

AdapThink: Adaptive Thinking Preferences for Reasoning Language Models

Wenyue Xu¹, Xu Wan^{2,3}, Wei Wang⁴, Wenqi Huang⁵,
Wotao Yin⁴, Shengjie Zhao¹, Mingyang Sun²

¹Tongji University, ²Peking University, ³Zhejiang University,
⁴Alibaba DAMO Academy, ⁵China Southern Power Grid

Correspondence: [Mingyang Sun](#), [Shengjie Zhao](#)

Abstract

The “slow thinking” paradigm has been widely validated to enhance the reasoning capabilities of Large Language Models (LLMs), but it introduces notable reasoning inefficiencies: models often overthink simple tasks while prematurely shifting their reasoning paths when addressing complex problems. To address this, we propose **AdapThink**, a simple yet efficient framework for adaptive reasoning preference control. Unlike methods imposing uniform length constraints, AdapThink dynamically adjusts reflection preferences based on group-level distributional statistics of reasoning length and reflection intensity. We further introduce a dispersion-based diversity sampling mechanism that maximizes the geometric spread of reasoning patterns, accelerating learning through exposure to diverse problem-solving strategies. Across mathematical reasoning and code generation benchmarks, AdapThink reduces average response length by 17.1%-21.4% while improving performance by 6.12-6.59 points under 32K token budgets, demonstrating superior efficiency and robustness against reward hacking compared to strong baselines.

1 Introduction

The breakthroughs in large language models (LLMs) such as OpenAI’s o1 (OpenAI, 2024) and DeepSeek R1 (Guo et al., 2025) have demonstrated that reinforcement learning (RL)-based post-training methods can substantially improve their reasoning capabilities. This enhancement is primarily attributed to the emergence of sophisticated self-reflection behaviors in models (Kumar et al., 2025; Kazemnejad et al., 2024). However, recent research has highlighted a significant inefficiency associated with this “*slow thinking*” pattern (Muennighoff et al., 2025; Han et al., 2024; Chen et al., 2024; Wang et al., 2025b; Aggarwal and Welleck, 2025; Shen et al., 2025b). Reasoning models frequently *overthink* simple problems, spending unnecessary

computational resources, while conversely *underthinking* complex challenges, leading to incomplete reasoning and incorrect answers. When presented with a simple problem (see Appendix A), a model can reach a correct answer with merely 479 tokens. Yet, its self-validation mechanisms—marked by phrases such as “Verify” and “Wait”—triggered unnecessary reflections, resulting in more than a quadrupled token consumption. In contrast, when attempting a complex problem, the same model exhibited frequent and unproductive shifts, marked by phrases “Alternatively” and “Another.” Upon reaching its token limit, the model stopped and arrived at an incorrect answer.

Therefore, an ideal chain of thought (CoT) would be capable of adjusting its self-reflection frequency and depth, adaptive to the problem difficulty and model’s capability. Empirically, (Ma et al., 2025; Xie et al., 2025) have demonstrated that employing reflection vocabularies does *not always* guarantee mathematical reasoning improvement. Drawing upon this insight, recent works explored direct control via modifying the input prompt (Jin et al., 2024; Muennighoff et al., 2025; Liu et al., 2025; Han et al., 2024; Chen et al., 2024) or providing indirect reasoning-length rewards (Wang et al., 2025b; Aggarwal and Welleck, 2025; Shen et al., 2025b). A primary limitation of these approaches lies in their reliance on rule-governed token budgets or rewards for adhering to a *static length budget*, thereby overlooking the dynamic association between a model’s inherent capabilities and its length preferences during post-training. On the one hand, models with limited reasoning abilities often benefit from extended CoT patterns, as redundant self-reflection may serendipitously lead to correct solutions. On the other hand, the reasoning process for exceptionally challenging problems should not be stifled by rigid length constraints, as the value of enhancing accuracy fundamentally outweighs the objective of minimizing

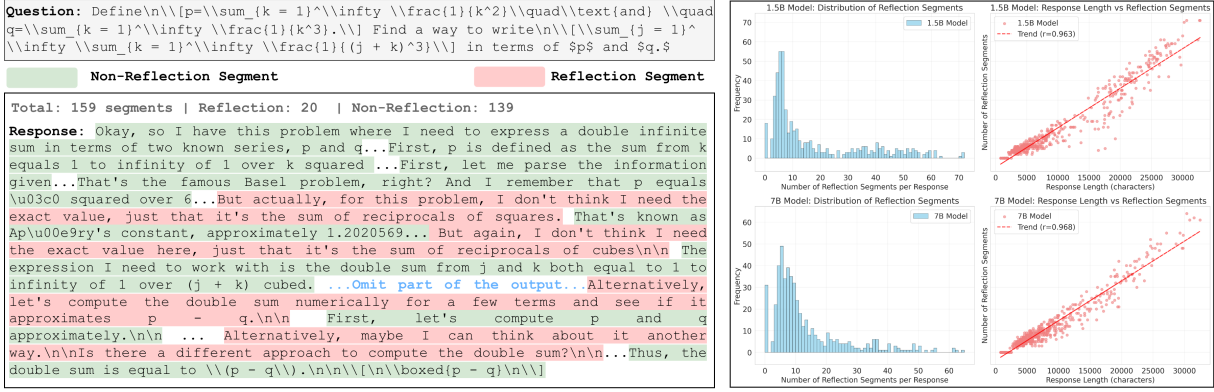


Figure 1: **Key Observations** from DeepSeek-Distill-Qwen models (1.5B and 7B) on MATH-500. *Left:* A representative example demonstrating reflection segment annotation. *Right:* Distribution of reflection segment counts and their correlation with response length across the dataset.

reasoning length.

Motivated by these insights and our observations in Section 2, we propose AdapThink, an adaptive reasoning pattern control framework for LLM post-training. It allows models to tailor their reasoning pattern preference to align with their current reasoning capabilities. Instead of directly limiting the budget of reasoning length, AdapThink adjusts the length and reflection preference through analyzing the distribution of diverse reasoning patterns observed in groups of generated samples. Overall, our main contributions are:

(1) First, we introduce a group-level reasoning process reward. Instead of setting a fixed token budget or restricting the number of reflection words, we quantify reflection-related preferences by analyzing statistical differences in response length and reflection segments within groups.

(2) Furthermore, we propose a dispersion-based diversity sampling mechanism that explicitly optimizes the distributional spread of reasoning patterns. Unlike existing oversampling strategies that optimize for correctness, AdapThink maximizes the covariance determinant of process-based reasoning features, systematically promoting exposure to diverse problem-solving strategies.

(3) By post-training models at 1.5B, 7B, and 32B scales on mathematical reasoning and code generation tasks, AdapThink outperforms multiple length-control baselines and mitigates overthinking without performance degradation.

2 Observations

To investigate potential overthinking and underthinking issues in current reasoning language models, we first conducted a comprehensive analysis of

the generation patterns of the DeepSeek-R1-Distill-Qwen-1.5B and DeepSeek-R1-Distill-Qwen-7B models (Guo et al., 2025) on the MATH-500 mathematical dataset. We measure the efficiency of CoT from two perspectives:

Response Length Distribution. Following the settings in (Aggarwal and Welleck, 2025), we set the maximum token limit to **32,768** during inference for each prompt. The distribution is shown in the right panel of Figure 1.

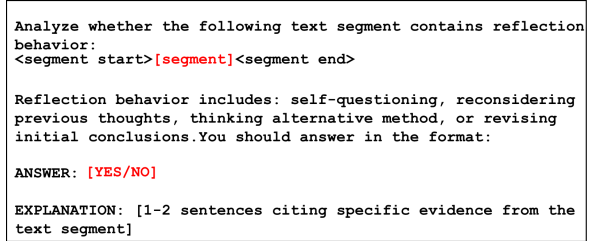


Figure 2: **Prompt Structure** for the DeepSeek-R1-Distill-Qwen-32B model. The generated responses are parsed to extract **YES/NO** labels, which serve as the dataset for training the segment-level reflection behavior detection model.

Reflection Segment Distribution. The occurrence count of reflection-related terms is commonly used to measure reasoning capabilities (Guo et al., 2025; Wang et al., 2025a; Ma et al., 2025; Xie et al., 2025; Aggarwal and Welleck, 2025), typically focusing on markers such as “wait” or “alternatively”. However, keyword-based detection is unreliable and susceptible to gaming. We therefore adopt a more robust approach where responses from the 1.5B model are segmented into 256-token chunks and annotated by DeepSeek-R1-

Distill-Qwen-32B for reflection presence (annotation prompt in Figure 2). This yields 11,620 segments, of which 45.3% exhibit reflective behavior. We train a lightweight BERT classifier (Devlin et al., 2019) on an 80/20 train-test split, achieving **98.86%** F1-score on the test set after 4 epochs. Using this classifier, we analyze reflection segment distributions across responses, with a representative example shown in Figure 1 (left panel).

As shown in Figure 1, **68.2%** of the 1.5B model outputs and **70.6%** of the 7B model outputs contain more than 6 reflection segments, with a clear positive correlation observed between response length and the number of reflection segments. More importantly, as illustrated in Appendix Figure 10, **95.3%** of incorrect outputs from the 1.5B model and **97.5%** of incorrect outputs from the 7B model exhibit high reflection behavior, with average reflection segments of **17.06** and **12.78**, respectively.

These observations motivate us to explore the value of the emergence of reflection-related segments for reasoning capabilities and reasoning efficiency. Instead of focusing on the token level, we conduct our research at the sequence level, aiming to adjust the preference for reflection based on the overall statistical information of reflection-related segments and length.

3 Method

Building on the observations and analyses presented in Section 2, this section elaborates on our proposed framework, **AdapThink**, by addressing two critical questions:

Q1. How can we construct *tailored budgets* for individual samples and establish dynamic *reflection preferences* corresponding to problems of varying difficulty?

Q2. How can we enhance the *latent diversity* of reasoning patterns for each question and *explore more efficient* problem-solving strategies?

We address Q1 and Q2 in Sections 3.1 and 3.2, respectively, with the overall framework of AdapThink illustrated in Figure 3 and the pseudo-code provided in Appendix C.

3.1 Group-relative Reasoning Process Reward

We begin with a pre-trained reasoning language model π_θ and a dataset $\mathcal{D} = \{(x^k, y^{*k})\}_{k=1}^N$, where each instance contains the input prompt x and the correct answer y^* . For each input $x \in \mathcal{D}$, π_θ performs reasoning to generate $|\mathcal{G}|$ samples

$$\mathcal{G} := \{\hat{y}_i\}_{i=1}^{|\mathcal{G}|}.$$

For each generated sample $\hat{y}_i \in \mathcal{G}$, we define two key metrics to characterize its reasoning pattern:

- $l(\hat{y}_i)$: the number of tokens in response \hat{y}_i .
- $r(\hat{y}_i)$: the count of reflection segments in \hat{y}_i .

Next, we analyze the output correctness and CoT characteristics of each response group.

Outcome Correctness. We represent the difficulty of problem x by the average correctness among $|\mathcal{G}|$ responses, denoted as φ :

$$\varphi(x, \hat{y}_{1, \dots, |\mathcal{G}|}) = \frac{1}{|\mathcal{G}|} \sum_{i=1}^{|\mathcal{G}|} \mathbb{I}(\hat{y}_i = y^*), \quad (1)$$

where $\mathbb{I}(\cdot)$ is the indicator function and y^* is the ground truth answer.

Process Characteristics. As analyzed in Section 2, incorrect reasoning exhibits significantly longer length and a higher number of reflection segments compared to correct reasoning. Therefore, to isolate the analysis of these two categories, we partition the response set \mathcal{G} into correct $\mathcal{G}_\mathcal{T}$ and incorrect $\mathcal{G}_\mathcal{F}$ sub-groups:

$$\mathcal{G}_\mathcal{T} = \{\hat{y} \in \mathcal{G} : \hat{y} = y^*\}, \quad \mathcal{G}_\mathcal{F} = \mathcal{G} \setminus \mathcal{G}_\mathcal{T} \quad (2)$$

For each sub-group, we compute statistical measures of key reasoning process metrics:

$$\mu_l(\mathcal{G}) = \frac{1}{|\mathcal{G}|} \sum_{\hat{y}_i \in \mathcal{G}} l(\hat{y}_i), \quad \mu_r(\mathcal{G}) = \frac{1}{|\mathcal{G}|} \sum_{\hat{y}_i \in \mathcal{G}} r(\hat{y}_i) \quad (3)$$

where $\mathcal{G} \in \{\mathcal{G}_\mathcal{T}, \mathcal{G}_\mathcal{F}\}$.

Adaptive Thinking Preference Based on the outcome correctness and process characteristics within each group, we introduce a group-level reasoning process reward $r_{\text{ref}}(\hat{y}_i)$ that adaptively regulates reflection preference using Equation 4.

$$r_{\text{ref}}(\hat{y}_i) = \text{clip} \left(\omega(\varphi) \cdot \left[\left(\frac{\mu_l(\mathcal{G}(\hat{y}_i)) - l(\hat{y}_i)}{\mu_l(\mathcal{G}(\hat{y}_i))} \right) + \left(\frac{\mu_r(\mathcal{G}(\hat{y}_i)) - r(\hat{y}_i)}{\mu_r(\mathcal{G}(\hat{y}_i))} \right) \right], -1, 1 \right) \quad (4)$$

where $\mathcal{G}(\hat{y}_i) \in \{\mathcal{G}_\mathcal{T}, \mathcal{G}_\mathcal{F}\}$ denotes the correctness group that sample \hat{y}_i belongs to. Each fraction $\frac{\mu - \text{individual}}{\mu}$ measures the relative difference between the group mean and the individual sample’s reasoning characteristics.

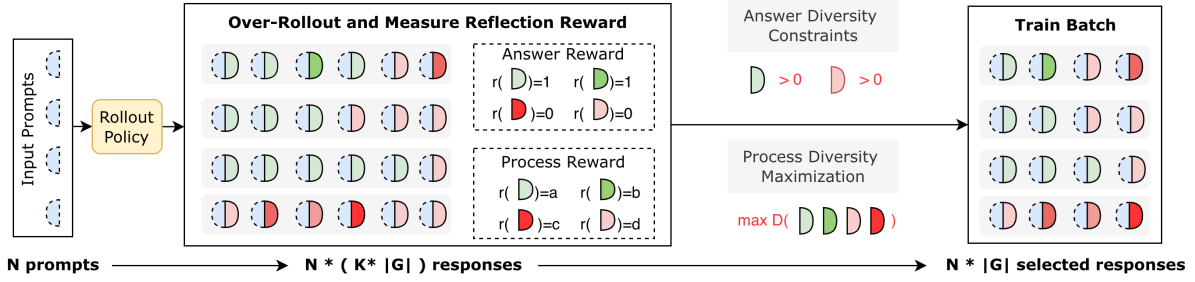


Figure 3: **The Framework Architecture of AdapThink.** The left semicircular shapes ◐ represent “prompts”, and the right semicircular shapes ◑ represent “responses”. Green ◑ indicates correct answers, while red ◑ indicates incorrect answers. Different shades correspond to varying reasoning process rewards based on Equation 4.

The dynamic nature of r_{ref} lies in the weight function $\omega(\varphi)$, which employ a cosine function to characterize the dynamic transition of thinking preferences.

$$\omega(\varphi) = \begin{cases} 0 & \text{if } \varphi < \varphi_{\text{thr}} \\ \cos\left(\frac{\pi}{2} \cdot \frac{1-\varphi}{1-\varphi_{\text{thr}}}\right) & \text{if } \varphi_{\text{thr}} \leq \varphi \leq 1 \end{cases} \quad (5)$$

In Equation 5, if the current group’s correctness φ falls below the threshold φ_{thr} , we do not impose any reflection preference on the reasoning process. This design ensures that when dealing with models lacking reasoning capabilities or tackling difficult problems, we still train them in **an performance-pursuing mode**. Conversely, as φ increases from φ_{thr} toward 1, the framework progressively transitions into a **length-aversion mode**. In this regime, the model is incentivized to produce more concise responses with fewer reflective tokens through a smooth, cosine-based transition.

3.2 Dispersion-based Diversity Sampling

As shown in Equation 4, the reasoning process reward r_{ref} incorporates group-relative distributional statistics from $\mathcal{G}_{\mathcal{T}}$ and $\mathcal{G}_{\mathcal{F}}$. Consequently, the distributional diversity of reasoning metrics $l(\hat{y})$ and $r(\hat{y})$ within each group critically determines the informativeness of r_{ref} signals. While prior work has addressed the zero-advantage problem through dynamic oversampling (Yu et al., 2025), such approaches do not explicitly account for diversity in reasoning processes. We argue that *homogeneous reasoning patterns fundamentally limit effective learning by constraining the model’s exposure to diverse problem-solving strategies and reducing the discriminative power of pattern-based rewards*.

To address this limitation, we propose a two-stage dispersion-based sampling strategy that explicitly maximizes reasoning diversity while main-

taining balanced group correctness. The complete algorithm is presented in Algorithm 2.

Stage 1: Oversampling and Diversity Quantification.

Given a question x , we first oversample by a factor of $K \in [1, 2]$ to construct an expanded candidate pool $\mathcal{G}' = \{\hat{y}_1, \dots, \hat{y}_{K|\mathcal{G}|}\}$. To quantify the geometric spread of reasoning patterns in this pool, we compute a dispersion-based diversity measure:

$$\mathcal{D}(\mathcal{G}') = \log \det(\Sigma_{\mathcal{G}'}) \quad (6)$$

where $\Sigma_{\mathcal{G}'} = \text{Cov}(\mathbf{F})$ is the 2×2 covariance matrix of the feature matrix $\mathbf{F} \in \mathbb{R}^{|\mathcal{G}'| \times 2}$:

$$\mathbf{F} = \begin{bmatrix} l(\hat{y}_1) & r(\hat{y}_1) \\ \vdots & \vdots \\ l(\hat{y}_{|\mathcal{G}'|}) & r(\hat{y}_{|\mathcal{G}'|}) \end{bmatrix}$$

Geometrically, $\det(\Sigma_{\mathcal{G}'})$ measures the squared hypervolume of the confidence ellipsoid spanned by the reasoning feature distribution. Larger values indicate greater dispersion in the (l, r) space.

Stage 2: Diversity-Maximizing Downsampling.

From the oversampled pool \mathcal{G}' , we construct the final sample set $\mathcal{G} \subset \mathcal{G}'$ with $|\mathcal{G}|$ samples by solving:

$$\begin{aligned} \mathcal{G}^* &= \arg \max_{\substack{\mathcal{G} \subset \mathcal{G}' \\ |\mathcal{G}|=n}} \mathcal{D}(\mathcal{G}) \\ \text{s.t. } & \varphi(\mathcal{G}) \in (0, 1) \text{ if } \varphi(\mathcal{G}') \in (0, 1) \end{aligned} \quad (7)$$

This formulation achieves two objectives: (1) **Diversity maximization**: the selected samples exhibit maximal dispersion in reasoning features (l, r) , providing rich gradient signals for r_{ref} learning; (2) **Zero-advantage avoidance**: the constraint $\varphi(\mathcal{G}) \in (0, 1)$ prevents degenerate cases where all responses are uniformly correct or incorrect, ensuring non-zero gradient.

4 Experimental Setup

Datasets and Base Models. The main experiments are based on a curated lightweight mathematics dataset that spans various difficulty levels. This dataset combines questions from *DeepScaleR-Preview-Dataset* (Luo et al., 2025b), including about 5K question-answer pairs sampled from AIME (1984-2023), AMC (prior to 2023), and MATH training sets. For mathematical tasks, we employ *DeepSeek-R1-Distill-Qwen-1.5B* and *DeepSeek-R1-Distill-Qwen-7B* (Guo et al., 2025), which are obtained through supervised fine-tuning on reasoning data generated by the DeepSeek-R1 model. For the competitive code generation task, we start from the DRIVE-SFT model (Zhu et al., 2025), which is a fine-tuned *Qwen2.5-32B* model using 1.27M open-source prompts and generated corresponding responses through distillation from the open-source model DeepSeekR1-0528.

Comparing Methods. We first conduct post-training on *DeepSeek-R1-Distill-Qwen-1.5B* and *DeepSeek-R1-Distill-Qwen-7B* using Group Relative Policy Optimization (GRPO) (Guo et al., 2025) as the base algorithm. The detailed algorithm is shown in Appendix C. To evaluate different length-control methods for enhancing reasoning efficiency, we implement the following methods: (1) Methods that **directly reduce reasoning length**, including O1-Pruner (Luo et al., 2025a) and No-Wait (Wang et al., 2025a); (2) Methods that **rely on preset computational budgets** before inference, including DAST (Shen et al., 2025b), FCS+Ref (Chen et al., 2024), and LCPO (Aggarwal and Welleck, 2025); (3) Methods that **dynamically determine termination**, including DEER (Yang et al., 2025) and MERA (Ha et al., 2025).

5 Main Results and Ablations

In this section, we evaluate and analyze the performance of AdapThink across various settings through three key aspects: answer accuracy, reasoning efficiency, and group diversity.

Overall Comparison on Math Tasks. As shown in Figure 5 and Table 1, AdapThink demonstrates superior performance across models of different scales. Compared to the original baseline, AdapThink achieves a **+6.59** overall improvement on the 1.5B model and a **+6.12** improvement on the 7B model, while reducing length by **33.15%** and **29.10%**, respectively.

Notably, although trained under a strict *2K token limit* for mathematical tasks, AdapThink develops a dynamic reasoning preference that allows it to adaptively adjust CoT preference to problem complexity when evaluated in a *32K limit context*. As shown in Table 1, on challenging benchmarks such as AIME, it produces longer, more reasoned responses compared to other length-control methods, leading to stronger performance on difficult tasks. In contrast, while MERA achieves shorter outputs, it incurs performance drops of **2.22** and **2.3** points relative to AdapThink, with a particularly pronounced gap on challenging benchmarks like AIME 2024 and AIME 2025.

Furthermore, to verify the effectiveness of each component in the AdapThink framework, we conducted comprehensive ablation experiments, and the evaluation results are shown in Figure 4 and Table 2.

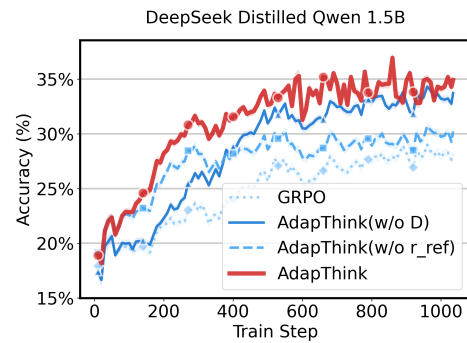


Figure 4: Ablations on \mathcal{D} and r_{ref} for DeepSeek-distilled Qwen 1.5B models.

Ablation of Dispersion-based Sampling \mathcal{D} . As shown in Figure 4, removing diversity sampling (AdapThink w/o \mathcal{D}) results in slower convergence during training. However, this has limited impact on final test performance: Table 2 (Row 2) shows that AdapThink w/o \mathcal{D} still achieves a **+7.1** point improvement over the baseline. While oversampling with $K = 2$ increases per-step sampling cost, the accelerated convergence compensates for this overhead: models with $K = 2$ reach the same performance at 600 steps as $K = 1$ models at 1000 steps, resulting in comparable overall training time.

To assess generalizability, we apply diversity sampling to other length-control methods. The mechanism consistently **accelerates convergence**: LCPO+ \mathcal{D} exhibits 36.4% faster convergence, while DAST+ \mathcal{D} achieves 31.6% improvement. These results further demonstrate that dispersion-based diversity sampling is a broadly applicable technique

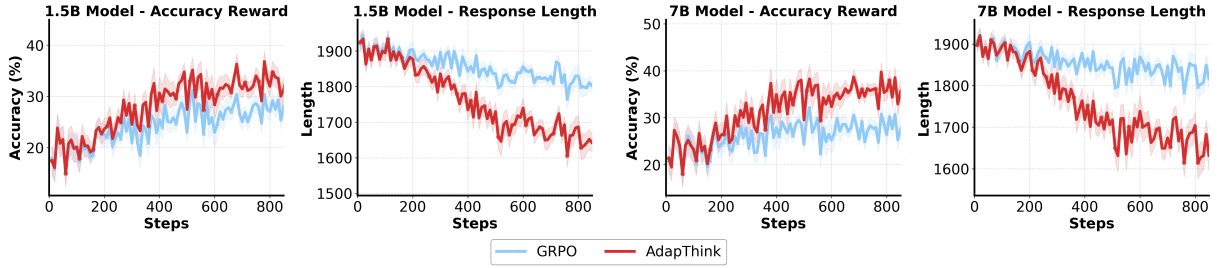


Figure 5: **Training Comparison between GRPO and AdapThink** on DeepSeek-Distill-Qwen-1.5B and DeepSeek-Distill-Qwen-7B with a constraint 2K token budget.

Table 1: **Performance Comparison on Mathematical Reasoning Benchmarks.** The Acc metric is determined by averaging pass@1 scores over 16 independent generation runs, and the Token metric measures the corresponding average response length for each benchmark. Overall results are the simple arithmetic mean across all benchmarks.

Method	MATH-500		AMC 2023		AIME 2024		AIME 2025		Overall	
	Acc↑	Tokens↓	Acc↑	Tokens↓	Acc↑	Tokens↓	Acc↑	Tokens↓	Acc↑	Tokens↓
<i>DeepSeek-R1-Distill-Qwen-1.5B</i>										
Original	83.0	3,978	67.7	7,160	29.3	13,832	26.9	14,680	51.73	5,348
O1-Pruner (Luo et al., 2025a)	82.3	2,446	69.5	5,622	27.5	12,155	24.1	12,701	50.85	3,787
No Wait (Wang et al., 2025a)	81.6	2,894	68.5	6,422	26.1	9,167	20.3	11,601	49.13	4,048
DAST (Shen et al., 2025b)	83.4	3,155	70.2	5,583	29.5	10,042	20.6	10,647	50.93	4,139
FCS+Ref. (Chen et al., 2024)	82.9	2,883	72.1	4,449	29.7	9,898	28.6	11,624	53.32	3,820
LCPO (Aggarwal and Welleck, 2025)	80.5	2,684	63.9	6,694	23.5	9,692	24.7	10,075	48.15	3,873
DEER (Yang et al., 2025)	84.1	2,398	70.2	4,179	26.6	9,732	20.2	9,890	50.27	3,320
MERA (Ha et al., 2025)	86.3	2,165	73.3	3,262	31.4	8,927	33.4	9,495	56.10	2,964
AdapThink (Ours)	86.8	2,618	77.2	4,208	33.3	9,728	36.0	9,719	58.32	3,575
<i>DeepSeek-R1-Distill-Qwen-7B</i>										
Original	89.4	3,816	77.4	6,738	53.1	12,185	48.2	13,276	67.03	4,719
O1-Pruner (Luo et al., 2025a)	89.0	2,678	82.8	7,501	52.2	9,412	49.4	10,973	68.35	4,002
No Wait (Wang et al., 2025a)	87.2	2,579	75.7	5,478	42.5	10,048	35.2	11,827	60.15	3,733
DAST (Shen et al., 2025b)	88.6	2,876	80.3	4,601	52.6	10,240	49.5	9,721	67.75	3,762
FCS+Ref. (Chen et al., 2024)	87.8	2,909	81.5	5,143	55.1	9,212	49.8	9,844	68.55	3,815
LCPO (Aggarwal and Welleck, 2025)	84.7	2,539	73.6	4,821	45.3	9,408	40.6	9,904	61.05	3,498
DEER (Yang et al., 2025)	88.9	1,908	82.2	5,194	46.4	9,932	39.6	9,302	64.28	3,052
MERA (Ha et al., 2025)	91.0	1,739	85.7	3,711	56.1	8,398	50.6	8,732	70.85	2,631
AdapThink (Ours)	92.0	2,472	90.0	4,047	60.4	9,464	50.2	9,896	73.15	3,346

Table 2: **Detailed Comparison of AdapThink Variants.** Max Token represents the maximum response token limit (in thousand) in training phase. The accuracy metric is determined by averaging PASS@1 scores over 8 independent generation runs under the 8K token limit on AIME 2025. The diversity metrics are from the training phase, while other metrics are from testing. Changes are relative to baseline (17.9% accuracy, $\mathcal{D}(\mathcal{G}_{\mathcal{T}}) = 5.0$, $\mathcal{D}(\mathcal{G}_{\mathcal{F}}) = 6.0$).

AdapThink Variants				Accuracy	Length		Reflection Segments		Dispersion	
μ_l	μ_r	K	Max Token	PASS@1	$\mu_l(\mathcal{G}_{\mathcal{T}})$	$\mu_l(\mathcal{G}_{\mathcal{F}})$	$\mu_r(\mathcal{G}_{\mathcal{T}})$	$\mu_r(\mathcal{G}_{\mathcal{F}})$	$\mathcal{D}(\mathcal{G}_{\mathcal{T}})$	$\mathcal{D}(\mathcal{G}_{\mathcal{F}})$
✓	✓	2	2	25.4 $\uparrow 7.5$	3561 $\downarrow 580$	7442 $\downarrow 478$	18.5 $\downarrow 3.0$	34.5 $\downarrow 4.1$	5.7 $\uparrow 0.7$	5.7 $\downarrow 0.3$
✓	✓	1	2	25.0 $\uparrow 7.1$	3803 $\downarrow 339$	7579 $\downarrow 340$	22.4 $\downarrow 1.4$	35.7 $\downarrow 2.9$	5.2 $\uparrow 0.2$	5.9 $\downarrow 0.1$
✗	✓	1	2	23.3 $\uparrow 5.4$	4772 $\uparrow 630$	8154 $\uparrow 235$	30.3 $\uparrow 5.8$	40.3 $\uparrow 1.7$	4.9 $\downarrow 0.1$	5.7 $\downarrow 0.3$
✓	✗	1	2	22.9 $\uparrow 5.0$	4395 $\downarrow 253$	7872 $\uparrow 48$	30.4 $\uparrow 6.5$	40.9 $\uparrow 2.3$	5.2 $\uparrow 0.2$	5.2 $\downarrow 0.8$
✓	✓	2	4	24.6 $\uparrow 6.7$	3861 $\downarrow 281$	7909 $\uparrow 10$	27.3 $\uparrow 3.4$	46.1 $\uparrow 7.5$	5.5 $\uparrow 0.5$	4.9 $\downarrow 1.1$
✓	✓	2	2→4	26.0 $\uparrow 8.1$	3339 $\downarrow 802$	6886 $\downarrow 1034$	15.5 $\downarrow 8.3$	38.6 $\downarrow 0.0$	7.0 $\uparrow 2.0$	5.4 $\downarrow 0.6$

for efficient post-training.

Ablation of Reasoning Process Reward r_{ref} . As shown in Equation 4, the r_{ref} is controlled by metrics μ_l , μ_r , and weight $w(\varphi)$. We study μ_l and μ_r through ablation, as shown in Rows 3,4 of Table 2. Using only μ_r or μ_l alone cannot fully control the reduction of reflection words, but they also bring **+5.4 and +5.0** performance gains over the original base. However, the complete AdapThink method uses both parts together, achieving a better performance gain while controlling reflection words more effectively.

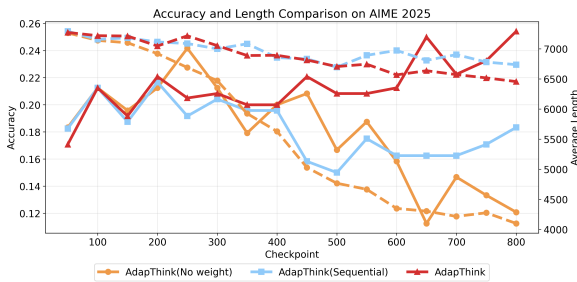


Figure 6: **Evaluation of three AdapThink variants** at 50-step training intervals. Solid lines denote accuracy dynamics, while dashed lines indicate the corresponding dynamics in response length under an 8K limit.

For $w(\varphi)$, we conduct the ablation *AdapThink(no weight)* with the training dynamic shown in Figure 7 and test dynamic in Figure 6. AdapThink(no weight) greatly reduced response lengths, which is helpful in the early stages of training due to the deeply overthinking phenomenon in the base model. However, the later training stage of AdapThink(no weight) led to significantly degraded performance on the AIME 2025 benchmark. As shown in Figure 6, we found that the shorter length in later stages of AdapThink(no weight) matched with lower PASS@1. It uncovers a notable insight for current length-control methods:

Takeaway

Indiscriminate suppression of reflection behaviors harms reasoning capabilities. By adaptively adjusting reflection preferences according to group-level correctness, AdapThink improves efficiency while maintaining or enhancing performance.

Main Result on Code Task To further validate the effectiveness of AdapThink, we apply

AdapThink to competitive code generation using a 32B model with 24K context. Table 3 demonstrates consistent improvements across two benchmarks. Critically, setting $\varphi_{\text{thr}} = 0.1$ prioritizes efficiency (**15.4%** token reduction, **+2.70** points), while $\varphi_{\text{thr}} = 0.25$ emphasizes accuracy (**+4.63** points on LiveCode-V6, **5.6%** token reduction). The detailed evaluation dynamics is shown in Appendix Figure 12.

6 Discussions

Impact of Reflection-Related Terms To isolate the effects of specific reflection behaviors, we measure $r(\hat{y})$ using distinct term categories: *AdapThink(check)* quantifies verification behaviors via terms like “wait”, “check”, and “verify”; *AdapThink(switch)* measures reasoning expansion through transitions such as “alternatively”, “however”, and “instead”; and *AdapThink(sequential)* serves as an ablation baseline using non-reflective sequential markers (“first”, “then”, “next”, “finally”).

As shown in Figure 7, AdapThink(Switch) functions similarly to AdapThink. This aligns with the motivation outlined in the study by (Wang et al., 2025b): Frequent thought switching exacerbates the model’s propensity to generate incorrect answers. AdapThink(Switch) addresses this issue by appropriately controlling its preference for such words. Additionally, AdapThink(sequential) exhibits no clear changes in either outcome rewards or process metrics, which aligns with our expectations, as reflection-unrelated terms exert little impact on reasoning patterns. Meanwhile, controlling via AdapThink(check) impairs model performance, which indicates that the model’s hesitation behavior is crucial for the reasoning patterns required to generate correct answers. Further analysis of various length-control algorithms is provided in Appendix Figure 11.

Detection of Reward Hacking. To further investigate potential reward hacking behaviors in length-control mechanisms, we quantify response quality using N -gram repetition rates ($N = 40$) following Ye et al. (2025), with results presented in Table 4. Our analysis reveals that LCPO exhibits severe repetitive patterns with an average N -gram repetition rate of 10.8%, especially 19.8% in incorrect answers. Conversely, AdapThink exhibits **uniformly low repetition rates** across correctness categories, indicating that its pattern-based reward for-

Table 3: Performance Comparison on Competitive Code Generation Benchmarks.

Method	LiveCode-V6		LiveCode-0811		Overall	
	Acc \uparrow	Tokens \downarrow	Acc \uparrow	Tokens \downarrow	Acc \uparrow	Tokens \downarrow
DRIVE-SFT (Zhu et al., 2025)	64.51	12,763	63.85	12,080	64.18	12,421
+AdapThink ($\varphi_{thr} = 0.1$)	66.29 $\uparrow 1.78$	10,677 $\downarrow 2,086$	67.47 $\uparrow 3.62$	10,346 $\downarrow 1,734$	66.88 $\uparrow 2.70$	10,511 $\downarrow 1,910$
+AdapThink ($\varphi_{thr} = 0.25$)	69.14 $\uparrow 4.63$	12,042 $\downarrow 721$	65.06 $\uparrow 1.21$	11,097 $\downarrow 983$	67.10 $\uparrow 2.92$	11,570 $\downarrow 851$

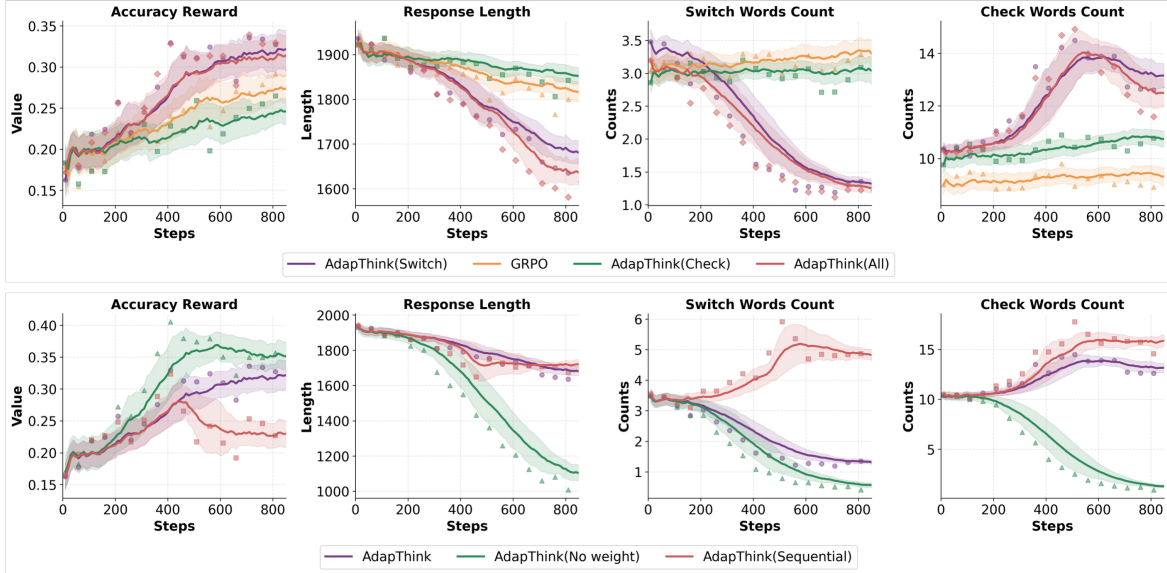


Figure 7: Training Comparison of AdapThink Variants with different calculation methods for r_{ref} (Equation 4).

Table 4: N-gram Repetition Analysis across different post-training methods.

Method	N-grams (%)		
	Total	\mathcal{G}_T	\mathcal{G}_F
GRPO	0.3	0.3	0.2
DAST	3.0	3.5	2.2
LCPO	10.8	5.9	19.8
AdapThink	0.7	0.6	0.9

mulation incentivizes reasoning exploration rather than specifically length constraint.

Out-of-Distribution Evaluation. We also validated AdapThink on non-math out-of-distribution (OOD) benchmarks in Table 5. We found that AdapThink’s improvements over the base model are not limited to in-distribution tasks. It also achieves performance gains on OOD tasks with improved **2.77**-point performance gain and reduced **26.7%** token usage.

Table 5: Performance Comparison on Out-of-Distribution (OOD) Benchmarks for AdapThink.

Model	LSAT		GPQA	
	Pass@1	Avg Token	Pass@1	Avg Token
1.5B Base	24.62	7481	29.42	6052
+AdapThink	24.78	6810	28.30	4436
7B Base	41.68	6723	45.61	5759
+AdapThink	42.83	5982	48.38	4944

7 Conclusion

In this paper, we present AdapThink, an adaptive reasoning pattern control framework for LLM post-training. Specifically, AdapThink dynamically modulates reasoning preferences by jointly considering group-level correctness and process-level characteristics. The dispersion-based diversity sampling strategy further enhances training efficiency by explicitly maximizing the geometric dispersion of reasoning patterns. Experimental results reveal that AdapThink consistently outperforms existing methods across mathematical reasoning and code generation benchmarks.

Acknowledgements

This work was supported in part by the Smart Grid-National Science and Technology Major Project (2025ZD0803600, 2025ZD0803604), the National Natural Science Foundation of China under Grants 72571007, and by Beijing Nova Program (No. 20250484850).

Limitations and Future Directions

While AdapThink demonstrates strong performance across reasoning tasks, several limitations merit further investigation. First, our current framework operates at the *response-level*, computing reflection metrics $r(\hat{y})$ over entire generated sequences. A promising direction is to extend this to *token-level reflection control*, where reflection intensity could be modulated dynamically during generation based on local reasoning states.

Second, our experiments focus on single-turn reasoning tasks. Extending AdapThink to *multi-turn agentic RL scenarios* remains an open challenge. In such settings, reasoning patterns may need to adapt not only to correctness signals but also to task-specific constraints (e.g., API call efficiency, action costs).

Ethical Considerations

This paper presents work whose goal is to advance the field of machine learning. There are many potential societal consequences of our work, none of which we feel must be specifically highlighted here. Our experiments utilize publicly available datasets and models, and do not involve sensitive data or raise ethical concerns regarding data collection or usage.

References

- Pranjal Aggarwal and Sean Welleck. 2025. L1: Controlling how long a reasoning model thinks with reinforcement learning. *arXiv preprint arXiv:2503.04697*.
- 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. 2024. Do not think that much for $2+3=?$ on the overthinking of o1-like llms. *arXiv preprint arXiv:2412.21187*.
- Jacob Devlin, Ming-Wei Chang, Kenton Lee, and Kristina Toutanova. 2019. Bert: Pre-training of deep bidirectional transformers for language understanding. In *Proceedings of the 2019 conference of the*

North American chapter of the association for computational linguistics: human language technologies, volume 1 (long and short papers), pages 4171–4186.

- Daya Guo, Dejian Yang, Haowei Zhang, Junxiao Song, Ruoyu Zhang, Runxin Xu, Qihao Zhu, Shitong Ma, Peiyi Wang, Xiao Bi, and 1 others. 2025. Deepseek-r1: Incentivizing reasoning capability in llms via reinforcement learning. *arXiv preprint arXiv:2501.12948*.
- Rui Ha, Chaozhuo Li, Rui Pu, and Sen Su. 2025. From "aha moments" to controllable thinking: Toward meta-cognitive reasoning in large reasoning models via decoupled reasoning and control. *arXiv preprint arXiv:2508.04460*.
- Tingxu Han, Zhenting Wang, Chunrong Fang, Shiyu Zhao, Shiqing Ma, and Zhenyu Chen. 2024. Token-budget-aware llm reasoning. *arXiv preprint arXiv:2412.18547*.
- Mingyu Jin, Qinkai Yu, Dong Shu, Haiyan Zhao, Wenye Hua, Yanda Meng, Yongfeng Zhang, and Mengnan Du. 2024. The impact of reasoning step length on large language models. In *ACL (Findings)*.
- Yu Kang, Xianghui Sun, Liangyu Chen, and Wei Zou. 2025. C3ot: Generating shorter chain-of-thought without compromising effectiveness. In *Proceedings of the AAAI Conference on Artificial Intelligence*, volume 39, pages 24312–24320.
- Amirhossein Kazemnejad, Milad Aghajohari, Eva Portelance, Alessandro Sordoni, Siva Reddy, Aaron Courville, and Nicolas Le Roux. 2024. Vineppo: Unlocking rl potential for llm reasoning through refined credit assignment. *arXiv preprint arXiv:2410.01679*.
- Komal Kumar, Tajamul Ashraf, Omkar Thawakar, Rao Muhammad Anwer, Hisham Cholakkal, Mubarak Shah, Ming-Hsuan Yang, Phillip HS Torr, Fahad Shahbaz Khan, and Salman Khan. 2025. Llm post-training: A deep dive into reasoning large language models. *arXiv preprint arXiv:2502.21321*.
- Yuliang Liu, Junjie Lu, Zhaoling Chen, Chaofeng Qu, Jason Klein Liu, Chonghan Liu, Zefan Cai, Yunhui Xia, Li Zhao, Jiang Bian, Chuheng Zhang, Wei Shen, and Zhouhan Lin. 2025. Adaptivestep: Automatically dividing reasoning step through model confidence. *arXiv preprint arXiv:2502.13943*.
- Haotian Luo, Li Shen, Haiying He, Yibo Wang, Shiwei Liu, Wei Li, Naiqiang Tan, Xiaochun Cao, and Dacheng Tao. 2025a. O1-pruner: Length-harmonizing fine-tuning for o1-like reasoning pruning. *arXiv preprint arXiv: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. 2025b. Deepscaler: Surpassing o1-preview with a 1.5b model by scaling rl. Notion Blog.

- Yan Ma, Steffi Chern, Xuyang Shen, Yiran Zhong, and Pengfei Liu. 2025. Rethinking rl scaling for vision language models: A transparent, from-scratch framework and comprehensive evaluation scheme. *arXiv preprint arXiv:2504.02587*.
- Ning Miao, Yee Whye Teh, and Tom Rainforth. 2023. Selfcheck: Using llms to zero-shot check their own step-by-step reasoning. *arXiv preprint arXiv:2308.00436*.
- Niklas Muennighoff, Zitong Yang, Weijia Shi, Xiang Lisa Li, Li Fei-Fei, Hannaneh Hajishirzi, Luke Zettlemoyer, Percy Liang, Emmanuel Candès, and Tatsunori Hashimoto. 2025. s1: Simple test-time scaling. *arXiv preprint arXiv:2501.19393*.
- Sania Nayab, Giulio Rossolini, Marco Simoni, Andrea Saracino, Giorgio Buttazzo, Nicolamaria Manes, and Fabrizio Giacomelli. 2024. Concise thoughts: Impact of output length on llm reasoning and cost. *arXiv preprint arXiv:2407.19825*.
- OpenAI. 2024. Learning to reason with LLMs. <https://openai.com/index/learning-to-reason-with-llms>. Accessed: 2024.
- Matthew Renze and Erhan Guven. 2024a. The benefits of a concise chain of thought on problem-solving in large language models. In *2024 2nd International Conference on Foundation and Large Language Models (FLLM)*, pages 476–483. IEEE.
- Matthew Renze and Erhan Guven. 2024b. Self-reflection in llm agents: Effects on problem-solving performance. *arXiv preprint arXiv:2405.06682*.
- William Saunders, Catherine Yeh, Jeff Wu, Steven Bills, Long Ouyang, Jonathan Ward, and Jan Leike. 2022. Self-critiquing models for assisting human evaluators. *arXiv preprint arXiv:2206.05802*.
- Maohao Shen, Guangtao Zeng, Zhenting Qi, Zhang-Wei Hong, Zhenfang Chen, Wei Lu, Gregory Wornell, Subhro Das, David Cox, and Chuang Gan. 2025a. Satori: Reinforcement learning with chain-of-action-thought enhances llm reasoning via autoregressive search. *arXiv preprint arXiv:2502.02508*.
- Yi Shen, Jian Zhang, Jieyun Huang, Shuming Shi, Wenjing Zhang, Jiangze Yan, Ning Wang, Kai Wang, Zhaoxiang Liu, and Shiguo Lian. 2025b. Dast: Difficulty-adaptive slow-thinking for large reasoning models. *arXiv preprint arXiv:2503.04472*.
- Yang Sui, Yu-Neng Chuang, Guanchu Wang, Jiamu Zhang, Tianyi Zhang, Jiayi Yuan, Hongyi Liu, Andrew Wen, Shaochen Zhong, Hanjie Chen, and Xia Hu. 2025. Stop overthinking: A survey on efficient reasoning for large language models. *arXiv preprint arXiv:2503.16419*.
- Chenlong Wang, Yuanning Feng, Dongping Chen, Zhaoyang Chu, Ranjay Krishna, and Tianyi Zhou. 2025a. Wait, we don't need to "wait"! removing thinking tokens improves reasoning efficiency. *arXiv preprint arXiv:2506.08343*.
- Yue Wang, Qiuzhi Liu, Jiahao Xu, Tian Liang, Xingyu Chen, Zhiwei He, Linfeng Song, Dian Yu, Juntao Li, Zhuosheng Zhang, Rui Wang, Zhaopeng Tu, Haitao Mi, and Dong Yu. 2025b. Thoughts are all over the place: On the underthinking of o1-like llms. *arXiv preprint arXiv:2501.18585*.
- Yixuan Weng, Minjun Zhu, Fei Xia, Bin Li, Shizhu He, Shengping Liu, Bin Sun, Kang Liu, and Jun Zhao. 2023. Large language models are better reasoners with self-verification. In *Findings of the Association for Computational Linguistics: EMNLP 2023*, pages 2550–2575.
- Tian Xie, Zitian Gao, Qingnan Ren, Haoming Luo, Yuqian Hong, Bryan Dai, Joey Zhou, Kai Qiu, Zhirong Wu, and Chong Luo. 2025. Logic-rl: Unleashing llm reasoning with rule-based reinforcement learning. *arXiv preprint arXiv:2502.14768*.
- Silei Xu, Wenhao Xie, Lingxiao Zhao, and Pengcheng He. 2025. Chain of draft: Thinking faster by writing less. *arXiv preprint arXiv:2502.18600*.
- Chenxu Yang, Qingyi Si, Yongjie Duan, Zheliang Zhu, Chenyu Zhu, Qiaowei Li, Minghui Chen, Zheng Lin, and Weiping Wang. 2025. Dynamic early exit in reasoning models. *arXiv preprint arXiv:2504.15895*.
- Edward Yeo, Yuxuan Tong, Morry Niu, Graham Neubig, and Xiang Yue. 2025. Demystifying long chain-of-thought reasoning in llms. *arXiv preprint arXiv:2502.03373*.
- Qiyang Yu, Zheng Zhang, Ruofei Zhu, Yufeng Yuan, Xiaochen Zuo, Yu Yue, Tiantian Fan, Gaohong Liu, Lingjun Liu, Xin Liu, and 1 others. 2025. Dapo: An open-source llm reinforcement learning system at scale. *arXiv preprint arXiv:2503.14476*.
- Speed Zhu, Jianwei Cai, Guang Chen, Lulu Wu, Saiyong Yang, and Wiggin Zhou. 2025. Drive: Data curation best practices for reinforcement learning with verifiable reward in competitive code generation. *arXiv preprint arXiv:2511.06307*.

A Examples of Base Model’s Thinking Process

A.1 Example 1: Overthinking on a Simple Problem

In Figure 8, the model (DeepSeek Distilled Qwen 1.5B) was presented with a straightforward probability problem. While it reached the correct answer early in its reasoning process (using only 479 tokens), it continued to reflect unnecessarily. The model’s self-validation tendencies, marked by frequent use of phrases like “Wait”, “check”, and “verify”, led to multiple alternative approaches and verifications. This resulted in consuming 2,613 tokens—more than five times the tokens needed for the initial correct solution.

A.2 Example 2: Underthinking on a Complex Problem

Example 2 (Figure 9) shows the model attempting a more challenging modular arithmetic problem. Here, the model exhibited frequent but unproductive shifts in thinking, marked by numerous “Wait” and “Alternatively” statements. Unlike the first example where excessive reflection was merely inefficient, it reached the maximum token limit (4,096 token) before completing its calculation, resulting in an incorrect answer.

B Related Work

Induce Longer CoT. For inducing longer reasoning length, several works (Weng et al., 2023; Miao et al., 2023; Saunders et al., 2022; Renze and Guven, 2024b; Jin et al., 2024) have encouraged models to engage in deeper thinking through natural language feedback. For zero-shot CoT, (Weng et al., 2023; Miao et al., 2023) stimulate model self-reflection by performing backward verification or multiple response voting. For few-shot CoT with demonstrations, (Jin et al., 2024) introduced five general standardized patterns to induce models to simulate human thinking and reshape the CoT. However, these one-size-fits-all approaches ignore the diversity of problem-solving paths, only have a monotonous processing mode. Similarly, (Muennighoff et al., 2025) designed several budget forcing mechanisms to increase guidance in CoT, such as appending “Wait” multiple times when the model tries to end, forcing it to double-check. Likewise, (Shen et al., 2025a) introduced special meta-action markers like `<|continue|>`, `<|reflect|>`, and `<|explore|>`, enabling the model to restart

from intermediate steps, lengthen responses, and correct errors.

Induce Shorter CoT. Simple prompt methods are also effective for encouraging a shorter CoT (Nayab et al., 2024; Xu et al., 2025; Renze and Guven, 2024a). For example, (Muennighoff et al., 2025) added “Final answer” to terminate the model’s thinking process. Besides, (Jin et al., 2024; Kang et al., 2025) used stronger models to compress long CoTs semantically into shorter ones. (Yeo et al., 2025; Shen et al., 2025b; Aggarwal and Welleck, 2025; Yu et al., 2025) design different length-budget signals from the perspective of rule-based reward design, encouraging models to balance accuracy and token efficiency during the thinking process. Although (Shen et al., 2025b) already takes problem complexity and model confidence into account for the budget, its adoption of a uniformly shorter response preference is less suitable for models with weaker reasoning abilities.

Moreover, some studies have further focused on addressing overthinking and underthinking in long CoTs. (Wang et al., 2025b; Sui et al., 2025) introduced thought switching penalties to influence the token decoding probability distribution early in CoT generation, reducing initial thought-switching. Similarly, (Chen et al., 2024) presented an efficiency metric to evaluate each token’s contribution to accuracy and used length preference optimization to achieve more efficient CoT patterns. However, both approaches depend on auxiliary judgments from more powerful reasoning models, making the performance improvements inherently constrained by the capabilities of the reference models.

C AdapThink Algorithm Details

Pseudo Code. We provide the complete training process of AdapThink in Algorithm 1 and the sampling strategy in Algorithm 2.

Objective. Following the setting of GRPO (Guo et al., 2025), the objective is formulated as:

$$\frac{1}{G} \sum_{i=1}^G \frac{1}{|y_i|} \sum_{t=1}^{|y_i|} \min \left(\rho_{i,t}(\theta) \hat{A}_{i,t}, \text{clip}(\rho_{i,t}(\theta), 1 - \varepsilon, 1 + \varepsilon) \hat{A}_{i,t} \right) - \beta \cdot \mathbb{D}_{\text{KL}}(\pi_{\theta} || \pi_{\text{ref}}) \quad (8)$$

where $\mathcal{G} = \{y_1, y_2, \dots, y_G\}$ represents a G -size group of responses sampled from $\pi_{\theta}(\cdot|x)$

Overthinking Example

Question

A bag contains 4 red balls and 3 blue balls. Two balls are drawn without replacement. What is the probability that both balls are of different colors?

Model Response

Thinking Process

Okay, let's see. Well, I can think ... **Alternatively**, maybe I should ... **Wait**, is it right? ... **Alternatively**, maybe it is easier to ... **So, the probability is 4/7.**
But ... **Alternatively** ... But let me **check** both approaches ... **Alternatively** ... Approach 1 ... Wait ... Let me **check** ... Approach 2 ... **Alternatively** ... **Wait**, but let me just **verify** ... So all methods lead to 4/7.

****Final Answer**:**

The probability that both balls are of different colors is $\frac{4}{7}$.

All methods confirm the probability that both balls are of different colors is $\frac{4}{7}$.

Correctness

Correct

Tokens Consumption

2613 (479 for first generating the correct answer.)

Reflection Segments Count

19

Figure 8: Example 1. Overthinking on a Simple Problem

Underthinking Example

Question

Let $a_n = 6^n + 8^n$. Determine the remainder upon dividing a_{83} by 49?

Model Response

Thinking Process

Okay, so I need to find the remainder ... Let me think ... First ... But **wait** ... **Alternatively** ... **Alternatively** ... **Wait** ... Let's **check**... **Wait**, perhaps there's a better way ... Oh! **Wait** a second ... **Wait** ... Okay, so that's the first part ... **Wait** ... **Wait** first ... **Wait**, perhaps ... **Wait**, but just to make sure ... Similarly, $8^7 \equiv 1 \pmod{13}$, 83 divided by 7 gives $11 \cdot 7 =$

Correctness

Incorrect

Tokens Consumption

4096 (max token limit)

Reflection Segments Count

35

Figure 9: Example 2. Underthinking on a Complex Problem

Algorithm 1 AdapThink

Require: Pre-trained model π_θ , Dataset \mathcal{D} , Up-sample factor K

Ensure: Updated model π_θ

- 1: **for** each $(x, y^*) \in \mathcal{D}$ **do**
 - 2: Generate \mathcal{G} responses using Algorithm 2
 - 3: Calculate φ using Equation 1
 - 4: **for** Sub-group $\mathcal{G}_{\text{sub}} \in \{\mathcal{G}_T, \mathcal{G}_F\}$ **do**
 - 5: Calculate length metric μ_l and reflection segments metric μ_r using Equation 3
 - 6: Calculate reasoning process reward r_{ref} using Equation. 4
 - 7: Get total rewards $r_{\text{total}}(\hat{y}_i) = r_{\text{answer}}(\hat{y}_i) + r_{\text{ref}}(\hat{y}_i)$
 - 8: **end for**
 - 9: Update π_θ using Equation 8
 - 10: **end for**
-

for each input x ; $\rho_{i,t}(\theta)$ is the probability ratio $\frac{\pi_\theta(y_i^t | y_i^{<t}, x)}{\pi_{\theta_{\text{old}}}(y_i^t | y_i^{<t}, x)}$ between current policy and old policy $\pi_{\theta_{\text{old}}}$ for the i -th responses' t -th token, ε limits the magnitude of policy updates; and \mathbb{D}_{KL} constrains the policy π_θ from deviating too far from a reference policy π_{ref} . Crucially, $\hat{A}_{i,t}$ denotes the estimated advantage of response y for input x within group \mathcal{G} , which is derived from the standardization of rewards using the statistical properties of group \mathcal{G} . For the i -th response $y_i \in \mathcal{G}$ with reward $r_i = r(x, y_i)$, the estimated advantage is:

$$\hat{A}_{i,t} = \frac{r_i - \text{mean}(\{r_\ell\})}{\sqrt{\text{std}^2(\{r_\ell\}) + \varepsilon}} \quad (9)$$

where $\text{mean}(\{r_\ell\})$ and $\text{std}^2(\{r_\ell\})$ are the empirical mean and variance of rewards in group \mathcal{G} , respectively.

For the reward r_i , we incorporate two components: the outcome reward r_{outcome} and the reflection-related process reward r_{ref} . The definition of r_{ref} is provided in Equation 4. The reward r_{outcome} is defined as follows:

$$r_{\text{outcome}}(x, y) = \begin{cases} 1, & \text{if } y \text{ is correct} \\ 0, & \text{otherwise} \end{cases}.$$

D More Results

Analysis of Detailed Reflection Words Distribution. To provide deeper insights into *how different post-training methods affect the model's reasoning behavior*, we analyze the distribution of reflection words across correct and incorrect responses,

Algorithm 2 Dispersion-based Diversity Sampling

Require: Question x , target size n , oversampling factor K

Ensure: Diverse sample set \mathcal{G} with $|\mathcal{G}| = n$

- 1: *// Stage 1: Oversampling and Diversity Quantification*
 - 2: $\mathcal{G}' \leftarrow \text{GenerateSamples}(x, K \times n)$
 - 3: $\mathbf{F} \leftarrow [l(\hat{y}_i), r(\hat{y}_i)]_{i=1}^{K \times n}$
 - 4: $\mathcal{D}(\mathcal{G}') \leftarrow \log \det(\text{Cov}(\mathbf{F}))$
 - 5: *// Stage 2: Diversity-Maximizing Downsampling*
 - 6: **if** $\varphi(\mathcal{G}') \in (0, 1)$ **then**
 - 7: $\mathcal{G}^* \leftarrow \emptyset$
 - 8: **while** $|\mathcal{G}^*| < n$ **do**
 - 9: $\hat{y}^* \leftarrow \arg \max_{\hat{y} \in \mathcal{G}' \setminus \mathcal{G}^*} \mathcal{D}(\mathcal{G}^* \cup \{\hat{y}\})$
 - 10: **s.t.** $\varphi(\mathcal{G}^* \cup \{\hat{y}^*\}) \in (0, 1)$
 - 11: $\mathcal{G}^* \leftarrow \mathcal{G}^* \cup \{\hat{y}^*\}$
 - 12: **end while** \mathcal{G}^*
 - 13: **else**
 - 14: $\arg \max_{\mathcal{G} \subset \mathcal{G}', |\mathcal{G}|=n} \mathcal{D}(\mathcal{G})$
 - 15: **end if**
-

as shown in Figure 11. For most reflection-related words, we observe that AdapThink significantly reduces the frequency in both correct and incorrect responses compared to the base model, suggesting a more efficient reasoning process. In contrast, GRPO and TLB exhibit less pronounced changes in these reflection words; LCPO shows elevated counts of pause words, particularly in incorrect responses.

E Hyperparameter Configuration

The core hyperparameters used in AdapThink are summarized in Table 6. The training is on 8 NVIDIA H20 GPUs, taking approximately 22 hours to complete 1000 steps for 1.5B model and 1 day 16 hours for 7B model.

F Example Responses: Base Model vs. AdapThink

We present representative response examples from the DeepSeek-Distill-Qwen 1.5B model under an 8K token limit, comparing the base model (Figure 13) with the AdapThink post-trained model (Figure 14).

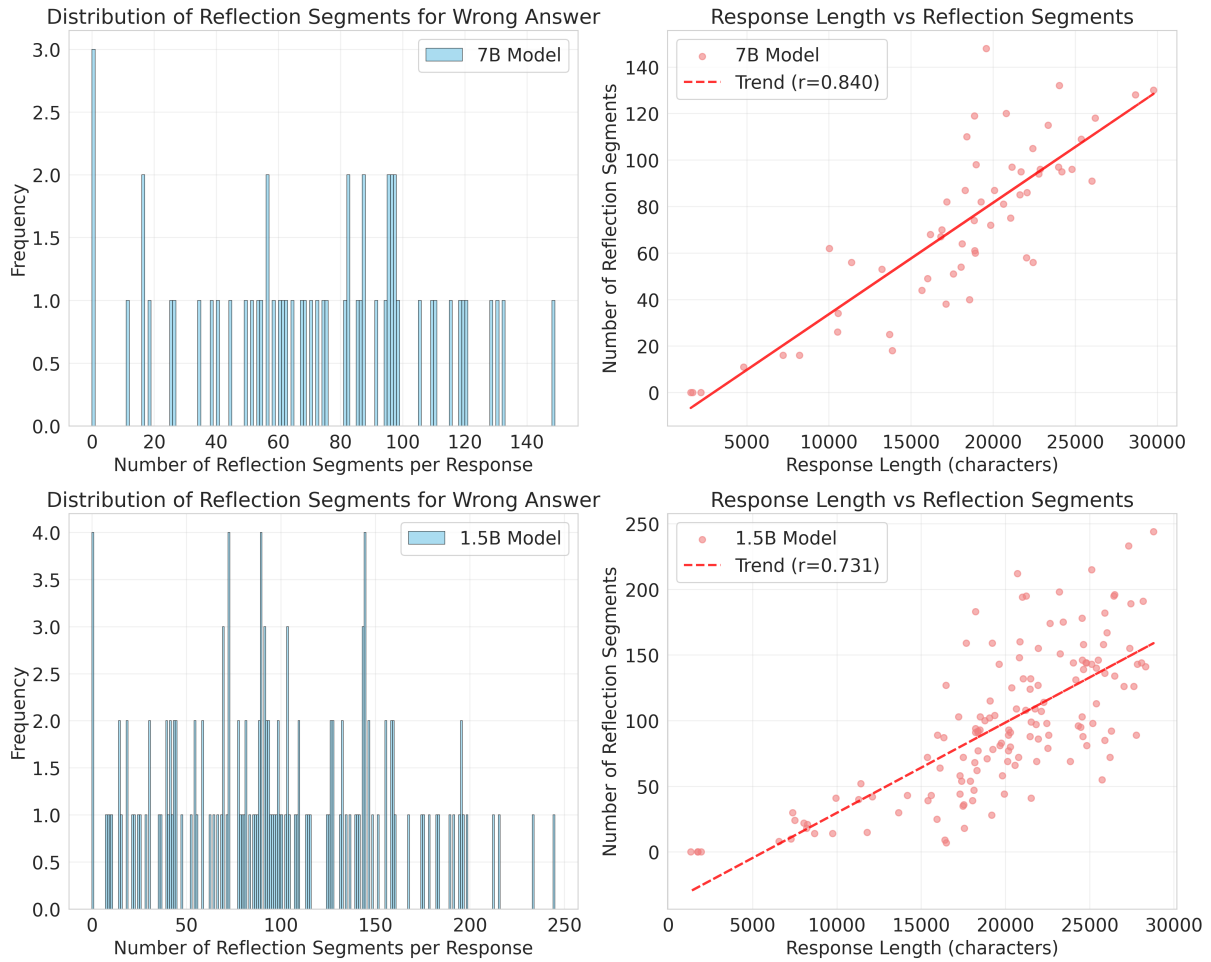


Figure 10: **Statistical Overview** of the **wrong** answers' reasoning pattern.

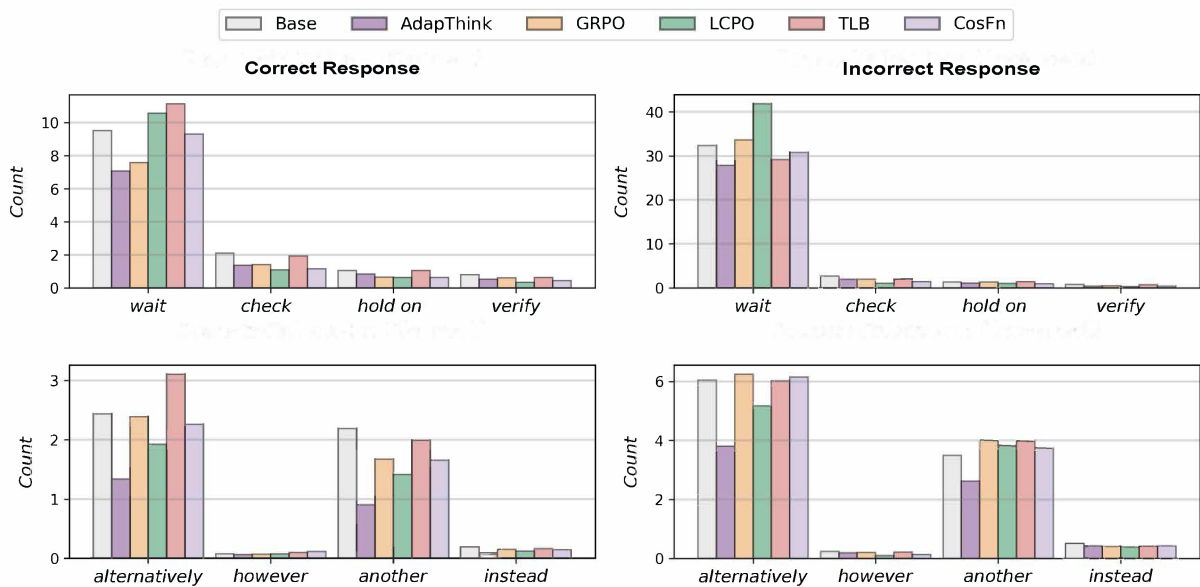


Figure 11: **Distribution of Reflection Words** across different post-training methods for correct and incorrect responses in MATH-500 datasets.



Figure 12: Evaluate dynamics of Adapthink at 8-step training intervals on competitive code generation benchmarks.

Table 6: Core hyperparameters for AdapThink post-training configuration

Category	Parameter	Value
Model Settings	Base Model	DeepSeek-R1-Distill-Qwen-1.5B, 7B
	Torch Dtype	bfloat16
	Max Prompt Length	1024
	Max Response Length	2048
Training Settings	Learning Rate	2e-6
	Number of Epochs	5
	Batch Size	8
	Gradient Accumulation Steps	8
	Warmup Steps	20
RL Settings	threshold φ_{thr}	0.1
	Oversampling ratio K	2
	Number of Generations $ \mathcal{G}' $	16
	Target Generations $ \mathcal{G} $	8
Evaluate Settings	Max Response Length	32768
	Number of Generations	16
	Temperature	0.7
	Top-p	0.95

DeepSeek-distilled Qwen 1.5B Example (from AIME 2025)

Question

Find the number of ordered pairs (x, y) , where both x and y are integers between -100 and 100 , inclusive, such that $12x^2 - xy - 6y^2 = 0$.

Model Response

Thinking Process

Alright, so I have this problem here: ... Hmm, okay. Let me try to figure this out step by step Let me write it down again:

$$12x^2 - xy - 6y^2 = 0$$

Alternatively, maybe factor the quadratic expression as $(ax + by)(cx + dy) = 0$. Let me try that...

Alternatively, maybe factor it as $(ax + by)(cx + dy) = 0$...

Alternatively, perhaps factor out a 3:

$$3(4x^2 - \frac{1}{3}xy - 2y^2) = 0$$

But that doesn't seem helpful.

Alternatively, maybe factor as $(4x + ay)(3x + by) = 0$...

So, I think I can confidently say the answer is 117. ****Final Answer****: 117.

We need to find the number of ordered pairs (x, y) where both x and y are integers between -100 and 100 , inclusive, such that $12x^2 - xy - 6y^2 = 0$... The total number of solutions is:

$$67 + 51 - 1 = 117$$

Thus, the number of ordered pairs (x, y) is: 117.

Correctness

Correct

Tokens Consumption

7377

Reflection Segments Count

46

Figure 13: A DeepSeek Distilled Qwen 1.5B example. The question is sampled from AIME 2025.

AdapThink Example (from AIME 2025)

Question

Find the number of ordered pairs (x, y) , where both x and y are integers between -100 and 100 , inclusive, such that $12x^2 - xy - 6y^2 = 0$.

Model Response

Thinking Process

Okay, so I need to find the number of ordered pairs (x, y) where both x and y are integers between -100 and 100 , inclusive. The condition given is $12x^2 - xy - 6y^2 = 0$. Hmm, let me try to figure this out step by step.

First, looking at the equation: $12x^2 - xy - 6y^2 = 0$. It seems like a quadratic in terms of x and y . Maybe I can factor it or rearrange it somehow. ... Let's see:

$$12x^2 - xy - 6y^2 = 0$$

Divide both sides by 3:

$$4x^2 - \frac{xy}{3} - 2y^2 = 0$$

Wait, that doesn't help much because we have a fraction. Maybe factoring differently.

...

So, seems consistent. Therefore, the number of ordered pairs is 117. ****Final Answer****: The number of ordered pairs is .

To find the number of ordered pairs (x, y) where both x and y are integers between -100 and 100 , inclusive, such that $12x^2 - xy - 6y^2 = 0$, we start by ... Thus, we subtract 1 to avoid double-counting. The total number of ordered pairs is:

$$51 + 67 - 1 = 117$$

Thus, the number of ordered pairs is .

Correctness

Correct

Tokens Consumption

4502

Reflection Segments Count

19

Figure 14: An AdapThink example. The question is sampled from AIME 2025.
9825