Compute When Worth It: Risk Control for Reasoning on a Compute Budget

Anushri Suresh[*], **Alvin Zhang**[*], **Rishi More**[*], **Xi Wang**, **William Jurayj**
Benjamin Van Durme, **Eric Nalisnick**[♡] and **Daniel Khashabi**[♡]
Department of Computer Science, Johns Hopkins University

Abstract

Test-time Compute (TTC) substantially improves LLM reasoning accuracy by allowing models to think through multi-step intermediate reasoning. However, extended reasoning chains can dramatically increase inference cost, motivating frameworks that dynamically trade off computational expense against the expected improvement. We introduce a **dual-threshold framework** that adaptively allocates computation: an **upper threshold** that halts reasoning once *sufficient confidence on answer correctness is reached*, and a **lower threshold** that abandons instances where continued *reasoning is unlikely to yield any correct answer*. To determine these thresholds in a principled way, we employ *Distribution-Free Risk Control* (DFRC), which guarantees user-specified bounds on the accuracy loss relative to full reasoning. Across two open-weight reasoning models and four benchmarks, our approach reduces reasoning cost by up to 52% (on AIME) while maintaining accuracy within a narrow margin of full TTC.

1 Introduction

Test-time compute (TTC) has become a key driver of advances in LLM reasoning, powering systems like DeepSeek-R1 [4]. By externalizing reasoning into intermediate steps, TTC delivers substantial accuracy gains. Yet these gains come at the cost of long reasoning chains and inference cost. The central challenge is deciding, step by step, *whether additional reasoning justifies the extra compute*.

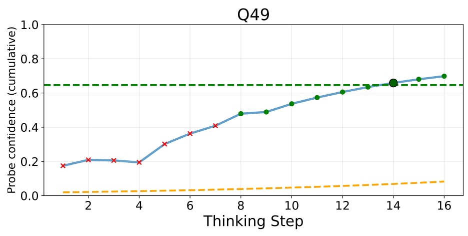
Prior frameworks typically terminate reasoning *once further steps add little to performance* [13, 16]. We refer to these as **"upper-threshold"** methods, which halt when the answer is deemed "good enough." In contrast, we explore the complementary idea of a **lower threshold**, which detects when an instance is too difficult within the compute budget and halts early to reallocate resources elsewhere. Specifically, we introduce a parametric lower-threshold mechanism (Fig. 1) alongside the upper threshold. It enforces monotonic growth in model confidence during reasoning, and halts if confidence falls below a target value, indicating insufficient progress.

A key practical challenge is selecting the lower-threshold function. Unlike model confidence—which, when well-calibrated, directly reflects predictive risk in terms of correctness—the lower threshold parameter lacks an obvious interpretability, and its value range can vary widely. To address this, we employ *Distribution-Free Risk Control* [3, DFRC], which jointly selects the lower-threshold parameter and upper-threshold value on a validation set given a user-specified target risk, thereby ensuring guaranteed risk bounds (§2).

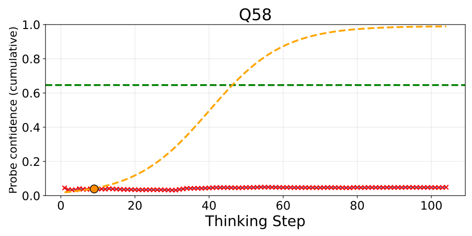
While DFRC has proven effective for early exiting in contexts such as layer-wise exiting [14, 7] and speculative decoding [9], it has not been explored for inference-time reasoning. Adapting DFRC

[*]Co-first authors. [♡] Equal advising.

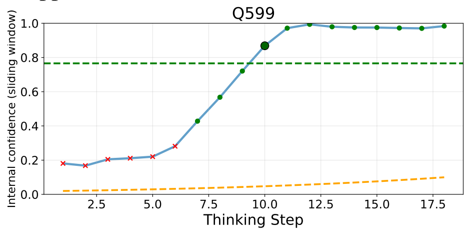
39th Conference on Neural Information Processing Systems (NeurIPS 2025) Workshop: NeurIPS 2025 Workshop on Efficient Reasoning.



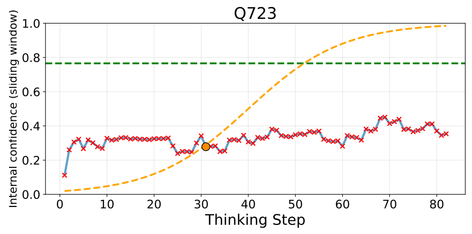
(a) Upper Threshold Exit (Probe-based confidence)



(b) Lower Threshold Exit (Probe-based confidence)



(c) Upper Threshold Exit (Internal confidence)



(d) Lower Threshold Exit (Internal confidence)

Figure 1: Different threshold configurations in our risk-control framework on four AIME questions. The x-axis denotes the **reasoning step index**, and y-axis shows proxy confidence signals computed using either probe-based or internal measures (§3) indicated by the blue curve traces. The label "cumulative" denotes the mean of all signals up to each step, while "sliding window" indicates the mean over the most recent five steps. Markers × indicate incorrect intermediate answers, and ● indicate correct ones. Each subfigure compares the **upper threshold** (green line; §2.1) and the **adaptive lower threshold** (orange curve; §2.2) that guide early exit. A larger ● indicates an early exit that yields a correct intermediate answer under the upper threshold, while a larger ● denotes an effective early exit on instances where the model's current answer is incorrect and unlikely to become correct with further reasoning. For illustration, the confidence trajectory is shown beyond the exit point to indicate how it would have evolved without early stopping. As shown, our framework calibrates per-instance thresholds that enable effective and well-timed early exits.

to this setting introduces unique challenges: reasoning steps are temporally correlated, confidence estimates fluctuate non-monotonically, and stopping decisions must be calibrated across entire reasoning trajectories rather than individual steps. To address these challenges, we introduce time-adaptive parametric lower thresholds that vary dynamically with the reasoning step index—forming a parametric family of thresholds rather than a single fixed one—and a dual-threshold calibration framework that jointly optimizes high- and low-confidence exits across full reasoning trajectories.

In summary, our contributions are twofold: (1) We cast selective TTC as a risk-control problem and propose a dual-threshold mechanism that unifies *stopping* (when the answer is sufficient) and *continuation* (when further reasoning is beneficial), leveraging signals such as model confidence, token entropy, and probe predictions. (2) We empirically evaluate our method on two open-weight reasoning models and four datasets, achieving up to 52% reduction in reasoning cost on AIME while maintaining or improving accuracy, and enabling explicit user control over the risk–compute tradeoff.

Related Work: Recent work has sought to reduce the cost of test-time reasoning by learning when to stop generating intermediate steps. These adaptive stopping methods terminate reasoning once further steps are unlikely to change the outcome, using strategies such as plateau detection [13, 10], high-confidence thresholds [8, 15], or intermediate-answer consistency [5]. We refer to these as *upper*-threshold methods, which halt once the model is sufficiently confident. In contrast, we introduce a complementary *lower*-threshold mechanism that identifies cases where continued reasoning is unproductive, allowing compute to be reallocated to more promising instances.

A related effort by Fu et al. [6] explores low-confidence exits but operates over multiple reasoning chains with a fixed lower threshold. In contrast, our method performs early exiting within a single reasoning trajectory, enabling a finer-grained and more adaptive stopping policy. Moreover, we use DFRC [3, 12] to set thresholds in a principled way, providing formal accuracy guarantees and proven effectiveness in early-exit and selective-prediction tasks [7, 9].

2 Risk Control for Inference-Time Early Stopping

To formalize the trade-off between computational efficiency and predictive reliability, we introduce a framework for *risk-controlled early stopping*. At each reasoning step t (see Appendix B for the definition), the model produces an intermediate prediction $\hat{y}_t$ with an associated confidence signal s_t. The central question is when to terminate reasoning so that computation is minimized while keeping the probability of an incorrect final answer provably small.

What is "risk"? Intuitively, *risk* quantifies how much accuracy we are willing to trade for efficiency. Bounding this risk ensures we save compute only when doing so does not significantly increase the chance of error. A formal definition of selective risk and its estimation appears in Appendix E.

Two types of early exit: We instantiate this principle through two complementary exit mechanisms:

- **Upper-threshold Exit:** stops reasoning once confidence exceeds a threshold, indicating further computation is unlikely to change the answer—saving compute on easy cases.
- **Lower-Threshold Exit:** halts reasoning when confidence stays low, avoiding wasted computation on problems the model is unlikely to solve correctly.

Throughout this section, $y_i^\star$ denotes the ground-truth answer for instance i, $\hat{y}i, t$ the model's step-t prediction, si, t its confidence signal, and $\hat{y}i,$ final its final output after full reasoning. Two thresholds, τhigh and τ_{low}, govern early exits triggered by high or low confidence, respectively.

2.1 Upper Threshold Exit: Halting Inference Once Confident in Correctness

To avoid wasting computational resources on instances where the model is already certain, we implement a high-confidence exit. The model stops generating further reasoning steps as soon as its confidence in an answer $\hat{y}_t$ exceeds a predefined upper threshold, τ_{high}. The primary risk associated with this strategy is being confidently wrong. We can formalize this risk as an expected loss, ensuring that the probability of an incorrect prediction, given high confidence, is bounded by a small value ε:

$$\mathbb{E}\big[\mathbf{1}\{\hat{y}_t \neq y_i^\star\} \,\big|\, s_{i,t} \geq \tau_{\text{high}}\big] \leq \varepsilon_{\text{high}}, \tag{1}$$

A second key risk to control is the performance-gap risk. This quantifies the change in loss incurred by early-exiting compared to the loss from letting the model run to completion. The goal is to ensure that computational savings do not come at the cost of a significant drop in performance. Let t be the step where an early exit occurs for a given query i. The performance-gap risk is the expected difference between the error of the early-exit answer, $\hat{y}_{i,t}$, and the error of the final answer from the full model, $\hat{y}_{i,\text{final}}$. We can bound this risk by a value ε_{gap}:

$$\mathbb{E}\left[\mathbf{1}\{\hat{y}_{i,t} \neq y_i^\star\} - \mathbf{1}\{\hat{y}_{i,\text{final}} \neq y_i^\star\}\right] \leq \varepsilon_{\text{gap}} \tag{2}$$

This ensures that the overall early-exit policy does not introduce substantially more errors than the full model would have.

2.2 Lower Threshold Exit: Stopping Early When the Answer Appears Unreachable

We also exit early when the model's confidence signals that continued reasoning is unlikely to yield a correct answer. This scenario captures two distinct failure modes: degrading reasoning, which occurs when the model's confidence deteriorates, falling below a static lower threshold τ_{low}, and unproductive reasoning, which occurs when the model becomes stuck and its confidence stagnates without significant improvement over a series of steps. This is detected using a parametric condition and prevents wasting computation on futile reasoning chains.

Let t denote the reasoning step at which an early exit is triggered and let $j > t$ index subsequent steps in the same trajectory. The key risk in this case is halting too soon—exiting at step t when the model might have recovered and produced a correct answer at some later step j. We bound this risk by a small tolerance ε_{low}:

$$\mathbb{E}\left[\max_{j>t} \mathbf{1}\{\hat{y}_{i,j} = y_i^\star\}\right] \leq \varepsilon_{\text{low}}. \tag{3}$$

While static thresholds τ_{low} provide a simple mechanism for controlling early exits, they can be overly rigid across different reasoning trajectories. In particular, a fixed τ_{low} may prematurely stop short reasoning chains yet fail to terminate longer, stagnant ones.

To address this, we introduce **adaptive threshold schedules** that evolve with the reasoning step t. A common choice is a parametric schedule that gradually increases $\tau_{\text{low}}(t)$ as t grows, discouraging very early exits while allowing termination after prolonged low-confidence phases. Detailed implementation and equations are shown in Appendix E.

This dynamic calibration improves both stability and coverage, ensuring that early-stopping behavior adapts to the difficulty and length of each reasoning trajectory. We show a few per-question examples with lower-threshold exits for a (given signal) below.

2.3 Distribution-Free Risk Control Calibration (DFRC)

Thresholds such as τ_{high} and c, whether static or adaptive, must be chosen to keep risk below a desired tolerance on unseen data. We achieve this via DFRC, which guarantees: $\Pr[R(\tau) \leq \varepsilon] \geq 1 - \delta$, ensuring risk $\leq \varepsilon$ with probability $1 - \delta$ on onseen data. We calibrate thresholds using one of three procedures: **Conformal Risk Control(CRC)** [2], **Upper Confidence Bound (UCB)** [3], and **Learn-Then-Test (LTT)** [1]. Given a target risk level ε, each method selects τ_{high} (and, when applicable, parameters for $\tau_{\text{low}}(t)$) to control empirical test risk from the validation set. CRC employs a regularized empirical bound, UCB applies a Hoeffding upper confidence bound, and LTT performs a Hoeffding–Bentkus hypothesis test. DFRC thus produces thresholds that maximize compute savings while maintaining formal accuracy guarantees (see Appendix E.4 for details).

3 Experimental Setup

Models: We evaluate two models: **DeepSeek-R1-Distill-Qwen-7B** and **DeepSeek-R1-Distill-Qwen-32B**. All models are run under a fixed budget forcing setup with the official decoding parameters (temperature $= 0.6$, top-$p = 0.9$). Please check Appendix B for the generation protocol.

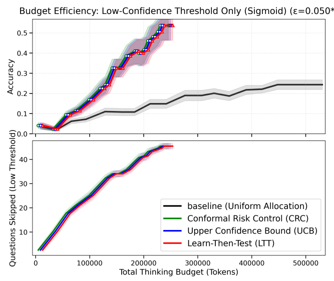
Datasets: We evaluate across four benchmarks: (a) **AIME (1983–2024):** Used both for probe training (1983–2022) and evaluation (2022–2024). We measure internal confidence, entropy, and probe signals on this dataset. (b) **GPQA-D:** Graduate-level physics reasoning questions. (c) **MuSR:** Subsets covering *murder mysteries* and *object placements*, evaluated separately. (d) **HLE:** Graduate-level questions, filtered to text-only math data as DeepSeek is not multimodal.

Extracting signals on model confidence: We study three types of confidence signals: two non-probe baselines and one probe-based signal.
(a) Internal confidence: For each final candidate answer, we compute the internal confidence of the model as the normalized logarithmic probability of the generated answer tokens. This measures the model's self-estimated likelihood under its decoding distribution.
(b) Entropy: We compute Shannon entropy over the distribution of the first token of the forced final answer following [16]. Please refer to Appendix C for details of implementation.
(c) Probe-based confidence: We train probes on AIME data to estimate stepwise correctness from hidden representations. This signal is math-specific and is therefore evaluated only on AIME and the math subset of HLE. Please check Appendix C for the details of probing implementation.

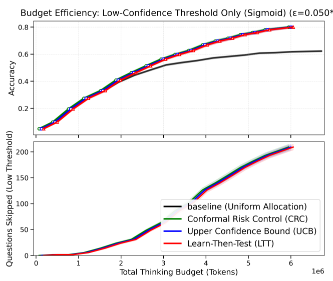
4 Experimental Results

We present our results in twofolds: we first present the results with the lower-threshold-only early-exiting, which is an under-explored domain that previous works have not considered. Then, we present the combination of lower and higher thresholds to demonstrate the effectiveness of our framework in token reduction and accuracy retention. Please check Appendix F for more results.

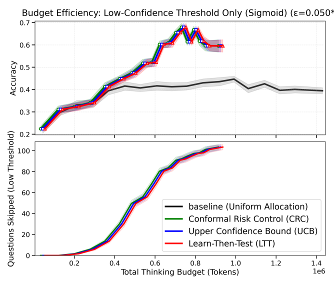
Lower-Threshold-Only Early Exiting Results: In the lower-threshold-only setting (Fig. 2), we observe that introducing a stopping rule based solely on low-confidence signals already yields meaningful token savings while maintaining accuracy close to the baseline. By abstaining on hopeless cases, we observe further gains in effective accuracy. For example, on AIME (1983–2024) with confidence as the signal, the curve demonstrates that early termination can discard unproductive reasoning without substantially increasing risk. Similarly, the probe confidence variant on the AIME (2022-2024) split shows that auxiliary correctness signals can act as effective triggers for early exit. These results confirm that the lower-threshold-only regime is not only viable but also offers a previously underexplored avenue for reducing unnecessary computation. We explain the discrepancy of the baseline accuracy in Appendix F.1



(a) DeepSeek-7B (AIME 2022–2024, probe confidence)



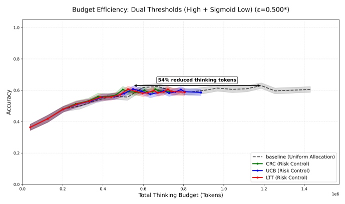
(b) DeepSeek-7B (AIME 1983-2024, confidence)



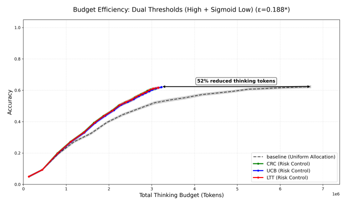
(c) DeepSeek-7B (GPQA-D, confidence)

Figure 2: Lower-threshold-only early-exiting results across datasets using DeepSeek-7B under different confidence signals. The *Baseline* denotes standard inference without early stopping. *Conformal Risk Control*, *Upper Confidence Bound*, and *Learn-Then-Test* correspond to the three early-exiting strategies introduced in §2.3. The top panels show early-stopping accuracy, computed over only those examples that were not terminated by the lower threshold. The bottom panels show the proportion of questions skipped as a function of total reasoning budget. In total, the AIME 2022–2024 dataset contains 73 questions, the AIME 1983–2024 dataset 933 questions, and the GPQA-D dataset 198 questions. These results highlight how lower-threshold exiting improves compute efficiency by reallocating reasoning steps from unproductive instances to those more likely to yield correct answers.

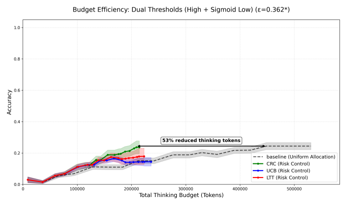
Combined Early Exiting Results: When we extend to the combined-threshold framework (Fig. 3), the benefits become more pronounced. The dual thresholds allow us to both (i) terminate early when reasoning is unproductive and (ii) stop once confidence is sufficiently high. This yields stronger budget–efficiency trade-offs across multiple datasets and models. In particular, DeepSeek-32B on GPQA-D and AIME exhibit smoother gains, showing that larger models are able to capitalize more effectively on the thresholding strategy. Meanwhile, the 7B model on AIME with probe confidence highlights that learned correctness signals can further enhance efficiency. Together, these results demonstrate that the combined framework consistently reduces token usage while preserving, and in some cases even slightly improving, accuracy relative to full test-time compute.



(a) DeepSeek-32B (GPQA-D, Internal Confidence)



(b) DeepSeek-32B (AIME 1983–2024, Internal Confidence)



(c) DeepSeek-7B (AIME 2022–2024, Probe-based confidence)

Figure 3: Combined upper and lower threshold early-exiting results across models and datasets. Each panel shows budget–efficiency comparison under different signals. The *Baseline* denotes standard inference without early stopping. *Conformal Risk Control*, *Upper Confidence Bound*, and *Learn-Then-Test* correspond to the three early-exiting strategies introduced in §2.3.

5 Conclusion

We introduced a framework for adaptive TTC that uses dual thresholds to decide when further computation is unnecessary or sufficient. This approach provides risk guarantees and reduces inference cost, offering a practical path for efficient reasoning-augmented LLMs.

References

[1] Anastasios N. Angelopoulos, Stephen Bates, Emmanuel J. Candès, Michael I. Jordan, and Lihua Lei. Learn then Test: Calibrating Predictive Algorithms to Achieve Risk Control. *arXiv Preprint (arXiv:2110.01052)*, 2021.

[2] Anastasios Nikolas Angelopoulos, Stephen Bates, Adam Fisch, Lihua Lei, and Tal Schuster. Conformal risk control. In *The Twelfth International Conference on Learning Representations*, 2024.

[3] Stephen Bates, Anastasios Angelopoulos, Lihua Lei, Jitendra Malik, and Michael Jordan. Distribution-free, risk-controlling prediction sets. *J. ACM*, 68(6), September 2021.

[4] DeepSeek-AI, Daya Guo, Dejian Yang, Haowei Zhang, Junxiao Song, Ruoyu Zhang, Runxin Xu, Qihao Zhu, Shirong Ma, Peiyi Wang, Xiao Bi, Xiaokang Zhang, Xingkai Yu, Yu Wu, Z. F. Wu, Zhibin Gou, Zhihong Shao, Zhuoshu Li, Ziyi Gao, Aixin Liu, Bing Xue, Bingxuan Wang, Bochao Wu, Bei Feng, Chengda Lu, Chenggang Zhao, Chengqi Deng, Chenyu Zhang, Chong Ruan, Damai Dai, Deli Chen, Dongjie Ji, Erhang Li, Fangyun Lin, Fucong Dai, Fuli Luo, Guangbo Hao, Guanting Chen, Guowei Li, H. Zhang, Han Bao, Hanwei Xu, Haocheng Wang, Honghui Ding, Huajian Xin, Huazuo Gao, Hui Qu, Hui Li, Jianzhong Guo, Jiashi Li, Jiawei Wang, Jingchang Chen, Jingyang Yuan, Junjie Qiu, Junlong Li, J. L. Cai, Jiaqi Ni, Jian Liang, Jin Chen, Kai Dong, Kai Hu, Kaige Gao, Kang Guan, Kexin Huang, Kuai Yu, Lean Wang, Lecong Zhang, Liang Zhao, Litong Wang, Liyue Zhang, Lei Xu, Leyi Xia, Mingchuan Zhang, Minghua Zhang, Minghui Tang, Meng Li, Miaojun Wang, Mingming Li, Ning Tian, Panpan Huang, Peng Zhang, Qiancheng Wang, Qinyu Chen, Qiushi Du, Ruiqi Ge, Ruisong Zhang, Ruizhe Pan, Runji Wang, R. J. Chen, R. L. Jin, Ruyi Chen, Shanghao Lu, Shangyan Zhou, Shanhuang Chen, Shengfeng Ye, Shiyu Wang, Shuiping Yu, Shunfeng Zhou, Shuting Pan, S. S. Li, Shuang Zhou, Shaoqing Wu, Shengfeng Ye, Tao Yun, Tian Pei, Tianyu Sun, T. Wang, Wangding Zeng, Wanjia Zhao, Wen Liu, Wenfeng Liang, Wenjun Gao, Wenqin Yu, Wentao Zhang, W. L. Xiao, Wei An, Xiaodong Liu, Xiaohan Wang, Xiaokang Chen, Xiaotao Nie, Xin Cheng, Xin Liu, Xin Xie, Xingchao Liu, Xinyu Yang, Xinyuan Li, Xuecheng Su, Xuheng Lin, X. Q. Li, Xiangyue Jin, Xiaojin Shen, Xiaosha Chen, Xiaowen Sun, Xiaoxiang Wang, Xinnan Song, Xinyi Zhou, Xianzu Wang, Xinxia Shan, Y. K. Li, Y. Q. Wang, Y. X. Wei, Yang Zhang, Yanhong Xu, Yao Li, Yao Zhao, Yaofeng Sun, Yaohui Wang, Yi Yu, Yichao Zhang, Yifan Shi, Yiliang Xiong, Ying He, Yishi Piao, Yisong Wang, Yixuan Tan, Yiyang Ma, Yiyuan Liu, Yongqiang Guo, Yuan Ou, Yuduan Wang, Yue Gong, Yuheng Zou, Yujia He, Yunfan Xiong, Yuxiang Luo, Yuxiang You, Yuxuan Liu, Yuyang Zhou, Y. X. Zhu, Yanhong Xu, Yanping Huang, Yaohui Li, Yi Zheng, Yuchen Zhu, Yunxian Ma, Ying Tang, Yukun Zha, Yuting Yan, Z. Z. Ren, Zehui Ren, Zhangli Sha, Zhe Fu, Zhean Xu, Zhenda Xie, Zhengyan Zhang, Zhewen Hao, Zhicheng Ma, Zhigang Yan, Zhiyu Wu, Zihui Gu, Zijia Zhu, Zijun Liu, Zilin Li, Ziwei Xie, Ziyang Song, Zizheng Pan, Zhen Huang, Zhipeng Xu, Zhongyu Zhang, and Zhen Zhang. Deepseek-r1: Incentivizing reasoning capability in llms via reinforcement learning, 2025.

[5] Yichao Fu, Junda Chen, Siqi Zhu, Zheyu Fu, Zhongdongming Dai, Yonghao Zhuang, Yian Ma, Aurick Qiao, Tajana Rosing, Ion Stoica, and Hao Zhang. Efficiently scaling llm reasoning with certaindex, 2025.

[6] Yichao Fu, Xuewei Wang, Yuandong Tian, and Jiawei Zhao. Deep think with confidence, 2025.

[7] Metod Jazbec, Alexander Timans, Tin Hadži Veljković, Kaspar Sakmann, Dan Zhang, Christian Andersson Naesseth, and Eric Nalisnick. Fast yet safe: Early-exiting with risk control. *Advances in Neural Information Processing Systems*, 37:129825–129854, 2024.

[8] William Jurayj, Jeffrey Cheng, and Benjamin Van Durme. Is that your final answer? test-time scaling improves selective question answering. In *Proceedings of the 63rd Annual Meeting of the Association for Computational Linguistics (Volume 2: Short Papers)*, pages 636–644, 2025.

[9] Yaniv Leviathan, Matan Kalman, and Yossi Matias. Fast inference from transformers via speculative decoding. In *International Conference on Machine Learning*, 2022.

[10] Minjia Mao, Bowen Yin, Yu Zhu, and Xiao Fang. Early stopping chain-of-thoughts in large language models, 2025.

[11] 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.

[12] Tal Schuster, Adam Fisch, Jai Gupta, Mostafa Dehghani, Dara Bahri, Vinh Tran, Yi Tay, and Donald Metzler. Confident adaptive language modeling. *Advances in Neural Information Processing Systems* (NeurIPS), 35:17456–17472, 2022.

[13] Menghua Wu, Cai Zhou, Stephen Bates, and T. Jaakkola. Thought calibration: Efficient and confident test-time scaling. *ArXiv*, abs/2505.18404, 2025.

[14] Andrea Wynn, Metod Jazbec, Charith Peris, Rinat Khaziev, Anqi Liu, Daniel Khashabi, and Eric Nalisnick. Safe and efficient in-context learning via risk control, 2025.

[15] 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, 2025.

[16] 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.

[17] 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, 2025.

A Test-Time Compute and Motivation

Test-time reasoning has been shown to substantially improve model accuracy on challenging domains such as mathematics and question answering. In particular, allocating a larger computation budget, by allowing the model to generate more tokens, can lead to higher final accuracy.

To study this effect, we evaluate **DeepSeek-Distill-Qwen-32B** under a budget-forcing setup, where the model is prompted with the string `Final answer: \boxed{` to elicit a boxed final answer per 100 generated tokens. We record accuracy as a function of the number of generated tokens.

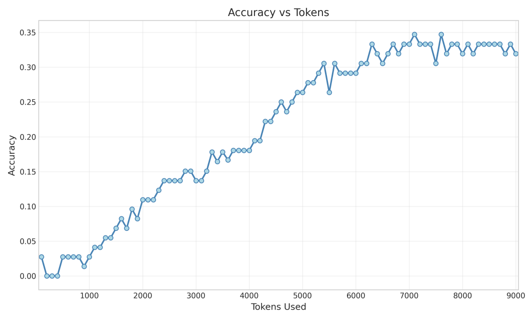


Figure 4: Performance of DeepSeek-Distill-Qwen-32B on AIME (2022–2024). Accuracy consistently improves as more tokens are generated.

While both benchmarks show clear gains from additional computation, we also observe a diminishing returns effect: beyond a certain point, generating extra tokens no longer yields substantial accuracy improvements. This naturally motivates the question: *can we identify and save the surplus tokens that do not contribute to accuracy gains?*

B Generation Protocol

Following [15], each problem is allotted a total budget of $10{,}000$ tokens: $9{,}000$ for the *thinking* phase and $1{,}000$ for the *final-answer* phase. If the model attempts to terminate early (e.g., by producing EOS or stop tokens), we append benign filler tokens (the token `wait`) until the 9k-token thinking budget is exhausted, following the protocol introduced in S1 Scaling [11]. We used the

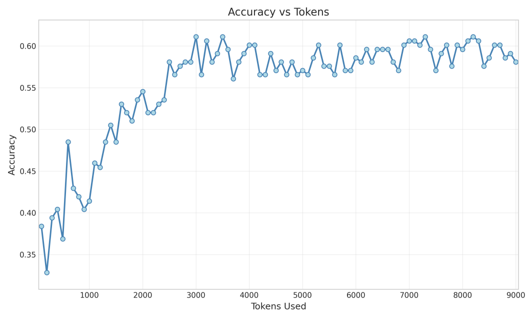


Figure 5: Performance of DeepSeek-Distill-Qwen-32B on GPQA-Diamond. Accuracy similarly increases with more tokens.

system prompt: `Please reason step by step, and put your final answer within boxed.` and we put the question from the give dataset to the user prompt. We used 4 A100 GPUs with 80 GB of memory for inference.

C Signal Extraction Details

We follow the method proposed in DEER [15], appending the string `Final answer: \boxed{` to the model prompt to force generation of a boxed answer. The confidence and entropy signals are computed over these forced answer tokens. We define **a reasoning step** as the thought until we hit the "wait" token. At this point we append wait again to prompt the model to continue thinking therefore the steps are of arbitrary length and constitute one logical thought until the model shifts it's thought process signaled by "wait"

C.1 Internal Confidence

Confidence Computation. Following [15], we define the confidence of a generated sequence $\mathbf{a} = (a_1, \dots, a_n)$ as the **geometric mean** of its per-token probabilities:

$$C = \left(\prod_{i=1}^{n} p(a_i)\right)^{\frac{1}{n}}, \qquad p(a_i) = \text{softmax}\big(\mathcal{M}(P, T, I, a_{<i})\big)_{a_i} \tag{4}$$

where $\mathcal{M}$ denotes the LM head producing logits and the softmax normalizes over the vocabulary to yield token probabilities.

Algorithm 1 Confidence Computation from vLLM Log-Probabilities

```
1: Input: Log-probabilities $\{\log p(a_1), \dots, \log p(a_n)\}$ from vLLM output
2: Initialize: log_prob_sum $\leftarrow 0$
3: for $i \leftarrow 2$ to $n$ do          ▷ Skip first generated token (e.g., "{" delimiter)
4:     log_prob_sum $\leftarrow$ log_prob_sum $+ \log p(a_i)$
5: end for
6: Compute confidence:
       $C \leftarrow \exp\Big(\frac{\texttt{log\_prob\_sum}}{n-1}\Big)$
7: Return $C$
```

Note. The log-probabilities $\log p(a_i)$ provided by vLLM are already *post-softmax*, i.e. $p(a_i) = \text{softmax}(\mathcal{M}(P, T, I, a_{<i}))_{a_i}$. Exponentiating them recovers the normalized probabilities directly, so no explicit softmax operation is needed in implementation.

C.2 Entropy

Definition. Following [16], we quantify the model's token-level uncertainty using the Shannon entropy:

$$H(p) = -\sum_{i=1}^{k} p_i \log p_i, \tag{5}$$

where $\{p_1, \ldots, p_k\}$ is a categorical probability distribution over candidate tokens (or answers).

Multiple-Choice Setting. For multiple-choice (MCQ) datasets, we compute entropy over the top-k tokens following the boxed-answer prefix:

$$H_{\text{MCQ}} = -\sum_{i=1}^{k} \tilde{p}_i \log \tilde{p}_i, \qquad \tilde{p}_i = \frac{p_i}{\sum_{j=1}^{k} p_j}, \tag{6}$$

i.e., a truncated and renormalized distribution over the top-k candidate options.

Note. We prefer top-k entropy over restricting to $\{A, B, C, D\}$ because it accounts for probability mass that the model assigns to spurious or off-distribution tokens. This yields a more faithful measure of uncertainty: if the model is highly confident in a non-option token, H_{MCQ} will reflect that low confidence in all valid options.

Free-Form Setting. For open-ended answers, our goal is to estimate the *sequence-level entropy* of the model's answer distribution:

$$H_{\text{true}} = -\sum_{y \in \mathcal{Y}} P(y \mid x) \log P(y \mid x),$$

where $\mathcal{Y}$ is the set of all possible completions and

$$P(y \mid x) = \prod_{t=1}^{|y|} p(a_t \mid x, a_{<t})$$

is the sequence probability of completion $y = (a_1, \ldots, a_{|y|})$ given prompt x, expressed as a product of token-level probabilities.

Since $\mathcal{Y}$ is exponentially large, we approximate this distribution by constructing a high-probability subset $\hat{\mathcal{Y}}$ using a lightweight beam search. Specifically, $\hat{\mathcal{Y}}$ contains all answer completions discovered by expanding the top-b partial sequences token-by-token until either a `\boxed{}` answer is closed or the search reaches a maximum depth T.

We then construct a truncated and renormalized distribution:

$$\tilde{P}(y \mid x) = \frac{P(y \mid x)}{\sum_{y' \in \hat{\mathcal{Y}}} P(y' \mid x)}, \qquad y \in \hat{\mathcal{Y}},$$

and compute the *beam entropy*:

$$H_{\text{beam}} = -\sum_{y \in \hat{\mathcal{Y}}} \tilde{P}(y \mid x) \log \tilde{P}(y \mid x).$$

This is analogous to top-k entropy in the MCQ setting, but defined over completed answers instead of tokens.

Algorithm 2 Beam-Search-Based Entropy Estimation for Free-Form Answers

```
1: Input: Prompt token sequence $\mathbf{x}$, beam size $b$, maximum expansion steps $T$
2: Initialize: Beam $\mathcal{B} \leftarrow \{(\mathbf{c} = \mathbf{x}, p = 1.0)\}$ where $\mathbf{c}$ is a token sequence (context) and $p$ its
   cumulative probability; completed map $\mathcal{C} \leftarrow \{\}$    ▷ dictionary mapping answer string →
   cumulative probability
3: for $t \leftarrow 1$ to $T$ do    ▷ repeat expansion for at most $T$ decoding steps
4:     Initialize empty list $\mathcal{B}_{\text{next}}$ for next-step beam
5:     for each $(\mathbf{c}, p) \in \mathcal{B}$ do
6:         Expand $\mathbf{c}$ by querying the model for its top-$b$ next tokens
7:         for each candidate next token $a$ from the top-$b$ distribution do
8:             $p' \leftarrow p \cdot p(a|\mathbf{c}), \quad \mathbf{c}' \leftarrow \mathbf{c} \cup a$
9:             if $\mathbf{c}'$ closes a boxed answer then
10:                $s \leftarrow \texttt{extract\_last\_boxed}(\mathbf{c}')$
11:                $\mathcal{C}[s] \leftarrow \mathcal{C}[s] + p'$ ▷ accumulate probability mass if $s$ was found via multiple paths
12:            else
13:                Add $(\mathbf{c}', p')$ to $\mathcal{B}_{\text{next}}$
14:            end if
15:        end for
16:    end for
17:    Keep only the top-$b$ elements of $\mathcal{B}_{\text{next}}$ by probability
18:    if $\mathcal{B}_{\text{next}}$ is empty then    ▷ no partial completions remain to expand; terminate search early
19:        break
20:    end if
21:    $\mathcal{B} \leftarrow \mathcal{B}_{\text{next}}$
22: end for
23: Normalize probabilities: $p_s \leftarrow p_s / \sum_{s \in \mathcal{C}} p_s$ for $s \in \mathcal{C}$
24: Compute entropy:
        $$H_{\text{beam}} = -\sum_{s \in \mathcal{C}} p_s \log p_s$$
25: Return $H_{\text{beam}}$
```

C.3 Probes

Representation extraction and labeling: Let P denote the original problem prompt and $x_{1:T}$ the budget-forced trajectory. Suppose the model emits K candidate final answers at token indices $1 \leq t_1 < \cdots < t_K \leq 9000$, with the s-th candidate answer denoted $\hat{a}_s$. For each step s, we reconstruct the decoding context

$$C_s = [P;\, x_{1:t_s}],$$

and extract the hidden representation of the last token:

$$h_s \in \mathbb{R}^d.$$

Each step is labeled according to correctness relative to the gold answer $a^\star$:

$$y_s = I[\hat{a}_s = a^\star] \in \{0, 1\}.$$

This yields a dataset

$$\mathcal{D} = \{(h_s, y_s)\}_{s=1}^{K}.$$

Probe model and training: We train a two-layer MLP probe [17] to predict stepwise correctness y_s from the representation h_s. Unless otherwise specified, features are extracted exactly as described above. The probe is trained on AIME 1983–2022, with optional mixing of ~ 400 additional problems from the MATH training split, yielding 2 probes for evaluation. Evaluation is performed on AIME 2022–2024.

Outputs and usage: After training, we apply the best-performing probe to AIME 2022–2024 trajectories, obtaining stepwise confidence scores

$$\pi_s = \text{Probe}(h_s) \in [0, 1].$$

These probe-based signals are used alongside internal confidence and entropy in our downstream analyses and decision policies.

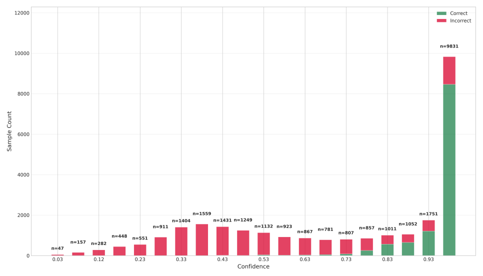
D Analysis of Signals

We analyze two dataset-level signals—*confidence* and *entropy*—by bucketing steps every 100 tokens and averaging accuracy per bucket. Per-token probe training is computationally heavy and poorly calibrated at this granularity, so we present probe results only as per-instance correctness timelines.

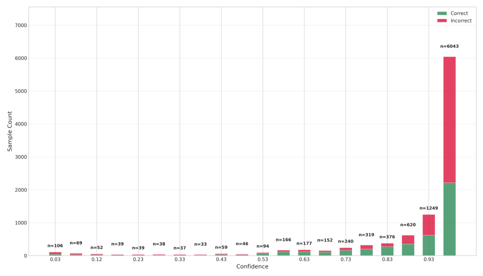
D.1 Confidence Calibration

D.1.1 Confidence and Entropy Calibration

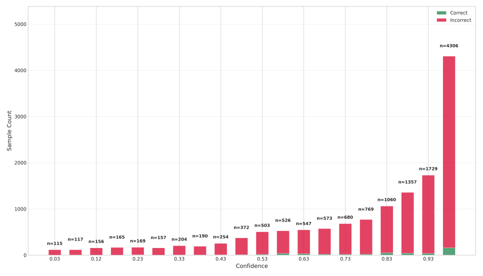
From the plots, we notice that the signals are more desirable on AIME questions since they have relatively balanced correct vs. incorrect samples. The signals on HLE-Math is overconfident with almost no correct answers but high confidence, while low entropy.



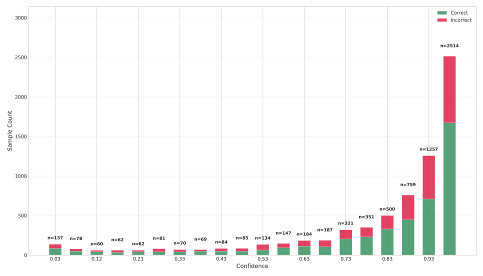
(a) AIME



(b) GPQA-D

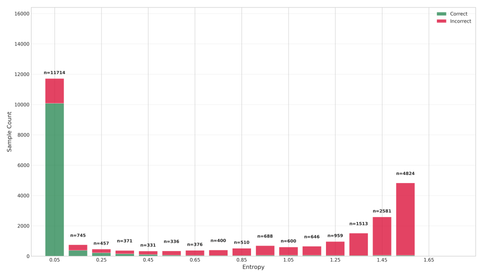


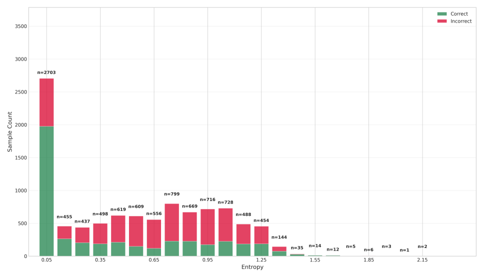
(c) HLE-Math

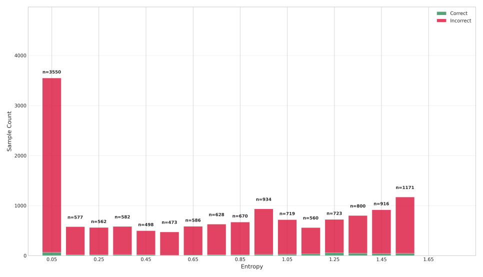


(d) Mysteries

Figure 6: **Confidence calibration (32B).** Per-100-token bucket accuracy vs. confidence across four datasets.


(a) AIME


(b) GPQA-D


(c) HLE-Math

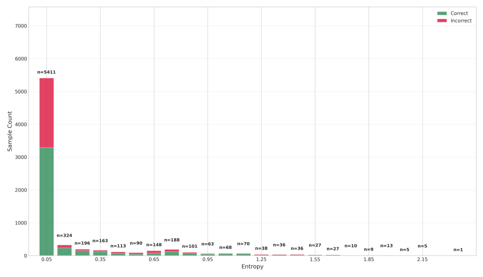

(d) Mysteries

Figure 7: **Entropy calibration (32B).** Per-100-token bucket accuracy vs. entropy across four datasets.

D.1.2 Probe Calibration

We evaluate probe calibration only on the last 73 questions from AIME 2022–2024, since the first 860 questions were used to train the probe. We show two calibration plots from probe training on the AIME dataset, as well as one calibration result on the HLE_math dataset.

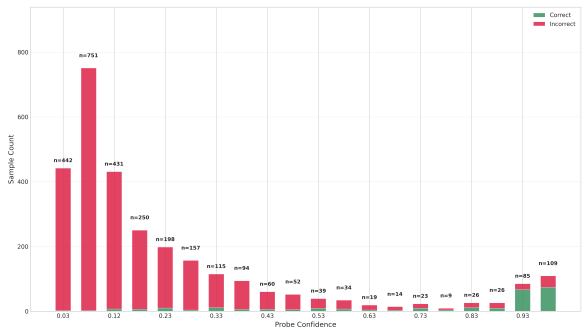


Figure 8: Probe calibration on AIME with mixed math training data.

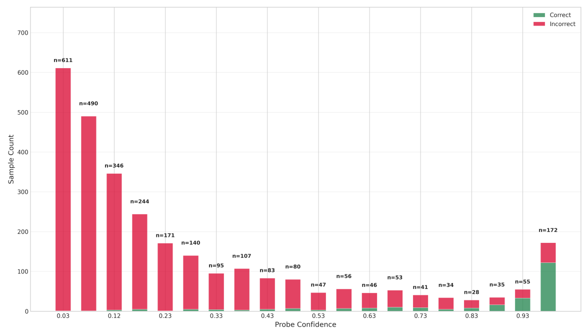


Figure 9: Probe calibration on AIME with AIME-only training data.

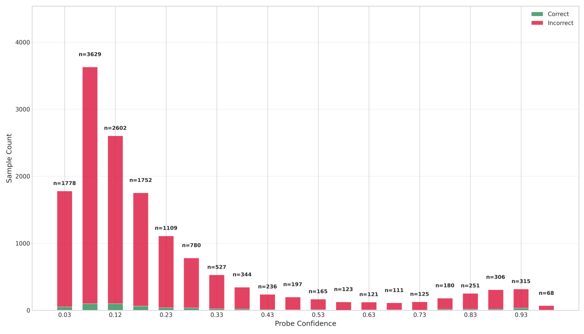
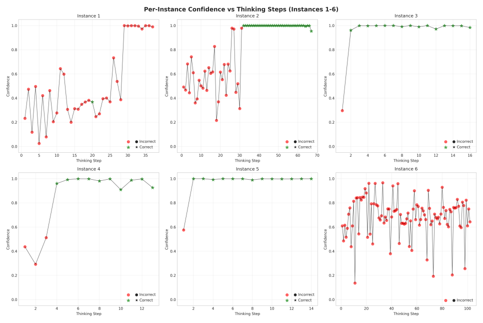


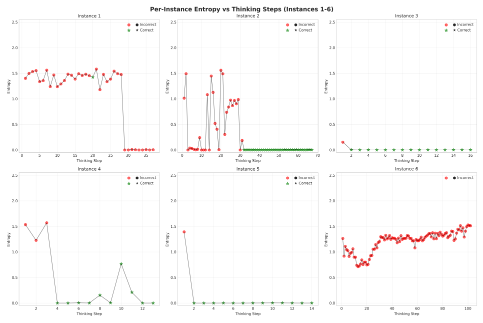
Figure 10: Probe calibration on the HLE_math dataset.

D.2 Per-Instance Example Analysis

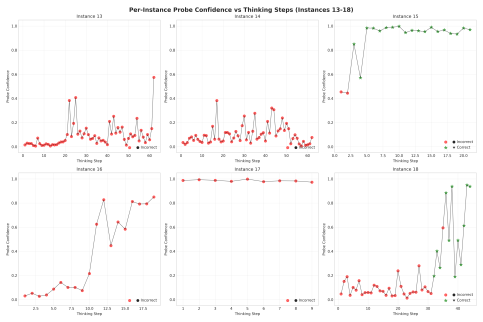
We plot the model's confidence, entropy, and probe's correctness signal over reasoning steps to illustrate per-instance behavior and motivate our risk-control framework.



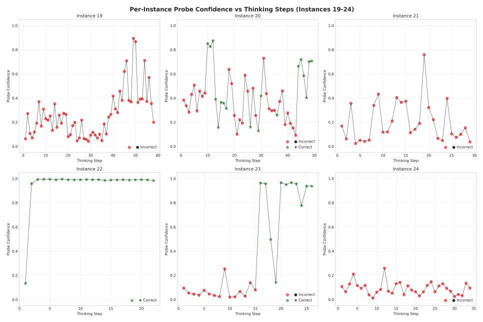
(a) Confidence



(b) Entropy

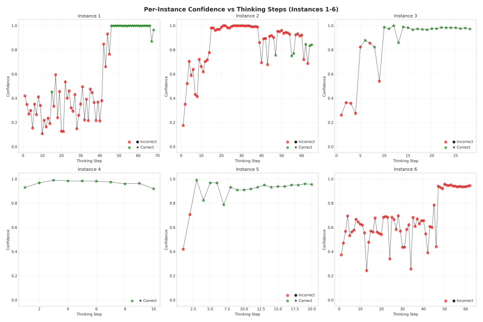


(c) Probe (ex. 1)

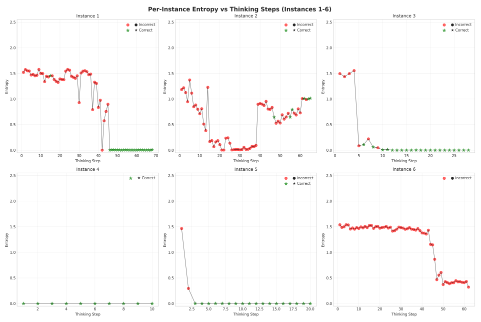


(d) Probe (ex. 2)

Figure 11: **7B model on AIME.** Per-instance signals over reasoning steps: confidence, entropy, and probe outputs.

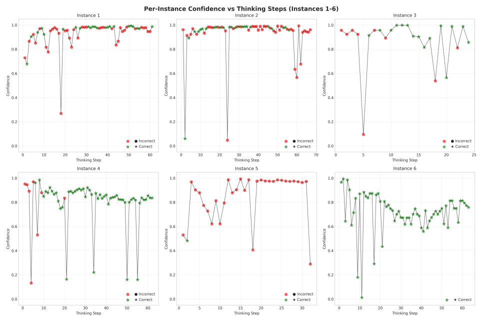


(a) Confidence

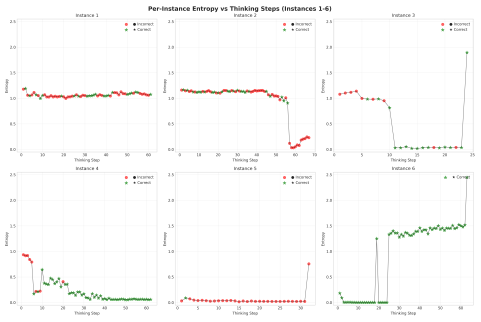


(b) Entropy

Figure 12: **32B model on AIME.** Per-instance confidence and entropy traces.



(a) Confidence



(b) Entropy

Figure 13: **32B model on GPQA-D.** Per-instance confidence and entropy traces.

E Risk Control Formalization

E.1 Notation and Data Splits

Let $\mathcal{X}$ be queries (problems) and $y^\star \in \mathcal{Y}$ the ground-truth answers. For each query $i \in \{1, \ldots, M\}$ the model generates a reasoning trajectory $\mathcal{T}_i = \{(t, \hat{y}_{i,t}, s_{i,t})\}_{t=1}^{T_i}$, where t indexes intermediate decision points (e.g., tokens, blocks, or checkpoints), $\hat{y}_{i,t}$ is the candidate answer at step t, and $s_{i,t} \in \mathbb{R}$ is a scalar *confidence signal*. Larger s means more confidence unless stated otherwise (for inverse signals we apply a monotone transform, Sec. E.3).

We partition data into:

- **Calibration set** $\mathcal{D}_{\text{cal}} = \{(\mathcal{T}_i, y_i^\star)\}_{i=1}^{M_{\text{cal}}}$ used to select thresholds and schedule parameters.
- **Test set** $\mathcal{D}_{\text{test}} = \{(\mathcal{T}_i, y_i^\star)\}_{i=1}^{M_{\text{test}}}$ used only for reporting.

We assume exchangeability between calibration and test (split-conformal style). Within-trajectory steps are dependent; guarantees are stated for *selected decisions* (often first-crossing events), not per-step i.i.d. samples.

E.2 Selective Risk, Coverage, and First-Crossing Events

Let a scalar threshold τ induce a *selective set* $S_\tau = \{i : s_i \geq \tau\}$ in the standard one-shot setting. The empirical selective risk is

$$R_n(\tau) \;=\; \frac{1}{|S_\tau|} \sum_{i \in S_\tau} \mathbf{1}\{\hat{y}_i \neq y_i^\star\}, \qquad C_n(\tau) = \frac{|S_\tau|}{n}.$$

Multi-step (reasoning) setting. A decision policy acts on a *trajectory prefix*. We define *first-crossing* indices to reduce multi-step dependence to a single selected step per query:

$$t_i^{\uparrow}(\tau_{\text{high}}) = \inf\{t : s_{i,t} \geq \tau_{\text{high}}\}, \qquad t_i^{\downarrow}(\tau_{\text{low}}(\cdot)) = \inf\{t : s_{i,t} \leq \tau_{\text{low}}(t)\}.$$

If the set is empty we define $t_i^{\uparrow} = +\infty$ (no high-confidence exit) or $t_i^{\downarrow} = +\infty$ (no low-confidence exit). A *dual-threshold* policy chooses the earliest of $t_i^{\uparrow}$ and $t_i^{\downarrow}$ when present, subject to guardrails (Sec. E.5).

Let $\mathcal{I}_{\text{sel}}$ be the indices of queries that produce a decision (accept or low-exit abstention) under a fixed policy. The empirical selective risk is

$$\widehat{R} = \frac{1}{|\mathcal{I}_{\text{acc}}|} \sum_{i \in \mathcal{I}_{\text{acc}}} \mathbf{1}\{\hat{y}_{i,t_i} \neq y_i^\star\}, \quad \widehat{C} = \frac{|\mathcal{I}_{\text{sel}}|}{M},$$

where $\mathcal{I}_{\text{acc}} \subseteq \mathcal{I}_{\text{sel}}$ are accepted instances and t_i is the decision step (first-crossing). When evaluating low-confidence exits as *abstentions*, we treat them as not contributing to numerator/denominator of $\widehat{R}$ (risk is defined over accepted answers). When evaluating *damage-control risk* (the probability that an early low exit loses a would-be correct final), we explicitly measure the performance-gap event (Sec. **??**).

E.3 Confidence Signals and Monotone Invariance

We support direct (larger = more confident) and inverse (smaller = more confident) signals. All decision rules are invariant to strictly monotone transforms g; i.e., thresholding s at τ is equivalent to thresholding $g(s)$ at $g(\tau)$. For inverse signals we use either $s' = -s$ or $s' = \text{rank}(s)$, then calibrate on s'.

E.4 Risk Guarantees and Calibration Methods

We give finite-sample, distribution-free procedures to select thresholds so that with probability at least $1-\delta$, the *true selective risk* of accepted answers is $\leq \varepsilon$.

Conformal Risk Control (CRC). Given n calibration points with indicators $Z_j = \mathbf{1}\{\hat{y}_j \neq y_j^\star\}$ among the selected set at τ, CRC's smoothed bound yields

$$\Pr\Big(R(\tau) \leq \tfrac{n}{n+1} R_n(\tau) + \tfrac{1}{n+1}\Big) \geq \tfrac{n+1}{n+2}.$$

We choose

$$\tau^\star_{\text{high}} \ = \ \inf\Big\{\tau : \tfrac{n}{n+1} R_n(\tau) + \tfrac{1}{n+1} \leq \varepsilon\Big\}.$$

This treats only *accepted* decisions (first-crossing at high). The same template applies to any monotone transform of the signal.

Hoeffding-Upper Confidence Bound (UCB) Let $N(\tau)$ be selected samples and $\widehat{R}(\tau)$ their empirical risk. Hoeffding's inequality gives: with probability $\geq 1-\delta$,

$$R(\tau) \leq \widehat{R}(\tau) + \sqrt{\tfrac{\log(1/\delta)}{2N(\tau)}}.$$

We pick

$$\tau^\star_{\text{high}} \ = \ \inf\left\{\tau : \widehat{R}(\tau) + \sqrt{\tfrac{\log(1/\delta)}{2N(\tau)}} \leq \varepsilon\right\}.$$

Learn-Then-Test (LTT). Let $\tau^\dagger \in \arg\min_\tau R_n(\tau)$ subject to $R_n(\tau) \leq \varepsilon$. Then perform a one-sided binomial test for $H_0 : R(\tau^\dagger) > \varepsilon$. With K observed errors among N accepted points, the p-value is

$$p \ = \ \Pr\{\text{Bin}(N, \varepsilon) \leq K\}.$$

Accept $\tau^\dagger$ if $p \leq \delta$. Equivalently, compute the Clopper–Pearson upper bound $\text{CP}_{1-\delta}(K, N)$ and accept if $\text{CP}_{1-\delta}(K, N) \leq \varepsilon$.

E.5 Dual-Threshold Policy and First-Crossing Semantics

High-confidence early exit. The policy accepts at the first step $t_i^{\uparrow}(\tau_{\text{high}})$ if it exists; the returned answer is $\hat{y}_{i,t_i^{\uparrow}}$. Calibration for τ_{high} controls *selective error* among accepted answers.

Low-confidence early exit (damage control). Define a time-varying lower schedule $\tau_{\text{low}}(t)$ (Sec. E.6). We *abstain* when $t_i^{\downarrow}(\tau_{\text{low}}) < t_i^{\uparrow}(\tau_{\text{high}})$ and $t_i^{\downarrow} \leq T_i$ (no earlier acceptance). The central risk here is *performance-gap*:

$$\underbrace{\hat{y}_{i,t_i^{\downarrow}} \neq y_i^{\star}}_{\text{current is wrong}} \quad \text{and} \quad \underbrace{\hat{y}_{i,\text{final}} = y_i^{\star}}_{\text{would recover if continued}}.$$

Let $\mathcal{E}_{\text{gap}}$ denote this event. We control $\Pr(\mathcal{E}_{\text{gap}} \mid t_i^{\downarrow} \text{ occurs}) \leq \varepsilon$ by calibrating $\tau_{\text{low}}(\cdot)$ on calibration traces that are *budget-forced* to observe counterfactual recovery (Sec. E.7). In practice we mark an event as *recoverable* if the final answer after full budget matches $y^{\star}$ while the early state was incorrect.

E.6 Parametric Low-Confidence Schedules

We use a logistic schedule for *direct* confidence:

$$\tau_{\text{low}}(t) \;=\; \frac{\tau_{\text{high}}}{1+\exp(-c\,t+b)},$$

with slope $c > 0$ and shift b (we use b=4 by default; adjust on calibration). Early steps: large denominator $\Rightarrow$ easy to continue; later steps: $\tau_{\text{low}}(t) \nearrow \tau_{\text{high}}$, encouraging early exit when confidence remains persistently low.

For *inverse* signals (e.g., entropy), use a decreasing schedule such as

$$\tau_{\text{low}}(t) \;=\; \frac{\tau_{\text{high}}}{1+\exp(c\,t-b)},$$

and apply the exit rule with reversed inequality.

Fitting c (and b). On $\mathcal{D}_{\text{cal}}$, for a grid $\mathcal{C}$ (and optionally $\mathcal{B}$), simulate first-crossing with fixed τ_{high} (already calibrated) and candidate $\tau_{\text{low}}(\cdot)$. Compute the damage-control risk $\widehat{R}_{\text{gap}}$ and compute-savings subject to a constraint on *overall accuracy drop* (e.g., within $\pm 3\%$ of full-TTR). Choose the (c, b) that (i) satisfies risk constraints and (ii) maximizes a utility (e.g., tokens saved).

E.7 Budget-Forcing and Counterfactual Recovery

To judge whether a low exit would forego a correct final answer, we require observing the *counterfactual* final outcome. We follow a *budget-forcing* protocol: continue generation to a fixed thinking budget (e.g., 9k tokens) even if the model would otherwise stop, ensuring the final decision $\hat{y}_{i,\text{final}}$ is observed. This enables labeling $\mathcal{E}_{\text{gap}}$ during calibration.

E.8 Deployment-Time Decision Rule

At inference time (test set or real-time), the policy executes online:

Algorithm 3 Dual-threshold confidence-based early exiting

```
Require: Question $Q$, max token limit $T$
Require: Reasoning LLM $\theta$
Ensure: Answer string $A$
 1: Initialize reasoning line buffer $R \leftarrow []$ and line counter $n \leftarrow 0$
 2: while $|R| < T$ do
 3:     Generate new reasoning line:
        $r_n \leftarrow \text{GenerateNewLine}\big(Q,\ \langle\text{think}\rangle,\ R;\ \theta\big)$
 4:     Append $r_n$ to $R$;    update $n \leftarrow n + 1$
 5:     Compute confidence $c \leftarrow \text{PredictConfidence}(Q, R; \theta)$
 6:     if $c \geq \tau_{\text{high}}$ then
 7:         break
 8:     end if
 9:     Compute low threshold $\tau_\ell \leftarrow \tau_{\text{low}}(n)$
10:     if $c \leq \tau_\ell$ then
11:         break
12:     end if
13:     if $\langle/\text{think}\rangle$ is generated then
14:         break
15:     end if
16: end while
17: return $A \sim \text{GenerateTillEos}\big(Q,\ \langle\text{think}\rangle,\ R,\ \langle/\text{think}\rangle;\ \theta\big)$
```

Algorithm 4 #UA@K based early exiting

```
Require: Question $Q$, token limit $T$; Number of rollouts $K$, unique answer threshold $\Delta$
Require: Reasoning LLM $\theta$
Ensure: An answer string: $A$
 1: Initialize reasoning line buffer $R \leftarrow []$ and line counter $n \leftarrow 0$.
 2: while $|R| < T$ do
 3:     Generate new reasoning line $r_n \leftarrow \text{GenerateNewLine}(Q, \langle\text{think}\rangle, R;\ \theta)$
 4:     Append $r_n$ to $R$, update line counter $n \leftarrow n + 1$
 5:     Randomly generate $K$ answer rollouts
        $A_k \sim \text{GenerateTillEos}(Q,\ \langle\text{think}\rangle,\ R,\ \langle/\text{think}\rangle,\ \text{Final answer:}\backslash n;\ \theta),\ k \in 1, \ldots, K$
 6:     Extract and count all unique answers from $A_1, \ldots, A_K$, denoted as $U$
 7:     if $(|U| \leq \Delta)$ or $\langle/\text{think}\rangle$ is generated then
 8:         exit
 9:     end if
10: end while
11: return $A \sim \text{GenerateTillEos}(Q, \langle\text{think}\rangle, R, \langle/\text{think}\rangle;\ \theta)$
```

Algorithm 5 Confidence Risk Control (CRC) threshold calibration

```
Require: Calibration points $\{(c_i, y_i)\}_{i=1}^m$ where $c_i$ is confidence and $y_i$ is correctness
Require: Target risk level $\epsilon$, scaling factor $B = 1.0$
Ensure: Confidence threshold $\tau$
 1: Sort unique confidence values $\{c_1, c_2, \ldots, c_u\}$ in descending order
 2: for each confidence value $c_j$ do
 3:     $S_j \leftarrow \{(c_i, y_i) \mid c_i \geq c_j\}$                  ▷ Points with confidence $\geq c_j$
 4:     $k \leftarrow \sum_{(c_i, y_i) \in S_j} \mathbf{1}[y_i = \text{True}]$                  ▷ Count correct predictions
 5:     $n \leftarrow |S_j|$                                                  ▷ Total predictions
 6:     if $1 - \frac{k}{n} \leq \epsilon \cdot B$ then
 7:         return $c_j$                        ▷ Found threshold satisfying risk constraint
 8:     end if
 9: end for
10: return 1.0               ▷ Default to maximum threshold if no suitable threshold found
```

Algorithm 6 UCB-based threshold calibration

```
Require: Calibration points $\{(c_i, y_i)\}_{i=1}^m$ where $c_i$ is confidence and $y_i$ is correctness
Require: Target risk level $\epsilon$, confidence level $\delta$
Ensure: Confidence threshold $\tau$
 1: Initialize parameters $w$ and parameter table $W = \{w^1, \dots, w^N\}$ using a single epoch with naive
    initialization
 2: Initialize running mean
        $G \leftarrow \mathbb{E}_m \mathbb{E}_\xi \nabla \tilde{f}(w; m, \xi)$          ▷ Sum over $m$, closed-form over $\xi$
 3: repeat
 4:     Sample $n$ and $\epsilon$
 5:     Compute base gradient
        $g \leftarrow \nabla f(w; n, \epsilon)$
 6:     Compute control variate
        $c \leftarrow \mathbb{E}_m \mathbb{E}_\xi \nabla \tilde{f}(w^m; m, \tilde{\xi}) - \nabla \tilde{f}(w^n; n, \epsilon)$          ▷ Use $\mathbb{E}_m \mathbb{E}_\xi \nabla \tilde{f}(w^m; m, \tilde{\xi}) = G$
 7:     Update running mean
        $G \leftarrow G + \frac{1}{N} \mathbb{E}_\xi(\nabla \tilde{f}(w; n, \xi) - \nabla \tilde{f}(w^n; n, \xi))$          ▷ Closed-form over $\xi$
 8:     Update parameter table
        $w^n \leftarrow w$
 9:     Update parameters
        $w \leftarrow w - \lambda(g + c)$          ▷ Or use $g + c$ in any stochastic optimization algorithm
10: until convergence
```

E.9 Small-Sample Behavior

If $N(\tau)$ is small, bounds widen and coverage collapses. We enforce a minimum calibration support $N_{\min}$ (e.g., $N_{\min} = 50$). When $N < N_{\min}$ we (i) enlarge ε grid or (ii) back off to a more conservative τ (increasing τ_{high} or flattening τ_{low}).

E.10 Compute Accounting and Cost–Risk Frontier

Let B_{think} be the maximum thinking tokens and B_{final} final-answer tokens. For each query i with decision step t_i:

$$\text{cost}_i \;=\; c_{\text{tok}} \cdot \underbrace{\min(t_i, B_{\text{think}})}_{\text{thinking}} \;+\; c_{\text{tok}} \cdot B_{\text{final}},$$

with c_{tok} a per-token cost (or wall-time). Report:

$$\text{AvgCost} \;=\; \frac{1}{M}\sum_i \text{cost}_i, \quad \text{Savings\%} \;=\; 100 \cdot \frac{\text{AvgCost}_{\text{baseline}} - \text{AvgCost}_{\text{policy}}}{\text{AvgCost}_{\text{baseline}}}.$$

We plot frontiers in $(\widehat{R}, \widehat{C}, \text{AvgCost})$ space for families of policies (static vs parametric vs high-only).

E.11 Signal Extraction Details

Internal confidence. We use normalized log-probability over final-answer tokens: if $\hat{y}$ spans tokens $a_{1:K}$ with log-probs ℓ_k, define

$$s \;=\; \frac{1}{K}\sum_{k=1}^{K} \ell_k,$$

or in probability space, the geometric mean $\exp(s)$. For multi-choice, we may directly use the logit margin between top-1 and runner-up to sharpen discrimination; calibration uses exactly the chosen s.

Entropy. For multiple-choice, compute Shannon entropy of the (renormalized) distribution over answer options. For free-form, use beam-based entropy of the first token of the forced final-answer head. When beams expose only the top-B candidates, renormalize over those B.

Probe-based confidence. Apply the trained probe to last-token hidden states at step t to obtain $\pi_t \in [0,1]$ interpreted as correctness probability. We use $s_t = \mathrm{logit}(\pi_t)$ by default (monotone transform).

E.12 Calibration Routines

E.12.1 High-Threshold calibration (with CRC)

```
**Input:** calibration traces with $(\hat{y}_{i,t}, s_{i,t})$, grid $\mathcal{T}$ of candidate thresholds, target $\varepsilon$.
**For each** $\tau \in \mathcal{T}$:
    1. For each $i$, compute $t_i^{\uparrow}(\tau)$. If finite, mark $\mathrm{acc}_i = 1$ and error $Z_i = \mathbf{1}\{\hat{y}_{i,t_i^{\uparrow}} \neq y_i^{\star}\}$.
    2. Let $N(\tau) = \sum_i \mathrm{acc}_i$, $\widehat{R}(\tau) = \frac{1}{N(\tau)} \sum_i Z_i$.
    3. Form CRC bound $B_{\mathrm{CRC}}(\tau) = \frac{N(\tau)}{N(\tau)+1}\widehat{R}(\tau) + \frac{1}{N(\tau)+1}$.
**Return** $\tau_{\mathrm{high}}^{\star} = \inf\{\tau : B_{\mathrm{CRC}}(\tau) \leq \varepsilon\}$ (if none, report infeasible at this $\varepsilon$).
```

E.12.2 Lower-Threshold Calibration

```
**Input:** fixed $\tau_{\mathrm{high}}^{\star}$, grids $\mathcal{C}$ (and shift $\mathcal{B}$), target $\varepsilon_{\mathrm{gap}}$, optional accuracy-drop budget $\Delta_{\max}$.
**For each** $(c, b) \in \mathcal{C} \times \mathcal{B}$:
    1. For each $i$, compute first low-crossing $t_i^{\downarrow}$ and high-crossing $t_i^{\uparrow}$. If $t_i^{\downarrow} < t_i^{\uparrow}$, mark a low-exit.
    2. For each low-exit, mark *gap* if $\hat{y}_{i,\text{final}} = y_i^{\star}$ but $\hat{y}_{i,t_i^{\downarrow}} \neq y_i^{\star}$.
    3. Let $N_{\mathrm{low}}$ be # low-exits, and $\widehat{R}_{\mathrm{gap}} = \frac{\#\text{gap events}}{N_{\mathrm{low}}}$.
    4. Check risk constraint via CRC/UCB/LTT (replace counts accordingly).
    5. Compute utility (e.g., tokens saved) and accuracy drop relative to full-TTR. Discard if drop $> \Delta_{\max}$.
**Return** $(c, b)$ maximizing utility among feasible candidates.
```

Epsilon tuning for budget efficiency. Sweep $\varepsilon \in \mathcal{E}$; for each, re-calibrate τ_{high} and (c, b), evaluate on a held-out fold, and select ε that meets accuracy constraints while maximizing compute savings.

E.13 Evaluation Metrics

- **Selective risk** $\widehat{R}$ and **coverage** $\widehat{C}$ of accepted answers.
- **Damage-control risk** $\widehat{R}_{\mathrm{gap}}$ over low exits.
- **Overall accuracy** vs full-TTR baseline (allowing a small drop, e.g., $\leq 3\%$).
- **Compute** (AvgCost, Savings%) and **exit-point distributions**.
- **Risk–coverage curves** and **risk–compute frontiers**.

E.14 Bootstrap and Uncertainty Quantification

We form percentile or BCa confidence intervals over metrics via stratified bootstrap on queries (not steps). For each bootstrap replicate $b = 1, \ldots, B$:

1. Re-sample calibration queries with replacement; re-fit τ_{high} and (c, b).
2. Re-sample test queries with replacement; evaluate metrics.

Aggregate quantiles across b to form CIs for $\widehat{R}$, $\widehat{C}$, Savings%, and frontier points. For heavy computation, reduce to B=200 and sub-sample $\mathcal{C}/\mathcal{E}$ grids.

E.15 Practical Defaults

Unless otherwise noted, we use: δ=0.1, CRC for τ_{high}, logistic $\tau_{\text{low}}(t)$ with b=4, grid $\mathcal{C} = \{0.01, \ldots, 2.0\}$, $N_{\min}$=50, and accuracy-drop budget $\Delta_{\max}$=3% relative to full TTR. For inverse signals, we set $s' = -s$ and re-use the same routines.

E.16 Limitations

Risk control guarantees depend on the core assumptions of exchangeability and semantic stationarity of the data s_t. These guarantees can be compromised by domain drift, where the underlying data distribution changes over time. Damage-control risk estimation requires budget-forced traces to observe counterfactual scenarios. However, if the assigned budget is too restrictive, this process can underestimate performance gaps, leading to an inaccurate assessment of risk. Finally, using an insufficiently small calibration set forces the system to set overly conservative thresholds. This leads to low task coverage and ultimately results in negligible budget savings, if any.

E.17 Thresholding Strategies

The core challenge is defining the form of these thresholds. A static threshold uses a single, fixed value (e.g., $\tau_{\text{high}}(t) = C_{\text{high}}$) for all reasoning steps. However, this rigidity is suboptimal as the meaning of a confidence score changes as the reasoning context evolves.

A more effective approach is parametric thresholding, where the threshold is a function of variables such as the number of thinking steps, t. This allows the decision rule to adapt. For the lower threshold, a canonical choice is a logistic schedule:

$$\tau_{\text{low}}(t) = \frac{\tau_{\text{high}}}{1 + e^{(-c \cdot t + 4)}}$$

Here, τ_{high} is the calibrated high-confidence threshold, t is the number of thinking steps, and c is a steepness parameter. The schedule starts with a low threshold, making very early exits rare, and gradually increases toward τ_{high}, allowing exits only after prolonged low confidence signals that signify computations being wasted.

To find the optimal parameters for these thresholding functions (e.g., τ_{high} and the steepness parameter c) in a principled way, we use Distribution-Free Risk Control (DFRC). The goal of DFRC is to find a decision rule that guarantees a user-specified risk level ε is not exceeded on unseen data, with high probability $1 - \delta$.

Formally, we use a calibration dataset to find parameters for our threshold τ (which can be a function like $\tau_{\text{low}}(t)$) such that the true risk $R(\tau)$ is bounded:

$$\mathbb{P}\left[R(\tau) \leq \varepsilon\right] \geq 1 - \delta$$

DFRC allows us to select parameters that maximize computational savings while rigorously adhering to the desired performance guarantee. There are several algorithms to achieve this, including Conformal Risk Control (CRC), Upper Confidence Bound (UCB), and Learn-Then-Test (LTT), which we use to calibrate our parametric thresholds.

F Applying Risk Control With Different Signals

We explored the risk control over two different models over 4 datasets. We mainly use confidence and entropy as our signals due to the computational and time limitations. We include probing results for **DeepSeek-R1-Distill-Qwen-7B** on **AIME (1983–2024)**. We present the risk control result with different parametric families, but we mainly focus on the sigmoid family since it usually gives us the best empirical results with more token reduction while preserving the accuracy.

F.1 AIME

F.1.1 Higher + Lower Threshold Plots

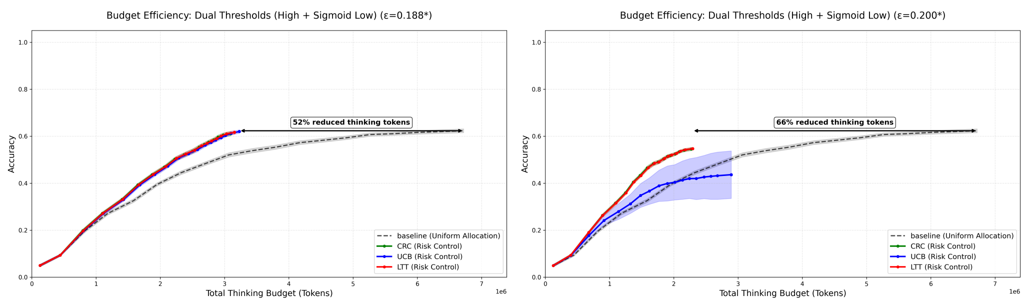


Figure 14: DeepSeek-7B on AIME: (Left) Sigmoid Confidence; (Right) Sigmoid Entropy.

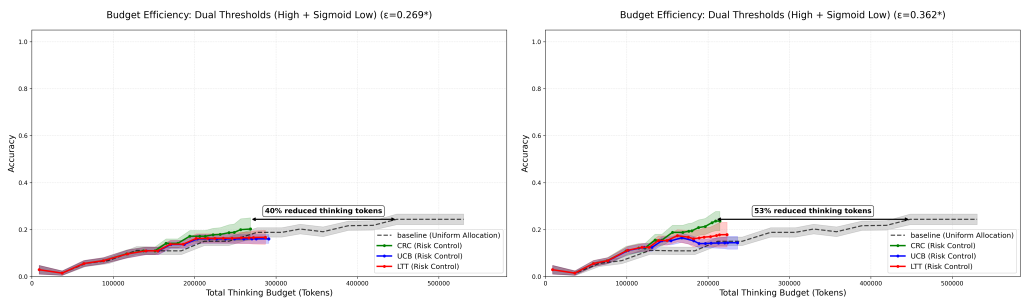


Figure 15: DeepSeek-7B on AIME: (Left) Probe Confidence with mixed training data from AIME and MATH; (Right) Probe Confidence with training data only on AIME.

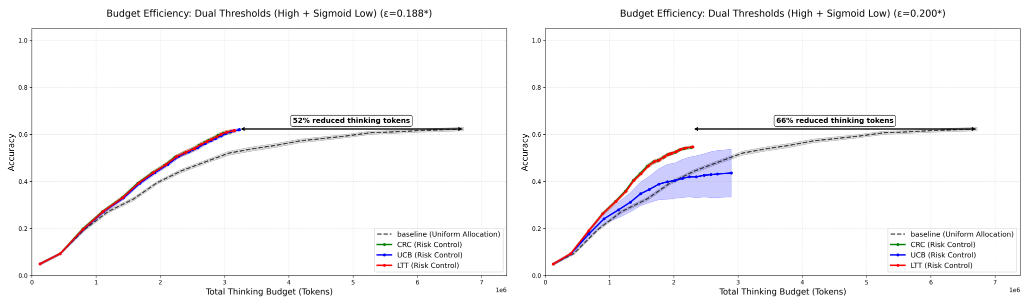


Figure 16: DeepSeek-32B on AIME: (Left) Sigmoid Confidence; (Right) Sigmoid Entropy.

Please note that the difference of 7B model's base results is due to the reason that the probes are trained on the 860 questions while tested only on the last 73 questions, while the confidence and entropy signals are tested on a held-out subset of the entire dataset.

F.1.2 Lower Threshold Only Plots

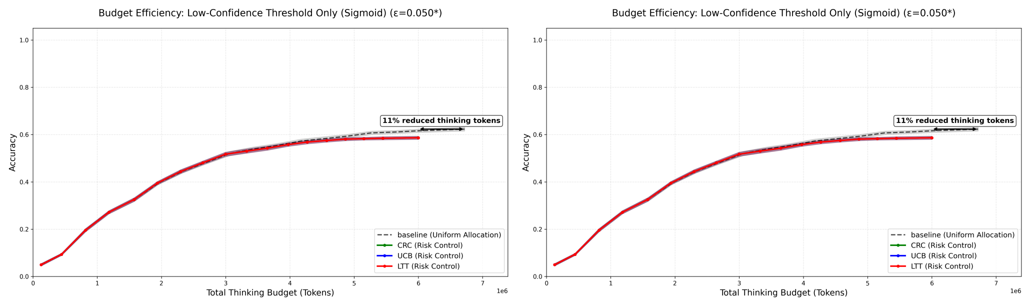


Figure 17: AIME (Lower-Threshold-Only, Confidence): (Left) DeepSeek-7B; (Right) DeepSeek-32B.

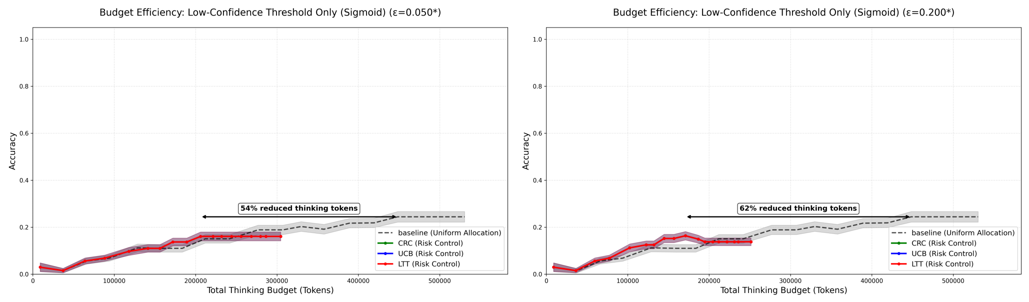


Figure 18: AIME (Lower-Threshold-Only, Probe Confidence): (Left) Probe trained with mixed AIME + MATH data; (Right) Probe trained only on AIME.

F.2 GPQA-D

F.2.1 Higher + Lower Threshold Plots

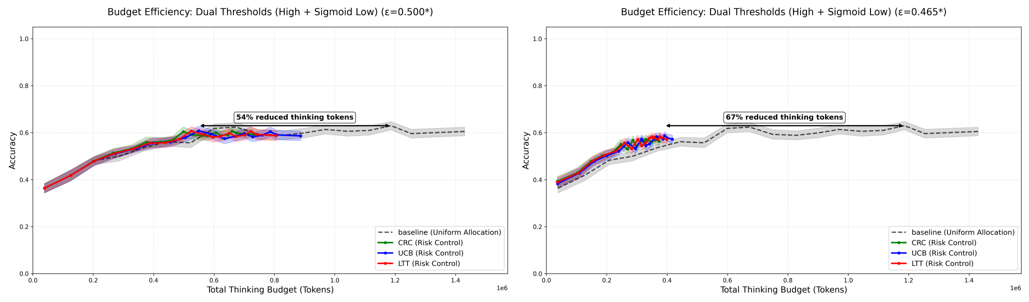


Figure 19: DeepSeek-32B on GPQA-D: (Left) Sigmoid Confidence; (Right) Sigmoid Entropy.

F.2.2 Lower Threshold Only Plots

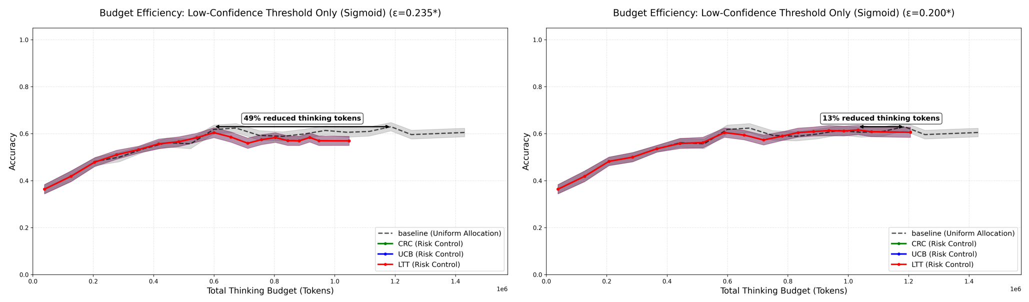


Figure 20: DeepSeek-32B on GPQA-D (Lower-Threshold-Only): (Left) Confidence; (Right) Entropy.

F.3 Murder Mysteries

F.3.1 Higher + Lower Threshold Plots

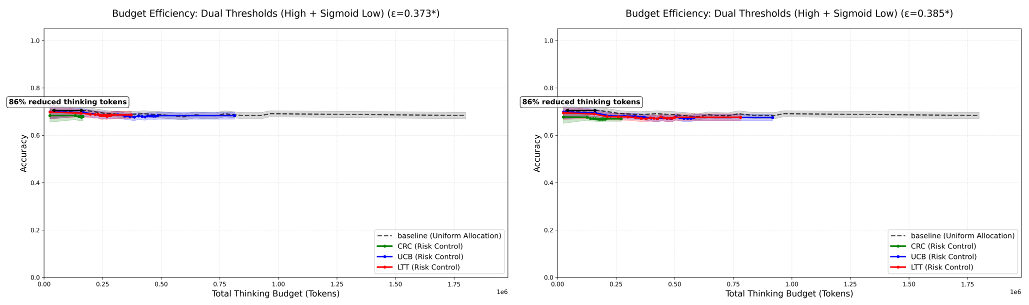


Figure 21: DeepSeek-32B on Murder Mysteries: (Left) Sigmoid Confidence; (Right) Sigmoid Entropy.

F.4 Object Placements

F.4.1 Higher + Lower Threshold Plots

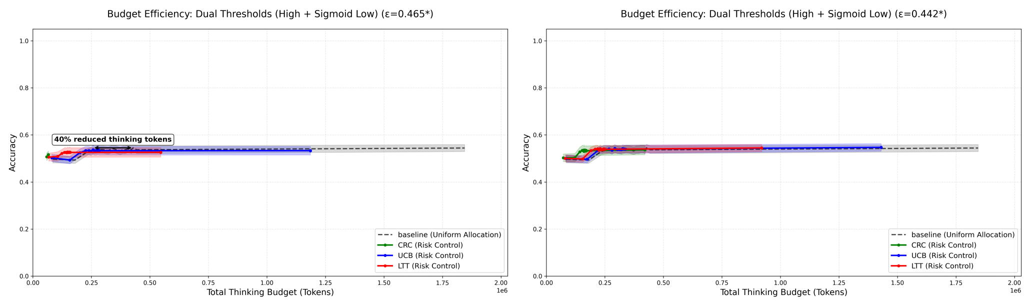


Figure 22: DeepSeek-32B on Object Placements: (Left) Sigmoid Confidence; (Right) Sigmoid Entropy.

F.5 HLE

For HLE, we only show the result on Deepseek-Distill-Qwen-7B with Entropy as the early stopping signal. The reason is that the model's poor performance on the dataset makes the token reduction graph less reliable.

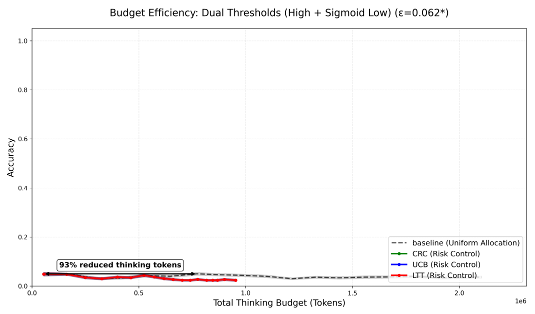


Figure 23: Deepseek-Distill-Qwen-7B with Entropy for Early Exiting

G Limitations

While our approach demonstrates promising results, several limitations remain. First, uncertainty calibration can be further improved, and we leave the development of more advanced calibration techniques to future work. Second, due to resource constraints, we were only able to apply probes to smaller models; extending this analysis to larger models is expected to yield stronger calibration performance. Finally, with the rapid emergence of new reasoning models, we plan to explore a broader range of architectures across diverse datasets in future studies.