Lost at the Beginning of Reasoning

Anonymous ACL submission

Abstract

Large language models (LLMs) with long chain-of-thought (long-CoT) reasoning often produce traces with backtracking, self-reflection, and self-correction. Yet it remains unclear how reliably these traces help models recover from early mistakes. We study this question by examining the first reasoning step. Across long-CoT reasoning models, we find that the first reasoning step is consistently the most semantically aligned with the final conclusion, suggesting that the beginning of the trace shapes the direction of the final answer. When the first-step prediction is incorrect, final accuracy drops substantially and token usage increases, despite many subsequent reasoning steps. This pattern persists under exploratory prompts and controlled perturbations. Building on this observation, we further ask whether the first step can serve as an early signal of trajectory quality. This leads to a simple early-pruning strategy that matches standard majority voting while reducing inference cost by up to 70%. Overall, our results suggest that long-CoT models are often lost at the beginning of reasoning, with the first step serving as both a vulnerable stage for error recovery and a useful signal for efficient inference.

1 Introduction

Large language models (LLMs) with long chain-of-thought (long-CoT) reasoning have substantially advanced complex problem-solving in mathematics, science, and code (Brown et al., 2020; Hurst et al., 2024; Anthropic, 2025; Team et al., 2024; Guo et al., 2025; Yang et al., 2025a; Wen et al., 2025; He et al., 2025). This progress is often attributed to their ability to backtrack, reflect on earlier steps, and correct mistakes as they refine their responses (Jaech et al., 2024; Guo et al., 2025; Ma et al., 2025; Yang et al., 2025a).

Whether long-CoT reasoning actually enables reliable error correction remains unclear. Long-CoT models often "overthink," producing lengthy traces that contribute little to the final answer (Chiang and Lee, 2024; Zhang et al., 2024; Wang et al., 2025b; Liao et al., 2025b,a). If extended reasoning does not alter the model's conclusion, it remains in question whether the additional steps detect and correct earlier mistakes, or simply justify a conclusion already reached.

In this work, we examine whether long-CoT models can recover from an error at the early stage of reasoning. We focus on models with explicit thinking traces, such as DeepSeek-R1 (Guo et al., 2025), since these traces allow us to inspect and intervene on intermediate steps directly. Such models are not merely generating continuations from a prefix; their reasoning traces include backtracking and self-correction, behaviors that emerge through reinforcement learning training (Guo et al., 2025; Yang et al., 2025a; Ma et al., 2025).

Across five open- and closed-source long-CoT model families, we find that the first reasoning step shows the strongest alignment with the final conclusion, suggesting that the initial direction of reasoning plays a strong role in shaping the final answer. Controlled perturbation experiments across open-source models confirm this: when the first step is incorrect, final accuracy drops by approximately 40% while inference tokens increase by a similar amount, even though subsequent generation has every opportunity to revise and correct the mistake. The effect persists under prompts that explicitly encourage exploration and under minor perturbations of an otherwise-correct first step. Explicit long-CoT reasoning thus offers only limited self-correction when the initial direction is incorrect.

If the first reasoning step is closely tied to the final outcome, can its quality — evaluated before generation completes — identify which trajectories are worth continuing? We test this with a simple early-pruning algorithm: for each question, we

sample multiple candidate first steps, score them with a process reward model, and continue full generation only for the most promising. Across the same five model families and five benchmarks in mathematics, science, and programming, this approach matches standard majority voting at $N = 64$ while reducing the inference budget by up to 70%. The success of this strategy is consistent with our central finding: the quality of the first reasoning step is informative about the quality of the trajectory it precedes.

Contributions. (1) We show that the first reasoning step is consistently the most semantically aligned with the final conclusion across five long-CoT model families (§3.1). (2) Through controlled perturbation experiments across open-source models from these families, we find that long-CoT LLMs often fail to recover from first-step errors, with an average accuracy drop of approximately 40% that persists across prompting and perturbation settings (§3.2, §3.3). (3) We introduce an early-pruning algorithm that uses first-step scores to select which trajectories to continue, matching majority voting at $N = 64$ while reducing inference cost by up to 70%. This both supports the predictive role of the first reasoning step and yields a practical efficiency strategy (§4).

2 Related Work

Positional and prefix effects in language models. Prior work has shown that position affects how language models use information, especially in input contexts such as the "lost in the middle" effect (Liu et al., 2024). Other work studies position within generation, including attention sinks (Xiao et al., 2024; Barbero et al., 2025), short prefixes that predict CoT correctness or faithfulness (Lanham et al., 2023; Zhou et al., 2022), and theoretical accounts of error propagation in reasoning chains (Gan et al., 2025; You et al., 2025). While these studies show that early content can matter, they do not address whether explicit long-CoT reasoning can recover from early mistakes in its own generated trace. This is the focus of our work. Long-CoT models produce extended traces with backtracking, reflection, and self-correction (Jaech et al., 2024; Guo et al., 2025; Ma et al., 2025; Yang et al., 2025a), so later reasoning is expected to revise flawed early steps. We show that this recovery is limited: the first reasoning step remains strongly associated with later outcomes and provides a useful signal for early pruning.

Efficient Test-Time Reasoning. Test-time scaling improves reasoning by allocating more compute to sampling, verification, and aggregation, but this becomes costly for long-CoT models. Prior work improves the accuracy–compute trade-off by reducing self-consistency samples (Li et al., 2024; Wan et al., 2025; Aggarwal et al., 2023; Xue et al., 2023), shortening reasoning traces (Chen et al., 2024; Luo et al., 2025; Hou et al., 2025; Fu et al., 2025a; Yang et al., 2025b), or filtering low-quality generations with confidence signals (Fu et al., 2025b). Our method intervenes earlier. Rather than generating full traces or pruning throughout the chain, we score candidate first reasoning steps and continue only the most promising starts. This directly exploits our empirical finding that the first step carries substantial signal about the quality of the trajectory that follows.

3 Lost at the Beginning of Reasoning

Motivated by Liu et al. (2024), which demonstrates that query-relevant information is more impactful when positioned at either the beginning or end of an LLM's context window, we first investigate whether a similar positional effect exists in long-CoT reasoning (§3.1). Our analysis reveals that the first reasoning step has great impact to the final conclusion. To validate this observation, we further perform two ablation studies, confirming the critical role of the first step in determining the model's final prediction (§3.2 and §3.3).

Notation. Let p represent the input prompt, consisting of both a system instruction and a user query. A reasoning model $\mathcal{M}$ produces a sequence of CoT reasoning steps $t = [t_1, t_2, ..., t_T]$, followed by a final conclusion c_T, such that the complete model output is given by $t \oplus c_T = \mathcal{M}(p)$, where $\oplus$ means concatenation. In models such as DeepSeek-R1 (Guo et al., 2025) and Qwen3 (Team, 2025), the input-output format adheres to the following:

$$p \text{ <think> } t_1, t_2, \ldots, t_T \text{ </think> } c_T$$

The final prediction q_T is then derived by applying an extraction function g to the conclusion, i.e., $q_T = g(c_T)$, where g may, for example, extract values enclosed within `\boxed{}`.

The conclusion c_T can be interpreted as a summary of the essential reasoning steps leading to the final prediction. This raises an interesting question: *Is there a positional bias in how reasoning*

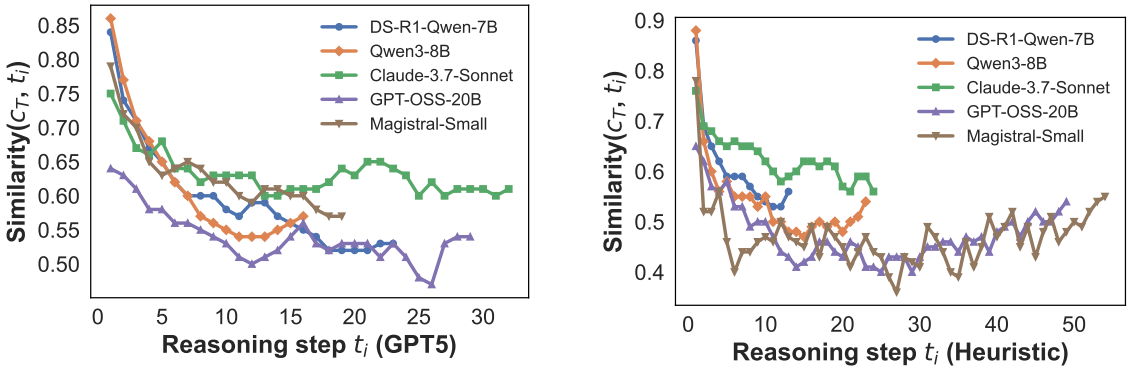

Figure 1: Cosine similarity between the embeddings of the i-th reasoning step t_i and the final conclusion c_T, using the average number of reasoning steps for interpolation. The reasoning steps are segmented either by GPT-5 (left) or by heuristic rules (right). See Figure F.1 for results interpolated based on the maximum number of reasoning steps.

steps contribute to the conclusion? In other words, do certain steps have a disproportionately greater influence on c_T than others?

3.1 Similarity Between Reasoning Steps And the Final Conclusion

To assess how intermediate reasoning contributes to the final outcome, we measure the semantic similarity between each reasoning step $\{t_i\}_{i=1}^{T}$ and the final conclusion c_T. This analysis reveals whether the reasoning process gradually aligns with the correct answer or diverges along the way.

Experimental Setup. We evaluate 60 questions from AIME24 and AIME25 (MAA Committees, 2025) using DeepSeek-R1-Distill-Qwen-7B (abbreviated as DS-R1-Qwen-7B in the remainder of this paper) (Guo et al., 2025), Qwen3-8B (Yang et al., 2025a), Claude-3.7-Sonnet with thinking (Anthropic, 2025), GPT-OSS-20B (Agarwal et al., 2025), and Magistral-Small (Rastogi et al., 2025).[1] Generations are produced with `temperature=1.0`, `top_p=0.9`, `min_p=0.05`, and `max_tokens=32768`; for Claude-3.7-Sonnet, only `max_tokens` is set. All subsequent experiments adopt this hyperparameter configuration.

Segmentation of Reasoning Steps. We define a reasoning step as a complete logical leap or self-contained unit (Xiong et al., 2025), and segment reasoning traces with GPT-5.[2] To complement this setup, we also employ heuristic segmentation based on reasoning switch keywords (Wang et al., 2025a), with details provided in Appendix C.

[1]For reproducibility, the exact model identifiers are provided in Appendix D.

[2]By default, we use GPT-5 mini for step segmentation; for GPT-OSS-20B, we use GPT-5, as the mini variant is incompatible. The segmentation prompt is provided in Appendix A.

Similarity Computation. We compute the semantic similarity between each step t_i and the conclusion c_T by taking the cosine similarity of their embeddings obtained from all-MiniLM-L6-v2 (Reimers and Gurevych, 2019; Wang et al., 2020). To avoid inflated similarity from problem restatement, we use GPT-5 mini to remove question-overlap text at the beginning of traces. As a robustness check, we also report results with SPLADE similarity (Formal et al., 2021) in Appendix F, confirming that our findings are not specific to dense embeddings. Since traces vary in length, similarity curves are interpolated to a fixed number of steps, either the average or maximum length.

This setup allows us to capture how reasoning trajectories semantically converge toward—or deviate from—the final answer across different models.

Result. Figure 1 shows that the first reasoning step exhibits the highest similarity to the final conclusion, after which similarity drops sharply. Beyond the initial few steps, similarity stabilizes at a lower level, with only minor fluctuations across the remainder of the reasoning process. Notably, we have removed the question restatement from both the intermediate steps and the final conclusion, so the similarity mainly reflects the reasoning parts.

These results suggest that the first step t_1 is most closely aligned with the final conclusion and likely sets the overall direction of the reasoning. Additional results on the GPQA dataset (Figure E.1) and with SPLADE similarities (Figure F.2) show the same trend, confirming that this pattern is robust across datasets and different similarity metrics. Taken together, these findings indicate that the first reasoning step plays a disproportionately important role in shaping the final conclusion.

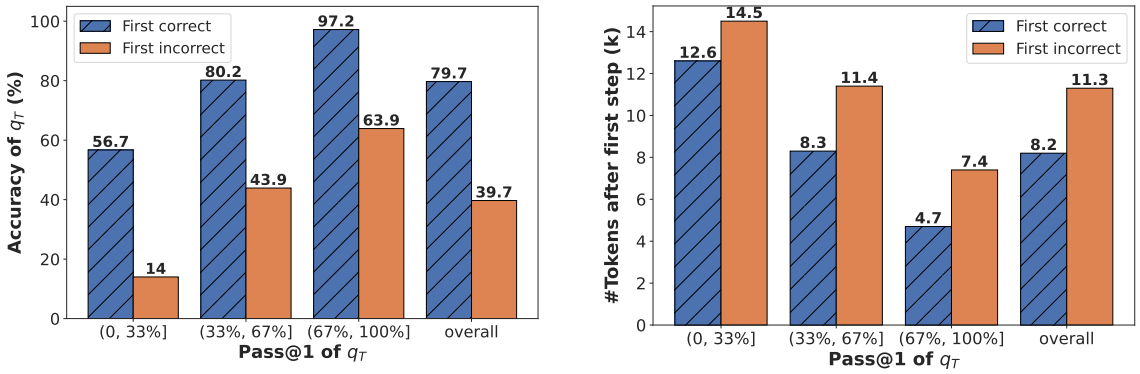

Figure 2: Accuracy and token usage on DS-R1-Qwen-7B. **Left:** Relationship between the accuracy of the final prediction (q_T) and the correctness of the prediction based solely on the first reasoning step (q_1) across different difficulty levels. When q_1 is incorrect, q_T is significantly more likely to be incorrect. **Right:** Number of tokens used after the first reasoning step t_1, i.e., for subsequent steps $[t_2, t_3, \ldots, t_T]$. Even when q_1 is correct, the model often expends a large number of additional tokens, indicating the overthinking behavior.

3.2 Correlation between the First Reasoning Step and the Final Prediction

Given that the first reasoning step closely resembles the final conclusion, we investigate whether the essential reasoning required for the final prediction is already encapsulated in the first step. To this end, we analyze the prediction when conditioned solely on the first reasoning step. Specifically, we compute $c_1 = \mathcal{M}(p\text{<think>}t_1\text{</think>})$, and derive the corresponding prediction $q_1 = g(c_1)$, which we compare against the ground truth a. Based on this comparison, we categorize each first reasoning step as either `first correct` (if $q_1 = a$) or `first incorrect` (otherwise).

Experimental Setup. To better analyze the correlation, we sample 64 CoT traces per question using the same datasets in §3.1. We exclude questions for which all 64 CoT traces result in either correct or incorrect predictions, as these are considered either too easy or too difficult, respectively, yielding 38 questions for DS-R1-Qwen-7B and 37 for Qwen3-8B. For each remaining question and its corresponding first reasoning step t_1, we obtain the initial prediction q_1 as previously described. To better visualize the final outcomes, we categorize the questions into three groups based on the pass@1 accuracy of the final prediction q_T,[3] corresponding to the intervals (0, 33%], (33%, 67%], and (67%, 100%). A higher pass@1 indicates a simpler question. This grouping allows us to assess whether our observations hold consistently across varying levels of question difficulty.

Result. As shown in Figure 2 (Left), when the first reasoning step is incorrect, the model's final prediction is also likely to be incorrect. On average, final prediction accuracy drops by 40% when the first reasoning step is incorrect, with the most substantial decrease (43%) occurring for difficult questions (0–33% range) and a notable decline (33%) even for easier questions (67–100% range). In addition, we also compute the Pearson correlation between the correctness of the first prediction p_1 and the final prediction p_T over all questions. The coefficient $r = 0.60$ and p-value $p = 0.0$ denote a moderately strong positive correlation. All these results underscore the pivotal role of the first reasoning step in steering the model toward a correct final answer, particularly in more complex instances where recovery from early mistakes is more challenging. Extending this analysis to DeepSeek and Qwen models of different sizes yields consistent trends: final accuracy remains substantially higher when the first step is correct, and the accuracy gap persists as model scale increases (Figure G.2).

Figure 2 (Right) further illustrates the model's tendency toward overthinking. Even when the first reasoning step is sufficient to arrive at the correct answer (i.e., first correct), the model continues to generate a substantial number of additional reasoning tokens, the same scale in length to those generated following an incorrect first step. Both findings are also observed for Qwen3-8B in Figure G.1, reinforcing this pattern across models.

3.3 Minor Perturbation to the Correct First Step Leads to Significant Loss

Building on our findings in §3.2, which demonstrate a positive correlation between the model's

[3] That is, the average accuracy across 64 CoT traces.

first and final predictions, we further investigate the significance of the first reasoning step by introducing minor perturbations. Specifically, we slightly alter an initially correct reasoning step and provide it as input to the model to assess whether the model can recover from such errors.

Experimental Setup. Unlike §3.2, where we analyze the correctness of the first reasoning step t_1, here we treat the final correct conclusion c_T, which satisfies $q_T = g(c_T) = a$, as the new first reasoning step, denoted t'_1. This choice ensures that this synthetic first step contains all necessary reasoning for arriving at the correct answer, which cannot be fully guaranteed by t_1. As illustrated in Figure 2 (Left), a first correct reasoning step can still lead to an incorrect final prediction.[4] To construct t'_1, we apply the following perturbations to c_T: (1) we remove not only the explicit answer formatting (e.g., `\boxed{a}`) but also any surrounding sentences that may directly disclose or repeat the final answer; (2) the resulting text from (1) is treated as the correct version of t'_1 (serving as our baseline); (3) we generate an incorrect version by replacing the correct answer a in the remaining reasoning with $a \pm 1$ or $a \pm 10$.[5] Please refer to Appendix J for a concrete example.

These perturbations are minimal, as they preserve the core reasoning structure while only altering the final prediction in the incorrect variant. We then combine the prompt p with the modified first reasoning step t'_1 and input it to the model as $\mathcal{M}(p\text{<think>}t'_1\text{Alternatively})$ to assess subsequent reasoning behavior.

Result. As shown in Table 1, we make two key observations: (1) Smaller models rarely reach 100% accuracy even when the first reasoning step is correct, suggesting that they may revise or deviate from their initial reasoning. In contrast, larger models (e.g., DS-R1-32B) consistently achieve 100% accuracy given a correct first step, indicating greater stability. (2) There is a substantial drop in accuracy when the first reasoning step is incorrect, highlighting that even minor errors early in the reasoning process can significantly affect the final prediction. These findings further strengthen our previous observation: the first reasoning step greatly shapes the overall reasoning path.

Model	Correct (%)	Incorrect (%)
DS-R1-Qwen-1.5B	95.4	64.4
DS-R1-Qwen-7B	94.8	28.5
DS-R1-Qwen-32B	100.0	85.8
Qwen3-1.7B	96.0	46.6
Qwen3-8B	71.4	37.0
Qwen3-30B-A3B	100.0	74.7
Qwen3-235B-A22B	100.0	78.7

Table 1: Perturbation experiments. Accuracy for correct vs. incorrect first step. Even minor perturbations cause significant drops.

In sum, the long-CoT model is particularly vulnerable at the initial stage of the reasoning process; an error in the first step can propagate and substantially degrade the final prediction. A subsequent question arises: *Can we develop a method to identify and retain more promising first reasoning steps while discarding suboptimal ones to enhance the overall generation efficiency?*

4 First-Step Guided Early Pruning

In this section, we propose an efficient and straightforward sampling method to identify a promising first reasoning step. By doing so, we can terminate the generation process early when a suboptimal first step is detected, thereby reducing unnecessary computational overhead.

4.1 Problem Definition

In addition to the notation introduced in §3, we incorporate a random seed ϵ to introduce stochasticity into the sampling process. Specifically, a sampled trace is computed as $t \oplus c_T = \mathcal{M}(p, \epsilon)$. By varying the random seed ϵ^n, we obtain diverse generations, denoted as $t^n \oplus c_T^n = \mathcal{M}(p, \epsilon^n)$, where $t^n = [t_1^n, t_2^n, \ldots, t_T^n]$.[6]

A widely adopted technique for reasoning tasks is majority voting or self-consistency (Wang et al., 2022). To promote exploration of the reasoning space, models are typically sampled at high temperatures, resulting in diverse outputs. Majority voting then serves to aggregate these outputs by selecting the most frequent final prediction. Formally, majority voting over K samples is defined as:

$$q_{\text{maj@}K} = \text{mode}(\{q_T^n\}_{n=1}^K), \text{ where } q_T^n = g(c_T^n).$$

[4] c_T is a summarization of all effective reasoning from the intermediate steps; If $q_T = a$, we can safely assume that c_T contains the necessary reasoning for obtaining a correct prediction, however, this is not the case for t_1. Xu et al. (2025) observes that the model sometimes do extra reasoning in the conclusion with a truncated trace. To make sure all necessary reasoning containing in the first step, we therefor use c_T as the synthetic first step.

[5] The answer of AIME question is an integer in $[0, 999)$.

[6] In prior experiments, we sampled 64 CoT traces per question using 64 distinct random seeds $\{\epsilon^n\}_{n=1}^{64}$.

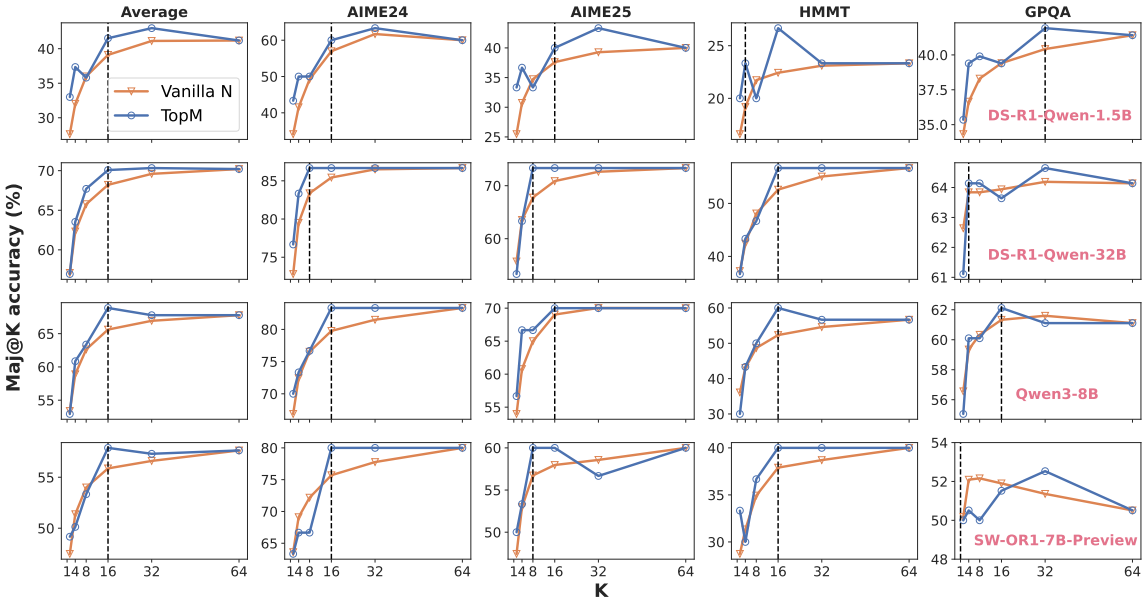

Figure 3: Majority voting accuracy with different number of samplings for three LLMs. The vertical dashed line denotes the smallest M whose accuracy is equal to or larger than the accuracy of $N = 64$.

However, for models generating long CoT traces, majority voting becomes computationally expensive, as it requires sampling N complete reasoning paths independently. In this section, we propose a more efficient approach that samples only M full traces, where $M < N$, while maintaining comparable performance to the one with N samplings.

4.2 Methodology

In §3, we demonstrated a strong positive correlation between the first reasoning step and the final prediction. This observation motivates a method that identifies the top M most promising first reasoning steps out of a total of N, and continues generation only for these selected M candidates, while discarding the remaining $(N - M)$.

Let a reasoning model generate N candidate first reasoning step $\{t_1^1, t_1^2, \ldots, t_1^N\}$ from a prompt p with different random seeds $\{\epsilon^n\}_{n=1}^N$. Each t_1^n is the first reasoning step of a full reasoning trajectory. We define a scoring function $r : t_1^n \to \mathbb{R}$ that estimates the promise of a first step, e.g., rating from a reward model. We then select the top M first steps based on their scores:

$$\mathcal{R}_{\text{top}} = \text{TopM}(\{r(t_1^n)\}_{n=1}^N).$$

Only the selected M first steps $\{t_1^n \mid n \in \mathcal{R}_{\text{top}}\}$ are used for further multi-step generation. The remaining $(N - M)$ are discarded. Since the first step typically requires only a small number of tokens to generate, this selection approach efficiently reduces computation by avoiding full sampling for the less promising $(N - M)$ candidates.

4.3 Experiments

Setup. We evaluate five families of reasoning models, DS-R1-Qwen (Guo et al., 2025), Qwen3 (Yang et al., 2025a), Skywork-OR1 (SW-OR1) (He et al., 2025), Magistral (Rastogi et al., 2025) and GPT-OSS (Agarwal et al., 2025), on five challenging reasoning benchmarks spanning mathematics, science and programming: AIME24, AIME25 (MAA Committees, 2025), HMMT Feb 2025 (Balunović et al., 2025), the GPQA Diamond set (Rein et al., 2024) and LiveCodeBench[7] (Jain et al., 2024), consisting of 30, 30, 30, 198, and 175 problems, respectively. For decoding, we adopt each model's recommended temperature and top_p, with a maximum generation length of 32K tokens.

We consider the values of N and M from $\{1, 2, 4, 8, 16, 32, 64\}$. For early pruning, we fix $N = 64$ and select the top M most promising first-step candidates using reward scores from a process reward model (PRM), Qwen2.5-Math-PRM-7B (Zhang et al., 2025). When $M = 64$, the accuracy is exactly the same as the one for $N = 64$, since all candidates are chosen. For PRM, a "step" is defined as a segment of text ending with "\n\n", and each step receives an individual reward score. The trajectory t_1^n contains multiple such steps, and we use the score of its final step as the overall re-

[7] Latest release, containing problems from Jan–Apr 2025.

Model	Method	Maj@K Accuracy (%) ↑					Avg. #Tokens ↓
		AIME24	AIME25	HMMT	GPQA	Avg.	
DS-R1-Qwen-1.5B	$N = 64$	**60.0**	**40.0**	23.3	**41.4**	41.2	$\times 1.00$
	$N = 16$	57.0	37.6	22.4	39.4	39.1 (-2.1)	$\times 0.25$
	$M = 16$	**60.0**	**40.0**	**26.7**	39.4	**41.5** (+0.3)	$\times 0.28$
DS-R1-Qwen-32B	$N = 64$	**86.7**	**73.3**	**56.7**	**64.1**	**70.2**	$\times 1.00$
	$N = 16$	85.4	70.9	52.6	63.9	68.2 (-2.0)	$\times 0.25$
	$M = 16$	**86.7**	**73.3**	**56.7**	63.6	70.1 (-0.1)	$\times 0.29$
Qwen3-8B	$N = 64$	**83.3**	**70.0**	56.7	61.1	67.8	$\times 1.00$
	$N = 16$	79.8	69.0	52.3	61.3	65.6 (-2.2)	$\times 0.25$
	$M = 16$	**83.3**	**70.0**	**60.0**	**62.1**	**68.9** (+1.1)	$\times 0.28$
SW-OR1-7B-Preview	$N = 64$	**80.0**	**60.0**	**40.0**	50.5	57.6	$\times 1.00$
	$N = 16$	75.7	58.0	37.9	51.9	55.9 (-1.7)	$\times 0.25$
	$M = 16$	**80.0**	**60.0**	**40.0**	**52.5**	**57.9** (+0.3)	$\times 0.29$
GPT-OSS-20B	$N = 64$	**86.7**	**83.3**	**80.0**	**73.2**	**80.8**	$\times 1.00$
	$N = 16$	85.3	81.7	73.3	72.7	78.3 (-2.5)	$\times 0.25$
	$M = 16$	**86.7**	**83.3**	**80.0**	**73.2**	**80.8** (-0.0)	$\times 0.27$

Table 2: Early pruning accuracy and efficiency. We select $M = 16$ first steps with the highest reward scores out of $N = 64$ candidate first steps. The number of tokens used for the 48 discarded first steps is also included for the early pruning method. Early pruning maintains the accuracy while only requiring $< 30\%$ original inference budget. Please refer to Table J.1 for Magistral-Small with similar results.

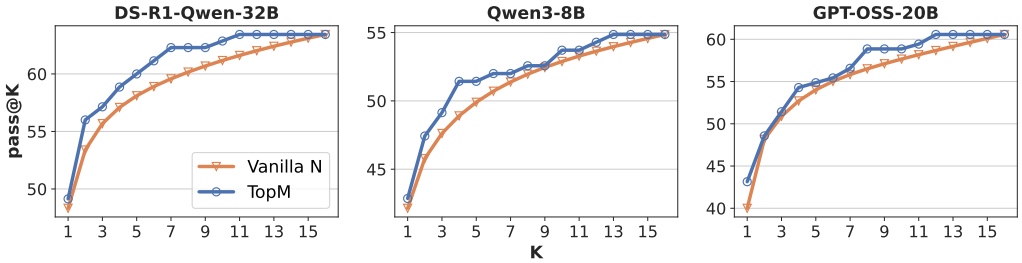

Figure 4: The pass rate on LiveCodeBench, where we set $N = 16$ for early pruning (TopM).

ward, $r(t_1^n)$. Notably, using a PRM to score t_1^n is really cheap, because its computation is similar to generate the embedding for $p \oplus t_1^n$.

Our main focus here is efficiency. To better control the computational budget for generating the first step, we redefine t_1^n in this section to have a fixed token length, $\text{len}(t_1^n)$. The previous definition in §3 made it difficult to control the computational budget for generating the first step, as the trigger phrase might not appear or could occur late in the sequence. An ablation study for these two definitions is in Table 3. By fixing the token length, we achieve precise control over the generation budget when sampling N candidate first steps. By default, $\text{len}(t_1^n) = 512$.

4.3.1 Main Results

In Figure 3, we analyze the performance as M varies. We find that selecting the top 16 first reasoning steps from 64 candidates and continuing generation from them achieves accuracy on par with, or even exceeding, conventional sampling with $N = 64$. This trend is consistent across diverse LLMs and benchmarks. Interestingly, for certain cases, such as AIME24, AIME25, and GPQA on DS-R1-Qwen-32B, using as few as $M \leq 8$ suffices to match the performance of $N = 64$.

Table 2 reports the detailed accuracy and token consumption of different methods. When using $M = 16$, early pruning consistently matches the accuracy of $N = 64$ across a range of LLMs and benchmarks, while substantially outperforming $N = 16$ under a comparable token budget. For Qwen3-8B, early pruning even yields a 1.1% improvement in average accuracy. Importantly, these gains come at less than 30% of the inference cost of majority voting with $N = 64$, underscoring the strong efficiency advantage of our method.

For the code generation benchmark, LiveCodeBench, where majority voting is not applicable, we present the pass rate in Figure 4. Early pruning consistently surpasses standard sampling given the same number of complete sampling.

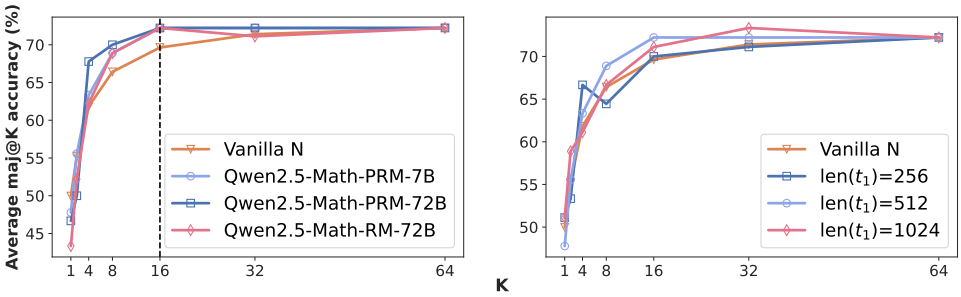

Figure 5: Average accuracy across AIME24, AIME25 and HMMT on DS-R1-Qwen-32B. **Left:** Reward signals from different reward models. The choice of reward model has minimal impact on the performance. **Right:** Effect of varying the length of the first reasoning step. Using a very short first step, like 256 tokens, leads to suboptimal performance, because it provides insufficient reasoning context to effectively evaluate the quality of the step.

Model	len(t_1)=512	Heuristic
DS-R1-Qwen-1.5B	41.5	**43.8**
DS-R1-Qwen-32B	70.1	**70.4**
Qwen3-8B	**68.9**	68.2
SW-OR1-7B-Preview	**57.9**	57.7

Table 3: Average maj@K accuracy for early pruning, with first step defined by length or phrase.

Model	$N = 64$	$M = 16$
DS-R1-Qwen-1.5B	×1.00	×**0.27**
Qwen3-8B	×1.00	×**0.37**

Table 4: Time spent by early pruning.

4.3.2 Ablation Studies

Here we further validate our default settings.

Choice of Reward Model. In Figure 5 (Left), we evaluate two additional reward models: a larger PRM, Qwen2.5-Math-PRM-72B, and an outcome reward model, Qwen2.5-Math-RM-72B (Yang et al., 2024). The results indicate that the choice of reward model has minimal impact on performance. Notably, the smaller 7B PRM achieves comparable results, highlighting the efficiency of our approach.

Length of First Step. In Figure 5 (Right), we examine the impact of varying the length of the first reasoning step. We observe that the shortest first step (i.e., len(t_1^n) = 256) leads to degraded performance. We hypothesize that shorter t_1^n sequences lack sufficient reasoning content, making them less informative for reliable reward model evaluation. Nevertheless, setting len(t_1^n) $\geq$ 512 tokens yields consistently better performance than vanilla sampling. Setting len(t_1)=4 performs at the N=16 random baseline (Appendix K), confirming the signal is step-level rather than token-level.

Effect of First Step Split. Table 3 examines how defining the first step influences the performance of early pruning. The heuristic approach follows the method described in §3. Overall, both definitions yield comparable results. Nevertheless, we recommend using the token-count–based definition, as it provides a clearer way to manage the token budget used by the first steps. Moreover, trigger phrases that signal step boundaries may vary across LLMs.

Overhead from Reward Model. Compared to vanilla sampling, early pruning requires scoring the first step with a reward model, which introduces additional overhead. To ensure a fair comparison, we avoid using extra GPUs for deploying the reward model. Instead, our procedure is as follows: (1) load the reasoning model to generate candidate first steps and then offload it; (2) load the reward model on the same GPU to evaluate these steps and offload it; and (3) reload the reasoning model to continue generation from the selected first steps. The timing results are reported in Table 4. Early pruning remains efficient both in terms of token usage and runtime, since evaluating the first step with the reward model is inexpensive, comparable to computing embeddings for the short first steps.

5 Conclusion

In this paper, we show that long-CoT models are highly sensitive to the beginning of the reasoning chain. The first reasoning step is strongly aligned with the final conclusion, and models often fail to recover when this step is incorrect. We further show that this early signal can support efficient inference: a simple first-step guided pruning strategy matches standard majority voting while reducing inference cost by up to 70%. Together, these results identify the first reasoning step as both a bottleneck for error recovery and a useful signal for efficient inference, motivating future work on understanding and mitigating early-step failures in long-CoT reasoning.

Limitations

Our lost-at-the-beginning observation mainly comes from the long-CoT reasoning models that consume much more tokens than the short-CoT models. In addition, we mainly focus on mathematical, code generation, and scientific reasoning benchmarks across five model families. Although these datasets and models are limited in scope, they are representative of structured multi-step reasoning and provide a controlled setting for evaluating the first-step effects.

Ethics Statement

This work builds on publicly available datasets (AIME24, AIME25, HMMT Feb 2025, GPQA, and LiveCodeBench) and a mix of openly released models (e.g., DeepSeek, Qwen) and closed-source models accessed via public APIs (e.g., Claude-3.7-Sonnet, GPT-5), all used in accordance with their respective licenses and terms of service. We report exact model identifiers, versions (where available), and dataset subsets to facilitate reproducibility. Due to licensing and terms-of-use restrictions, we do not redistribute proprietary model weights, API outputs, or any data that is not permitted for redistribution. We plan to release our code, evaluation scripts, and prompts under a permissive open-source license, subject to these constraints.

We note that results involving proprietary models may be subject to limitations in reproducibility due to restricted access and potential model updates. We further acknowledge that improvements in efficiency may reduce computational cost barriers, which could enable misuse (e.g., large-scale automated disinformation). However, our focus is on the rigorous evaluation of reasoning processes and on improving efficiency for legitimate research purposes. We encourage responsible use of our findings and emphasize that the proposed methods are intended for research and educational contexts, not for harmful applications.

References

Sandhini Agarwal, Lama Ahmad, Jason Ai, Sam Altman, Andy Applebaum, Edwin Arbus, Rahul K Arora, Yu Bai, Bowen Baker, Haiming Bao, and 1 others. 2025. gpt-oss-120b & gpt-oss-20b model card. *arXiv preprint arXiv:2508.10925*.

Pranjal Aggarwal, Aman Madaan, Yiming Yang, and 1 others. 2023. Let's sample step by step: Adaptive-consistency for efficient reasoning and coding with llms. In *Proceedings of the 2023 Conference on Empirical Methods in Natural Language Processing*, pages 12375–12396.

Anthropic. 2025. Claude 3.7 sonnet system card.

Mislav Balunović, Jasper Dekoninck, Ivo Petrov, Nikola Jovanović, and Martin Vechev. 2025. MathArena: Evaluating LLMs on uncontaminated math competitions.

Federico Barbero, Alvaro Arroyo, Xiangming Gu, Christos Perivolaropoulos, Petar Veličković, Razvan Pascanu, and Michael M. Bronstein. 2025. Why do LLMs attend to the first token? In *Second Conference on Language Modeling*.

Tom Brown, Benjamin Mann, Nick Ryder, Melanie Subbiah, Jared D Kaplan, Prafulla Dhariwal, Arvind Neelakantan, Pranav Shyam, Girish Sastry, Amanda Askell, and 1 others. 2020. Language models are few-shot learners. *Advances in neural information processing systems*, 33:1877–1901.

Xingyu Chen, Jiahao Xu, Tian Liang, Zhiwei He, Jianhui Pang, Dian Yu, Linfeng Song, Qiuzhi Liu, Mengfei Zhou, Zhuosheng Zhang, and 1 others. 2024. Do not think that much for 2+ 3=? on the overthinking of o1-like llms. *arXiv preprint arXiv:2412.21187*.

Cheng-Han Chiang and Hung-yi Lee. 2024. Over-reasoning and redundant calculation of large language models. In *Proceedings of the 18th Conference of the European Chapter of the Association for Computational Linguistics (Volume 2: Short Papers)*, pages 161–169, St. Julian's, Malta. Association for Computational Linguistics.

Thibault Formal, Carlos Lassance, Benjamin Piwowarski, and Stéphane Clinchant. 2022. From distillation to hard negative sampling: Making sparse neural ir models more effective. *arXiv preprint*.

Thibault Formal, Benjamin Piwowarski, and Stéphane Clinchant. 2021. Splade: Sparse lexical and expansion model for first stage ranking. In *Proceedings of the 44th International ACM SIGIR Conference on Research and Development in Information Retrieval*, pages 2288–2292.

Yichao Fu, Junda Chen, Yonghao Zhuang, Zheyu Fu, Ion Stoica, and Hao Zhang. 2025a. Reasoning without self-doubt: More efficient chain-of-thought through certainty probing. In *ICLR 2025 Workshop on Foundation Models in the Wild*.

Yichao Fu, Xuewei Wang, Yuandong Tian, and Jiawei Zhao. 2025b. Deep think with confidence. *arXiv preprint arXiv:2508.15260*.

Zeyu Gan, Yun Liao, and Yong Liu. 2025. Rethinking external slow-thinking: From snowball errors to probability of correct reasoning. In *Forty-second International Conference on Machine Learning*.

Daya Guo, Dejian Yang, Haowei Zhang, Junxiao Song, Ruoyu Zhang, Runxin Xu, Qihao Zhu, Shirong Ma, Peiyi Wang, Xiao Bi, and 1 others. 2025. DeepSeek-R1: Incentivizing reasoning capability in LLMs via reinforcement learning. *arXiv preprint arXiv:2501.12948*.

Jujie He, Jiacai Liu, Chris Yuhao Liu, Rui Yan, Chaojie Wang, Peng Cheng, Xiaoyu Zhang, Fuxiang Zhang, Jiacheng Xu, Wei Shen, Siyuan Li, Liang Zeng, Tianwen Wei, Cheng Cheng, Yang Liu, and Yahui Zhou. 2025. Skywork open reasoner series.

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

Aaron Hurst, Adam Lerer, Adam P. Goucher, Adam Perelman, Aditya Ramesh, Aidan Clark, AJ Ostrow, Akila Welihinda, Alan Hayes, Alec Radford, and 1 others. 2024. GPT-4o system card. *arXiv preprint arXiv:2410.21276*.

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

Naman Jain, King Han, Alex Gu, Wen-Ding Li, Fanjia Yan, Tianjun Zhang, Sida Wang, Armando Solar-Lezama, Koushik Sen, and Ion Stoica. 2024. LiveCodeBench: Holistic and contamination free evaluation of large language models for code. *arXiv preprint arXiv:2403.07974*.

Tamera Lanham, Anna Chen, Ansh Radhakrishnan, Benoit Steiner, Carson Denison, Danny Hernandez, Dustin Li, Esin Durmus, Evan Hubinger, Jackson Kernion, and 1 others. 2023. Measuring faithfulness in chain-of-thought reasoning. *arXiv preprint arXiv:2307.13702*.

Yiwei Li, Peiwen Yuan, Shaoxiong Feng, Boyuan Pan, Xinglin Wang, Bin Sun, Heda Wang, and Kan Li. 2024. Escape sky-high cost: Early-stopping self-consistency for multi-step reasoning. In *ICLR*.

Baohao Liao, Hanze Dong, Yuhui Xu, Doyen Sahoo, Christof Monz, Junnan Li, and Caiming Xiong. 2025a. Fractured chain-of-thought reasoning. *arXiv preprint arXiv:2505.12992*.

Baohao Liao, Yuhui Xu, Hanze Dong, Junnan Li, Christof Monz, Silvio Savarese, Doyen Sahoo, and Caiming Xiong. 2025b. Reward-guided speculative decoding for efficient LLM reasoning. *arXiv preprint arXiv:2501.19324*.

Nelson F. Liu, Kevin Lin, John Hewitt, Ashwin Paranjape, Michele Bevilacqua, Fabio Petroni, and Percy Liang. 2024. Lost in the middle: How language models use long contexts. *Transactions of the Association for Computational Linguistics*, 12:157–173.

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

Ruotian Ma, Peisong Wang, Cheng Liu, Xingyan Liu, Jiaqi Chen, Bang Zhang, Xin Zhou, Nan Du, and Jia Li. 2025. S^2R: Teaching LLMs to self-verify and self-correct via reinforcement learning. In *Proceedings of the 63rd Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, pages 22632–22654, Vienna, Austria. Association for Computational Linguistics.

MAA Committees. 2025. AIME Problems and Solutions. https://artofproblemsolving.com/wiki/index.php/AIME_Problems_and_Solutions.

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*.

Abhinav Rastogi, Albert Q Jiang, Andy Lo, Gabrielle Berrada, Guillaume Lample, Jason Rute, Joep Barmentlo, Karmesh Yadav, Kartik Khandelwal, Khyathi Raghavi Chandu, and 1 others. 2025. Magistral. *arXiv preprint arXiv:2506.10910*.

Nils Reimers and Iryna Gurevych. 2019. Sentence-BERT: Sentence embeddings using siamese BERT-networks. In *Proceedings of the 2019 Conference on Empirical Methods in Natural Language Processing*. Association for Computational Linguistics.

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

Gemini Team, Petko Georgiev, Ving Ian Lei, Ryan Burnell, Libin Bai, Anmol Gulati, Garrett Tanzer, Damien Vincent, Zhufeng Pan, Shibo Wang, and 1 others. 2024. Gemini 1.5: Unlocking multimodal understanding across millions of tokens of context. *arXiv preprint arXiv:2403.05530*.

Qwen Team. 2025. Qwen3.

Guangya Wan, Yuqi Wu, Jie Chen, and Sheng Li. 2025. Reasoning aware self-consistency: Leveraging reasoning paths for efficient LLM sampling. In *Proceedings of the 2025 Conference of the Nations of the Americas Chapter of the Association for Computational Linguistics: Human Language Technologies (Volume 1: Long Papers)*, pages 3613–3635, Albuquerque, New Mexico. Association for Computational Linguistics.

Wenhui Wang, Furu Wei, Li Dong, Hangbo Bao, Nan Yang, and Ming Zhou. 2020. MiniLM: Deep self-attention distillation for task-agnostic compression

of pre-trained transformers. *Advances in Neural Information Processing Systems*.

Xuezhi Wang, Jason Wei, Dale Schuurmans, Quoc Le, Ed Chi, Sharan Narang, Aakanksha Chowdhery, and Denny Zhou. 2022. Self-consistency improves chain of thought reasoning in language models. *arXiv preprint arXiv:2203.11171*.

Yue Wang, Qiuzhi Liu, Jiahao Xu, Tian Liang, Xingyu Chen, Zhiwei He, Linfeng Song, Dian Yu, Juntao Li, Zhuosheng Zhang, and 1 others. 2025a. Thoughts are all over the place: On the underthinking of o1-like LLMs. *arXiv preprint arXiv:2501.18585*.

Zezhong Wang, Xingshan Zeng, Weiwen Liu, Yufei Wang, Liangyou Li, Yasheng Wang, Lifeng Shang, Xin Jiang, Qun Liu, and Kam-Fai Wong. 2025b. Chain-of-probe: Examining the necessity and accuracy of CoT step-by-step. In *Findings of the Association for Computational Linguistics: NAACL 2025*, pages 2586–2606. Association for Computational Linguistics.

Liang Wen, Yunke Cai, Fenrui Xiao, Xin He, Qi An, Zhenyu Duan, Yimin Du, Junchen Liu, Lifu Tang, Xiaowei Lv, Haosheng Zou, Yongchao Deng, Shousheng Jia, and Xiangzheng Zhang. 2025. Light-R1: Curriculum SFT, DPO and RL for long COT from scratch and beyond. *arXiv preprint arXiv:2503.10460*.

Guangxuan Xiao, Yuandong Tian, Beidi Chen, Song Han, and Mike Lewis. 2024. Efficient streaming language models with attention sinks. In *The Twelfth International Conference on Learning Representations*.

Wei Xiong, Wenting Zhao, Weizhe Yuan, Olga Golovneva, Tong Zhang, Jason Weston, and Sainbayar Sukhbaatar. 2025. StepWiser: Stepwise generative judges for wiser reasoning. *arXiv preprint arXiv:2508.19229*.

Yuhui Xu, Hanze Dong, Lei Wang, Doyen Sahoo, Junnan Li, and Caiming Xiong. 2025. Scalable chain of thoughts via elastic reasoning. *arXiv preprint arXiv:2505.05315*.

Mingfeng Xue, Dayiheng Liu, Wenqiang Lei, Xingzhang Ren, Baosong Yang, Jun Xie, Yidan Zhang, Dezhong Peng, and Jiancheng Lv. 2023. Dynamic voting for efficient reasoning in large language models. In *Findings of the Association for Computational Linguistics: EMNLP 2023*, pages 3085–3104.

An Yang, Anfeng Li, Baosong Yang, Beichen Zhang, Binyuan Hui, Bo Zheng, Bowen Yu, Chang Gao, Chengen Huang, Chenxu Lv, and 1 others. 2025a. Qwen3 technical report. *arXiv preprint arXiv:2505.09388*.

An Yang, Beichen Zhang, Binyuan Hui, Bofei Gao, Bowen Yu, Chengpeng Li, Dayiheng Liu, Jianhong Tu, Jingren Zhou, Junyang Lin, Keming Lu, Mingfeng Xue, Runji Lin, Tianyu Liu, Xingzhang Ren, and Zhenru Zhang. 2024. Qwen2.5-math technical report: Toward mathematical expert model via self-improvement. *arXiv preprint arXiv:2409.12122*.

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

Weiqiu You, Anton Xue, Shreya Havaldar, Delip Rao, Helen Jin, Chris Callison-Burch, and Eric Wong. 2025. Probabilistic soundness guarantees in llm reasoning chains. In *Proceedings of the 2025 Conference on Empirical Methods in Natural Language Processing*, pages 7517–7536.

Yusen Zhang, Sarkar Snigdha Sarathi Das, and Rui Zhang. 2024. Verbosity $\neq$ veracity: Demystify verbosity compensation behavior of large language models. *arXiv preprint arXiv:2411.07858*.

Zhenru Zhang, Chujie Zheng, Yangzhen Wu, Beichen Zhang, Runji Lin, Bowen Yu, Dayiheng Liu, Jingren Zhou, and Junyang Lin. 2025. The lessons of developing process reward models in mathematical reasoning. *arXiv preprint arXiv:2501.07301*.

Lexin Zhou, Fernando Martínez-Plumed, José Hernández-Orallo, Cèsar Ferri, and Wout Schellaert. 2022. Reject before you run: Small assessors anticipate big language models. *EBeM@ IJCAI*, 3169.

A Reasoning Step Segmentation Prompt

We report the prompt used for automatic reasoning step segmentation with GPT-5, following Xiong et al. (2025).

Prompt for Reasoning Step Segmentation

You are a segmentation engine. Your ONLY task is to insert <chunk> ... </chunk> tags into the given reasoning trace.

CRITICAL INSTRUCTIONS:
- Do not change, reword, delete, or reorder ANY part of the text.
- Do not fix grammar, spelling, or formatting.
- Do not add commentary or explanations.
- The output must be EXACTLY the same as the input reasoning trace, character for character, except that you insert <chunk> and </chunk> tags.

SEGMENTATION PRINCIPLES:
1. Unified purpose: A chunk should serve a single, clear objective. For example: setting up an initial equation, executing a self-contained calculation (like integration by parts), or stating a final/intermediate conclusion. All content within the chunk must directly serve this one core goal.
2. Logical Cohesion: All lines within a chunk must form a continuous and uninterrupted logical flow. A new chunk should begin as soon as the focus or purpose of the reasoning shifts.
3. Clear Transition: A new chunk must begin when the problem-solving process enters a new phase. This includes transitioning from "solving for a variable" to "verifying the answer," or inserting an "explanatory side-note" into the main workflow.

FORMAT RULES:
1. Wrap each segment with <chunk> and </chunk>.
2. Preserve all original text and newlines EXACTLY.
3. Output ONLY the tagged reasoning trace — nothing else.

B Prompt for Question-Restatement Removal

We report the prompt used to remove question-restatement text from the beginning of reasoning traces, as described in §3.1. This preprocessing step is applied using GPT-5 mini to mitigate inflated similarity due to problem restatement.

Prompt for Question-Restatement Removal

You are an overlap marker.

INPUT FORMAT:
Question: {QUESTION_TEXT}
Reasoning: {REASONING_TEXT}

TASK:
- In the REASONING_TEXT, identify any span at the very beginning that repeats or restates the QUESTION_TEXT.
- Wrap that span in <question>...</question>.
- If the span is fewer than 10 words, ignore it (do not mark anything).
- Preserve all original text, characters, spacing, and newlines.
- Output only the REASONING_TEXT (with tags if applied), nothing else.

C Keywords for Reasoning Step Segmentation

Good reasoning switching keywords should signal a clear switch in reasoning, while should not occur too frequently which often indicate only minor digressions, or too rare to be reliable delimiters. We collect a list of reasoning switching keywords from papers (Muennighoff et al., 2025; Hou et al., 2025). To assess which reasoning switch keywords could serve a similar role, we compute their average frequency per question in model outputs in Table C.1. After cosidering the frequencies and manually check the segment results, we found that *"alternatively"* strikes a balance, appearing often enough to capture step boundaries while still reflecting shifts in reasoning, for DS-R1-Qwen-7B and Qwen3-8B. We uses both *"alternatively"* and *"wait"* for Claude-3.7-Sonnet, GPT-OSS-20B and Magistral-Small.

	but	wait	alternatively	hmm	hold on	let me confirm	however
DS-R1-Qwen-7B	82	72	14	7	3	<1	10.1
Qwen3-8B	91	66	20	6	1	<1	10.4
Claude-3.7-Sonnet	28.5	22.6	0.2	6.1	0.7	0.0	0.9
Magistral-Small	47.1	54.8	3.0	0.0	0.9	0.2	1.4
GPT-OSS-20B	171.6	60.5	4.9	1.2	0.7	0.0	5.9

Table C.1: Average frequency per question of discourse markers in model outputs.

D Model Identifiers

For reproducibility, we provide the exact model identifiers used in our experiments. The models are: DeepSeek-R1-Distill-Qwen-7B (`deepseek-ai/DeepSeek-R1-Distill-Qwen-7B`), Qwen3-8B (`Qwen/Qwen3-8B`), Claude 3.7 (`claude-3-7-sonnet-20250219`, accessed via the Claude API), GPT-OSS-20B (`openai/gpt-oss-20b`), and Magistral-Small (`mistralai/Magistral-Small-2509`).

E Semantic Similarity on GPQA

As an auxiliary analysis, we evaluate the semantic similarity between each intermediate reasoning step and the final conclusion on an additional dataset, GPQA Diamond set (Rein et al., 2024), following the procedure described in §3.1. Due to budget constraints, experiments are limited to open-source models, and we do not use GPT-5 mini to remove question-overlap text at the beginning of reasoning steps.

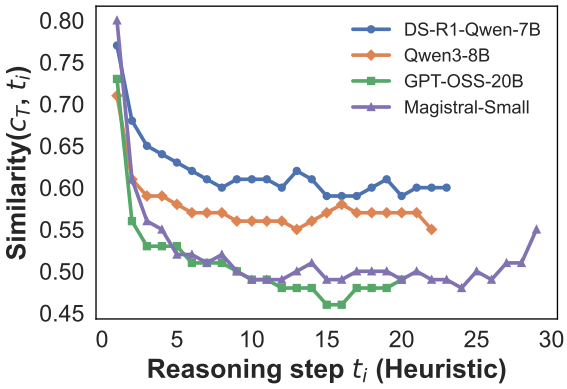

Figure E.1: Cosine similarity between the embeddings of the i-th reasoning step t_i and the final conclusion c_T on GPQA, using interpolation to the average number of reasoning steps. Reasoning steps are segmented using heuristic rules.

The results are shown in Figure E.1. Consistent with the observations in §3.1, the first reasoning step exhibits the highest semantic similarity to the final conclusion, after which similarity drops sharply in subsequent steps. This trend is consistent across the evaluated models and provides additional supporting evidence for the main analysis.

F SPLADE Similarity Analysis

We report the cosine similarity between the embedding of the i-th reasoning step t_i and the final conclusion c_T, interpolated using the maximum number of reasoning steps (Figure F.1).

In addition, we report SPLADE similarity (Formal et al., 2021), which provides a sparse lexical–semantic relevance signal by expanding tokens into weighted lexical features. We compute SPLADE similarity using the `naver/splade-cocondenser-ensembledistil` model (Formal et al., 2022), following standard practice. As shown in Figure F.2, the trends closely mirror those observed under cosine similarity. These results serve as complementary checks, confirming that our findings are not tied to dense embedding models and hold across both dense and sparse similarity measures.

G Correlation between first reasoning step and final prediction

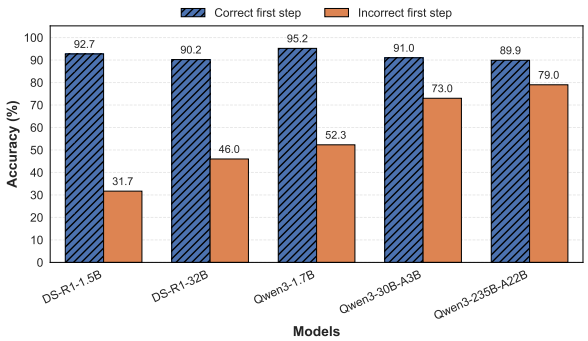

Figure G.2: More results from Qwen and DeepSeek models of different sizes. We report the relationship between the accuracy of the final prediction (q_T) and the correctness of the first reasoning step (q_1). If q_1 is incorrect, q_T is more likely to be incorrect.

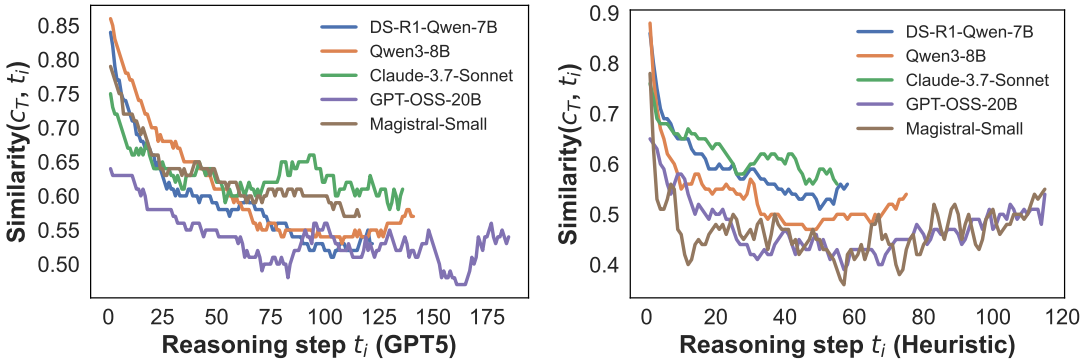

Figure F.1: Cosine similarity between the embeddings of the i-th reasoning step t_i and the final conclusion c_T, using the maximum number of reasoning steps for interpolation. Reasoning steps are segmented either by GPT-5 (left) or by heuristic rules (right).

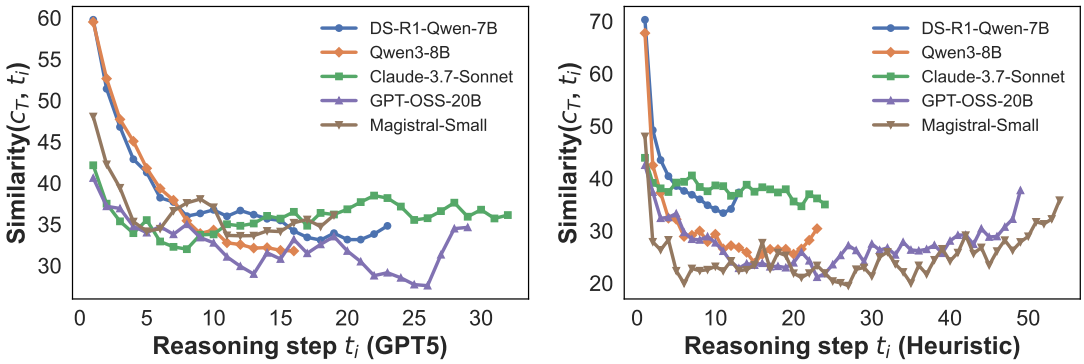

Figure F.2: SPLADE similarity between the representations of the i-th reasoning step t_i and the final conclusion c_T, using the average number of reasoning steps for interpolation. Reasoning steps are segmented either by GPT-5 (left) or by heuristic rules (right).

In the main text, we reported results on DS-R1-Qwen-7B. Here we extend the analysis to Qwen3-8B (Figure G.1) and further include DeepSeek and Qwen3 models of different sizes (Figure G.2). These additional results verify that the correlation between the first reasoning step and the final prediction persists as model size scales.

H Prompt Examples

The template for using the DeepSeek-R1 model is provided below. We apply the same structure to other open-source thinking models, with slight adjustments for model-specific input and output formats.

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

I Example of Incorrect First Step

Figure I.1 illustrates a case where the initial reasoning step is incorrect.

J Examples for Minor Perturbation

Original Final Conclusion

To solve the problem, we are given triangle ABC with side lengths $AB = 5$, $BC = 9$, and $AC = 10$.
[...]
Step 4: Compute AP. Since P lies on line segment AD, and D lies outside the circle, the point P lies between A and D. Therefore:

$$AP = DA - DP = \frac{325}{22} - \frac{2025}{286}.$$

Find a common denominator:

$$\frac{325}{22} = \frac{4225}{286} \Rightarrow$$
$$AP = \frac{4225 - 2025}{286} = \frac{2200}{286} = \frac{100}{13}.$$

Final Step: Compute $m + n$.
The reduced form of AP is $\frac{100}{13}$, so:

$$m = 100, \quad n = 13 \Rightarrow m + n = \text{\boxed{113}}.$$

Final Answer: \boxed{113}

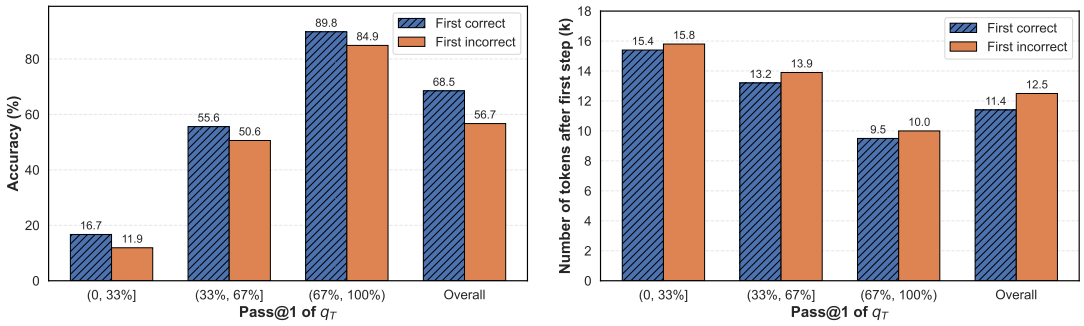

Figure G.1: Accuracy and token usage on Qwen3-8B. **Left:** Relationship between the correctness of the final prediction (q_T) and that based solely on the first reasoning step (q_1) across different difficulty levels. When q_1 is incorrect, q_T is significantly more likely to be incorrect. **Right:** Number of tokens used after the first reasoning step t_1, i.e., for subsequent steps $[t_2, \ldots, t_T]$. Even when q_1 is correct, the model often consumes a large number of additional tokens, indicating overthinking behavior.

New Correct First Reason

To solve the problem, we are given triangle ABC with side lengths $AB = 5$, $BC = 9$, and $AC = 10$.
[...]
Step 4: Compute AP. Since P lies on line segment AD, and D lies outside the circle, the point P lies between A and D. Therefore:

$$AP = DA - DP = \frac{325}{22} - \frac{2025}{286}.$$

Find a common denominator:

$$\frac{325}{22} = \frac{4225}{286} \Rightarrow$$
$$AP = \frac{4225 - 2025}{286} = \frac{2200}{286} = \frac{100}{13}.$$

Final Step: Compute $m + n$.
The reduced form of AP is $\frac{100}{13}$, so:

$$m = 100, \quad n = 13 \Rightarrow m + n = 113.$$

New Incorrect First Reason

To solve the problem, we are given triangle ABC with side lengths $AB = 5$, $BC = 9$, and $AC = 10$.
[...]
Step 4: Compute AP. Since P lies on line segment AD, and D lies outside the circle, the point P lies between A and D. Therefore:

$$AP = DA - DP = \frac{325}{22} - \frac{2025}{286}.$$

Find a common denominator:

$$\frac{325}{22} = \frac{4225}{286} \Rightarrow$$
$$AP = \frac{4225 - 2025}{286} = \frac{2200}{286} = \frac{100}{13}.$$

Final Step: Compute $m + n$.
The reduced form of AP is $\frac{100}{13}$, so:

$$m = 100, \quad n = 13 \Rightarrow m + n = 103.$$

K Token-level vs. step-level early pruning

To verify that the early-pruning signal in §4 operates at the reasoning-step level rather than at the first-token level, we evaluate first-step pruning at an extreme lower bound of length: len(t_1)=4 tokens. This length is comparable in scale to the attention-sink phenomenon studied in transformer LLMs more broadly (Xiao et al., 2024; Barbero et al., 2025), where the first few tokens of a sequence receive disproportionate attention regardless of semantic content.

Table K.1 reports the results on Qwen3-8B averaged over AIME24, AIME25, HMMT, and GPQA. Pruning on the first 4 tokens performs at the N=16 random-sampling baseline (65.3 vs. 65.6), while

Model	Method	Maj@K accuracy (%) ↑					Avg. #Tokens ↓
		AIME24	AIME25	HMMT	GPQA	Avg.	
Magistral-Small	$N = 64$	86.7	**83.3**	**76.7**	**70.2**	**79.2**	$\times 1.00$
	$N = 16$	**87.1**	82.6	71.1	69.2	77.5 (-1.7)	$\times 0.25$
	$M = 16$	86.7	**83.3**	73.3	**70.2**	78.4 (-0.8)	$\times 0.28$

Table J.1: Early pruning accuracy and efficiency. We select $M = 16$ first steps with the highest reward scores out of $N = 64$ candidate first steps. The number of tokens used for the 48 discarded first steps is also included for the early pruning method. Early pruning maintains the accuracy while only requiring $< 30\%$ original inference budget.

pruning on the first reasoning step (∼512 tokens) matches or exceeds the accuracy of N=64 majority voting (68.9 vs. 67.8). The early-pruning signal we exploit is therefore carried by reasoning-step-scale content — a complete planning unit of several hundred tokens — and not by the first few tokens.

Method	Avg. Accuracy
N=64 (vanilla)	67.8
N=16 (random)	65.6 (-2.2)
M=16, len(t_1)=4	65.3 (-2.5)
M=16, len(t_1)=512	**68.9 (+1.1)**

Table K.1: Token-level vs. step-level early pruning on Qwen3-8B, averaged over AIME24, AIME25, HMMT, and GPQA. Pruning on only the first 4 tokens (comparable to the attention-sink scale) performs at the N=16 random baseline, while pruning on the first reasoning step (∼512 tokens) matches the accuracy of N=64 majority voting.

Example of Incorrect First Step

Question: Four unit squares form a 2×2 grid. Each of the 12 unit line segments forming the sides of the squares is colored either red or blue in such a way that each unit square has 2 red sides and 2 blue sides. One example is shown below (red is solid, blue is dashed). Find the number of such colorings.
[asy] size(4cm); defaultpen(linewidth(1.2)); draw((0, 0) – (2, 0) – (2, 1)); draw((0, 1) – (1, 1) – (1, 2) – (2,2)); draw((0, 0) – (0, 1), dotted); draw((1, 0) – (1, 1) – (2, 1) – (2, 2), dotted); draw((0, 1) – (0, 2) – (1, 2), dotted); [/asy]
First reason: Alright, so I have this problem about coloring the sides of a 2x2 grid of squares. Each square has four sides, and each of these sides is colored either red or blue. The condition is that each individual square must have exactly two red sides and two blue sides. The question is asking for the number of such colorings.
First, let me visualize the grid. It's a 2x2 grid, so there are four squares arranged in a larger square. Each square has four sides, and each side is shared with another square except for the outermost sides. But in the problem, it says there are 12 unit line segments forming the sides of the squares. Wait, let me check that: a 2x2 grid has 3 horizontal lines and 3 vertical lines, each of length 2 units. So, each horizontal line is divided into two segments, right? So, the total number of horizontal segments is 3 rows times 2 segments per row, which is 6, and similarly, the vertical segments are also 3 columns times 2 segments per column, which is another 6. So, 6 + 6 = 12 segments in total. Okay, that makes sense.
Each of these 12 segments can be colored either red or blue. However, the constraint is that each of the four squares must have exactly two red sides and two blue sides. So, even though each segment can be colored independently, we have these local constraints for each square.
I think a good way to approach this problem is by considering each square and the possible colorings for its sides, then seeing how they can fit together without violating the constraints. However, since the coloring of one square's side affects the adjacent squares, this could get complicated.
Maybe I can model this as a graph where each segment is a node, and the squares impose constraints on the nodes (i.e., exactly two red and two blue edges around each square). But I'm not too familiar with graph theory, so maybe that's not the best approach.
Mistakes in first reasoning step: Mistake 1 incorrectly assumes that in a 2×2 grid, each corner square has two shared and two unshared sides, without properly accounting for edge distribution. Mistake 2 introduces unnecessary abstraction by treating segments as "nodes," which obscures the natural structure: the segments themselves are the edges whose colours we choose, and each square simply places a rule on its four incident edges. Working directly with those four constraints is far cleaner than introducing an extra level of abstraction.

Figure I.1: Example of incorrect first step.