

# Answer Convergence as a Signal for Early Stopping in Reasoning

Anonymous ACL submission

## Abstract

Chain-of-thought (CoT) prompting enhances reasoning in large language models (LLMs) but often leads to verbose and redundant outputs, thus increasing inference cost. We hypothesize that many reasoning steps are unnecessary for producing correct answers. To investigate this, we start with a systematic study to investigate what is the minimum reasoning required for a model to reach a stable decision. Based on the insights, we propose three inference-time strategies to improve efficiency: (1) early stopping via answer consistency, (2) boosting the probability of generating end-of-reasoning signals, and (3) a supervised method that learns when to stop based on internal activations. Experiments across five benchmarks and five open-weights LLMs show that our methods largely reduce token usage with little or no accuracy drop. In particular, on NaturalQuestions, Answer Consistency reduces tokens by over 40% while further improving accuracy. Our work underscores the importance of cost-effective reasoning methods that operate at inference time, offering practical benefits for real-world applications.<sup>1</sup>

## 1 Introduction

Large language models (LLMs) exhibit strong reasoning capabilities through step-by-step generation, known as chain-of-thought (CoT) reasoning (Wei et al., 2022; Cobbe et al., 2021; Fang et al., 2024). However, this approach often leads to unnecessarily long and verbose reasoning traces, resulting in high inference cost and latency. This *overthinking phenomenon* has become a practical bottleneck, especially in real-time or resource-constrained scenarios (Sui et al., 2025).

To improve reasoning efficiency, recent studies have introduced methods that allow models to generate accurate answers with fewer steps. These

Question:

Marilyn's first record sold 10 times as many copies as Harald's. If they sold 88,000 copies combined, how many copies did Harald sell?

Model Generation:

<think> Okay, so I need to figure out how many copies Harald sold. </think> 1000  
...  
So  $11H = 88,000$  I need to solve for it by dividing both sides by 11 </think> 8800  
88,000 divided by 11 should be 8,000 </think> 8000  
Wait, let me double-check that. </think> 8000  
If I multiply 11 by 8,000, I should get 88,000. That checks out. </think> 8000  
I don't think I made any mistakes here. </think> 8000  
So, Harald sold 8,000 copies. </think> 8000

Converged!

Figure 1: R1-Qwen-32B converges early on a GSM8K example, suggesting that later steps could be skipped.

include reinforcement learning with length-aware rewards (Luo et al., 2025; Hou et al., 2025), fine-tuning on variable-length CoT traces (Han et al., 2024; Xia et al., 2025), and prompt-based approaches that request concise reasoning (Xu et al., 2025; Nayab et al., 2024; Han et al., 2024). These methods typically require retraining on curated data or task-specific prompt design. In contrast, we explore inference-time techniques that improve efficiency without sacrificing accuracy. Particularly, we hypothesize that LLMs often internally converge on an answer before completing the full reasoning trace, an insight we formalize as *answer convergence*. Recognizing such convergence can enable more efficient inference by allowing early stopping without sacrificing accuracy.

To investigate this, we conduct a reasoning model early-stopping study that systematically truncates explicit CoTs to assess when the model’s answer generated answer converges, i.e., when the answer remains unchanged despite additional reasoning steps. Our experiments reveal that models often converge well before completing the full reasoning chain, suggesting substantial redundancy and highlighting the potential for improving efficiency through early stopping. Figure 1 shows an example where the model answer converges early despite receiving only partial reasoning, indicating that the remaining steps contribute little to the final prediction. As shown in Figure 2, this pattern holds

<sup>1</sup>Code will be released upon publication.

across five datasets, even on GPQA, a challenging dataset, many examples converge early.

Motivated by this finding, we propose inference-time strategies to dynamically truncate explicit reasoning based on the observation that models often reach converged answers early: (1) Early stopping via **Answer Consistency**, which halts generation when consecutive reasoning chunks yield identical answers; (2) **Think Token Adjustment**, which encourages models to signal early termination explicitly; and (3) a supervised approach, **Learn-to-Stop**, which utilizes internal activations to predict optimal stopping points. Our methods are *model-agnostic*, require no additional training or LM modification, and significantly reduce inference cost without sacrificing accuracy. Importantly, this method does not require ground-truth answer labels, relying only on self-consistency signals during inference.

We evaluate our methods on five reasoning benchmarks and five open-weights LLMs. Results show that early stopping strategies consistently reduce token usage without hurting accuracy or even improving accuracy, especially on simpler tasks. Specifically, (1) **Performance vs. token length**: *Learn-to-Stop* cuts up to 48% of tokens on NQ with QwQ-32B, sometimes even improving accuracy, suggesting that excessive reasoning may introduce unnecessary noise. (2) **Supervised vs. unsupervised**: Unsupervised methods work well on NQ and GSM8K, while the supervised approach generalizes better to harder tasks like MATH-500 and GPQA.

## 2 Related Work

Prior work on CoT efficiency falls into three main categories: (1) reinforcement learning with length-aware rewards or difficulty-adaptive reasoning (Luo et al., 2025; Hou et al., 2025; Shen et al., 2025); (2) supervised fine-tuning that skips unimportant tokens, enforces token budgets, or uses self-training to learn shorter rationales (Xia et al., 2025; Han et al., 2024; Munkhbat et al., 2025); and (3) prompt-based methods that request concise reasoning or dynamically select reasoning paths (Xu et al., 2025; Aytes et al., 2025; Cheng et al., 2025).

Unlike these methods which rely on training-time optimization on carefully designed datasets or application-specific prompt design, our method operates entirely at inference time. By identifying internal answer convergence, we enable dynamic early stopping without retraining, model changes, labeled data, or prompt engineering.

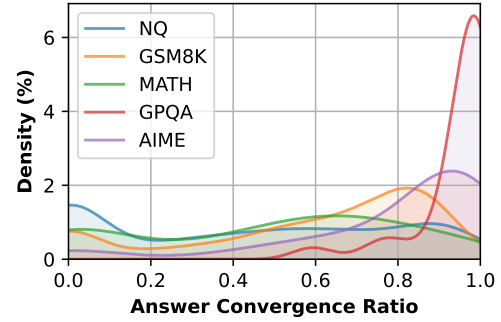


Figure 2: Distribution of Answer Convergence Ratios (ACRs) across tasks. Models often converge early, suggesting that many generated steps might be redundant from the model’s perspective.

## 3 Preliminary

We investigate whether all parts of a CoT reasoning chain are necessary for the model to converge on its predicted answer, and whether later steps can be omitted without affecting the decision. To this end, we first split the CoT into sentence-level chunks using the NLTK tokenizer (Bird et al., 2009). These chunks are then incrementally concatenated (e.g., chunk1, chunk1+chunk2, etc.), each followed by an end-of-reasoning token `</think>`. The model is prompted to generate an answer from each partial chain via greedy decoding. By tracking when the model’s prediction remains unchanged across successive reasoning steps, we identify the earliest point of answer convergence, approximating *the minimum reasoning required for the model to reach a stable decision*.

We apply this protocol to five tasks with varying levels of reasoning: **NaturalQuestions (NQ)** (Kwiatkowski et al., 2019), **GSM8K** (Cobbe et al., 2021), **MATH-500** (Lightman et al., 2024), **GPQA-Diamond** (Rein et al., 2023), and **AIME’24**<sup>2</sup>. NQ involves minimal reasoning as an information-seeking task, while the others are math and logic benchmarks of increasing difficulty, with AIME’24 containing the most advanced problems. Experiments are conducted using the R1-distilled Qwen-32B model (DeepSeek-AI, 2025)<sup>3</sup>. To quantify when the model converges, we define the **Answer Convergence Ratio (ACR)** as the proportion of reasoning required before the predicted answer remains unchanged. For each instance, we detect the earliest point where the answer persists through the end of the chain. For example, convergence

<sup>2</sup>[https://huggingface.co/datasets/HuggingFaceH4/aime\\_2024](https://huggingface.co/datasets/HuggingFaceH4/aime_2024)

<sup>3</sup>Results for various models are provided in Appendix A.1.

after 7 out of 10 chunks yields an ACR of 0.7.

Figure 2 displays the distribution of ACRs across the five tasks. We observe that the model often converges early, suggesting that many steps are unnecessary. The ACR distribution peaks near 0.0 for NQ, which means nearly no reasoning is needed for the model to come up with its final answer, around 0.8 for GSM8K and MATH-500, and near 0.9 for GPQA and AIME’24, mirroring the increasing reasoning loads. These results suggest that early stopping is often feasible and can reduce inference-time costs without affecting answer quality.

## 4 Early Stopping at Inference Time

We propose three methods to improve reasoning efficiency: two unsupervised approaches based on answer consistency (§4.1) and decoding signals (§4.2), and one supervised method (§4.3) that predicts when to stop reasoning without retraining the LM or modifying its parameters.

### 4.1 Detecting Answer Consistency

Since LLMs often reach converged answers before completing the full reasoning chain, we introduce an unsupervised stopping criterion based on output consistency. During decoding, we monitor the model’s outputs and append the `</think>` token at predicted natural sentence boundaries, prompting it to produce an answer via greedy decoding. If the same answer is produced for a fixed number  $k$  of consecutive chunks, we consider the reasoning converged and terminate further generation.

### 4.2 Think Token Adjustment

During decoding, the model uses the `</think>` token to indicate the end of reasoning. Ideally, if the model has reached the final answer, it should generate this token early. However, we observe that while `</think>` often ranks among the top 10 candidates after sentence boundaries, the model tends to prefer tokens like `wait`, `or`, or `but`, which unnecessarily prolong reasoning.

To address this, we boost the probability of `</think>` during decoding by applying a linear logit transformation<sup>4</sup>:  $y_{t^*} \leftarrow y_{t^*} + \alpha \cdot (\max(\mathbf{y}) - \frac{1}{|\mathbf{y}|} \sum_i y_i)$ , where  $y_{t^*}$  is the logit of the `</think>` token,  $\alpha$  controls the boost strength, and  $\mathbf{y}$  denotes all vocabulary logits. This encourages the model to terminate reasoning earlier when appropriate.

### 4.3 Learning When to Stop Reasoning

Recent work shows that LLM activations encode useful signals such as knowledge and confidence

(Kapoor et al., 2024; Liu et al., 2024). We hypothesize that they also capture reasoning progress, including when to stop. Specifically, the final-layer activation  $\mathbf{h}_t$  may reflect both the model’s certainty and the need for further computation.

To leverage this, we train a supervised model to predict optimal stopping points using the model’s internal activations. Given the sequential nature of reasoning, we use an LSTM to encode the activation sequence  $\{\mathbf{h}_1, \dots, \mathbf{h}_T\}$ . At each chunk  $t$ , the LSTM output  $\mathbf{z}_t$  is passed to a sigmoid classifier:  $\hat{p}_t = \sigma(\mathbf{W}\mathbf{z}_t + b)$ , where  $\hat{p}_t \in [0, 1]$  represents the probability of stopping at step  $t$ .

Training labels are constructed by identifying the earliest chunk where the predicted answer matches the final answer and remains unchanged. Chunks from that point onward are labeled 1, and all earlier chunks are labeled 0. We optimize a binary cross-entropy loss:  $\mathcal{L} = -\frac{1}{T} \sum_{t=1}^T [p_t \log \hat{p}_t + (1 - p_t) \log(1 - \hat{p}_t)]$ , where  $p_t \in \{0, 1\}$  is the ground truth label, and  $T$  is the number of chunks. At inference time, reasoning stops when  $\hat{p}_t \geq \tau$ , with threshold  $\tau$  tuned on validation data. This approach enables us to utilize the model’s internal dynamics to *improve reasoning efficiency without training or modifying the base LLM*.

## 5 Experimental Setup

We evaluate our methods on the five tasks introduced in § 3, with dataset splits and statistics detailed in Appendix A.2. Experiments are conducted on five LLMs across three families: R1-distilled Qwen and Llama (DeepSeek-AI, 2025), and QwQ (Team, 2025). We report Accuracy (Acc.), average generated tokens (Tokens #), and token reduction (%) relative to the original model. As baselines, we include the original model and Concise CoT (CCoT) (Nayab et al., 2024), which limits reasoning length via prompt token budgets.<sup>5</sup>

## 6 Results and Analysis

From the results in Table 1, we observe the following key findings:

**Early stopping improves performance on tasks with low reasoning demand.** On NQ, which involves minimal reasoning, all early stopping methods match or surpass the original model’s accuracy while greatly reducing token usage. This suggests that the original model may overthink and generate unnecessary reasoning steps, which even harms

<sup>4</sup>Implementation details are provided in Appendix A.3.

<sup>5</sup>The implementation details and prompts are provided in Appendix A.4.

Model	Method	NQ		GSM8K		MATH-500		GPQA		AIME'24	
		Acc.(%) ↑	Tokens # ↓	Acc.(%) ↑	Tokens # ↓	Acc.(%) ↑	Tokens # ↓	Acc.(%) ↑	Tokens # ↓	Acc.(%) ↑	Tokens # ↓
R1-Qwen-7B	Original	11.6	522.3	81.0	180.7	91.0	1873.9	22.2	3131.2	56.7	10768.4
	CCoT	<u>12.4</u>	349.5 (-33.1%)	80.0	179.4 (-0.7%)	<u>88.7</u>	1345.0 (-28.2%)	<u>21.2</u>	2230.7 (-28.8%)	<b>53.3</b>	8109.4 (-24.7%)
	Answer Consistency	<u>12.4</u>	371.1 (-28.9%)	<b>81.0</b>	169.4 (-6.3%)	67.0	<u>620</u> (-66.9%)	11.1	<u>1076.0</u> (-65.6%)	13.3	<u>1355.2</u> (-87.4%)
	Think Token Adjustment	11.8	406.7 (-22.1%)	75.8	158.2 (-12.5%)	65.0	1317.7 (-29.7%)	19.2	1894.0 (-39.5%)	<u>36.7</u>	6225.0 (-42.2%)
	Learn to Stop	<b>12.6</b>	<u>280.9</u> (-46.2%)	<u>80.4</u>	<u>158.0</u> (-12.6%)	<b>89.0</b>	<u>1356.1</u> (-27.6%)	<b>22.2</b>	<u>2786.6</u> (-11.0%)	—	—
R1-Qwen-32B	Original	35.0	522.7	89.4	236.3	91.0	1958.8	33.3	3278.5	73.3	8471.5
	CCoT	37.2	305.4 (-41.6%)	<b>93.4</b>	239.2 (+1.2%)	88.0	<u>1298.7</u> (-33.7%)	<u>27.3</u>	<u>1928.6</u> (-41.2%)	<b>60.0</b>	4869.4 (-42.5%)
	Answer Consistency	<b>38.4</b>	331.1 (-36.7%)	87.6	<u>193.1</u> (-18.3%)	55.0	<u>496.4</u> (-74.7%)	14.1	<u>771.8</u> (-76.5%)	13.3	<u>1522.2</u> (-82.0%)
	Think Token Adjustment	<u>38.2</u>	426.8 (-18.3%)	87.6	221.2 (-6.4%)	68.0	954.0 (-51.3%)	24.2	1544.5 (-52.9%)	<u>43.3</u>	<u>2431.3</u> (-71.3%)
	Learn to Stop	38.0	<u>273.2</u> (-47.7%)	86.8	<u>157.7</u> (-33.3%)	<b>90.0</b>	<u>1621.1</u> (-17.2%)	<b>30.3</b>	<u>2684.9</u> (-18.1%)	—	—
R1-Llama-8B	Original	20.6	426.8	78.6	401.3	79.0	2011.8	14.1	2546.0	40.0	7914.4
	CCoT	24.8	308.7 (-27.7%)	72.0	266.1 (-33.7%)	<u>71.0</u>	1683.3 (-16.3%)	12.2	1705.0 (-33.0%)	<b>30.0</b>	4060.8 (-48.7%)
	Answer Consistency	22.4	336.3 (-21.2%)	69.2	319.8 (-20.3%)	56.0	<u>841.6</u> (-58.2%)	12.1	<u>1138.6</u> (-55.3%)	16.7	<u>1522.2</u> (-80.8%)
	Think Token Adjustment	<b>25.2</b>	407.5 (-4.5%)	66.8	329.4 (-17.9%)	<u>71.0</u>	981.5 (-51.2%)	9.1	1558.1 (-38.8%)	<u>23.3</u>	<u>3571.3</u> (-54.9%)
	Learn to Stop	21.8	<u>281.2</u> (-34.1%)	<b>77.4</b>	<u>350.1</u> (-12.8%)	<b>74.0</b>	<u>1801.8</u> (-10.4%)	<b>13.1</b>	<u>2468.4</u> (-3.0%)	—	—
R1-Llama-70B	Original	50.4	417.2	90.6	291.2	91.0	1577.4	29.3	2419.8	56.7	4972.3
	CCoT	52.0	294.4 (-29.4%)	<u>90.0</u>	227.6 (-21.8%)	<b>87.0</b>	1101.5 (-30.2%)	<u>29.3</u>	1935.1 (-20.0%)	<b>56.7</b>	4235.9 (-14.8%)
	Answer Consistency	<b>54.4</b>	273.5 (-34.4%)	86.2	235.4 (-19.2%)	65.0	<u>578.2</u> (-63.3%)	20.2	<u>925.8</u> (-61.7%)	16.7	<u>1295.6</u> (-73.9%)
	Think Token Adjustment	52.0	378.5 (-9.3%)	88.6	279.5 (-4.0%)	85.0	1322.2 (-16.2%)	28.3	2164.6 (-10.5%)	<u>53.3</u>	<u>4057.5</u> (-18.4%)
	Learn-to-Stop	<u>52.2</u>	<u>223.5</u> (-46.4%)	<b>91.2</b>	<u>252.8</u> (-13.2%)	<u>86.0</u>	<u>1204.1</u> (-23.7%)	<b>30.3</b>	<u>2268.9</u> (-6.2%)	—	—
QwQ-32B	Original	41.6	646.0	96.8	755.1	98.0	2996.8	25.3	6557.9	76.7	12160.3
	CCoT	42.4	392.2 (-39.3%)	<u>95.2</u>	419.3 (-44.5%)	<u>95.0</u>	2266.7 (-24.4%)	22.3	4663.5 (-28.9%)	<b>73.0</b>	8890.8 (-26.9%)
	Answer Consistency	<b>43.0</b>	355.3 (-45.0%)	89.0	342.6 (-54.6%)	<u>99.0</u>	<u>684.3</u> (-77.2%)	15.2	<u>1255.3</u> (-80.9%)	20.0	<u>2334.6</u> (-80.8%)
	Think Token Adjustment	42.8	686.6 (+6.3%)	<b>96.6</b>	745.2 (-1.3%)	<b>96.0</b>	3055.7 (+2.0%)	<b>31.3</b>	6131.1 (-6.5%)	<u>70.0</u>	11489.6 (-5.5%)
	Learn to Stop	<b>43.0</b>	<u>335.9</u> (-48.0%)	<b>96.6</b>	<u>418.9</u> (-44.5%)	93.0	<u>2050.0</u> (-31.6%)	<u>24.2</u>	<u>3988.3</u> (-39.2%)	—	—

Table 1: Evaluation results on five reasoning tasks and five model families. For each task, the best and second-best accuracies are shown in **bold** and underline, respectively. In the Tokens # columns, darker green indicates lower inference cost. Our early stopping methods notably reduce tokens, with Learn-to-Stop offering the best efficiency–performance tradeoff.

**Question:** Carlos is planting a lemon tree. The tree will cost \$90 to plant. Each year it will grow 7 lemons, which he can sell for \$1.5 each. It costs \$3 a year to water and feed the tree. How many years will it take before he starts earning money on the lemon tree?

**Original:** To determine how many years ..... Solving for  $n$ :  $n = \$90 / \$7.5$ ,  $n =$ 12  
Wait, let me double-check my calculations to make sure I didn't make a mistake ... That seems correct.

</think>\boxed{12}

**Token #:** 439

**Learn to Stop:** Okay, so Carlos is planting ..... Calculating that, 90 divided by 7.5 equals 12.

</think>\boxed{12}

**Token #:** 338

**Model:** R1-Qwen-32B

Table 2: Example from GSM8K comparing the original model and Learn-to-Stop. The original generates redundant reasoning after reaching 12, while Learn-to-Stop terminates reasoning for a more concise response.

the performance. Manual inspection reveals that the original model often overthinks—adding unnecessary self-reflection or even revising correct answers—whereas early stopping prevents such degradation.<sup>6</sup> The early stopping methods effectively mitigate this issue by terminating reasoning earlier, leading to improved performance.

**Unsupervised methods are effective for simpler tasks.** On NQ and GSM8K, Answer Consistency and Think Token Adjustment reduce tokens without harming accuracy, despite requiring no additional training. However, on harder tasks like MATH-500 and GPQA, their performance becomes less stable, indicating that shallow signals

like output consistency may be insufficient under high reasoning complexity.

## Learn-to-Stop offers robust efficiency gains.

This supervised method consistently balances accuracy and token savings across models and tasks. For example, on GSM8K with QwQ-32B, it reduces tokens by 44% (755.1 to 418.9) with only a 0.2% drop in accuracy; on the harder task GPQA, it cuts 39.2% tokens with comparable performance (24.2% vs. 25.3%). Compared to CCoT, Learn-to-Stop achieves greater reductions while maintaining or improving accuracy. Since it operates purely at inference time and keeps the prompt unchanged, it is compatible with prompt-level strategies like CCoT, offering opportunities for further gains through combination. Table 5 exemplify how Learn-to-Stop avoids redundant reasoning while preserving output quality.<sup>7</sup>

## 7 Conclusion

We study how to reduce redundancy in chain-of-thought (CoT) reasoning to improve LLM inference efficiency. Across five benchmarks and five open-weight LLMs, we find that answer convergence often occurs early, revealing substantial redundancy. Based on this, we propose three inference-time methods that stop generation once reasoning is sufficient. These methods cut token usage by up to 40% without accuracy loss, offering a practical alternative to full-chain reasoning without retraining or model changes.

<sup>6</sup>Examples in Appendix A.5.

<sup>7</sup>Examples of other datasets in Appendix A.5.1.



## Limitations

While our proposed early stopping strategies significantly reduce inference cost with minimal or no loss in accuracy, several limitations remain. First, our methods rely on the assumption that the model’s answer convergence correlates with the correctness of the final output. However, convergence does not guarantee correctness, especially in tasks with higher reasoning complexity (e.g., GPQA and AIME). Second, by enabling models to make predictions without observing the full reasoning trace, our approach may compromise the faithfulness of reasoning. Future work should aim to jointly optimize for both faithfulness and conciseness, ensuring that reasoning remains both efficient and trustworthy.

## References

- Simon A. Aytes, Jinheon Baek, and Sung Ju Hwang. 2025. [Sketch-of-thought: Efficient LLM reasoning with adaptive cognitive-inspired sketching](#). *CoRR*, abs/2503.05179.
- Steven Bird, Edward Loper, and Ewan Klein. 2009. *Natural Language Processing with Python: Analyzing Text with the Natural Language Toolkit*. O’Reilly Media, Inc.
- Xiaoxue Cheng, Junyi Li, Wayne Xin Zhao, and Ji-Rong Wen. 2025. [Think more, hallucinate less: Mitigating hallucinations via dual process of fast and slow thinking](#). *CoRR*, abs/2501.01306.
- Karl Cobbe, Vineet Kosaraju, Mohammad Bavarian, Mark Chen, Heewoo Jun, Lukasz Kaiser, Matthias Plappert, Jerry Tworek, Jacob Hilton, Reiichiro Nakano, Christopher Hesse, and John Schulman. 2021. [Training verifiers to solve math word problems](#). *CoRR*, abs/2110.14168.
- DeepSeek-AI. 2025. [Deepseek-r1: Incentivizing reasoning capability in llms via reinforcement learning](#). *Preprint*, arXiv:2501.12948.
- Meng Fang, Shilong Deng, Yudi Zhang, Zijing Shi, Ling Chen, Mykola Pechenizkiy, and Jun Wang. 2024. [Large language models are neurosymbolic reasoners](#). In *Thirty-Eighth AAAI Conference on Artificial Intelligence, AAAI 2024, Thirty-Sixth Conference on Innovative Applications of Artificial Intelligence, IAAI 2024, Fourteenth Symposium on Educational Advances in Artificial Intelligence, EAAI 2014, February 20-27, 2024, Vancouver, Canada*, pages 17985–17993. AAAI Press.
- Tingxu Han, Zhenting Wang, Chunrong Fang, Shiyu Zhao, Shiqing Ma, and Zhenyu Chen. 2024. [Token-budget-aware LLM reasoning](#). *CoRR*, abs/2412.18547.
- Bairu Hou, Yang Zhang, Jiabao Ji, Yujian Liu, Kaizhi Qian, Jacob Andreas, and Shiyu Chang. 2025. [Thinkprune: Pruning long chain-of-thought of llms via reinforcement learning](#). *arXiv preprint arXiv:2504.01296*.
- Sanyam Kapoor, Nate Gruver, Manley Roberts, Katie Collins, Arka Pal, Umang Bhatt, Adrian Weller, Samuel Dooley, Micah Goldblum, and Andrew Gordon Wilson. 2024. [Large language models must be taught to know what they don’t know](#). In *Advances in Neural Information Processing Systems 38: Annual Conference on Neural Information Processing Systems 2024, NeurIPS 2024, Vancouver, BC, Canada, December 10 - 15, 2024*.
- Tom Kwiatkowski, Jennimaria Palomaki, Olivia Redfield, Michael Collins, Ankur P. Parikh, Chris Alberti, Danielle Epstein, Illia Polosukhin, Jacob Devlin, Kenton Lee, Kristina Toutanova, Llion Jones, Matthew Kelcey, Ming-Wei Chang, Andrew M. Dai, Jakob Uszkoreit, Quoc Le, and Slav Petrov. 2019. [Natural questions: a benchmark for question answering research](#). *Trans. Assoc. Comput. Linguistics*, 7:452–466.
- Woosuk Kwon, Zhuohan Li, Siyuan Zhuang, Ying Sheng, Lianmin Zheng, Cody Hao Yu, Joseph Gonzalez, Hao Zhang, and Ion Stoica. 2023. [Efficient memory management for large language model serving with pagedattention](#). In *Proceedings of the 29th Symposium on Operating Systems Principles, SOSP 2023, Koblenz, Germany, October 23-26, 2023*, pages 611–626. ACM.
- Hunter Lightman, Vineet Kosaraju, Yuri Burda, Harrison Edwards, Bowen Baker, Teddy Lee, Jan Leike, John Schulman, Ilya Sutskever, and Karl Cobbe. 2024. [Let’s verify step by step](#). In *The Twelfth International Conference on Learning Representations, ICLR 2024, Vienna, Austria, May 7-11, 2024*. OpenReview.net.
- Xin Liu, Farima Fatahi Bayat, and Lu Wang. 2024. [Enhancing language model factuality via activation-based confidence calibration and guided decoding](#). In *Proceedings of the 2024 Conference on Empirical Methods in Natural Language Processing, EMNLP 2024, Miami, FL, USA, November 12-16, 2024*, pages 10436–10448. Association for Computational Linguistics.
- Haotian Luo, Li Shen, Haiying He, Yibo Wang, Shiwei Liu, Wei Li, Naiqiang Tan, Xiaochun Cao, and Dacheng Tao. 2025. [O1-pruner: Length-harmonizing fine-tuning for o1-like reasoning pruning](#). *CoRR*, abs/2501.12570.
- Tergel Munkhbat, Namgyu Ho, Seo Hyun Kim, Yongjin Yang, Yujin Kim, and Se-Young Yun. 2025. [Self-training elicits concise reasoning in large language models](#). *CoRR*, abs/2502.20122.
- Sania Nayab, Giulio Rossolini, Giorgio C. Buttazzo, Nicolamaria Manes, and Fabrizio Giacomelli. 2024.

Concise thoughts: Impact of output length on LLM reasoning and cost. *CoRR*, abs/2407.19825.

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

Yi Shen, Jian Zhang, Jieyun Huang, Shuming Shi, Wenjing Zhang, Jiangze Yan, Ning Wang, Kai Wang, and Shiguo Lian. 2025. DAST: difficulty-adaptive slow-thinking for large reasoning models. *CoRR*, abs/2503.04472.

Yang Sui, Yu-Neng Chuang, Guanchu Wang, Jiamu Zhang, Tianyi Zhang, Jiayi Yuan, Hongyi Liu, Andrew Wen, Shaochen Zhong, Hanjie Chen, and Xia Ben Hu. 2025. Stop overthinking: A survey on efficient reasoning for large language models. *CoRR*, abs/2503.16419.

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

Jason Wei, Xuezhi Wang, Dale Schuurmans, Maarten Bosma, Brian Ichter, Fei Xia, Ed H. Chi, Quoc V. Le, and Denny Zhou. 2022. Chain-of-thought prompting elicits reasoning in large language models. In *Advances in Neural Information Processing Systems 35: Annual Conference on Neural Information Processing Systems 2022, NeurIPS 2022, New Orleans, LA, USA, November 28 - December 9, 2022*.

Heming Xia, Yongqi Li, Chak Tou Leong, Wenjie Wang, and Wenjie Li. 2025. Tokenskip: Controllable chain-of-thought compression in llms. *CoRR*, abs/2502.12067.

Silei Xu, Wenhao Xie, Lingxiao Zhao, and Pengcheng He. 2025. Chain of draft: Thinking faster by writing less. *CoRR*, abs/2502.18600.

## A Appendix

### A.1 ACR Distribution on Various Models

We present the ACR distributions for several models: R1-distilled Llama-8B, R1-distilled Llama-70B, R1-distilled Qwen-7B, and QwQ-32B. The results are shown in Figure 3. All models exhibit similar trends to the R1-distilled Qwen-32B baseline, confirming their ability to produce stable answers before completing the full reasoning chain. Moreover, tasks with higher reasoning demands tend to correspond to higher ACRs. When comparing distributions across models, we observe that larger models generally achieve lower ACRs, suggesting that they require fewer reasoning steps to converge on an answer. This implies that larger models may possess a more efficient internal reasoning process, enabling them to reach final answers more quickly.

Task	Train	Validation	Test
NQ	800	200	3610
GSM8K	800	200	1319
MATH-500	320	80	100
GPQA-Diamond	78	20	100
AIME’24	—	—	30

Table 3: Dataset statistics for each task.

### A.2 Dataset Splits and Statistics

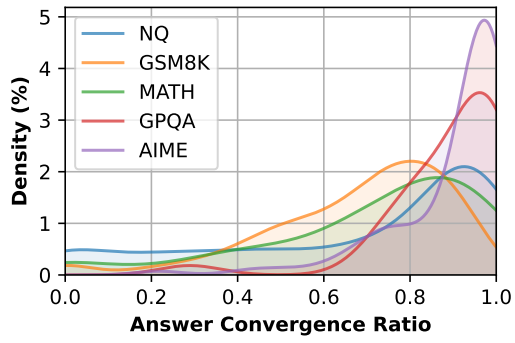
For tasks with available training data (NQ and GSM8K), we sample 1,000 examples for training the supervised method and 100 for validation. For tasks without predefined training data (MATH-500 and GPQA-Diamond), we reserve 100 examples as the test set and split the remaining data into 80% for training and 20% for validation. Due to the limited size of AIME’24, which includes only 30 test examples, we evaluate only the unsupervised methods on this task. The statistics for each task are shown in Table 3.

### A.3 Implementation of Early Stopping via Boosting End of Think Token

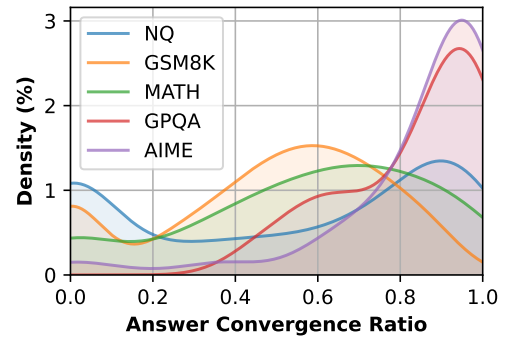
To implement boosting of the `</think>` token, we design a logit processor that adjusts the model’s output logits before sampling. Specifically, it increases the logit of the `</think>` token according to §4.2. Once the `</think>` is generated, the processor is disabled to prevent further modifications. This mechanism allows the model to emit the `</think>` token earlier if it has already reached a confident answer. We integrate the logit processor into the VLLM framework (Kwon et al., 2023), enabling efficient logit manipulation during decoding without compromising throughput. We run all experiments three times and report the average results.

### A.4 Implementation Details

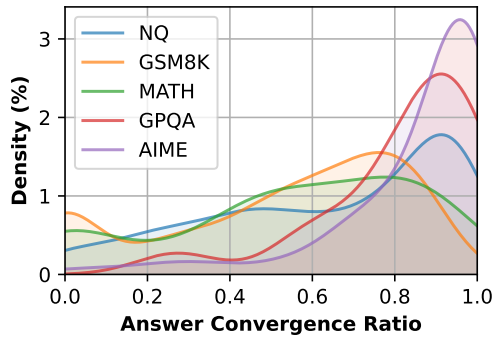
When applying early stopping via answer consistency, we empirically set the number of consecutive chunks  $k$  to 10. For early stopping via boosting, we set the hyperparameter  $\alpha$  to 0.6. Regarding the supervised method, we use a single-layer LSTM with 128 hidden units and a dropout rate of 0.1. The model is trained for 200 epochs with a batch size of 32, using the Adam optimizer with a learning rate of  $5e^{-4}$ . The confidence threshold  $\tau$  is tuned on the validation set, and we set it to 0.50,



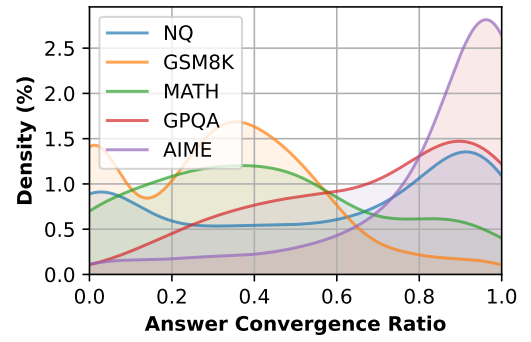
(a) R1-distilled-Llama-8B



(b) R1-distilled-Llama-70B



(c) R1-distilled-Qwen-7B



(d) QwQ-32B

Figure 3: ACR distributions for various models.

0.99, 0.99, and 0.50 for NQ, GSM8K, MATH-500, and GPQA-Diamond, respectively. For the CCoT baseline, we set the token budget to 100 for all tasks.

We use the VLLM framework (Kwon et al., 2023) to sample the model outputs for all experiments to ensure efficient inference. All the prompts we used are provided in Appendix A.6.

### A.5 Test cases on NQ

We list two examples from NQ in Table 4. In the first example, the original model initially generates the correct answer (Moir Kelly), but then adds unnecessary reasoning steps, including self-reflection, and revises the answer to an incorrect one (Julie Kavner). Similarly, in the second example, the model starts with the correct answer (Ashoka), but again adds unnecessary reasoning, ultimately revising it to an incorrect answer (Kalinga Nanda).

#### A.5.1 Case Study

We provide a case study of the original model and the Learn-to-Stop method in Table 5.

### A.6 Prompts

#### A.6.1 NQ

```
Answer the following question.
Directly output your final answer within \boxed{}. DO
NOT say anything else.

Question: {question}
```

#### A.6.2 Math Reasoning Tasks

```
Solve the following math problem.
Directly output your final answer within \boxed{}. DO
NOT say anything else.

Question: {question}
```

#### A.6.3 CCoT Prompt

```
Answer the following question.
You should think step by step, and limit the thinking
process length to {LENGHT_BUGGED} words.
Directly output your final answer within \boxed{}. DO
NOT say anything else.

Question: {question}
```

```
Solve the following math problem.
You should think step by step, and limit the thinking
process length to {LENGHT_BUGGED} words.
Directly output your final answer within \boxed{}. DO
NOT say anything else.

Question: {question}
```



<b>Question:</b> who does the voice of nala in the lion king
<b>Original:</b> Okay, so I need to figure out who voices Nala in The Lion King. I remember watching the movie when I was a kid, but I can't recall the voice actress. Let me think. I know that in the original 1994 version, some of the main voices were well-known actors. Simba was voiced by Jonathan Taylor Thomas, Mufasa by James Earl Jones, and Scar by Jeremy Irons. But Nala... I'm not sure. Maybe it was someone like Jennifer Hudson? Wait, no, she was in the 2019 remake, I think. In the original, I think it was a different actress. Maybe someone like, I don't know, <b>Moira Kelly</b> ? No, that doesn't sound right. Maybe it's someone else. Let me think of other voice actors from that era. Maybe it's someone like <b>Eliza Dushku</b> ? Or maybe it's someone less famous. I think it's a British actress. Oh, maybe it's someone like, I don't know, <b>Julie Kavner</b> ? Wait, no, she did Mufasa's sister in the original...
<b>Correct Answer #:</b> <b>Moira Kelly</b>
<b>Question:</b> the king of kalinga in the mauryan period was
<b>Original:</b> Okay, so I need to figure out who the king of Kalinga was during the Mauryan period. Hmm, I remember that the Mauryan Empire was a significant empire in ancient India, and it was around the 3rd century BCE. The most famous king from that period is <b>Ashoka</b> the Great, known for his role in the Kalinga war., but the question is about the king of Kalinga, not the Mauryan king. So <b>Ashoka</b> was the Mauryan emperor, and he conquered Kalinga. I think the king of Kalinga at that time might have been a local ruler who resisted <b>Ashoka</b> 's expansion. There's a specific name that comes to mind, something like <b>Kalinga Nanda</b> . I'm not entirely sure, but I recall that Nanda was a common name for kings in that region, and <b>Kalinga Nanda</b> was the ruler who opposed <b>Ashoka</b> .should check if there are any other possible names. Some sources mention that the king of Kalinga was indeed <b>Kalinga Nanda</b> , and that <b>Ashoka</b> 's conquest of Kalinga was a major event in his reign, leading to his conversion to Buddhism. So yeah, <b>Kalinga Nanda</b> is the name I'm looking for...
<b>Correct Answer #:</b> <b>Ashoka</b>
<b>Model:</b> R1-Qwen-32B

Table 4: Examples from NQ with responses from the original model.

---

**Question:** Each week, between 30 and 50 students show up for an archery class run by Betty and Wilma. Usually the students break up into groups of equal size for target practice. However, this week, Betty noticed that she could not break the students up into multiple groups of equal size. Wilma noticed that if she and Betty both joined the students in practicing, they still could not break the archers up into groups of equal size. How many students showed up to the archery class this week?

**Original:** Okay, so I have this math problem here about Betty and Wilma's archery class. Let me try to understand what it's asking..... $47:47 + 2 = 49$ . 49 is not prime because  $7*7=49$ . So 47 is out.the only number that fits both conditions is 41. When you add 2, it becomes 43, which is also prime. Therefore, there must have been 41 students at the archery class this week.

`</think>\boxed{41}`

**Token #:** 741

---

**Learn to Stop:** Okay, so I need to figure out how many students showed up to the archery class this week. The problem says that each week, between 30 and ..... 41 fits both conditions.

`</think>\boxed{41}`

**Token #:** 673

---

**Dataset:** MATH-500

**Question:** In a specific region of the sky, astronomers have observed that the number of stars varies with parallax as  $1/\text{plx}^5$ . How does the number of stars in that region of the sky change with distance (per unit range of distance,  $r$ )?

**Original:** Okay, so I've got this problem here where astronomers observed that the number of stars varies with parallax as  $1/\text{plx}$  to the fifth power. Hmm, I need to figure out how the number of stars changes with distance, specifically per unit range of distance  $r$ ..... Therefore,  $5k / (1/r^6 * r) = 5k r^6 / r = 5k r^4$ . Yes, that's correct. So, the number of stars per unit distance is proportional to  $r^4$ . Wait, but that would mean that as you go further out, the number of stars per unit distance increases rapidly. That seems odd because usually, the number of stars in a volume element increases with the square of the radius (since volume is  $4\pi r^3$ ), but here it's increasing faster than that. Maybe because the number density itself is increasing with distance? Wait, no. The problem states that the number of stars varies with parallax as  $1/\text{plx}^5$ . So, if parallax decreases (distance increases), the number increases as  $1/\text{plx}^5$ , which is equivalent to  $d^5$ . So, the total number up to distance  $d$  is proportional to  $d^5$ . But the number per unit distance would require considering the derivative. Wait, maybe I'm confusing cumulative counts with differential counts. Let me clarify. If  $N(\text{pl})$  is the total number of stars with parallax less than  $\text{pl}$ , then  $N(\text{pl}) = k / \text{pl}^5$ . But parallax less than  $\text{pl}$  corresponds to distances greater than  $d = 1/\text{pl}$ . So,  $N(d) = k d^5$ . But then the number of stars beyond distance  $d$  would be  $N_{\text{total}} - N(d) = k d^5$ . Hmm, that doesn't make much sense because as  $d$  increases,  $N(d)$  increases, meaning more stars are at larger distances, which is expected. But to find the number per unit distance, we take the derivative of  $N(d)$  with respect to  $d$ . So,  $dN/dd = 5k d^4$ . So, the number of stars per unit distance at distance  $d$  is proportional to  $d^4$ . Wait, but that would imply that the number density is increasing with distance, which might not align with standard stellar distributions unless there's an actual increase in star density with distance, which isn't typically the case. Normally, the number density of stars in the galaxy decreases with distance from the galactic center, but in this problem, we're probably considering a region where the distribution follows this specific law. So, putting it all together, the number of stars per unit distance ( $r$ ) is proportional to  $r^4$ .

`</think>\boxed{r^4}`

**Token #:** 1952

---

**Learn to Stop:** Okay, so I have this problem where the number of stars varies with parallax as  $1/\text{plx}$  to the fifth power. Hmm, I remember that parallax is related to distance. Let me think about how they're connected., parallax ( $\text{pl}$ ) is inversely proportional to distance ( $d$ ) ..... Therefore, the number of stars per unit distance ( $N(r)$ ) is proportional to  $r^4$ .

`</think>\boxed{r^4}`

**Token #:** 1431

---

**Dataset:** GPQA-Diamond

Table 5: Examples from MATH-500 and GPQA-Diamond with responses from the original model and the Learn-to-Stop method. The original model generates redundant reasoning steps after producing the correct answer (41 and  $r^4$ ), while the Learn-to-Stop method terminates reasoning earlier, resulting in a more concise response.