TERMINATOR: LEARNING OPTIMAL EXIT POINTS FOR EARLY STOPPING IN CHAIN-OF-THOUGHT REASONING

Alliot Nagle*
UT Austin, US

Jakhongir Saydaliev
EPFL, Switzerland

Dhia Garbaya
EPFL, Switzerland
ENS Paris-Saclay, France

Michael Gastpar
EPFL, Switzerland

Ashok Vardhan Makkuva
Télécom Paris (IP Paris), France

Hyeji Kim
UT Austin, US

ABSTRACT

Large Reasoning Models (LRMs) achieve impressive performance on complex reasoning tasks via Chain-of-Thought (CoT) reasoning, which enables them to generate intermediate thinking tokens before arriving at the final answer. However, LRMs often suffer from significant *overthinking*, spending excessive compute time even after the answer is generated early on. Prior work has identified the existence of an optimal reasoning length such that truncating reasoning at this point significantly shortens CoT outputs with virtually no change in performance. However, determining optimal CoT lengths for practical datasets is highly non-trivial as they are fully task and model-dependent. In this paper, we precisely address this and design TERMINATOR, an early-exit strategy for LRMs at inference to mitigate overthinking. The central idea underpinning TERMINATOR is that the first arrival of an LRM's final answer is often predictable, and we leverage these first answer positions to create a novel dataset of optimal reasoning lengths to train TERMINATOR. Powered by this approach, TERMINATOR achieves significant reductions in CoT lengths of 14%–55% on average across four challenging practical datasets: MATH-500, AIME 2025, HumanEval, and GPQA, whilst outperforming current state-of-the-art methods.

1 INTRODUCTION

Large Reasoning Models (LRMs) extend Large Language Models (LLMs) by using test-time compute to explicitly reason before answering, achieving strong performance gains on challenging tasks OpenAI (2024). However, this comes at a substantial computational cost: LRMs often generate thousands of reasoning tokens, many of which are spent redundantly verifying solutions after the model has already produced the final answer, a phenomenon known as *overthinking* Guo et al. (2025); Luo et al. (2025); Chen et al. (2025). Prior work shows that reasoning traces can frequently be shortened by over 50% with little loss in accuracy Kang et al. (2025); Zhang et al. (2025b); Yang et al. (2025b), highlighting significant inefficiencies in LRM inference. To formalize this inefficiency, we introduce *hindsight-optimal reasoning length*: the minimum number of tokens an LRM needs to generate before first arriving at the same final answer it would produce with full reasoning. Using this notion, we identify the *hindsight-optimal exit* as the first logical occurrence of the final answer within the reasoning trace, and propose TERMINATOR, a novel inference-time early-exit method that leverages distinctive shifts in token-level confidence and usage patterns to substantially reduce CoT lengths on challenging benchmarks.

Main Contributions. In summary, we make the following contributions:

- We introduce the novel notion of hindsight-optimal reasoning, using which we show that the first arrival of an LRM's final answer is marked by observable and meaningful signals (Appendix B). To the best of our knowledge, this is the first such analysis of its kind.
- We design TERMINATOR, a novel inference-time early-exit algorithm for LRMs that leverages optimal-length CoTs (Sec. 3.2).

*Correspondence to Alliot Nagle: acnagle@utexas.edu

- We introduce a robust pipeline for identifying the first arrival of the final answer in CoTs, using which we construct a novel optimal-length CoT dataset (Sec. 3.1).

2 PRELIMINARIES

2.1 NOTATION

A Large Reasoning Model, LRM, takes as input a prompt $\boldsymbol{x}$ and generates a reasoning trace $\boldsymbol{r} = (r_1, \ldots, r_M)$ followed by a final solution $\boldsymbol{s}$ that contains the model's final answer $\hat{\boldsymbol{a}}$. The reasoning trace is generated auto-regressively as $r_i \sim \texttt{LRM}(\boldsymbol{x}, \boldsymbol{r}_{<i})$. Throughout the paper, $\hat{\boldsymbol{a}}$ always refers to the final answer obtained from the *full* CoT, not from any early-exited trace. For an early-exit method, we measure efficiency using the per-sample compression rate $\mathrm{CR} = M_{\text{early}}/M$, where M_{early} is the exit position in $\boldsymbol{r}$. Accuracy (Acc) denotes the fraction of problems where the produced answer matches the ground truth.

3 TERMINATOR: METHODOLOGY

Given a full CoT and the corresponding final solution from an LRM, the earliest logical arrival to the LRM's final answer can be detected in the CoT. However, reliable detection for tens of thousands of CoTs is a unique challenge, which we address through our pipeline in Sec. 3.1. We then present our method for training TERMINATOR, which is a probe classifier, in Sec. 3.2. An overview of our method is given in Fig. 3 of the appendix.

3.1 EARLY ANSWER EXTRACTION, IDENTIFICATION, AND VERIFICATION

We construct an automated pipeline that uses an LRM to (1) extract the final answer $\hat{\boldsymbol{a}}$ from the model's final solution $\boldsymbol{s}$, (2) identify the earliest span in the reasoning trace $\boldsymbol{r}$ that logically arrives at $\hat{\boldsymbol{a}}$, and (3) verify correctness of this identification. Simple pattern matching proved unreliable at scale, producing many false positives, whereas an LRM-based approach enables robust extraction across diverse formats. The LRM first extracts $\hat{\boldsymbol{a}}$ from $\boldsymbol{s}$ (where it is explicitly marked, e.g., via `\boxed`), then proposes a unique substring $\boldsymbol{d}$ of $\boldsymbol{r}$ that includes the earliest occurrence of $\hat{\boldsymbol{a}}$. A verification step confirms that $\boldsymbol{d}$ indeed contains $\hat{\boldsymbol{a}}$; failed attempts trigger retries with feedback, and traces with no valid span after a fixed budget are discarded. Upon success, $\boldsymbol{d}$ is located in $\boldsymbol{r}$ to recover the earliest answer token position $i^\star$ (see Algorithm 1 of the appendix).

3.2 TERMINATOR: BINARY PROBING CLASSIFIER

We train a binary probe $\boldsymbol{\theta}$ that predicts, at each CoT position $i \in [M]$, whether the final answer has already been generated. The probe operates on the LRM's final-layer hidden states $\boldsymbol{h}_i$ and reuses a copy of the LRM's final transformer block with an added prediction head. At each step, the model outputs $p_i = \mathbb{P}\boldsymbol{\theta}(b_i = 1 \mid \boldsymbol{x}, \boldsymbol{r} \leqslant i)$, indicating whether the hindsight-optimal exit has been reached. Training labels are derived from the extracted earliest answer positions, and predictions are made causally but independently across positions. To address severe class imbalance, we use class-weighted binary cross-entropy loss (Eq.1) with inverse-frequency weights (Eq.2). Unlike prior early-exit methods, TERMINATOR requires no threshold calibration and is trained jointly on hindsight-optimal CoTs from multiple domains, encouraging immediate exit upon first answer arrival.

$$L(\boldsymbol{\theta}) = -\frac{1}{M} \sum_{i=1}^{M} \Big[w_1 \cdot y_i \cdot \log p_i + w_0 \cdot (1 - y_i) \cdot \log(1 - p_i) \Big], \tag{1}$$

$$w_0 = \frac{n_0 + n_1}{2n_0}, \quad w_1 = \frac{n_0 + n_1}{2n_1}. \tag{2}$$

Table 1: **Performance of TERMINATOR and Baselines.** ↑ Indicates that higher values are better, while ↓ indicates that lower values are better. CR is the compression rate, reported here as the mean per-sample compression rate. **Bold** and Underlined values highlight the best and second-best performing early exit methods, respectively. TERMINATOR demonstrates superior accuracy-efficiency trade-offs (best or second-best performance across all 16 metrics).

Method	Math				Coding		Science		Overall	
	MATH-500		AIME25		HumanEval		GPQA			
	Acc↑	CR↓	Acc↑	CR↓	Acc↑	CR↓	Acc↑	CR↓	Acc↑	CR↓
Qwen3-8B										
Vanilla	91.1%	100%	74.4%	100%	86.4%	100%	57.6%	100%	77.4%	100%
NoThinking	80.7%	16.1%	22.0%	18.6%	78.5%	11.8%	30.7%	15.8%	53.0%	15.6%
DEER	79.9%	52.0%	21.4%	**67.8%**	77.4%	83.6%	50.5%	99.6%	57.3%	75.8%
Thought-Calib	90.1%	93.9%	65.8%	81.5%	71.8%	92.9%	**52.6%**	**78.9%**	70.1%	86.8%
TERMINATOR	**90.7%**	**45.1%**	**69.4%**	70.7%	**82.9%**	**69.9%**	52.1%	85.7%	**72.6%**	**67.8%**
Qwen3-14B										
Vanilla	92.0%	100%	79.9%	100%	76.0%	100%	59.9%	100%	77.0%	100%
NoThinking	84.1%	17.5%	26.3%	19.9%	78.3%	12.2%	32.1%	18.8%	55.2%	17.1%
DEER	80.9%	56.2%	27.6%	**71.0%**	80.7%	87.3%	49.0%	97.4%	59.6%	78.0%
Thought-Calib	89.8%	92.0%	63.3%	71.3%	74.8%	87.1%	**54.6%**	**81.9%**	70.6%	83.1%
TERMINATOR	**90.7%**	**46.8%**	**74.2%**	**71.0%**	**83.3%**	**70.9%**	53.9%	87.1%	**78.0%**	**65.0%**

4 EXPERIMENTS

4.1 IMPLEMENTATION DETAILS

Models. We train and evaluate our method on LRMs from two different model families: Qwen3-8B and Qwen3-14B Yang et al. (2025a), and Ministral-3-8B-Reasoning-2512 and Ministral-3-8B-Reasoning-2512 Liu et al. (2026). We use Qwen3-30B-A3B-Thinking-2507 for our answer extraction, identification, and verification pipeline. Our trained models consist of a single transformer layer initialized from the final layer of the LRM and a binary prediction head.

We compare TERMINATOR to prompt-based baselines (Vanilla, NoThinking (Ma et al., 2025a), DEER (Yang et al., 2025b)) and a probe-based baseline, Thought Calibration (Wu et al., 2025). Implementation details are in Appendix C.4.

Datasets. We form a training data mix with AIME (1983–2024) Art of Problem Solving, MATH Lightman et al. (2024), OpenCoder-SFT Huang et al. (2024), and OpenScience NVIDIA (2025). We form our training datasets by sampling three CoTs from each of these datasets, identifying the answer positions (see Sec. 3.1), and assigning the corresponding training labels. We evaluate our method and all baselines on AIME 2025 Art of Problem Solving, MATH-500 Lightman et al. (2024), HumanEval Chen et al. (2021), and GPQA Rein et al. (2024). Additional details on our training datasets are available in Appendix C.3.

4.2 EVALUATION OF THE PROPOSED METHOD

Table 1 shows the performance of TERMINATOR and relevant baselines with respect to the Compression Rate (lower is better) and Accuracy. All methods are evaluated using the same CoTs that are used in the vanilla baseline, except the NoThinking baseline, which is generated without CoT generation. Our results show state-of-the-art performance over early-exit methods from prior work. Results for the Ministral-3 models can be found in Appendix D.

4.3 ANALYTICAL RESULTS

We run ablation studies of TERMINATOR with respect to (1) out-of-distribution (OOD) performance, (2) its performance against the truncated early-exit baseline, and (3) latency and throughput over the vanilla baseline. Our results in this section will be reported on Qwen3-8B alone with results on Qwen3-14B, Ministral-3-8B, and Ministral-3-14B reported in Table 5 and Fig. 7. Here, we present the results for (1) and (2), while (3) along with some additional supplemental results are provided in Appendix D.

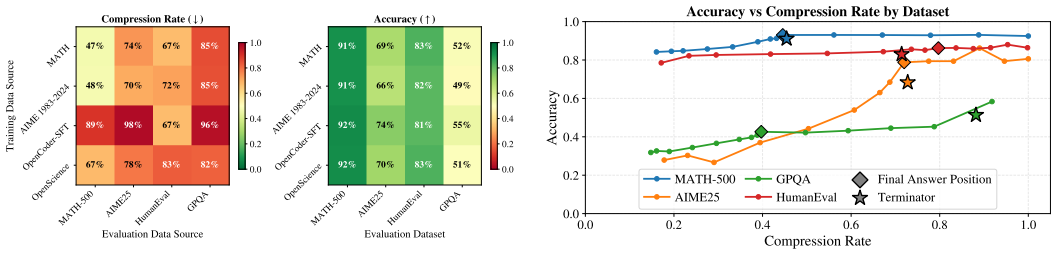

Figure 1: **OOD Performance of TERMINATOR (Left).** The out-of-distribution performance of TERMINATOR with respect to the compression rate **(left)** and the accuracy **(right)** for Qwen3-8B is shown. Training datasets are listed along the row axis, and the evaluation sets are listed across the column axis. For example, training TERMINATOR on MATH and evaluating on HumanEval yields 67% compression rate and 83% accuracy. Every training dataset has an in-domain evaluation dataset, i.e. MATH → MATH-500, AIME 1983–2024 → AIME25, OpenCoder-SFT → HumanEval, and OpenScience → GPQA.).
Effects of Early CoT Termination (Right). Test set CoTs are evaluated after truncating them at various points via `</think>` and asking the LRM for a final solution and answer. Diamond-shaped points show the hindsight-optimal reasoning length, and TERMINATOR falls close to optimality for three out of the four datasets.

Out-of-Distribution Evaluation. To better understand the out-of-distribution performance of TERMINATOR, we train on each of the four training datasets separately, and evaluate each resulting model on the test datasets. Fig. 1 shows heatmaps for the compression rate and the accuracy, where the row and column axes correspond to the training dataset and the testing dataset, respectively, and the value indicates the performance on the testing dataset. Observe that the best compression rate is achieved when the test dataset is in-distribution with the training dataset (i.e. along the diagonal). However, the best accuracy is not always on the diagonal; this is especially true for AIME25 evaluation, where training on AIME (1983–2024) yields the *lowest* accuracy. This makes sense when considering the compression rates for AIME25, as the better compression rates correspond to the lowest accuracy. This behavior does not happen with the other datasets, suggesting two things: (1) TERMINATOR tends to be overconfident on challenging tasks like AIME25 and is exiting too early, and (2) TERMINATOR is less confident when training on simpler tasks, thereby prolonging reasoning (worse compression rate) and increasing the accuracy. Training on OpenCoder-SFT shows behavior similar to item (2), yielding high (often the best) accuracy scores across all test sets but also the worst compression rates. This suggests that training to early-exit for coding tasks is not as useful for determining the early-exit position for the other tasks.

Hindsight-Optimal CoTs. Fig. 1 shows the accuracy with respect to CoT progress. Each dot on the curves represents the average accuracy when each CoT was truncated early, and the LRM was forced to give a final solution and final answer. We vary the truncation positions to cover the entire range of compression rates. The diamond-shaped dots represent the position of the first occurrence of $\hat{\boldsymbol{a}}$, and therefore represent the points where hindsight-optimal reasoning is achieved. As expected, the accuracy remains relatively constant after this point, suggesting that additional reasoning beyond $\hat{\boldsymbol{a}}$ does not yield significant accuracy gains, if any. We plot TERMINATOR alongside these curves to show how close it is to the hindsight-optimal CoT length and performance. Notably, TERMINATOR is close to hindsight-optimality for MATH-500, AIME25, and HumanEval, but is quite far from the hindsight-optimal compression rate for GPQA. This suggests that GPQA is a challenging dataset for TERMINATOR to generalize to, but it might be improved with a more rigorous data curation process for GPQA-style questions.

5 CONCLUSION

We present TERMINATOR, an early-exit method for LRM reasoning. Training TERMINATOR requires an optimal-length dataset of CoTs, which are obtainable through our robust answer extraction, identification, and verification pipeline. Furthermore, we also provide novel analysis and insights into the behaviors of an LRM's (1) Token-Confidence during reasoning (Fig. 2), and (2) shift in "thinking token" usage (Fig. 4). While our training data curation pipeline works well, future work can explore making training more efficient as tens of thousands of CoTs are used to train TERMINATOR.

ACKNOWLEDGEMENTS

This work was partly supported by ARO Award W911NF2310062, ONR Award N000142412542, NSF 2443857 and the 6G@UT center within the Wireless Networking and Communications Group (WNCG) at the University of Texas at Austin. Our work used the Vista cluster at the Texas Advanced Computing Center (TACC) at The University of Texas at Austin. We thank TACC for providing the high-performance computing resources that supported this research.

REFERENCES

Art of Problem Solving. Aime problems and solutions. https://artofproblemsolving.com/wiki/index.php/AIME_Problems_and_Solutions. Accessed: 2026-01-07.

Mark Chen, Jerry Tworek, Heewoo Jun, Qiming Yuan, Henrique Ponde de Oliveira Pinto, Jared Kaplan, Harrison Edwards, Yuri Burda, Nicholas Joseph, Greg Brockman, Alex Ray, Raul Puri, Gretchen Krueger, Michael Petrov, Heidy Khlaaf, Girish Sastry, Pamela Mishkin, Brooke Chan, Scott Gray, Nick Ryder, Mikhail Pavlov, Alethea Power, Lukasz Kaiser, Mohammad Bavarian, Clemens Winter, Philippe Tillet, Felipe Petroski Such, Dave Cummings, Matthias Plappert, Fotios Chantzis, Elizabeth Barnes, Ariel Herbert-Voss, William Hebgen Guss, Alex Nichol, Alex Paino, Nikolas Tezak, Jie Tang, Igor Babuschkin, Suchir Balaji, Shantanu Jain, William Saunders, Christopher Hesse, Andrew N. Carr, Jan Leike, Joshua Achiam, Vedant Misra, Evan Morikawa, Alec Radford, Matthew Knight, Miles Brundage, Mira Murati, Katie Mayer, Peter Welinder, Bob McGrew, Dario Amodei, Sam McCandlish, Ilya Sutskever, and Wojciech Zaremba. Evaluating large language models trained on code. *CoRR*, abs/2107.03374, 2021. URL https://arxiv.org/abs/2107.03374.

Xingyu Chen, Jiahao Xu, Tian Liang, Zhiwei He, Jianhui Pang, Dian Yu, Linfeng Song, Qiuzhi Liu, Mengfei Zhou, Zhuosheng Zhang, Rui Wang, Zhaopeng Tu, Haitao Mi, and Dong Yu. Do NOT think that much for 2+3=? on the overthinking of long reasoning models. In *Forty-second International Conference on Machine Learning*, 2025. URL https://openreview.net/forum?id=MSbU3L7V00.

Jeffrey Cheng and Benjamin Van Durme. Compressed chain of thought: Efficient reasoning through dense representations, 2024. URL https://arxiv.org/abs/2412.13171.

Alexis Chevalier, Alexander Wettig, Anirudh Ajith, and Danqi Chen. Adapting language models to compress contexts. In Houda Bouamor, Juan Pino, and Kalika Bali (eds.), *Proceedings of the 2023 Conference on Empirical Methods in Natural Language Processing*, pp. 3829–3846, Singapore, December 2023. Association for Computational Linguistics. doi: 10.18653/v1/2023.emnlp-main.232. URL https://aclanthology.org/2023.emnlp-main.232/.

Bowen Ding, Yuhan Chen, Futing Wang, Lingfeng Ming, and Tao Lin. Do thinking tokens help or trap? towards more efficient large reasoning model, 2025. URL https://arxiv.org/abs/2506.23840.

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, 2025a. URL https://arxiv.org/abs/2412.20993.

Yichao Fu, Xuewei Wang, Yuandong Tian, and Jiawei Zhao. Deep think with confidence, 2025b. URL https://arxiv.org/abs/2508.15260.

Tao Ge, Hu Jing, Lei Wang, Xun Wang, Si-Qing Chen, and Furu Wei. In-context autoencoder for context compression in a large language model. In *The Twelfth International Conference on Learning Representations*, 2024. URL https://openreview.net/forum?id=uREj4ZuGJE.

Daya Guo, Dejian Yang, Haowei Zhang, Junxiao Song, Peiyi Wang, Qihao Zhu, Runxin Xu, Ruoyu Zhang, Shirong Ma, 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, Chong Ruan, Damai Dai, Deli Chen, Dongjie Ji, Erhang Li, Fangyun Lin, Fucong Dai, Fuli Luo, Guangbo Hao, Guanting Chen, Guowei Li,

H. Zhang, Hanwei Xu, Honghui Ding, Huazuo Gao, Hui Qu, Hui Li, Jianzhong Guo, Jiashi Li, Jingchang Chen, Jingyang Yuan, Jinhao Tu, Junjie Qiu, Junlong Li, J. L. Cai, Jiaqi Ni, Jian Liang, Jin Chen, Kai Dong, Kai Hu, Kaichao You, 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, Mingxu Zhou, 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, Tao Yun, Tian Pei, Tianyu Sun, T. Wang, Wangding Zeng, 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, 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 incentivizes reasoning in llms through reinforcement learning. *Nature*, 645 (8081):633–638, September 2025. ISSN 1476-4687. doi: 10.1038/s41586-025-09422-z. URL http://dx.doi.org/10.1038/s41586-025-09422-z.

Shibo Hao, Sainbayar Sukhbaatar, DiJia Su, Xian Li, Zhiting Hu, Jason Weston, and Yuandong Tian. Training large language models to reason in a continuous latent space, 2025. URL https://arxiv.org/abs/2412.06769.

Siming Huang, Tianhao Cheng, Jason Klein Liu, Jiaran Hao, Liuyihan Song, Yang Xu, J. Yang, J. H. Liu, Chenchen Zhang, Linzheng Chai, Ruifeng Yuan, Zhaoxiang Zhang, Jie Fu, Qian Liu, Ge Zhang, Zili Wang, Yuan Qi, Yinghui Xu, and Wei Chu. Opencoder: The open cookbook for top-tier code large language models. 2024. URL https://arxiv.org/pdf/2411.04905.

Guochao Jiang, Guofeng Quan, Zepeng Ding, Ziqin Luo, Dixuan Wang, and Zheng Hu. Flashthink: An early exit method for efficient reasoning, 2025. URL https://arxiv.org/abs/2505.13949.

Huiqiang Jiang, Qianhui Wu, Xufang Luo, Dongsheng Li, Chin-Yew Lin, Yuqing Yang, and Lili Qiu. LongLLMLingua: Accelerating and enhancing LLMs in long context scenarios via prompt compression. In *ICLR 2024 Workshop on Mathematical and Empirical Understanding of Foundation Models*, 2024. URL https://openreview.net/forum?id=9YvfRrpmyw.

Hoyoun Jung and Kyung-Joong Kim. Discrete prompt compression with reinforcement learning. *IEEE Access*, 12:72578–72587, 2024. ISSN 2169-3536. doi: 10.1109/access.2024.3403426. URL http://dx.doi.org/10.1109/ACCESS.2024.3403426.

Yu Kang, Xianghui Sun, Liangyu Chen, and Wei Zou. C3ot: Generating shorter chain-of-thought without compromising effectiveness. *Proceedings of the AAAI Conference on Artificial Intelligence*, 39(23):24312–24320, Apr. 2025. doi: 10.1609/aaai.v39i23.34608. URL https://ojs.aaai.org/index.php/AAAI/article/view/34608.

Woosuk Kwon, Zhuohan Li, Siyuan Zhuang, Ying Sheng, Lianmin Zheng, Cody Hao Yu, Joseph E. Gonzalez, Hao Zhang, and Ion Stoica. Efficient memory management for large language model serving with pagedattention. In *Proceedings of the ACM SIGOPS 29th Symposium on Operating Systems Principles*, 2023.

Hunter Lightman, Vineet Kosaraju, Yuri Burda, Harrison Edwards, Bowen Baker, Teddy Lee, Jan Leike, John Schulman, Ilya Sutskever, and Karl Cobbe. Let's verify step by step. In *ICLR*, 2024. URL https://openreview.net/forum?id=v8L0pN6EOi.

Alexander H. Liu, Kartik Khandelwal, Sandeep Subramanian, Victor Jouault, Abhinav Rastogi, Adrien Sadé, Alan Jeffares, Albert Jiang, Alexandre Cahill, Alexandre Gavaudan, Alexandre Sablayrolles, Amélie Héliou, Amos You, Andy Ehrenberg, Andy Lo, Anton Eliseev, Antonia Calvi, Avinash Sooriyarachchi, Baptiste Bout, Baptiste Rozière, Baudouin De Monicault, Clémence Lanfranchi, Corentin Barreau, Cyprien Courtot, Daniele Grattarola, Darius Dabert, Diego de las Casas, Elliot Chane-Sane, Faruk Ahmed, Gabrielle Berrada, Gaëtan Ecrepont, Gauthier Guinet, Georgii Novikov, Guillaume Kunsch, Guillaume Lample, Guillaume Martin, Gunshi Gupta, Jan Ludziejewski, Jason Rute, Joachim Studnia, Jonas Amar, Joséphine Delas, Josselin Somerville Roberts, Karmesh Yadav, Khyathi Chandu, Kush Jain, Laurence Aitchison, Laurent Fainsin, Léonard Blier, Lingxiao Zhao, Louis Martin, Lucile Saulnier, Luyu Gao, Maarten Buyl, Margaret Jennings, Marie Pellat, Mark Prins, Mathieu Poirée, Mathilde Guillaumin, Matthieu Dinot, Matthieu Futeral, Maxime Darrin, Maximilian Augustin, Mia Chiquier, Michel Schimpf, Nathan Grinsztajn, Neha Gupta, Nikhil Raghuraman, Olivier Bousquet, Olivier Duchenne, Patricia Wang, Patrick von Platen, Paul Jacob, Paul Wambergue, Paula Kurylowicz, Pavankumar Reddy Muddireddy, Philomène Chagniot, Pierre Stock, Pravesh Agrawal, Quentin Torroba, Romain Sauvestre, Roman Soletskyi, Rupert Menneer, Sagar Vaze, Samuel Barry, Sanchit Gandhi, Siddhant Waghjale, Siddharth Gandhi, Soham Ghosh, Srijan Mishra, Sumukh Aithal, Szymon Antoniak, Teven Le Scao, Théo Cachet, Theo Simon Sorg, Thibaut Lavril, Thiziri Nait Saada, Thomas Chabal, Thomas Foubert, Thomas Robert, Thomas Wang, Tim Lawson, Tom Bewley, Tom Bewley, Tom Edwards, Umar Jamil, Umberto Tomasini, Valeriia Nemychnikova, Van Phung, Vincent Maladière, Virgile Richard, Wassim Bouaziz, Wen-Ding Li, William Marshall, Xinghui Li, Xinyu Yang, Yassine El Ouahidi, Yihan Wang, Yunhao Tang, and Zaccharie Ramzi. Ministral 3, 2026. URL https://arxiv.org/abs/2601.08584.

Xin Liu and Lu Wang. Answer convergence as a signal for early stopping in reasoning, 2025. URL https://arxiv.org/abs/2506.02536.

Steven J. Luck. *An introduction to the event-related potential technique / Steven J. Luck.* The MIT Press, Cambridge, Massachusetts, second edition. edition, 2014. ISBN 0-262-32406-7.

Haotian Luo, Li Shen, Haiying He, Yibo Wang, Shiwei Liu, Wei Li, Naiqiang Tan, Xiaochun Cao, and Dacheng Tao. O1-pruner: Length-harmonizing fine-tuning for o1-like reasoning pruning, 2025. URL https://arxiv.org/abs/2501.12570.

Wenjie Ma, Jingxuan He, Charlie Snell, Tyler Griggs, Sewon Min, and Matei Zaharia. Reasoning models can be effective without thinking, 2025a. URL https://arxiv.org/abs/2504.09858.

Xinyin Ma, Guangnian Wan, Runpeng Yu, Gongfan Fang, and Xinchao Wang. CoT-valve: Length-compressible chain-of-thought tuning. In Wanxiang Che, Joyce Nabende, Ekaterina Shutova, and Mohammad Taher Pilehvar (eds.), *Proceedings of the 63rd Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, pp. 6025–6035, Vienna, Austria, July 2025b. Association for Computational Linguistics. ISBN 979-8-89176-251-0. doi: 10.18653/v1/2025.acl-long.300. URL https://aclanthology.org/2025.acl-long.300/.

Minjia Mao, Bowen Yin, Yu Zhu, and Xiao Fang. Early stopping chain-of-thoughts in large language models, 2025. URL https://arxiv.org/abs/2509.14004.

Jesse Mu, Xiang Lisa Li, and Noah Goodman. Learning to compress prompts with gist tokens. In *Thirty-seventh Conference on Neural Information Processing Systems*, 2023. URL https://openreview.net/forum?id=2DtxPCL3T5.

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, 2025. URL https://arxiv.org/abs/2501.19393.

Alliot Nagle, Adway Girish, Marco Bondaschi, Michael Gastpar, Ashok Vardhan Makkuva, and Hyeji Kim. Fundamental limits of prompt compression: A rate-distortion framework for black-box language models. In A. Globerson, L. Mackey, D. Belgrave, A. Fan, U. Paquet, J. Tomczak, and C. Zhang (eds.), *Advances in Neural Information Processing Systems*, volume 37, pp. 94934–94970. Curran Associates, Inc., 2024. doi: 10.52202/

079017-3009. URL https://proceedings.neurips.cc/paper_files/paper/2024/file/ac8fbba029dadca99d6b8c3f913d3ed6-Paper-Conference.pdf.

NVIDIA. OpenScience dataset (v1). https://huggingface.co/datasets/nvidia/OpenScience, 2025. Last updated: June 18 (per repository history). Accessed: 2026-01-07.

OpenAI. Learning to reason with llms. https://openai.com/index/learning-to-reason-with-llms/, September 12 2024. Accessed: 2025-01-19.

Zhuoshi Pan, Qianhui Wu, Huiqiang Jiang, Menglin Xia, Xufang Luo, Jue Zhang, Qingwei Lin, Victor Ruhle, Yuqing Yang, Chin-Yew Lin, H. Vicky Zhao, Lili Qiu, and Dongmei Zhang. LLMLingua-2: Data distillation for efficient and faithful task-agnostic prompt compression. In Lun-Wei Ku, Andre Martins, and Vivek Srikumar (eds.), *Findings of the Association for Computational Linguistics ACL 2024*, pp. 963–981, Bangkok, Thailand and virtual meeting, August 2024. Association for Computational Linguistics. URL https://aclanthology.org/2024.findings-acl.57.

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

Guanghui Qin, Corby Rosset, Ethan Chau, Nikhil Rao, and Benjamin Van Durme. Dodo: Dynamic contextual compression for decoder-only LMs. In Lun-Wei Ku, Andre Martins, and Vivek Srikumar (eds.), *Proceedings of the 62nd Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, pp. 9961–9975, Bangkok, Thailand, August 2024. Association for Computational Linguistics. doi: 10.18653/v1/2024.acl-long.536. URL https://aclanthology.org/2024.acl-long.536/.

David Rein, Betty Li Hou, Asa Cooper Stickland, Jackson Petty, Richard Yuanzhe Pang, Julien Dirani, Julian Michael, and Samuel R. Bowman. GPQA: A graduate-level google-proof q&a benchmark. In *First Conference on Language Modeling*, 2024. URL https://openreview.net/forum?id=Ti67584b98.

Tal Schuster, Adam Fisch, Jai Gupta, Mostafa Dehghani, Dara Bahri, Vinh Tran, Yi Tay, and Donald Metzler. Confident adaptive language modeling. In S. Koyejo, S. Mohamed, A. Agarwal, D. Belgrave, K. Cho, and A. Oh (eds.), *Advances in Neural Information Processing Systems*, volume 35, pp. 17456–17472. Curran Associates, Inc., 2022. URL https://proceedings.neurips.cc/paper_files/paper/2022/file/6fac9e316a4ae75ea244ddcef1982c71-Paper-Conference.pdf.

Chenlong Wang, Yuanning Feng, Dongping Chen, Zhaoyang Chu, Ranjay Krishna, and Tianyi Zhou. Wait, we don't need to "wait"! removing thinking tokens improves reasoning efficiency. In Christos Christodoulopoulos, Tanmoy Chakraborty, Carolyn Rose, and Violet Peng (eds.), *Findings of the Association for Computational Linguistics: EMNLP 2025*, pp. 7459–7482, Suzhou, China, November 2025a. Association for Computational Linguistics. ISBN 979-8-89176-335-7. doi: 10.18653/v1/2025.findings-emnlp.394. URL https://aclanthology.org/2025.findings-emnlp.394/.

Xi Wang, James McInerney, Lequn Wang, and Nathan Kallus. Entropy after ⟨/Think⟩ for reasoning model early exiting, 2025b. URL https://arxiv.org/abs/2509.26522.

Menghua Wu, Cai Zhou, Stephen Bates, and Tommi Jaakkola. Thought calibration: Efficient and confident test-time scaling. In Christos Christodoulopoulos, Tanmoy Chakraborty, Carolyn Rose, and Violet Peng (eds.), *Proceedings of the 2025 Conference on Empirical Methods in Natural Language Processing*, pp. 14302–14316, Suzhou, China, November 2025. Association for Computational Linguistics. ISBN 979-8-89176-332-6. doi: 10.18653/v1/2025.emnlp-main.722. URL https://aclanthology.org/2025.emnlp-main.722/.

Heming Xia, Chak Tou Leong, Wenjie Wang, Yongqi Li, and Wenjie Li. TokenSkip: Controllable chain-of-thought compression in LLMs. In Christos Christodoulopoulos, Tanmoy Chakraborty, Carolyn Rose, and Violet Peng (eds.), *Proceedings of the 2025 Conference on Empirical Methods in Natural Language Processing*, pp. 3351–3363, Suzhou, China, November 2025. Association for

Computational Linguistics. ISBN 979-8-89176-332-6. doi: 10.18653/v1/2025.emnlp-main.165. URL https://aclanthology.org/2025.emnlp-main.165/.

An Yang, Anfeng Li, Baosong Yang, Beichen Zhang, Binyuan Hui, Bo Zheng, Bowen Yu, Chang Gao, Chengen Huang, Chenxu Lv, Chujie Zheng, Dayiheng Liu, Fan Zhou, Fei Huang, Feng Hu, Hao Ge, Haoran Wei, Huan Lin, Jialong Tang, Jian Yang, Jianhong Tu, Jianwei Zhang, Jianxin Yang, Jiaxi Yang, Jing Zhou, Jingren Zhou, Junyang Lin, Kai Dang, Keqin Bao, Kexin Yang, Le Yu, Lianghao Deng, Mei Li, Mingfeng Xue, Mingze Li, Pei Zhang, Peng Wang, Qin Zhu, Rui Men, Ruize Gao, Shixuan Liu, Shuang Luo, Tianhao Li, Tianyi Tang, Wenbiao Yin, Xingzhang Ren, Xinyu Wang, Xinyu Zhang, Xuancheng Ren, Yang Fan, Yang Su, Yichang Zhang, Yinger Zhang, Yu Wan, Yuqiong Liu, Zekun Wang, Zeyu Cui, Zhenru Zhang, Zhipeng Zhou, and Zihan Qiu. Qwen3 technical report, 2025a. URL https://arxiv.org/abs/2505.09388.

Chenxu Yang, Qingyi Si, Yongjie Duan, Zheliang Zhu, Chenyu Zhu, Qiaowei Li, Minghui Chen, Zheng Lin, and Weiping Wang. Dynamic early exit in reasoning models, 2025b. URL https://arxiv.org/abs/2504.15895.

Rubing Yang, Huajun Bai, Song Liu, Guanghua Yu, Runzhi Fan, Yanbin Dang, Jiejing Zhang, Kai Liu, Jianchen Zhu, and Peng Chen. Specexit: Accelerating large reasoning model via speculative exit, 2025c. URL https://arxiv.org/abs/2509.24248.

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. In *Second Conference on Language Modeling*, 2025a. URL https://openreview.net/forum?id=O6I0Av7683.

Jiajie Zhang, Nianyi Lin, Lei Hou, Ling Feng, and Juanzi Li. AdaptThink: Reasoning models can learn when to think. In Christos Christodoulopoulos, Tanmoy Chakraborty, Carolyn Rose, and Violet Peng (eds.), *Proceedings of the 2025 Conference on Empirical Methods in Natural Language Processing*, pp. 3716–3730, Suzhou, China, November 2025b. Association for Computational Linguistics. ISBN 979-8-89176-332-6. doi: 10.18653/v1/2025.emnlp-main.184. URL https://aclanthology.org/2025.emnlp-main.184/.

Jintian Zhang, Yuqi Zhu, Mengshu Sun, Yujie Luo, Shuofei Qiao, Lun Du, Da Zheng, Huajun Chen, and Ningyu Zhang. LightThinker: Thinking step-by-step compression. In Christos Christodoulopoulos, Tanmoy Chakraborty, Carolyn Rose, and Violet Peng (eds.), *Proceedings of the 2025 Conference on Empirical Methods in Natural Language Processing*, pp. 13307–13328, Suzhou, China, November 2025c. Association for Computational Linguistics. ISBN 979-8-89176-332-6. doi: 10.18653/v1/2025.emnlp-main.673. URL https://aclanthology.org/2025.emnlp-main.673/.

Zhen Zhang, Xuehai He, Weixiang Yan, Ao Shen, Chenyang Zhao, Shuohang Wang, Yelong Shen, and Xin Eric Wang. Soft thinking: Unlocking the reasoning potential of llms in continuous concept space, 2025d. URL https://arxiv.org/abs/2505.15778.

A Appendix

A.1 Notation

A Large Reasoning Model LRM takes as input the prompt sequence $\boldsymbol{x} = (x_1, x_2, x_3, \ldots, x_L)$ and produces two outputs $\boldsymbol{r}$ and $\boldsymbol{s}$ auto-regressively (Fig. 3. Here $\boldsymbol{r} = (r_1, r_2, r_3, \ldots, r_M)$ is the CoT sequence generated during the thinking stage, i.e. $r_i = \texttt{LRM}(\boldsymbol{x}, \boldsymbol{r}_{<i})$ for $i \in [M] \triangleq \{1, \ldots, M\}$, and $\boldsymbol{s} = (s_1, s_2, s_3, \ldots, s_N)$ is the solution that summarizes this CoT and contains a final answer $\hat{\boldsymbol{a}}$, which could be a single numerical answer, a math expression, code, a multiple-choice option, etc. Here $s_j = \texttt{LRM}(\boldsymbol{x}, \boldsymbol{r}, \boldsymbol{s}_{<j})$ for $j \in [N]$. Note that the final answer $\hat{\boldsymbol{a}}$ is separate from the ground-truth answer $\boldsymbol{a}$; they may or may not be in agreement with each other. Throughout the paper, $\hat{\boldsymbol{a}}$ always refers to the final answer of the full CoT, not the final answer generated after exiting a CoT early. For any early-exit strategy, a key metric to gauge its performance is the per-sample compression rate (CR): $\frac{M_{\text{early}}}{M}$, where $M_{\text{early}} \in [M]$ is the token index of early exit in $\boldsymbol{r}$. Accuracy (Acc) measures the proportion of problems where the correct answer is produced.

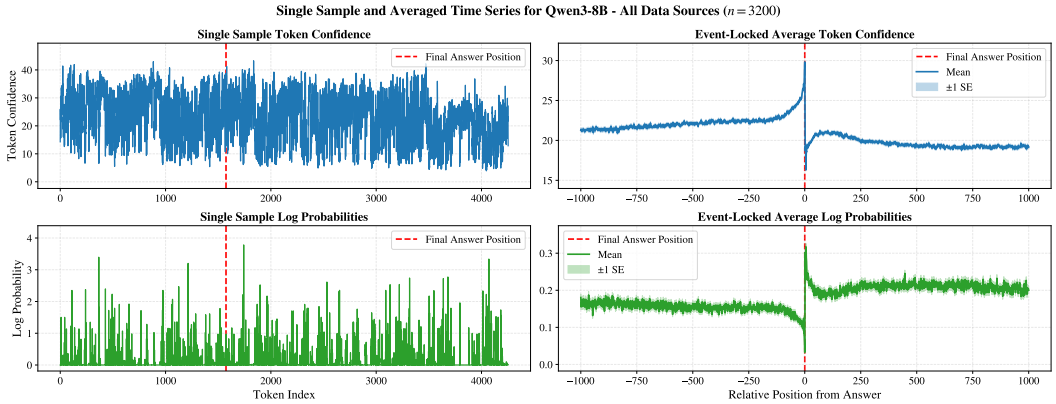

Figure 2: **Event-Locked Averaging of Token-Confidence.** Figures on the **left** show the Token-Confidence Fu et al. (2025b) and log-probability trajectories throughout reasoning for a single, randomly selected sample; figures on the **right** show the effect of *event-locked averaging* on the position of the first arrival of the final answer across all CoTs. The 3200 CoTs used are a random subset of our training set, which includes AIME (1983–2024), MATH, OpenCoder-SFT, and OpenScience. Event-locked averaging shows a consistent agreement on spiking behavior at the answer position in each CoT, but disagrees elsewhere.

A.2 Token-Confidence

Our analytical experiments require a measure of LRM's confidence during the generation of a CoT. To this end, we use the Token-Confidence metric, that gauges the uncertainty of a chosen token. Mathematically, for every $i \in [M]$, the corresponding Token-Confidence C_i is defined as

$$C_i \triangleq -\frac{1}{K} \sum_{k \in \mathcal{T}_K(i)} \log \mathbb{P}_{\texttt{LRM}}\left(r_i = k \mid \boldsymbol{x}, \boldsymbol{r}_{<i}\right), \tag{3}$$

where $\mathbb{P}_{\texttt{LRM}}\left(r_i = \cdot \mid \boldsymbol{x}, \boldsymbol{r}_{<i}\right)$ is the LRM prediction probability at position i and $\mathcal{T}_K(i) \triangleq \text{Top-}K\left[\mathbb{P}_{\texttt{LRM}}\left(r_i = \cdot \mid \boldsymbol{x}, \boldsymbol{r}_{<i}\right)\right]$ is the set of vocabulary tokens corresponding to the Top-K probabilities. In other words, Token-Confidence is the average log-probability across the Top-K probabilities (we set $K = 20$ in our experiments). The higher it is, the more confident the model is in its predictions.

B Motivation

Shortly after the breakthrough of LRMs, it was observed that they exhibit an overthinking phenomenon where, despite arriving at the correct answer, they continue to consider alternative solution paths, possibly leading to other incorrect answers Chen et al. (2025); Luo et al. (2025). While LRMs achieve greatly improved performance over their non-reasoning counterparts, they do so at a much higher inference-time cost: up to thousands of additional tokens are generated to form the CoT before arriving at the final solution Guo et al. (2025).

Once an LRM generates a CoT $\boldsymbol{r}$ and a final solution $\boldsymbol{s}$, we can observe the final answer $\hat{\boldsymbol{a}}$. Then, *in hindsight*, we can determine precisely where the LRM should have exited the CoT to avoid wasting tokens, and instead generate the final solution. First, we only need to check for $\hat{\boldsymbol{a}}$, not $\boldsymbol{a}$, in $\boldsymbol{r}$, as the LRM may not have ever generated the correct (ground-truth) answer, so it doesn't exist in $\boldsymbol{r}$. Second, by choosing to terminate reasoning after the arrival of $\hat{\boldsymbol{a}}$ in $\boldsymbol{r}$, all steps that are useful to arriving at $\hat{\boldsymbol{a}}$ are kept, and anything after is skipped as it is not necessary.

Detecting the Answer Early. We ran experiments to analyze a few different signals with potential for answer detection and report the results in Figs. 2 and 4. In particular, Fig. 2 shows the evolution of the Token-Confidences and log-probabilities for a randomly selected CoT in the left plots, and the *event-locked averaging* of Token-Confidences and log-probabilities across many CoTs in the right plots. To clarify, the event-locked average is formed by aligning the position of the first $\hat{\boldsymbol{a}}$ occurrence in each CoT's signal to 0, and then taking the average across many CoTs. Interestingly, while the single CoT plots do not show a clear indication of the arrival of $\hat{\boldsymbol{a}}$, the arrival is very clear in the event-locked averaged plots. Moreover, these plots use CoTs sampled with problems in math,

science, and coding, showing that this behavior is consistent across different data sources. Figs. 8 to 11 in the appendix show similar plots for each data source separately.

Fig. 4 shows the frequency of specific tokens before and after the first arrival of $\hat{\boldsymbol{a}}$ for three "thinking tokens," of which `hmm` in the left plot, `okay` in the middle plot, and `another` in the right plot are shown. "Thinking tokens" are special tokens often associated with thinking, such as `wait`, `so`, `alternatively`, `hmm`, `therefore`, etc., that suggest ongoing reasoning patterns Wang et al. (2025a); Qian et al. (2025); Ding et al. (2025). That is, their existence suggests a continuation of reasoning, which we aim to minimize, and we hypothesize that "thinking token" usage should change once $\hat{\boldsymbol{a}}$ has been generated. Each axis shows the frequency of the given token occurring before and after the answer, expressed as a rate computed as the raw count divided by the total number of tokens before and after the answer, respectively. The diameter of each point reflects the relative length of a CoT, where a smaller diameter corresponds to a shorter CoT and a larger diameter corresponds to a longer CoT. All three plots show a bias in the token rates before and after the first occurrence of $\hat{\boldsymbol{a}}$. Namely, `hmm` and `okay` occur more often before the answer, while `another` occurs more frequently after the answer. While not all "thinking tokens" show a clear bias before and after the answer is generated, this shift in the token-frequency distribution signals that $\hat{\boldsymbol{a}}$ has been generated. Please see Fig. 12 in the appendix for similar results on other "thinking tokens."

Moving to Online Inference. Although these results indicate the early arrival of $\hat{\boldsymbol{a}}$, applying them during online inference remains challenging. Event-locked averaging (Fig. 2) requires multiple CoTs with reliable answer position estimates to reveal spiking behavior, which is unavailable for a single CoT at inference time. Similarly, the token-rate analysis in Fig. 4 relies on full access to each $\boldsymbol{r}$ and the position of $\hat{\boldsymbol{a}}$, enabling post hoc rate computation. While both analyses reveal consistent shifts in model behavior around the first occurrence of $\hat{\boldsymbol{a}}$, converting these signals into a practical online inference algorithm remains non-trivial.

Our Approach. Under the hood of the LRM, there are clearly meaningful signals to indicate the earliest arrival of $\hat{\boldsymbol{a}}$. But since there are unique challenges to designing an online inference algorithm that exits *as early as possible* with hand-designed signals, we frame this problem as a prediction task.

The core idea of our approach is to train a probe classifier to predict when $\hat{\boldsymbol{a}}$ has been generated. However, we seek to do so at the token level, offering much finer-grained predictions than prior work. That is, our dataset is curated to perform a token classification task. To the best of our knowledge, all previous methods use much coarser granularity in their training dataset where they chunk each

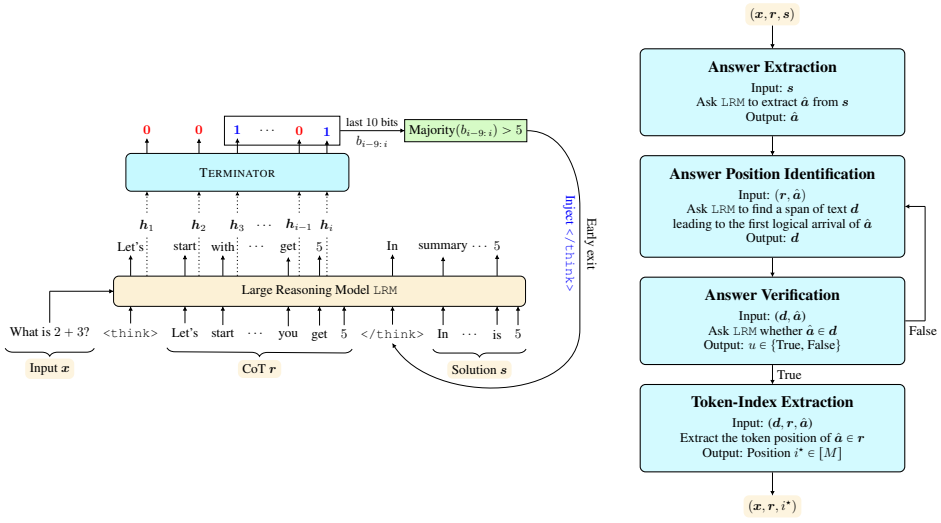

Figure 3: **Early stopping via TERMINATOR (Left).** TERMINATOR is a binary probe classifier that predicts whether to exit or not at every CoT token. Once the majority of prediction bits within a window (10 here) are 1, `</think>` is injected into the LRM's token stream to stop thinking (Sec. 3).
Training-Dataset Curation Process (Right). We use an LRM to (1) extract final answer $\hat{\boldsymbol{a}}$ from final solution $\boldsymbol{s}$, (2) identify the earliest position of $\hat{\boldsymbol{a}}$ in the CoT $\boldsymbol{r}$, and (3) verify that the position was correct. If it was, then we can extract the exact position of $\hat{\boldsymbol{a}}$ from the CoT and retry the identification step with feedback otherwise.

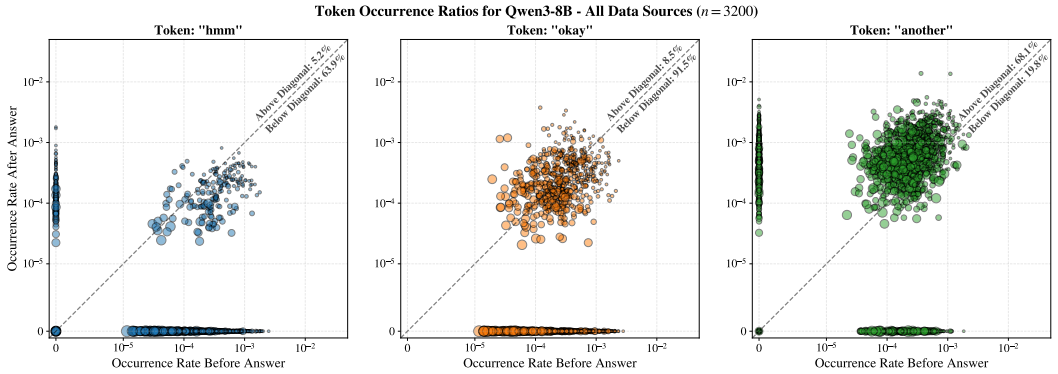

Figure 4: **Token Usage Frequency Shift.** "Thinking token" usage changes depending on whether the final answer has been generated in the CoT. Rates are computed by counting the raw number of occurrences of the token before and after the answer, and then normalizing each count by the respective number of tokens in the before and after bins. The arrival of the final answer is hinted at by changes in the rates for these tokens. The relative length of a CoT is captured by its dot size, where a longer CoT has a larger dot.

$\boldsymbol{r}$ according to some heuristic, such as "thinking tokens" or paragraph delimiters such as `\n\n`. Then, at inference time, they exit once the predicted probability crosses a data-calibrated threshold Liu & Wang (2025); Wu et al. (2025); Zhang et al. (2025a). The advantage of our approach at inference-time is two-fold: (1) a probe classifier trained with our dataset has the ability to exit immediately after $\hat{\boldsymbol{a}}$ is generated, and (2) while our approach is amenable to using a data-calibrated threshold, it is not necessary. The main drawback of data-calibrated thresholding is that it requires additional samples from the evaluation data distribution, and the resulting threshold is therefore specific to that distribution and may not transfer well to other distributions.

However, obtaining the $\hat{\boldsymbol{a}}$ positions to create our dataset in a scalable way is challenging and highly non-trivial, which we precisely address in the next section.

C Additional Details on Our Methods

C.1 Training and Inference.

During training, we optimize for high performance on a holdout validation set for our prediction task; we choose our model based solely on how well it performs on the binary predictive task, without peeking at the evaluation dataset performance. Our validation metric of choice is the Macro-F1 score.

We use vLLM Kwon et al. (2023) with asynchronous requests to sample CoTs when curating our training datasets. During inference with our trained model, a sliding window of the 10 most recent predictions is used, and the `</think>` token is injected when more than 50% of the labels are 1 (majority voting). We set the threshold of predicting 1 to 0.7.

C.2 Early Answer Extraction, Identification, and Verification

Algorithm 1 contains pseudocode for our pipeline. $i^\star$, the index of the earliest token position containing $\hat{\boldsymbol{a}}$, is used to construct the label set of our training data by setting all positions prior to $i^\star$ to 0 and setting all positions after $i^\star$ to 1. Each of the three steps (extraction, identification, and verification) requires separate calls to an LRM; please refer to our codebase for details on the exact system prompts that we used for each step.

C.3 Training Dataset Details

Our training dataset consists of CoTs from AIME (1983–2024) Art of Problem Solving, MATH Lightman et al. (2024), OpenCoder-SFT Huang et al. (2024), and OpenScience NVIDIA (2025). All 933 samples and all 12,000 samples from the AIME (1983–2024) and MATH datasets, respectively, are used. We randomly select 12,000 samples from the `educational_instruct` subset of the OpenCoder-SFT-Stage2 dataset, which we refer to as "OpenCoder-SFT" in the main paper. This

Algorithm 1 Answer Span Extraction, Identification, and Verification With Feedback

Input: CoT $\boldsymbol{r}$, final solution $\boldsymbol{s}$, `LRM`, `tokenizer`, max retries K
Output: answer position $i^\star$ (token index where answer is reached)

// Extract final answer value from model output
$\hat{\boldsymbol{a}} \leftarrow \texttt{LRM}(\text{“Extract the final answer from: ”} + \boldsymbol{s})$

// Iteratively extract and verify span with feedback
$\boldsymbol{z} \leftarrow \varnothing$ *// feedback provided to the LRM*
for $k = 1, \ldots, K$ **do**
  // Ask LRM to identify a string span containing the first occurrence of $\hat{\boldsymbol{a}}$ in $\boldsymbol{r}$
  $\boldsymbol{d} \leftarrow \texttt{LRM}(\text{“Find first occurrence of ”} + \hat{\boldsymbol{a}} + \text{“ in: ”} + \boldsymbol{r} + \boldsymbol{z})$

  // Verify the identified span contains the answer
  $v \leftarrow \texttt{LRM}(\text{“Does ”} + \boldsymbol{d} + \text{“ contain ”} + \hat{\boldsymbol{a}} + \text{“?”})$

  if $v ==$ true **then**
    break *// span verified, proceed*
  end if

  $\boldsymbol{z} \leftarrow \boldsymbol{z} +$ “\n Previous span ” **+** $\boldsymbol{d}$ **+** “ was incorrect, try again”
end for

// Pattern match span text to get character-wise positioning of the span
$c \leftarrow \texttt{FuzzyMatch}(\boldsymbol{d}, \boldsymbol{r})$ *// c is an integer-based character index of $\boldsymbol{d}$ in $\boldsymbol{r}$*

// Convert to token position where answer is reached
$i^\star \leftarrow \texttt{CharToTokenPos}(c + \texttt{len}(\boldsymbol{d}), \boldsymbol{r}, \texttt{tokenizer})$

return $i^\star$

subset consists of generated and validated Python coding examples. Our sampling procedure for this dataset was not uniform, unlike the others. Instead, problems are grouped by their `entry_point` field, and sampling is split into rounds, with each round randomly sampling one problem from that group without replacement. Finally, we randomly sample an additional 12,000 samples from the `OS-Q3-235B-4` subset of the OpenScience dataset. This subset consists of multiple-choice STEM question-answer pairs that were synthetically generated from Qwen3-235B-A22B Yang et al. (2025a).

Three CoTs are sampled per problem by the target LRM (we used Qwen and Mistral models), yielding a set of approximately 110,799 CoTs per LRM. The respective answer positions for each set of CoTs is obtained with our extraction method outlined in Sec. 3.1. However, this procedure is not perfect, as even with our retry logic, the answer extractor, identifier, and verifier LRM (Qwen3-30B-A3B-Thinking-2507) cannot always identify the earliest final answer position. Thus, all three of these steps are successful for roughly 70%–80% of CoTs. Finally, a training-ready dataset for each LRM is formed by preparing label vectors (based on the answer positions), loss masks (based on the positions of `<think>` and `</think>`), and tokenizing the CoTs.

C.4 Implementation of the Baseline Methods

Thought Calibration. Wu et al. (2025) propose training a linear probe to predict optimal stopping points during reasoning generation. The method first segments reasoning trajectories (CoTs) into individual steps, delimited by “\n\n” and containing words like “wait” or “but”. Three probe variants are introduced: *Supervised* predicts whether the LRM is correct based on current thoughts; *Consistent* predicts whether the current answer ($\hat{\boldsymbol{a}}$) matches the final answer ($\boldsymbol{a}$); and *Novel Leaf* predicts whether the current step is a leaf node but not novel.

The stopping decision is controlled by two hyperparameters: *tolerance* δ (the maximum acceptable risk of stopping incorrectly) that implies a threshold λ (the calibrated probe score cutoff that triggers

stopping), and *window size* (the number of consecutive reasoning steps averaged to smooth predictions before threshold comparison). We retrain the *Supervised* and *Consistent* probes on the S1-K dataset Muennighoff et al. (2025) for all four models in our experiments: Qwen3-8B, Qwen3-14B, Ministral-3-8B-Reasoning-2512, and Ministral-3-8B-Reasoning-2512. For inference, we use a *window size* of 10, and the rest of the hyperparameters are specified in Table 2. As shown in Table 2, the Qwen3 models require lower thresholds compared to the Ministral models, as their probe outputs yield systematically lower confidence scores. Similar model-specific calibration requirements have been observed in prior work Yang et al. (2025b).

	Supervised		**Consistent**	
Model	tolerance (δ)	threshold (λ)	tolerance (δ)	threshold (λ)
Qwen3-8B	0.25	0.6526	0.25	0.8790
Qwen3-14B	0.25	0.6377	0.25	0.8679
Ministral-3-8B-Reasoning-2512	0.10	0.8173	0.025	0.9973
Ministral-3-14B-Reasoning-2512	0.10	0.8306	0.025	0.9973

Table 2: Hyperparameters used for *Supervised* and *Consistent* linear probs for Thought Calibration.

We reported the results for the *Supervised* probe in Table 5, as it performed better across test datasets than the *Consistent* probe. For comparison, we report the results for both probes in Table 3. Note that for the *Consistent* probe with Ministral models, tolerance 0.025 represents the smallest feasible setting among values suggested in Wu et al. (2025). The smallest possible setting, being tolerance 0.01, yields threshold=1.0, resulting in no compression.

	MATH-500		**AIME 2025**		**GPQA**		**HumanEval**	
Model	Acc (↑)	CR (↓)	Acc (↑)	CR (↓)	Acc (↑)	CR (↓)	Acc (↑)	CR (↓)
Supervised								
Qwen3-8B	90.1%	93.89%	65.8%	81.54%	52.6%	78.87%	71.8%	92.92%
Qwen3-14B	89.8%	91.92%	63.3%	71.29%	54.6%	81.90%	74.8%	87.12%
Ministral-3-8B-Reasoning-2512	87.7%	87.80%	83.7%	91.16%	47.3%	71.78%	47.2%	87.16%
Ministral-3-14B-Reasoning-2512	87.3%	95.86%	59.4%	79.77%	49.5%	73.81%	27.6%	96.25%
Consistent								
Qwen3-8B	88.2%	72.09%	43.1%	54.64%	44.7%	45.17%	70.7%	73.90%
Qwen3-14B	85.9%	64.40%	41.7%	45.57%	48.3%	53.83%	70.9%	39.60%
Ministral-3-8B-Reasoning-2512	75.8%	80.53%	23.3%	60.50%	42.4%	62.69%	34.2%	75.24%
Ministral-3-14B-Reasoning-2512	68.7%	77.40%	9.4%	34.42%	40.6%	59.97%	9.6%	95.90%

Table 3: Performance comparison of *Supervised* and *Consistent* probes for Thought Calibration across models and tasks.

D Additional Experimental Results

Main Results. Table 5 and Fig. 7 show the performance of TERMINATOR and relevant baselines for both Qwen3 and Ministral-3 families with respect to the Compression Rate (lower is better) and Accuracy. All methods are evaluated using the same CoTs that are used in the vanilla baseline, except the NoThinking baseline, which is generated without CoT generation. Our results show state-of-the-art performance over early-exit methods from prior work.

Latency Analysis. We develop a vLLM-compatible implementation of TERMINATOR, benchmark the latency and throughput, and compare with running the vanilla LRM in vLLM. The results are presented in Table 4. Both methods are evaluated on the same subset of MATH-500 questions with a batch size of 1, disabled prefix caching, and on a single GH200. TERMINATOR halves the average latency over the vanilla LRM, but does incur a small overhead of 10.8% for Qwen3-8B and 7.5% for Qwen3-14B. However, as the base LRM size increases, TERMINATOR incurs a proportionally smaller overhead since its architecture (a single transformer layer and an FFN) remains fixed.

Event-Locked Averaging Token-Confidence. The results shown in Figs. 2 and 4 motivate our approach by confirming that the first arrival of $\hat{\boldsymbol{a}}$ is (1) marked by spiking behavior in the Token-Confidence, which is most easily seen in the event-locked average case, and (2) by a shift in the "thinking token" usage distribution. Focusing on the averaged plots in Fig. 2, we observe that the confidence of the LRM grows up to the point when $\hat{\boldsymbol{a}}$ is first generated, where the confidence

Table 4: **Latency Analysis.** Latency and throughput benchmarks on MATH-500 problems (batch size 1) for Qwen3-based vanilla and TERMINATOR models. TERMINATOR reduces latency costs by a factor of over $2\times$, but incurs a slight throughput overhead. Values are reported as mean $\pm$ 95% CI.

	Qwen3-8B		Qwen3-14B	
Method	**Lat. (s)**	**Thr. (tok/s)**	**Lat. (s)**	**Thr. (tok/s)**
Vanilla	32.68 ± 9.59	151.5 ± 4.4	43.38 ± 13.98	98.0 ± 2.0
Terminator	14.10 ± 6.27	135.2 ± 2.0	18.76 ± 6.52	90.6 ± 0.8

finally peaks. The confidence immediately drops after $\hat{\boldsymbol{a}}$ is generated, which makes sense given that the LRM immediately begins to doubt itself, often producing "thinking tokens" like `wait` or `but`, signaling uncertainty about the answer that was just generated. The LRM's confidence improves a little as it continues to re-think through the problem.

We liken the averaged result in Fig. 2 to the field of *event-related potential* (ERP) research. An ERP is a measurable brain response elicited by a sensory, cognitive, or motor event, captured by electroencephalogram (EEG) recordings Luck (2014). However, EEG recordings are often noisy, so ERPs are estimated using time-locked statistical estimators (e.g. averaging) across multiple EEG trials. While we do not claim that our findings will align exactly with ERP research, it is quite interesting that meaningful and observable signals can be extracted from LRMs using similar approaches, and we believe this warrants further exploration in future work.

Figs. 8 to 11 show the Token-Confidence and log-probabilities for the single-sample and event-locked averaging case, similar to what's shown in Fig. 2, separately for each data source. While the exact contours of these two signal types vary for different data sources, the same idea applies to all: the LRM's Token-Confidence has a sharp spike at the position of the first occurrence of $\hat{\boldsymbol{a}}$, followed by a sharp decrease. In all cases, the Token-Confidence then has a quick recovery before plateauing or decaying.

Token Usage Frequency Shift. Beyond providing motivation for our method, the results of Fig. 4 offer some interesting insights on the usage of "thinking tokens." These plots show the strong usage bias that can occur with respect to the first occurrence of $\hat{\boldsymbol{a}}$. For example, 63.9% and 91.5% of CoTs contain the tokens `hmm` and `okay` more often before $\hat{\boldsymbol{a}}$ than after it, respectively. Other tokens, like `another` are more frequently used after. Moreover, Figs. 12 to 16 of the appendix show that the occurrence rates can differ drastically between data sources. For example, the token `alternatively` has an above diagonal rate of 80.4% for MATH, but only 19.2% for OpenCoder-SFT.

Fig. 4 also shows the length of each CoT by its dot size; it appears that there is some correlation between the dot size and the occurrence rates. We show a plots of the before and after occurrence rates for these three tokens in Fig. 17 of the appendix. Notably, shorter CoTs do in fact correlate with higher token occurrences for these three "thinking tokens."

Fig. 12 shows an expanded view of Fig. 4, including six additional "thinking tokens." Figs. 13 to 16 reproduces this expanded plot, but with the data sources separated. Interestingly, the occurrence rates can vary substantially across different data sources. For example, for the token `alternatively`, only 19.2% of points lie above the diagonal (only 19.2% of CoTs have the token `alternatively` occurring more frequently after the answer than before), and 45.8% (nearly half!) of the points are at the origin (the token `alternatively` never occurs in 45.8% of CoTs) for OpenCoder-SFT. However, for MATH, 80.4% of points lie above the diagonal and only 4.1% lie at the origin for the same token. Other tokens, like `therefore`, are strongly biased toward occurring after the answer.

Fig. 17 shows that the average occurrence rate across CoTs from all data sources changes depending on how long the CoT is. We normalize by the number of tokens occurring before and after the first occurrence of $\hat{\boldsymbol{a}}$ for the before and after rates, respectively, so these plots suggest that the LRM uses `hmm`, `okay`, and `another` more frequently for shorter CoTs than longer ones.

TERMINATOR Recovers Early-Exit Signals. Using TERMINATOR's predicted exit positions, the same event-locked averaging (Fig. 2) and "thinking token" frequency (Fig. 4) phenomena are recovered. Fig. 5 mirrors Fig. 2, but uses all samples from the test datasets instead of 3,200 randomly selected samples from our training dataset (gathering TERMINATOR's predictions on the

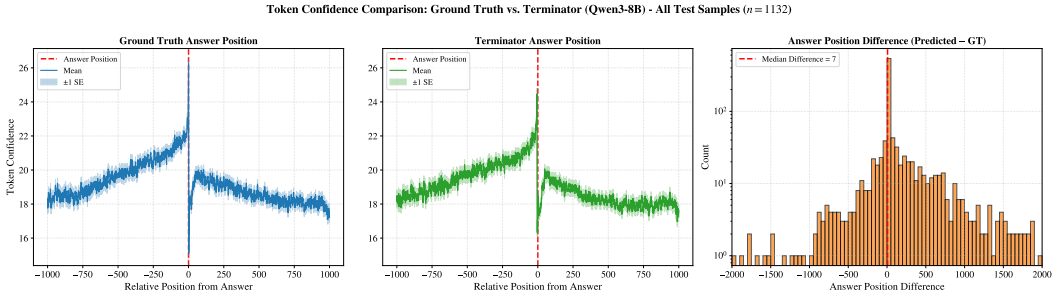

Figure 5: **TERMINATOR Recovers Event-Locked Average Spiking.** The exit positions predicted by TERMINATOR **(center)** recover the same spiking behavior in the event-locked averaged Token-Confidence as the ground-truth answer positions **(left)**. The histogram of differences between the exit positions **(right)** shows that TERMINATOR's predicted exit positions are close to the ground-truth. Note that the y-axis on the histogram is log-scaled.

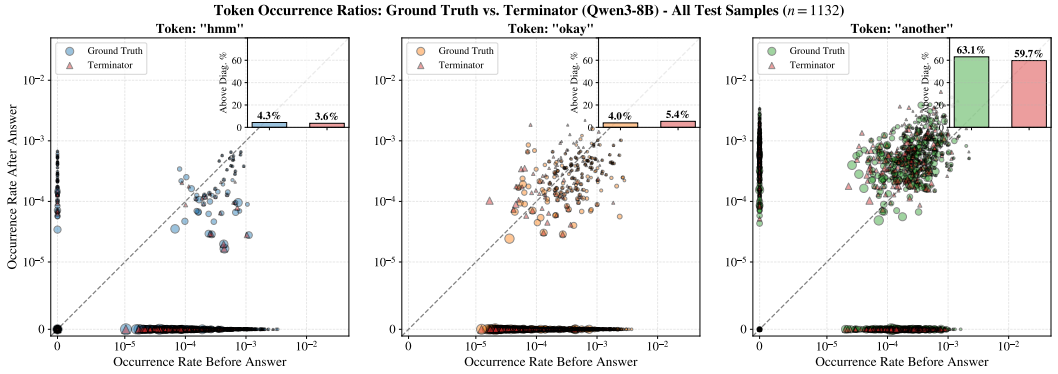

Figure 6: **TERMINATOR Token Usage Biases.** The exit positions predicted by TERMINATOR recover the same biases in the "thinking token" occurrence rates as the ground-truth answer positions. The inset axes on each panel show the percentage of dots that lie above the diagonal when the ground-truth and TERMINATOR answer positions are used.

3,200 samples requires additional compute; the test dataset predictions are readily available). The leftmost panel shows the event-locked average Token-Confidence based on the ground-truth answer positions, while the center panel shows the event-locked average Token-Confidence using the answer positions predicted by TERMINATOR. The rightmost panel shows the histogram of differences between TERMINATOR's predicted exit positions and the ground-truth positions; these differences are concentrated around zero, with a median difference of 7, which helps explain why TERMINATOR recovers most of the same signal.

Similarly, Fig. 6 is analogous to Fig. 4, but superimposes the resulting scatter plots when the ground-truth answer positions are used and when TERMINATOR's answer positions are used. Inset axes show that the above-diagonal percentages are nearly the same between these two, demonstrating that the "thinking token" occurrence rates before and after TERMINATOR's predicted answer positions reveal before/after token usage biases.

The results of Figs. 5 and 6 together show that training TERMINATOR on the LRM's hidden states is enough for TERMINATOR to independently recover the same early-exit signals discovered earlier.

Event-Related Potentials for LRMs. We liken the averaged result in Fig. 2 to the field of *event-related potential* (ERP) research. An ERP is a measurable brain response elicited by a sensory, cognitive, or motor event, captured by electroencephalogram (EEG) recordings Luck (2014). However, EEG recordings are often noisy, so ERPs are estimated using time-locked statistical estimators (e.g. averaging) across multiple EEG trials. While we do not claim that our findings will align exactly with ERP research, it is quite interesting that meaningful and observable signals can be extracted from LRMs using similar approaches, and we believe this warrants further exploration in future work.

Fig. 4 also shows the length of each CoT by its dot size; it appears that there is some correlation between the dot size and the occurrence rates. We show plots of the before and after occurrence

rates for these three tokens in Fig. 17. Notably, shorter CoTs do in fact correlate with higher token occurrences for these three "thinking tokens."

TERMINATOR Predictions During Reasoning. Fig. 18 shows the event-locked average of the predicted probabilities from TERMINATOR for each data source, and Fig. 19, Fig. 20, Fig. 21, and Fig. 22 show the predicted probabilities from four randomly chosen examples for AIME25, MATHH-500, HumanEval, and GPQA, respectively. The event-locked average and the individual examples from MATH-500 and HumanEval show sharp transitions in predicted confidence at the exiting threshold, with good separation (dotted gray line). However, AIME25 and GPQA examples do not show such a sharp transition, suggesting that it is challenging for TERMINATOR to identify a good exit position for very hard tasks.

TERMINATOR Thresholding. Fig. 23 shows how the compression rate and the accuracy change as a result of varying the threshold used by TERMINATOR to predict 0 or 1. For example, a threshold $\tau = 0.7$ requires TERMINATOR to have a predictive confidence of at least 0.7 to predict 1. Interestingly, varying the threshold has a greater impact on compression rate than on accuracy; setting $\tau = 0.1$ yields 8% and 17% better compression rates for MATH-500 and HumanEval, respectively, compared to $\tau = 0.9$, with almost no change in accuracy! However, more challenging datasets, such as AIME25 and GPQA, exhibit notable changes in accuracy as the threshold varies, though the percent drop in accuracy from $\tau = 0.9$ to $\tau = 0.1$ is less than the improvement in compression rate. For example, from $\tau = 0.9$ to $\tau = 0.1$ yields a 6% drop in accuracy but a 27% improvement in the compression rate for GPQA. Since the accuracy drop is so small, this suggests that TERMINATOR is reasonably confident at determining an appropriate exit position on the harder datasets, and that additional reasoning time can only improve accuracy a little.

Answer Prediction Histograms. Fig. 24 gives an overview of the position of the first occurrence of $\hat{\boldsymbol{a}}$ in the CoTs for each data source we used for training. Fig. 25 shows the compression rate statistics of TERMINATOR for each test dataset.

Case Study of TERMINATOR. By manual inspection, we have seen that for easier problems in the MATH-500 dataset, there is a clearer transition to overthinking, which is well detected by TERMINATOR. However, for harder AIME25 problems, the transition is less obvious. Figs. 26 to 29 show the generated CoTs of the Qwen3-14B model for a single sample from MATH-500 and AIME25 with the predicted probabilities from TERMINATOR.

E RELATED WORK

Prompt Compression. This line of work is concerned with compressing the input prompt (or context) before passing it to an LLM. Some methods use *soft-prompts* Mu et al. (2023); Chevalier et al. (2023); Ge et al. (2024); Qin et al. (2024) to compress tokenized inputs into a sequence of embeddings. These embeddings serve as the LLM's input, allowing richer expressivity, but they are not amenable to black-box LLMs and are difficult to analyze theoretically. Other methods use *hard-prompts* Jung & Kim (2024); Jiang et al. (2024); Pan et al. (2024); Nagle et al. (2024), keeping the final compressed input prompt fixed to the same token vocabulary as the LLM.

Efficient Reasoning. Analogously to soft-prompt compression, *latent* or *continuous* reasoning is a technique where reasoning unfolds across latent output embeddings (or hidden states) rather than discrete tokens. Methods like Coconut Hao et al. (2025), CCoT Cheng & Durme (2024), and Soft Thinking Zhang et al. (2025d) feed the LLM's output embeddings back into the input of the LLM during the reasoning stage, which significantly decreases the number of passes through the LLM before arriving at the final answer. LightThinker Zhang et al. (2025c) uses an idea similar to AutoCompressor Chevalier et al. (2023), where each reasoning step is generated as discrete tokens first, compressed, and then the compressed summary of the reasoning thus far is fed back into the LLM to generate the next step. Other methods, like TokenSkip and C3oT Xia et al. (2025); Kang et al. (2025), are closer to hard-prompt compression, where a prompt compressor (or a summarization) model first compresses the CoTs into much shorter versions, and the LLM is retrained on these shorter CoTs. CoT-Valve Ma et al. (2025b) extends this idea by introducing a parameter that adds explicit control over the reasoning length after fine-tuning. CALM Schuster et al. (2022) also keeps its outputs in the token space, but saves compute by making layer-wise token-level early-exit decisions.

Table 5: **Performance of TERMINATOR and Baselines.** ↑ Indicates that higher values are better, while ↓ indicates that lower values are better. CR is the compression rate, reported here as the mean per-sample compression rate. **Bold** and Underlined values highlight the best and second-best performing early exit methods, respectively. TERMINATOR demonstrates superior accuracy-efficiency trade-offs (best or second-best performance across 31 out of 32 metrics).

Method	Math				Coding		Science		Overall	
	MATH-500		AIME25		HumanEval		GPQA			
	Acc↑	CR↓	Acc↑	CR↓	Acc↑	CR↓	Acc↑	CR↓	Acc↑	CR↓
Qwen3-8B										
Vanilla	91.1%	100%	74.4%	100%	86.4%	100%	57.6%	100%	77.4%	100%
NoThinking	80.7%	16.1%	22.0%	18.6%	78.5%	11.8%	30.7%	15.8%	53.0%	15.6%
DEER	79.9%	52.0%	21.4%	**67.8%**	77.4%	83.6%	50.5%	99.6%	57.3%	75.8%
Thought-Calib	90.1%	93.9%	65.8%	81.5%	71.8%	92.9%	**52.6%**	**78.9%**	70.1%	86.8%
TERMINATOR	**90.7%**	**45.1%**	**69.4%**	70.7%	**82.9%**	**69.9%**	52.1%	85.7%	**72.6%**	**67.8%**
Qwen3-14B										
Vanilla	92.0%	100%	79.9%	100%	76.0%	100%	59.9%	100%	77.0%	100%
NoThinking	84.1%	17.5%	26.3%	19.9%	78.3%	12.2%	32.1%	18.8%	55.2%	17.1%
DEER	80.9%	56.2%	27.6%	**71.0%**	80.7%	87.3%	49.0%	97.4%	59.6%	78.0%
Thought-Calib	89.8%	92.0%	63.3%	71.3%	74.8%	87.1%	**54.6%**	**81.9%**	70.6%	83.1%
TERMINATOR	**90.7%**	**46.8%**	**74.2%**	**71.0%**	**83.3%**	**70.9%**	53.9%	87.1%	**78.0%**	**65.0%**
Ministral-3-8B-Reasoning-2512										
Vanilla	93.5%	100%	92.6%	100%	92.1%	100%	62.3%	100%	85.1%	100%
NoThinking	83.2%	28.1%	43.6%	36.5%	70.5%	16.4%	39.7%	16.0%	59.3%	14.6%
DEER	71.0%	60.3%	67.1%	77.0%	75.8%	84.0%	**61.1%**	94.1%	68.8%	78.9%
Thought-Calib	87.7%	87.8%	**83.7%**	91.2%	47.2%	87.2%	47.3%	**71.8%**	66.5%	84.5%
TERMINATOR	**89.1%**	**47.8%**	57.4%	**67.1%**	**89.0%**	**66.6%**	58.2%	77.4%	**73.4%**	**64.7%**
Ministral-3-14B-Reasoning-2512										
Vanilla	93.0%	100%	88.1%	100%	89.6%	100%	62.7%	100%	83.4%	100%
NoThinking	79.1%	11.7%	20.5%	13.8%	80.0%	14.1%	41.0%	6.6%	55.2%	13.7%
DEER	69.8%	74.6%	55.9%	84.9%	19.3%	**46.5%**	58.5%	95.0%	50.9%	75.3%
Thought-Calib	87.3%	95.9%	59.4%	79.8%	27.6%	96.3%	49.5%	**73.8%**	56.0%	86.5%
TERMINATOR	**90.2%**	**43.9%**	**65.8%**	**68.7%**	**89.6%**	71.0%	**60.6%**	76.5%	**76.5%**	**65.0%**

Early-Exit Reasoning. These methods seek to make reasoning more efficient by terminating the CoT early. All existing methods use a consistency-based approach, injecting the `</think>` token at various points to force the model to generate an answer or a useful signal. Some methods, like EAT Wang et al. (2025b), DEER Yang et al. (2025b), ES-CoT Mao et al. (2025), and Dynasor Fu et al. (2025a) are training-free; they track signals throughout the reasoning process and exit when a threshold is crossed. Other methods, like SpecExit Yang et al. (2025c), Learn To Stop Liu & Wang (2025), Thought Calibration Wu et al. (2025), and FlashThink Jiang et al. (2025) rely on training a separate probe classifier by using consistency as the main approach for gathering their training signals. By contrast, our work constructs a training signal to predict the immediate arrival of $\hat{\boldsymbol{a}}$, thereby training on hindsight-optimal length CoTs.

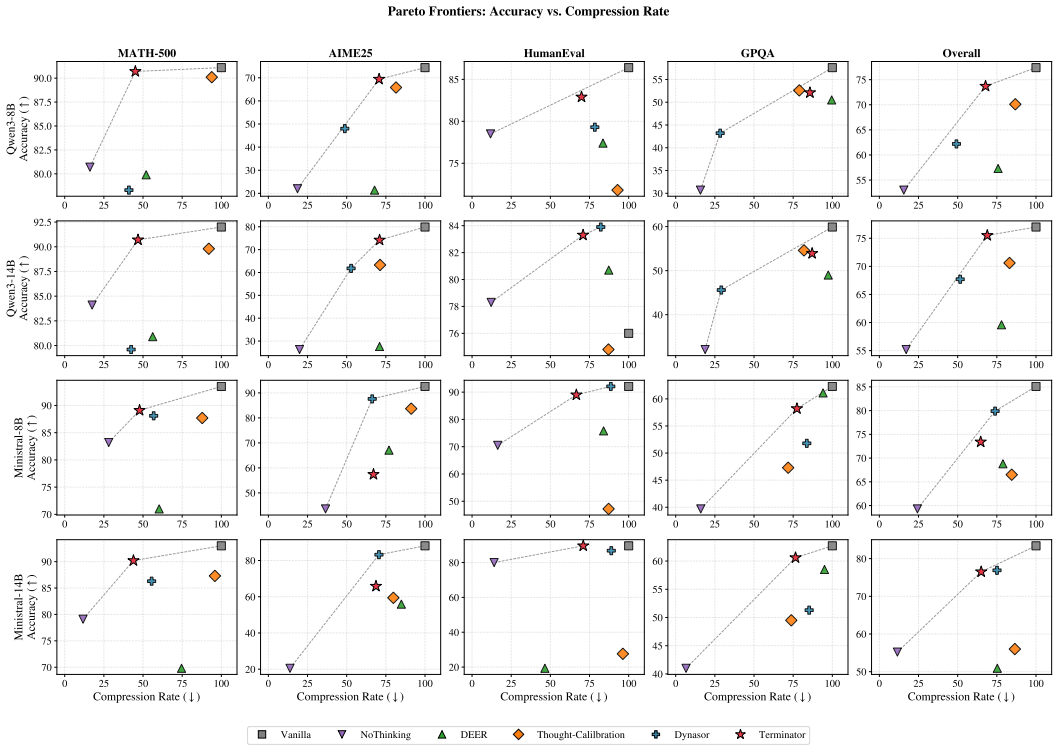

Figure 7: **Pareto Frontier.** TERMINATOR consistently achieves strong Pareto efficiency, reflected by the best accuracy–efficiency tradeoff across models and tasks. Here we plot the Pareto frontiers of accuracy versus compression rate across four reasoning models (Qwen3-8B, Qwen3-14B, Ministral-8B, Ministral-14B) and four benchmarks (MATH-500, AIME25, HumanEval, GPQA). Each point represents a method's accuracy and compression rate, with lower compression rates indicating greater token savings. The dashed line traces the Pareto frontier connecting non-dominated solutions. The data used to generate this figure is the same as the data from Table 5.

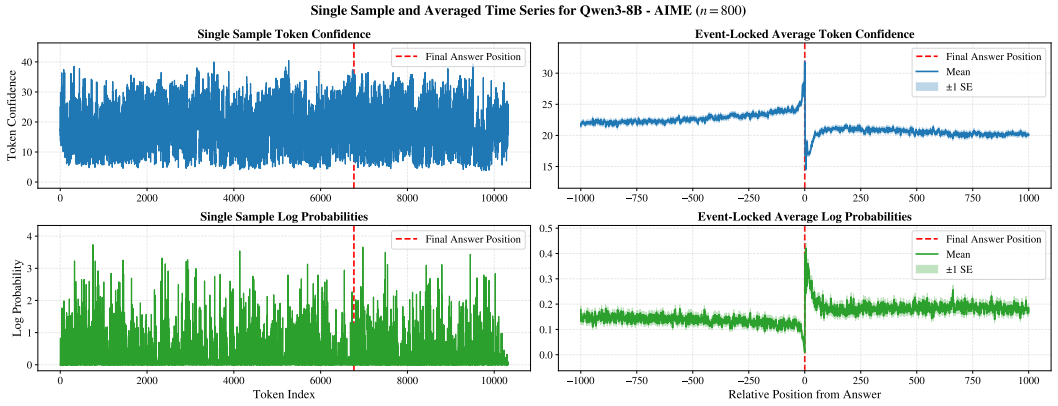

Figure 8: **Event-Locked Averaging of Token-Confidence for AIME (1983–2024).** A reproduction of Fig. 2, but only using CoTs from 800 randomly selected AIME (1983–2024) problems.

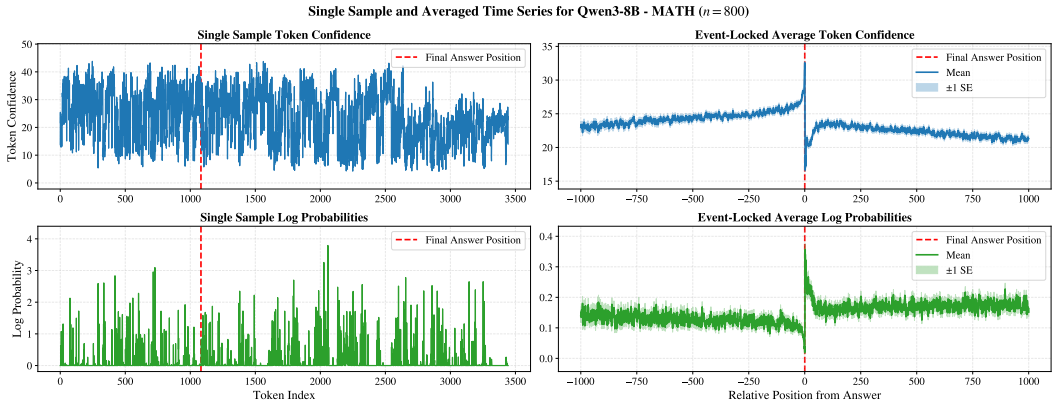

Figure 9: **Event-Locked Averaging of Token-Confidence for MATH.** A reproduction of Fig. 2, but only using CoTs from 800 randomly selected MATH problems.

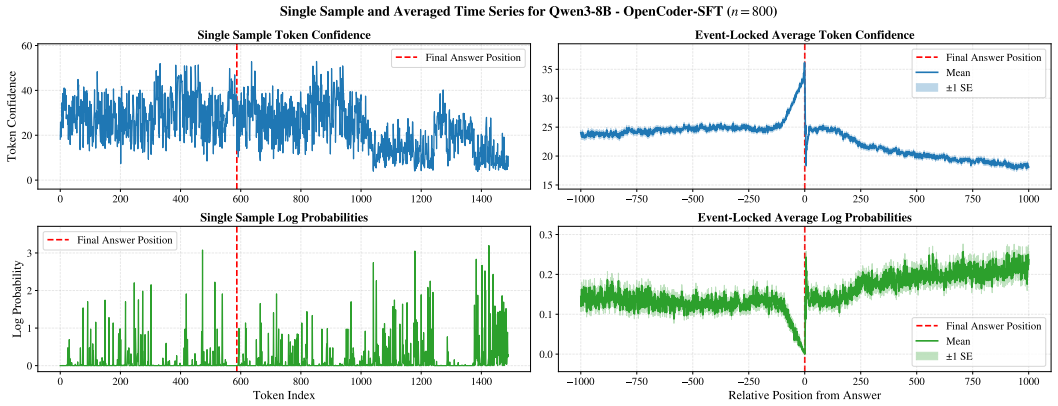

Figure 10: **Event-Locked Averaging of Token-Confidence for OpenCoder-SFT.** A reproduction of Fig. 2, but only using CoTs from 800 randomly selected OpenCoder-SFT problems.

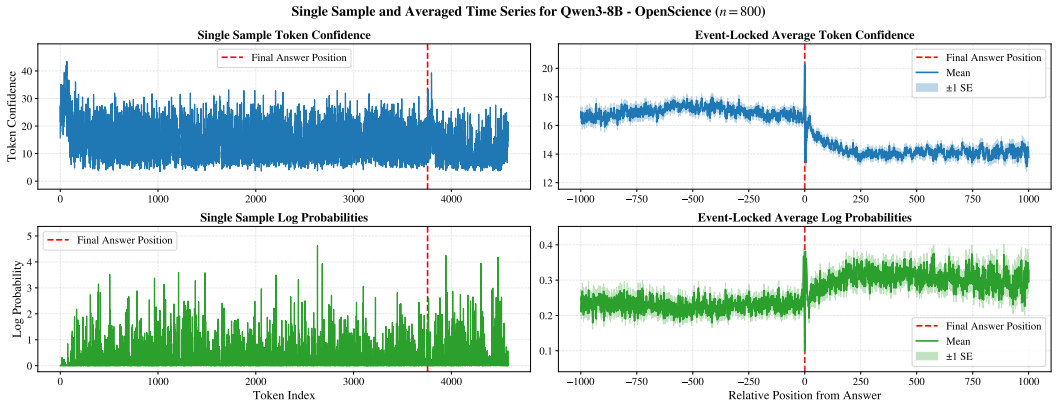

Figure 11: **Event-Locked Averaging of Token-Confidence for OpenScience.** A reproduction of Fig. 2, but only using CoTs from 800 randomly selected OpenScience problems.

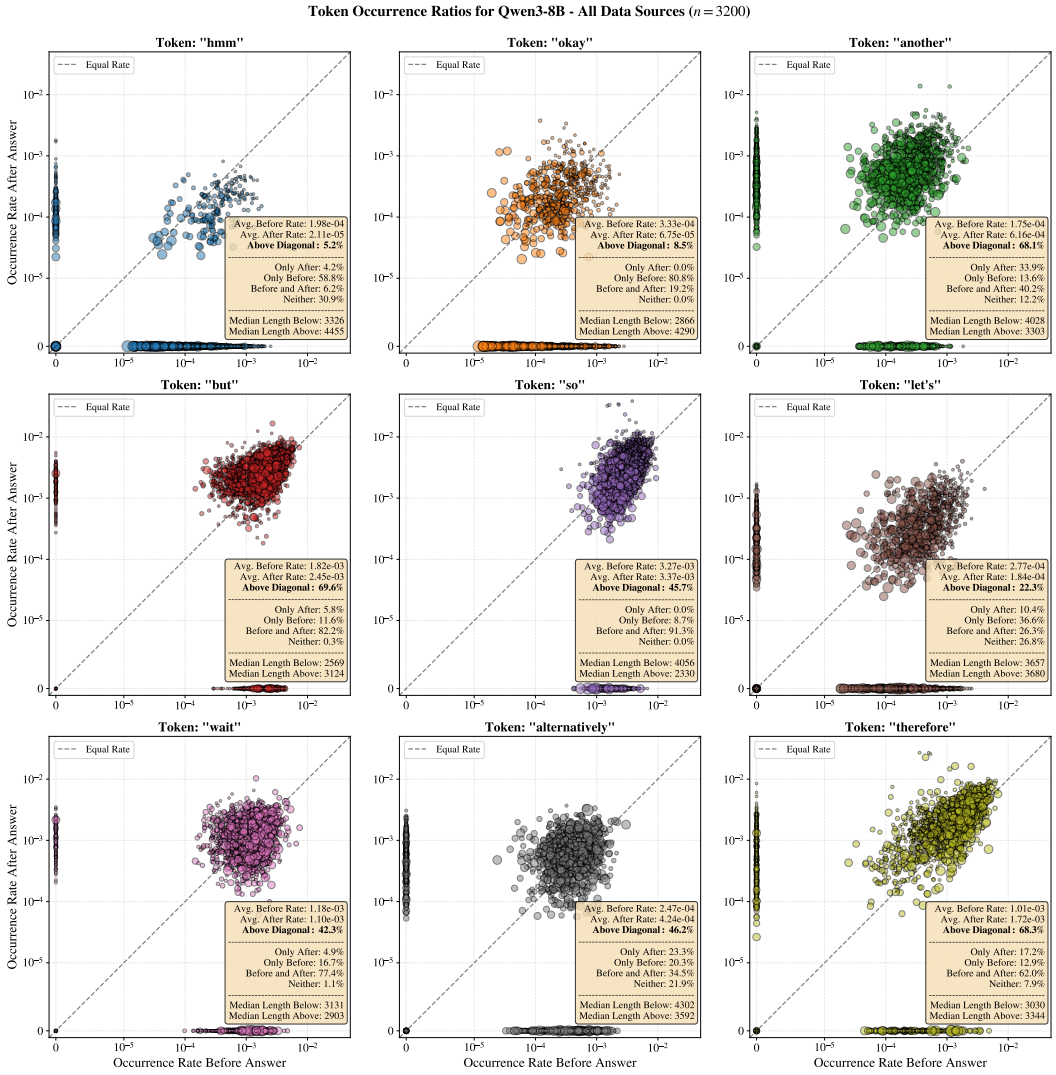

Figure 12: **Token Usage Frequency Shift.** An extension of the results shown in Fig. 4, highlighting additional "thinking tokens." While most "thinking tokens" shown here have some bias, often occurring before and after the first occurrence of the final answer as indicated by the **Above Diagonal** statistic, some tokens, like "so," are close to an equal rate on average.

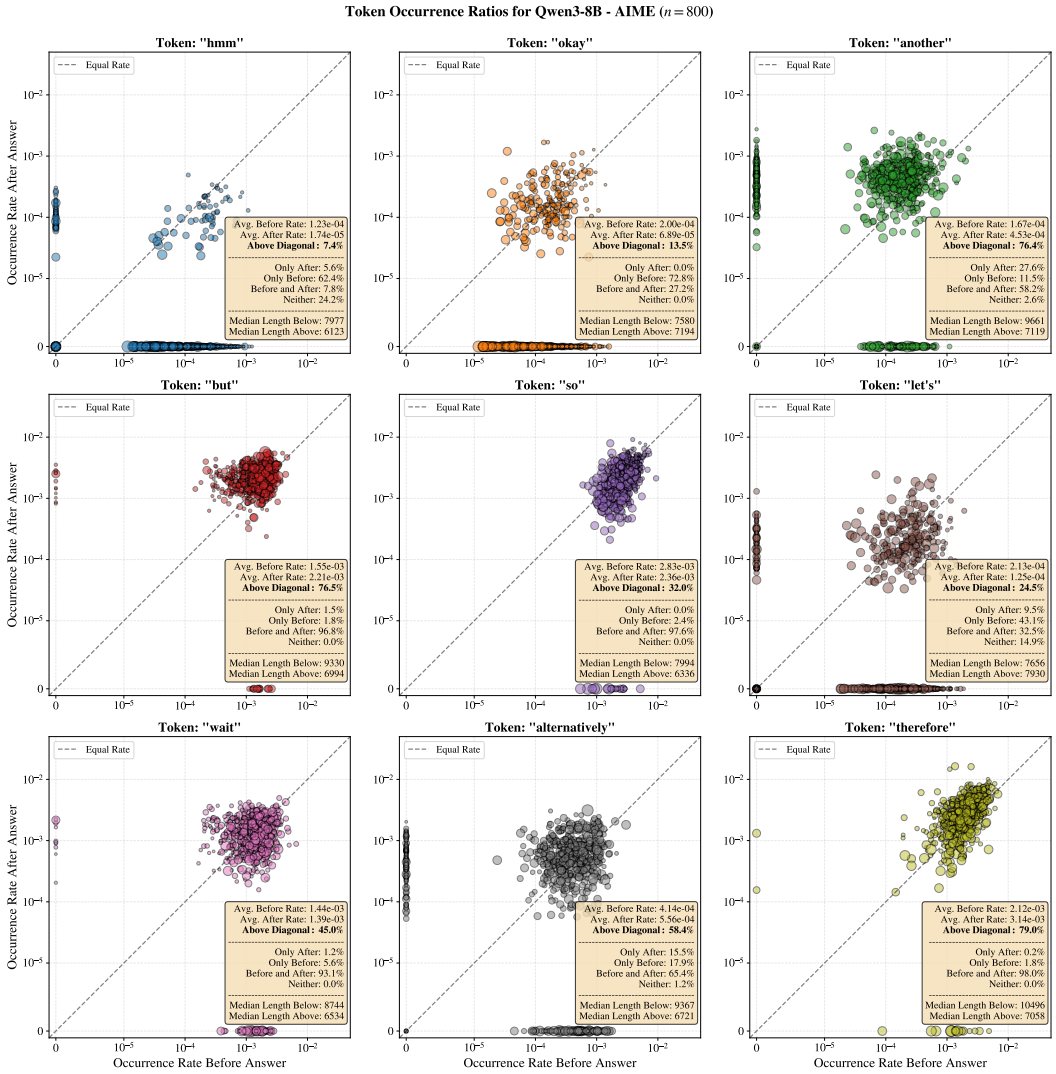

Figure 13: **Token Usage Frequency Shift for AIME (1983–2024.** A reproduction of Fig. 12, but only using CoTs from 800 randomly selected AIME (1983–2024) problems.

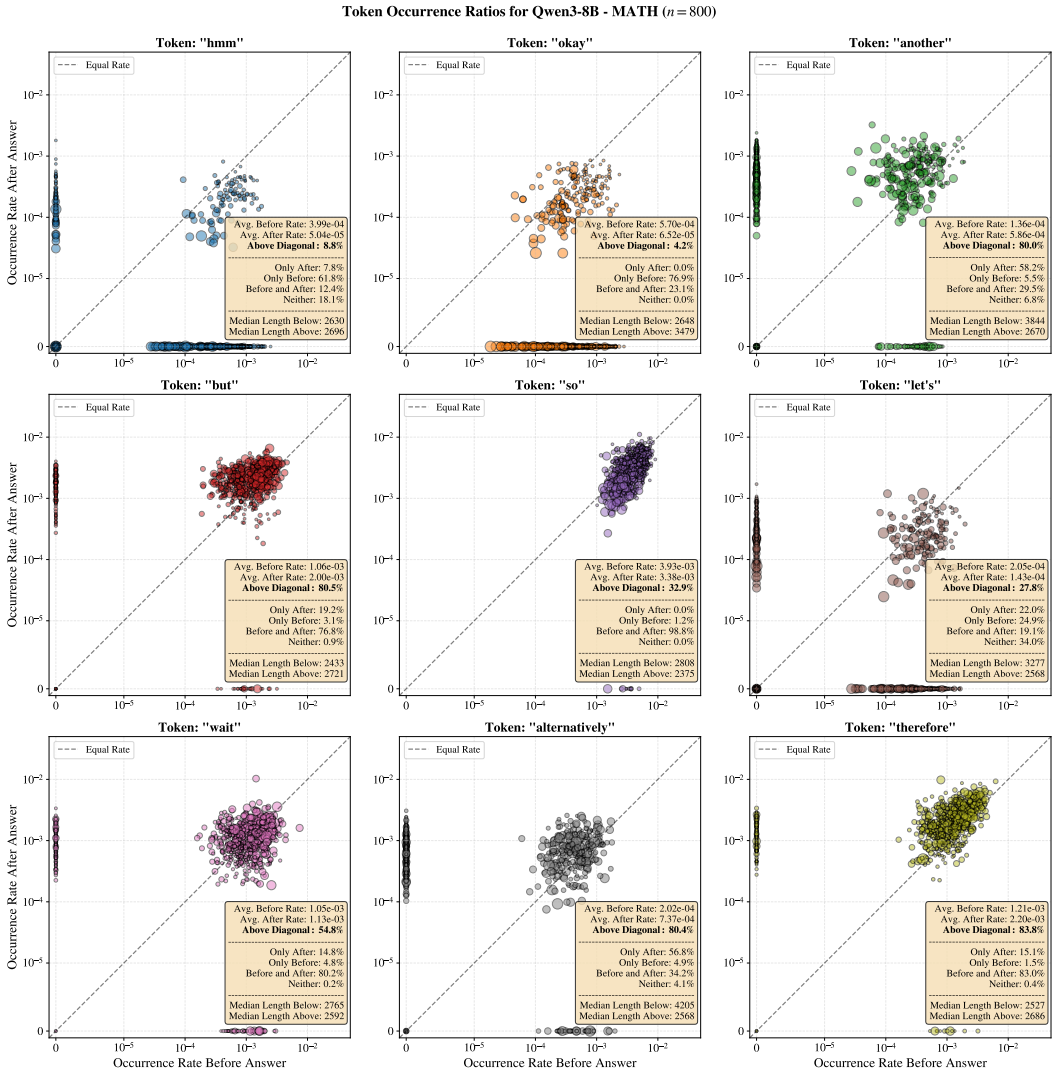

Figure 14: **Token Usage Frequency Shift for MATH.** A reproduction of Fig. 12, but only using CoTs from 800 randomly selected MATH problems.

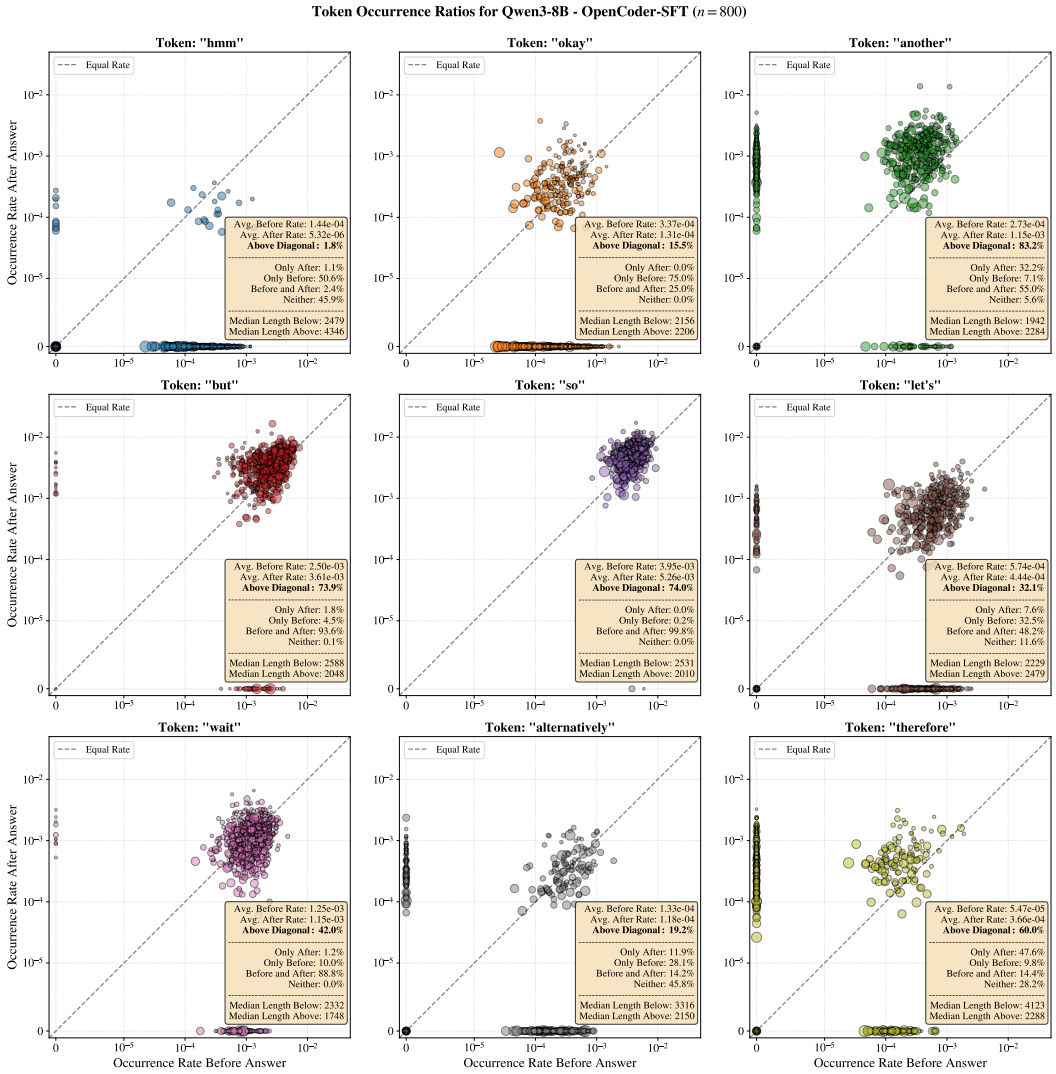

Figure 15: **Token Usage Frequency Shift for OpenCoder-SFT.** A reproduction of Fig. 12, but only using CoTs from 800 randomly selected OpenCoder-SFT problems.

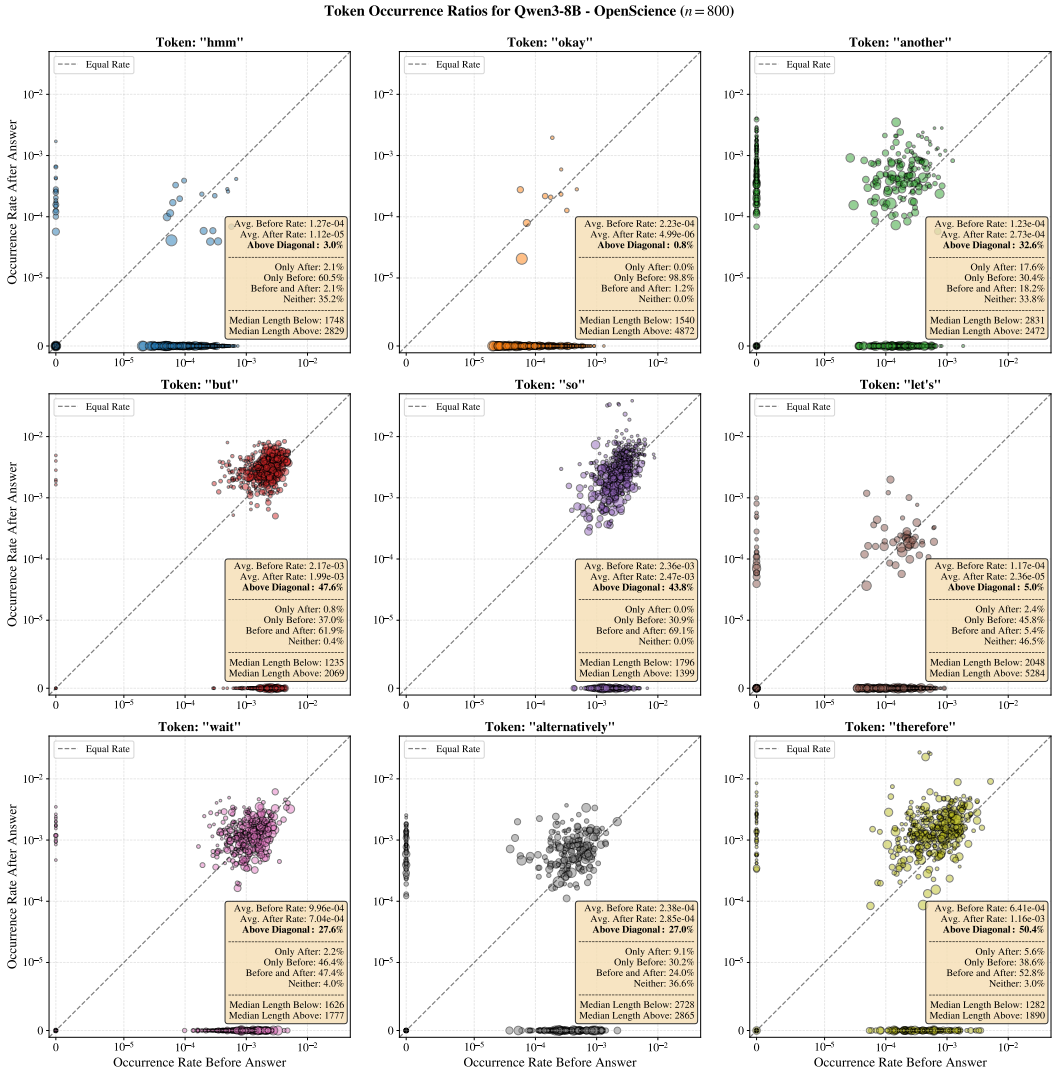

Figure 16: **Token Usage Frequency Shift for OpenScience.** A reproduction of Fig. 12, but only using CoTs from 800 randomly selected OpenScience problems.

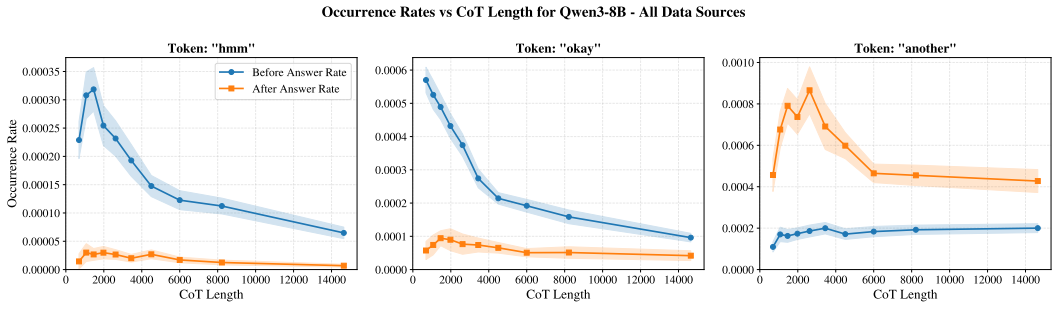

Figure 17: **Token Occurrence Rate vs CoT Length.** For some tokens, such as these three "thinking tokens," occurrence rates decrease rapidly as CoT length increases. Interestingly, the side of the answer (either "before" or "after") with the highest rate is the one that decays the most (the "before" rate for `hmm` and `okay` and the "after" rate for `another`), while the side with the lower rate sees only a slight decrease or increase. For each plot, the lengths are placed into ten bins with percentile-based bin edges. In other words, each bin contains approximately 10% of the samples. Shaded regions indicate the 95% confidence interval.

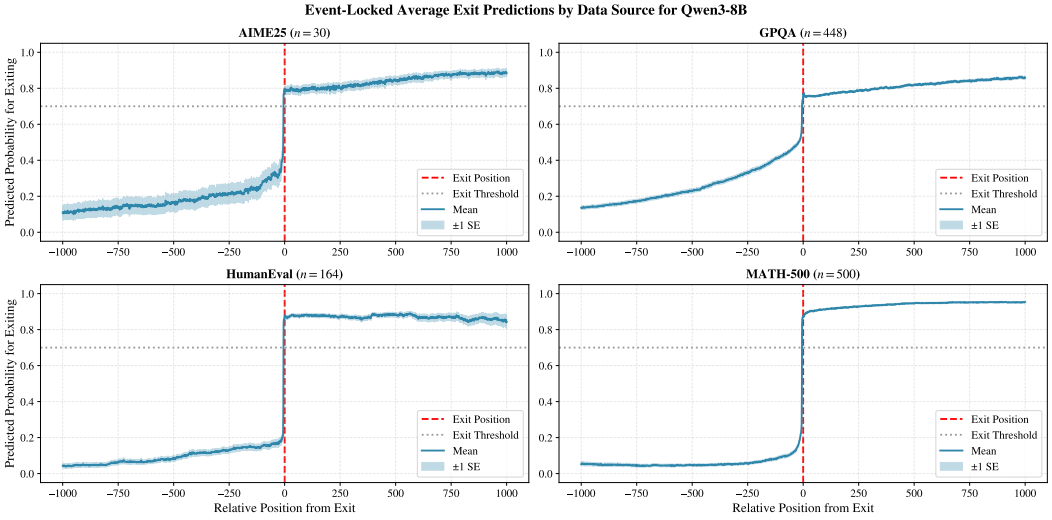

Figure 18: **Predicted Probabilities Event-Locked Averaging.** The dashed vertical line shows where TERMINATOR terminates the CoT with a sliding window of 10 and an exit threshold of 0.7, as indicated by the horizontal dotted line. We show the average predicted probability stream across all test problems from MATH-500, AIME25, HumanEval, and GPQA. Figs. 19 to 22 show predictions streams from individual, randomly drawn samples from each data source.

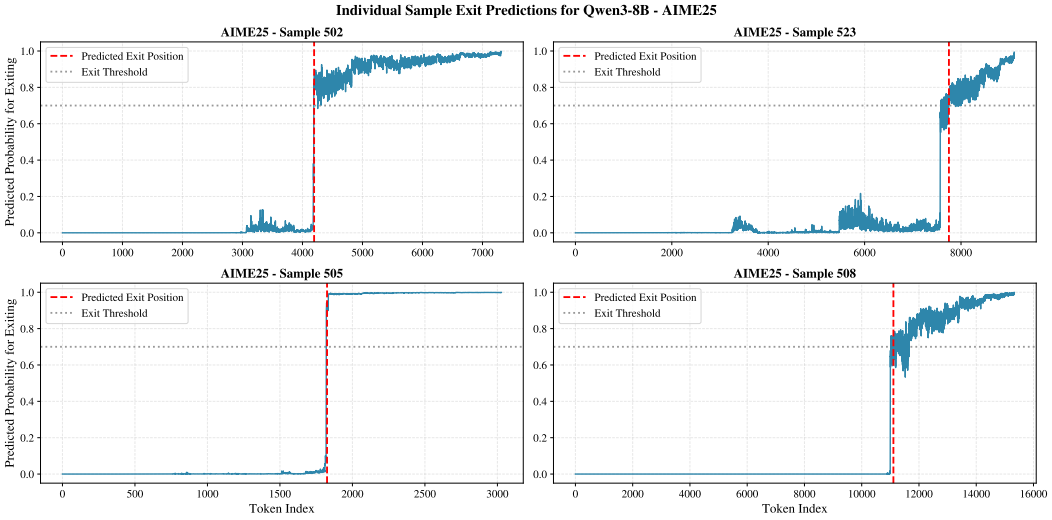

Figure 19: **Predicted Probabilities for AIME25.** TERMINATOR's predicted probability stream for early-exiting on four randomly chosen samples from AIME25.

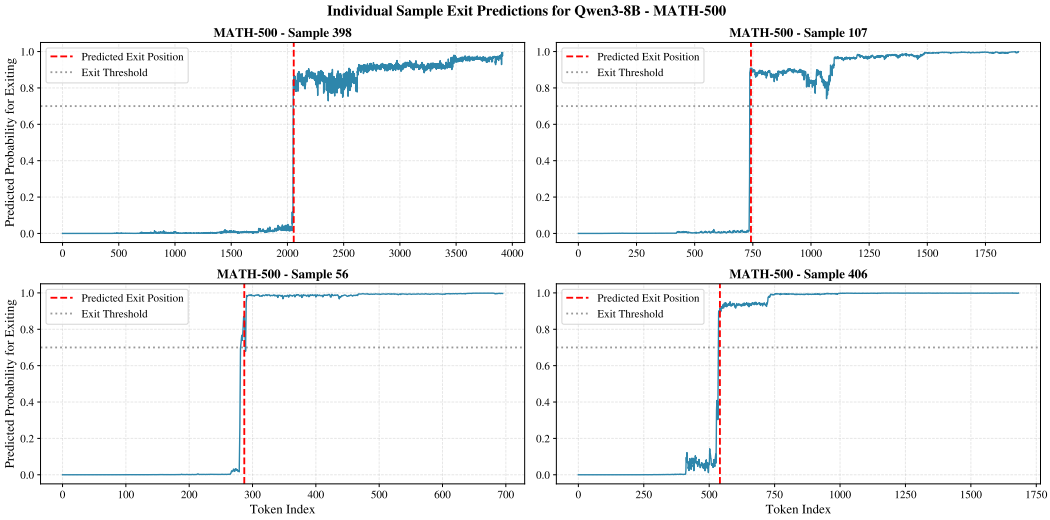

Figure 20: **Predicted Probabilities for MATH-500.** TERMINATOR's predicted probability stream for early-exiting on four randomly chosen samples from MATH-500.

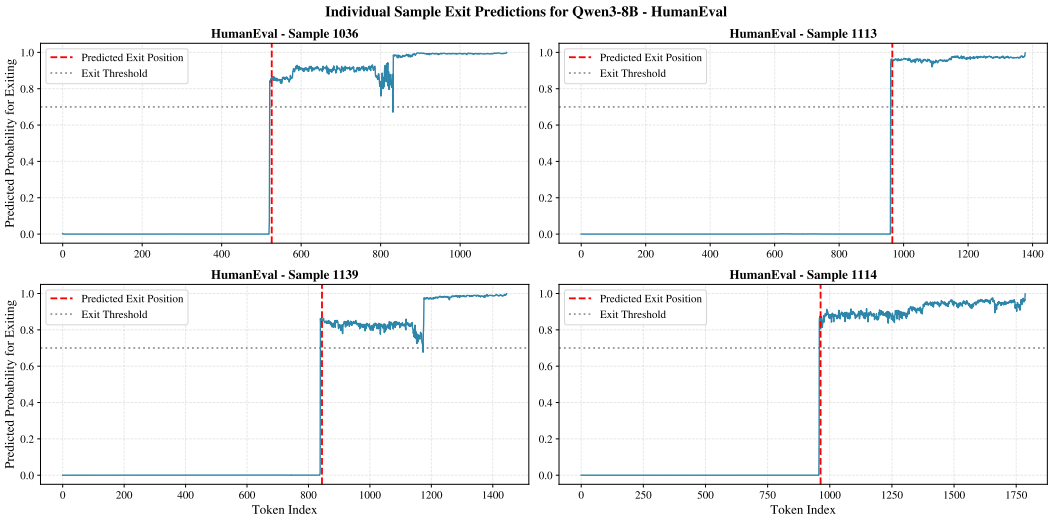

Figure 21: **Predicted Probabilities for HumanEval.** TERMINATOR's predicted probability stream for early-exiting on four randomly chosen samples from HumanEval.

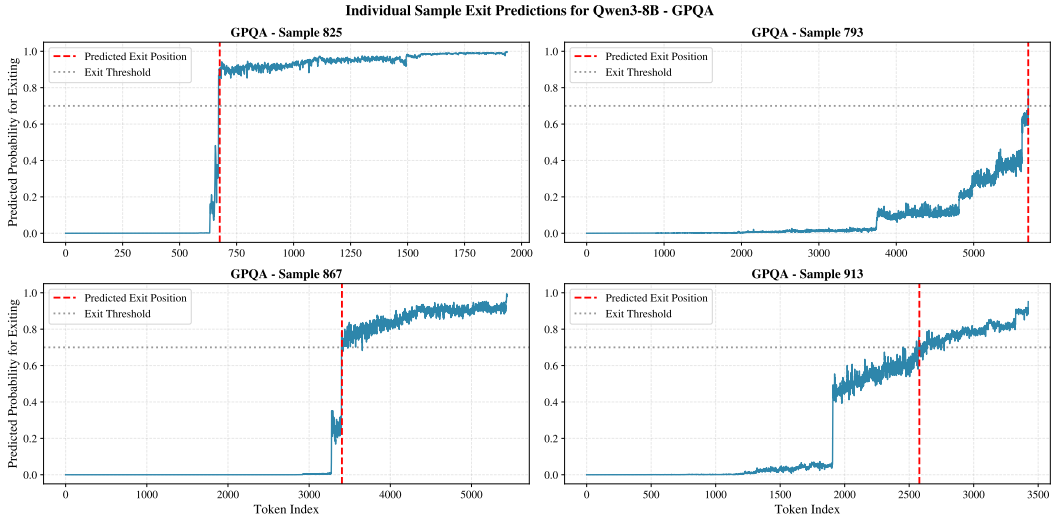

Figure 22: **Predicted Probabilities for GPQA.** TERMINATOR's predicted probability stream for early-exiting on four randomly chosen samples from GPQA.

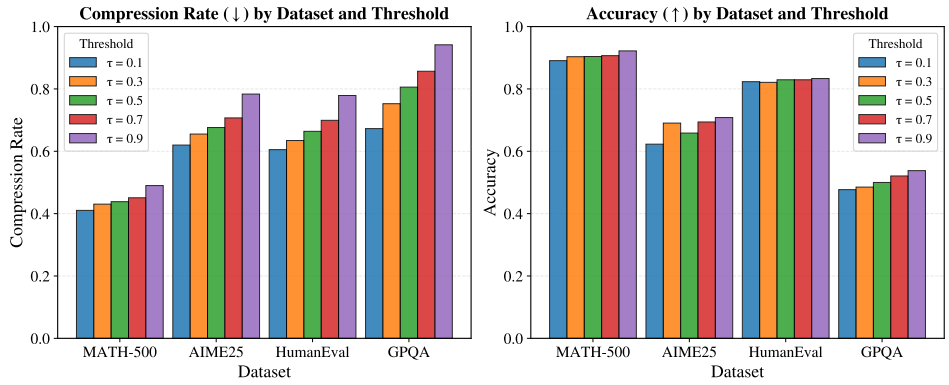

Figure 23: **Changing the Predictive Threshold.** The effect of changing the threshold of predicting 0 or 1 is altered from 0.1 to 0.9. Notably, the compression rate exhibits the largest change from $\tau = 0.1$ to $\tau = 0.9$ for all datasets, compared to the change in accuracy. In particular, some datasets, like MATH-500 and HumanEval, exhibit very little performance drop.

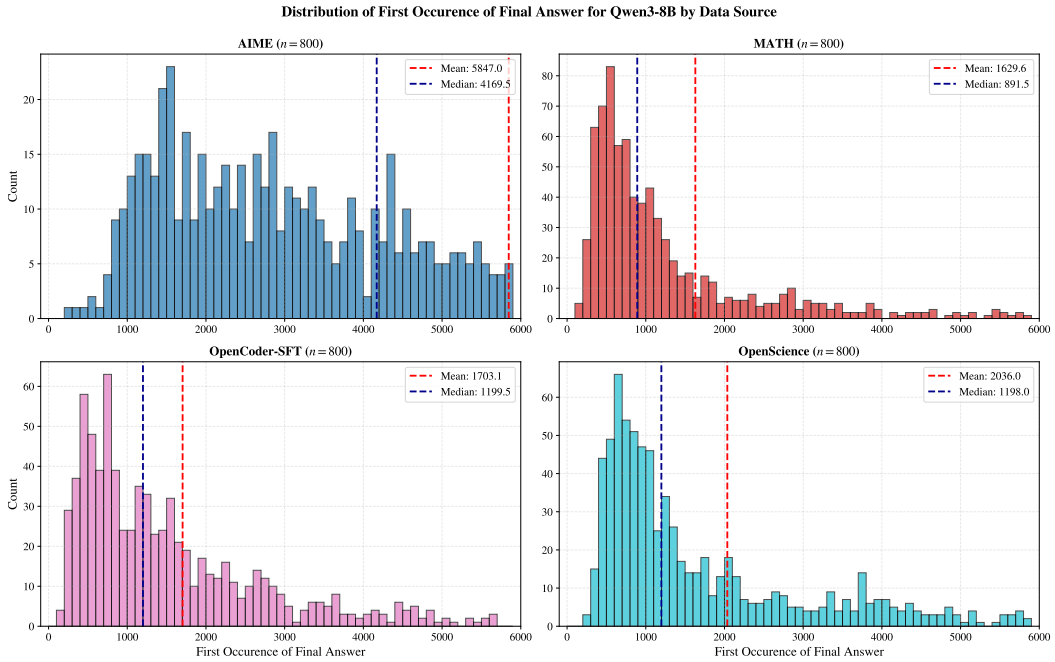

Figure 24: **First Answer Occurrence Histogram.** A histogram of the first occurrence of the final answer for each data source used in our training dataset is shown.

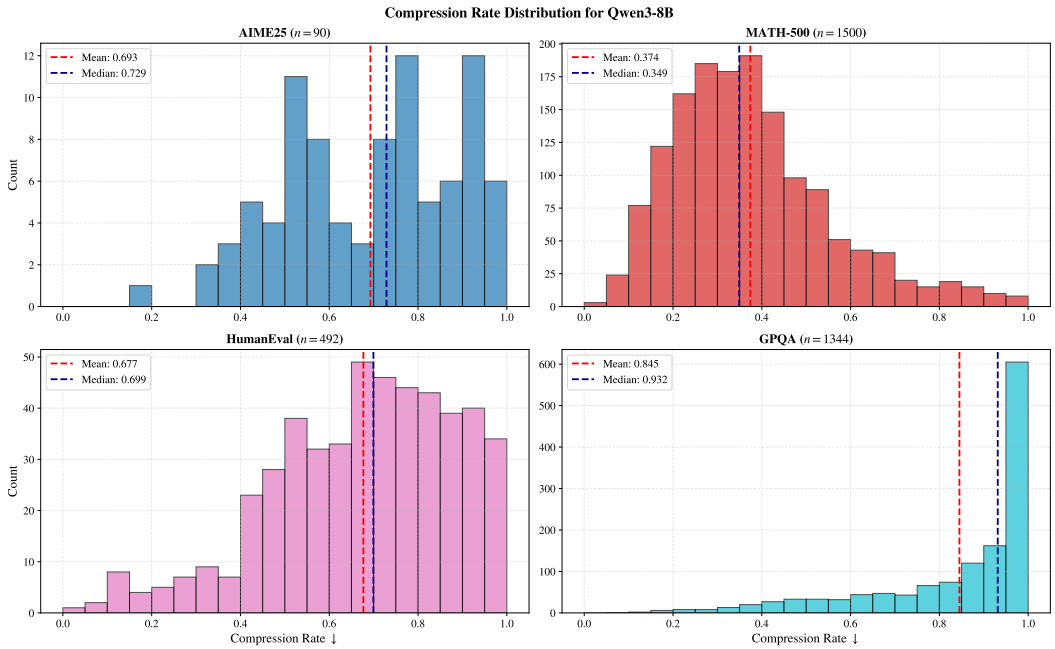

Figure 25: **Compression Ratio Histograms for TERMINATOR.** Each histogram shows the frequency of an achieved compression rate when early-exiting with TERMINATOR. ↓ indicates that a lower compression rate is better, as it results in more tokens saved with our method. Three CoTs are sampled per data source.

Figure 26: **Predicted Probabilities for MATH-500.** TERMINATOR's predicted probabilities for early-exit on a randomly chosen sample from MATH-500. The beginning and the end are truncated for better visibility.

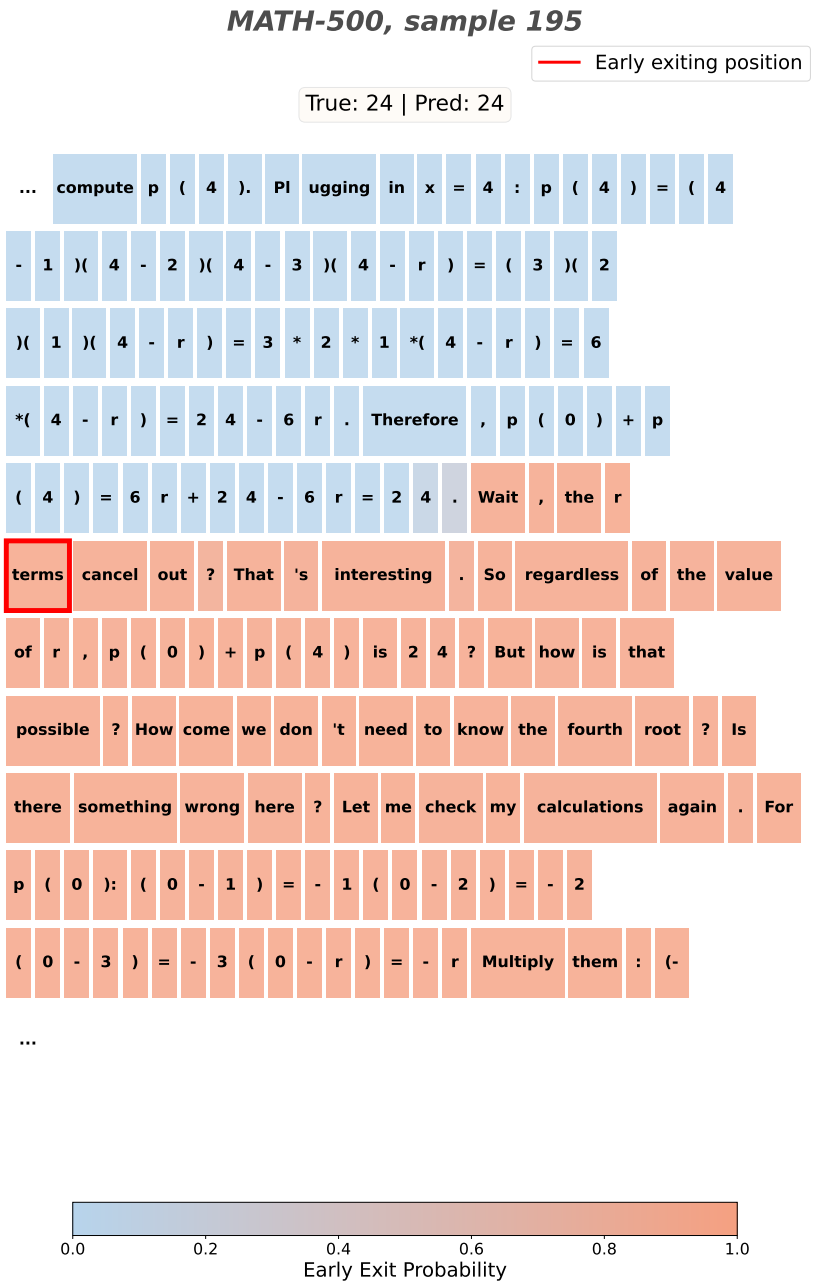

Figure 27: **Predicted Probabilities for MATH-500.** TERMINATOR's predicted probabilities for early-exit on a randomly chosen sample from MATH-500. The beginning and the end are truncated for better visibility.

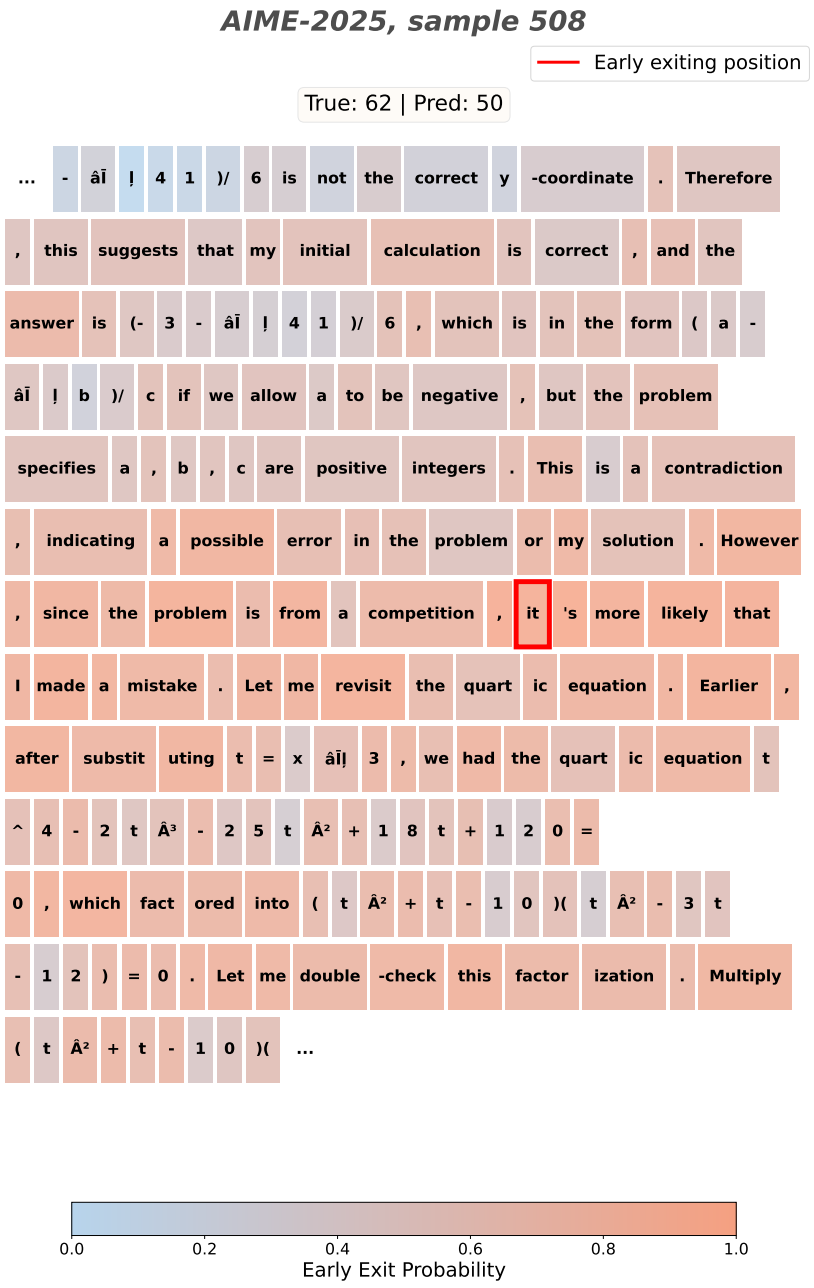

Figure 28: **Predicted Probabilities for AIME25.** TERMINATOR's predicted probabilities for early-exit on a randomly chosen sample from AIME-2025. The beginning and the end are truncated for better visibility.

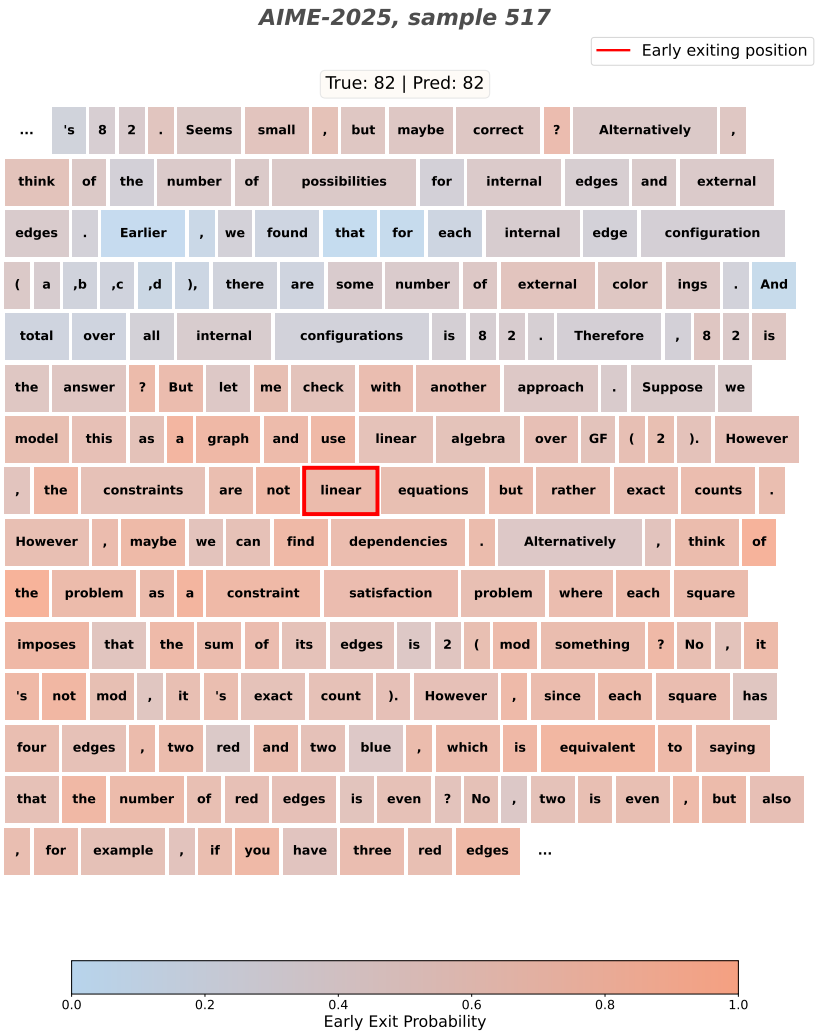

Figure 29: **Predicted Probabilities for AIME25.** TERMINATOR's predicted probabilities for early-exit on a randomly chosen sample from AIME-2025. The beginning and the end are truncated for better visibility.