Wang, Chunrong Fang, Shiyu Zhao, Shiqing Ma, and Zhenyu Chen. Token-budget-aware llm reasoning, 2025. URL https://arxiv.org/abs/2412.18547.

Bingxiang He, Zekai Qu, Zeyuan Liu, Yinghao Chen, Yuxin Zuo, Cheng Qian, Kaiyan Zhang, Weize Chen, Chaojun Xiao, Ganqu Cui, et al. Justrl: Scaling a 1.5 b llm with a simple rl recipe. *arXiv preprint arXiv:2512.16649*, 2025.

Bairu Hou, Yang Zhang, Jiabao Ji, Yujian Liu, Kaizhi Qian, Jacob Andreas, and Shiyu Chang. Thinkprune: Pruning long chain-of-thought of llms via reinforcement learning, 2025. URL https://arxiv.org/abs/2504.01296.

Jonas Hübotter, Frederike Lübeck, Lejs Behric, Anton Baumann, Marco Bagatella, Daniel Marta, Ido Hakimi, Idan Shenfeld, Thomas Kleine Buening, Carlos Guestrin, and Andreas Krause. Reinforcement learning via self-distillation, 2026. URL https://arxiv.org/abs/2601.20802.

Natasha Jaques, Asma Ghandeharioun, Judy Hanwen Shen, Craig Ferguson, Agata Lapedriza, Noah Jones, Shixiang Gu, and Rosalind Picard. Way off-policy batch deep reinforcement learning of implicit human preferences in dialog, 2019. URL https://arxiv.org/abs/1907.00456.

Nan Jiang and Lihong Li. Doubly robust off-policy value evaluation for reinforcement learning. In Maria Florina Balcan and Kilian Q. Weinberger (eds.), *Proceedings of The 33rd International Conference on Machine Learning*, volume 48 of *Proceedings of Machine Learning Research*, pp. 652–661, New York, New York, USA, 20–22 Jun 2016. PMLR. URL https://proceedings.mlr.press/v48/jiang16.html.

H. Kahn and A. W. Marshall. Methods of reducing sample size in monte carlo computations. *Operations Research*, 1(5):263–278, 1953. doi: 10.1287/opre.1.5.263.

Woosuk Kwon, Zhuohan Li, Siyuan Zhuang, Ying Sheng, Lianmin Zheng, Cody Hao Yu, Joseph E. Gonzalez, Hao Zhang, and Ion Stoica. Efficient memory management for large language model serving with pagedattention, 2023. URL https://arxiv.org/abs/2309.06180.

Gengyang Li, Wang Cai, Yifeng Gao, and Yunfang Wu. Syncthink: A training-free strategy to align inference termination with reasoning saturation, 2026. URL https://arxiv.org/abs/2601.03649.

Hanyu Li, Jiangshan Duo, Bofei Gao, Hailin Zhang, Sujian Li, Xiaotie Deng, and Liang Zhao. Reinforcement learning for chain of thought compression with one-domain-to-all generalization, 2025a. URL https://arxiv.org/abs/2601.06052.

Zeju Li, Jianyuan Zhong, Ziyang Zheng, Xiangyu Wen, Zhijian Xu, Yingying Cheng, Fan Zhang, and Qiang Xu. Compressing chain-of-thought in llms via step entropy, 2025b. URL https://arxiv.org/abs/2508.03346.

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

Shih-Yang Liu, Xin Dong, Ximing Lu, Shizhe Diao, Mingjie Liu, Min-Hung Chen, Hongxu Yin, Yu-Chiang Frank Wang, Kwang-Ting Cheng, Yejin Choi, Jan Kautz, and Pavlo Molchanov. Dler: Doing length penalty right - incentivizing more intelligence per token via reinforcement learning, 2025a. URL https://arxiv.org/abs/2510.15110.

Shih-Yang Liu, Xin Dong, Ximing Lu, Shizhe Diao, Peter Belcak, Mingjie Liu, Min-Hung Chen, Hongxu Yin, Yu-Chiang Frank Wang, Kwang-Ting Cheng, Yejin Choi, Jan Kautz, and Pavlo Molchanov. Gdpo: Group reward-decoupled normalization policy optimization for multi-reward rl optimization, 2026. URL https://arxiv.org/abs/2601.05242.

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, 2025b. URL https://arxiv.org/abs/2505.15612.

Zichen Liu, Changyu Chen, Wenjun Li, Penghui Qi, Tianyu Pang, Chao Du, Wee Sun Lee, and Min Lin. Understanding r1-zero-like training: A critical perspective, 2025c. URL https://arxiv.org/abs/2503.20783.

Kevin Lu and Thinking Machines Lab. On-policy distillation. *Thinking Machines Lab: Connectionism*, 2025. doi: 10.64434/tml.20251026. https://thinkingmachines.ai/blog/on-policy-distillation.

Haotian Luo, Li Shen, Haiying He, Yibo Wang, Shiwei Liu, Wei Li, Naiqiang Tan, Xiaochun Cao, and Dacheng Tao. O1-pruner: Length-harmonizing fine-tuning for o1-like reasoning pruning, 2025a. URL https://arxiv.org/abs/2501.12570.

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

Xingtai Lv, Yuxin Zuo, Youbang Sun, Hongyi Liu, Yuntian Wei, Zhekai Chen, Xuekai Zhu, Kaiyan Zhang, Bingning Wang, Ning Ding, and Bowen Zhou. Towards a unified view of large language model post-training, 2026. URL https://arxiv.org/abs/2509.04419.

Wenjie Ma, Jingxuan He, Charlie Snell, Tyler Griggs, Sewon Min, and Matei Zaharia. Reasoning models can be effective without thinking, 2025. URL https://arxiv.org/abs/2504.09858.

Alberto Maria Metelli, Matteo Papini, Francesco Faccio, and Marcello Restelli. Policy optimization via importance sampling. In S. Bengio, H. Wallach, H. Larochelle, K. Grauman, N. Cesa-Bianchi, and R. Garnett (eds.), *Advances in Neural Information Processing Systems*, volume 31. Curran Associates, Inc., 2018. URL https://proceedings.neurips.cc/paper_files/paper/2018/file/6aed000af86a084f9cb0264161e29dd3-Paper.pdf.

Yingqian Min, Zhipeng Chen, Jinhao Jiang, Jie Chen, Jia Deng, Yiwen Hu, Yiru Tang, Jiapeng Wang, Xiaoxue Cheng, Huatong Song, Wayne Xin Zhao, Zheng Liu, Zhongyuan Wang, and Ji-Rong Wen. Imitate, explore, and self-improve: A reproduction report on slow-thinking reasoning systems, 2024. URL https://arxiv.org/abs/2412.09413.

Youssef Mroueh, Nicolas Dupuis, Brian Belgodere, Apoorva Nitsure, Mattia Rigotti, Kristjan Greenewald, Jiri Navratil, Jerret Ross, and Jesus Rios. Revisiting group relative policy optimization: Insights into on-policy and off-policy training, 2025. URL https://arxiv.org/abs/2505.22257.

Tergel Munkhbat, Namgyu Ho, Seo Hyun Kim, Yongjin Yang, Yujin Kim, and Se-Young Yun. Self-training elicits concise reasoning in large language models, 2025. URL https://arxiv.org/abs/2502.20122.

Zhuoshi Pan, Qianhui Wu, Huiqiang Jiang, Menglin Xia, Xufang Luo, Jue Zhang, Qingwei Lin, Victor Rühle, Yuqing Yang, Chin-Yew Lin, H. Vicky Zhao, Lili Qiu, and Dongmei Zhang. Llmlingua-2: Data distillation for efficient and faithful task-agnostic prompt compression, 2024. URL https://arxiv.org/abs/2403.12968.

Shishir G Patil, Huanzhi Mao, Fanjia Yan, Charlie Cheng-Jie Ji, Vishnu Suresh, Ion Stoica, and Joseph E. Gonzalez. The berkeley function calling leaderboard (BFCL): From tool use to agentic evaluation of large language models. In Aarti Singh, Maryam Fazel, Daniel Hsu, Simon Lacoste-Julien, Felix Berkenkamp, Tegan Maharaj, Kiri Wagstaff, and Jerry Zhu (eds.), *Proceedings of the 42nd International Conference on Machine Learning*, volume 267 of *Proceedings of Machine Learning Research*, pp. 48371–48392. PMLR, 13–19 Jul 2025. URL https://proceedings.mlr.press/v267/patil25a.html.

Chongli Qin and Jost Tobias Springenberg. Supervised fine tuning on curated data is reinforcement learning (and can be improved), 2025. URL https://arxiv.org/abs/2507.12856.

Nathanaël Carraz Rakotonirina, Ren Pang, Neha Anna John, Michael Bohlke-Schneider, and Momchil Hardalov. Correct, concise and complete: Multi-stage training for adaptive reasoning, 2026. URL https://arxiv.org/abs/2601.02972.

John Schulman. Approximating kl divergence. http://joschu.net/blog/kl-approx.html, 2020.

John Schulman, Sergey Levine, Pieter Abbeel, Michael Jordan, and Philipp Moritz. Trust region policy optimization. In Francis Bach and David Blei (eds.), *Proceedings of the 32nd International Conference on Machine Learning*, volume 37 of *Proceedings of Machine Learning Research*, pp. 1889–1897, Lille, France, 07–09 Jul 2015. PMLR. URL https://proceedings.mlr.press/v37/schulman15.html.

John Schulman, Filip Wolski, Prafulla Dhariwal, Alec Radford, and Oleg Klimov. Proximal policy optimization algorithms, 2017. URL https://arxiv.org/abs/1707.06347.

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.

Idan Shenfeld, Mehul Damani, Jonas Hübotter, and Pulkit Agrawal. Self-distillation enables continual learning, 2026. URL https://arxiv.org/abs/2601.19897.

Mohammad Shoeybi, Mostofa Patwary, Raul Puri, Patrick LeGresley, Jared Casper, and Bryan Catanzaro. Megatron-lm: Training multi-billion parameter language models using model parallelism, 2020. URL https://arxiv.org/abs/1909.08053.

Janvijay Singh and Dilek Hakkani-Tür. Do llms encode functional importance of reasoning tokens?, 2026. URL https://arxiv.org/abs/2601.03066.

Nisan Stiennon, Long Ouyang, Jeff Wu, Daniel M. Ziegler, Ryan Lowe, Chelsea Voss, Alec Radford, Dario Amodei, and Paul Christiano. Learning to summarize from human feedback, 2022. URL https://arxiv.org/abs/2009.01325.

Xinyu Tang, Yuliang Zhan, Zhixun Li, Wayne Xin Zhao, Zhenduo Zhang, Zujie Wen, Zhiqiang Zhang, and Jun Zhou. Rethinking sample polarity in reinforcement learning with verifiable rewards, 2025. URL https://arxiv.org/abs/2512.21625.

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.

Hu Wang, Congbo Ma, Ian Reid, and Mohammad Yaqub. Kalman filter enhanced grpo for reinforcement learning-based language model reasoning, 2025a. URL https://arxiv.org/abs/2505.07527.

Xi Wang, James McInerney, Lequn Wang, and Nathan Kallus. Entropy after ⟨/Think⟩ for reasoning model early exiting, 2025b. URL https://arxiv.org/abs/2509.26522.

Jason Wei, Xuezhi Wang, Dale Schuurmans, Maarten Bosma, Brian Ichter, Fei Xia, Ed Chi, Quoc Le, and Denny Zhou. Chain-of-thought prompting elicits reasoning in large language models, 2023. URL https://arxiv.org/abs/2201.11903.

Ronald J Williams. Simple statistical gradient-following algorithms for connectionist reinforcement learning. *Machine learning*, 8(3):229–256, 1992.

Tong Wu, Chong Xiang, Jiachen T. Wang, G. Edward Suh, and Prateek Mittal. Effectively controlling reasoning models through thinking intervention, 2025a. URL https://arxiv.org/abs/2503.24370.

Yongliang Wu, Yizhou Zhou, Zhou Ziheng, Yingzhe Peng, Xinyu Ye, Xinting Hu, Wenbo Zhu, Lu Qi, Ming-Hsuan Yang, and Xu Yang. On the generalization of sft: A reinforcement learning perspective with reward rectification, 2025b. URL https://arxiv.org/abs/2508.05629.

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

Fengli Xu, Qianyue Hao, Zefang Zong, Jingwei Wang, Yunke Zhang, Jingyi Wang, Xiaochong Lan, Jiahui Gong, Tianjian Ouyang, Fanjin Meng, Chenyang Shao, Yuwei Yan, Qinglong Yang, Yiwen Song, Sijian Ren, Xinyuan Hu, Yu Li, Jie Feng, Chen Gao, and Yong Li. Towards large reasoning models: A survey of reinforced reasoning with large language models, 2025a. URL https://arxiv.org/abs/2501.09686.

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

An Yang, Anfeng Li, Baosong Yang, Beichen Zhang, Binyuan Hui, Bo Zheng, Bowen Yu, Chang Gao, Chengen Huang, Chenxu Lv, Chujie Zheng, Dayiheng Liu, Fan Zhou, Fei Huang, Feng Hu, Hao Ge, Haoran Wei, Huan Lin, Jialong Tang, Jian Yang, Jianhong Tu, Jianwei Zhang, Jianxin Yang, Jiaxi Yang, Jing Zhou, Jingren Zhou, Junyang Lin, Kai Dang, Keqin Bao, Kexin Yang, Le Yu, Lianghao Deng, Mei Li, Mingfeng Xue, Mingze Li, Pei Zhang, Peng Wang, Qin Zhu, Rui Men, Ruize Gao, Shixuan Liu, Shuang Luo, Tianhao Li, Tianyi Tang, Wenbiao Yin, Xingzhang Ren, Xinyu Wang, Xinyu Zhang, Xuancheng Ren, Yang Fan, Yang Su, Yichang Zhang, Yinger Zhang, Yu Wan, Yuqiong Liu, Zekun Wang, Zeyu Cui, Zhenru Zhang, Zhipeng Zhou, and Zihan Qiu. Qwen3 technical report, 2025a. URL https://arxiv.org/abs/2505.09388.

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, 2025b. URL https://arxiv.org/abs/2504.15895.

Fengkai Yang, Zherui Chen, Xiaohan Wang, Xiaodong Lu, Jiajun Chai, Guojun Yin, Wei Lin, Shuai Ma, Fuzhen Zhuang, Deqing Wang, Yaodong Yang, Jianxin Li, and Yikun Ban. Your group-relative advantage is biased, 2026. URL https://arxiv.org/abs/2601.08521.

Anqi Zhang, Yulin Chen, Jane Pan, Chen Zhao, Aurojit Panda, Jinyang Li, and He He. Reasoning models know when they're right: Probing hidden states for self-verification, 2025a. URL https://arxiv.org/abs/2504.05419.

Ruofan Zhang, Bin Xia, Zhen Cheng, Cairen Jian, Minglun Yang, Ngai Wong, and Yuan Cheng. Dart: Difficulty-adaptive reasoning truncation for efficient large language models, 2025b. URL https://arxiv.org/abs/2511.01170.

Siyan Zhao, Zhihui Xie, Mengchen Liu, Jing Huang, Guan Pang, Feiyu Chen, and Aditya Grover. Self-distilled reasoner: On-policy self-distillation for large language models, 2026. URL https://arxiv.org/abs/2601.18734.

Weixiang Zhao, Jiahe Guo, Yang Deng, Xingyu Sui, Yulin Hu, Yanyan Zhao, Wanxiang Che, Bing Qin, Tat-Seng Chua, and Ting Liu. Exploring and exploiting the inherent efficiency within large reasoning models for self-guided efficiency enhancement, 2025. URL https://arxiv.org/abs/2506.15647.

Wenhong Zhu, Ruobing Xie, Rui Wang, Xingwu Sun, Di Wang, and Pengfei Liu. Proximal supervised fine-tuning, 2025. URL https://arxiv.org/abs/2508.17784.

APPENDIX

A RELATED WORK

Efficient reasoning Efficient reasoning methods can be broadly grouped into three categories: *training-free methods, SFT-based methods, and RL-based methods*. (i) **Training-free** approaches aim to improve reasoning efficiency purely at inference time without additional training. Works in this line can be further categorized into prompt-based techniques, early-exit strategies, and activation steering. Prompt-based methods encourage models to produce more concise reasoning through explicit instructional prompts (Xu et al., 2025b; Ma et al., 2025; Han et al., 2025). Early-exit strategies terminate the reasoning process based on carefully designed confidence or uncertainty metrics when further computation is deemed unnecessary (Yang et al., 2025b; Wang et al., 2025b; Li et al., 2026). Activation steering, by contrast, intervenes directly in the model's internal activations to bias generation toward more efficient reasoning trajectories (Wu et al., 2025a; Zhang et al., 2025a; Zhao et al., 2025). (ii) **SFT-based** methods construct compressed reasoning datasets and fine-tune models on top of them. Concise Reasoning Self-Training (CRST) (Munkhbat et al., 2025) fine-tunes models on self-generated concise reasoning paths obtained via best-of-N sampling. TokenSkip (Xia et al., 2025) prunes semantically unimportant tokens from CoT trajectories to produce compressed training data with varying compression ratios. StepEntropy (Li et al., 2025b) replaces low-entropy steps in self-generated reasoning trajectories with a special `[skip]` token. DART (Zhang et al., 2025b) distills concise reasoning patterns from stronger models. Singh & Hakkani-Tür (2026) proposes an iterative pruning approach that removes reasoning tokens whose removal minimally degrades the model likelihood under a specified objective. (iii) **RL-based** methods optimize reasoning efficiency through policy optimization. KIMI k1.5 (Team et al., 2025) promotes shorter responses and penalizes longer responses among correct outputs, while explicitly penalizing long responses with incorrect answers. ThinkPrune (Hou et al., 2025) assigns a unit reward to responses that are both correct and within a predefined maximum length. O1-Pruner (Luo et al., 2025a) introduces a length-harmonizing reward defined as the ratio between the reasoning length of a reference model and that of the current policy. L1 (Aggarwal & Welleck, 2025) trains models to satisfy user-specified length constraints. ER-RL (Arora & Zanette, 2025) assigns rewards only to correct responses, with shorter outputs receiving higher rewards. LASER (Liu et al., 2025b) employs a step-function reward based on a target length and dynamically adjusts this target during training. DLER (Liu et al., 2025a) introduces a training recipe that combines batch-wise reward normalization, higher clipping thresholds, dynamic sampling, and a simple truncation length penalty. Li et al. (2025a) proposes a mastery-gated, sample-level, soft RL compression scheme that penalizes long rollouts only when the model already solves the problem and has produced a shorter rollout.

Supervised fine-tuning and reinforcement learning The connection between supervised learning and reinforcement learning through importance weighting has deep theoretical roots (Kahn & Marshall, 1953; Dayan & Hinton, 1997). From the perspective of policy optimization, importance sampling provides a principled mechanism for leveraging data collected under one policy to optimize another by appropriately reweighting samples (Metelli et al., 2018; Jiang & Li, 2016). A well-known challenge of this approach is that importance weights may become highly variable when the behavior and target policies differ substantially (Andradóttir et al., 1995). To control this instability, trust-region methods (Schulman et al., 2015) and proximal policy optimization (Schulman et al., 2017) explicitly restrict the magnitude of policy updates relative to a reference policy. Motivated by these ideas, Recent work on LLM fine-tuning has proposed a range of techniques, including importance-weighted supervised objectives (Qin & Springenberg, 2025), proximal variants of supervised fine-tuning (Zhu et al., 2025), probability-based reweighting schemes in dynamic fine-tuning (Wu et al., 2025b), and unified perspectives that reconcile supervised and reinforcement learning paradigms (Lv et al., 2026). In contrast to prior approaches, our method anchors the auxiliary distribution directly to the base model, extending importance-weighted formulations with an explicit stabilization mechanism while maintaining their theoretical guarantees. In contrast to these approaches, we place SFT in an on-policy data regime and directly connect SFT and RL.

Prompt Template

```
<|begin_of_sentence|><|User|>{question} Please reason step by
step, and put your final answer within \\boxed{}.
<|Assistant|><think>
```

Figure 11: Prompt template used for both training and evaluation.

CoD Prompt Template

```
<|begin_of_sentence|><|User|>{question} Please reason step by
step, but only keep a minimum draft for each thinking step,
with 5 words at most, and put your final answer within
\\boxed{}.
<|Assistant|><think>
```

Figure 12: Prompt template used for CoD.

B Unified Reward Formulations

As discussed in section 2, existing RL-based efficient reasoning rewards share a unified form:

$$R_{\text{Eff}}(o_i \mid q) = R_{\text{Acc}}(o_i \mid q) + \gamma(o_i \mid q)\, R_{\text{Len}}(o_i \mid q), \tag{11}$$

Below we summarize how representative prior methods instantiate $R_{\text{Acc}}(o_i \mid q)$, $\gamma(o_i \mid q)$, and $R_{\text{Len}}(o_i \mid q)$. We use $L(o_i)$ to denote the generated token length of o_i.

ER-RL (Arora & Zanette, 2025)

$$R_{\text{Acc}}(o_i \mid q) = R_i, \tag{12}$$

$$\gamma(o_i \mid q) = \mathbb{I}\big(R_i = 1\big), \tag{13}$$

$$R_{\text{Len}}(o_i \mid q) = -\alpha \cdot \sigma\left(\frac{L(o_i) - \text{Mean}(L)}{\text{Std}(L)}\right), \tag{14}$$

where $R_i \in \{0, 1\}$ rewards correctness, $\alpha \in [0, 1)$ scales the penalty, and σ is the sigmoid.

Kimi-k1.5 (Team et al., 2025)

$$R_{\text{Acc}}(o_i \mid q) = R_i, \tag{15}$$

$$\gamma(o_i \mid q) = 1, \tag{16}$$

$$R_{\text{Len}}(o_i \mid q) = \begin{cases} 0.5 - \dfrac{L(o_i) - L_{\min}}{L_{\max} - L_{\min}}, & \mathbb{I}(R_i = 1), \\ \min\left(0,\, 0.5 - \dfrac{L(o_i) - L_{\min}}{L_{\max} - L_{\min}}\right), & \mathbb{I}(R_i = 0), \end{cases} \tag{17}$$

where $L_{\min} = \min_i L(o_i)$ and $L_{\max} = \max_i L(o_i)$.

L1-Exact (Aggarwal & Welleck, 2025)

$$R_{\text{Acc}}(o_i \mid q) = R_i, \tag{18}$$

$$\gamma(o_i \mid q) = 1, \tag{19}$$

$$R_{\text{Len}}(o_i \mid q) = -\alpha \cdot |L(o_i) - L_T|, \tag{20}$$

where L_T is a user-specific target length.

L1-Max (Aggarwal & Welleck, 2025)

$$R_{\text{Acc}}(o_i \mid q) = 0, \tag{21}$$
$$\gamma(o_i \mid q) = \mathbb{I}\big(R_i = 1\big), \tag{22}$$
$$R_{\text{Len}}(o_i \mid q) = \text{clip}(\alpha \cdot (L(o_i) - L_T) + \delta,\ 0,\ 1)\,, \tag{23}$$

where L_T is a user-specific maximum length constraint.

LASER-DE (Liu et al., 2025b)

$$R_{\text{Acc}}(o_i \mid q) = R_i, \tag{24}$$
$$\gamma(o_i \mid q) = 1, \tag{25}$$
$$R_{\text{Len}}(o_i \mid q) = \alpha\,\mathbb{I}(R_i = 1)\mathbb{I}(L(o_i) \le L_A) + \alpha\,\mathbb{I}(R_i = 0)\mathbb{I}(L(o_i) > L_A), \tag{26}$$

where L_A denotes a dynamically adjusted length threshold.

Mastery-Gated Length Penalty (Li et al., 2025a)

$$R_{\text{Acc}}(o_i \mid q) = R_i, \tag{27}$$
$$\gamma(o_i \mid q) = \mathbb{I}\big(\hat{p}(q) = 1\big), \tag{28}$$
$$R_{\text{Len}}(o_i \mid q) = \begin{cases} 0, & L(o_i) \le L_{\text{start}}(q), \\ -1, & L(o_i) > L_{\max}(q), \\ -\dfrac{L(o_i) - L_{\text{start}}(q)}{L_{\max}(q) - L_{\text{start}}(q)}, & L_{\text{start}}(q) < L(o_i) \le L_{\max}(q), \end{cases} \tag{29}$$

where $R_i \in \{0, 1\}$ is a binary correctness reward, $\hat{p}(q) = \frac{1}{N}\sum_{i=1}^{N} R_i$ denotes the correctness rate over N rollouts for question q, $L_{\text{start}}(q)$ is the median generated token length among correct rollouts, and $L_{\max}(q)$ is the maximum generated token length among correct rollouts.

C Derivation of the GRPO Gradient

We derive the gradient of the GRPO objective used in the main text. We consider $(q, a) \sim \mathcal{D}$ and G rollouts $\{o_i\}_{i=1}^{G}$ sampled from a fixed rollout snapshot $\pi_{\theta_{\text{old}}}(\cdot \mid q)$, where $o_i = (o_{i,1}, \ldots, o_{i,|o_i|})$.

GRPO objective. The GRPO objective is

$$J_{\text{GRPO}}(\theta) = \mathbb{E}_{(q,a)\sim\mathcal{D},\,\{o_i\}_{i=1}^{G}\sim\pi_{\theta_{\text{old}}}(\cdot|q)}\Bigg[\frac{1}{G}\sum_{i=1}^{G}\frac{1}{|o_i|}\sum_{t=1}^{|o_i|}\min\Big(r_{i,t}(\theta)\hat{A}_{i,t}, \\ \text{clip}(r_{i,t}(\theta), 1-\epsilon, 1+\epsilon)\hat{A}_{i,t}\Big) - \beta\, D_{\text{KL}}(\pi_\theta \,\|\, \pi_{\text{ref}})\Bigg], \tag{30}$$

where $r_{i,t}(\theta) = \pi_\theta(o_{i,t} \mid q, o_{i,<t}) / \pi_{\theta_{\text{old}}}(o_{i,t} \mid q, o_{i,<t})$. The KL term is estimated using the following unbiased per-token estimator (Schulman, 2020):

$$D_{\text{KL},\,i,t}(\theta) = \frac{\pi_{\text{ref}}(o_{i,t} \mid o_{i,<t})}{\pi_\theta(o_{i,t} \mid q, o_{i,<t})} - \log\frac{\pi_{\text{ref}}(o_{i,t} \mid o_{i,<t})}{\pi_\theta(o_{i,t} \mid q, o_{i,<t})} - 1. \tag{31}$$

Step 1: Differentiate the original objective. Since rollouts are sampled from the fixed snapshot $\pi_{\theta_{\text{old}}}$, the sampling distribution does not depend on θ, so we can move ∇_θ inside the expectation:

$$\nabla_\theta J_{\text{GRPO}}(\theta) = \mathbb{E}\left[\frac{1}{G}\sum_{i=1}^{G}\frac{1}{|o_i|}\sum_{t=1}^{|o_i|}\nabla_\theta\,\ell_{i,t}(\theta)\right], \tag{32}$$

where

$$\ell_{i,t}(\theta) \triangleq \min\Big(r_{i,t}(\theta)\hat{A}_{i,t},\ \text{clip}(r_{i,t}(\theta), 1-\epsilon, 1+\epsilon)\hat{A}_{i,t}\Big) - \beta\, D_{\text{KL},\,i,t}(\theta). \tag{33}$$

Step 2: Simplification under the single-update assumption. To simplify the analysis, we consider a single policy update after each exploration stage, so that $\pi_{\theta_{\text{old}}} = \pi_\theta$. In this case, the PPO-style $\min(\cdot)$ and clipping operations can be removed:

$$\min\Big(r_{i,t}(\theta)\hat{A}_{i,t},\ \text{clip}(r_{i,t}(\theta), 1-\epsilon, 1+\epsilon)\hat{A}_{i,t}\Big) = r_{i,t}(\theta)\hat{A}_{i,t}. \tag{34}$$

Taking the gradient of $r_{i,t}(\theta)\hat{A}_{i,t}$ with respect to θ, we obtain

$$\nabla_\theta\big(r_{i,t}(\theta)\hat{A}_{i,t}\big) = \hat{A}_{i,t}\nabla_\theta r_{i,t}(\theta) = \hat{A}_{i,t}\nabla_\theta \log \pi_\theta(o_{i,t} \mid q, o_{i,<t}), \tag{35}$$

where

$$\begin{aligned}
\nabla_\theta r_{i,t}(\theta) &= \nabla_\theta \frac{\pi_\theta(o_{i,t} \mid q, o_{i,<t})}{\pi_{\theta_{\text{old}}}(o_{i,t} \mid q, o_{i,<t})} \\
&= \frac{1}{\pi_{\theta_{\text{old}}}(o_{i,t} \mid q, o_{i,<t})}\nabla_\theta \pi_\theta(o_{i,t} \mid q, o_{i,<t}) \\
&= \frac{\pi_\theta(o_{i,t} \mid q, o_{i,<t})}{\pi_{\theta_{\text{old}}}(o_{i,t} \mid q, o_{i,<t})}\nabla_\theta \log \pi_\theta(o_{i,t} \mid q, o_{i,<t}) \\
&= r_{i,t}(\theta)\nabla_\theta \log \pi_\theta(o_{i,t} \mid q, o_{i,<t}) \\
&= \nabla_\theta \log \pi_\theta(o_{i,t} \mid q, o_{i,<t}),
\end{aligned} \tag{36}$$

where the last equality uses $\pi_{\theta_{\text{old}}} = \pi_\theta$, hence $r_{i,t}(\theta) = 1$.

Step 3: Gradient of the KL estimator. Taking the gradient of $D_{\text{KL},\,i,t}(\theta)$ in (31), we obtain

$$\begin{aligned}
\nabla_\theta D_{\text{KL},\,i,t}(\theta) &= \nabla_\theta\left(\frac{\pi_{\text{ref}}(o_{i,t} \mid o_{i,<t})}{\pi_\theta(o_{i,t} \mid q, o_{i,<t})} - \log\frac{\pi_{\text{ref}}(o_{i,t} \mid o_{i,<t})}{\pi_\theta(o_{i,t} \mid q, o_{i,<t})} - 1\right) \\
&= \nabla_\theta\left(\frac{\pi_{\text{ref}}(o_{i,t} \mid o_{i,<t})}{\pi_\theta(o_{i,t} \mid q, o_{i,<t})} + \log \pi_\theta(o_{i,t} \mid q, o_{i,<t})\right) \\
&= -\frac{\pi_{\text{ref}}(o_{i,t} \mid o_{i,<t})}{\pi_\theta(o_{i,t} \mid q, o_{i,<t})^2}\nabla_\theta \pi_\theta(o_{i,t} \mid q, o_{i,<t}) + \nabla_\theta \log \pi_\theta(o_{i,t} \mid q, o_{i,<t}) \\
&= -\frac{\pi_{\text{ref}}(o_{i,t} \mid o_{i,<t})}{\pi_\theta(o_{i,t} \mid q, o_{i,<t})^2}\pi_\theta(o_{i,t} \mid q, o_{i,<t})\nabla_\theta \log \pi_\theta(o_{i,t} \mid q, o_{i,<t}) + \\
&\quad \nabla_\theta \log \pi_\theta(o_{i,t} \mid q, o_{i,<t}) \\
&= -\left(\frac{\pi_{\text{ref}}(o_{i,t} \mid o_{i,<t})}{\pi_\theta(o_{i,t} \mid q, o_{i,<t})} - 1\right)\nabla_\theta \log \pi_\theta(o_{i,t} \mid q, o_{i,<t}).
\end{aligned} \tag{37}$$

Step 4: Combine terms. Combining (35) and (37) in $\nabla_\theta \ell_{i,t}(\theta) = \nabla_\theta\big(r_{i,t}(\theta)\hat{A}_{i,t}\big) - \beta\nabla_\theta D_{\text{KL},\,i,t}(\theta)$, we obtain

$$\begin{aligned}
\nabla_\theta \ell_{i,t}(\theta) &= \hat{A}_{i,t}\nabla_\theta \log \pi_\theta(o_{i,t} \mid q, o_{i,<t}) \\
&\quad - \beta\Big(-\Big(\frac{\pi_{\text{ref}}(o_{i,t} \mid o_{i,<t})}{\pi_\theta(o_{i,t} \mid q, o_{i,<t})} - 1\Big)\nabla_\theta \log \pi_\theta(o_{i,t} \mid q, o_{i,<t})\Big) \\
&= \Big(\hat{A}_{i,t} + \beta\Big(\frac{\pi_{\text{ref}}(o_{i,t} \mid o_{i,<t})}{\pi_\theta(o_{i,t} \mid q, o_{i,<t})} - 1\Big)\Big)\nabla_\theta \log \pi_\theta(o_{i,t} \mid q, o_{i,<t}).
\end{aligned} \tag{38}$$

Substituting (38) into (32) yields

$$\begin{aligned}
\nabla_\theta J_{\text{GRPO}}(\theta) = \mathbb{E}_{(q,a)\sim\mathcal{D},\,\{o_i\}_{i=1}^{G}\sim\pi_{\theta_{\text{old}}}(\cdot|q)}\Bigg[\frac{1}{G}\sum_{i=1}^{G}\frac{1}{|o_i|}\sum_{t=1}^{|o_i|}\Big(\hat{A}_{i,t} + \\
\beta\Big(\frac{\pi_{\text{ref}}(o_{i,t} \mid o_{i,<t})}{\pi_\theta(o_{i,t} \mid q, o_{i,<t})} - 1\Big)\Big)\nabla_\theta \log \pi_\theta(o_{i,t} \mid q, o_{i,<t})\Bigg],
\end{aligned} \tag{39}$$

which matches the gradient expression reported in the main text.

D REINFORCE ALGORITHM

REINFORCE Williams (1992) is a Monte Carlo policy gradient method that directly optimizes a parameterized stochastic policy $\pi_\theta(a \mid s)$. We consider an episodic RL setting in which a trajectory $\tau = (s_0, a_0, r_0, s_1, a_1, r_1, \ldots, s_T)$ is generated by following π_θ. The discounted return from time step t is defined as $G_t = \sum_{k=t}^{T-1} \gamma^{k-t} r_k$, where $\gamma \in [0, 1]$ is the discount factor. The objective is to maximize the expected cumulative return

$$J(\theta) = \mathbb{E}_{\tau \sim \pi_\theta} \left[\sum_{t=0}^{T-1} \gamma^t r_t \right].$$

Using the log-derivative trick, the gradient of the objective can be written as

$$\nabla_\theta J(\theta) = \mathbb{E}_{\tau \sim \pi_\theta} \left[\sum_{t=0}^{T-1} \nabla_\theta \log \pi_\theta(a_t \mid s_t)\, G_t \right].$$

Approximating this expectation with Monte Carlo samples from complete episodes gives:

$$\theta \leftarrow \theta + \alpha \sum_{t=0}^{T-1} \nabla_\theta \log \pi_\theta(a_t \mid s_t)\, G_t,$$

where α denotes the learning rate.

E ON-POLICY SUPERVISED FINE-TUNING

Algorithm 1 presents the complete on-policy SFT training procedure.

F EXPERIMENTAL SETUP DETAILS

Dataset Details DeepScaleR (Luo et al., 2025b) is a large-scale collection of mathematical reasoning problems aggregated from multiple established sources, including AIME (1983–2023), AMC, Omni-Math (Gao et al., 2025), and STILL (Min et al., 2024).

Training Details During training, we use a group size of 8 and a global batch size of 64, and train for a single epoch. We adopt the AdamW optimizer with a learning rate of 1×10^{-7}.

Evaluation Details We evaluate the effectiveness of On-Policy SFT on five widely used mathematical reasoning benchmarks: GSM8K (Cobbe et al., 2021), MATH-500 (Lightman et al., 2023), AMC23 (AMC2023, 2023), AIME24 (AIME2024, 2024), and AIME25 (AIME2025, 2025). Following the recommendations of Guo et al. (2025), we adopt decoding hyperparameters with temperature set to 0.6 and top-p set to 0.95. The maximum number of output tokens is fixed at 32,768. For each question, we sample the model outputs N times to support multi-sample evaluation. The value of N is set to 64 for AMC23, AIME24, and AIME25, and to 16 for the remaining datasets. All inference is conducted on a single A800 GPU using the vLLM library (Kwon et al., 2023). Details of the prompt templates are provided in Appendix G.1.

Metrics We report five evaluation metrics: accuracy (**Acc**, in %), **Pass@**N (in %), average response token length (**Tok**), compression rate (**CR**, in %), and token efficiency (**Eff**). Acc is defined as the proportion of correct responses among all sampled model outputs in the evaluation set. Pass@N measures the fraction of problems for which at least one correct solution is produced among N independently sampled outputs, where N corresponds to the per-problem sampling budget used during evaluation. We report Pass@N to assess whether the training procedure preserves the model's ability to explore diverse solution paths across multiple sampling attempts. Tok measures the average number of generated tokens per sampled response. CR is defined as the ratio between the average number of tokens in the generated response and that of the original CoT. Eff is defined as the ratio of accuracy to average response length (Eff = Acc / Tok, in %), serving as an indicator of the correctness and reasoning efficiency trade-off.

Algorithm 1 On-Policy Supervised Fine-Tuning

Require: Training set $\mathcal{D} = \{(q, a)\}$; initial policy π_θ; rollouts per prompt G; length limit L; batch size B; learning rate η; total steps T.
Ensure: Updated policy parameters θ.

for $k = 1$ to T **do**
  Sample a minibatch $\mathcal{B} = \{(q_b, a_b)\}_{b=1}^{B}$ from $\mathcal{D}$.
  Set $\theta_{\text{old}} \leftarrow \theta$. ▷ on-policy snapshot
  Initialize $\mathcal{C}_L \leftarrow \emptyset$ and $M \leftarrow 1$. ▷ valid rollouts and max valid length
  for $b = 1$ to B **do**
    for $i = 1$ to G **do**
      Sample a rollout $o_{b,i} \sim \pi_{\theta_{\text{old}}}(\cdot \mid q_b)$.
      if $\textsc{Correct}(o_{b,i}, a_b) \ \wedge \ |o_{b,i}| \le L$ **then**
        Add $(q_b, o_{b,i})$ to $\mathcal{C}_L$.
        $M \leftarrow \max(M, |o_{b,i}|)$.
      end if
    end for
  end for
  if $|\mathcal{C}_L| > 0$ **then**
    Compute the on-policy SFT objective over $\mathcal{C}_L$:
      $J(\theta) = \frac{1}{BG} \sum_{(q,o)\in\mathcal{C}_L} \frac{1}{M} \sum_{t=1}^{|o|} \log \pi_\theta(o_t \mid q, o_{<t})$.
    Compute loss $\mathcal{L}(\theta) \leftarrow -J(\theta)$.
    Update $\theta \leftarrow \theta - \eta \nabla_\theta \mathcal{L}(\theta)$.
  end if
end for

G Prompt Templates

G.1 On-Policy SFT Prompt Template

The prompt template is illustrated in Figure 11, which follows the design proposed in DeepSeek-R1 (Guo et al., 2025). This prompt template is used by all methods in our experiments except for CoD. For these methods, identical prompts are used for training and evaluation, avoiding potential distribution shifts from prompt mismatches and ensuring that performance gains can be attributed to On-Policy SFT rather than prompt engineering.

G.2 Chain-of-Draft (CoD) Prompt Template

We use the Chain-of-Draft prompt template (Xu et al., 2025b), shown in Figure 12.

H Baseline Implementation Details

TokenSkip Due to the limited context length of XLM-RoBERTa (Conneau et al., 2020), which serves as the backbone of LLMLingua-2 (Pan et al., 2024), the model only supports a maximum context window of 512 tokens. In our implementation, self-generated CoT sequences are first split into non-overlapping chunks of 512 tokens. LLMLingua is then applied independently to each chunk to identify semantically unimportant tokens, which are subsequently removed to construct the training dataset. After training, we evaluate compression ratios ranging from 0.1 to 0.9 in increments of 0.1, and select 0.7 as it provides the best performance–efficiency trade-off.

L1 we report the *L1-max* variant, which requires the output to be no longer than the specified target length, with the target length set to 3,500.

I COEFFICIENT OF VARIATION

Formally, for a given question q, we generate N responses, where N is set identically to the evaluation setup described in section 4. Let ℓ_{qj} denote the token length of the j-th response. The coefficient

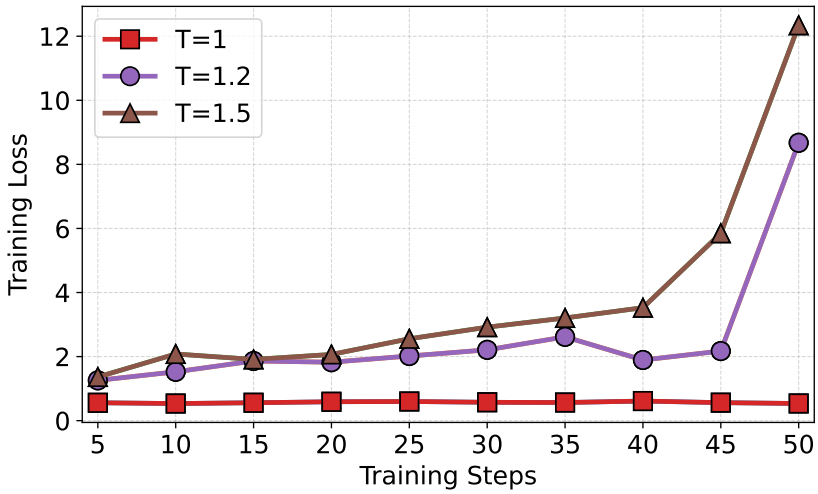

Figure 13: MATH-500 training loss for the 1.5B model at rollout temperatures 1.0, 1.2, and 1.5.

of variation of generation lengths for question q is given by

$$\mathrm{NormStd}_q = \frac{\sqrt{\frac{1}{N}\sum_{j=1}^{N}\left(\ell_{qj} - \frac{1}{N}\sum_{j=1}^{N}\ell_{qj}\right)^2}}{\frac{1}{N}\sum_{j=1}^{N}\ell_{qj}}. \tag{40}$$

J On-Policy Requirement for Rollout Temperature

Let $z_{\theta_{\text{old}}}(y \mid q, o_{<t})$ denote the pre-softmax logit assigned by the model with parameters θ_{old} to token y, given the question q and the previously generated prefix $o_{<t}$. When rollouts are generated using temperature-scaled sampling, the token-level behavior policy at timestep t is defined as

$$\pi^{(T)}_{\theta_{\text{old}}}(o_t \mid q, o_{<t}) = \frac{\exp(z_{\theta_{\text{old}}}(o_t \mid q, o_{<t})/T)}{\sum_y \exp(z_{\theta_{\text{old}}}(y \mid q, o_{<t})/T)}. \tag{41}$$

This token-level policy induces the following distribution over responses $o = (o_1, \ldots, o_{|o|})$:

$$\Pr_T(o \mid q) = \prod_{t=1}^{|o|} \pi^{(T)}_{\theta_{\text{old}}}(o_t \mid q, o_{<t}). \tag{42}$$

Therefore, when rollouts are sampled with temperature T, the expectation in Equation 10 is taken with respect to the distribution $\Pr_T(\cdot \mid q)$, rather than $\pi_{\theta_{\text{old}}}(\cdot \mid q)$.

When $T = 1$, the temperature-scaled token distribution reduces to the original model policy, i.e.,

$$\pi^{(1)}_{\theta_{\text{old}}}(o_t \mid q, o_{<t}) = \pi_{\theta_{\text{old}}}(o_t \mid q, o_{<t}), \tag{43}$$

and consequently $\Pr_1(o \mid q) = \pi_{\theta_{\text{old}}}(o \mid q)$. In contrast, for any $T \neq 1$,

$$\pi^{(T)}_{\theta_{\text{old}}}(\cdot \mid q, o_{<t}) \neq \pi_{\theta_{\text{old}}}(\cdot \mid q, o_{<t}), \tag{44}$$

which implies $\Pr_T(\cdot \mid q) \neq \pi_{\theta_{\text{old}}}(\cdot \mid q)$. Hence, when $T \neq 1$, rollouts come from a behavior policy different from $\pi_{\theta_{\text{old}}}$, making Equation 10 an off-policy objective.

K A Diagnostic Tool for Identifying Inefficient Tokens

Inspired by recent work on on-policy distillation (Lu & Lab, 2025; Shenfeld et al., 2026; Zhao et al., 2026; Hübotter et al., 2026), we introduce a principled diagnostic tool for identifying which

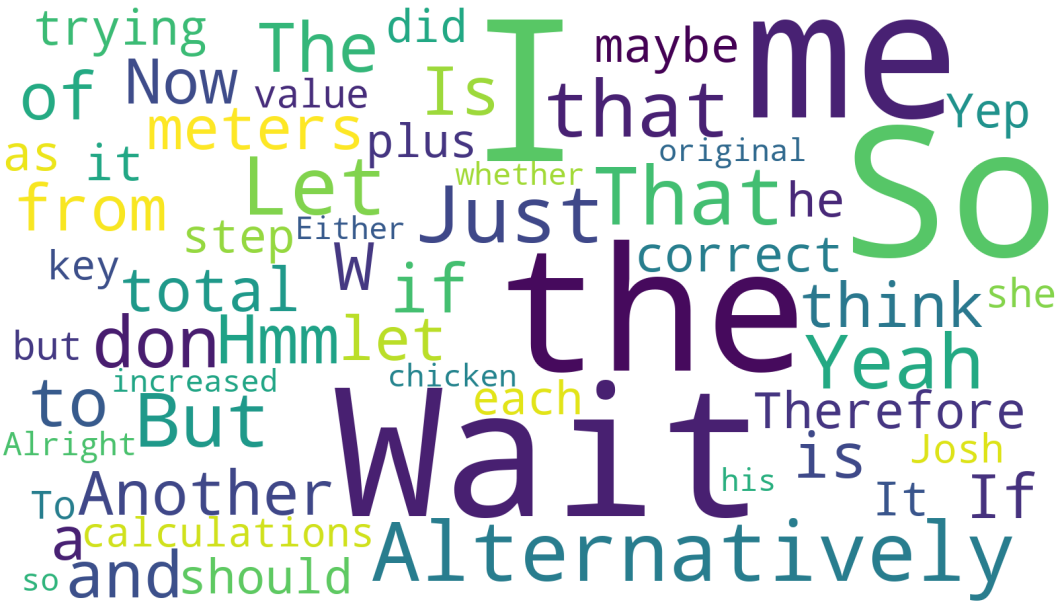

Figure 14: Next-token word cloud of the original model at positions with large KL divergence from the on-policy SFT–trained efficient model. Larger font sizes indicate greater divergence.

tokens in an original CoT are most likely to steer the model toward inefficient reasoning trajectories. Specifically, we first sample an original CoT from the base model, and then feed the same prefix (the prompt together with the sampled CoT prefix) into the efficient model obtained after on-policy SFT via teacher forcing. At each position, we measure the divergence between the next-token distributions of the original and efficient models conditioned on the same history. Let $x_{1:t}$ denote the shared prefix at position t. We denote the next-token distributions of the original and efficient models as $p_{\text{orig}}(\cdot \mid x_{1:t})$ and $p_{\text{eff}}(\cdot \mid x_{1:t})$, respectively. The token-level divergence is computed as

$$D_t = \mathrm{KL}(p_{\text{orig}}(\cdot \mid x_{1:t}) \,\|\, p_{\text{eff}}(\cdot \mid x_{1:t})) = \sum_{v \in \mathcal{V}} p_{\text{orig}}(v \mid x_{1:t}) \Big(\log p_{\text{orig}}(v \mid x_{1:t}) - \log p_{\text{eff}}(v \mid x_{1:t}) \Big). \tag{45}$$

A larger D_t indicates greater disagreement between the models' next-token distributions, pinpointing where the efficient model diverges most sharply from the original trajectory.

We visualize the next-token word cloud of the original model at positions with large KL divergence under a fixed prefix in Figure 14, where larger font sizes indicate greater disagreement with the efficient model. We observe that tokens such as "Wait", "Alternatively", and "Hmm", which often signal hesitation or shifts in reasoning, are consistently identified as inefficient. This suggests that excessive reflection or backtracking is not always required, and that a more confident reasoning trajectory may yield correct solutions with substantially lower latency. In addition, we uncover another intriguing pattern: first-person singular pronouns such as "I" and "me" also exhibit large divergence. In contrast, the efficient variant tends to prefer plural forms such as "we" and "us." We hypothesize that this behavior may be related to pretraining data distributions, where corpora involving collective expressions are often more concise, potentially offering a new perspective on achieving efficient reasoning. We include several question-level diagnostic examples in the Appendix L.

L Qualitative Examples of KL-Based Diagnostics

In this section, we present qualitative, question-level examples to illustrate how the proposed diagnostic identifies tokens associated with inefficient reasoning. We provide two concrete examples, each consisting of two complementary visualizations. For each question, we first visualize the token-level KL divergence between the original model and its on-policy SFT–trained efficient counterpart along the generated CoT. The divergence is computed at each token position under a fixed prefix,

Table 2: Additional BFCL tool-use and agentic-memory tasks.

Task	Size	Format	One-line description
BFCL Python Simple Function	400	AST	One function document is given, and exactly one function call is required.
BFCL Python Multiple Function	200	AST	The model selects one correct function call from 2–4 candidates.
BFCL Python Parallel Function	200	AST	The model must issue multiple function calls in parallel for one request.
BFCL Python Parallel Multiple Function	200	AST	Multiple candidate functions are given, and each may be called zero or more times.
BFCL Memory Key Value Store	155	–	A multi-turn memory task using explicit key-value storage and retrieval.
BFCL Memory Vector Store	155	–	A multi-turn memory task using semantic retrieval from a vector store.

Table 3: BFCL tool-use results on DeepSeek-R1-Distill-Qwen-1.5B and 7B. Overall is averaged over the displayed tool-use subsets.

Model	Variant	Non-Live Overall	Python Simple	Multiple	Parallel	Parallel Multiple
R1-Distill-1.5B	Original model	10.06%	18.75%	7.50%	12.00%	2.00%
	After on-policy SFT	11.06%	18.75%	9.00%	11.50%	5.00%
R1-Distill-7B	Original model	48.50%	62.00%	50.50%	53.00%	28.50%
	After on-policy SFT	49.00%	65.00%	52.00%	51.00%	28.00%

with larger values highlighted in darker red, indicating stronger disagreement between the two models. The resulting visualizations are shown in Figure 15 and Figure 17. Second, we capture a subset of tokens with the largest KL divergence and display the alternative tokens selected by the efficient model at the same positions, as shown in Figure 16 and Figure 18.

From the KL visualizations, we observe that tokens such as "Wait," "Alternatively," and "Hmm," which often signal hesitation or shifts in reasoning, consistently exhibit larger divergence. From the alternative-token analysis, we further find that the efficient counterpart frequently replaces tokens such as "." with ".\n\n", suggesting a tendency to terminate the current line of analysis and transition more decisively to the next reasoning step. Together, these case studies demonstrate that our proposed diagnostic provides a reliable mechanism for localizing inefficiency-inducing token positions, supporting more interpretable efficiency analyses and future research on efficient reasoning models.

M Additional Evaluation Beyond Mathematical Reasoning

Although our main experiments focus on mathematical reasoning, the proposed method is not tied to a specific math benchmark. To further examine whether on-policy SFT preserves broader reasoning and execution capabilities, we evaluate the original and fine-tuned models on additional tool-use and agentic-memory tasks from BFCL (see Table 2) (Patil et al., 2025). These tasks provide a complementary testbed: success requires structured function calls or information retrieval across multi-turn interactions, rather than just a final numerical answer.

As shown in Tables 3 and 4, on-policy SFT generally preserves or improves the tool-use and agentic-memory performance of the original models. The absolute scores remain limited, especially for the smaller model, indicating that strong tool-use ability is not already present in these reasoning-distilled models. Nevertheless, these relative gains suggest that shortening reasoning trajectories via correctness-preserving on-policy updates does not degrade structured tool-use behavior.

Table 4: BFCL agentic-memory results on DeepSeek-R1-Distill-Qwen-1.5B and 7B. Overall is averaged over the displayed agentic-memory subsets.

Model	Variant	Agentic Overall	Memory KV	Memory Vector
R1-Distill-Qwen-1.5B	Original model	1.94%	2.58%	1.29%
	After on-policy SFT	2.58%	2.58%	2.58%
R1-Distill-Qwen-7B	Original model	2.26%	3.87%	0.65%
	After on-policy SFT	3.66%	4.73%	2.58%

Table 5: OOD generalization to the harder math competition benchmarks AIME24 and AIME25, measured by accuracy (%) and average response length in tokens.

Method	AIME24		AIME25	
	Acc ↑	Tok ↓	Acc ↑	Tok ↓
Original model	25.3	17,418	26.0	16,348
RL-based method (trained with MATH-500)	25.3	5,389	18.0	4,851
On-policy SFT (trained with MATH-500)	30.7	4,523	23.3	3,838
On-policy SFT (trained with DeepScaler; main paper setting)	30.7	3,808	23.3	3,096

N Generalization and Capability Preservation

We further evaluate whether on-policy SFT overfits to the training distribution or degrades capabilities outside the fine-tuning domain. To isolate this effect, we train the 1.5B model on MATH-500 and evaluate both out-of-distribution mathematical generalization and non-math capability preservation.

N.1 Out-of-Distribution Generalization

Table 5 shows that on-policy SFT trained only on MATH-500 transfers well to harder competition-style benchmarks. Compared with the RL-based method trained on the same data source, on-policy SFT achieves higher accuracy on both AIME24 and AIME25 while also maintaining substantially shorter responses than the original model. This suggests that the concise reasoning behavior learned by on-policy SFT is not merely a dataset-specific shortcut.

N.2 Cross-Domain Capability Preservation

Table 6 evaluates whether math-only fine-tuning harms broader model capabilities. On-policy SFT remains close to the original model on both MMLU and IFEval and retains more non-math performance than the RL-based method trained under the same data setting. These results indicate that the proposed update does not simply trade general capability for shorter math reasoning.

O Additional Comparisons with RL-Based Objectives

This section complements the analysis in the main text by comparing on-policy SFT with additional RL-style objectives under matched truncation-based length control. These experiments are designed to isolate whether the gains come from the simplified optimization formulation rather than from a particular implementation of a single RL baseline. Table 7 shows that when the same truncation-based length criterion is applied, on-policy SFT achieves the best compression across all three benchmarks and the highest accuracy on GSM8K and AIME24. This supports the view that additional GRPO-family machinery is not necessary for obtaining a strong accuracy–efficiency trade-off in this verifiable efficient-reasoning setting.

P Component-Wise Ablation

Table 8 provides an empirical decomposition of the path from a full RL-based formulation to on-policy SFT. Removing the KL term substantially reduces response length, while removing group-

Table 6: Cross-domain capability preservation on the non-math benchmarks MMLU and IFEval. We report accuracy (%) on MMLU and prompt-level strict accuracy (%) on IFEval.

Method	MMLU	IFEval
Original model	24.00	25.69
RL-based method (trained with MATH-500)	22.60	23.70
On-policy SFT (trained with MATH-500)	23.56	25.14

Table 7: Comparison of GRPO, GRPO variants, and on-policy SFT on GSM8K, MATH-500, and AIME24 using DeepSeek-R1-Distill-Qwen-1.5B as the base model.

Method	GSM8K			MATH-500			AIME24		
	Acc ↑	Tok ↓	CR ↓	Acc ↑	Tok ↓	CR ↓	Acc ↑	Tok ↓	CR ↓
DeepSeek-R1-Distill-Qwen-1.5B CoT	85.0	2,533	100.0%	86.0	5,420	100.0%	25.3	17,418	100.0%
truncation-based GRPO	84.7	710	28.0%	84.2	1,546	28.5%	25.3	4,888	28.1%
truncation-based DAPO	84.5	840	33.2%	84.3	1,824	33.7%	25.3	5,888	33.8%
truncation-based Dr.GRPO	85.3	816	32.2%	85.2	1,668	30.8%	25.3	5,321	30.5%
on-policy SFT	86.1	696	27.5%	85.7	1,354	25.0%	30.7	3,808	21.8%

Table 8: Component-wise ablation study from a standard RL baseline to on-policy SFT.

Method	GSM8K			MATH			AIME		
	Acc ↑	Tok ↓	CR ↓	Acc ↑	Tok ↓	CR ↓	Acc ↑	Tok ↓	CR ↓
Base CoT Model	85.0	2,533	100.0%	86.0	5,420	100.0%	26.0	16,348	100.0%
L1 (Full RL Baseline)	86.0	1,607	63.4%	85.3	1,922	35.4%	20.0	2,762	16.9%
L1 (w/o KL divergence)	86.0	986	38.9%	85.3	1,524	28.1%	22.6	3,254	19.9%
L1 (w/o KL & Group Norm)	86.0	957	37.8%	85.3	1,498	27.6%	22.6	3,469	21.2%
On-policy SFT (Ours)	86.1	696	27.5%	85.7	1,354	25.0%	23.3	3,096	18.9%

wise normalization yields a smaller additional change. The final on-policy SFT objective achieves the best overall trade-off among these variants, suggesting that the simplified objective and threshold-based selection work together to produce stable compression.

Q Limitations and Broader Impact

Limitations. Our study focuses on efficient reasoning for mathematical tasks, where correctness and response length are directly verifiable. This setting enables a simple truncation-based filtering criterion, but the same assumption may not hold for open-ended domains such as writing, dialogue, coding, or long-form factual generation, where correctness, usefulness, and conciseness are harder to define automatically. In addition, our experiments are conducted on distilled DeepSeek-R1 models at 1.5B and 7B scales, so further validation is needed to assess whether the same conclusions hold for larger models, different base model families, and non-mathematical reasoning tasks.

Broader Impact. This work aims to make reasoning models more efficient by reducing unnecessary generation length and simplifying the training pipeline. By lowering inference and training costs, on-policy SFT may help improve the accessibility and sustainability of reasoning models, especially in resource-constrained research and deployment settings. At the same time, more efficient reasoning systems could also be used in applications where automated problem solving has significant consequences, such as education, software development, or scientific assistance. Therefore, practical deployments should carefully monitor accuracy, robustness, and failure modes, rather than relying solely on shorter outputs or improved efficiency metrics. Our findings encourage prioritizing simple, principled training objectives over increasingly complex optimization recipes.

Question

Natalia sold clips to 48 of her friends in April, and then she sold half as many clips in May. How many clips did Natalia sell altogether in April and May?

CoT

Okay , so I need to figure out how many clips Natal ia sold altogether in April and May . Let me read the problem again to make sure I understand it correctly .\n\n N atal ia sold clips to 4 8 of her friends in April . Then , in May , she sold half as many clips as she did in April . So , I need to find the total number of clips she sold in both months combined .\n\n Alright , let 's break this down step by step . First , in April , she sold clips to 4 8 friends . So , that means she sold 4 8 clips in April . Got that part .\n\n Now , for May , she sold half as many clips as she did in April . Hmm , so if she sold 4 8 in April , then in May , she sold half of 4 8 . Let me write that down to visualize it better .\n\n April sales : 4 8 clips .\n\n May sales : Half of April 's sales . So , that would be 4 8 divided by 2 . Let me calculate that . 4 8 divided by 2 is 2 4 . So , she sold 2 4 clips in May .\n\n Now , to find the total number of clips sold in both months , I need to add the April sales and May sales together . So , that would be 4 8 (April) plus 2 4 (May) . Let me add those numbers .\n\n 4 8 plus 2 4 . Hmm , 4 0 plus 2 0 is 6 0 , and 8 plus 4 is 1 2 . So , 6 0 plus 1 2 is 7 2 . Therefore , the total number of clips sold in April and May is 7 2 .\n\n Wait , let me double -check my calculations to make sure I didn 't make a mistake . So , 4 8 divided by 2 is definitely 2 4 , and 4 8 plus 2 4 is 7 2 . Yeah , that seems right .\n\n Alternatively , I can think of it as 4 8 multiplied by 1 . 5 , since she sold 1 . 5 times as much in May as she did in April . Let me verify that . 4 8 times 1 . 5 is 7 2 . Yep , that matches my previous result .\n\n So , whether I add them directly or multiply by 1 . 5 , I get the same answer . That gives me more confidence that 7 2 is the correct total number of clips sold .\n\n Just to recap , the steps were :\n\n 1 . Determine April sales : 4 8 clips .\n 2 . Calculate May sales : 4 8 / 2 = 2 4 clips .\n 3 . Add both months together : 4 8 + 2 4 = 7 2 .\n\n I think that 's solid . I don 't see any errors in my reasoning . So , I 'm pretty sure the total number of clips sold in April and May is 7 2 .\n\n ** Final Answer **\n N atal ia sold a total of \ boxed { 7 2 } clips in April and May .\n </think>

Figure 15: Token-level KL divergence between the original and on-policy SFT–trained models along a generated GSM8K CoT (Example 1). Darker colors indicate larger divergence.

Question

Natalia sold clips to 48 of her friends in April, and then she sold half as many clips in May. How many clips did Natalia sell altogether in April and May?

CoT

Okay , so → let I need to figure out how many clips Natal ia sold altogether in April and May . Let → Hmm me → 's read → break the problem again to → .\n\n make sure I understand it correctly → .\n\n .\n\n N atal ia sold clips to 4 8 of her friends in April . Then , in → she May , she sold half as many clips as she did in April . So , I need to find the total number of clips she → sold sold in both months combined → .\n\n .\n\n Alright → Hmm , let 's break this → it down step by step . → .\n\n First , in April , she sold clips to 4 8 friends . So → Hmm , that → does means → 's she sold 4 8 clips in April . → .\n\n Got that part → .\n\n .\n\n Now → Then , for → in May , she sold half as many clips as she did in April . Hmm , → . so if she sold 4 8 in April , then in May , → she she sold half of 4 8 . Let me write that down to → .\n\n visualize → keep it better .\n\n April sales : 4 8 clips .\n\n May sales : Half of April 's sales . So , that would → 's be 4 8 divided by 2 . → .\n\n Let me calculate that . → .\n\n 4 8 divided by 2 is 2 4 . So , she sold 2 4 clips in May .\n\n Now , to find the total number of clips sold in both months , I need to add the → April April sales and May sales together . → .\n\n So , that → would → 's be 4 8 (→ plus April) plus 2 4 (May) . →) .\n\n Let me → 's add those numbers .\n\n 4 8 plus 2 4 . Hmm , 4 0 → 8 plus 2 0 is 6 0 , and 8 plus 4 is 1 2 . → , So , 6 0 plus 1 2 is 7 2 . Therefore → So , the total number of clips sold in → is April and May is 7 2 .\n\n Wait , let me double -check my calculations to make sure I didn 't make a mistake . → .\n\n So , → April 4 8 divided → plus by 2 is definitely → indeed 2 4 , → . and 4 8 plus 2 4 is → indeed 7 2 . Yeah → Yep , that seems right .\n\n Alternatively → So , I → another can think of it as → multiplying 4 8 multiplied by 1 . 5 , → because since she sold → half 1 . 5 times as much in May as she → in did in April . Let → But me → 's verify → see that . → .\n\n 4 8 times 1 . 5 is → indeed 7 2 . Yep , that matches → works my → .\n\n previous result .\n\n So , whether → both I add them directly or multiply by 1 . 5 , I get the same answer . That gives me more confidence that 7 2 is the correct total number → .\n\n of clips sold .\n\n Just → Therefore to recap , → :\n\n the → step steps were :\n\n 1 . Determine → April April sales : 4 8 clips .\n 2 . Calculate May sales : 4 8 / 2 = 2 4 clips .\n 3 . Add both months → amounts together : 4 8 + 2 4 = 7 2 .\n\n I → Yep think that 's solid . I → So don 't see any errors → mistakes in my → the reasoning . → .\n\n So , I 'm pretty → confident sure → confident the total → answer number of clips sold in → is April and May is 7 2 .\n\n ** Final Answer **\n N → \atal ia sold a total of \ boxed { 7 2 } clips in April and May .\n </think>

Figure 16: Tokens with the largest KL divergence and the corresponding alternative tokens selected by the efficient model for a GSM8K example (Example 1).

Question

Janet’s ducks lay 16 eggs per day. She eats three for breakfast every morning and bakes muffins for her friends every day with four. She sells the remainder at the farmers' market daily for $2 per fresh duck egg. How much in dollars does she make every day at the farmers' market?

CoT

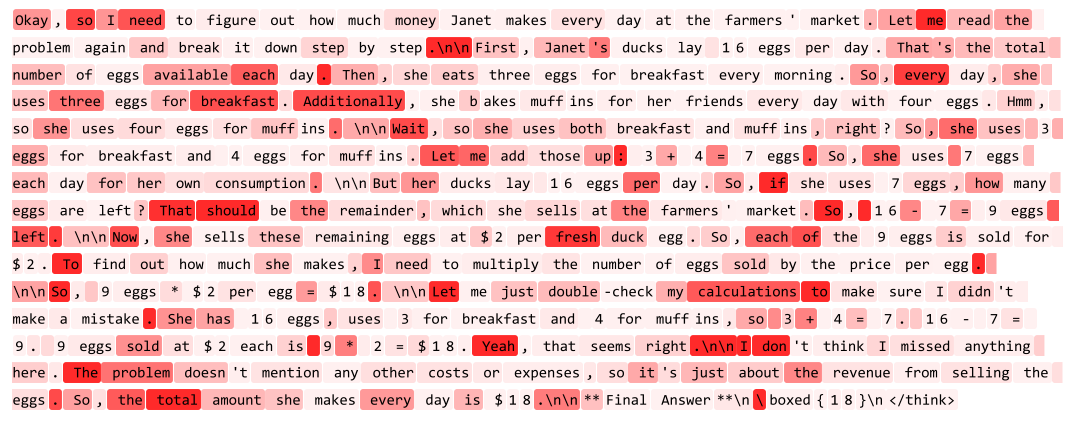

Figure 17: Token-level KL divergence between the original and on-policy SFT–trained models along a generated GSM8K CoT (Example 2). Darker colors indicate larger divergence.

Question

Janet’s ducks lay 16 eggs per day. She eats three for breakfast every morning and bakes muffins for her friends every day with four. She sells the remainder at the farmers' market daily for $2 per fresh duck egg. How much in dollars does she make every day at the farmers' market?

CoT

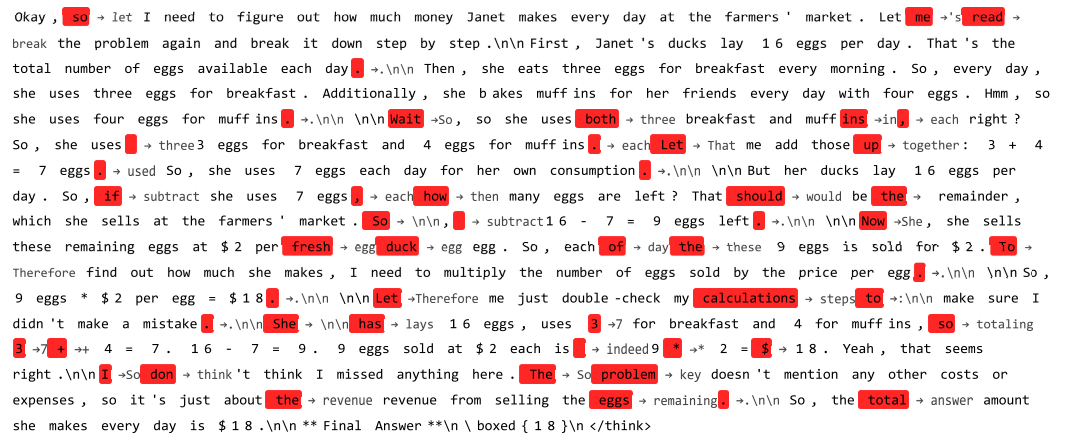

Figure 18: Tokens with the largest KL divergence and the corresponding alternative tokens selected by the efficient model for a GSM8K example (Example 2).