The Zero-Step Thinking: An Empirical Study of Mode Selection as Harder Early Exit in Reasoning Models

Yuqiao Tan[1,2], **Shizhu He**[1,2]*, **Kang Liu**[1,2,3], **Jun Zhao**[1,2]
[1] The Key Laboratory of Cognition and Decision Intelligence for Complex Systems,
Institute of Automation, Chinese Academy of Sciences, Beijing, China
[2] School of Artificial Intelligence, University of Chinese Academy of Sciences, Beijing, China
[3] Shanghai Artificial Intelligence Laboratory
tanyuqiao2025@ia.ac.cn {shizhu.he, jzhao, kliu}@nlpr.ia.ac.cn

Abstract

Reasoning models have demonstrated exceptional performance in tasks such as mathematics and logical reasoning, primarily due to their ability to engage in step-by-step thinking during the reasoning process. However, this often leads to overthinking, resulting in unnecessary computational overhead. To address this issue, Mode Selection aims to automatically decide between Long-CoT (Chain-of-Thought) or Short-CoT by utilizing either a THINKING or NOTHINKING mode. Simultaneously, Early Exit determines the optimal stopping point during the iterative reasoning process. Both methods seek to reduce the computational burden. In this paper, we first identify Mode Selection as a more challenging variant of the Early Exit problem, as they share similar objectives but differ in decision timing. While Early Exit focuses on determining the best stopping point for concise reasoning at inference time, Mode Selection must make this decision at the beginning of the reasoning process, relying on pre-defined fake thoughts without engaging in an explicit reasoning process, referred to as zero-step thinking. Through empirical studies on nine baselines, we observe that prompt-based approaches often fail due to their limited classification capabilities when provided with minimal hand-crafted information. In contrast, approaches that leverage internal information generally perform better across most scenarios but still exhibit issues with stability. Our findings indicate that existing methods relying solely on the information provided by models are insufficient for effectively addressing Mode Selection in scenarios with limited information, highlighting the ongoing challenges of this task. Our code is available at `https://github.com/Trae1ounG/Zero_Step_Thinking`.

1 Introduction

Recent advances in large reasoning models (LRMs), such as DeepSeek-R1 [12], OpenAI o1 [14], QwQ [32], and others [11, 1, 31], have demonstrated significant progress in complex reasoning capabilities by increasing inference-time compute [23]. These models achieve success through long chain-of-thought (CoT)[36] processes, which involve behaviors such as exploration, self-reflection, and verification[18, 7, 8]. However, this strength can also become a limitation. Previous studies have shown that reasoning models often tend to overthink, exploring additional reasoning paths even after arriving at a correct answer [2, 27, 34].

The adaptive nature of human cognition commonly switches between System 1 and System 2 based on task difficulty. Inspired by this, recent research on **Adaptive Thinking** seeks to enable a single,

39th Conference on Neural Information Processing Systems (NeurIPS 2025) Workshop: Efficient Reasoning.

*Corresponding author

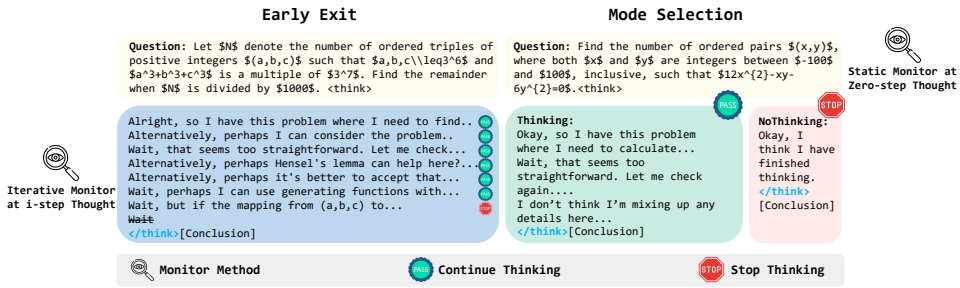


Figure 1: Illustration of **Early Exit** and **Mode Selection**. Early Exit employs an iterative monitor to decide whether to stop reasoning at the end of each step in the thought process. In contrast, Mode Selection operates prior to explicit reasoning, determining the optimal thinking strategy at zero-step. While both methods can leverage the same monitoring mechanisms, they differ in their timing: Early Exit monitors iteratively during the reasoning process, whereas Mode Selection monitors only once at the beginning to append `</think>` and terminate the reasoning process.

powerful reasoning model to operate in two distinct modes: its native long-CoT mode for complex problems and an efficient short-CoT mode for simpler tasks [46, 33, 20]. Many of these approaches utilize reinforcement learning to train reasoning models to think more concisely and switch modes adaptively [45, 5, 21].

Another efficient reasoning solution is **Early Exit**, which aims to determine the optimal stopping point in the reasoning process by truncating it to directly produce an answer [39, 15, 44]. This approach helps LRMs avoid redundant reasoning paths that may lead to overthinking and potentially performance degradation. Existing studies have demonstrated that reasoning models can inherently perform short-CoT in NoThinking mode by appending fake thoughts directly (e.g., `<think>` Okay, I think I have finished thinking. `</think>`), thereby significantly reducing token usage while maintaining strong performance [22, 16, 17, 47].

In this study, we focus on a specific subtype of Adaptive Thinking called **Mode Selection**, which determines the preferred prompt mode without relying on an explicit reasoning trajectory. This process can be regarded as zero-step thinking [19]. Interestingly, we find that Mode Selection is conceptually similar to Early Exit but differs in decision timing and is significantly more challenging. As shown in Figure 1, Early Exit operates iteratively by taking previously generated thinking tokens into account to make decisions at the end of each reasoning step. This iterative process not only delivers better performance but also reduces token usage due to choosing optimal stop points [39, 44, 10]. In contrast, Mode Selection relies solely on input tokens with human-crafted fake thoughts to determine the optimal mode at the outset of the reasoning process, making it a more difficult task due to the limited information available. Intuitively, both methods employ a monitoring mechanism $\text{Exit}(\cdot)$ to facilitate exit decisions. This observation raises our core research question: *Can existing well-performing Early Exit methods effectively address the harder Mode Selection problem?*

To better understand this question, we conduct empirical studies using several methods from Early Exit, categorized into two distinct types: *Prompt-based* and *Internal States-based* methods, which utilize different prompt templates and internal model information for Mode Selection respectively. We rigorously evaluate these methods across diverse reasoning benchmarks, including GSM8K [3], MATH-500 [13], AIME25 [4], and GPQA-D [25], covering both mathematical and scientific reasoning tasks. Our experiments reveal that prompt-based approaches often fail due to their limited classification capabilities when provided with minimal information. In contrast, internal states-based methods, which leverage internal model information, perform better in most scenarios but still struggle with stability. Through the analysis of various metrics, including ROC-AUC, Expected Calibration Error (ECE) [24], and Brier score [9], we observe that existing evaluation metrics are insufficient for fully assessing and explaining the underlying reasons behind the performance of different methods. This highlights that Mode Selection is a highly challenging task, requiring deeper investigation into the model's internal mechanisms of Thinking and NoThinking to achieve effective solutions. We summarize our contributions as follows:

- We first formalize the Mode Selection task as a more challenging variant of the Early Exit problem, which shares similar objectives but differs in decision timing. Both are unified under the same Exit$(\cdot)$ function.
- Through extensive experiments across various datasets, we observe that prompt-based approaches often fail due to their limited classification capabilities, whereas internal states-based methods generally perform better but still struggle with stability.
- By utilizing metrics such as ROC-AUC, ECE, and Brier score, we uncover the instability in method decisions and the model's inherent abilities, emphasizing the persistent challenges of this task.

2 Preliminaries of Early Exit in LRMs

Overthinking is a critical phenomenon in large reasoning models (LRMs), where model performance initially improves with extended reasoning chains but deteriorates beyond a certain point [2, 27, 34]. This implies that thinking longer is not always better. Existing Early Exit methods aim to determine the optimal moment to terminate the reasoning process. In reasoning-focused LRMs, these models function as System 2, where the generation process is divided into two stages: *thinking* and *conclusion*, collectively referred to as THINKING mode [30, 14]:

$$\text{THINKING}: \underbrace{\text{[Prompt]} + \texttt{<think>}}_{\text{Input}} + \underbrace{\text{[Thoughts]} + \texttt{</think>} + \text{[Conclusion]}}_{\text{Generate}} \tag{1}$$

Here, `<think>` and `</think>` serve as the delimiters for the beginning and end of the reasoning process, respectively. [Thoughts] represents the detailed reasoning process, while [Conclusion] contains the concise answer. Formally, the reasoning segment [Thoughts] reveals the trajectory of problem-solving, commonly referred to as Chain of Thought (CoT) [36]. At a micro level, temporally consecutive tokens in [Thoughts] can be grouped into short semantic, logical, and grammatically coherent chunks T_j:

$$\texttt{[Thoughts]} = T := [T_1, \cdots, T_n] \tag{2}$$

Each reasoning chunk in this paradigm is auto-regressively generated by conditioning on the question Q and the preceding reasoning token: $T_i = \text{LRM}(Q, T_{<i})$. For simplicity, the Early Exit method performs $\text{Exit}(Q, T_{<i})$ based on the question and the reasoning trajectory up to $T_{<i}$. This determines whether to terminate the reasoning process at chunk T_i by appending `</think>`. The fundamental technique in Early Exit is predicting whether the current information is sufficient to solve the question. To this end, we collect several methods capable of expressing their confidence in the current reasoning process. Considering existing methods in LRMs, we categorize them into three types based on their information sources: *Prompt-based* and *Internal States-based*. Each type implements the $\text{Exit}(Q, T_{<i})$ function differently, leveraging distinct resources to determine when to stop reasoning.

2.1 Prompt-based Early Exit

FLASHTHINK [15] employs a separate verification model, π_ϕ, parameterized by ϕ, along with a specific prompt. This model is used to decide when to stop reasoning by iteratively querying at the end of each thought:

$$s_i = \pi_\phi(Q, T_{<i}, \text{[Verification Prompt]}) \tag{3}$$

where if s_i is true will stop the following thinking process.

PROMPT CONFIDENCE (PROMPTCONF) [42] employs a prompt-based method, enabling LRMs to generate confidence scores during the reasoning process based on specific rules. At the end of the process, these scores are mapped into one of ten bins, ranging from "Almost no chance (0–0.1)" to "Almost certain (0.9–1.0)." Each bin includes both a linguistic descriptor (e.g., "Almost certain") and its corresponding numerical probability C_i (e.g., "0.9–1.0") to improve interpretability. The decision to terminate reasoning is then made as follows:

$$C_i = \pi_\theta(Q, T_{<i}, \text{[Confidence Prompt]}) \tag{4}$$

$$s_i = \mathbb{I}(C_i > \lambda) \tag{5}$$

where π_θ represents the LRM parameterized by θ, λ is the confidence threshold, and $\mathbb{I}(\cdot)$ is an indicator function that returns true when C_i exceeds λ. The method sequentially evaluates each

intermediate answer in the reasoning trace, using the prompt to output confidence scores for the temporary answers.

DYNASOR-COT [6] leverages a "Probe-In-The-Middle" approach, which appends carefully designed guidance prompts at intermediate stages of the reasoning process to explicitly elicit the model's current answer (e.g., "Oh, I suddenly got the answer to the whole problem, Final Answer: \boxed{}"). This method monitors LRMs at regular intervals (e.g., every 32, 64, or 128 tokens) to check for consistent answers across multiple intervals. Once consistent answers are detected, the reasoning process is terminated.

2.2 Internal States-based

PROBE CONFIDENCE (PROBECONF) [44] trains a MLP-based probing model to detect the intermediate answer correctness. Specifically, the probing model takes the last layer hidden states at the last token position of T_i as input h_i, output a probability C_i of intermediate correctness by:

$$C_i = \text{MLP}(h_i) \tag{6}$$

where the decision to exit is made by comparing the obtained confidence with the empirical threshold λ, as described in Equation 5.

DEER [39] monitors the action transition points as potential early exit points (e.g. wait) to inducing intermediate answer, which incorporated the answer delimiters (i.e., \boxed{}) into the prompt to facilitate a more precise identification of the trial answers, as follows: $A_i = \text{LRM}(Q, T_{<i}, I)$ where Q denotes the input prompt, $T_{<i}$ denotes already generated thoughts, I denotes the answer induced prompt and $A_i = [a_{0,i}, a_{1,i}, \cdots, a_{l,i}]$ is the trail answer.

The confidence evaluator module computes the confidence of the induced trial answer. We take the maximum predicted probability of each token as its confidence. For multi-token trial answers, the overall confidence is computed as the mean confidence across all constituent tokens as follows:

$$p(a_{t,i}) = \text{softmax}(\mathcal{M}(Q, T_{<i}, I, a_{<t,i})), \quad \mathcal{C}_i = \left(\prod_{j=1}^{l} \max_{a_{t,i} \in \mathcal{V}} p(a_{t,i}) \right)^{1/l} \tag{7}$$

where the $\mathcal{M}$ is the LM head of LRMs. Finally, the decision to exit early is made by comparing the obtained confidence C_i with the empirical threshold λ, as described in Equation 5.

ENTROPY [41] proves that each reasoning step contributes to reducing entropy over the answer space and increasing confidence in the correct answer. ENTROPY uses the similar trick in DEER to obtain the trail answer A_i and compute the conditional entropy at step i:

$$C_i = -\sum_{t=1}^{l} P(a_{t,i} \mid Q, T_{<i}, I, a_{<t,i}) \log P(a_{t,i} \mid Q, T_{<i}, I, a_{<t,i}) \tag{8}$$

where $P(a_{t,i} \mid Q, T_{<i}, I, a_{<t,i})$ is estimated from the model's output probabilities. After each intermediate reasoning step, the model computes the average entropy $C_i^{\text{avg}} = \frac{1}{l} \sum_{i=1}^{l} C_i$ over the answer space. Reasoning is terminated early once the average entropy falls below a confidence threshold, which is parameterized by a hyperparameter $\alpha \in [0, 1]$:

$$s_i = \mathbb{I}(C_i^{\text{avg}} \leq \alpha \cdot \frac{1}{e \ln 2}) \tag{9}$$

3 Mode Selection As a Harder Early Exit Problem

As introduced in Section 2, we construct a unified view of early exit. In this section, we first introduce the NOTHINKING mode as an alternative option for mode selection [22] representing short-CoT. Recent studies have highlighted NOTHINKING as an effective method for adaptive reasoning, which can significantly reduce token usage while preserving performance [19, 17].

3.1 NoThinking Mode

NoThinking is designed to bypass the explicit reasoning process of LRMs by carefully crafting input prompts that already include the end-of-thinking delimiter `</think>` within pre-defined fake thoughts. This methodology has been adopted by systems like Qwen3 [38] and DeepSeek-V3.1 [30] to enable hybrid inference in LRMs. By using fake thoughts, many LRMs are able to save token usage while maintaining strong performance on simpler tasks. The process follows this pattern:

$$\text{NoThinking}: \underbrace{[\text{Prompt}] + \texttt{<think>} + [\text{Fake Thoughts}] + \texttt{</think>}}_{\text{Input}} + \underbrace{[\text{Conclusion}]}_{\text{Generate}} \tag{10}$$

where the [Fake Thoughts] for T_0^{fake} fall into two categories: (i) an empty block, represented as `<think></think>`, and (ii) predefined thinking-completion statements (e.g., `<think>` Okay, I think I have finished thinking. `</think>`), which are intended to encourage LRMs to skip the reasoning process. We refer to this as *Zero-Step Thinking*.

3.2 Mode Selection formulates as Early Exit Problem

In this paper, we present an interesting perspective: Mode Selection can be viewed as a more challenging variant of Early Exit for performing adaptive reasoning. As illustrated in Figure 1, the key difference between these two approaches lies in their decision-making processes. Early Exit decides when to stop reasoning dynamically during the reasoning process, whereas Mode Selection operates as a static method, making decisions before explicit reasoning begins at the zero-step.

For simplicity, Early Exit iteratively performs $\text{Exit}(Q, T_{<i})$ at each reasoning step, while Mode Selection executes $\text{Exit}(Q, T_0^{\text{fake}})$ by replacing dynamically generated [Thoughts] with pre-defined [Fake Thoughts]. This means that Mode Selection lacks any question-specific reasoning information when making decisions, making the task significantly more difficult. In this study, we conduct a systematic empirical investigation to explore whether reasoning models can effectively "think" using only zero-step thoughts.

4 Experiments

4.1 Experiments Setup

Datasets. We evaluate model performance across 4 benchmarks, including three mathematical reasoning benchmarks: GSM8K [3], MATH-500 [13], and AIME 2025 [4], as well as one scientific reasoning benchmark: GPQA Diamond [25]. Among the mathematical reasoning benchmarks, GSM8K and MATH-500 are generally regarded as relatively simple reasoning tasks, whereas AIME 2025 is considered more challenging.

Metrics. We selected Accuracy (**Acc**), Token Number (**Tok**), and NoThinking Ratio (**NR**) as the evaluation metrics. Acc denotes the final answer accuracy. Tok denotes the average generation length per sample to evaluate the cost. NR denotes the choose of NoThinking mode ratio.

Backbone LRMs. We conducted experiments using the open-source DeepSeek-R1-Distill-Qwen series of models (1.5B, 7B, and 32B), which are distilled from DeepSeek-R1 [12]. These models utilize Qwen2.5-1.5B [38] as the backbone and are fine-tuned on 800k high-quality reasoning samples. This fine-tuning process enables the models to achieve superior performance on logical and mathematical reasoning tasks.

Baselines. In this work, we aim to systematically analyze whether it is possible to use zero-step thoughts to perform Mode Selection. We evaluate six baseline methods discussed in Section 2, including FlashThink, PromptConf, Dynasor-CoT, ProbeConf, DEER, and Entropy. Additionally, we introduce three new baselines: Thinking, NoThinking, and Pre-judge [43]. Thinking directly evaluates the performance of LRMs without any external intervention. NoThinking incorporates pre-defined fake thoughts to bypass explicit reasoning. Pre-judge prompts the LRMs to decide whether reasoning is required, based on carefully designed input prompts. For

Table 1: Experimental results across various types of reasoning models are presented. "Acc" denotes accuracy, "Tok" represents the token count, and "NR" refers to the NoThinking mode rate. ↑ indicates that higher values are better, while ↓ indicates that lower values are preferable. Additionally, the baselines are categorized as follows: represents basic baselines, represents prompt-based methods, and indicates internal states-based methods.

Method	GSM8K			MATH-500			AIME25			GPQA-D		
	Acc↑	Tok↓	NR	Acc↑	Tok↓	NR	Acc↑	Tok↓	NR	Acc↑	Tok↓	NR
DeepSeek-R1-Distill-Qwen-1.5B												
Thinking	85.7 (**0.0**)	2,455 (**0.0%**)	0%	83.8 (**-0.0**)	5,445 (**0.0%**)	0%	33.3 (**-0.0**)	15,214 (**0.0%**)	0%	24.8 (**-0.0**)	9,818 (**0.0%**)	0%
NoThinking	72.7 (-13.0)	261 (-89%)	100%	68.6 (-15.2)	1,411 (-74.1%)	0%	20.0 (-13.3)	6,614 (-56.5%)	100%	18.2 (-6.6)	945 (-90.4%)	100%
FlashThink	85.7 (**0.0**)	2,455 (**0.0%**)	100%	83.8 (**-0.0**)	5,445 (**0.0%**)	100%	33.3 (**-0.0**)	15,214 (**0.0%**)	100%	24.8 (**-0.0**)	9,818 (**0.0%**)	100%
PromptConf	85.0 (-0.7)	2,272 (-7.5%)	6.6%	82.0 (-1.8)	5,159 (-5.3%)	8.0%	40.0 (+6.7)	9,731 (-36.0%)	66.7%	24.2 (-0.6)	8,598 (-12.4%)	16.2%
Dynasor-CoT	84.2 (-1.5)	2,103 (-14.3%)	20.3%	80.8 (-3.0)	4,752 (-12.7%)	23.2%	33.3 (**-0.0**)	14,227 (-6.5%)	6.7%	23.7 (-1.1)	8,959 (-8.7%)	15.2%
Pre-Judge	77.8(-7.9)	1,269 (-48.3%)	59.3%	81.8 (-2.0)	5,100 (-6.3%)	12.0%	33.3 (**-0.0**)	15,214 (**0.0%**)	100%	22.7 (-2.1)	5,193 (-47.1%)	55.6%
ProbeConf	84.4(-1.3)	2,299 (-6.4%)	9.2%	82.6 (-1.2)	4,941 (-9.3%)	13.8%	40.0 (+6.7)	11,596 (-23.8%)	20.0%	22.7 (-2.1)	7,399 (-24.6%)	26.8%
DEER	85.3 (-0.4)	2,276 (-7.3%)	9.7%	82.6 (-1.2)	5,067 (-6.9%)	11.6%	33.3 (**-0.0**)	8334 (-45.2%)	53.3%	25.3 (+0.5)	9,274 (-5.5%)	9.6%
Entropy	85.3 (-0.4)	2,278 (-7.2%)	8.5%	82.6 (-1.2)	5,199 (-4.5%)	11.0%	40.0 (+6.7)	12,784 (-16.0%)	26.7%	28.3 (+3.5)	6,268 (-36.2%)	43.4%
DeepSeek-R1-Distill-Qwen-7B												
Thinking	92.3 (**-0.0**)	1,687 (**-0.0%**)	0%	93.0 (**-0.0**)	4,100 (**-0.0%**)	0%	40.0 (**-0.0**)	15,024 (**-0.0%**)	0%	47.5 (**-0.0**)	8,447 (**-0.0%**)	0%
NoThinking	87.6 (-4.7)	268 (-84.1%)	100%	78.0 (-15.0)	781 (-81.0%)	100%	26.7 (-13.3)	1,352 (-91.0%)	100%	19.2 (-28.3)	688 (-91.9%)	100%
FlashThink	92.3 (**-0.0**)	1,687 (**-0.0%**)	0%	93.0 (**-0.0**)	4,100 (**-0.0%**)	0%	40.0 (**-0.0**)	15,024 (**-0.0%**)	0%	47.5 (**-0.0**)	8,447 (**-0.0%**)	0%
PromptConf	89.5 (-2.8)	691 (-59.0%)	72.9%	82.6 (-10.4)	1,776 (-56.7%)	76.8%	33.3 (-6.7)	8,301 (-44.7%)	60.0%	29.3 (-18.2)	4,093 (-51.5%)	54.5%
Dynasor-CoT	92.1 (-0.2)	1330 (-21.2%)	30.0%	89.2 (-3.8)	2,993 (-27.0%)	34.6%	40.0 (**-0.0**)	13,587 (-9.6%)	13.3%	31.3 (-16.2)	4778 (-43.4%)	53.0%
Pre-Judge	91.3(-1.0)	1,211 (-28.2%)	37.1%	90.0 (-3.0)	3,381 (-17.5%)	26.6%	40.0 (**-0.0**)	14,875 (-1.0%)	6.7%	38.9 (-8.6)	6,925 (-18.0%)	20.7%
ProbeConf	91.6(-0.7)	1,394 (-17.4%)	18.1%	92.2 (-0.8)	3,950 (-27.5%)	8.4%	46.7 (+6.7)	11,025 (-26.6%)	26.7%	47.0 (-0.5)	8,190 (-3.0%)	6.6%
DEER	92.6 (+0.3)	1,476 (-10.8%)	18.1%	93.2 (+0.2)	3,912 (-4.6%)	8.4%	40.0 (**-0.0**)	14,603 (-2.8%)	6.7%	47.0 (-0.5)	8,269 (-2.1%)	4.0%
Entropy	92.2(-0.1)	1,505 (-7.2%)	18.1%	92.6 (-0.4)	3,871 (-5.6%)	8.4%	40.0 (**-0.0**)	12,784 (-14.9%)	26.7%	48.0 (+0.5)	6,693 (-20.8%)	26.8%
DeepSeek-R1-Distill-Qwen-32B												
Thinking	95.8 (**-0.0**)	1,453 (**-0.0%**)	0%	94.0 (**-0.0**)	3,462 (**-0.0%**)	0%	66.7 (**-0.0**)	11,155 (**-0.0%**)	0%	62.1 (**-0.0**)	6,690 (**-0.0%**)	0%
NoThinking	95.9 (+0.1)	1,280 (-11.9%)	100%	94.2 (+0.2)	3,550 (+2.5%)	100%	60.0 (-6.7)	13,933 (+24.9%)	100%	62.6 (+0.5)	6,576 (-1.7%)	100%
FlashThink	95.8 (**-0.0**)	1,453 (**-0.0%**)	0%	94.0 (**-0.0**)	3,462 (**-0.0%**)	0%	66.7 (**-0.0**)	11,155 (**-0.0%**)	0%	62.1 (**-0.0**)	6,690 (**-0.0%**)	0%
PromptConf	95.6 (-0.2)	1,338 (-59.0%)	83.1%	94.0 (**-0.0**)	3,488 (+0.8%)	64.2%	60.0 (-6.7)	11,295 (+1.3%)	26.7%	61.6 (-0.5)	6,557 (-2.0%)	31.3%
Dynasor-CoT	95.7 (-0.1)	1,384 (-7.9%)	33.7%	93.8 (-0.2)	3,392 (-2.0%)	39.0%	66.7 (**-0.0**)	11,054 (-0.9%)	13.3%	63.6 (+1.5)	6,699 (+0.1%)	46.5%
Pre-Judge	95.7(-0.1)	1,317 (-9.4%)	37.1%	94.4 (+0.4)	3,467 (+0.1%)	39.4%	66.7 (**-0.0**)	11,155 (**-0.0%**)	0%	62.1 (**-0.0**)	6,669 (-0.3%)	0.51%
ProbeConf	96.0(+0.2)	1,389 (-4.4%)	38.1%	94.2 (+0.2)	3,413 (-1.4%)	20.8%	66.7 (**-0.0**)	11,182 (+0.2%)	20.0%	65.2 (+3.1)	6,312 (-5.7%)	90.4%
DEER	96.1 (+0.3)	1,342 (-7.6%)	53.4%	94.2 (+0.2)	3,419 (-1.2%)	20.8%	66.7 (**-0.0**)	10,947 (-1.9%)	53.3%	65.7 (+3.6)	6,686 (-0.1%)	73.7%
Entropy	95.7(-0.1)	1,423 (-2.1%)	38,1%	94.2 (+0.2)	3,512 (+1.4%)	40.8%	66.7 (**-0.0**)	10,567 (-5.3%)	46.7%	62.6 (+0.5)	6578 (-1.7%)	90.4%

all baselines, we first randomly sample instances for both Thinking and NoThinking with a temperature setting of 0.6. Next, we use the existing baselines to determine the flag s_i for each instance. In the main results, we manually select the best performance for each baseline by varying the threshold λ. Detailed implementation details for each baseline are provided in Appendix A.

4.2 Main Results

In this section, we present the main experimental results and an in-depth analysis of how different types of baselines perform on the Mode Selection task. Notably, different baselines exhibit varying decision strategies. For FlashThink and Pre-Judge, the model determines the mode at the zero-step stage without relying on threshold-based decisions. In contrast, baselines like PromptConf and Dynasor-CoT, which depend on fixed discrete scores, consistently use the highest score as the threshold λ. However, for baselines such as DEER, ProbeConf, and Entropy, fixed thresholds fail to effectively capture their optimal performance. Consequently, we manually selected the best-performing thresholds for these baselines in the main experiment.

Limitations of Prompt-based Methods. We begin by analyzing prompt-based methods, which rely on a verification model to decide whether to terminate the reasoning process. Due to the limited information available from fake thoughts T_0^{fake}, FlashThink consistently determines that LRMs must continue reasoning, leading to a 0% NR rate across all scenarios. Moreover, the performance of other methods varies significantly across different datasets and base models. For example, PromptConf achieves notable token reductions while maintaining performance stability on smaller models (e.g., a 6.7 accuracy improvement on AIME25 with a 36.0% reduction in token usage for a 1.5B model). However, its effectiveness decreases as model size increases (e.g., 7B and 32B models). Similar trends are observed in Dynasor-CoT and PromptConf. Nonetheless, these two methods demonstrate improved stability by leveraging self-generated scores to evaluate reasoning states.

Internal States Tell More Than Language. Due to the limited effectiveness of prompt-based methods, we further explore approaches that leverage model internal states, including ProbeConf, DEER, and Entropy. For manual selection of λ, we first retain thresholds that yield significant

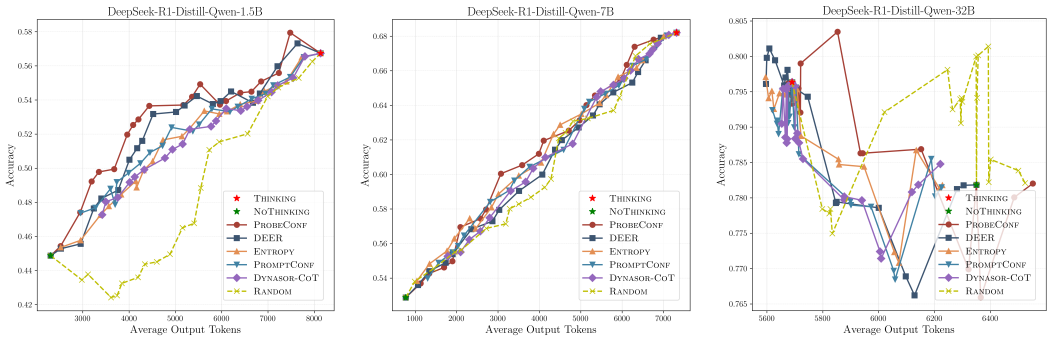


Figure 2: Trade-off between average accuracy and average output tokens across different methods and three model sizes. Each point on the curve represents a distinct λ value from 0.1 to 1.0.

accuracy improvements. If such improvements are absent, we instead select thresholds that align the accuracy or NR across these three methods, which facilitates more consistent analysis. As shown in Table 1, on the 1.5B model, DEER and ENTROPY achieve better accuracy retention on GSM8k and GPQA-D, while PROBECONF provides higher token compression in MATH-500 and even yields a 6.7-point accuracy gain on AIME25 with 26.6% token compression. A similar trend is observed for the 7B and 32B models. These methods consistently reduce token usage while maintaining performance, and in some cases even surpass baseline results. For example, DEER achieves superior performance on the 32B model, reducing token usage while preserving accuracy and even outperforming THINKING. Overall, the results demonstrate that signals from the model's internal states provide more reliable indicators for selecting the appropriate mode.

5 Analyses

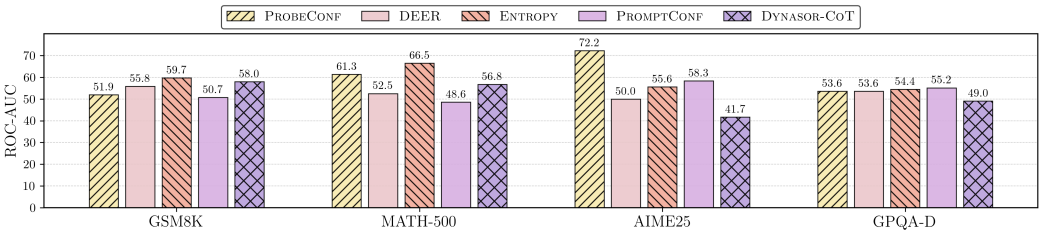
To better understand the behavior and effectiveness of these methods in the Mode Selection task, we conduct detailed analyses. These include examining threshold dynamics (Section 5.1), evaluating ROC-AUC scores across methods (Section 5.2), and analyzing the correlation between C_i and NOTHINKING mode accuracy (Section 5.3).

5.1 Analysis of Threshold Dynamics

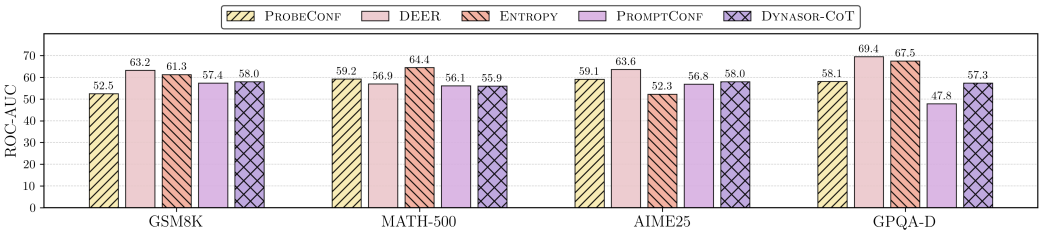
Since internal state–based methods primarily rely on a well-defined threshold λ, whose optimal value varies unpredictably across tasks and models, we evaluate the trade-off between accuracy and cost across three model scales: DeepSeek-R1-Distill-Qwen-1.5B, 7B, and 32B. For each model, we plot average accuracy against average token cost by systematically sweeping the decision threshold λ. The resulting trade-off curves are shown in Figure 2.

On the 1.5B model, all methods consistently outperform the random baseline, demonstrating their ability to dynamically select appropriate reasoning modes to balance performance and computational cost. In particular, PROBECONF and DEER show stronger performance than the other methods, suggesting that leveraging internal states can yield a better Pareto frontier by achieving more favorable accuracy–cost trade-offs across operating points.

A clear trend emerges in the accuracy–cost curves under varying thresholds. For the 1.5B model, mode selection methods are clearly distinguished from the random baseline, highlighting their effectiveness on smaller models. However, as the scale increases to 7B, the performance gap between methods narrows. For example, even the best-performing method, PROBECONF, occasionally underperforms relative to the RANDOM baseline, resulting in weaker separation. This phenomenon becomes more pronounced with the 32B model, where the effectiveness of THINKING and NOTHINKING is reversed (Table 1). On datasets such as MATH-500 and AIME25, NOTHINKING generates more tokens than THINKING, because certain examples under the NOTHINKING strategy still produce lengthy outputs. We hypothesize that this behavior arises because LRMs have internalized the reasoning process: forcing them to bypass it with fake thoughts not only fails to elicit direct summarization but may also cause the model to restart its reasoning. A similar trend has also been observed in QwQ-32B by [47].



(a) ROC AUC comparison for DeepSeek-R1-Distill-Qwen-1.5B model.



(b) ROC AUC comparison for DeepSeek-R1-Distill-Qwen-7B model.

Figure 3: ROC-AUC comparison between different model sizes: (a) 1.5B and (b) 7B.

Table 2: Expected Calibration Error (ECE) and Brier score in 1.5B and 7B cross datasets. ↓ means smaller is better.

Baselines	GSM8K		MATH-500		AIME25		GPQA-D	
	ECE ↓	Brier ↓	ECE ↓	Brier ↓	ECE ↓	Brier ↓	ECE ↓	Brier ↓
DeepSeek-R1-Distill-Qwen-1.5B								
ProbeConf	**0.122**	**0.215**	**0.078**	**0.215**	0.447	0.320	0.407	0.322
DEER	0.218	0.262	0.198	0.270	**0.224**	**0.203**	**0.297**	**0.250**
Dynasor-CoT	0.235	0.261	0.188	0.264	0.200	0.259	0.441	0.391
DeepSeek-R1-Distill-Qwen-7B								
ProbeConf	0.296	0.222	**0.128**	**0.190**	0.401	0.364	0.528	0.436
DEER	0.272	**0.213**	0.192	0.232	0.145	**0.181**	**0.354**	**0.273**
Dynasor-CoT	**0.268**	**0.213**	0.229	0.234	**0.111**	0.200	0.636	0.579

Therefore, in the following sections, we focus on the 1.5B and 7B models to gain deeper insights into the factors influencing performance in mode selection.

5.2 Analysis of ROC-AUC across methods

In this section, we move beyond fixed thresholds and instead use ROC-AUC scores for more robust comparisons. As shown in Figure 3, the results reinforce our earlier findings: these methods are highly sensitive to both dataset and model, and no single approach consistently achieves optimal performance across all scenarios. For example, on the 1.5B model, ProbeConf attains the highest ROC-AUC score of 72.2 on AIME25, substantially outperforming other methods. However, this advantage does not generalize to the 7B model or to other datasets. Overall, methods leveraging internal states tend to outperform prompt-based approaches in most cases, suggesting that a model's intrinsic states encode richer information than the lossy representations conveyed by language.

5.3 Correlation Analysis of C_i with NoThinking Mode Accuracy

Since these methods rely on the NoThinking prompt template to compute a score C_i as the mode selection indicator, we further investigate whether C_i correlates with answer correctness in NoThinking mode. To this end, we report the corresponding Expected Calibration Error (ECE) [24] and Brier score [9]. For a fair comparison, Entropy is excluded due to numerical issues.

As shown in Table 2, different methods exhibit varying trends across datasets. Notably, PROBECONF and DEER generally yield lower errors compared to the prompt-based method DYNASOR-COT. Moreover, when examining how these metrics align with performance, we observe that the Brier score on the 7B model is highly consistent with the ROC-AUC results in Figure 3. For example, DEER achieves the lowest Brier score of 0.181 on AIME25, corresponding to the best ROC-AUC among all methods. In contrast, these metrics appear less informative for the 1.5B model. We hypothesize that smaller models suffer a substantial drop in performance under NOTHINKING mode, making both internal and external scores less reliable for performance estimation.

6 Related Work

Adaptive Thinking for Overthinking. Recent studies have shown that longer CoT do not always improve performance [2, 37], and in some cases may even lead to overthinking, particularly in high-capacity models [16]. This has sparked growing interest in minimal or implicit reasoning strategies [22], highlighting the need for more nuanced approaches and adaptive control of reasoning depth through RL or related techniques [5, 40, 28]. In this work, we focus on a specific subtype of adaptive thinking, namely mode selection, which aims to determine the appropriate reasoning mode prior to explicit reasoning. Existing approaches either train a routing mechanism or directly prompt the model to decide [19, 6, 29]. In contrast, we present a systematic analysis of how methods developed for Early Exit can be leveraged to address this challenging problem.

Early Exit for Efficient Reasoning. Efficiency in LLMs is an active research area, with methods that adapt the number of reasoning steps according to task difficulty, confidence, or resource constraints [26, 17]. Among these methods, Early Exit has proven especially effective: it determines an optimal stopping point, reducing token usage and sometimes even improving performance. Formally, this process can be abstracted as the function $\text{Exit}(Q, T_{<i})$ [39, 15, 44]. Building on how such methods are implemented, we focus on prompt-based approaches, which rely on the reasoning model itself or an auxiliary model to decide whether to stop thinking based on textual information [15, 42, 6]. In contrast, internal state–based approaches leverage intrinsic signals from the model, such as hidden states or output logits, to make this decision [39, 44, 41]. We extend this framework to the Mode Selection paradigm, where the task becomes more complex as performing $\text{Exit}(Q, T_0^{\text{fake}})$.

7 Conclusion

In this work, we introduce Mode Selection as a more challenging variant of the Early Exit problem, where the model should determine the appropriate reasoning mode before explicit reasoning begins. Through extensive empirical studies on multiple reasoning benchmarks, we find that prompt-based approaches are often limited by weak classification capability under minimal information, while internal states–based approaches achieve better performance but suffer from instability. Our analysis further shows that existing evaluation metrics are insufficient to fully explain method behaviors, highlighting the complexity of Mode Selection. Overall, our findings point to the need for more robust approaches that better exploit model internal mechanisms of THINKING and NOTHINKING, paving the way for future research on adaptive reasoning strategies in LRMs.

Acknowledgments

This work was supported by the National Key R&D Program of China (No. 2022ZD0160503) and Beijing Natural Science Foundation (L243006) and the National Natural Science Foundation of China (No.62376270).

References

[1] Claude 3.7 sonnet system card.

[2] 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.

[3] 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.

[4] MAA Committees. Aime problems and solutions. `https://artofproblemsolving.com/wiki/index.php/AIME_Problems_and_Solutions`. Accessed: 2025-07-22.

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

[6] 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.

[7] Kanishk Gandhi, Ayush Chakravarthy, Anikait Singh, Nathan Lile, and Noah D Goodman. Cognitive behaviors that enable self-improving reasoners, or, four habits of highly effective stars. arXiv preprint arXiv:2503.01307, 2025.

[8] Soumya Suvra Ghosal, Souradip Chakraborty, Avinash Reddy, Yifu Lu, Mengdi Wang, Dinesh Manocha, Furong Huang, Mohammad Ghavamzadeh, and Amrit Singh Bedi. Does thinking more always help? understanding test-time scaling in reasoning models. arXiv preprint arXiv:2506.04210, 2025.

[9] W Brier Glenn et al. Verification of forecasts expressed in terms of probability. Monthly weather review, 78(1):1–3, 1950.

[10] Ruihan Gong, Yue Liu, Wenjie Qu, Mingzhe Du, Yufei He, Yingwei Ma, Yulin Chen, Xiang Liu, Yi Wen, Xinfeng Li, et al. Efficient reasoning via chain of unconscious thought. arXiv preprint arXiv:2505.19756, 2025.

[11] Google. Gemini 2.5 pro. `https://blog.google/technology/google-deepmind/gemini-model-thinking-updates-march-2025/`. Accessed: 2025-07-22.

[12] 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.

[13] 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.

[14] Aaron Jaech, Adam Kalai, Adam Lerer, Adam Richardson, Ahmed El-Kishky, Aiden Low, Alec Helyar, Aleksander Madry, Alex Beutel, Alex Carney, et al. Openai o1 system card. arXiv preprint arXiv:2412.16720, 2024.

[15] 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.

[16] 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.

[17] Wei Li, Yanbin Wei, Qiushi Huang, Jiangyue Yan, Yang Chen, James T Kwok, and Yu Zhang. Dynamicmind: A tri-mode thinking system for large language models. arXiv preprint arXiv:2506.05936, 2025.

[18] Zhong-Zhi Li, Duzhen Zhang, Ming-Liang Zhang, Jiaxin Zhang, Zengyan Liu, Yuxuan Yao, Haotian Xu, Junhao Zheng, Pei-Jie Wang, Xiuyi Chen, et al. From system 1 to system 2: A survey of reasoning large language models. arXiv preprint arXiv:2502.17419, 2025.

[19] Guosheng Liang, Longguang Zhong, Ziyi Yang, and Xiaojun Quan. Thinkswitcher: When to think hard, when to think fast. arXiv preprint arXiv:2505.14183, 2025.

[20] Wanlong Liu, Junxiao Xu, Fei Yu, Yukang Lin, Ke Ji, Wenyu Chen, Yan Xu, Yasheng Wang, Lifeng Shang, and Benyou Wang. Qfft, question-free fine-tuning for adaptive reasoning. arXiv preprint arXiv:2506.12860, 2025.

[21] Haotian Luo, Haiying He, Yibo Wang, Jinluan Yang, Rui Liu, Naiqiang Tan, Xiaochun Cao, Dacheng Tao, and Li Shen. Ada-r1: Hybrid-cot via bi-level adaptive reasoning optimization. arXiv preprint arXiv:2504.21659, 2025.

[22] 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.

[23] 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.

[24] Mahdi Pakdaman Naeini, Gregory Cooper, and Milos Hauskrecht. Obtaining well calibrated probabilities using bayesian binning. In Proceedings of the AAAI conference on artificial intelligence, volume 29, 2015.

[25] 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.

[26] 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.

[27] Yang Sui, Yu-Neng Chuang, Guanchu Wang, Jiamu Zhang, Tianyi Zhang, Jiayi Yuan, Hongyi Liu, Andrew Wen, Shaochen Zhong, Hanjie Chen, et al. Stop overthinking: A survey on efficient reasoning for large language models. arXiv preprint arXiv:2503.16419, 2025.

[28] Yuqiao Tan, Shizhu He, Huanxuan Liao, Jun Zhao, and Kang Liu. Dynamic parametric retrieval augmented generation for test-time knowledge enhancement. arXiv preprint arXiv:2503.23895, 2025.

[29] Yuqiao Tan, Shizhu He, Kang Liu, and Jun Zhao. Neural incompatibility: The unbridgeable gap of cross-scale parametric knowledge transfer in large language models. In Proceedings of the 63rd Annual Meeting of the Association for Computational Linguistics, 2025.

[30] DeepSeek Team. Deepseek-v3.1 release. https://api-docs.deepseek.com/news/news250821. Accessed: 2025-08-27.

[31] NovaSky Team. Sky-t1: Train your own o1 preview model within $450. https://novasky-ai.github.io/posts/sky-t1/. Accessed: 2025-07-22.

[32] Qwen Team. Qwq-32b: Embracing the power of reinforcement learning. https://qwenlm.github.io/blog/qwq-32b/. Accessed: 2025-07-22.

[33] Songjun Tu, Jiahao Lin, Qichao Zhang, Xiangyu Tian, Linjing Li, Xiangyuan Lan, and Dongbin Zhao. Learning when to think: Shaping adaptive reasoning in r1-style models via multi-stage rl. arXiv preprint arXiv:2505.10832, 2025.

[34] Minzheng Wang, Yongbin Li, Haobo Wang, Xinghua Zhang, Nan Xu, Bingli Wu, Fei Huang, Haiyang Yu, and Wenji Mao. Adaptive thinking via mode policy optimization for social language agents. arXiv preprint arXiv:2505.02156, 2025.

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

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

[37] Yuyang Wu, Yifei Wang, Ziyu Ye, Tianqi Du, Stefanie Jegelka, and Yisen Wang. When more is less: Understanding chain-of-thought length in llms. arXiv preprint arXiv:2502.07266, 2025.

[38] 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.

[39] 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.

[40] Wenkai Yang, Shuming Ma, Yankai Lin, and Furu Wei. Towards thinking-optimal scaling of test-time compute for llm reasoning. arXiv preprint arXiv:2502.18080, 2025.

[41] 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.

[42] Dongkeun Yoon, Seungone Kim, Sohee Yang, Sunkyoung Kim, Soyeon Kim, Yongil Kim, Eunbi Choi, Yireun Kim, and Minjoon Seo. Reasoning models better express their confidence. arXiv preprint arXiv:2505.14489, 2025.

[43] XING Zeyu, Xing Li, Huiling Zhen, Xianzhi Yu, Mingxuan Yuan, and Sinno Jialin Pan. Large reasoning models know how to think efficiently. In ES-FoMo III: 3rd Workshop on Efficient Systems for Foundation Models.

[44] 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.

[45] Jiajie Zhang, Nianyi Lin, Lei Hou, Ling Feng, and Juanzi Li. Adaptthink: Reasoning models can learn when to think. arXiv preprint arXiv:2505.13417, 2025.

[46] Ruiqi Zhang, Changyi Xiao, and Yixin Cao. Long or short cot? investigating instance-level switch of large reasoning models. arXiv preprint arXiv:2506.04182, 2025.

[47] Rongzhi Zhu, Yi Liu, Zequn Sun, Yiwei Wang, and Wei Hu. When can large reasoning models save thinking? mechanistic analysis of behavioral divergence in reasoning. arXiv preprint arXiv:2505.15276, 2025.

A Implementation Details

A.1 Basic Setup

Decoding details. All evaluations were conducted in a zero-shot Chain-of-Thought (CoT) setting using the prompt: "Please reason step by step, and put your final answer within \boxed." For decoding, we used sampling with a temperature of 0.6. The ground-truth answers in our experiments consist exclusively of well-structured numerical values or categorical options; accordingly, we applied rule-based checks to verify mathematical equivalence. We set the maximum generation length to 16,384 tokens to ensure complete problem-solving attempts were captured.

Benchmarks. To comprehensively evaluate the models' reasoning capabilities, we employ four representative benchmarks widely used in the field. GSM8K [3] is a carefully curated dataset of 1,319 elementary mathematics problems, specifically designed to assess multi-step reasoning in foundational math tasks. Each problem typically requires two to eight sequential operations, relying primarily on basic arithmetic applied across multiple intermediate steps. MATH-500 [13] is a challenging benchmark composed of competition-level problems drawn from diverse high school mathematics domains, including Prealgebra, Algebra, and Number Theory. For consistency with prior research, we adopt the 500-problem subset originally curated by OpenAI. AIME 2025 [4] consists of 30 problems selected from the 2025 American Invitational Mathematics Examination (AIME). This prestigious contest evaluates mathematical reasoning across a broad range of domains, including arithmetic, algebra, counting, geometry, number theory, probability, and other advanced secondary school topics. Beyond math, we also evaluate on scientific reasoning tasks. GPQA [25] is a PhD-level benchmark covering physics, chemistry, and biology. Domain experts with PhDs in these areas achieve only 69.7% accuracy on this dataset [25], highlighting its difficulty. For our experiments, we specifically use the highest-quality subset, GPQA Diamond, which comprises 198 questions.

A.2 Baselines Implementations

In this section, we provide the clear implementation of each baseline methods.

THINKING. The THINKING mode refers to the default chat template of existing LRMs [12, 38] which only appends the `<think>` token to enable thinking process:

Prompt Template for THINKING

```
<BOS_TOKEN><|USER|>{Question}
Please reason step by step, and put your final answer within \boxed{}.
<|Assistant|><think>
```

NOTHINKING. The NOTHINKING mode proposed by [22] can significant reduce token usage to achieve better performance for sampling more times. In recent studies, it has been seen as a straight but useful baseline to stimulate short-CoT ability of LRMs by appending fake thoughts with `</think>` to skip the thinking process. We use the same prompt template as follow:

Prompt Template for NOTHINKING

```
<BOS_TOKEN><|USER|>{Question}
Please reason step by step, and put your final answer within \boxed{}.
<|Assistant|><think>
Okay, I think I have finished thinking.
</think>
```

FLASHTHINK. FLASHTHINK utilizes another verification model π_ϕ to decide whether to skip thinking. Following [15], we select Qwen2.5-7B-Instruct[2] as π_ϕ to verify with the following prompt where the {Question} will be replace with the real question Q:

[2]https://huggingface.co/Qwen/Qwen2.5-7B-Instruct

Prompt Template for FLASHTHINK

<BOS_TOKEN><|USER|>
Based on the following question and thought, please judge whether the thought is sufficient to support solving the question. Please directly output yes or no instead of outputting other content.
Question
{Question}
Thought
Okay, I think I have finished thinking.
<|Assistant|><think>

PROMPTCONF. PROMPTCONF utilizes prompt method to make LRMs themselves generate confidence score during thinking process based on specific rule. The scores range from "Almost no chance (0–0.1)" to "Almost certain (0.9–1.0)", and we dirrectly append "0." to make LRMs output correct pattern and we select the lower bound of the output range as final score:

Prompt Template for PROMPTCONF

<BOS_TOKEN><|USER|>
For the following question, classify your confidence into one of the following classes based on how likely your answer is to be correct:
- "Almost no chance" (0.0-0.1)
- "Highly unlikely" (0.1-0.2)
- "Chances are slight" (0.2-0.3)
- "Unlikely" (0.3-0.4)
- "Less than even" (0.4-0.5)
- "Better than even" (0.5-0.6)
- "Likely" (0.6-0.7)
- "Very good chance" (0.7-0.8)
- "Highly likely" (0.8-0.9)
- "Almost certain" (0.9-1.0)
Each category reflects the probability that your answer is correct.
At the end of your output, format your answer and confidence as
Confidence: $SCORE
where SCORE is one of the probability ranges of the scores above.
Here is the question:
{Question}
<|Assistant|><think>
</think>
Confidence: 0.

DYNASOR-COT. DYNASOR-COT employs a carefully designed guidance prompt to elicit the model's immediate answer (e.g., "Oh, I suddenly got the answer to the whole problem, Final Answer: \boxed{..."). However, mode selection is a static process and thus cannot adopt the monitoring setup in [42], which checks intermediate states at regular intervals. Inspired by self-consistency [35], we randomly sample three outputs and use the maximum agreement ratio as the score. This results in three possible scores 33.3%, 66.7%, and 100%, where higher values indicate greater model confidence on a given question.

Prompt Template for DYNASOR-COT

<BOS_TOKEN><|USER|>{Question}
Please reason step by step, and put your final answer within \boxed{}.
<|Assistant|><think>
Okay, I think I have finished thinking.
Oh, I suddenly got the answer to the whole problem, Final Answer: \boxed{

PRE-JUDGE. PRE-JUDGE is similar to FLASHTHINK but uses LRM itself to verify. We use the same prompt in [43] to output the boolean value of 'require_slow_thinking' as s_i:

Prompt Template for PRE-JUDGE

<BOS_TOKEN><|USER|>
You are a math problem solver. For the following question, determine if it requires slow thinking or can be solved quickly. You do not need to give me any explanation, just give me a json with the following keys: require_slow_thinking.
For example: {'require_slow_thinking': true}
Here is the question:
{Question}
<|Assistant|><think>
</think>
{'require_slow_thinking':

PROBECONF. PROBECONF relys on a MLP-based probing model to detect the intermediate answer correctness. For the probing models we used in our experiments, we adopt the already trained models which trained in MATH-500[3] , and using the same training script to perform grid search to obtain best hyper-parameters. During evaluation, we use the same prompt template of NOTHINKING to extract the hidden states h_i of `</think>`:

Prompt Template for PROBECONF

<BOS_TOKEN><|USER|>{Question}
Please reason step by step, and put your final answer within \boxed{}.
<|Assistant|><think>
Okay, I think I have finished thinking.
</think>

DEER. DEER utilizes the induced prompt I to output the intermediate answer, and calculate C_i based on the average of output logits to show model's confidence:

Prompt Template for DEER

<BOS_TOKEN><|USER|>{Question}
Please reason step by step, and put your final answer within \boxed{}.
<|Assistant|><think>
Okay, I think I have finished thinking.
</ think>
Final Answer
The final answer is \boxed{

ENTROPY. ENTROPY leverages the entropy of output logits to estimate the model's confidence. For simplicity, we compute entropy only at the final thinking position to assess whether the LRM exhibits sufficient confidence:

Prompt Template for ENTROPY

<BOS_TOKEN><|USER|>{Question}
Please reason step by step, and put your final answer within \boxed{}.
<|Assistant|><think>
Okay, I think I have finished thinking.

[3]https://github.com/AngelaZZZ-611/reasoning_models_probing