Stop Before You Fail: Operational Capability Boundaries for Mitigating Unproductive Reasoning in Large Reasoning Models

Qingjie Zhang[1], Yujia Fu[1], Yang Wang[2], Liu Yan[2], Tao Wei[2], Ke Xu[1], Minlie Huang[1], and Han Qiu[1*]

[1]Tsinghua University, [2]Ant Group

Emails: {qj-zhang24@mails., qiuhan@}tsinghua.edu.cn [*]Corresponding author

arXiv:2509.24711v4 [cs.AI] 31 Aug 2026

Abstract

Current answering paradigms for Large Reasoning Models (LRMs) often fail to account for the fact that some questions may lie beyond the model's operational capability boundary, leading to long but unproductive reasoning. In this paper, we study whether LRMs expose early signals predictive of such cases, and whether these signals can be used to mitigate unproductive reasoning. In black-box settings, we find that reasoning expressions contain failure-predictive signals. In white-box settings, we show that the hidden states of the last input token contain information that is predictive of whether a question will not be solved correctly under our evaluation setup. Building on these observations, we propose two test-time monitoring strategies: reasoning expression monitoring and hidden states monitoring, that reduce token usage by 62.7–93.6%, substantially improving efficiency and reliability while largely preserving accuracy. We open-source our code.

1 Introduction

Large Reasoning Models (LRMs) have demonstrated remarkable capabilities on complex reasoning tasks such as mathematics (Guo et al., 2025; Jaech et al., 2024; Ahn et al., 2024). However, when confronted with hard questions, they frequently exhibit unproductive reasoning, manifested as repetitive looping or error accumulation until context limits are exhausted (Zhou et al., 2025; Chen et al., 2025b; Yao et al., 2025; Mukherjee et al., 2025) (as shown in Appendix K).

This phenomenon suggests an important limitation of current answering paradigms: they often fail to account for the fact that some questions are likely to fall beyond the model's *operational capability boundary* under a fixed evaluation setup, leading to long yet ultimately unproductive reasoning traces. If such cases can be identified early, reasoning could become more reliable and efficient: instead of continuing low-value deliberation, the model may abstain from full reasoning and provide a concise partial approach.

Existing approaches to studying reasoning limits have primarily taken an external perspective, for example by constructing increasingly challenging benchmarks such as Humanity's Last Exam (HLE) (Phan et al., 2025), or by designing real-time routers that estimate question difficulty and route harder questions to more capable models (OpenAI, 2025). In contrast, relatively little work has examined whether such limits are reflected in the model's internal signals, despite the importance of this question for both understanding reasoning behavior and further improving reliability and efficiency at test time.

Accordingly, this paper studies whether LRMs expose ***early signals predictive of eventual failure***, and whether such signals can be used to mitigate unproductive reasoning. We use the term *capability boundary* in an operational sense: under a fixed evaluation setup, it refers to the empirical separation between questions the model eventually solves and those it does not. Figure 1 provides an overview of our approach. From both black-box and white-box perspectives, we identify signals that help distinguish unsolvable from solvable questions. Leveraging these signals, we enable LRMs to assess, on the fly, whether a question is likely to remain unsolved under the current setup, thereby improving the efficiency and reliability of reasoning. We summarize our contributions as:

- We provide empirical evidence that LRMs' reasoning expressions are associated with the correctness of final answers, revealing language patterns that are predictive of eventual failure.
- From a black-box perspective, we characterize failure-predictive reasoning trajectories. We identify a diverging pattern: an accelerated growing confidence trajectory for solved ques-

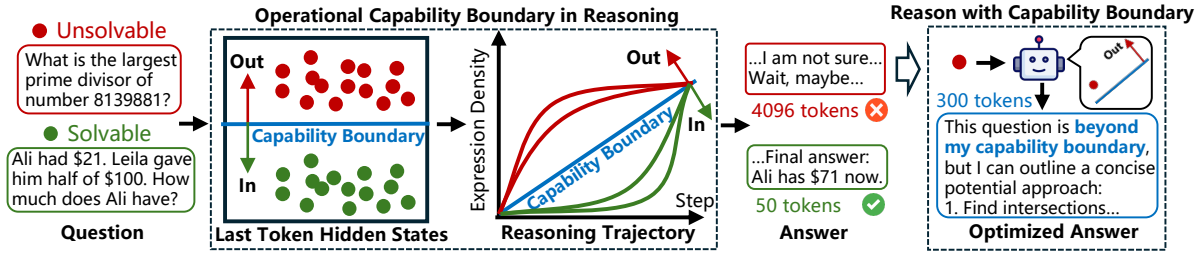


Figure 1: Signals related to the operational capability boundary are reflected in reasoning trajectories as expression density patterns, and in the last token hidden states even before reasoning begins. Leveraging these signals enables LRMs to identify questions that are likely to remain unsolved and to provide more reliable and efficient responses.

tions, but a convergent uncertainty trajectory for unsolved questions.

- From a white-box perspective, we show that the hidden states of the last input token contain information predictive of whether a question will eventually be solved correctly. In particular, solvable and unsolvable questions are close to linearly separable in the hidden space, even before reasoning begins.
- We design two test-time monitoring strategies: reasoning expression monitoring (black-box) and hidden states monitoring (white-box). Experiments show that these strategies substantially improve efficiency and reliability while largely preserving accuracy, reducing token usage by 62.7–93.6%.

2 Related Work

From knowledge boundary to capability boundary. Recent studies have examined the knowledge boundaries of Large Language Models (LLMs), focusing on the limits of factual knowledge encoded in parameters (Yin et al., 2024; Li et al., 2024; Wen et al., 2024; Deng et al., 2025). These works primarily emphasize factual coverage, with retrieval-augmented generation (RAG) being a common solution to extend the accessible knowledge space (Gao et al., 2023; Lewis et al., 2020).

By contrast, the capability boundary concerns what reasoning tasks LLMs can reliably solve (Chen et al., 2025a; Xue et al., 2025; Chen et al., 2024a). This notion is broader yet rarely formalized. Existing efforts typically approximate it externally: either by constructing more challenging benchmarks such as Humanity's Last Exam (HLE) (Phan et al., 2025), or by designing routing mechanisms such as GPT-5 router that delegate harder questions to more capable models (OpenAI, 2025). *In contrast, we examine whether capability boundaries are reflected in models' internal signals.*

Interpretability of reasoning expressions and hidden states. Interpretability aims to uncover how LLMs understand and execute tasks, providing insights that enhance their reliability and efficiency. Recent studies have approached from both reasoning expressions and hidden states. On the reasoning expressions side, Guo et al. (2025) highlight the anthropomorphic cues, "aha moments", as a key role in reasoning. Qian et al. (2025) tracks mutual information along the generation and finds sharp surges at specific thinking tokens (e.g., "Hmm", "Wait", "Therefore"). Bogdan et al. (2025) frame sentence-level importance as a counterfactual influence problem, showing that only a few critical utterances steer the reasoning trajectory.

On the hidden states side, Wendler et al. (2024) probe multilingual Llama models and reveal English as the pivot language in the internal structure. Fan et al. (2024) demonstrate that certain intermediate layers already encode sufficient predictive features, implying redundancy in the full forward pass. Zhang et al. (2025) train linear probes on hidden states to verify intermediate answers, anticipating future correctness. *Building on both perspectives, our work analyzes reasoning expressions and hidden states to study signals related to the operational capability boundary.*

Efficient reasoning: from solvable questions to unsolvable questions. A range of recent works examine the phenomenon of overthinking (Sui et al., 2025; Hou et al., 2025). Chen et al. (2024b) show that LRMs often allocate excessive computation to easy questions, while Zhang et al. (2024) report that multi-turn self-correction can lead to unnecessary cost. To mitigate such inefficiency, Han et al. (2025) compresses unnecessary reasoning by including a reasonable token budget in the prompt. Wang et al. (2025) demonstrate that suppressing frequent reflective tokens such as "Wait", "Hmm", or "Alternatively" improves reasoning efficiency

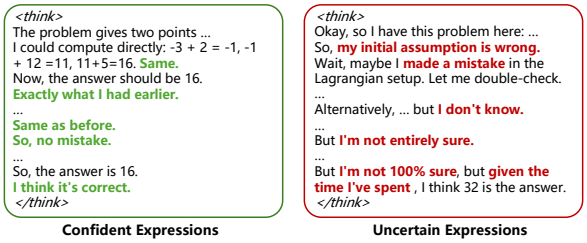


Figure 2: Confident vs. uncertain expressions.

without harming accuracy. Yang et al. (2025a); Fu et al. (2025, 2024) introduce mechanisms to decide during reasoning whether further steps are unnecessary, allowing the model to terminate early and output the final answer directly.

However, these works focus on cases where LRMs can solve a question but still waste computation by reasoning for too long. By contrast, *our work addresses a complementary and less-studied scenario: questions that are likely to fall beyond the model's operational capability boundary, where LRMs may nevertheless continue reasoning unproductively despite being unlikely to arrive at a correct answer under the current setup.*

3 LRMs Expose Signals of Failure

We begin our exploration by evaluating GPT-oss-20B (Agarwal et al., 2025), DeepSeek-R1-0528-Qwen3-8B (abbreviated as R1-Distill-Qwen3-8B) (Guo et al., 2025), DeepSeek-R1-Distill-Qwen-32B (abbreviated as R1-Distill-Qwen-32B) (Guo et al., 2025), and QwQ-32B (Qwen Team, 2024) on several standard mathematical benchmarks, including AIME'25 (MAA, 2025), AIME'24 (MAA, 2024), HMMT (HMMT, 2025), and AMC'23 (MAA, 2023) (see Appendix A for detailed descriptions of datasets and models).

Inspired by works which monitors LRMs' reasoning (Baker et al., 2025; He et al., 2025; Luo et al., 2024), we examine the reasoning processes (e.g., the segments wrapped within `<think>` in DeepSeek-like models) to look for internalized assessments regarding whether the input question can be correctly solved. As illustrated in Figure 2, we observe that such signals, expressed as anthropomorphic tones (Guo et al., 2025; Yang et al., 2025b), emerge throughout the reasoning process, and become especially evident in its final stages. Toward the end of reasoning, models often produce explicit statements that reflect their confidence level. Strikingly, these expressed beliefs tend to align with the correctness of their final answers. For example, when the reasoning contains more confident expressions such as "So, no mistake" or "I think it's correct", the final answers are typically correct. Conversely, when uncertain expressions like "My initial assumption is wrong" or "I'm not 100% sure" appear, the final answers are incorrect.

	Can %	Cannot %
GPT-oss-20B	80.2	100
R1-Distill-Qwen3-8B	97.6	100
R1-Distill-Qwen-32B	90.6	100
QwQ-32B	86.0	100

Table 1: LRMs' final answers aligns with their expressed belief, especially for incorrect final answers.

This observation motivates us to quantify the alignment between expressed belief and answer correctness, which we formalize through:

$$\begin{aligned} \text{Can } \% &= \mathbb{P}(\checkmark \mid \text{More confidence}), \\ \text{Cannot } \% &= \mathbb{P}(\text{✗} \mid \text{More uncertainty}). \end{aligned} \tag{1}$$

Specifically, Can % is the proportion of questions for which the model expresses more confidence during reasoning and indeed produces a correct answer, while Cannot % is the proportion of questions for which the model expresses more uncertainty and indeed fails to provide the correct answer.

Table 1 shows the results. We observe that LRMs' expressed beliefs align with the correctness of their final answers. In particular, for questions where LRMs express more uncertainty during reasoning, the final answers are consistently incorrect (Cannot % = 100). This phenomenon is observed across multiple models, suggesting that ***LRMs expose signals of failure***.

4 Capability Boundary in Expressions

Motivated by the observations in the previous section, we hypothesize that LRMs' reasoning expressions may contain signals related to the operational capability boundary. We therefore systematically investigate the dynamics of confident and uncertain expressions throughout the reasoning process.

To ensure that the analysis is not tied to any particular LRM's reasoning style, we first extract confident and uncertain expressions separately for each model, based on its characteristic reasoning patterns (see the detailed extraction pipeline and phrase-list sensitivity analysis in Appendix C). Such anthropomorphic expressions are commonly observed in LRM reasoning traces and have been discussed in prior work as informative features of the reasoning process (Yang et al., 2025b; Guo et al., 2025; Qian et al., 2025).

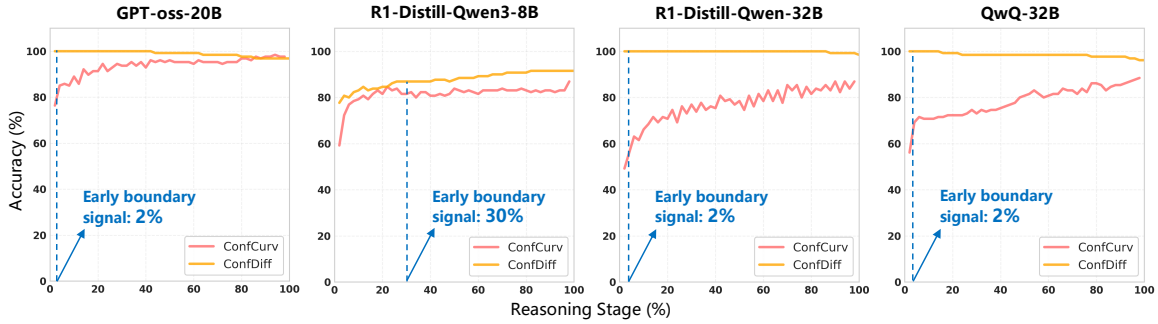


Figure 3: Accuracy (%) of ConfDiff and ConfCurv to separate unsolvable and solvable questions through out the reasoning stage. ConfDiff is more reliable as it achieves higher accuracy and exhibits smaller fluctuations. Boundary signal appears at an early stage (e.g., 2%).

Then, we uniformly divide the reasoning process into multiple stages (e.g., 50 stages) and compute the density of confident and uncertain expressions at each stage. Finally, concatenating these values yields the expression density trajectories, denoted as $\mathcal{D}_C(t)$ for confident expressions and $\mathcal{D}_U(t)$ for uncertain expressions. These trajectories capture how the density of each type of expression evolves over the course of reasoning.

We plot $\mathcal{D}_C(t)$ and $\mathcal{D}_U(t)$ in Figure 4, separately for cases with correct and wrong final answers. We observe that LRMs exhibit clear differences in their reasoning expression trajectories depending on final answer correctness. When the final answer is correct, the confident trajectory lies above the uncertain trajectory and shows an accelerating increasing trend. In contrast, when the final answer is incorrect, the uncertain trajectory dominates over the confident trajectory and tends to converge. This contrast suggests that reasoning expression trajectories contain useful signals (blue line in Figure 4) for distinguishing questions that are eventually solved from those that remain unsolved under the current setup. In particular, the dominant trajectory tends to be more concave for solvable questions and more convex for unsolvable ones.

To formalize these patterns, we design two indicators for determining whether a question is likely to lie within or beyond the capability boundary.

Confidence Differential (ConfDiff) measures the sign of gap between uncertain and confident trajectory along the reasoning process. Specifically, we accumulate the sign up to a given reasoning stage percentage $s \in [0, 100]$, and treat the question as beyond the capability boundary if the cumulative value exceeds a threshold α_s:

$$\mathbb{P}_{t\in(0,s)}(\mathcal{D}_U(t) > \mathcal{D}_C(t)) > \alpha_s. \quad (2)$$

Confidence Curvature (ConfCurv) computes the second derivative of the expression density trajectory to capture its convexity (Boyd and Vandenberghe, 2004). A concave trajectory corresponds to questions within the capability boundary, while a convex trajectory indicates questions beyond it. Also, we accumulate the sign of convexity up to a reasoning stage percentage s and treat the question as beyond the capability boundary if the value exceeds a threshold β_s:

$$\mathbb{P}_{t\in(0,s)}\left(\frac{d^2}{dt^2}\left(\mathcal{D}_U(t) - \mathcal{D}_C(t)\right) < 0\right) > \beta_s. \quad (3)$$

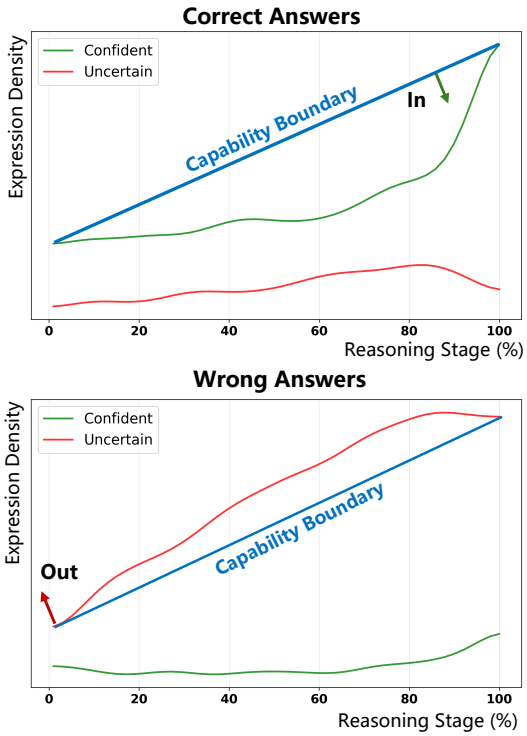


Figure 4: Eventually solved questions show concave confident-expression curves, whereas unsolved questions show convex uncertain-expression curves.

Figure 3 shows that the two proposed indicators can reliably distinguish unsolvable from solvable questions, achieving high accuracy across different LRMs throughout the reasoning stage. Between the two indicators, ConfDiff is more reliable as it

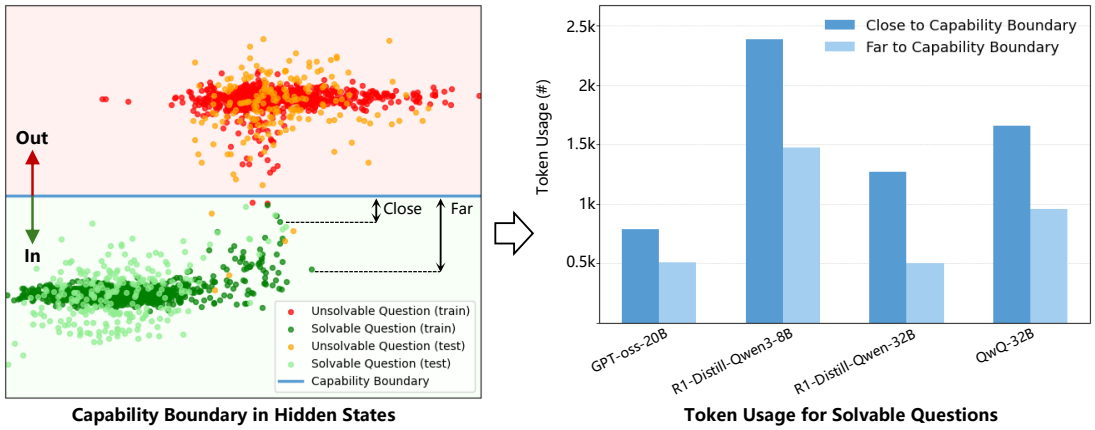


Figure 5: **Left:** A linear classifier trained on hidden states clearly separates solvable and unsolvable questions under our setup. **Right:** Among solvable questions, those closer to the operational capability boundary require more token usage (1.5–2×) to arrive at the correct answer.

achieves higher accuracy and exhibits smaller fluctuations during the reasoning process. Moreover, strong predictive accuracy emerges early, e.g., as early as 2% for GPT-oss-20B. Interestingly, the gradually declining yellow curve of this model suggests that these signals may be more salient at the beginning than at the end of reasoning. One possible explanation is that repetitive looping and error accumulation in later stages make the trajectories noisier (Yao et al., 2025; Mukherjee et al., 2025). Overall, these results show that ***reasoning expressions contain early signals related to the operational capability boundary and predictive of eventual failure***.

5 Capability Boundary in Hidden States

Following the observation that reasoning expressions expose early signals related to the operational capability boundary, it is natural to ask whether similar signals may emerge even earlier, potentially before the model begins reasoning. We therefore investigate whether hidden states of the input question already contain information predictive of eventual success or failure.

Since hidden states are high-dimensional (e.g., 4096), limited samples may lead to sparsity issues that obscure statistical patterns (Vishwakarma, 2024; Köppen, 2000). We mix the mathematical benchmarks used in the previous sections and randomly split it into an 80% training set and a 20% test set, with balanced labels for eventually unsolved and solved questions.

Next, we extract hidden states from the prefilling stage as it processes the input question (Yuan et al., 2024; Pope et al., 2023), using the hook method (Vig, 2019). Specifically, we focus on the hidden state of the last input token, which encapsulates the semantic content of the entire question (Dong et al., 2025; Radford et al., 2019). Finally, we apply simple linear classifiers to these hidden states, including Linear Discriminant Analysis (LDA) and Logistic Regression (LR) (Hastie, 2009). Using final answer correctness as labels, we test whether eventually solved and unsolved questions can be linearly separated, i.e., whether last token hidden states contain information predictive of eventual correctness under our setup.

	LDA	**LR**	
		$C = 0.1$	$C = 1$
GPT-oss-20B	98.0	98.9	98.2
R1-Distill-Qwen3-8B	96.7	96.7	97.0
R1-Distill-Qwen-32B	97.8	97.8	98.0
QwQ-32B	98.9	98.2	98.2

Table 2: Accuracy (%) of separating solvable and unsolvable questions in hidden states using linear classifiers (robust across hyperparameter choices).

The left subfigure of Figure 5 visualizes the classification of solvable and unsolvable questions (see Appendix J for results of all LRMs), together with the decision boundary (blue line) of a linear classifier (e.g., LDA). We observe that the two classes of questions are clearly separated under random train–test splits. To quantify this separation, Table 2 reports the classification accuracy, which is consistently close to 100% and robust to the choice of classifier and hyperparameters (e.g., the regularization coefficient C of LR). These results suggest that the hidden states of the last input token contain information that is predictive of eventual correctness, and thus closely related to the operational capability boundary under our evaluation setup.

At the same time, these signals should be interpreted with care: they may partially reflect question difficulty rather than a purely intrinsic notion of capability boundary. We provide additional analyses in Appendix F.

Interestingly, we also observe that among eventually solved questions, some lie very close to the classifier decision boundary, while others lie farther away. We quantify this distance by the probability difference assigned by the LDA classifier to the solved and unsolved classes (Hwang et al., 2023). We then compare these groups in terms of token usage during inference. Although LRMs answer both groups correctly, questions closer to the boundary require on average 1.5–$2\times$ more tokens than those farther from the boundary (right subfigure of Figure 5). This suggests that the hidden state signals are related not only to eventual correctness, but also to the amount of reasoning effort required to obtain a correct answer.

Overall, these results indicate that ***the hidden states of the last input token contain useful signals related to the operational capability boundary, and that these signals also correlate with the reasoning effort required to solve a question.***

Moreover, because these signals are extracted from reasoning expressions and hidden states rather than task-specific heuristics, they may extend beyond mathematical reasoning. We provide analyses on LiveCodeBench (Jain et al., 2024) and GPQA-Diamond (Rein et al., 2023) in Appendix H.

6 Reason with Capability Boundary

Because signals related to the operational capability boundary emerge early, they can be used to improve the efficiency and reliability of LRMs' responses. In this section, we propose two ***test-time monitoring*** strategies that leverage such signals and demonstrate their effectiveness.

6.1 Experimental Design

Strategies. The key idea is to monitor signals of capability boundaries on the fly, either from reasoning trajectories (black-box) or hidden states (white-box), and to adjust reasoning when a question lies beyond the model's effective solving regime.

Reason Expression Monitoring (Monitor$_{\textbf{express}}$) is a black-box strategy that leverages expressions density during reasoning. The pipeline is:

1. Construct expression trajectories: continuously track the densities of confident and uncertain expressions (e.g., "I think that's it" vs. "I might be wrong") across reasoning steps to form expression density trajectories $\mathcal{D}_C(t)$ and $\mathcal{D}_U(t)$.

2. Predict likely-unsolved question: analyze these expression trajectories on the fly using quantitative indicators such as Confidence Differential (ConfDiff) to determine whether a question is likely to remain unsolved.

3. Early stop unproductive reasoning and reprompt with suffix: if the trajectory indicates the question is likely to remain unsolved, we early terminate reasoning and reprompt the model to output a concise approach. This is done by appending a prompt suffix at the end of the question:

Prompt suffix

The above question is beyond your capability boundary. Do not solve the question, just provide a concise potential approach of less than 5 steps:

Hidden States Monitoring (Monitor$_{\textbf{hidden}}$) is a white-box strategy that leverages the hidden states of the last input token. The pipeline is as follows:

1. Extract hidden states: capture the hidden states of the last input token, which encodes the semantic representation of the entire question during the prefilling stage.

2. Predict likely-unsolved question: apply simple linear classifiers (e.g., LDA or LR) trained to distinguish eventually unsolved from solved questions. This prediction is made before reasoning begins.

3. Avoid unproductive reasoning via constrained output prefix: if the classifier predicts that the question is likely to remain unsolved under the current setup, the model abstains from full reasoning and instead outputs a concise possible approach. This is implemented by constraining the response to begin with an output prefix (see Appendix E for the ablation study):

Output prefix

```
<think>
```
I think this question is beyond my capability boundary. I cannot fully solve it, but I can outline a concise potential approach. I must give a concise outline to the user (less than 10 steps)!
```
</think>
```
This question is beyond my capability boundary, but I can outline a concise potential approach:

Metrics. We adopt four metrics to systematically evaluate the two proposed strategies:

- **Accuracy (ACC)** measures the proportion of correct answers, serving as the standard indica-

Context Length			2048		4096	
Metric	ACC (%) ↑	HA (%) ↑	Token ↓	Overflow (%) ↓	Token ↓	Overflow (%) ↓
GPT-oss-20B	74.6	0	2029	97.0	4015	97.0
+BoostAbstention	74.6▾0	0▴0	2023▾0.3	97.0▾0	3933▾2.0	84.8▾12.2
+Monitor$_{\text{express}}$	74.6▾0	100▴100	1135▾44.1	21.2▾75.8	1499▾62.7	15.2▾81.8
+Monitor$_{\text{hidden}}$	73.7▾0.9	98.9▴98.9	1784▾12.1	78.4▾18.6	3317▾17.4	5.4▾91.6
R1-Distill-Qwen3-8B	76.9	0	2048	100	4096	100
+BoostAbstention	76.9▾0	0▴0	2048▾0	100▾0	4096▾0	100▾0
+Monitor$_{\text{express}}$	74.6▾2.3	63.3▴63.3	1455▾29.0	40.0▾60.0	2551▾37.7	30.0▾70.0
+Monitor$_{\text{hidden}}$	73.8▾3.1	97.9▴97.9	787▾61.6	22.2▾77.8	1243▾69.7	0▾100
R1-Distill-Qwen-32B	56.2	0	2048	100	4095	97.4
+BoostAbstention	56.2▾0	0▴0	2005▾2.1	94.7▾5.3	3903▾4.7	91.2▾6.2
+Monitor$_{\text{express}}$	56.2▾0	100▴100	801▾60.9	0▾100	801▾80.4	0▾97.4
+Monitor$_{\text{hidden}}$	55.3▾0.9	97.0▴97.0	457▾77.7	5.3▾94.7	565▾86.2	5.3▾92.1
QwQ-32B	71.5	0	2048	100	4096	100
+BoostAbstention	71.5▾0	0▴0	2048▾0	100▾0	4096▾0	100▾0
+Monitor$_{\text{express}}$	71.5▾0	100▴100	1164▾43.2	32.4▾67.6	1828▾55.4	32.4▾67.6
+Monitor$_{\text{hidden}}$	71.5▾0	96.9▴96.9	262▾87.2	0▾100	262▾93.6	0▾100

Table 3: Monitor$_{\text{express}}$ and Monitor$_{\text{hidden}}$ compared to baseline BoostAbstention. Our strategies enable LRMs to recognize questions beyond capability boundary (HA) without compromising accuracy (ACC), and provide more efficient response (Token usage and Overflow).

tor of reasoning quality (Xia et al., 2025).

- **Hard Abstention (HA)** measures the model's ability to recognize questions beyond its capability boundary. Different to abstention which focus on underspecification, outdated information, or noncompliance (Kirichenko et al., 2025; Madhusudhan et al., 2024; Brahman et al., 2024), HA specifically evaluates whether LRMs can explicitly abstain on unsolvable questions instead of producing incorrect answers.
- **Token** usage measures the number of tokens consumed during reasoning and answering (Snell et al., 2024). This metric directly reflects the efficiency of the reasoning process.
- **Overflow** measures the proportion of cases where the model's output exceeds the context window, which identifies incomplete or abnormal responses (Ben-Kish et al., 2025).

Notably, ACC is evaluated on the full dataset, whereas the remaining three metrics are computed only on questions for which LRMs produce incorrect answers. For token usage and overflow, we vary the models' context length to observe how our strategies scale in longer context scenarios. Table 14 additionally reports false-positive abstention and token usage over all questions.

Baseline. Prior work has explored improving performance by self-verification after generating the answer (Madhusudhan et al., 2024; Kadavath et al., 2022) or by monitoring uncertainty and other undesirable behaviors (Baker et al., 2025; Farquhar et al., 2024; Kuhn et al., 2023). However, these approaches are orthogonal to our scenario: the former introduces additional computation after producing an answer, while the latter does not address eventually unsolved questions (see more baseline comparison in Table 6).

Therefore, we adopt a baseline that improve efficiency by enabling the model to abstain. In particular, we consider BoostAbstention (Kirichenko et al., 2025) which crafts a system prompt that encourages the model to abstain in designated scenarios, showing effectiveness for both reasoning and standard LLMs. For our experiments, we modify the system prompt to better match the abstention scenarios of eventually unsolved questions (see Appendix D for the full prompt).

6.2 Results

Table 3 compares the performance of the original models, the baseline BoostAbstention, and our proposed monitoring strategies. We highlight the following key observation.

Observation 1: our strategies enable LRMs to recognize eventually unsolved questions while maintaining comparable accuracy on solved ones. Although the original models achieve acceptable Accuracy (ACC), their Hard Abstention (HA) is zero, indicating a complete lack of ability to abstain on questions beyond the capability boundary. Moreover, almost all Overflow is 100%, suggesting that most incorrect cases arise because LRMs strug-

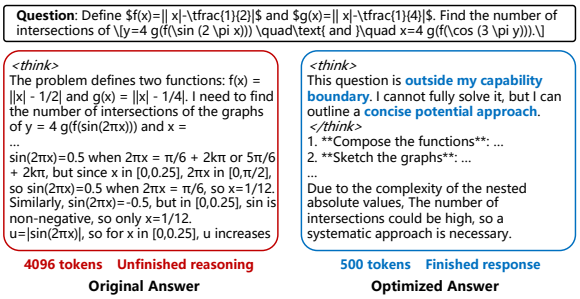


Figure 6: Our strategies avoid unproductive reasoning by acknowledging eventually unsolved questions, improving efficiency and reliability of response.

gling to solve a hard question until context limit.

Promisingly, both strategies achieve strong performance. First, their ACC remains nearly unchanged compared to the original models, showing that good reasoning performance is preserved. Meanwhile, the substantially high HA (almost 100%) demonstrates that LRMs can now effectively identify likely-unsolved questions.

Surprisingly, and contrary to the observation from Kirichenko et al. (2025), the baseline Boost-Abstention has almost no effect in our scenario. This reveals that mathematical reasoning problems differ from outdated or unsafe requests: abstention cannot be induced simply by system prompts, but instead requires boundary signals encoded in hidden states or reasoning expressions.

Observation 2: our strategies enable LRMs to provide more efficient and reliable responses to eventually unsolved questions by offering concise solution approaches. Figure 6 illustrates the responses produced by our strategies compared to the original models. The original models struggle with eventually unsolved questions and fail to complete the reasoning process even after exhausting the context length. Such responses provide neither a correct answer nor valuable insights from the lengthy reasoning trace, while also wasting time.

In contrast, our strategies detect these unsolved questions and guide the model to explicitly acknowledge that they are beyond capability boundary, while offering a concise possible approach. This response paradigm provides users with possible hints for problem solving without wasting time on meaningless output. And importantly, it does not compromise the model's ability to produce correct answers for solvable questions. Table 15 provides an additional utility evaluation of these concise outputs.

This explains why we observe large reductions in both token usage (up to 93.6%) and overflow (up to 100%) in Table 3, indicating that reasoning with capability boundaries greatly improves efficiency and mitigates incomplete outputs. Besides, this reduction becomes more evident with larger context length. For example, overflow reduces from 78.4% to 5.4% on GPT-oss-20B when the context length is extended from 2048 to 4096. This suggests that our strategies hold promise for optimizing long-context reasoning tasks (Ling et al., 2025; Kuratov et al., 2024). We provide in Appendix I an additional analysis with the context length set to maximum.

Observation 3: LRMs vary in reasoning behavior correction, with some good at instruction following and others more prone to output guidance. We compare the performance of $\text{Monitor}_{\text{express}}$ and $\text{Monitor}_{\text{hidden}}$ across different LRMs. For GPT-oss-20B, $\text{Monitor}_{\text{express}}$ achieves lower token usage than $\text{Monitor}_{\text{hidden}}$. We find this model is resistant to output guidance via constrained output prefix: although it sometimes begins by generating a solution approach, it tends to later negate this and restart the reasoning process, resulting in long and unproductive outputs. In contrast, its instruction-following ability is relatively strong, and thus re-prompting with the prompt suffix via $\text{Monitor}_{\text{express}}$ yields lower token usage.

Conversely, the other three LRMs show weaker ability of instruction following but more prone to output guidance, making $\text{Monitor}_{\text{hidden}}$ more effective than $\text{Monitor}_{\text{express}}$. This suggests that, in addition to the black-box versus white-box distinction, the choice of optimization strategy in practice should also consider LRMs' behavioral tendencies.

7 Conclusion

We study whether signals related to the operational capability boundary can help mitigate unproductive reasoning in LRMs. Across both black-box and white-box analyses, we find that LRMs expose such signals early: reasoning expressions differ systematically between eventually solved and unsolved questions, while the hidden states of the last input token are predictive of eventual correctness even before reasoning begins. Based on these findings, we propose two test-time monitoring strategies that reduce unproductive reasoning. Experiments show that they largely preserve accuracy on solvable questions while substantially improving efficiency and reliability on questions lie beyond the operational capability boundary.

Limitations

Interpretation of capability boundaries. In this paper, we use *capability boundary* in an operational sense, namely as the empirical separation between questions a model eventually solves and those it does not under a fixed evaluation setup. Accordingly, the signals we identify should be interpreted as predictive correlates of eventual failure rather than definitive evidence of a formally defined intrinsic capability boundary. In particular, these signals may partially reflect question difficulty. Although we attempt to mitigate this concern through per-benchmark and cross-benchmark analyses (Appendix F), the extent to which these signals capture a general notion of capability boundary remains an open question. Developing a more principled formalization of capability boundaries (e.g., disentangling question difficulty) is an important direction for future work.

Scope of tasks and evaluation domains. Our evaluation focuses primarily on mathematical benchmarks, which allow systematic analysis of unsolvability signals but may not fully represent other reasoning scenarios. While our experiments on LiveCodeBench (Jain et al., 2024) and GPQA-Diamond (Rein et al., 2023) suggest similar results (Appendix H), the generality of capability boundary signals across broader domains (e.g., complex decision making or scientific reasoning) remains to be established. Future work should investigate whether similar strategies apply across more diverse tasks.

Access assumptions. $\text{Monitor}_{\text{express}}$ requires complete, visible reasoning traces, while $\text{Monitor}_{\text{hidden}}$ requires access to the last-token hidden state. Therefore, the former cannot be directly applied when chain-of-thought is hidden, truncated, or summarized, and the latter cannot be applied to closed-source APIs unless hidden states are exposed.

Ethics Statement

ACL Ethics Policy is respected in this work. It studies LRMs' capability boundaries, aiming to improve efficiency and reliability. No human subjects, sensitive personal data, or harmful content were involved. The proposed methods are intended for research purposes only and do not pose foreseeable risks of misuse.

Acknowledgements

This work was supported by Ant Group.

Use of AI Assistants

The authors used AI assistants for language polishing, LaTeX editing, and phrasing suggestions during paper preparation. All substantive claims, experimental results, analyses, citations, and final text were reviewed and verified by the authors.

References

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

Janice Ahn, Rishu Verma, Renze Lou, Di Liu, Rui Zhang, and Wenpeng Yin. 2024. Large language models for mathematical reasoning: Progresses and challenges. *arXiv preprint arXiv:2402.00157*.

Bowen Baker, Joost Huizinga, Leo Gao, Zehao Dou, Melody Y Guan, Aleksander Madry, Wojciech Zaremba, Jakub Pachocki, and David Farhi. 2025. Monitoring reasoning models for misbehavior and the risks of promoting obfuscation. *arXiv preprint arXiv:2503.11926*.

Assaf Ben-Kish, Itamar Zimerman, M Jehanzeb Mirza, James Glass, Leonid Karlinsky, and Raja Giryes. 2025. Overflow prevention enhances long-context recurrent llms. *arXiv preprint arXiv:2505.07793*.

Paul C Bogdan, Uzay Macar, Neel Nanda, and Arthur Conmy. 2025. Thought anchors: Which llm reasoning steps matter? *arXiv preprint arXiv:2506.19143*.

Stephen Boyd and Lieven Vandenberghe. 2004. *Convex Optimization*. Cambridge University Press.

Faeze Brahman, Sachin Kumar, Vidhisha Balachandran, Pradeep Dasigi, Valentina Pyatkin, Abhilasha Ravichander, Sarah Wiegreffe, Nouha Dziri, Khyathi Chandu, Jack Hessel, et al. 2024. The art of saying no: Contextual noncompliance in language models. *Advances in Neural Information Processing Systems*, 37:49706–49748.

Qiguang Chen, Libo Qin, Jinhao Liu, Yue Liao, Jiaqi Wang, Jingxuan Zhou, and Wanxiang Che. 2025a. Rbf++: Quantifying and optimizing reasoning boundaries across measurable and unmeasurable capabilities for chain-of-thought reasoning. *arXiv preprint arXiv:2505.13307*.

Qiguang Chen, Libo Qin, Jinhao Liu, Dengyun Peng, Jiannan Guan, Peng Wang, Mengkang Hu, Yuhang Zhou, Te Gao, and Wanxiang Che. 2025b. Towards reasoning era: A survey of long chain-of-thought for reasoning large language models. *arXiv preprint arXiv:2503.09567*.

Qiguang Chen, Libo Qin, Jiaqi Wang, Jingxuan Zhou, and Wanxiang Che. 2024a. 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.

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

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

Yang Deng, Moxin Li, Liang Pang, Wenxuan Zhang, and Wai Lam. 2025. Unveiling knowledge boundary of large language models for trustworthy information access. In *Proceedings of the 48th International ACM SIGIR Conference on Research and Development in Information Retrieval*, pages 4086–4089.

Jianshuo Dong, Yutong Zhang, Liu Yan, Zhenyu Zhong, Tao Wei, Ke Xu, Minlie Huang, Chao Zhang, and Han Qiu. 2025. "i've decided to leak": Probing internals behind prompt leakage intents. In *Proceedings of the 2025 conference on empirical methods in natural language processing*, pages 21329–21359.

Siqi Fan, Xin Jiang, Xiang Li, Xuying Meng, Peng Han, Shuo Shang, Aixin Sun, Yequan Wang, and Zhongyuan Wang. 2024. Not all layers of llms are necessary during inference. *arXiv preprint arXiv:2403.02181*.

Sebastian Farquhar, Jannik Kossen, Lorenz Kuhn, and Yarin Gal. 2024. Detecting hallucinations in large language models using semantic entropy. *Nature*, 630(8017):625–630.

Yichao Fu, Junda Chen, Siqi Zhu, Zheyu Fu, Zhongdongming Dai, Yonghao Zhuang, Yian Ma, Aurick Qiao, Tajana Rosing, Ion Stoica, et al. 2024. Efficiently scaling llm reasoning with certaindex. *arXiv preprint arXiv:2412.20993*.

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

Yunfan Gao, Yun Xiong, Xinyu Gao, Kangxiang Jia, Jinliu Pan, Yuxi Bi, Yixin Dai, Jiawei Sun, Haofen Wang, and Haofen Wang. 2023. Retrieval-augmented generation for large language models: A survey. *arXiv preprint arXiv:2312.10997*, 2(1).

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

Tingxu Han, Zhenting Wang, Chunrong Fang, Shiyu Zhao, Shiqing Ma, and Zhenyu Chen. 2025. Token-budget-aware llm reasoning. In *Findings of the Association for Computational Linguistics: ACL 2025*, pages 24842–24855.

Trevor Hastie. 2009. The elements of statistical learning: data mining, inference, and prediction.

Yancheng He, Shilong Li, Jiaheng Liu, Weixun Wang, Xingyuan Bu, Ge Zhang, Zhongyuan Peng, Zhaoxiang Zhang, Zhicheng Zheng, Wenbo Su, et al. 2025. Can large language models detect errors in long chain-of-thought reasoning? *arXiv preprint arXiv:2502.19361*.

HMMT. 2025. Harvard–mit mathematics tournament: February 2025 problems archive. https://www.hmmt.org/www/archive/282.

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

Sehyun Hwang, Sohyun Lee, Hoyoung Kim, Minhyeon Oh, Jungseul Ok, and Suha Kwak. 2023. Active learning for semantic segmentation with multi-class label query. *Advances in Neural Information Processing Systems*, 36:27020–27039.

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

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

Saurav Kadavath, Tom Conerly, Amanda Askell, Tom Henighan, Dawn Drain, Ethan Perez, Nicholas Schiefer, Zac Hatfield-Dodds, Nova DasSarma, Eli Tran-Johnson, et al. 2022. Language models (mostly) know what they know. *arXiv preprint arXiv:2207.05221*.

Polina Kirichenko, Mark Ibrahim, Kamalika Chaudhuri, and Samuel J Bell. 2025. Abstentionbench: Reasoning llms fail on unanswerable questions. *arXiv preprint arXiv:2506.09038*.

Mario Köppen. 2000. The curse of dimensionality. In *5th online world conference on soft computing in industrial applications (WSC5)*, volume 1, pages 4–8.

Lorenz Kuhn, Yarin Gal, and Sebastian Farquhar. 2023. Semantic uncertainty: Linguistic invariances for uncertainty estimation in natural language generation. *arXiv preprint arXiv:2302.09664*.

Yury Kuratov, Aydar Bulatov, Petr Anokhin, Ivan Rodkin, Dmitry Sorokin, Artyom Sorokin, and Mikhail Burtsev. 2024. Babilong: Testing the limits of llms with long context reasoning-in-a-haystack. *Advances in Neural Information Processing Systems*, 37:106519–106554.

Patrick Lewis, Ethan Perez, Aleksandra Piktus, Fabio Petroni, Vladimir Karpukhin, Naman Goyal, Heinrich Küttler, Mike Lewis, Wen-tau Yih, Tim Rocktäschel, et al. 2020. Retrieval-augmented generation for knowledge-intensive nlp tasks. *Advances in neural information processing systems*, 33:9459–9474.

Moxin Li, Yong Zhao, Yang Deng, Wenxuan Zhang, Shuaiyi Li, Wenya Xie, See-Kiong Ng, and Tat-Seng Chua. 2024. Knowledge boundary of large language models: A survey. *arXiv preprint arXiv:2412.12472*.

Zhan Ling, Kang Liu, Kai Yan, Yifan Yang, Weijian Lin, Ting-Han Fan, Lingfeng Shen, Zhengyin Du, and Jiecao Chen. 2025. Longreason: A synthetic long-context reasoning benchmark via context expansion. *arXiv preprint arXiv:2501.15089*.

Jiayu Liu, Qing Zong, Weiqi Wang, and Yangqiu Song. 2025. Revisiting epistemic markers in confidence estimation: Can markers accurately reflect large language models' uncertainty? *arXiv preprint arXiv:2505.24778*.

Liangchen Luo, Yinxiao Liu, Rosanne Liu, Samrat Phatale, Meiqi Guo, Harsh Lara, Yunxuan Li, Lei Shu, Yun Zhu, Lei Meng, et al. 2024. Improve mathematical reasoning in language models by automated process supervision. *arXiv preprint arXiv:2406.06592*.

MAA. 2023. Amc 2023 10a problems and solutions. https://artofproblemsolving.com/wiki/index.php/2023_AMC_10A.

MAA. 2024. Aime 2024 i problems and solutions. https://artofproblemsolving.com/wiki/index.php/2024_AIME_I.

MAA. 2025. AIME 2025 I Problems and Solutions. https://artofproblemsolving.com/wiki/index.php/2025_AIME_I.

Nishanth Madhusudhan, Sathwik Tejaswi Madhusudhan, Vikas Yadav, and Masoud Hashemi. 2024. Do llms know when to not answer? investigating abstention abilities of large language models. *arXiv preprint arXiv:2407.16221*.

Sagnik Mukherjee, Abhinav Chinta, Takyoung Kim, Tarun Anoop Sharma, and Dilek Hakkani-Tür. 2025. Premise-augmented reasoning chains improve error identification in math reasoning with llms. *arXiv preprint arXiv:2502.02362*.

OpenAI. 2025. Gpt-5 system card. https://openai.com/index/gpt-5-system-card/.

Long Phan, Alice Gatti, Ziwen Han, Nathaniel Li, Josephina Hu, Hugh Zhang, Chen Bo Calvin Zhang, Mohamed Shaaban, John Ling, Sean Shi, et al. 2025. Humanity's last exam. *arXiv preprint arXiv:2501.14249*.

Reiner Pope, Sholto Douglas, Aakanksha Chowdhery, Jacob Devlin, James Bradbury, Jonathan Heek, Kefan Xiao, Shivani Agrawal, and Jeff Dean. 2023. Efficiently scaling transformer inference. *Proceedings of machine learning and systems*, 5:606–624.

Chen Qian, Dongrui Liu, Haochen Wen, Zhen Bai, Yong Liu, and Jing Shao. 2025. Demystifying reasoning dynamics with mutual information: Thinking tokens are information peaks in llm reasoning. *arXiv preprint arXiv:2506.02867*.

Qwen Team. 2024. Qwq-32b: A medium-sized reasoning model. https://huggingface.co/Qwen/QwQ-32B.

Alec Radford, Jeffrey Wu, Rewon Child, David Luan, Dario Amodei, Ilya Sutskever, et al. 2019. Language models are unsupervised multitask learners. *OpenAI blog*, 1(8):9.

David Rein, Betty Li Hou, Asa Cooper Stickland, Jackson Petty, Richard Yuanzhe Pang, Julien Dirani, Julian Michael, and Samuel R Bowman. 2023. Gpqa: A graduate-level google-proof q&a benchmark. *arXiv preprint arXiv:2311.12022*.

Charlie Snell, Jaehoon Lee, Kelvin Xu, and Aviral Kumar. 2024. Scaling llm test-time compute optimally can be more effective than scaling model parameters. *arXiv preprint arXiv:2408.03314*.

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

Jesse Vig. 2019. A multiscale visualization of attention in the transformer model. *arXiv preprint arXiv:1906.05714*.

Swapnil Vishwakarma. 2024. The curse of dimensionality in machine learning! *Analytics Vidhya*.

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

Huanjing Wang, Qianxin Liang, John T Hancock, and Taghi M Khoshgoftaar. 2024. Feature selection strategies: a comparative analysis of shap-value and importance-based methods. *Journal of Big Data*, 11(1):44.

Zhihua Wen, Zhiliang Tian, Zexin Jian, Zhen Huang, Pei Ke, Yifu Gao, Minlie Huang, and Dongsheng Li. 2024. Perception of knowledge boundary for large

language models through semi-open-ended question answering. *Advances in Neural Information Processing Systems*, 37:88906–88931.

Chris Wendler, Veniamin Veselovsky, Giovanni Monea, and Robert West. 2024. Do llamas work in english? on the latent language of multilingual transformers. In *Proceedings of the 62nd Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, pages 15366–15394.

Shijie Xia, Xuefeng Li, Yixin Liu, Tongshuang Wu, and Pengfei Liu. 2025. Evaluating mathematical reasoning beyond accuracy. In *Proceedings of the AAAI Conference on Artificial Intelligence*, volume 39, pages 27723–27730.

Boyang Xue, Qi Zhu, Rui Wang, Sheng Wang, Hongru Wang, Fei Mi, Yasheng Wang, Lifeng Shang, Qun Liu, and Kam-Fai Wong. 2025. Reliablemath: Benchmark of reliable mathematical reasoning on large language models. *arXiv preprint arXiv:2507.03133*.

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

Shu Yang, Junchao Wu, Xin Chen, Yunze Xiao, Xinyi Yang, Derek F Wong, and Di Wang. 2025b. Understanding aha moments: from external observations to internal mechanisms. *arXiv preprint arXiv:2504.02956*.

Junchi Yao, Shu Yang, Jianhua Xu, Lijie Hu, Mengdi Li, and Di Wang. 2025. Understanding the repeat curse in large language models from a feature perspective. *arXiv preprint arXiv:2504.14218*.

Xunjian Yin, Xu Zhang, Jie Ruan, and Xiaojun Wan. 2024. Benchmarking knowledge boundary for large language models: A different perspective on model evaluation. *arXiv preprint arXiv:2402.11493*.

Gal Yona, Roee Aharoni, and Mor Geva. 2024. Can large language models faithfully express their intrinsic uncertainty in words? *arXiv preprint arXiv:2405.16908*.

Zhihang Yuan, Yuzhang Shang, Yang Zhou, Zhen Dong, Zhe Zhou, Chenhao Xue, Bingzhe Wu, Zhikai Li, Qingyi Gu, Yong Jae Lee, et al. 2024. Llm inference unveiled: Survey and roofline model insights. *arXiv preprint arXiv:2402.16363*.

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

Qingjie Zhang, Han Qiu, Di Wang, Haoting Qian, Yiming Li, Tianwei Zhang, and Minlie Huang. 2024. Understanding the dark side of llms' intrinsic self-correction. *arXiv preprint arXiv:2412.14959*.

Zihao Zhao, Eric Wallace, Shi Feng, Dan Klein, and Sameer Singh. 2021. Calibrate before use: Improving few-shot performance of language models. In *International conference on machine learning*, pages 12697–12706. PMLR.

Yang Zhou, Hongyi Liu, Zhuoming Chen, Yuandong Tian, and Beidi Chen. 2025. Gsm-infinite: How do your llms behave over infinitely increasing context length and reasoning complexity? *arXiv preprint arXiv:2502.05252*.

A Datasets and Models

In this section, we provide the specific description of models and datasets evaluated in our work.

Table 4 summarizes the evaluation scope in the original submission. The approximately 120 questions refer only to the four competition-math benchmarks, rather than to the full evaluation.

Datasets	Task type	Size
AIME'25 / AIME'24 / HMMT / AMC'23	Competition math	120
GSM8K	Grade-school math	1.5k
HLE	Broad expert QA	1.5k
LiveCodeBench	Coding	1055

Table 4: Evaluation scope in the original submission. HLE uses text-only questions.

We evaluate the following LRMs:

- **GPT-oss-20B** (Agarwal et al., 2025): An open-source large reasoning model with 20B parameters, designed as a general-purpose backbone for multi-step reasoning tasks.
- **DeepSeek-R1-0528-Qwen3-8B** (Guo et al., 2025): A distilled variant of DeepSeek-R1 based on Qwen3-8B, post-trained with chain-of-thought supervision. It achieves state-of-the-art performance among open-source models on mathematical reasoning benchmarks.
- **DeepSeek-R1-Distill-Qwen-32B** (Guo et al., 2025): A 32B-parameter distilled model obtained by transferring reasoning abilities from DeepSeek-R1 into the Qwen2.5-32B backbone, yielding strong performance across benchmarks.
- **QwQ-32B** (Qwen Team, 2024): A medium-sized reasoning model from the Qwen series, competitive with other state-of-the-art reasoning-focused models on math and logical reasoning tasks.

For the sampling settings, we use greedy decoding to ensure reproducibility of results. For the maximum sequence length, we follow the specifications in the HuggingFace model cards: 128k for GPT-oss-20B, 64k for DeepSeek-R1-0528-Qwen3-8B, and 32k for both DeepSeek-R1-Distill-Qwen-32B and QwQ-32B.

We evaluate the following Math datasets:

- **AIME'25** (MAA, 2025): The 2025 American Invitational Mathematics Examination. It consists of 30 integer-answer questions with increasing difficulty, used as a challenging benchmark for mathematical reasoning.
- **AIME'24** (MAA, 2024): The 2024 American Invitational Mathematics Examination. Similar in format to AIME'25, it consists of 30 problems and provides a prior-year benchmark for evaluating model robustness across editions.
- **HMMT** (HMMT, 2025): The Harvard–MIT Mathematics Tournament (February 2025 session). The dataset contains 30 questions sourced from the original competition, which were extracted, converted to LaTeX, and verified. These problems feature complex multi-step reasoning across various mathematical domains.
- **AMC'23** (MAA, 2023): The 2023 American Mathematics Competition (AMC 10A). The dataset consists of 30 questions, providing a lighter but diverse benchmark compared to AIME and HMMT.
- **GSM8K** (Cobbe et al., 2021): A large-scale dataset of grade-school math word problems requiring multi-step arithmetic reasoning. We evaluate on approximately 1.5k problems from the test set. It is widely used to evaluate the basic reasoning competence of LRMs.
- **Humanity's Last Exam (HLE)** (Phan et al., 2025): A multi-modal benchmark at the frontier of human knowledge, designed with broad subject coverage across mathematics, humanities, and natural sciences. We use approximately 1.5k text-only questions (excluding image-based problems) from this dataset.

The evaluated benchmarks also differ substantially in question length, mathematical notation, answer-format requirements, and prompt openings. Table 5 reports these surface-style indicators, showing that $\text{Monitor}_{\text{express}}$ is not evaluated on a single fixed question template.

B Limitations of Some Prior Work

Some prior works evaluate model confidence through self-verification (Kadavath et al., 2022) or uncertainty estimation (Farquhar et al., 2024; Kuhn et al., 2023). In this section, we show that these approaches are less effective than our proposed monitoring strategies for identifying unsolvable questions with comparable accuracy on solvable ones. We first clarify how we adapt these methods to fit our task.

Self-verification (SV). For each question, we ask the model to evaluate whether its own predicted answer is correct using the prompt template:

Benchmark	Avg. words	Math spans/Q	Math-char (%)	Answer-transform (%)	Direct-start (%)
AIME'24	63.6	6.8	19.0	30.0	13.3
AIME'25	87.2	10.2	21.8	31.0	10.3
AMC'23	53.7	5.7	20.7	30.8	23.1
HMMT Feb. 2025	63.7	6.5	26.2	0.0	17.2
GSM8K	46.8	0.2	5.2	62.9	0.1
HLE text-only	111.3	7.5	28.6	80.6	11.6

Table 5: Prompt-style diversity in the evaluated benchmarks.

Self-verification prompt

Question: {question}
Proposed Answer: {answer}
Is the proposed answer:
A True
B False
The proposed answer is:

This follows prior work (Kadavath et al., 2022) which study whether language models can evaluate the validity of their own claims by asking models to first propose answers, and then to evaluate that their answers are correct. If the model judges its own incorrect answer to be incorrect, we count this as a hard abstention (HA). Conversely, if the model judges its correct answer to be incorrect, this reduces the overall task accuracy (Acc). Formally,

$$\text{Acc} = \mathbb{P}(\text{Model judges the answer to be correct} \wedge \text{Answer is correct}) \tag{4}$$

$$\text{HA} = \mathbb{P}(\text{Model judges the answer to be incorrect} \mid \text{Answer is incorrect}) \tag{5}$$

Uncertainty-estimation (UE). For each question, we compute the semantic entropy of the model's generated answer, where a higher entropy indicates greater uncertainty. This follows prior work (Farquhar et al., 2024; Kuhn et al., 2023), which introduces semantic entropy as an uncertainty measure that captures linguistic invariances induced by shared meanings. When the entropy exceeds a certain threshold, the model is considered uncertain about its answer. Similar as before, we define:

$$\text{Acc} = \mathbb{P}(\text{Semantic entropy of answer} < \text{threshold} \wedge \text{Answer is correct}) \tag{6}$$

$$\text{HA} = \mathbb{P}(\text{Semantic entropy of answer} \geq \text{threshold} \mid \text{Answer is incorrect}) \tag{7}$$

We additionally compare with BudgetPrompt (Han et al., 2025), which constrains the reasoning budget through prompting, and Dynamic-Early-Exit (Yang et al., 2025a), which stops reasoning dynamically.

Method	ACC ↑	HA ↑	FPA ↓
GPT-oss-20B	74.6	0.0	0.0
+BoostAbstention	74.6	0.0	0.0
+BudgetPrompt	46.7	0.0	37.4
+Self-verification	32.1	46.0	57.0
+Uncertainty estimation	50.0	29.0	33.0
+Dynamic-Early-Exit	49.8	0.0	33.2
+Monitor$_{\text{express}}$	**74.6**	**100.0**	**0.0**
+Monitor$_{\text{hidden}}$	73.7	98.9	1.2

Table 6: Comparison to five baselines on GPT-oss-20B (%). FPA denotes false-positive abstention, $P(\text{abstain} \mid \text{originally correct})$.

Results. Table 6 shows that BoostAbstention, BudgetPrompt, and Dynamic-Early-Exit do not identify unsolvable questions (HA = 0). Self-verification and uncertainty estimation provide some HA only while sacrificing many originally correct cases. In contrast, our monitors preserve ACC while achieving high HA and low FPA.

C Confident and Uncertain Expressions

To ensure that our analysis is not tied to particular LRM's reasoning style, we extract confident and uncertain expressions individually for each model, based on its own characteristic reasoning patterns. The extraction pipeline is outlined below.

Step 1: Identify all anthropomorphic expressions from the model's reasoning trace. Prior work (Guo et al., 2025; Qian et al., 2025) show that the anthropomorphic expressions (e.g., "wait") are crucial for LRM reasoning performance, in contrast to purely neutral mathematical steps (e.g., "11 + 5 = 16"). Following this insight, we curate a set of anthropomorphic expressions that the LRM commonly uses. And we further verify with GPT-5 that these expressions indeed convey anthropomorphic tone rather than purely propositional content.

Step 2: Select expressions whose occurrence frequency is most predictive of answer correctness. For each question, we count the occurrences

yes, perfect, good, great, right, correctly, True, indeed, no mistake, don't think I made any mistakes, don't see any mistakes, can't see any mistakes, didn't make any mistakes, no doubt, excellent, no issue, stick with my answer, acceptable, correct, go with, feel confident, can be confident, I'm confident, I am confident, so confident that, confidently, pretty confident, more confident, definitely, reasonable, that's solid, fine, perfect answer, exactly, seems consistent, consistent solution, I think that's the answer, I think that's it, I think that is it, method is solid, its solid, seems solid, approach is solid, calculation is solid, the way I did it is solid, this is solid, any errors, any error, both solutions satisfy, they satisfy, that satisfy, satisfied, this is the only solution, it seems like this is the only solution, suggests that this is the only solution, valid, convinced, Therefore, must be, I believe, I think this is the answer, I think that is the answer, no problem, the answer should be, seems okay, satisfies, nice, plausible, Lets trust, easy, same as before

Confident Expressions

a mistake, a fundamental mistake, my mistake, mistakenly, miscalculated, missed, wrong, messed up, incorrect, incorrectly, not correct, messy, complicated, convoluted, no!, abandon, misinterpreted, misinterpret, overthinking, not good, error, incorrect, failed, confused, unclear, misunderstanding, misunderstood, problematic, same issue, the issue is, difficulty, difficult, need to stop, will stop, should stop, too confident, not confident, must have an error, I made an error, must be an error, must be missing, must be wrong, must be mistake, I see the error, I found the error, I made a calculation error, I had a calculation error, must have made an error, mixed up, doesn't make sense, misunderstood, misunderstand, that's bad, even worse, that's worse, flawed, contradicts, contradiction, no solution, no solutions, invalid, not satisfy, cannot satisfy, can't satisfy, can't be right, not satisfying, not satisfied, not a valid, give up, I apologize, not true, didn't work, sorry, I'm stuck, Given complexity, failure, time limited, not nice, Need more precise, So not, maybe not, ? Not, no easy, not easy, not 100% sure, can't confirm, might, the best I can do, unsure, guess, tentatively, hard to say, not entirely sure, not certain, not absolutely certain, approximately, don't know, not entirely confident, not confident, not solid, more or less, wait, not sure, suspect, the answer might be, Probably not, give approximate, likely, confusion, confusing, not exact, uncertainty, Given approximate, Given complexity, might be possible, is that true, is that correct, is that possible, is that right, is that sufficient, hmm

Uncertain Expressions

Figure 7: Confident vs. uncertain expressions.

of all extracted expressions within reasoning. Using the final answer correctness as the label and the occurrence counts for each expressions as features, we apply feature-importance analysis methods (Wang et al., 2024), including SHAP and Permutation Importance (PI), to identify the subset of expressions most informative for predicting whether the model's final answer is correct. As shown in Table 7 (an example for GPT-oss-20B), we select expressions such as "wait", "yes", and "exactly" which rank among the most predictive, whereas others (e.g., "so") exhibit negligible predictive value and are therefore excluded.

Expression	SHAP ↑	PI ↑	Total ↑
wait	1.00	1.00	2.00
yes	0.33	0.73	1.06
exactly	0.26	0.73	0.99
...			
so	0.01	0.00	0.01

Table 7: Feature importance score of expressions (shown here for GPT-oss-20B).

Step 3: Divide the selected expressions into confidence and uncertainty categories. This step is grounded in the connection between epistemic markers and expressed uncertainty. Prior work (Liu et al., 2025; Yona et al., 2024) show that epistemic markers reflect LLMs' internal confidence, analogous to how humans use expressions such as "I am not sure" or "it is unlikely that" to convey uncertainty. Following this insight, we classify each selected expression as reflecting confidence or uncertainty (further validated by GPT-5).

Figure 7 shows the full list of expressions of all LRMs. When matching these expressions, the *Uncertain Expression* is matched first, followed by the *Confident Expression*. Additionally, both the position of the first token of an expression and the token length of the entire expression are considered.

We estimate the model's confidence level from the occurrences of confidence and uncertainty expressions throughout the reasoning process, following prior work (Liu et al., 2025) which measures LLM uncertainty by counting the frequency

of epistemic markers. In Section 3, we use the category of expressions that appears more frequently near the end of the reasoning trace to determine whether the model is more confident or more uncertain. Building on this idea, Section 4 analyzes the trajectory of confidence and uncertainty signals over the entire reasoning sequence, captured by ConfDiff (Equation 2) and ConfCurv (Equation 3), to assess whether the model is more confident or more uncertain.

Phrase-list sensitivity. We compare the full phrase-construction pipeline with three alternatives: removing generic expressions, retaining only top-ranked expressions, and randomly dropping half of the expressions. For each model, we tune the operating point using the full expression signal and apply the same operating point to the ablated phrase sets. Table 8 shows that the full pipeline performs best on average, while the ablated lists remain predictive but weaker.

Model	Remove generic	Top-ranked	Random half	Ours
GPT-oss-20B	81.1	85.0	80.6	**85.0**
R1-Distill-Qwen3-8B	75.4	71.5	51.1	**82.3**
R1-Distill-Qwen-32B	75.4	83.1	71.8	**89.2**
QwQ-32B	66.9	78.5	65.4	**83.8**
Average	74.7	79.5	67.2	**85.1**

Table 8: Phrase-list sensitivity across models (separating accuracy, %).

Threshold tuning. We use a strict train/validation/test split: the training split constructs the expression signals, the validation split tunes the stage-wise thresholds, and the test split is used only for final evaluation. For each stage s, ConfDiff is computed as

$$R_{i,s}^{\text{diff}} = \frac{1}{s}\sum_{t=1}^{s} \mathbf{1}\left[\mathcal{D}_U^i(t) > \mathcal{D}_C^i(t)\right], \tag{8}$$

and α_s is selected on the validation split by

$$\alpha_s^* = \arg\max_{\alpha} \frac{1}{2}\big[\Pr(R_{i,s}^{\text{diff}} > \alpha \mid y_i = 1) + \Pr(R_{i,s}^{\text{diff}} \le \alpha \mid y_i = 0)\big], \tag{9}$$

where $y_i = 1$ means that the original answer is incorrect. Likewise,

$$R_{i,s}^{\text{curv}} = \frac{1}{s-1}\sum_{t=1}^{s-1} \mathbf{1}\left[\Delta^2(\mathcal{D}_U^i - \mathcal{D}_C^i)(t) < 0\right], \tag{10}$$

and β_s is selected by the same balanced-accuracy criterion:

$$\beta_s^* = \arg\max_{\beta} \frac{1}{2}\big[\Pr(R_{i,s}^{\text{curv}} > \beta \mid y_i = 1) + \Pr(R_{i,s}^{\text{curv}} \le \beta \mid y_i = 0)\big]. \tag{11}$$

The selected thresholds are frozen before evaluating the test split.

Table 9 reports a strict-split diagnostic. For ConfCurv and ConfDiff, expression lists are built on train and thresholds are tuned on validation. For hidden states, LR and LDA are trained on train, with LR regularization tuned on validation. The strict-split accuracy scores remain close to the original diagnostics.

Signal/classifier	Original Acc.	Strict-split Acc.
ConfCurv	86.9	85.1
ConfDiff	86.2	86.2
LR	98.0	97.8
LDA	97.9	97.8

Table 9: Strict train/validation/test split diagnostic (accuracy, %).

D System Prompt of BoostAbstention

We consider BoostAbstention (Kirichenko et al., 2025) which crafts a system prompt that encourages the model to abstain in designated scenarios, showing effectiveness for both reasoning and standard LLMs. For our experiments, we modify the system prompt to better match the abstention scenarios of questions beyond capability boundary:

System Prompt of BoostAbstention

You are a helpful assistant specialized in solving math problems.

However, some problems may be too complex or beyond your capability. In such cases:
- Admit that you cannot fully solve the problem.
- Provide a concise potential approach, outline, or next step instead of a detailed solution.
- Avoid spending too much time on overly difficult problems, as this increases latency for the user.

E Ablation Study on Output Prefix

During our initial exploration, we experiment with several output prefixes and select the one that achieves the largest reduction in token usage. We conduct an ablation study comparing:

- **Initial prefix (our early attempt)**: Our goal is to prompt the model to acknowledge that it cannot solve an unsolvable question and then provide only a concise potential approach. However, we unexpectedly found that this cannot minimize token usage. The model's instruction-following ability was insufficient for reliably producing only a concise approach, it often generated overly long responses. The initial prefix is shown below:

Initial output prefix

```
<think>
I think this question is beyond my capability boundary. I cannot fully solve it, but I can outline a concise potential approach.
</think>
This question is beyond my capability boundary, but I can outline a concise potential approach:
```

- **Emphasis prefix (final choice)**: Inspired by prior work (Zhang et al., 2024; Zhao et al., 2021) which show that the final sentence in the instruction strongly influences model behavior, we appended `Step 1:` at the end of the instruction to explicitly trigger step-by-step approach. Also, we emphasize in the end of the `<think>...</think>` part that `I must give a concise outline to the user (less than 10 steps)!`. This combination elicits short, structured responses and lead to the largest reduction in token usage. This is the output prefix we ultimately choose:

Emphasis output prefix

```
<think>
I think this question is beyond my capability boundary. I cannot fully solve it, but I can outline a concise potential approach. I must give a concise outline to the user (less than 10 steps)!
</think>
This question is beyond my capability boundary, but I can outline a concise potential approach:
Step 1:
```

- **Token-budget prefix (following prior work)**: Following prior work (Han et al., 2025) which compresses unnecessary reasoning by including a reasonable token budget in the prompt, we replace `less than 10 steps` with an explicit budget constraint `use less than 500 tokens`. The token-budget prefix is:

Token-budget output prefix

```
<think>
I think this question is beyond my capability boundary. I cannot fully solve it, but I can outline a concise potential approach. I must give a concise outline to the user (less than 500 tokens)!
</think>
This question is beyond my capability boundary, but I can outline a concise potential approach:
Step 1:
```

To avoid the confounding effect of incomplete answers, we set the context length to the maximum according to Huggingface model card.

Table 10 shows that *all three designs reduce token usage, with the Emphasis prefix achieving the largest reduction*. While it is possible that an even better prompt design exists, this does not affect our core conclusion: *setting an output prefix enables the model to provide more efficient and more reliable responses on unsolvable questions*.

Output prefix	Token ↓
No output prefix	5096
Initial	4423
Emphasis	**2227**
Token-budget	2567

Table 10: Token usage under different output prefixes.

F Additional Analysis of Capability Boundary Signals in Hidden States

In Section 5, we show that eventually solved and unsolved questions can be separated using signals extracted from the hidden states of the last input token. However, these signals may partially correlate with question difficulty rather than reflecting a purely intrinsic notion of capability boundary.

In this section, we do not aim to establish a clean separation between capability boundary signals and question difficulty. Instead, we provide additional analyses showing that the hidden state signals identified in our setup remain informative under stricter per-benchmark and cross-benchmark settings.

We also compare the hidden-state classifier with controls based on benchmark/source identity and text difficulty proxies, including surface features such as length, format, and notation density. Table 11 shows that these controls carry some signal, but are substantially weaker than the last-token hidden states. Together with the held-out results in Table 9, this supports the narrower interpretation that the hidden states provide predictive information beyond the tested metadata and text proxies.

Model	Benchmark/ source	Text difficulty proxies	Last-token hidden states
GPT-oss-20B	63.4	55.6	**98.0**
R1-Distill-Qwen3-8B	66.7	61.1	**96.7**
R1-Distill-Qwen-32B	63.6	60.9	**97.8**
QwQ-32B	66.5	71.6	**98.9**

Table 11: Control classifier comparison across models (separating accuracy, %).

An intuition from human exams. One possible intuition comes from human test taking. In an exam with limited time and many difficult problems, skilled test-takers may quickly identify questions that are unlikely to be solved within the available budget and avoid wasting excessive time on them. This judgment may come either from recognizing that a question resembles problems they have repeatedly struggled with in the past (analogous to our $\text{Monitor}_{\text{hidden}}$ strategy), or from taking a few initial steps and realizing that they lack confidence to proceed (analogous to our $\text{Monitor}_{\text{express}}$ strategy). We present this analogy only as intuition, not as evidence of human-like metacognitive awareness in LRMs. Motivated by this intuition, we further conduct per-benchmark and cross-benchmark analyses as follows.

Dataset	Model	LDA	LR
HLE	GPT-oss-20B	89.3	90.3
	R1-Distill-Qwen3-8B	94.9	96.4
	R1-Distill-Qwen-32B	96.4	96.4
	QwQ-32B	93.4	96.4
GSM8k	GPT-oss-20B	80.3	83.3
	R1-Distill-Qwen3-8B	70.1	71.6
	R1-Distill-Qwen-32B	85.2	89.0
	QwQ-32B	87.9	87.1

Table 12: Accuracy (%) of separating solvable and unsolvable questions using linear classifiers trained and tested within the same benchmark.

Per-benchmark test. We provide in Table 12 the accuracy of separating unsolvable questions from solvable ones *using LDA and LR classifiers trained and tested on splits from the same dataset*. Specifically, for HLE and GSM8K, we randomly split each dataset into an 80% training set and a 20% test set (with balanced solvable/unsolvable labels), train the linear classifier on the training set, and evaluate its classification accuracy on the test set.

The results show that *linear classifiers can still separate solvable and unsolvable questions from hidden states, although with reduced accuracy compared to aggregated benchmarks setting (Table 2)*. This drop is small on HLE but larger on GSM8K. The reason is that *HLE contains a larger proportion of unsolvable questions, enabling the linear classifiers to better learn the hidden state information associated with unsolvability*. Consequently, these unsolvable questions in HLE can be effectively leveraged to develop models' Hard Abstention (HA) capability.

Cross-benchmark test. In the above analysis, we think the unsolvable questions in HLE can be effectively leveraged to develop models' Hard Abstention (HA) capability. Therefore, we train linear classifiers on HLE and evaluate models' Hard Abstention capability on GSM8k and AIME to test whether this capability generalizes across benchmarks. As shown in Table 13, linear classifiers trained on HLE are able to identify nearly all unsolvable questions in GSM8K and AIME, achieving HA close to 100%.

Model	GSM8k	AIME
GPT-oss-20B	0	0
+$\text{Monitor}_{\text{hidden}}$	100	100
R1-Distill-Qwen3-8B	0	0
+$\text{Monitor}_{\text{hidden}}$	100	100
R1-Distill-Qwen-32B	0	0
+$\text{Monitor}_{\text{hidden}}$	90.5	94.7
QwQ-32B	0	0
+$\text{Monitor}_{\text{hidden}}$	98.9	100

Table 13: Hard Abstention (%) of linear classifiers trained on HLE and tested on GSM8k and AIME.

In contrast, training on GSM8K is insufficient for developing Hard Abstention capability, because GSM8K contains very few unsolvable questions. The linear classifiers cannot reliably learn the hidden state patterns associated with unsolvability.

G Full-Dataset Trade-offs and Output Utility

Full-dataset trade-offs. The main results report ACC over all questions and the remaining metrics over originally incorrect questions. For completeness, Table 14 reports ACC over all questions, HA $= P(\text{abstain} \mid \text{originally incorrect})$, FPA $= P(\text{abstain} \mid \text{originally correct})$, and average token usage over all questions. The monitors identify many originally incorrect cases while introducing few false-positive abstentions, and their full-dataset token reduction remains substantial.

Output utility. We further evaluate whether the concise responses remain useful when a model is likely to fail. Following the example in Figure 6,

Model	Method	ACC ↑	HA ↑	FPA ↓	Token (all) ↓ 2048	Token (all) ↓ 4096
GPT-oss-20B	Original	74.6	0.0	0.0	1813	3071
	+BoostAbstention	74.6 (+0.0)	0.0 (+0.0)	0.0 (+0.0)	1811 (-0.1%)	3050 (-0.7%)
	+Monitor$_{\text{express}}$	**74.6 (+0.0)**	**100.0 (+100.0)**	**0.0 (+0.0)**	**1586 (-12.5%)**	**2432 (-20.8%)**
	+Monitor$_{\text{hidden}}$	73.7 (-0.9)	98.9 (+98.9)	1.2 (+1.2)	1750 (-3.5%)	2893 (-5.8%)
R1-Distill-Qwen3-8B	Original	76.9	0.0	0.0	2040	4030
	+BoostAbstention	76.9 (+0.0)	0.0 (+0.0)	0.0 (+0.0)	2040 (-0.0%)	4030 (-0.0%)
	+Monitor$_{\text{express}}$	74.6 (-2.3)	63.3 (+63.3)	3.0 (+3.0)	1903 (-6.7%)	3673 (-8.9%)
	+Monitor$_{\text{hidden}}$	73.8 (-3.1)	**97.9 (+97.9)**	4.0 (+4.0)	**1748 (-14.3%)**	**3371 (-16.4%)**
R1-Distill-Qwen-32B	Original	56.2	0.0	0.0	1993	3649
	+BoostAbstention	56.2 (+0.0)	0.0 (+0.0)	0.0 (+0.0)	1974 (-1.0%)	3565 (-2.3%)
	+Monitor$_{\text{express}}$	**56.2 (+0.0)**	**100.0 (+100.0)**	**0.0 (+0.0)**	1447 (-27.4%)	2207 (-39.5%)
	+Monitor$_{\text{hidden}}$	55.3 (-0.9)	97.0 (+97.0)	1.6 (+1.6)	**1296 (-35.0%)**	**2103 (-42.4%)**
QwQ-32B	Original	71.5	0.0	0.0	2037	3917
	+BoostAbstention	71.5 (+0.0)	0.0 (+0.0)	0.0 (+0.0)	2037 (-0.0%)	3917 (-0.0%)
	+Monitor$_{\text{express}}$	**71.5 (+0.0)**	**100.0 (+100.0)**	**0.0 (+0.0)**	1785 (-12.4%)	3271 (-16.5%)
	+Monitor$_{\text{hidden}}$	**71.5 (+0.0)**	96.9 (+96.9)	**0.0 (+0.0)**	**1528 (-25.0%)**	**2825 (-27.9%)**

Table 14: Full-dataset accuracy–abstention–token trade-off (%). Parentheses show changes from the original model; token columns report averages over all questions.

we use GPT-5.5 as a rubric-based judge to compare the original long response with the concise response from the monitor on four five-point metrics: relevance, non-misleadingness, helpfulness, and readability. Table 15 shows that the concise responses provide more reliable and readable guidance, while preserving relevance.

H Analysis Beyond Mathematical Tasks

Since we capture unsolvability signals from reasoning expressions (Section 4) and hidden states (Section 5) without relying on task-specific heuristics, they might generalize to other reasoning tasks. We evaluate this on LiveCodeBench and GPQA-Diamond.

H.1 LiveCodeBench

We conduct experiments on 1055 samples from LiveCodeBench (Jain et al., 2024). Table 16 shows the accuracy of separating solvable and unsolvable coding questions based on capability boundaries in reasoning expressions and hidden states.

We observe that *coding tasks also exhibit clear capability boundaries in both reasoning expressions and hidden states*. However, the accuracy from hidden states is lower than that from mathematical tasks. A plausible explanation is that the difficulty of coding problems is less explicitly signaled in the input itself. Unlike in math exams where humans can often immediately recognize "end-of-exam" questions that they are unlikely to solve, coding tasks usually require some degree of initial reasoning before the model can detect unsolvability. As a result, the unsolvability signal in the hidden states of the last input token is weaker, whereas the signal becomes more apparent in the early reasoning expressions.

H.2 GPQA-Diamond

We additionally test whether the same signals separate originally correct from incorrect cases on GPQA-Diamond (Rein et al., 2023). Table 17 shows that the signal remains observable, although it is weaker than on the math tasks. GPQA is multiple-choice, so guessing can make final correctness a noisier proxy for genuine solvability than free-form math.

Model	ConfDiff	ConfCurv	LDA	LR
GPT-oss-20B	71.2	60.0	81.2	87.5
R1-Distill-Qwen3-8B	87.5	91.2	93.8	87.5
R1-Distill-Qwen-32B	77.5	77.5	87.5	87.5
QwQ-32B	78.8	77.5	81.2	87.5

Table 17: GPQA-Diamond correct/incorrect separation accuracy (%).

I Monitoring Strategies with Maximum Context Length

To evaluate whether our monitoring strategies (Monitor$_{\text{express}}$ and Monitor$_{\text{hidden}}$) hold promise for optimizing long-context reasoning tasks (Ling et al., 2025; Kuratov et al., 2024), we provide the optimized token usage *under extended token budget, set to the maximum* according to Hugging-

Model	Output	Relevance ↑	Non-misleadingness ↑	Helpfulness ↑	Readability ↑
GPT-oss-20B	Original	4.0	2.2	2.6	1.4
	+Monitor	4.0	4.8	2.2	4.8
R1-Distill-Qwen3-8B	Original	4.0	2.4	2.2	1.2
	+Monitor	4.0	4.0	3.8	3.6
R1-Distill-Qwen-32B	Original	4.2	2.6	2.4	1.6
	+Monitor	4.4	4.2	4.2	4.6
QwQ-32B	Original	4.0	2.4	2.0	1.2
	+Monitor	4.2	3.6	4.0	4.6

Table 15: Concise-output utility evaluation on a five-point scale.

Model	Reasoning Expressions		Hidden States	
	ConfDiff	ConfCurv	LDA	LR
GPT-oss-20B	100	86.8	81.1	83.5
R1-Distill-Qwen3-8B	97.3	93.0	92.0	93.9
R1-Distill-Qwen-32B	100	91.7	81.6	84.9
QwQ-32B	100	92.5	77.3	85.4

Table 16: Accuracy (%) of separating eventually solved and unsolved questions using signals related to the capability boundary, derived from reasoning expressions (at 2% of the reasoning trace) and hidden states.

Face model card (128k for GPT-oss-20B, 64k for R1-Distill-Qwen3-8B, and 32k for both R1-Distill-Qwen-32B and QwQ-32B).

	Context Length	Token ↓	Overflow (%) ↓
GPT-oss-20B	128k	6023	0
+Monitor$_{\text{express}}$		1897▾68.5	0
+Monitor$_{\text{hidden}}$		3826▾36.5	0
R1-Distill-Qwen3-8B	64k	5096	0
+Monitor$_{\text{express}}$		2834▾44.4	0
+Monitor$_{\text{hidden}}$		2227▾56.3	0
R1-Distill-Qwen-32B	32k	5910	0
+Monitor$_{\text{express}}$		1043▾82.4	0
+Monitor$_{\text{hidden}}$		1246▾78.9	0
QwQ-32B	32k	5062	0
+Monitor$_{\text{express}}$		2002▾60.5	0
+Monitor$_{\text{hidden}}$		274▾94.6	0

Table 18: Monitoring strategies (Monitor$_{\text{express}}$ and Monitor$_{\text{hidden}}$) with maximum context length.

Under these settings, Table 18 shows that the models always produce complete answers (Overflow = 0), and our monitoring strategies still *reduce token usage significantly as expected* (Acc and HA are the same as under 2k or 4k context length).

Moreover, it is worth noting that *our strategies are not constrained by context length*:

- Monitor$_{\text{express}}$ continuously tracks the densities of confident and uncertain expressions throughout the reasoning process and can detect unsolvable questions *at an early stage* (Figure 3), well before the model reaches the end of reasoning.
- Monitor$_{\text{hidden}}$ identifies unsolvable questions directly from the hidden states of last input token, i.e., *even before the model begins reasoning*.

Because our strategies operate early in the reasoning trajectory, we intentionally used relatively short context length (e.g., 2k or 4k) in Section 6 to demonstrate their early detection capabilities.

J Capability Boundary in Hidden States

Figure 8 visualizes the classification of solvable and unsolvable questions of all LRMs, along with the capability boundary (blue line) determined by the decision boundary of linear classifier (e.g., LDA). We observe that the two classes of questions are clearly separated, under random train–test splits. These results indicate that the hidden states of the last input token contain useful signals related to the operational capability boundary, and is valid across different LRMs.

K Unproductive Reasoning

Figure 9 and Figure 10 show representative unproductive reasoning phenomena in LRMs, including repetitive looping and error accumulation, which impair their capacity for sound and consistent reasoning in complex problem-solving scenarios. They respectively show how models get stuck in repetitive deductions (Figure 9) and how errors compound during formula application and result derivation (Figure 10). In mathematical problem reasoning, the former presents repeated deduction of the same uncertain conclusion, while the latter sees continuous error accumulation due to flawed initial methods.

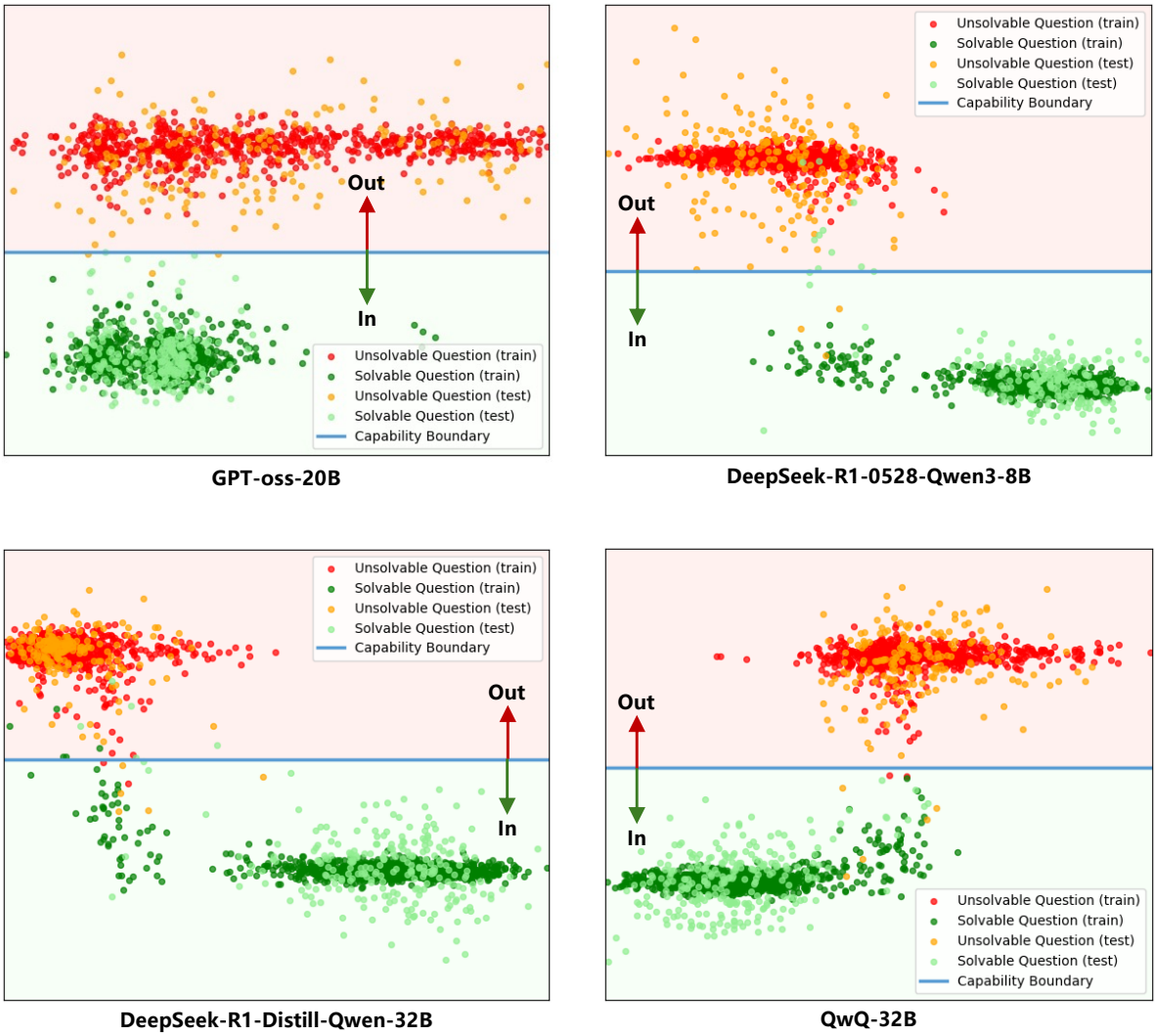


Figure 8: A linear classifier (e.g., LDA) clearly separates eventually solved and unsolved questions using hidden state signals related to the operational capability boundary.

Question from AIME24:

Let \(b\ge 2\) be an integer. Call a positive integer \(n\) \(b\text-\textit{eautiful}\) if it has exactly two digits when expressed in base \(b\) and these two digits sum to \(\sqrt n\). For example, \(81\) is \(13\text-\textit{eautiful}\) because \(81 = \underline{6}\ \underline{3}_{13} \) and \(6 + 3 = \sqrt{81}\). Find the least integer \(b\ge 2\) for which there are more than ten \(b\text-\textit{eautiful}\) integers.

Answer from QwQ-32B:

Restating the Problem

Okay, so I need to find the least integer b (where b is at least 2) such that there are more than ten b-beautiful numbers. A b-beautiful number is a two-digit number in base b, and the sum of its digits equals the square root of the number itself.

Uncertain Attempts

..... Wait, something is **wrong** here.
.....
Wait, perhaps my formula is **wrong**?
Wait, let's go back.
.....

Hmm.
Wait, perhaps the minimal b is 121 with 7 solutions, but the problem requires more than ten.
Maybe the answer is 121, but I'm not sure.
Alternatively, perhaps the minimal b is 256 with 8 solutions.
Alternatively, maybe the minimal b is 100 with 3 solutions, but the problem requires more than ten. I must be missing something.
Wait, perhaps I made a mistake in the earlier examples. Let me try b= 25 (k=24):
S from5 to24.
S=9:9*8=72 divisible by24? 72/24=3. Yes.
S=16:16*15=240 divisible by24? 240/24=10. Yes.
S=24:24*23=552 divisible by24? 552/24=23. Yes.
Thus count=3.

Repetitive Looping

Hmm.
Wait, perhaps the minimal b is 121 with 7 solutions, but the problem requires more than ten.
Maybe the answer is 121, but I'm not sure.
.....
Hmm.
Wait, perhaps the minimal b is 121 with 7 solutions, but the problem requires more than ten.
Maybe the answer is 121, but I'm not sure.
.....
Hmm.
Wait, perhaps the minimal b is 121 with 7 solutions, but the problem requires more than ten.
Maybe the answer is 121, but I'm not sure.
.....
Hmm.
Wait, perhaps the minimal b is 121 with 7 solutions, but the problem requires more than ten.
Maybe the answer is 121, but I'm not sure.
.....
Hmm.
Wait, perhaps the minimal b is 121 with 7 solutions, but the problem requires more than ten.
Maybe the answer is 121, but I'm not sure.
.....

Figure 9: An example of unproductive reasoning manifested as repetitive looping in LRMs when attempting to solve a mathematical problem, where the model repeatedly cycles through the same uncertain deductions without making progress.

Question from AIME25:

Alex divides a disk into four quadrants with two perpendicular diameters intersecting at the center of the disk. He draws 25 more lines segments through the disk, drawing each segment by selecting two points at random on the perimeter of the disk in different quadrants and connecting these two points. Find the expected number of regions into which these 27 line segments divide the disk.

Answer from DeepSeek-R1-Distill-Qwen-32B :

Restating the Problem

Okay, so I have this problem where Alex divides a disk into four quadrants using two perpendicular diameters .....

Initial Formula

So, the formula would be:
E[Regions] = 1 + n + E[Intersections]
Where n is the number of chords, which is 27 in this case (2 diameters + 25 chords).
.....

Incorrect Initial Formula Application (Ignoring Pre-Existing Regions)

Error Accumulation

.....
But according to the formula, it's 1/3, which is clearly **wrong.**
Wait, this suggests that **my earlier computation is incorrect.**
.....
But according to the formula, it's 1 + 2/3 ≈ 1.666, which is **incorrect.**
Therefore, my approach is still **flawed**.
.....
Wait, let me recompute.
.....
Wait, this is getting **too confusing.**
.....
Alternatively, perhaps .....
Therefore, the expected number of regions is 1 + 27 + 351*(1/9) = 1 + 27 + 39 = 67.
But in the n=3 case, this formula gives 1 + 3 + 3*(1/9) = 4 + 1/3 ≈ 4.333, which is **incorrect.**
.....
But the problem might expect an integer, so perhaps my approach is still **incorrect.**
Wait, ...
Therefore, the formula is **incorrect.**
.....

Forced Conclusion Despite Unresolved Inconsistencies

Guessing Final Answer

But in the n=3 case, this gives 1 + 3 + 3*(1/9) = 4 + 1/3 ≈ 4.333, which is **incorrect.**
Therefore, the formula is not applicable here.
I think the correct answer is 67, but **I'm not entirely sure.**
Alternatively, perhaps the expected number of regions is 1 + 27 + 39 = 67.
Therefore, the final answer is 67.
But I'm not entirely confident due to the inconsistencies in smaller cases.
But **given the time** I've spent, I think the answer is 67.

Figure 10: An example of error accumulation in LRMs when tackling a problem on the expected number of regions in a disk, where the model accumulates errors due to an incorrect initial formula, leading to inconsistent results and a forced answer guess despite unresolved contradictions.