Mitigating Overthinking in Large Reasoning Language Models via Reasoning Path Deviation Monitoring

Weixin Guan[1,2], **Liang Li**[1*], **Jiapeng Liu**[1,2], **Bing Li**[1], **Peng Fu**[1]
Chengyang Fang[3], **Xiaoshuai Hao**[4†], **Can Ma**[1], **Weiping Wang**[1]
[1]Institute of Information Engineering, Chinese Academy of Sciences, Beijing, China
[2]School of Cyberspace Security, University of Chinese Academy of Sciences, Beijing, China
[3]School of Computer and Artificial Intelligence,
Jiangxi University of Finance and Economics, Jiangxi, China
[4]Xiaomi EV
liliang@iie.ac.cn

Abstract

Large Reasoning Language Models (LRLMs) leverage long Chain-of-Thought (CoT) reasoning to solve complex tasks, yet often engage in overthinking. Specifically, they continue to generate reasoning steps that provide meaningless contribution to or even degrade the final answer, even after sufficient reasoning has been produced. Early-exit strategies are proposed to dynamically terminate meaningless or even harmful reasoning, aiming to improve both efficiency and accuracy. However, existing methods either suffer from over-truncation that degrades accuracy below vanilla CoT, or yield only spurious efficiency gains, which reduce token counts but increase wall-clock inference time. We observe that when an LRLM's reasoning path deviates into meaningless or even harmful wandering, this deviation is accompanied by an anomalous surge of high-entropy transition tokens. Building on this insight, we propose RPDI-EE, a training-free early-exit method that monitors the Reasoning Path Deviation Index (RPDI), defined as the ratio of local to global average entropy, which serves as a proxy for reasoning path deviation. By tracking this index, RPDI-EE detects and terminates likely meaningless or even harmful reasoning while preserving productive inference. Experiments across multiple benchmarks and four LRLMs of varying types and scales show that RPDI-EE achieves the largest average accuracy improvement over vanilla CoT among all tested early-exit methods, while mitigating the spurious efficiency gains of existing alternatives.

1 Introduction

Large Reasoning Language Models (LRLMs) [13, 32, 44] demonstrate impressive capabilities on complex tasks by leveraging long Chain-of-Thought (CoT) reasoning. However, extended chains introduce a critical weakness. LRLMs often engage in *overthinking* [3, 30], continuing to generate reasoning steps that provide meaningless contribution to or even degrade the final answer, even after sufficient information for a correct solution has already been produced [9]. Recent studies further confirm that longer reasoning does not necessarily improve performance and can, counter-intuitively,

*Corresponding author.
†Project leader.

40th Conference on Neural Information Processing Systems (NeurIPS 2026).

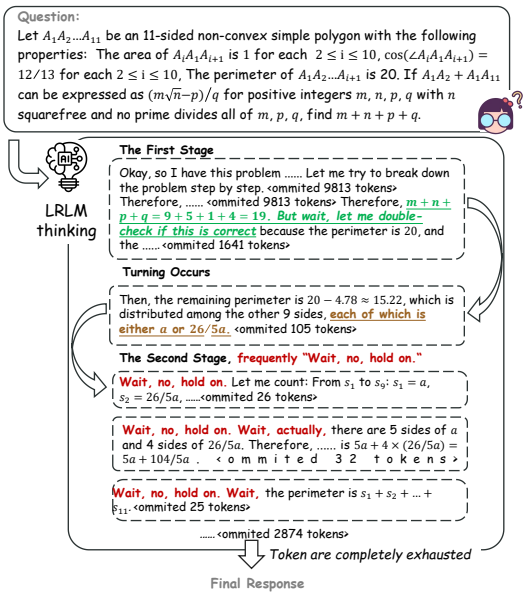


Figure 1: Reasoning trajectory dissection revealing the reasoning deviation trap. The model initially achieves the correct result but fails due to a flawed verification loop. The frequent emergence of transition tokens (*e.g.*, “Wait," “But") serves as a key indicator of reasoning deviation.

degrade accuracy [15, 8], underscoring the practical need to detect and terminate meaningless or even harmful reasoning in a timely manner.

To address this, early-exit strategies are proposed to dynamically terminate the reasoning process and switch to answer generation, aiming to improve both efficiency and reasoning quality. Existing approaches can be broadly grouped into fixed token-budget methods [29, 26, 22], proxy-model-based methods [46, 18, 49, 1], and answer-probing methods [45, 48, 12]. Despite their diversity, these approaches collectively face two key limitations. First, many exhibit *over-truncation*. They frequently make premature stopping decisions that terminate the model before it can self-correct, leading to accuracy *below* vanilla CoT in some scenarios (e.g., 29, 49, 12). Second, obtaining stopping signals introduces additional overhead, such as proxy-model inference, tentative-answer generation, and context switching. This overhead can offset the savings from shorter reasoning, so fewer generated tokens do not necessarily translate into lower end-to-end latency. These limitations raise the possibility that relying on signals external to the reasoning trajectory, whether probed answer or proxy-model, may not always be sufficient. We suspect such signals may not reliably track whether the model is truly progressing toward a correct solution, while obtaining them incurs additional computational cost. It may be necessary to **look inward** at the reasoning trajectory itself rather than solely **looking forward** at tentative answers or **looking outward** at proxy models.

Building on this ‘inward’ perspective, this paper explores the use of latent trajectory signals to mitigate overthinking by monitoring reasoning path deviation, where the model wanders among alternative reasoning path without making progress, leading to error accumulation. Information entropy serves as a dynamic indicator of the model’s internal uncertainty during generation. Moreover, existing studies suggest that high-entropy tokens within a reasoning trajectory often manifest as transition terms, such as ‘wait’, ‘alternatively’, or ‘but’ [39]. Intuitively, a high frequency of these transition tokens indicates that the model is wandering among alternative approaches, struggling to deepen a single reasoning path. Figure 1 illustrates this behavior in an LRLM solving a geometry problem. Although the model initially finds the correct answer, it makes a small calculation error during verification. This error triggers a logical contradiction with the problem’s conditions, leaving the model “stuck” in meaningless or even harmful wandering. This state is clearly marked by the repetitive use of transition tokens like “Wait”, showing that the model cannot deepen the reasoning path. Based on this, we hypothesize that an anomalous frequency of these high-entropy tokens can serve as an internal signal that the reasoning path has deviated.

While directly counting high-entropy tokens is intuitive, such a "hard count" approach relies on specific entropy thresholds that fail to generalize across different models and task difficulties. Through an extensive visualization of reasoning trajectories (Section 3), we observe that token entropy follows a long-tail distribution: a small number of transition tokens contribute most of the average entropy, making average entropy a natural soft proxy for their density. Inspired by this insight, we propose RPDI-EE, an early-exit method based on the Reasoning Path Deviation Index (RPDI), defined as the ratio of local to global average entropy. This index quantifies anomalous spikes in transition token frequency to flag likely meaningless or even harmful wandering. When the RPDI exceeds a predefined threshold λ, RPDI-EE triggers an early exit, suppressing likely meaningless or even harmful inference before it degrades performance. The detailed formulation is presented in Section 4.

Our contributions are summarized as follows:

- We present an empirical finding that reasoning path deviation in LRLMs is accompanied by an anomalous surge of high-entropy transition tokens, providing a new inward signal for detecting meaningless or even harmful reasoning.
- Based on this finding, we propose RPDI-EE, a training-free early-exit method that monitors the Reasoning Path Deviation Index (RPDI) to detect reasoning path deviation via entropy dynamics, rather than relying on proxy models or probed answers, thereby incurring no additional training or context-switching overhead.
- Extensive experiments across multiple benchmarks and four LRLMs of varying types and scales demonstrate that RPDI-EE achieves the largest average accuracy improvement over vanilla CoT among all tested early-exit methods, while achieving lower average end-to-end latency than the other early-exit methods in our runtime comparisons.

2 Related works

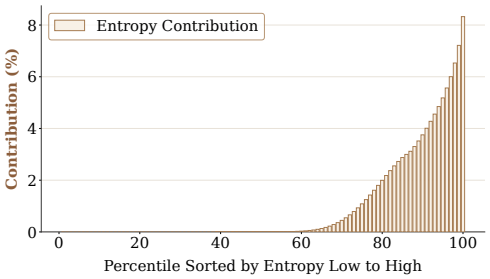
Existing strategies for reducing redundant reasoning include training-time methods such as distillation into concise chains [11, 35], RL with length penalties [31, 20, 24], and latent reasoning compression [47, 25], as well as inference-time methods such as multi-path reasoning [7, 40, 37], difficulty-based activation [10, 19, 4], prompt-based length control [42, 14, 2, 34, 21], and reasoning path compression [43, 50]. However, these approaches either permanently encode efficiency biases into model parameters or fail to precisely control when reasoning should stop. A comprehensive discussion is provided in Appendix C.

Among these, **early-exit strategies** directly address the problem of when to terminate reasoning and are most relevant to our work. Fixed token-budget methods [29, 22, 26] impose length constraints but lack task-level adaptivity. Proxy-model-based methods [18, 46, 49, 1] train auxiliary detectors to monitor reasoning progress, while answer-probing methods [45, 48, 12] interrupt the reasoning stream to generate and evaluate tentative answers. Our method differs from all three categories by using an *internal* entropy signal that requires no additional models, no answer probing, and no interruption of the generation process. A feature-level comparison is provided in Appendix A.3.

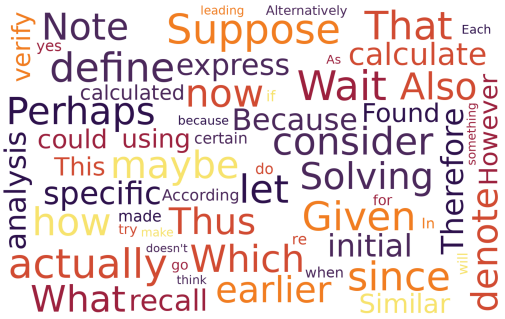
3 Preliminary experiment

Before establishing our methodology, we conduct a preliminary analysis to characterize token entropy across various reasoning trajectories. The goal is to investigate how the token entropy fluctuates as the model navigates complex logical steps and to identify statistical patterns that might govern these transitions. Following previous work [39], we utilize the vanilla CoT to conduct statistical analysis on the 45.9 million tokens generated by the DeepSeek-R1-Distill-Qwen series (7B, 14B, and 32B) models across seven datasets. To ensure consistency, all experimental configurations are aligned to the vanilla CoT settings used in our main experiments.

As illustrated in Figure 2a, we observe that the distribution of entropy contributions exhibits a striking long tail. The vast majority of tokens possess negligible entropy, while a few transition tokens contribute most to the average entropy. Specifically, the first 60% of tokens (sorted by entropy) contribute almost zero to the total entropy sum. This implies that during the reasoning process, the model maintains extremely low entropy for most tokens, while a few high-entropy tokens predominantly drive the overall average entropy. Furthermore, we visualize the 20% of tokens that



(a) Token entropy contribution distribution. Tokens are sorted by entropy and divided into 100 percentile bins. The y-axis shows each bin's percentage contribution to the total entropy.



(b) Visualization of high-frequency tokens among the top 20% entropy contributors. Only tokens that constitute complete English words are retained.

Figure 2: Preliminary analysis of token entropy across 45.9 million tokens generated by the DeepSeek-R1-Distill-Qwen series (7B, 14B, 32B).

contributed most to the average entropy, and the results are displayed in Figure 2b. It can be seen that the high-entropy tokens are mainly transition tokens.

In summary, we draw the following empirical conclusion. Since the average entropy is mainly determined by these few high-entropy tokens, it can serve as a soft counting proxy for the density of transition tokens in a sequence. We verify that this long-tail pattern generalizes across architectures in Appendix D.5.

4 Methodology

Section 3 shows that average entropy is predominantly driven by a small number of high-entropy transition tokens and therefore serves as a natural soft proxy for their local density. Building on this insight, we propose a dynamic early-exit method that "looks inward" at the reasoning trajectory to make better-informed truncation decisions in real time. As illustrated in Figure 3, the framework consists of three components: Real-time Trajectory Entropy Tracking, Path Deviation Index Construction, and Dynamic Early-Exit. The complete algorithm is provided in Appendix A.

4.1 Real-time trajectory entropy tracking

This component computes the Shannon entropy of the model's output distribution at each generation step, producing a real-time entropy sequence that captures the model's token-level uncertainty throughout the reasoning process.

Token-level entropy extraction To quantify the model's uncertainty at each token generation step, we extract the complete probability distribution from the output layer and compute its Shannon entropy. Specifically, at the i-th generation step, the model produces a probability distribution p_i for the next token t_i, conditioned on the initial prompt P and the previously generated reasoning content R. The entropy of this distribution is defined as:

$$H(t_i) = -\sum_{v \in \mathcal{V}} p_i(v) \log p_i(v), \tag{1}$$

where $\mathcal{V}$ denotes the model's vocabulary. The resulting entropy values $H(t_i)$ are recorded sequentially to form an entropy sequence $\mathcal{H}$. The resulting entropy sequence $\mathcal{H}$ provides the basis for detecting anomalous concentrations of high-entropy tokens.

Incremental entropy accumulation To efficiently maintain both local and global entropy statistics, we design an incremental accumulation mechanism that updates running sums in $O(1)$ per token using a sliding window, avoiding the $O(n)$ cost of re-summation. The detailed formulation is provided in Appendix A.2.

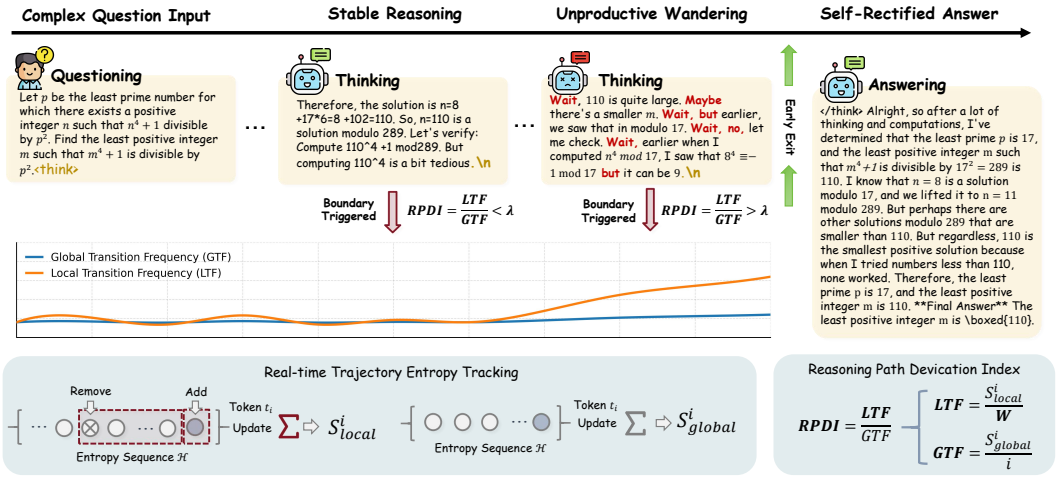


Figure 3: **Overview of RPDI-EE.** RPDI-EE performs continuous entropy scanning during CoT generation. For each new token, it measures local uncertainty via the Local Transition Frequency (LTF), defined as the average token entropy in a sliding window, and compares it to the global reasoning stability reflected by the Global Transition Frequency (GTF). Their ratio forms the Reasoning Path Deviation Index (RPDI), which is checked at boundary tokens marking semantic step transitions. When RPDI surpasses a preset threshold, the model is prompted to stop reasoning and proceed with answer generation.

4.2 Reasoning path deviation index

We introduce the Reasoning Path Deviation Index (RPDI) to transform the entropy sequence into a stable, interpretable indicator signal. This enables early-exit decisions that adapt flexibly to varying model capabilities and task difficulties. RPDI is computed as the ratio of the Local Transition Frequency (LTF) to the Global Transition Frequency (GTF).

Local transition frequency The Local Transition Frequency (LTF) is measured as the average token entropy within the sliding window, characterizing the density of transition tokens in the model's most recently generated content:

$$\mathrm{LTF}_i = \frac{\mathcal{S}^i_{\mathrm{local}}}{W}. \tag{2}$$

Here $\mathcal{S}^i_{\mathrm{local}} = \sum_{j=\max(1,i-W+1)}^{i} H(t_j)$ is the sum of token entropies in the current window, and W is its length. RPDI is checked at semantic boundaries once the window is full ($i \geq W$). A significant increase in LTF typically indicates that the model has generated a large number of transition tokens within a short period. This frequent switching of reasoning paths is closely associated with reasoning path deviation.

Global transition frequency However, a fixed LTF threshold lacks generalizability across different scenarios because the baseline frequency of transition tokens depends on a model's entropy distribution and the task's complexity. To solve this problem, we further introduce the Global Transition Frequency (GTF) to provide an adaptive reference baseline. GTF is defined as the average entropy of all generated tokens from the initiation of reasoning to the current point. It quantifies the overall frequency of transition token occurrences, providing an adaptive reference baseline for detecting relative abnormal increases in this frequency:

$$\mathrm{GTF}_i = \frac{\mathcal{S}^i_{\mathrm{global}}}{i}. \tag{3}$$

Here $\mathcal{S}^i_{\mathrm{global}} = \sum_{j=1}^{i} H(t_j)$ is the sum of entropies of all i reasoning tokens generated so far.

In the i-th reasoning step, the path deviation index standardizes the measurement of the abnormal increase in local uncertainty by calculating the ratio of LTF_i to GTF_i:

$$\mathrm{RPDI}_i = \frac{\mathrm{LTF}_i}{\mathrm{GTF}_i}. \tag{4}$$

When the model advances steadily along the correct reasoning path, LTF_i and GTF_i are roughly comparable, and RPDI_i fluctuates around 1. However, when the model deviates into meaningless or even harmful reasoning and frequently produces high-entropy content locally, LTF_i becomes significantly higher than GTF_i, causing RPDI_i to increase sharply. Therefore, RPDI_i, as a dimensionless metric, serves as a useful indicator of whether the reasoning path has likely deviated.

4.3 Dynamic early-exit

During the LRLM reasoning process, when RPDI-EE detects that the RPDI score exceeds the threshold, it triggers an early exit, prompting the model to switch from thinking mode to answering mode. To achieve this within a prescribed reasoning token budget L_{max}, RPDI-EE inserts an explicit mode termination marker T (e.g., `</think>`) into the current trajectory R to prompt the LRLM to conclude its reasoning and generate the final answer A. Subsequently, the LRLM generates the final answer A based on $P \oplus R$ within the remaining token budget ($L_{\text{max}} - i$). This process can be formalized as:

$$A = \mathrm{LLM}([P \oplus R \oplus T]), \quad \text{for } \mathrm{RPDI}_i > \lambda, \tag{5}$$

where $\oplus$ denotes the concatenation operation, λ is the early exit threshold, which is a pre-defined hyperparameter.

In practice, to reduce computational overhead, RPDI-EE evaluates the RPDI only when the current token belongs to a predefined set of boundary symbols $\mathcal{B}$ that mark semantic step completions, following Yang et al. [45].

It is worth noting that RPDI-EE differs from probing-based methods such as DEER not merely in the choice of stopping signal, but in the monitoring paradigm itself. Probing-based methods adopt a *forward-looking* approach. They interrupt the reasoning stream, generate tentative answers, and evaluate answer consistency to decide whether to continue, incurring additional forward passes and context-switching overhead. In contrast, RPDI-EE adopts an *inward-looking* approach that monitors entropy signals directly within the native autoregressive generation process, requiring no additional forward passes and no interruption of the reasoning flow. This design makes RPDI-EE naturally compatible with continuous-batching inference frameworks such as vLLM.

5 Experimental setup

Model selection and architectures To evaluate the efficacy and generalizability of RPDI-EE across varying model scales and training paradigms, we conduct experiments on four open-source large reasoning language models: DeepSeek-R1-Distill-Qwen-7B/14B/32B [13] (distillation-based) and Qwen3-30B-A3B-Thinking-2507 [44] (RL-trained). Please refer to Appendix B for more details.

Benchmarks and evaluation Our evaluation benchmarks cover a wide range of mathematical and scientific reasoning tasks, categorized by their complexity and domain. For foundational and competitive mathematics we utilize GSM8K [5], AMC23 [27], and the MATH500 [17] datasets, denoted as **MATH EASY**. To evaluate the models on elite-level problem-solving, we include the AIME2024, AIME2025 [28], and OlympiadBench [16], denoted as **MATH HARD**. Finally, we assess high-level scientific reasoning using GPQA-Diamond [33] (marked as **SCIENTIFIC**), a benchmark comprising graduate-level questions verified by subject-matter experts in physics, biology, and chemistry. This multi-level benchmark suite provides a rigorous testing ground for RPDI-EE's capabilities. For evaluation, we align with DEER and employ accuracy (Acc.) and generated token length (Len.) as metrics. We report 95% nonparametric bootstrap confidence intervals using 20,000 resamples, independently resampling examples within each benchmark before aggregation (Appendix D.8). These intervals quantify finite-test-sample uncertainty under fixed decoding settings.

Baselines To establish a clear performance gain, we compare RPDI-EE against five representative baseline strategies. The first is Vanilla CoT, which represents standard autoregressive generation. To evaluate early-exit performance across different paradigms, we include NoThinking [26] and ThinkLess [22] as representatives of fixed token-budget methods. For comparison with dynamic early-exit frameworks, we choose DEER [45] and Dynasor-CoT [12], both of which use answer probing and context switching to determine termination. We evaluate all baselines on all four

Table 1: Performance comparison. Subscripts in all Acc. columns denote the accuracy change relative to the Vanilla CoT. All methods are evaluated on all four LRLMs.

Method	Math Easy		Math Hard		Scientific		Average	
	Acc.↑	Len.↓	Acc.↑	Len.↓	Acc.↑	Len.↓	Acc.↑	Len.↓
DeepSeek-R1-Distill-Qwen-7B	87.9	3801	32.5	10395	32.3	10319	56.2	7558
+ NoThinking	77.6$_{-10.3}$	912	25.2$_{-7.3}$	3277	31.3$_{-1.0}$	841	48.5$_{-7.7}$	1915
+ ThinkLess	76.8$_{-11.1}$	758	24.7$_{-7.8}$	3736	31.8$_{-0.5}$	1175	48.0$_{-8.2}$	2094
+ Dynasor-CoT	73.8$_{-14.1}$	1673	32.4$_{-0.1}$	3366	**45.5**$_{+13.2}$	1876	52.0$_{-4.2}$	2428
+ DEER	85.4$_{-2.5}$	2733	45.0$_{+12.5}$	9005	28.3$_{-4.0}$	9433	59.9$_{+3.7}$	6378
+ **RPDI-EE**	**88.0**$_{+0.1}$	3825	**47.9**$_{+15.4}$	9700	24.8$_{-7.5}$	9727	**61.8**$_{+5.6}$	7186
DeepSeek-R1-Distill-Qwen-14B	90.0	3549	53.2	9464	50.5	6963	68.6	6572
+ NoThinking	80.5$_{-9.5}$	1187	31.1$_{-22.1}$	5641	44.4$_{-6.1}$	1309	54.2$_{-14.4}$	3113
+ ThinkLess	63.8$_{-26.2}$	1394	27.7$_{-25.5}$	5937	34.9$_{-15.6}$	1748	44.2$_{-24.4}$	3391
+ Dynasor-CoT	76.4$_{-13.6}$	1535	30.8$_{-22.4}$	2985	48.5$_{-2.0}$	1686	52.9$_{-15.7}$	2178
+ DEER	90.9$_{+0.9}$	2665	53.2	9159	52.0$_{+1.5}$	6575	69.2$_{+0.6}$	6007
+ **RPDI-EE**	**93.5**$_{+3.5}$	3426	**59.2**$_{+6.0}$	9282	**56.1**$_{+5.6}$	6575	**73.4**$_{+4.8}$	6386
DeepSeek-R1-Distill-Qwen-32B	88.7	3753	50.4	9685	56.1	7386	67.6	6814
+ NoThinking	82.1$_{-6.6}$	797	40.3$_{-10.1}$	6531	51.0$_{-5.1}$	2249	59.7$_{-7.9}$	3462
+ ThinkLess	77.1$_{-11.6}$	485	29.2$_{-21.2}$	2796	53.5$_{-2.6}$	4365	53.2$_{-14.4}$	2030
+ Dynasor-CoT	75.1$_{-13.6}$	1390	34.0$_{-16.4}$	2392	57.6$_{+1.5}$	1655	55.0$_{-12.6}$	1857
+ DEER	90.9$_{+2.2}$	2757	**61.0**$_{+10.6}$	8637	52.0$_{-4.1}$	7134	72.5$_{+4.9}$	5902
+ **RPDI-EE**	**93.5**$_{+4.8}$	3210	**61.0**$_{+10.6}$	9119	**64.7**$_{+8.6}$	6972	**75.5**$_{+7.9}$	6280
Qwen3-30B-A3B-Thinking-2507	96.7	5178	82.1	15621	74.8	7394	87.3	9970
+ NoThinking	92.2$_{-4.5}$	2820	61.7$_{-20.4}$	6923	71.2$_{-3.6}$	6267	76.2$_{-11.1}$	5071
+ ThinkLess	96.9$_{+0.2}$	4409	81.8$_{-0.3}$	14418	74.8	6841	87.2$_{-0.1}$	9046
+ Dynasor-CoT	74.6$_{-22.1}$	1504	28.6$_{-53.5}$	2525	59.1$_{-15.7}$	1735	52.7$_{-34.6}$	1975
+ DEER	97.2$_{+0.5}$	4477	80.5$_{-1.6}$	15285	74.8	7167	86.8$_{-0.5}$	9493
+ **RPDI-EE**	**97.4**$_{+0.7}$	5039	**84.6**$_{+2.5}$	14927	**77.8**$_{+3.0}$	7393	**89.1**$_{+1.8}$	9613

models. Additional comparisons with RCPD, NoWait, and NEAT on DeepSeek-R1-Distill-Qwen-7B are provided in Appendix D.9. Detailed implementation specifics, including hyperparameters, computational environment, and further configuration details, are provided in Appendix B.

6 Experimental results and analysis

6.1 Main results

We conduct extensive experiments across multiple benchmarks, including Math Easy, Math Hard, and Scientific, using four models of varying scales, including those trained with distillation and reinforcement learning. As shown in Table 1, RPDI-EE delivers the largest average performance improvement over vanilla CoT among all tested early-exit methods, with a mean accuracy gain of 5.0 percentage points. The fixed token-budget methods (NoThinking, ThinkLess) do not adapt their reasoning budgets to problem difficulty, which can lead to premature termination. In these experiments, they reduce average accuracy by up to 24.4 percentage points relative to vanilla CoT. Dynasor-CoT also has lower average accuracy than vanilla CoT on all four models. A possible explanation is that consistency among probed answers need not indicate productive reasoning progress, which can lead to premature termination. RPDI-EE bases its stopping decisions on an entropy proxy for likely reasoning path deviation, rather than on fixed token budgets or external probing signals. Per-benchmark results are provided in Appendix D.7. Furthermore, RPDI-EE achieves larger average gains on the three distillation models (6.1 percentage points) than on the RL-trained Qwen3-30B (1.8 percentage points). This is consistent with the reasoning path deviation perspective. Distillation models may capture surface patterns of long CoT without fully learning the underlying logic [6, 38], which could make them more susceptible to reasoning path deviation and more likely to benefit from RPDI-based early exit.

To directly quantify the reliability of these early-exit decisions, we track sample-level answer transitions between vanilla CoT and RPDI-EE across all four models and seven benchmarks. As shown in Table 2, the correction rate (19%–27%) consistently and substantially exceeds the degradation rate

Table 2: Sample-level transition analysis of RPDI-EE. Correction rate denotes the percentage of originally incorrect samples that become correct. Degradation rate denotes the percentage of originally correct samples that become incorrect. Both metrics are computed with respect to the vanilla CoT baseline.

Model	Correction Rate (%)	Degradation Rate (%)
DeepSeek-R1-Distill-Qwen-7B	21.00	5.00
DeepSeek-R1-Distill-Qwen-14B	25.23	3.93
DeepSeek-R1-Distill-Qwen-32B	26.92	2.73
Qwen3-30B-A3B-Thinking-2507	19.56	1.48

(1%–5%). These results show that RPDI-EE corrects a portion of answers that are incorrect under Vanilla CoT while preserving most answers that are already correct. When early exit is triggered, RPDI-EE generates the final answer from the reasoning accumulated so far. This pattern is consistent with the intended role of RPDI-based early exit in limiting unproductive continuation.

Inference efficiency We conduct experiments to quantify the spurious efficiency gains problem discussed in Section 1 under identical deployment settings (vLLM, 8×H20 GPUs, max GPU memory utilization 0.85). A detailed theoretical analysis is provided in Appendix D.2. Table 3 reports end-to-end latency, including decoding and stopping-related work. Across the four models, DEER takes 1.3×–2.3× the Vanilla CoT time despite generating fewer tokens on average, whereas RPDI-EE takes 1.1×–1.2×. DEER obtains stopping signals through tentative-answer generation and context switching, whereas RPDI-EE computes entropy within the autoregressive generation process without switching to answer probing. RPDI-EE thus has lower latency than DEER, with a small overhead relative to Vanilla CoT. The additional comparison with NEAT is provided in Appendix D.9.

Table 3: Average per-sample wall-clock inference time (seconds) across benchmarks.

Model	Method	AIME24	AIME25	AMC	GPQA	GSM8K	MATH500	Olympiad	Avg
DeepSeek-R1-7B	Vanilla CoT	3.0	2.9	1.9	1.5	0.2	0.5	0.9	1.5
	DEER	7.4	6.5	4.1	2.8	0.2	1.2	1.6	3.4
	RPDI-EE	3.3	3.5	2.1	1.9	0.2	0.6	1.2	1.8
DeepSeek-R1-14B	Vanilla CoT	4.1	4.7	2.6	1.6	0.2	0.7	1.3	2.2
	DEER	9.4	11.7	6.2	2.5	0.5	1.5	2.4	4.9
	RPDI-EE	4.5	5.2	2.9	1.8	0.3	0.8	1.6	2.4
DeepSeek-R1-32B	Vanilla CoT	6.0	6.7	4.0	2.3	0.3	0.9	2.0	3.2
	DEER	18.2	10.9	10.7	4.6	0.6	2.8	3.8	7.4
	RPDI-EE	6.4	7.4	4.2	2.6	0.3	1.0	2.3	3.5
Qwen3-30B-Think	Vanilla CoT	10.7	10.8	4.0	2.0	0.1	1.1	3.1	4.5
	DEER	13.2	13.4	7.7	2.5	0.2	1.4	3.4	6.0
	RPDI-EE	10.7	11.9	4.7	2.3	0.2	1.3	3.4	4.9

6.2 Analysis of early-exit triggering impact

To comprehensively analyze the impact of early-exit triggering on the reasoning process, we conduct a unified evaluation across four representative models. All experimental configurations are consistent with the main experiments. As shown in Table 4, the early-exit trigger rate of RPDI-EE exhibits a clear positive trend with task difficulty, ranging from 2.0% on GSM8K to 45.0% on AIME2025. This is consistent with the view in Section 1 that reasoning path deviation is associated with problem difficulty [36]. On harder tasks, models are more likely to make errors that lead to meaningless or even harmful wandering among alternative approaches, producing the characteristic surge of high-entropy transition tokens that RPDI detects. On simpler tasks, models typically reach the correct answer without such deviation, and RPDI-EE accordingly leaves their reasoning intact. Furthermore, across all benchmarks, the accuracy of triggered samples improves consistently after early exit ($+8.6\%$ to $+51.2\%$). This confirms that the truncated content is genuinely meaningless or even harmful, matching the definition of overthinking established in Section 1.

Table 4: Early-exit triggering analysis across benchmarks of increasing difficulty (averaged over four models). Trigger rate denotes the proportion of samples where RPDI-EE activates early exit. Δ Acc. denotes the accuracy change of triggered samples after early exit compared to Vanilla CoT.

	GSM8K	**MATH500**	**AMC**	**GPQA**	**AIME24**	**Olympiad**	**AIME25**
Trigger Rate (%)	2.0	6.8	13.1	25.6	34.2	34.7	45.0
Δ Acc. (%)	+8.6	+22.8	+51.2	+36.2	+20.4	+16.4	+25.2

Table 5: Effectiveness ablation of the RPDI components.

METHOD	MATH EASY	MATH HARD	SCIENTIFIC	AVERAGE
Vanilla CoT	88.7	50.4	56.1	67.6
RPDI-EE	93.5	**61.0**	64.7	**75.5**
w/o GTF	93.5	57.7	64.7	74.0
w/o LTF	90.2	56.4	62.1	71.7
w/o BTM	**93.6**	53.5	**67.7**	72.7

6.3 Ablation

In this section, we conduct experiments to verify the effectiveness of the components of RPDI and its sensitivity to hyperparameters. We employ DeepSeek-R1-Distill-Qwen-32B as the primary model with experimental settings consistent with the main experiments.

Ablation on effectiveness of the RPDI components We ablate the Local Transition Frequency (LTF), Global Transition Frequency (GTF), and Boundary-Triggered mechanism (BTM) in RPDI (Table 5). Fixing $GTF_i \equiv 1$ reduces RPDI to a raw local entropy threshold: accuracy remains 93.5% on MATH EASY but drops by 3.3% on MATH HARD. This supports GTF's adaptive normalization for difficulty-dependent transition-token frequencies, avoiding over-truncation on complex tasks. Removing LTF by setting $W = 1$ compares each token's entropy against the global baseline and reduces average accuracy by 3.8%. An isolated high-entropy token is insufficient to indicate deviation; a rise in local density indicates frequent switching among alternatives (Section 3). Removing BTM evaluates RPDI at every token generation step, reducing average accuracy by 2.8% and accuracy on MATH HARD by 7.5%. This may interrupt productive computation within a reasoning step when local entropy temporarily rises. BTM restricts evaluation to semantic boundaries between coherent reasoning units.

Ablation on hyperparameter settings We evaluate RPDI-EE across a range of hyperparameter settings including the window size W, the RPDI threshold λ, and the token budget. As shown in Figure 4, RPDI-EE consistently outperforms the vanilla CoT baseline under all tested hyperparameter settings, confirming its robustness. Among the tested settings, $W = 512$ and $\lambda = 2.0$ give the highest accuracy in the window-size and threshold sweeps, respectively. A detailed analysis of each hyperparameter's influence is provided in Appendix D.1, along with additional window-size and threshold ablations on DeepSeek-R1-Distill-Llama-8B (Table 7).

7 Conclusion

We observe that when an LRLM's reasoning path deviates into meaningless or even harmful wandering, this deviation is accompanied by an anomalous surge of high-entropy transition tokens. Based on this, we propose RPDI-EE, a training-free early-exit method that monitors internal entropy signals as a proxy for reasoning path deviation to make better-informed truncation decisions during inference. Unlike existing early-exit methods that often suffer from over-truncation or spurious efficiency gains, RPDI-EE only terminates reasoning when entropy dynamics indicate likely deviation into meaningless or even harmful wandering. It requires no additional training, proxy models, or answer probing. Extensive experiments across multiple benchmarks and four LRLMs demonstrate that RPDI-EE achieves the largest average accuracy improvement over vanilla CoT among all tested early-exit methods, while achieving lower average end-to-end latency than the other early-exit methods in our runtime comparisons.

Acknowledgments and Disclosure of Funding

The authors received no third-party funding or support for this work. The authors declare no competing interests.

References

[1] Ömer Faruk Akgül, Yusuf Hakan Kalaycı, Rajgopal Kannan, Willie Neiswanger, and Viktor Prasanna. Lynx: Learning dynamic exits for confidence-controlled reasoning. *arXiv preprint arXiv:2512.05325*, 2025.

[2] Qiguang Chen, Libo Qin, Jiaqi Wang, Jingxuan Zhou, and Wanxiang Che. Unlocking the capabilities of thought: A reasoning boundary framework to quantify and optimize chain-of-thought. *Advances in Neural Information Processing Systems*, 37:54872–54904, 2024.

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

[4] Yu-Neng Chuang, Prathusha Kameswara Sarma, Parikshit Gopalan, John Boccio, Sara Bolouki, Xia Hu, and Helen Zhou. Learning to route llms with confidence tokens. *arXiv preprint arXiv:2410.13284*, 2024.

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

[6] Chengwei Dai, Kun Li, Wei Zhou, and Songlin Hu. Capture the key in reasoning to enhance cot distillation generalization. In *Proceedings of the 63rd Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, pages 441–465, 2025.

[7] Yifu Ding, Wentao Jiang, Shunyu Liu, Yongcheng Jing, Jinyang Guo, Yingjie Wang, Jing Zhang, Zengmao Wang, Ziwei Liu, Bo Du, et al. Dynamic parallel tree search for efficient llm reasoning. *arXiv preprint arXiv:2502.16235*, 2025.

[8] Roy Eisenstadt, Itamar Zimerman, and Lior Wolf. Overclocking llm reasoning: Monitoring and controlling thinking path lengths in llms. *arXiv preprint arXiv:2506.07240*, 2025.

[9] Chenrui Fan, Ming Li, Lichao Sun, and Tianyi Zhou. Missing premise exacerbates overthinking: Are reasoning models losing critical thinking skill? *arXiv preprint arXiv:2504.06514*, 2025.

[10] Gongfan Fang, Xinyin Ma, and Xinchao Wang. Thinkless: Llm learns when to think. *arXiv preprint arXiv:2505.13379*, 2025.

[11] Mehdi Fatemi, Banafsheh Rafiee, Mingjie Tang, and Kartik Talamadupula. Concise reasoning via reinforcement learning. *arXiv preprint arXiv:2504.05185*, 2025.

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

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

[14] Tingxu Han, Zhenting Wang, Chunrong Fang, Shiyu Zhao, Shiqing Ma, and Zhenyu Chen. Token-budget-aware llm reasoning. *arXiv preprint arXiv:2412.18547*, 2024.

[15] Michael Hassid, Gabriel Synnaeve, Yossi Adi, and Roy Schwartz. Don't overthink it. preferring shorter thinking chains for improved llm reasoning. *arXiv preprint arXiv:2505.17813*, 2025.

[16] Chaoqun He, Renjie Luo, Yuzhuo Bai, Shengding Hu, Zhen Leng Thai, Junhao Shen, Jinyi Hu, Xu Han, Yujie Huang, Yuxiang Zhang, et al. Olympiadbench: A challenging benchmark for promoting agi with olympiad-level bilingual multimodal scientific problems. *arXiv preprint arXiv:2402.14008*, 2024.

[17] Dan Hendrycks, Collin Burns, Saurav Kadavath, Akul Arora, Steven Basart, Eric Tang, Dawn Song, and Jacob Steinhardt. Measuring mathematical problem solving with the math dataset. *arXiv preprint arXiv:2103.03874*, 2021.

[18] Guochao Jiang, Guofeng Quan, Zepeng Ding, Ziqin Luo, Dixuan Wang, and Zheng Hu. Flashthink: An early exit method for efficient reasoning. *arXiv preprint arXiv:2505.13949*, 2025.

[19] Lingjie Jiang, Xun Wu, Shaohan Huang, Qingxiu Dong, Zewen Chi, Li Dong, Xingxing Zhang, Tengchao Lv, Lei Cui, and Furu Wei. Think only when you need with large hybrid-reasoning models. *arXiv preprint arXiv:2505.14631*, 2025.

[20] Yu Kang, Xianghui Sun, Liangyu Chen, and Wei Zou. C3ot: Generating shorter chain-of-thought without compromising effectiveness. In *Proceedings of the AAAI Conference on Artificial Intelligence*, pages 24312–24320, 2025.

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

[22] Gengyang Li, Yifeng Gao, Yuming Li, and Yunfang Wu. Thinkless: A training-free inference-efficient method for reducing reasoning redundancy. *arXiv preprint arXiv:2505.15684*, 2025.

[23] Kang Liu, Yongkang Liu, Xiaocui Yang, Peidong Wang, Wen Zhang, Shi Feng, Yifei Zhang, and Daling Wang. NEAT: Neuron-based early exit for large reasoning models. *arXiv preprint arXiv:2602.02010*, 2026.

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

[25] Chenwei Lou, Zewei Sun, Xinnian Liang, Meng Qu, Wei Shen, Wenqi Wang, Yuntao Li, Qingping Yang, and Shuangzhi Wu. Adacot: Pareto-optimal adaptive chain-of-thought triggering via reinforcement learning. *arXiv preprint arXiv:2505.11896*, 2025.

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

[27] MAA. Amc 2023 problems, 2023. Accessed: 2025-05-11.

[28] MAA Committees. Aime problems and solutions. `https://artofproblemsolving.com/wiki/index.php/AIME_Problems_and_Solutions`, 2025.

[29] Niklas Muennighoff, Zitong Yang, Weijia Shi, Xiang Lisa Li, Li Fei-Fei, Hannaneh Hajishirzi, Luke Zettlemoyer, Percy Liang, Emmanuel Candès, and Tatsunori Hashimoto. s1: Simple test-time scaling. *arXiv preprint arXiv:2501.19393*, 2025.

[30] Keqin Peng, Liang Ding, Yuanxin Ouyang, Meng Fang, and Dacheng Tao. Revisiting overthinking in long chain-of-thought from the perspective of self-doubt. *arXiv preprint arXiv:2505.23480*, 2025.

[31] Ziqing Qiao, Yongheng Deng, Jiali Zeng, Dong Wang, Lai Wei, Fandong Meng, Jie Zhou, Ju Ren, and Yaoxue Zhang. Concise: Confidence-guided compression in step-by-step efficient reasoning. *arXiv preprint arXiv:2505.04881*, 2025.

[32] Qwen Team. Qwq-32b: Embracing the power of reinforcement learning, 2025.

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

[34] Matthew Renze and Erhan Guven. 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, 2024.

[35] Yi Shen, Jian Zhang, Jieyun Huang, Shuming Shi, Wenjing Zhang, Jiangze Yan, Ning Wang, Kai Wang, Zhaoxiang Liu, and Shiguo Lian. Dast: Difficulty-adaptive slow-thinking for large reasoning models. *arXiv preprint arXiv:2503.04472*, 2025.

[36] Jinyan Su, Jennifer Healey, Preslav Nakov, and Claire Cardie. Between underthinking and overthinking: An empirical study of reasoning length and correctness in llms. *arXiv preprint arXiv:2505.00127*, 2025.

[37] Hanshi Sun, Momin Haider, Ruiqi Zhang, Huitao Yang, Jiahao Qiu, Ming Yin, Mengdi Wang, Peter Bartlett, and Andrea Zanette. Fast best-of-n decoding via speculative rejection. *Advances in Neural Information Processing Systems*, 37:32630–32652, 2024.

[38] Chenlong Wang, Yuanning Feng, Dongping Chen, Zhaoyang Chu, Ranjay Krishna, and Tianyi Zhou. Wait, we don't need to" wait"! removing thinking tokens improves reasoning efficiency. *arXiv preprint arXiv:2506.08343*, 2025.

[39] Shenzhi Wang, Le Yu, Chang Gao, Chujie Zheng, Shixuan Liu, Rui Lu, Kai Dang, Xionghui Chen, Jianxin Yang, Zhenru Zhang, et al. Beyond the 80/20 rule: High-entropy minority tokens drive effective reinforcement learning for llm reasoning. *arXiv preprint arXiv:2506.01939*, 2025.

[40] Yiming Wang, Pei Zhang, Siyuan Huang, Baosong Yang, Zhuosheng Zhang, Fei Huang, and Rui Wang. Sampling-efficient test-time scaling: Self-estimating the best-of-n sampling in early decoding. *arXiv preprint arXiv:2503.01422*, 2025.

[41] Zihao Wei, Liang Pang, Jiahao Liu, Wenjie Shi, Jingcheng Deng, Shicheng Xu, Zenghao Duan, Fei Sun, Huawei Shen, and Xueqi Cheng. The evolution of thought: Tracking LLM overthinking via reasoning dynamics analysis. *arXiv preprint arXiv:2508.17627*, 2025.

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

[43] Yuchen Yan, Yongliang Shen, Yang Liu, Jin Jiang, Mengdi Zhang, Jian Shao, and Yueting Zhuang. Inftythink: Breaking the length limits of long-context reasoning in large language models. *arXiv preprint arXiv:2503.06692*, 2025.

[44] An Yang, Anfeng Li, Baosong Yang, Beichen Zhang, Binyuan Hui, Bo Zheng, Bowen Yu, Chang Gao, Chengen Huang, Chenxu Lv, et al. Qwen3 technical report. *arXiv preprint arXiv:2505.09388*, 2025.

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

[46] Rubing Yang, Huajun Bai, Song Liu, Guanghua Yu, Runzhi Fan, Yanbin Dang, Jiejing Zhang, Kai Liu, Jianchen Zhu, and Peng Chen. Specexit: Accelerating large reasoning model via speculative exit. *arXiv preprint arXiv:2509.24248*, 2025.

[47] Edward Yeo, Yuxuan Tong, Morry Niu, Graham Neubig, and Xiang Yue. Demystifying long chain-of-thought reasoning in llms. *arXiv preprint arXiv:2502.03373*, 2025.

[48] Xixian Yong, Xiao Zhou, Yingying Zhang, Jinlin Li, Yefeng Zheng, and Xian Wu. Think or not? exploring thinking efficiency in large reasoning models via an information-theoretic lens. *arXiv preprint arXiv:2505.18237*, 2025.

[49] Anqi Zhang, Yulin Chen, Jane Pan, Chen Zhao, Aurojit Panda, Jinyang Li, and He He. Reasoning models know when they're right: Probing hidden states for self-verification. *arXiv preprint arXiv:2504.05419*, 2025.

[50] Jintian Zhang, Yuqi Zhu, Mengshu Sun, Yujie Luo, Shuofei Qiao, Lun Du, Da Zheng, Huajun Chen, and Ningyu Zhang. Lightthinker: Thinking step-by-step compression. *arXiv preprint arXiv:2502.15589*, 2025.

Algorithm 1 RPDI-EE early-exit algorithm.

Require: Model M, prompt P, token budget L_{max}, local content size W, threshold λ, boundary set $\mathcal{B}$
Ensure: Final answer A

/* Initialize trajectory and entropy buffers */
$R \leftarrow \emptyset, \mathcal{H} \leftarrow [\,], S_{\text{global}} \leftarrow 0.0, S_{\text{local}} \leftarrow 0.0$
$i \leftarrow 0$, early_exit $\leftarrow$ False
/* Introspective loop with real-time scanning */
while not early_exit **do**
  $i \leftarrow i + 1$
  Sample $t_i \sim M(P \oplus R)$, compute entropy $H(t_i)$
  $R \leftarrow R \oplus t_i, \mathcal{H}$.append$(H(t_i))$
  /* Incremental entropy tracking */
  $S_{\text{global}} \leftarrow S_{\text{global}} + H(t_i), S_{\text{local}} \leftarrow S_{\text{local}} + H(t_i)$
  /* Local context update */
  if $i > W$ **then**
    $S_{\text{local}} \leftarrow S_{\text{local}} - \mathcal{H}[i - W]$
  end if
  /* Early exit via RPDI detection */
  if $i \geq W$ **and** $t_i \in \mathcal{B}$ **then**
    GTF $\leftarrow S_{\text{global}}/i$, LTF $\leftarrow S_{\text{local}}/W$
    RPDI $\leftarrow$ LTF/GTF
    early_exit $\leftarrow$ RPDI $> \lambda$
  end if
end while
/* Transition to answer synthesis phase */
$R \leftarrow R \oplus$ `</think>`
Sample $A \sim M(P \oplus R)$ within budget $L_{\text{max}} - i$
Return A

A Algorithm and method details

A.1 Algorithm

Algorithm 1 presents the complete execution flow of our method in pseudocode. Together, these three components monitor entropy dynamics in real time, detect reasoning path deviation, and trigger early exit before meaningless or even harmful reasoning accumulates.

A.2 Incremental entropy accumulation

To efficiently compute both local and global entropy sums without redundant calculations, we design an incremental entropy accumulation mechanism. This mechanism maintains running sums of entropy values, eliminating the need to re-sum the entire history each time local or global statistics are required. Specifically, we maintain two accumulator variables: $\mathcal{S}^i_{\text{global}}$ for the cumulative sum of all token entropies from the beginning of reasoning, and $\mathcal{S}^i_{\text{local}}$ for the sum of entropies within the most recent sliding window of size W. Upon generating a new token t_i with entropy $H(t_i)$, the global sum is updated at every step, while the local update has two cases. Both sums are initialized to zero:

$$\mathcal{S}^i_{\text{global}} = \mathcal{S}^{i-1}_{\text{global}} + H(t_i), \tag{6}$$

$$\mathcal{S}^i_{\text{local}} = \mathcal{S}^{i-1}_{\text{local}} + H(t_i), \quad \text{if } i \leq W. \tag{7}$$

When $i > W$, the local update also subtracts the entropy leaving the window, where $\mathcal{H}[i - W] = H(t_{i-W})$:

$$\mathcal{S}^i_{\text{local}} = \mathcal{S}^{i-1}_{\text{local}} + H(t_i) - \mathcal{H}[i - W], \quad \text{if } i > W. \tag{8}$$

This $O(1)$ update strategy ensures that both local and global entropy sums can be computed efficiently, avoiding the $O(n)$ complexity of frequently summing over the entire history or window. The stored entropy sequence $\mathcal{H}$ is only used for this subtraction operation, not for recomputing sums from scratch.

Table 6: Feature-level comparison between RPDI and existing early-exit frameworks. We evaluate methods across six critical dimensions: (1) **Probing-free**: avoids intermediate answer generation to maintain a continuous reasoning flow; (2) **Proxy-Model-free**: operates without auxiliary verifiers or additional training; (3) **Dynamic Method**: adaptively adjusts the reasoning length based on problem complexity; (4) **Efficiency Gain**: provides measurable reduction in generated token count compared to vanilla CoT; (5) **Performance Gain**: achieves accuracy improvements through reliable truncation decisions that safeguard against over-truncation; (6) **Real-time**: functions during the generation process rather than via offline post-processing.

Method	Probe-free	Proxy-free	Dynamic	Effic. Gain	Perf. Gain	Real-time
RPDI (Ours)	✓	✓	✓	✓	✓	✓
Answer-Probing Based						
Deer [45]	✗	✓	✓	✓	✓	✓
Adaptive Think [48]	✗	✓	✓	✓	✓	✗
Dynasor-CoT [12]	✗	✓	✓	✓	✗	✓
Proxy-Model Based						
SpecExit [46]	✓	✗	✓	✓	✗	✓
Flashthink [18]	✓	✗	✓	✓	✗	✓
Zhang et al. [49]	✓	✗	✓	✓	✗	✓
LYNX [1]	✗	✗	✓	✓	✓	✓
Fixed Length-Based						
Token Budget [29]	✓	✓	✗	✓	✗	✓
ThinkLess [22]	✓	✓	✗	✓	✗	✓
NoThinking [26]	✓	✓	✗	✓	✗	✓

A.3 Comparison of early-exit methods

As shown in Table 6, RPDI-EE is the only method that is simultaneously probing-free, proxy-model-free, dynamic, and real-time, while mitigating the spurious efficiency gains of existing methods.

B Implementation details

All experiments are implemented using the vLLM framework to ensure efficient inference and memory management. For RPDI-EE, we set the entropy scanning window size W to 512 tokens and the RPDI threshold to 2.0. The boundary symbol set $\mathcal{B}$ consists of tokens that mark the completion of a reasoning step: `}`, `}\n`, `}\n\n`, `}.`, `}.\n`, `}\`, `}}`, `)}`, `)}.`, and `)}\n`, following Yang et al. [45]. For Dynasor-CoT, we set the number of consecutive consistent results to 3 and the probing interval to 512 tokens. Regarding context limits, we use a standard maximum length of 16,384 tokens for most models, while extending it to 32,768 tokens for the Qwen3-Thinking series to accommodate their extended reasoning chains. The computational experiments are carried out on a cluster of 8×H20 GPUs. For all tasks, we employ greedy decoding to ensure deterministic results and use accuracy and the number of tokens generated as the primary evaluation metrics.

C Extended related works

Training-time optimization aims to encode efficient reasoning patterns directly into model parameters. A common strategy fine-tunes models on datasets where verbose reasoning is distilled into concise chains [11, 35]. Another line of work employs reinforcement learning with efficiency-driven rewards such as length penalties to encourage shorter reasoning [31, 20, 24]. More radical approaches compress reasoning into latent representations [47, 25], improving token efficiency but often at the expense of the verifiability provided by explicit reasoning. While these methods can produce efficient reasoners, the efficiency bias becomes permanently encoded in model parameters, limiting adaptivity to dynamic computational budgets without costly retraining.

Inference-time optimization aims to adjust the reasoning procedure without modifying model parameters. Multi-path reasoning generates several chains to mitigate individual errors [7, 40, 37],

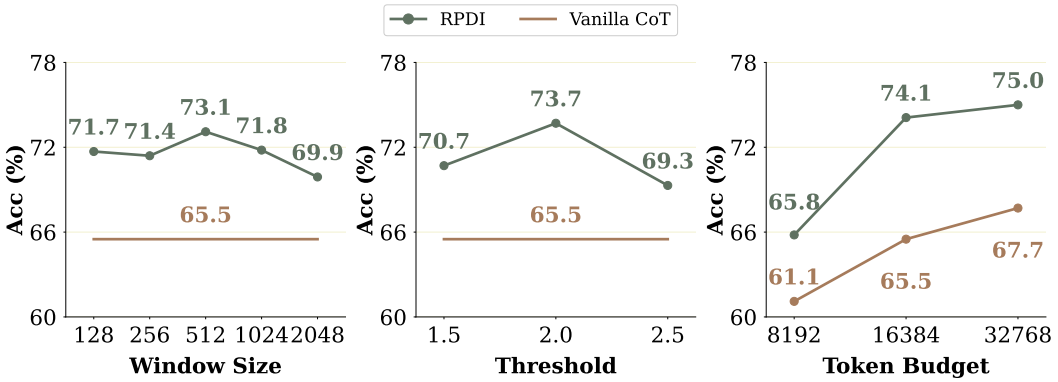


Figure 4: Hyperparameter ablation of RPDI-EE across different window sizes W, RPDI thresholds λ, and token budgets.

but is computationally expensive and subject to diminishing accuracy gains from additional reasoning paths due to high correlations across chains. Difficulty-based approaches trigger long CoT only when problems appear challenging [10, 19, 4], but once activated, they provide little control and may still lead to redundant reasoning. Prompt-based techniques dynamically modulate reasoning length [42, 14, 2, 34, 21], though they remain sensitive to prompt design and rely on the model's intrinsic ability. Additionally, compressing generated reasoning paths can reduce context size, but does not reduce the cost of generating redundant tokens, and compressing generated reasoning paths may lead to discarding essential information [43, 50].

D Additional experiments

D.1 Hyperparameter sensitivity analysis

As shown in Figure 4, RPDI-EE consistently outperforms vanilla CoT under all tested configurations. Among the tested settings, $W = 512$ and $\lambda = 2.0$ give the highest accuracy in the window-size and threshold sweeps, respectively. The window size W governs the trade-off between detecting sustained deviation and reacting to transient noise: at W=256, the window is too small to distinguish isolated high-entropy tokens from a genuine surge in transition token density; at W=1024, the local average is diluted by too much history, delaying detection. The threshold λ controls the sensitivity of RPDI: a lower value (e.g., λ=1.5) makes the model more prone to over-truncation, as even modest fluctuations in local entropy can trigger early exit; a higher value (e.g., λ=3.0) raises the bar for triggering, potentially allowing some meaningless or even harmful reasoning to persist but reducing the risk of premature termination.

Table 7: Hyperparameter sensitivity on DeepSeek-R1-Distill-Llama-8B. Accuracy is in percent; Avg. averages the three benchmarks. The W sweep fixes $\lambda = 2.0$, and the λ sweep fixes $W = 512$. Their shared default configuration appears once and is shaded.

W	λ	AMC23	AIME25	GPQA-D	Avg.
Vanilla CoT		57.50	30.00	21.21	36.24
256	2.0	85.00	30.00	23.74	46.25
512	2.0	87.50	30.00	25.25	47.58
1024	2.0	87.50	26.67	24.75	46.31
512	1.5	77.50	23.33	23.23	41.35
512	2.5	77.50	26.67	24.75	42.97

Table 7 adds a sensitivity analysis on DeepSeek-R1-Distill-Llama-8B using AMC23, AIME25, and GPQA-Diamond. Among the tested configurations, $W = 512$ and $\lambda = 2.0$ achieve the highest mean accuracy (47.58%). The window sweep fixes $\lambda = 2.0$, and the threshold sweep fixes $W = 512$.

Table 8: Controlled study on reasoning budget and answer generation effects.

Method	AMC23	GSM8K	MATH500	AIME2024	AIME2025	GPQA	Avg.
Vanilla CoT	87.50	89.69	89.00	63.33	36.67	56.06	70.38
RPDI-EE	**92.50**	**95.68**	**92.40**	**70.00**	**53.33**	**64.65**	**78.09**
Fixed-Length CoT	87.50	87.19	86.40	66.67	53.33	58.08	73.20
CoT (80% Budget)	87.50	95.07	91.20	56.67	50.00	57.07	72.92
CoT (90% Budget)	85.00	95.45	92.00	53.33	46.67	59.60	72.01

Table 9: Performance and token usage under stochastic decoding (T=0.6, top-k=30, top-p=0.95, 16 samples).

		AMC23		AIME2025		GPQA		Avg.	
Method	**Model**	Acc	Tokens	Acc	Tokens	Acc	Tokens	Acc	Tokens
Vanilla CoT	QwQ-32B	95.00	6993	66.67	15078	60.10	8263	73.92	10112
RPDI-EE	QwQ-32B	100.00	6893	63.33	15248	68.18	7187	77.17	9776
Vanilla CoT	DeepSeek-R1-Distill-Llama-70B	95.00	5157	50.00	12255	59.09	5652	68.03	7688
RPDI-EE	DeepSeek-R1-Distill-Llama-70B	95.00	4847	53.33	10627	62.12	5600	70.15	7025

D.2 Latency comparison

Theoretical analysis Let T_{reason} and M_{reason} denote the time and memory for generating the main reasoning chain (including KV cache). The total overhead of each method is as follows.

For DEER:

$$T_{\text{DEER}} = T_{\text{reason}} + T_{\text{probe}} + T_{\text{anseval}} + T_{\text{switch}}, \quad M_{\text{DEER}} = M_{\text{reason}} + M_{\text{probe}}, \tag{9}$$

where T_{probe} is the cost of generating tentative answers at reasoning boundaries, T_{anseval} is the cost of evaluating their confidence, T_{switch} is the context-switching cost between reasoning and probing, and M_{probe} is the memory for tentative branches and their KV cache.

For RPDI-EE:

$$T_{\text{RPDI}} = T_{\text{reason}} + T_{\text{entropy}} + T_{\text{stat}} + T_{\text{trigger}}, \quad M_{\text{RPDI}} = M_{\text{reason}}, \tag{10}$$

where T_{entropy} is the cost of token-level entropy extraction, T_{stat} is the cost of updating local/global running statistics, and T_{trigger} is the cost of triggering RPDI checks at boundary positions. T_{entropy}, T_{stat}, and T_{trigger} are lightweight ($O(1)$ per token and only checked at boundaries), and M_{RPDI} only requires a small entropy buffer and a few running statistics beyond M_{reason}, so the additional memory overhead is negligible.

Empirical validation The empirical latency comparison is presented in the main text (Table 3 in Section 6). The results are consistent with the theoretical analysis above. RPDI-EE avoids the $T_{\text{probe}} + T_{\text{anseval}} + T_{\text{switch}}$ overhead associated with DEER's probing mechanism. The measurements report total inference time, including reasoning generation and the work required to obtain stopping signals.

D.3 Budget control experiment

To disentangle the effect of trajectory-aware truncation from answer-budget reallocation, we compare RPDI-EE against three controlled baselines on DeepSeek-R1-Distill-Qwen-32B. These baselines are: (1) Fixed-Length CoT, which truncates all samples at the average reasoning length of RPDI-EE; (2) CoT with 80% budget, which limits the maximum reasoning budget to 80% of the original; and (3) CoT with 90% budget.

As shown in Table 8, fixed-length truncation at the same average token budget yields substantially lower accuracy than RPDI-EE, confirming that the benefit does not stem from simply reducing reasoning length. Budget reduction baselines (80% and 90% CoT budgets) also underperform RPDI-EE, indicating that freely allocating more tokens to answer generation, without detecting where reasoning path deviation occurs, captures at best a minor fraction of the gain. The primary improvement

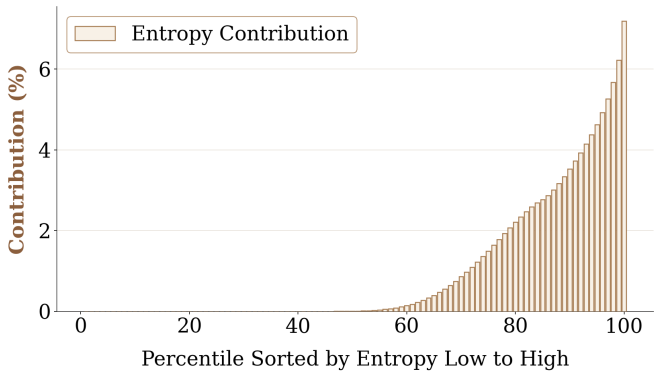


Figure 5: Token entropy contribution distribution for Qwen3-30B-A3B-Thinking-2507, showing the same long-tail pattern as the DeepSeek-R1-Distill-Qwen series in Figure 2a.

of RPDI-EE comes from detecting reasoning path deviation through RPDI and truncating only meaningless or even harmful segments, rather than uniformly reducing reasoning length.

D.4 Sampling decoding experiment

We evaluate RPDI-EE under stochastic decoding (T=0.6, top-k=30, top-p=0.95) with 16 samples per input to verify that its advantage is not tied to greedy decoding.

Table 9 shows that RPDI-EE achieves higher mean accuracy and lower mean token usage than Vanilla CoT on both evaluated models under stochastic decoding.

D.5 Cross-architecture entropy distribution

To verify that the long-tail entropy distribution observed in Section 3 generalizes beyond the DeepSeek-R1-Distill-Qwen architecture, we conduct the same analysis on Qwen3-30B-A3B-Thinking-2507 across the same seven benchmarks under vanilla CoT, covering 15.5 million tokens in total.

As shown in Figure 5, the same long-tail pattern holds for Qwen3-30B-A3B-Thinking-2507. A small portion of tokens contributes the majority of entropy. This cross-architecture consistency ensures that the design premise of RPDI-EE, namely that average entropy serves as a soft proxy for high-entropy transition token density, is not specific to a single model family but generalizes across training paradigms and architectures.

D.6 Extended results on additional models

To further validate the generalizability of RPDI-EE beyond the four models presented in the main text, we evaluate it on six additional LRLMs spanning different architectures and scales: DeepSeek-R1-Distill-Qwen-1.5B, DeepSeek-R1-Distill-Llama-8B/70B [13], DeepSeek-R1-0528-Qwen3-8B, QwQ-32B [32], and Qwen3-235B-A22B-Thinking-2507 [44]. Due to the computational cost of running all baselines on large-scale models (up to 235B), we compare RPDI-EE with Vanilla CoT and DEER, the most competitive baseline in our main experiments, on these additional models.

Table 10 presents the accuracy and token length comparison. RPDI-EE improves average accuracy over Vanilla CoT on all six additional models and achieves the highest average accuracy among the compared methods on five of them. The trend observed in the main text, that distillation models benefit more from RPDI-EE than RL-trained models, is further confirmed. The three distillation models (1.5B, Llama-8B, Llama-70B) exhibit average gains of $+3.4\%$, $+5.9\%$, and $+2.7\%$ respectively, while the RL-trained models show smaller but consistent improvements.

Table 11 reports the wall-clock inference time. Across the six additional models, RPDI-EE has lower average end-to-end latency than DEER, with a small overhead relative to Vanilla CoT (Table 11).

Table 10: Extended accuracy and token length comparison on six additional LRLMs. Format follows Table 1. Subscripts denote accuracy change relative to Vanilla CoT. Due to computational constraints, only Vanilla CoT and DEER are compared.

Method	Math Easy		Math Hard		Scientific		Average	
	Acc.↑	Len.↓	Acc.↑	Len.↓	Acc.↑	Len.↓	Acc.↑	Len.↓
DeepSeek-R1-Distill-Qwen-1.5B	65.7	5615	25.7	12646	6.1	13386	40.0	9738
+ DEER	64.3$_{-1.4}$	3202	24.7$_{-1.0}$	10155	4.6$_{-1.5}$	12685	38.8$_{-1.2}$	7536
+ **RPDI-EE**	**69.4**$_{+3.7}$	5177	**29.2**$_{+3.5}$	11394	**8.1**$_{+2.0}$	12823	**43.4**$_{+3.4}$	8934
DeepSeek-R1-Distill-Llama-8B	70.8	5036	32.5	10876	21.2	9604	47.3	8192
+ DEER	73.6$_{+2.8}$	3383	**33.1**$_{+0.6}$	10575	**34.3**$_{+13.1}$	9515	50.6$_{+3.3}$	7341
+ **RPDI-EE**	**82.8**$_{+12.0}$	4271	32.8$_{+0.3}$	10905	25.3$_{+4.1}$	9769	**53.2**$_{+5.9}$	7900
DeepSeek-R1-Distill-Llama-70B	89.0	3639	51.4	9870	60.1	5538	68.8	6581
+ DEER	**91.7**$_{+2.7}$	2200	46.4$_{-5.0}$	8528	62.1$_{+2.0}$	5659	68.0$_{-0.8}$	5406
+ **RPDI-EE**	89.4$_{+0.4}$	3876	**55.8**$_{+4.4}$	9386	**64.7**$_{+4.6}$	5940	**71.5**$_{+2.7}$	6532
DeepSeek-R1-0528-Qwen3-8B	93.8	7444	70.3	20930	57.6	11323	78.5	13778
+ DEER	**95.9**$_{+2.1}$	5138	69.7$_{-0.6}$	19117	59.6$_{+2.0}$	10920	79.5$_{+1.0}$	11955
+ **RPDI-EE**	94.8$_{+1.0}$	6933	**72.6**$_{+2.3}$	18048	**60.6**$_{+3.0}$	11048	**80.4**$_{+1.9}$	12284
QwQ-32B	94.3	4063	59.2	11082	65.7	7438	75.2	7553
+ DEER	**94.7**$_{+0.4}$	3307	**65.4**$_{+6.2}$	9867	**67.2**$_{+1.5}$	7138	**78.2**$_{+3.0}$	6666
+ **RPDI-EE**	94.4$_{+0.1}$	4114	63.7$_{+4.5}$	10953	64.7$_{-1.0}$	6953	77.0$_{+1.8}$	7451
Qwen3-235B-A22B-Thinking-2507	97.7	5805	79.8	17185	80.8	8893	87.6	11123
+ DEER	97.7	4591	82.5$_{+2.7}$	16332	**82.8**$_{+2.0}$	9005	89.1$_{+1.5}$	10253
+ **RPDI-EE**	**97.8**$_{+0.1}$	5869	**84.8**$_{+5.0}$	16642	80.3$_{-0.5}$	8910	**89.7**$_{+2.1}$	10921

Table 11: Extended wall-clock inference time (seconds) on six additional LRLMs. Format follows Table 3.

Model	Method	AIME24	AIME25	AMC	GPQA	GSM8K	MATH500	Olympiad	Avg
DeepSeek-R1-1.5B	Vanilla CoT	2.8	2.5	2.0	1.7	0.2	0.6	1.1	1.5
	DEER	5.2	3.9	3.1	3.3	0.2	0.8	1.4	2.6
	RPDI-EE	2.9	3.3	2.1	2.1	0.2	0.8	1.4	1.8
DeepSeek-R1-8B	Vanilla CoT	3.2	3.0	2.3	1.3	0.2	0.5	1.0	1.7
	DEER	10.8	6.5	7.7	2.3	0.3	1.3	1.8	4.4
	RPDI-EE	3.8	3.7	2.3	1.9	0.2	1.4	1.4	2.1
DeepSeek-R1-70B	Vanilla CoT	9.4	9.8	6.3	2.5	0.3	1.2	3.2	4.7
	DEER	20.9	32.4	9.6	6.1	0.5	3.4	5.0	11.1
	RPDI-EE	9.3	10.0	7.0	3.0	0.3	1.3	3.3	4.9
Qwen3-8B	Vanilla CoT	8.7	9.0	5.0	2.2	0.2	1.2	3.2	4.2
	DEER	12.7	8.2	5.3	2.6	0.2	1.5	3.8	4.9
	RPDI-EE	8.4	8.9	5.3	2.6	0.3	1.5	3.1	4.3
QwQ-32B	Vanilla CoT	6.6	7.4	4.3	2.3	0.3	0.9	2.3	3.5
	DEER	25.4	27.2	13.1	3.2	0.5	2.5	3.8	10.8
	RPDI-EE	6.9	8.2	4.7	2.4	0.3	1.1	2.4	3.7
Qwen3-235B-Think	Vanilla CoT	27.2	33.8	12.8	8.5	0.6	5.5	19.4	15.4
	DEER	35.7	40.8	19.1	21.5	0.6	5.2	18.7	20.2
	RPDI-EE	26.4	35.4	14.0	9.0	0.6	5.7	18.5	15.7

These results demonstrate that the entropy-based deviation signal generalizes across diverse model families, scales (1.5B to 235B), and training paradigms.

D.7 Per-benchmark accuracy

Table 12 reports per-benchmark Pass@1 accuracy for Vanilla CoT, DEER, and RPDI-EE across all ten models. Avg. is the unweighted mean across the seven benchmarks.

Table 12: Per-benchmark Pass@1 accuracy (%) for all ten models. Avg. is the unweighted mean across seven benchmarks. Bold denotes the best value within each model, including ties.

Method	AMC23	GSM8K	MATH 500	AIME24	AIME25	Olympiad Bench	GPQA-D	Avg.
DeepSeek-R1-Distill-Qwen-7B								
Vanilla CoT	**85.00**	89.61	89.00	36.67	26.67	34.22	**32.32**	56.21
DEER	77.50	**90.90**	87.80	50.00	30.00	**55.11**	28.28	59.94
RPDI-EE	**85.00**	89.69	**89.40**	**53.33**	**36.67**	53.63	24.75	**61.78**
DeepSeek-R1-Distill-Qwen-14B								
Vanilla CoT	87.50	93.63	89.00	66.67	40.00	53.04	50.51	68.62
DEER	92.50	90.67	89.40	56.67	**46.67**	56.30	52.02	69.18
RPDI-EE	**95.00**	**94.47**	**91.00**	**73.33**	**46.67**	**57.48**	**56.06**	**73.43**
DeepSeek-R1-Distill-Qwen-32B								
Vanilla CoT	87.50	89.69	89.00	63.33	36.67	51.26	56.06	67.64
DEER	87.50	94.69	90.40	60.00	**63.33**	**59.85**	52.02	72.54
RPDI-EE	**92.50**	**95.68**	**92.40**	**70.00**	53.33	59.56	**64.65**	**75.45**
Qwen3-30B-A3B-Thinking-2507								
Vanilla CoT	**100.00**	95.83	94.40	86.67	83.33	76.15	74.75	87.30
DEER	**100.00**	95.75	95.80	83.33	80.00	**78.07**	74.75	86.81
RPDI-EE	**100.00**	**96.21**	**96.00**	**90.00**	**86.67**	77.04	**77.78**	**89.10**
DeepSeek-R1-Distill-Qwen-1.5B								
Vanilla CoT	52.50	75.28	69.40	**30.00**	16.67	30.37	6.06	40.04
DEER	**60.00**	67.48	65.40	20.00	20.00	**34.22**	4.55	38.81
RPDI-EE	**60.00**	**75.66**	**72.60**	**30.00**	**23.33**	**34.22**	**8.08**	**43.41**
DeepSeek-R1-Distill-Llama-8B								
Vanilla CoT	57.50	76.57	78.40	**26.67**	**30.00**	40.89	21.21	47.32
DEER	67.50	74.22	79.00	**26.67**	26.67	**46.07**	**34.34**	50.64
RPDI-EE	**87.50**	**78.24**	**82.60**	23.33	**30.00**	45.19	25.25	**53.16**
DeepSeek-R1-Distill-Llama-70B								
Vanilla CoT	82.50	94.77	89.80	56.67	**40.00**	57.63	60.10	68.78
DEER	**90.00**	**95.07**	90.00	50.00	30.00	59.11	62.12	68.04
RPDI-EE	82.50	94.54	**91.20**	**66.67**	**40.00**	**60.74**	**64.65**	**71.47**
DeepSeek-R1-0528-Qwen3-8B								
Vanilla CoT	95.00	93.25	93.00	**76.67**	70.00	64.15	57.58	78.52
DEER	**100.00**	93.33	**94.40**	73.33	66.67	**69.04**	59.60	79.48
RPDI-EE	97.50	**94.09**	92.80	**76.67**	**73.33**	67.70	**60.61**	**80.39**
QwQ-32B								
Vanilla CoT	**92.50**	96.36	94.00	60.00	53.33	64.30	65.66	75.16
DEER	**92.50**	**96.44**	**95.20**	**73.33**	**56.67**	**66.22**	**67.17**	**78.22**
RPDI-EE	**92.50**	**96.44**	94.20	**73.33**	53.33	64.44	64.65	76.98
Qwen3-235B-A22B-Thinking-2507								
Vanilla CoT	**100.00**	96.66	96.40	**90.00**	76.67	72.74	80.81	87.61
DEER	**100.00**	96.66	96.40	**90.00**	80.00	**77.63**	**82.83**	89.07
RPDI-EE	**100.00**	**96.82**	**96.60**	**90.00**	**86.67**	**77.63**	80.30	**89.72**

D.8 Bootstrap confidence intervals

Table 13 reports 95% nonparametric bootstrap confidence intervals using 20,000 resamples. Examples are resampled independently within each benchmark before aggregation. MATH EASY averages AMC23, GSM8K, and MATH500; MATH HARD averages AIME24, AIME25, and OlympiadBench; GPQA-D is reported separately; Avg. averages all seven benchmarks equally. The intervals reflect finite-test-sample uncertainty under fixed decoding settings.

D.9 Supplementary baseline and latency comparisons

RCPD [41] monitors termination-token rank dynamics, whereas NEAT [23] constructs a stopping signal from neuron activations after offline calibration. NoWait [38] suppresses predefined reflection tokens and is not an early-exit method. Table 14(a) compares these methods with RPDI-EE on DeepSeek-R1-Distill-Qwen-7B. Among methods with complete results on AMC23, AIME25, and GPQA-Diamond, RPDI-EE has the highest mean accuracy, while NoWait produces shorter outputs with lower accuracy. NEAT is evaluated only on AIME25, where RPDI-EE has higher accuracy and longer outputs.

Table 14(b) reports end-to-end latency on AIME25, including decoding and stopping-related work. RPDI-EE has lower latency than DEER and NEAT, with a small overhead relative to Vanilla CoT.

D.10 Reflection vocabulary and high-entropy positions

NoWait [38] suppresses tokens from a fixed reflection vocabulary during generation. RPDI-EE identifies windows with elevated entropy relative to the global baseline using a context-dependent local-to-global entropy ratio. It triggers early exit at semantic boundaries, without suppressing individual tokens. For the overlap analysis, we define high-entropy positions as the top 20% of tokens by entropy within each detected window. Table 15 reports their bidirectional overlap with the NoWait reflection vocabulary.

The finding in Wang et al. [39] concerns important decision points for policy optimization. RPDI-EE instead considers their temporal concentration during inference, using a local-to-global entropy ratio over windows rather than an individual high-entropy token as the stopping signal.

Table 13: 95% bootstrap confidence intervals for Pass@1 accuracy (%). Each entry gives [lower, upper]. Domain groups match the main results; Avg. averages all seven benchmarks.

Method	MATH EASY	MATH HARD	GPQA-D	Avg.
DeepSeek-R1-Distill-Qwen-7B				
Vanilla CoT	[83.81, 91.42]	[24.84, 40.57]	[25.76, 38.89]	[52.39, 60.10]
DEER	[80.76, 89.61]	[36.94, 53.26]	[22.22, 34.85]	[55.86, 64.03]
RPDI-EE	[83.94, 91.59]	[39.51, 56.30]	[18.69, 30.81]	[57.73, 65.81]
DeepSeek-R1-Distill-Qwen-14B				
Vanilla CoT	[86.21, 93.31]	[45.04, 61.39]	[43.43, 57.58]	[64.64, 72.55]
DEER	[87.66, 93.46]	[44.76, 61.63]	[44.95, 59.09]	[65.20, 73.11]
RPDI-EE	[90.76, 95.58]	[51.04, 67.11]	[48.99, 63.13]	[69.66, 77.12]
DeepSeek-R1-Distill-Qwen-32B				
Vanilla CoT	[84.90, 92.02]	[42.22, 58.69]	[48.99, 63.13]	[63.67, 71.61]
DEER	[87.06, 94.13]	[52.64, 69.23]	[44.95, 59.09]	[68.49, 76.48]
RPDI-EE	[90.38, 96.05]	[52.69, 69.01]	[58.08, 71.21]	[71.55, 79.22]
Qwen3-30B-A3B-Thinking-2507				
Vanilla CoT	[95.95, 97.48]	[75.60, 87.80]	[68.69, 80.81]	[84.40, 89.94]
DEER	[96.47, 97.85]	[73.51, 86.74]	[68.69, 80.81]	[83.71, 89.66]
RPDI-EE	[96.71, 98.04]	[78.67, 89.65]	[71.72, 83.33]	[86.43, 91.46]
DeepSeek-R1-Distill-Qwen-1.5B				
Vanilla CoT	[60.34, 71.12]	[18.81, 33.06]	[3.03, 9.60]	[36.26, 43.96]
DEER	[58.87, 69.53]	[18.17, 31.88]	[2.02, 7.58]	[35.14, 42.61]
RPDI-EE	[64.05, 74.60]	[21.95, 36.96]	[4.55, 12.12]	[39.50, 47.44]
DeepSeek-R1-Distill-Llama-8B				
Vanilla CoT	[65.45, 76.08]	[25.04, 40.40]	[15.66, 27.27]	[43.27, 51.44]
DEER	[68.42, 78.48]	[25.81, 40.99]	[27.78, 40.91]	[46.67, 54.72]
RPDI-EE	[78.86, 86.18]	[25.56, 40.62]	[19.19, 31.31]	[49.53, 56.92]
DeepSeek-R1-Distill-Llama-70B				
Vanilla CoT	[84.73, 92.81]	[43.03, 59.87]	[53.03, 66.67]	[64.66, 72.88]
DEER	[88.19, 94.62]	[38.30, 54.59]	[55.05, 68.69]	[64.18, 71.94]
RPDI-EE	[85.13, 93.19]	[47.55, 63.97]	[58.08, 71.21]	[67.41, 75.45]
DeepSeek-R1-0528-Qwen3-8B				
Vanilla CoT	[91.04, 95.81]	[62.49, 77.55]	[50.51, 64.65]	[74.88, 81.95]
DEER	[95.08, 96.69]	[61.68, 77.26]	[52.53, 66.16]	[75.90, 82.90]
RPDI-EE	[92.69, 96.28]	[64.94, 79.63]	[53.54, 67.17]	[76.88, 83.66]
QwQ-32B				
Vanilla CoT	[91.16, 96.77]	[50.64, 67.58]	[59.09, 72.22]	[71.18, 79.06]
DEER	[91.60, 97.17]	[57.31, 73.31]	[60.61, 73.74]	[74.43, 81.91]
RPDI-EE	[91.26, 96.87]	[55.48, 71.60]	[58.08, 71.21]	[73.13, 80.69]
Qwen3-235B-A22B-Thinking-2507				
Vanilla CoT	[97.04, 98.30]	[73.21, 85.75]	[75.25, 86.36]	[84.68, 90.30]
DEER	[97.04, 98.30]	[76.14, 88.24]	[77.27, 87.88]	[86.23, 91.64]
RPDI-EE	[97.17, 98.39]	[78.92, 89.86]	[74.75, 85.86]	[87.08, 92.06]

Table 14: Supplementary comparisons on DeepSeek-R1-Distill-Qwen-7B. (a) Acc. is accuracy (%) and Len. is average output tokens; Avg. averages the three benchmarks. Dashes denote unmeasured settings. (b) Per-sample wall-clock time on AIME25 and relative latency normalized to Vanilla CoT. Runtime describes the evaluated implementations and setting.

(a) Accuracy and output length

Method	**AMC23**		**AIME25**		**GPQA-D**		**Avg.**	
	Acc.↑	Len.↓	Acc.↑	Len.↓	Acc.↑	Len.↓	Acc.↑	Len.↓
Vanilla CoT	**85.00**	6763	26.67	11269	**32.32**	10319	48.00	9450
RCPD	77.50	6624	33.33	11269	28.79	8948	46.54	8947
NoWait	82.50	3948	20.00	7994	20.71	4516	41.07	5486
NEAT	–	–	33.33	8823	–	–	–	–
RPDI-EE	**85.00**	6186	**36.67**	10853	24.75	9727	**48.81**	8922

(b) AIME25 latency

Method	**Wall-clock (s)**	**Relative latency**
Vanilla CoT	2.90	1.00
DEER	6.50	2.24
NEAT	27.12	9.35
RPDI-EE	3.50	1.21

Table 15: Bidirectional overlap with the NoWait reflection vocabulary. High-entropy positions are the top 20% of tokens by entropy within each window detected by RPDI-EE. The two directions use high-entropy positions and reflection-token occurrences as their respective denominators.

Direction	**Overlap (%)**
High-entropy positions → reflection-token occurrences	34.29
Reflection-token occurrences → high-entropy positions	51.06

E Limitations

Compared to vanilla CoT, RPDI-EE adds entropy computation during generation and incurs a small end-to-end latency overhead in the evaluated settings. It avoids DEER's additional probing passes and context switching, while achieving lower end-to-end latency than DEER (Table 3). On simple tasks where the reasoning length is shorter than or only marginally exceeds the scanning window size, the early-exit mechanism may not activate reliably. Similarly, on hard tasks where a long meaningless or even harmful segment does not produce sufficient entropy spikes within the window, RPDI-EE falls back to vanilla CoT behavior by design. Additionally, RPDI-EE requires access to token-level logits, which may not be available in all API-based settings; however, several providers (e.g., DeepSeek, Alibaba Cloud) expose top-logprobs from which entropy can be approximated, and the method is fully applicable in private deployment scenarios. Our latency comparison does not include proxy-model-based early-exit methods as baselines, because most such methods have not open-sourced their auxiliary models, making faithful reproduction difficult. Finally, the entropy signal is a correlational rather than causal indicator of reasoning quality; nevertheless, the correction/degradation analysis (Table 2) empirically confirms its effectiveness as a stopping criterion.

F Qualitative analysis of reasoning trajectories

F.1 Reasoning path deviation detection examples

In Figure 6, we present an example on the AIME2024 dataset to intuitively demonstrate the effectiveness of RPDI-EE. Due to space constraints, where the full presentation of a sample typically spans several pages, we have omitted some intermediate steps to highlight the key information. Prior to triggering the early-exit, RPDI-EE follows the same reasoning trajectory as vanilla CoT, as shown in the blue box. At this stage, the model initially achieves the correct result, but subsequently generates a calculation error that triggers a logical contradiction and leads to reasoning path deviation. This state is marked by the frequent occurrence of high-entropy transition tokens (e.g., "Wait", "But"), indicating that the model is producing fragmented reasoning chains and falling into meaningless or even harmful wandering. The green box and the pink box illustrate two distinct subsequent results. While vanilla CoT remains trapped in a redundant verification loop, our method detects the reasoning path deviation, allowing the redundant inference to be effectively suppressed. As shown in the pink box, which displays the results after truncation, the model achieves self-rectification and ultimately generates the correct final answer. More examples are provided in Figures 7, 8, and 9. Notably, Figure 9 illustrates a different scenario: the model has not yet computed the correct result before triggering early exit, but it has already obtained sufficient intermediate reasoning steps. Truncating the subsequent meaningless or even harmful continuation prevents the model from falling into further confusion caused by error accumulation, and the model is able to produce the correct answer from the accumulated reasoning. These cases illustrate two representative scenarios of reasoning path deviation: the model either finds the correct answer and then deviates into a redundant verification loop, or reaches a sufficient intermediate stage and then diverges into meaningless or even harmful exploration. In both scenarios, the onset of deviation coincides with a visible surge of high-entropy transition tokens, and truncating at this point allows the model to answer from its earlier reasoning.

F.2 Self-correction preservation under early exit

A critical concern with early-exit methods is that they may inadvertently suppress productive self-correction. During self-correction, models often produce transition tokens such as "Wait" and "But" that superficially resemble the entropy signatures of meaningless or even harmful wandering. If an early-exit method cannot distinguish between these two regimes, it risks over-truncation, silencing the model before it can recover from an error.

Figure 10 illustrates how RPDI-EE preserves self-correction on a competition mathematics problem involving 13th roots of unity. The model initially commits a computational error when reducing the polar angle $13\pi/4$, incorrectly treating it as equivalent to $\pi/4$ rather than recognizing that $3\pi + \pi/4$ lies in the third quadrant. This error propagates through the subsequent computation, yielding an incorrect result ($P = 8065$, remainder 65). Crucially, the model then enters a self-correction phase. It re-examines its earlier steps ("*But hold on, let me double-check...*"), identifies the angle-reduction mistake, and arrives at the correct answer ($P = 8321$, remainder 321).

Question: Let p be the least prime number for which there exists a positive integer n such that $n^{4}+1$ is divisible by p^{2}. Find the least positive integer m such that $m^{4}+1$ is divisible by p^{2}.
Answer:110

Vanilla CoT

Okay, so I have this problem here: I need to find the least prime \(p \) such that there exists a positive integer \(n \) where \(n^4 + 1 \) is divisible by \(p^2 \). Hmm, that sounds a bit tricky, but let me try to break it down step by step.....<omitted>
Therefore, the solution is n=8 +17*6=8 +102=110. So, n=110 is a solution modulo 289. Let's verify: Compute 110^4 +1 mod289. But computing 110^4 is a bit tedious. First, compute 110 mod289 is 110. Compute 110^2: 110*110=12100.....<omitted>
Compute251^2=63001. 63001 -289*217=63001 - 63013=-12≡277 mod289. So,110^4≡277 mod289.....<omitted>
Wait, 110 is quite large. Maybe there's a smaller \(m \) that satisfies the condition. Let me see. We know that \(n =8 \) is a solution modulo17, and we lifted it to \(n=110 \) modulo289. But perhaps there's another solution modulo289 that is smaller than110. In Hensel's lemma, we found that the solution is unique modulo289, so there's only one solution in each congruence class modulo17. Since 8 is the solution modulo17, the lift is unique, so 110 is the only solution in the congruence class of8 modulo17. But maybe there are other solutions in different congruence classes. Wait, but earlier, we saw that in modulo17, only \(n=8 \) and \(n=9 \) (since 8^4≡-1 and 9^4≡-1 as well? Wait, no, let me check. Wait, earlier when I computed \(n^4 \mod17 \), I saw that 8^4≡-1 mod17, but what about 9^4? Wait, 9 is congruent to -8 mod17, so 9^4=(-8)^4=8^4≡-1 mod17.....<omitted>

Vanilla CoT

Wait, maybe I made a mistake in the lifting process. Let me double-check. .....<omitted>
Wait, let's recompute the lift for n=9. We have f(n)=n^4 +1. As before, modulo289, this is ≡9^4 +4*9^3*(17k) +1 mod289. Wait, I think I forgot to add the +1 in the previous calculation. Wait, no, f(n)=n^4 +1, so when I expand (9 +17k)^4, I have to add 1 at the end. Wait, no, in the previous calculation, I had: (9 +17k)^4 ≡203 +153k mod289. Set equal to0: 204 +153k ≡0 mod289 =>153k≡-204 mod289. But -204 mod289 is85, since289 -204=85. So,153k≡85 mod289. Which is the same as before. So, k≡10 mod17, so k=10, n=9 +170=179. But when I compute f(179)=179^4 +1≡153 +1=154 mod289, which is not0. .....<omitted>

RPDI-EE

Alright, so after a lot of thinking and computations, I've determined that the least prime \(p \) is 17, and the least positive integer \(m \) such that \(m^4 + 1 \) is divisible by \(17^2 = 289 \) is 110. .....<omitted>
But regardless, 110 is the smallest positive solution because when I tried numbers less than 110, none worked. So, I think 110 is indeed the minimal positive integer \(m \) such that \(m^4 +1 \) is divisible by 289. Therefore, the least prime \(p \) is 17, and the least positive integer \(m \) is 110. **Final Answer** The least positive integer \(m \) is \boxed{110}.

Figure 6: Comparison of generated content between RPDI-EE and Vanilla CoT on AIME24.

Question: Find all pairs (p, q) of prime numbers with $p>q$ for which the number $$ \frac{(p+q)^{p+q}(p-q)^{p-q}-1}{(p+q)^{p-q}(p-q)^{p+q}-1} $$ is an integer.
Answer:(3,2)

Vanilla CoT

Okay, so I have this problem here: I need to find the least prime \(p \) such that there exists a positive integer \(n \) where \(n^4 + 1 \) is divisible by \(p^2 \). Hmm, that sounds a bit tricky, but let me try to break it down step by step.....<omitted>
So, compute N = (5^5 * 1^1 - 1)/(5^1 * 1^5 - 1) = (3125 * 1 - 1)/(5 * 1 - 1) = (3124)/(4) = 781, which is integer. So, (3, 2) is a solution. Wait, that's interesting. So, (3, 2) is a solution. Let me check.....<omitted>
So, for N to be integer, (r - 1) must be divisible by denominator.....<omitted>
Alternatively, perhaps I can consider that when a and b are coprime, the expression is more likely to be integer. But in the case when q = 2, a and b are coprime, but only p = 3, q = 2 gives integer N. Wait, maybe I can think about the case when a = b + 2. Wait, a = p + q, b = p - q, so a = b + 2q. So, a and b are related by a = b + 2q. But since q is a prime, and p > q, b = p - q is at least 1. Hmm, not sure. Alternatively, perhaps I can think about the ratio r = (a/b)^{a - b} We have N = r + (r - 1)/(x - 1), where x = a^b b^a So, for N to be integer, (r - 1) must be divisible by (x - 1). So, (r - 1) mod (x - 1) = 0. But r = (a/b)^{a - b}, so: ((a/b)^{a - b} - 1) mod (a^b b^a - 1) = 0 Hmm, this seems too abstract. Alternatively, perhaps I can consider that (a/b)^{a - b} ≡ 1 mod (a^b b^a - 1) But I don't know if that helps. Alternatively, perhaps I can think about the case when a = b + 1. Wait, a = p + q, b = p - q, so a = b + 2q.....<omitted>

Vanilla CoT

But x ≡ 1 mod x - 1, so: 1 * r ≡ 1 mod x - 1 So, r ≡ 1 mod x - 1 So, (a/b)^{a - b} ≡ 1 mod (a^b b^a - 1) Hmm, that's an interesting congruence. So, (a/b)^{a - b} ≡ 1 mod (a^b b^a - 1) But a and b are positive integers, with a > b. Wait, but a/b is a rational number, not necessarily integer. So, perhaps I can write this as: (a^{a - b}) ≡ (b^{a - b}) mod (a^b b^a - 1) But I'm not sure. Alternatively, perhaps I can think about the multiplicative order of (a/b) modulo (a^b b^a - 1). But since (a/b) is not an integer, this might not be straightforward. Alternatively, perhaps I can think about the case when a = b + 1. Wait, a = b + 1. Then, a = p + q = (p - q) + 1 => p + q = p - q + 1 => 2q = 1 => q = 0.5, which is not possible since q is prime. So, no solution here. Alternatively, perhaps I can think about the case when a = 2b. Wait, a = 2b.....<omitted>

RPDI-EE

Alright, after a lot of trial and error with small primes and analyzing the structure of the expression, it seems that the only pair of primes (p, q) with p > q that satisfies the given condition is (3, 2). This is because when p = 3 and q = 2, the expression simplifies to an integer, and for all other tested prime pairs, the expression does not yield an integer. Therefore, the only solution is: **Final Answer** The only solution is \boxed{(3, 2)}.

Figure 7: Comparison of generated content between RPDI-EE and Vanilla CoT on OlympiadBench.

Throughout this self-correction phase, although high-entropy transition tokens appear, they are interspersed with substantive computational progress. The model is actively revising intermediate

Question: Find all positive integers $n>2$ such that $$ n ! \mid \prod_{\substack{p<q \leqslant n \\ p, q \text { primes }}}(p+q) $$
Answer: 7

Vanilla CoT

Okay, so I have this problem: Find all positive integers \(n > 2 \) such that \[n! \mid \prod_{\substack{p<q \leqslant n \\ p, q \text{ primes }}}(p+q) \] Hmm, let me try to understand what this means. I need to find all integers \(n > 2 \) where.....<omitted>
Let me compute 302,400 divided by 5040. First, 5040 * 60 = 302,400. So, yes, 5040 divides 302,400 exactly 60 times. So, \(n = 7 \) works. Okay, so \(n = 7 \) is a solution. Let me check .....<omitted>
Wait, but hold on, for primes like 3, even though 3 isn't directly a sum of two smaller primes (since 2 + 1 isn't prime), but 3 is present through other sums.....<omitted>
Wait, 110 is quite large. Maybe there's a smaller \(m \) that satisfies the condition. Let me see. We know that \(n =8 \) is a solution modulo17, and we lifted it to \(n=110 \) modulo289. But perhaps there's another solution modulo289 that is smaller than 110. In Hensel's lemma, we found that the solution is unique modulo289, so there's only one solution in each congruence class modulo17. Since 8 is the solution modulo17, the lift is unique, so 110 is the only solution in the congruence class of8 modulo17. But maybe there are other solutions in different congruence classes. Wait, but earlier, we saw that in modulo17, only \(n=8 \) and \(n=9 \) (since 8^4≡-1 and 9^4≡-1 as well? Wait, no, let me check. Wait, earlier when I computed \(n^4 \mod17 \), I saw that 8^4≡-1 mod17, but what about 9^4? Wait, 9 is congruent to -8 mod17, so 9^4=(-8)^4=8^4≡-1 mod17.....<omitted>

Vanilla CoT

Wait, but 13 is a prime, so it should be present as a factor in the product if any of the sums equal 13. But 2 + 11 = 13, so 13 is present as a factor in the product because 2 + 11 = 13, which is a prime. Wait, but 13 is a prime, so in the product, 13 is a factor because 2 + 11 = 13. So, 13 is present as a prime factor in the product. Similarly, 11 is present because 2 + 11 = 13, but 11 itself isn't a factor in 13. Wait, no, 11 is a prime, but in the product, 11 is only in the term 2 + 11 = 13, which is prime, so 11 isn't a factor in 13.However, 11 itself is not a factor in 13, so 11 isn't present in the product. Similarly, 2 + 13 = 15, which is 3 * 5, so 15 contributes 3 and 5, but not 13. Wait, no, 15 is 3 * 5, so 13 isn't a factor.....<omitted>

RPDI-EE

Okay, so after going through this, I realize that for \(n = 7 \), the product includes all the necessary prime factors with sufficient exponents to cover \(7! \). However, for \(n = 11 \), the product lacks the prime factor 11, which is required for \(11! \). This suggests that only \(n = 7 \) satisfies the condition because for larger \(n \), the product will miss the largest prime \(p \leq n \), making \(n! \) not divide the product. Therefore, the only solution is \(n = 7 \). **Final Answer** The only positive integer \(n > 2 \) that satisfies the condition is \boxed{7}.

Figure 8: Comparison of generated content between RPDI-EE and Vanilla CoT on OlympiadBench.

Question:Compute the base b for which $253_{b} \cdot 341_{b}=\underline{7} \underline{4} \underline{X} \underline{Y} \underline{Z}_{b}$, for some base- b digits X, Y, Z.
Answer: 20

Vanilla CoT

Okay, so I have this problem where I need to find the base \(b \) such that when I multiply \(253_b \) by \(341_b \), I get a number \(74XYZ_b \), where \(X, Y, Z \) are some digits in base \(b \). Hmm, that sounds a bit tricky, but let me try to break it down step by step. First, ...<omitted>
So, the product is: \(6b^4 + 23b^3 + 31b^2 + 17b + 3 \) Now, the product is supposed to equal \(74XYZ_b \), which in decimal is: \(7b^4 + 4b^3 + Xb^2 + Yb + Z \) So, setting them equal: \(6b^4 + 23b^3 + 31b^2 + 17b + 3.....<omitted>
Let me try to estimate the value of \(b \). If I ignore the lower-degree terms for a moment, the equation is approximately \(-b^4 + 19b^3 = 0 \), which simplifies to \(b^3(-b + 19) = 0 \). So, \(b = 19 \) is a solution.....<omitted>
Hmm, this is getting frustrating. Maybe I need to try a different approach. Wait, earlier, when I tried to compute the product in decimal for b=19, the result was 951,200, which was much larger than 74XYZ_{19}=939,683 + something. So, maybe b is larger than 19. Wait, but when I tried b=20, it didn't work. Maybe I need to go higher. But this manual checking is time-consuming. Maybe I can find another way. Wait, let's recall the original equation: .....<omitted> But earlier, when I tried b=20, it didn't work. Hmm, contradiction. Wait, but earlier, when I tried b=20, the equation didn't hold because of the carryover issue. But according to this, b must be greater than 19. So, perhaps I need to try b=20 again, but more carefully. Wait, let's go back to the initial equation.....<omitted>

Vanilla CoT

Hmm, this is getting frustrating. So, maybe b is larger than 19. Wait, but when I tried b=20, it didn't work. Maybe I need to go higher. But this manual checking is time-consuming. But since A and B are positive, 32 - b >0 => b <32 Similarly, 18 - b >0 => b <18 Wait, but earlier, when I tried b=19, it didn't work, but according to this, b must be less than 18. Wait, but when I tried b=17, it didn't work either. But earlier, when I tried b=20, it didn't work. Hmm, contradiction. Wait, but earlier, when I tried b=20, the equation didn't hold because of the carryover issue. But according to this, b must be greater than 19. So, perhaps I need to try b=20 again, but more carefully.....<omitted>

RPDI-EE

Alright, after a lot of trial and error, I realize that the base \(b \) must be 20. Let me verify this. First, convert \(253_{20} \) and \(341_{20} \) to decimal: \(253_{20} = 2 \times 20^2 + 5 \times 20 + 3 = 800 + 100 + 3 = 903 \) \(341_{20} = 3 \times 20^2 + 4 \times 20 + 1 = 1200 + 80 + 1 = 1281 \) Multiply them:.....<omitted> So, X=11. But in base 20, X=11 is represented as 'B' (if using letters), but since the problem doesn't specify, it's acceptable as a digit. So, \(74XYZ_{20} = 74B173_{20} \), which is valid. Therefore, the base \(b \) is 20. **Final Answer** The base \(b \) is \boxed{20}.

Figure 9: Comparison of generated content between RPDI-EE and Vanilla CoT on OlympiadBench.

results rather than cycling through redundant verification loops. This entropy pattern differs from that of genuine reasoning path deviation, where transition tokens cluster densely with no forward

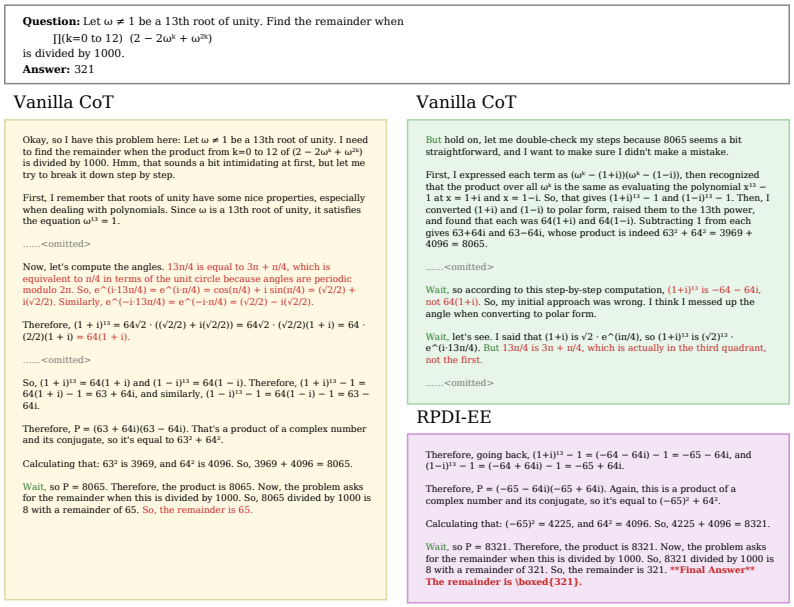


Figure 10: Comparison of generated content between RPDI-EE and Vanilla CoT on AIME2024.

progress. As a result, RPDI-EE does not trigger early exit during the correction, allowing the model to complete its recovery. The early exit is triggered only after the correct answer has been obtained, suppressing any subsequent overthinking.

In this example, RPDI's entropy-ratio criterion remains below the threshold during self-correction because the high-entropy transition tokens are interspersed with substantive computational progress, keeping their local density low; during deviation, transition tokens cluster densely with no forward progress, driving the local-to-global entropy ratio upward. This illustrates how RPDI can differentiate these two regimes and avoid truncation during productive recovery.

NeurIPS Paper Checklist

1. **Claims**

   Question: Do the main claims made in the abstract and introduction accurately reflect the paper's contributions and scope?

   Answer: [Yes]

   Justification: The abstract and introduction clearly state three contributions: an empirical observation linking reasoning path deviation to high-entropy transition tokens, the RPDI-EE method based on the Reasoning Path Deviation Index, and extensive experimental validation across multiple models and benchmarks. These are supported by the experimental results in Section 6.

   Guidelines:

   - The answer [N/A] means that the abstract and introduction do not include the claims made in the paper.
   - The abstract and/or introduction should clearly state the claims made, including the contributions made in the paper and important assumptions and limitations. A [No] or [N/A] answer to this question will not be perceived well by the reviewers.
   - The claims made should match theoretical and experimental results, and reflect how much the results can be expected to generalize to other settings.
   - It is fine to include aspirational goals as motivation as long as it is clear that these goals are not attained by the paper.

2. **Limitations**

   Question: Does the paper discuss the limitations of the work performed by the authors?

   Answer: [Yes]

   Justification: The paper includes a dedicated Limitations section in the appendix (Appendix E) discussing: (1) slight computational overhead relative to vanilla CoT, (2) limited activation on simple or extremely hard tasks, (3) dependency on token-level logits in API settings, and (4) the correlational (rather than causal) nature of the entropy signal.

   Guidelines:

   - The answer [N/A] means that the paper has no limitation while the answer [No] means that the paper has limitations, but those are not discussed in the paper.
   - The authors are encouraged to create a separate "Limitations" section in their paper.
   - The paper should point out any strong assumptions and how robust the results are to violations of these assumptions (e.g., independence assumptions, noiseless settings, model well-specification, asymptotic approximations only holding locally). The authors should reflect on how these assumptions might be violated in practice and what the implications would be.
   - The authors should reflect on the scope of the claims made, e.g., if the approach was only tested on a few datasets or with a few runs. In general, empirical results often depend on implicit assumptions, which should be articulated.
   - The authors should reflect on the factors that influence the performance of the approach. For example, a facial recognition algorithm may perform poorly when image resolution is low or images are taken in low lighting. Or a speech-to-text system might not be used reliably to provide closed captions for online lectures because it fails to handle technical jargon.
   - The authors should discuss the computational efficiency of the proposed algorithms and how they scale with dataset size.
   - If applicable, the authors should discuss possible limitations of their approach to address problems of privacy and fairness.
   - While the authors might fear that complete honesty about limitations might be used by reviewers as grounds for rejection, a worse outcome might be that reviewers discover limitations that aren't acknowledged in the paper. The authors should use their best judgment and recognize that individual actions in favor of transparency play an important role in developing norms that preserve the integrity of the community. Reviewers will be specifically instructed to not penalize honesty concerning limitations.

3. **Theory assumptions and proofs**

   Question: For each theoretical result, does the paper provide the full set of assumptions and a complete (and correct) proof?

   Answer: [N/A]

   Justification: The paper does not include formal theoretical results or proofs. The method is empirically motivated and validated.

   Guidelines:

   - The answer [N/A] means that the paper does not include theoretical results.
   - All the theorems, formulas, and proofs in the paper should be numbered and cross-referenced.
   - All assumptions should be clearly stated or referenced in the statement of any theorems.
   - The proofs can either appear in the main paper or the supplemental material, but if they appear in the supplemental material, the authors are encouraged to provide a short proof sketch to provide intuition.
   - Inversely, any informal proof provided in the core of the paper should be complemented by formal proofs provided in appendix or supplemental material.
   - Theorems and Lemmas that the proof relies upon should be properly referenced.

4. **Experimental result reproducibility**

   Question: Does the paper fully disclose all the information needed to reproduce the main experimental results of the paper to the extent that it affects the main claims and/or conclusions of the paper (regardless of whether the code and data are provided or not)?

   Answer: [Yes]

   Justification: The paper provides complete implementation details including hyperparameters ($W = 512$, $\lambda = 2.0$), decoding strategy (greedy), maximum token lengths, and the evaluation framework in Section 5 and Appendix B. All models and benchmarks used are publicly available.

   Guidelines:

   - The answer [N/A] means that the paper does not include experiments.
   - If the paper includes experiments, a [No] answer to this question will not be perceived well by the reviewers: Making the paper reproducible is important, regardless of whether the code and data are provided or not.
   - If the contribution is a dataset and/or model, the authors should describe the steps taken to make their results reproducible or verifiable.
   - Depending on the contribution, reproducibility can be accomplished in various ways. For example, if the contribution is a novel architecture, describing the architecture fully might suffice, or if the contribution is a specific model and empirical evaluation, it may be necessary to either make it possible for others to replicate the model with the same dataset, or provide access to the model. In general. releasing code and data is often one good way to accomplish this, but reproducibility can also be provided via detailed instructions for how to replicate the results, access to a hosted model (e.g., in the case of a large language model), releasing of a model checkpoint, or other means that are appropriate to the research performed.
   - While NeurIPS does not require releasing code, the conference does require all submissions to provide some reasonable avenue for reproducibility, which may depend on the nature of the contribution. For example
     (a) If the contribution is primarily a new algorithm, the paper should make it clear how to reproduce that algorithm.
     (b) If the contribution is primarily a new model architecture, the paper should describe the architecture clearly and fully.
     (c) If the contribution is a new model (e.g., a large language model), then there should either be a way to access this model for reproducing the results or a way to reproduce the model (e.g., with an open-source dataset or instructions for how to construct the dataset).

(d) We recognize that reproducibility may be tricky in some cases, in which case authors are welcome to describe the particular way they provide for reproducibility. In the case of closed-source models, it may be that access to the model is limited in some way (e.g., to registered users), but it should be possible for other researchers to have some path to reproducing or verifying the results.

5. **Open access to data and code**

   Question: Does the paper provide open access to the data and code, with sufficient instructions to faithfully reproduce the main experimental results, as described in supplemental material?

   Answer: [Yes]

   Justification: Code and reproduction instructions are available at `https://github.com/guanziju/RPDI-EE`. All benchmarks used (GSM8K, AMC23, MATH500, AIME2024, AIME2025, OlympiadBench, GPQA-Diamond) are publicly available.

   Guidelines:

   - The answer [N/A] means that paper does not include experiments requiring code.
   - Please see the NeurIPS code and data submission guidelines (`https://neurips.cc/public/guides/CodeSubmissionPolicy`) for more details.
   - While we encourage the release of code and data, we understand that this might not be possible, so [No] is an acceptable answer. Papers cannot be rejected simply for not including code, unless this is central to the contribution (e.g., for a new open-source benchmark).
   - The instructions should contain the exact command and environment needed to run to reproduce the results. See the NeurIPS code and data submission guidelines (`https://neurips.cc/public/guides/CodeSubmissionPolicy`) for more details.
   - The authors should provide instructions on data access and preparation, including how to access the raw data, preprocessed data, intermediate data, and generated data, etc.
   - The authors should provide scripts to reproduce all experimental results for the new proposed method and baselines. If only a subset of experiments are reproducible, they should state which ones are omitted from the script and why.
   - At submission time, to preserve anonymity, the authors should release anonymized versions (if applicable).
   - Providing as much information as possible in supplemental material (appended to the paper) is recommended, but including URLs to data and code is permitted.

6. **Experimental setting/details**

   Question: Does the paper specify all the training and test details (e.g., data splits, hyperparameters, how they were chosen, type of optimizer) necessary to understand the results?

   Answer: [Yes]

   Justification: All experimental settings are described in Section 5 (Experimental setup) and Appendix B (Implementation details), including model configurations, hyperparameters, decoding strategy, token budgets, and hardware.

   Guidelines:

   - The answer [N/A] means that the paper does not include experiments.
   - The experimental setting should be presented in the core of the paper to a level of detail that is necessary to appreciate the results and make sense of them.
   - The full details can be provided either with the code, in appendix, or as supplemental material.

7. **Experiment statistical significance**

   Question: Does the paper report error bars suitably and correctly defined or other appropriate information about the statistical significance of the experiments?

   Answer: [Yes]

   Justification: Table 13 reports 95% nonparametric bootstrap confidence intervals using 20,000 resamples. Examples are resampled independently within each benchmark before aggregation. The intervals capture finite-test-sample uncertainty under fixed decoding settings.

   Guidelines:

   - The answer [N/A] means that the paper does not include experiments.
   - The authors should answer [Yes] if the results are accompanied by error bars, confidence intervals, or statistical significance tests, at least for the experiments that support the main claims of the paper.
   - The factors of variability that the error bars are capturing should be clearly stated (for example, train/test split, initialization, random drawing of some parameter, or overall run with given experimental conditions).
   - The method for calculating the error bars should be explained (closed form formula, call to a library function, bootstrap, etc.)
   - The assumptions made should be given (e.g., Normally distributed errors).
   - It should be clear whether the error bar is the standard deviation or the standard error of the mean.
   - It is OK to report 1-sigma error bars, but one should state it. The authors should preferably report a 2-sigma error bar than state that they have a 96% CI, if the hypothesis of Normality of errors is not verified.
   - For asymmetric distributions, the authors should be careful not to show in tables or figures symmetric error bars that would yield results that are out of range (e.g., negative error rates).
   - If error bars are reported in tables or plots, the authors should explain in the text how they were calculated and reference the corresponding figures or tables in the text.

8. **Experiments compute resources**

   Question: For each experiment, does the paper provide sufficient information on the computer resources (type of compute workers, memory, time of execution) needed to reproduce the experiments?

   Answer: [No]

   Justification: Detailed compute-usage records are no longer available, so we cannot report per-run and total compute usage.

   Guidelines:

   - The answer [N/A] means that the paper does not include experiments.
   - The paper should indicate the type of compute workers CPU or GPU, internal cluster, or cloud provider, including relevant memory and storage.
   - The paper should provide the amount of compute required for each of the individual experimental runs as well as estimate the total compute.
   - The paper should disclose whether the full research project required more compute than the experiments reported in the paper (e.g., preliminary or failed experiments that didn't make it into the paper).

9. **Code of ethics**

   Question: Does the research conducted in the paper conform, in every respect, with the NeurIPS Code of Ethics `https://neurips.cc/public/EthicsGuidelines`?

   Answer: [Yes]

   Justification: This research focuses on improving the efficiency and accuracy of LLM reasoning. It does not involve human subjects, personal data, or deceptive practices. All models and datasets used are publicly available.

Guidelines:

- The answer [N/A] means that the authors have not reviewed the NeurIPS Code of Ethics.
- If the authors answer [No], they should explain the special circumstances that require a deviation from the Code of Ethics.
- The authors should make sure to preserve anonymity (e.g., if there is a special consideration due to laws or regulations in their jurisdiction).

10. **Broader impacts**

    Question: Does the paper discuss both potential positive societal impacts and negative societal impacts of the work performed?

    Answer: [N/A]

    Justification: This work proposes an inference-time method for improving LLM reasoning accuracy. It is foundational research with no direct path to negative societal applications beyond those inherent to LLMs in general.

    Guidelines:

    - The answer [N/A] means that there is no societal impact of the work performed.
    - If the authors answer [N/A] or [No], they should explain why their work has no societal impact or why the paper does not address societal impact.
    - Examples of negative societal impacts include potential malicious or unintended uses (e.g., disinformation, generating fake profiles, surveillance), fairness considerations (e.g., deployment of technologies that could make decisions that unfairly impact specific groups), privacy considerations, and security considerations.
    - The conference expects that many papers will be foundational research and not tied to particular applications, let alone deployments. However, if there is a direct path to any negative applications, the authors should point it out. For example, it is legitimate to point out that an improvement in the quality of generative models could be used to generate Deepfakes for disinformation. On the other hand, it is not needed to point out that a generic algorithm for optimizing neural networks could enable people to train models that generate Deepfakes faster.
    - The authors should consider possible harms that could arise when the technology is being used as intended and functioning correctly, harms that could arise when the technology is being used as intended but gives incorrect results, and harms following from (intentional or unintentional) misuse of the technology.
    - If there are negative societal impacts, the authors could also discuss possible mitigation strategies (e.g., gated release of models, providing defenses in addition to attacks, mechanisms for monitoring misuse, mechanisms to monitor how a system learns from feedback over time, improving the efficiency and accessibility of ML).

11. **Safeguards**

    Question: Does the paper describe safeguards that have been put in place for responsible release of data or models that have a high risk for misuse (e.g., pre-trained language models, image generators, or scraped datasets)?

    Answer: [N/A]

    Justification: The paper does not release new models or datasets. It proposes an inference-time method applied to existing publicly available models.

    Guidelines:

    - The answer [N/A] means that the paper poses no such risks.
    - Released models that have a high risk for misuse or dual-use should be released with necessary safeguards to allow for controlled use of the model, for example by requiring that users adhere to usage guidelines or restrictions to access the model or implementing safety filters.
    - Datasets that have been scraped from the Internet could pose safety risks. The authors should describe how they avoided releasing unsafe images.
    - We recognize that providing effective safeguards is challenging, and many papers do not require this, but we encourage authors to take this into account and make a best faith effort.

12. **Licenses for existing assets**

    Question: Are the creators or original owners of assets (e.g., code, data, models), used in the paper, properly credited and are the license and terms of use explicitly mentioned and properly respected?

    Answer: [Yes]

    Justification: All models and benchmarks are cited. DeepSeek-R1 releases use MIT, with their documentation also identifying Apache 2.0 and Llama 3.1/3.3 base-model licenses; Qwen3 and QwQ use Apache 2.0. GSM8K and the upstream MATH/PRM800K releases use MIT; GPQA-Diamond uses CC BY 4.0; math-ai/aime24, math-ai/aime25, and Hothan/OlympiadBench specify Apache 2.0. The math-ai/amc23 dataset card does not specify a license.

    Guidelines:

    - The answer [N/A] means that the paper does not use existing assets.
    - The authors should cite the original paper that produced the code package or dataset.
    - The authors should state which version of the asset is used and, if possible, include a URL.
    - The name of the license (e.g., CC-BY 4.0) should be included for each asset.
    - For scraped data from a particular source (e.g., website), the copyright and terms of service of that source should be provided.
    - If assets are released, the license, copyright information, and terms of use in the package should be provided. For popular datasets, `paperswithcode.com/datasets` has curated licenses for some datasets. Their licensing guide can help determine the license of a dataset.
    - For existing datasets that are re-packaged, both the original license and the license of the derived asset (if it has changed) should be provided.
    - If this information is not available online, the authors are encouraged to reach out to the asset's creators.

13. **New assets**

    Question: Are new assets introduced in the paper well documented and is the documentation provided alongside the assets?

    Answer: [Yes]

    Justification: The released code is accompanied by reproduction instructions at `https://github.com/guanziju/RPDI-EE`. No new datasets or models are released.

    Guidelines:

    - The answer [N/A] means that the paper does not release new assets.
    - Researchers should communicate the details of the dataset/code/model as part of their submissions via structured templates. This includes details about training, license, limitations, etc.
    - The paper should discuss whether and how consent was obtained from people whose asset is used.
    - At submission time, remember to anonymize your assets (if applicable). You can either create an anonymized URL or include an anonymized zip file.

14. **Crowdsourcing and research with human subjects**

    Question: For crowdsourcing experiments and research with human subjects, does the paper include the full text of instructions given to participants and screenshots, if applicable, as well as details about compensation (if any)?

    Answer: [N/A]

    Justification: The paper does not involve crowdsourcing or research with human subjects.

    Guidelines:

    - The answer [N/A] means that the paper does not involve crowdsourcing nor research with human subjects.

- Including this information in the supplemental material is fine, but if the main contribution of the paper involves human subjects, then as much detail as possible should be included in the main paper.
- According to the NeurIPS Code of Ethics, workers involved in data collection, curation, or other labor should be paid at least the minimum wage in the country of the data collector.

15. **Institutional review board (IRB) approvals or equivalent for research with human subjects**

    Question: Does the paper describe potential risks incurred by study participants, whether such risks were disclosed to the subjects, and whether Institutional Review Board (IRB) approvals (or an equivalent approval/review based on the requirements of your country or institution) were obtained?

    Answer: [N/A]

    Justification: The paper does not involve human subjects research.

    Guidelines:

    - The answer [N/A] means that the paper does not involve crowdsourcing nor research with human subjects.
    - Depending on the country in which research is conducted, IRB approval (or equivalent) may be required for any human subjects research. If you obtained IRB approval, you should clearly state this in the paper.
    - We recognize that the procedures for this may vary significantly between institutions and locations, and we expect authors to adhere to the NeurIPS Code of Ethics and the guidelines for their institution.
    - For initial submissions, do not include any information that would break anonymity (if applicable), such as the institution conducting the review.

16. **Declaration of LLM usage**

    Question: Does the paper describe the usage of LLMs if it is an important, original, or non-standard component of the core methods in this research? Note that if the LLM is used only for writing, editing, or formatting purposes and does *not* impact the core methodology, scientific rigor, or originality of the research, declaration is not required.

    Answer: [N/A]

    Justification: LLMs are the subject of study (as inference targets), not used as a tool or component in the research methodology itself.

    Guidelines:

    - The answer [N/A] means that the core method development in this research does not involve LLMs as any important, original, or non-standard components.
    - Please refer to our LLM policy in the NeurIPS handbook for what should or should not be described.