

Think Twice Before You Write - an Entropy-based Decoding Strategy to Enhance LLM Reasoning

Anonymous ACL submission

Abstract

Decoding strategies play a central role in shaping the reasoning ability of large language models (LLMs). Traditional methods such as greedy decoding and beam search often suffer from error propagation, while sampling-based approaches introduce randomness without adequate robustness. Self-consistency improves reliability by aggregating multiple rollouts, but incurs significant computational overhead. We propose an entropy-guided decoding framework that introduces token-level adaptivity into generation. At each step, the model computes the entropy of the token distribution, identifies high-uncertainty positions, and selectively branches on these vulnerable points. A dynamic pool of partial rollouts is maintained and expanded until solutions are completed, concentrating computation where uncertainty is greatest and avoiding unnecessary exploration in confident regions. To enable efficient termination, we apply a rollout-level Entropy After $\langle /Think \rangle$ (EAT) stopping criterion by performing entropy evaluation after the full reasoning trace, rather than incrementally at every step. Experiments on GSM8K, AMC2023, and their perturbed variants demonstrate that our method achieves consistently strong accuracy. Notably, on smaller LLMs, performance is comparable to GPT-5 while operating at a fraction of the cost.

1 Introduction

Large language models (LLMs) have achieved remarkable performance across a wide range of tasks, including natural language understanding, summarization, code generation and scientific question answering (Bommasani et al., 2021; Chowdhery et al., 2023; OpenAI, 2023). Despite these successes, their reliability on complex reasoning tasks, such as mathematical problem solving, symbolic logic, and multi-step inference remains limited (Wanga et al., 2022). These tasks require maintaining long

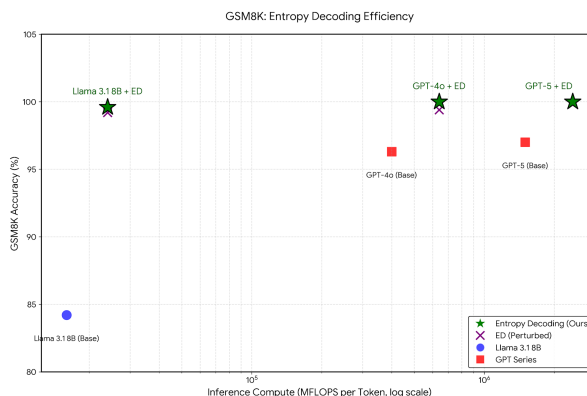


Figure 1: Performance on GSM8K, our Entropy Decoding approach enables the 8B Llama model to outperform the base GPT models with $\sim 33x$ less compute.

chains of precise intermediate steps, where a single early misstep can propagate unchecked and produce a final answer that appears logical and convincing yet is fundamentally incorrect (Boye and Moell, 2025; Zhang, 2025). In such settings, even a single erroneous token can alter the course of the reasoning process and irreversibly compromise the final outcome (Williamson et al., 2025).

A central challenge arises from the fact that standard decoding strategies, such as top- k sampling (Fan et al., 2018; Holtzman et al., 2020a) and beam search (Sutskever et al., 2014; Wiseman and Rush, 2016) are not designed to accommodate the brittleness of reasoning trajectories in large language models. Top- k sampling applies a fixed truncation at every decoding step, irrespective of whether the model is highly confident or deeply uncertain. This leads to inefficiencies: search effort is spent on unambiguous positions while decision-critical tokens receive insufficient exploration. Beam search, although maintains multiple hypotheses, expands candidates according to locally high probabilities. In reasoning tasks, such local bias can be misleading. An incorrect reasoning step may receive high probability due to patterns learned during pretraining, causing the beam to converge prematurely on

069	flawed continuations while pruning the correct but	across varying difficulty levels, while main-	119
070	lower probability trajectories (Yin et al., 2019).	taining pareto search efficiency.	120
071	Consequently, both methods lack mechanisms to		
072	adapt their search behavior to the model’s evolving	2 Preliminaries	121
073	uncertainty profile over time.		
074	Our central observation is that decision-critical	2.1 LLM Reasoning	122
075	moments (Wang and et al, 2022) in reasoning of-	Large language models (LLMs), trained on mas-	123
076	ten coincide with high-entropy tokens positions	sive and diverse text corpora, have shown surpris-	124
077	where the model’s next-token probability distribu-	ing emergent reasoning capabilities, enabling them	125
078	tion is relatively uniform. This uniformity reflects	to solve tasks requiring multi-step inference, sym-	126
079	uncertainty about which reasoning step to take next,	bolic manipulation, and compositional problem	127
080	indicating that the model’s internal knowledge is	solving. These capabilities are especially relevant	128
081	insufficient to decisively select one continuation. In	in domains such as mathematical problem solving,	129
082	contrast, low-entropy tokens, where a single choice	scientific question answering, program synthesis,	130
083	dominates the distribution, offer limited potential	and formal logical reasoning, where arriving at a	131
084	gain from alternative exploration. Identifying and	correct answer often demands a structured chain of	132
085	focusing search on high-entropy positions thus pro-	intermediate steps rather than a single direct pre-	133
086	vides a principled way to allocate computational	dition (et al., 2022; Yang et al., 2025; Pan et al.,	134
087	effort where it is most likely to affect correctness.	2023).	135
088	Based on this insight, we propose HN-decode,	Although different reasoning-oriented ap-	136
089	a high-entropy-token-guided decoding framework	proaches improve accuracy and interpretability,	137
090	that allocates search budget adaptively rather than	they often share a fundamental limitation: the	138
091	uniformly. HN-decode operates in two stages: (1)	generation of intermediate reasoning steps is	139
092	it generates an initial solution using a base decod-	typically carried out in a single forward pass	140
093	ing strategy; and (2) it scans the output to iden-	for each sampled trajectory. Errors made at	141
094	tify high-entropy positions, selectively expanding	early tokens can propagate irreversibly through	142
095	alternative continuations from these points while	subsequent steps, producing final answers that	143
096	preserving low-entropy regions. This design mim-	may appear coherent yet remain incorrect (et al.,	144
097	ics the behavior of a human problem solver who,	2022; Wang et al., 2024). Figure 2 shows one such	145
098	upon encountering uncertainty, reconsiders mul-	example.	146
099	multiple possible next steps before committing. By	Furthermore, most decoding strategies used in	147
100	concentrating exploration on decision-critical mo-	these methods apply uniform policies across all	148
101	ments, HN-decode improves robustness by reduc-	positions, without adaptively allocating search ca-	149
102	ing cascading reasoning errors, increases accuracy	capacity to decision-critical steps where model un-	150
103	on challenging tasks, and achieves more efficient	certainty is highest. As a result, they do not alter	151
104	use of search resources than conventional decoding	the underlying left-to-right decoding dynamics at	152
105	approaches.	inference time; the model still commits to tokens se-	153
106	Our contributions are threefold:	quentially without explicit mechanisms to explore	154
107		or revise uncertain regions (Creswell and Shanahan,	155
108	• We identify the limitations of uniform decod-	2022a; Welleck et al., 2022).	156
109	ing strategies in reasoning tasks and formalize	Consequently, reasoning-intensive tasks remain	157
110	the notion of decision-critical moments via	brittle, since a single incorrect inference step, such	158
111	token-wise entropy.	as a misapplied algebraic transformation, can inval-	159
112		idate the entire solution. Overcoming this brittle-	160
113	• We introduce HN-decode, a high-entropy-	ness requires inference-time methods that identify	161
114	token-guided decoding/search framework that	and revisit critical decision points rather than treat-	162
115	adaptively allocates search budget to uncer-	ing all tokens as equally reliable during generation.	163
116	tain positions, enabling targeted exploration		
117	without exhaustive expansion.	2.2 Decoding Strategies	164
118	• We demonstrate that HN-decode improves so-	Large language models (LLMs) rely critically on	165
	lution accuracy and reasoning ability on chal-	decoding strategies to convert token-level prob-	166
	lenging mathematical reasoning benchmarks	ability distributions into coherent outputs. The	167

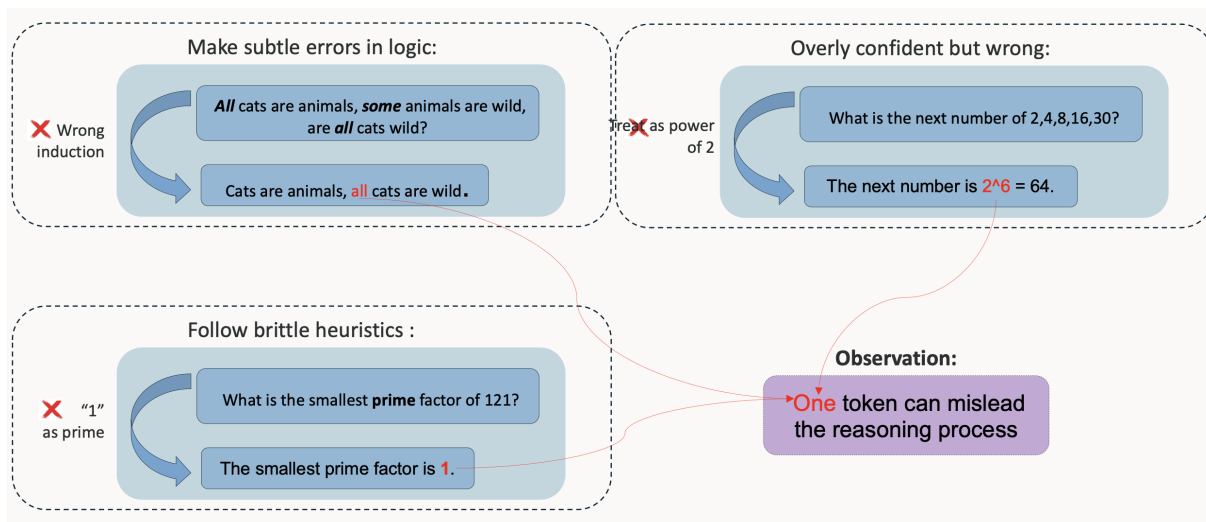


Figure 2: Different kinds of reasoning errors made by LLMs in complex tasks are due to one or a few wrong tokens which mislead the problem-solving direction.

choice of decoding method directly affects not only fluency and diversity, but also the correctness of reasoning-heavy tasks such as mathematics, programming, and scientific problem solving (Holtzman et al., 2020b; Zhu et al., 2024). Despite considerable progress, existing decoding approaches each exhibit limitations that constrain their effectiveness for complex reasoning.

Greedy decoding selects the highest-probability token at each step and is computationally efficient, but it is highly prone to error propagation: an early local mistake irreversibly derails the entire generation (Meister et al., 2020). Beam search extends greedy decoding by maintaining multiple candidate sequences, retaining the top- k partial hypotheses. While this increases exploration, beam search often collapses to high-likelihood yet uninformative or repetitive trajectories (Meister et al., 2021). Moreover, it optimizes likelihood rather than semantic or logical validity (Kasai et al., 2022), making it misaligned with reasoning objectives.

Stochastic methods such as top- k sampling and nucleus (top- p) sampling (Holtzman et al., 2020b) introduce randomness to mitigate determinism. Although these strategies improve diversity, they are agnostic to reasoning structure: a single mis-sampled token can derail multi-step reasoning chains, which is problematic in domains where correctness is fragile.

Self-consistency (Wang et al., 2023) has been proposed as a decoding-time strategy to improve reasoning robustness by sampling multiple reasoning trajectories and aggregating their final answers via majority voting. While effective, this ap-

proach is inefficient: many sampled trajectories are low-quality and consume computational resources without improving accuracy. Furthermore, self-consistency only evaluates end results and does not identify or correct intermediate reasoning errors.

Across deterministic (greedy, beam) and stochastic (sampling, self-consistency) (Brown et al., 2020; Wang et al., 2023) strategies, a shared limitation emerges: all tokens are treated as equally reliable, with no adaptivity to local uncertainty. In practice, reasoning trajectories often hinge on vulnerable decision points (e.g. in Figure 3) where the model is most uncertain or misleading (Shorinwa et al., 2025). Existing decoding methods fail to identify or revisit these points, leading either to wasted computation or overconfidence in flawed continuations (Holtzman et al., 2020a). This gap motivates entropy-based decoding approaches (Qiu et al., 2025; Fei et al., 2025; Xu et al., 2025) that explicitly target uncertainty at the token level and guide exploration toward critical decision steps.

3 Proposed Method

We introduce an entropy-guided parallel decoding framework that departs from conventional left-to-right generation. Instead of committing to a single greedy rollout or exploring via indiscriminate sampling, our approach explicitly targets uncertain tokens points in the sequence where the model’s distribution is high-entropy and prone to error. By branching at these tokens (Luo et al., 2025) and maintaining a pool of partial rollouts, the method encourages parallel reasoning and allows multiple plausible continuations to be explored efficiently.

Example: parallel thinking on vulnerable tokens

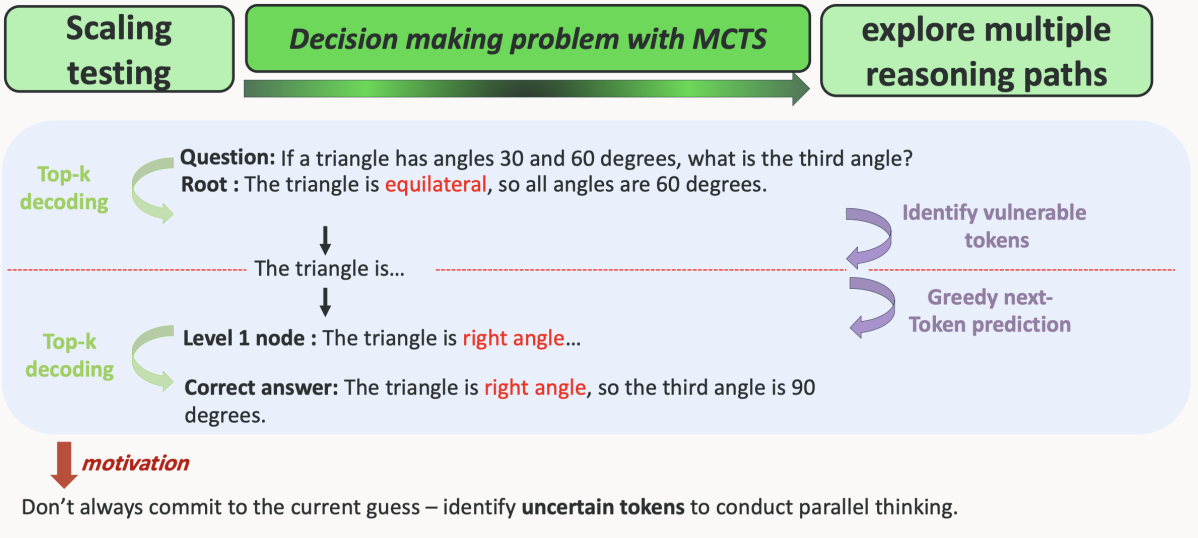


Figure 3: An example of parallel thinking on vulnerable tokens. Here, "equilateral" is the vulnerable token.

Figure 1 shows our approach efficacy and key contribution, in that, we enable small LLMs to close the performance gap to top models' at a small fraction of the cost.

3.1 Entropy-Based Token Selection

At each decoding step, the model outputs a probability distribution over the vocabulary. We compute the Shannon entropy (Shannon, 1948) of this distribution to quantify uncertainty:

$$H(p) = - \sum_i p_i \log p_i, \quad (1)$$

where p_i is the probability of the candidate token i . Tokens or positions with higher entropy indicate greater ambiguity, and are thus prime candidates for exploration. Rather than expanding every token equally, our algorithm allocates resources toward these vulnerable points, where errors are most likely to occur and corrections are most impactful.

3.2 Rollout Pool Construction

Given a rollout generated using a standard decoding procedure (e.g. Top-k decoding), when a vulnerable token is identified, the algorithm does not random sample for more rollouts. Instead, it:

- **Branches:** breaks the rollout at the top-k unconfident tokens to form a set of partial rollouts. Each partial rollouts contains tokens from the start of the rollout to the previous token of each high-entropy one.
- **Stores:** appends the corresponding partial rollouts into a job pool.

- **Continues:** generation resumes by sampling a partial rollout from the pool, greedily decoding the next token, and then extending it to completion using the standard decoding procedure (e.g. Top-k decoding).

This creates a dynamic set of reasoning paths, each diverging at critical points but reusing common low-entropy segments where the model is confident.

3.3 Parallel Thinking via Partial Rollouts

The job pool functions as a reservoir of alternative reasoning paths. Each time a candidate is sampled, greedy decoding is applied to generate the next token, in order to replace the vulnerable high-entropy one, it is rolled out to completion until a final answer is obtained. Figure 3 shows an example of parallel thinking on vulnerable tokens. Correctness checks are then applied to the finished sequence (e.g., verifying the mathematical solution). Unlike self-consistency, which samples full rollouts indiscriminately, this approach concentrates computational resources on meaningful divergences, making exploration more targeted and efficient.

3.4 Verification and Stopping Criterion

Several prior methods (Wang et al., 2025; Graves, 2016; Teerapittayanon et al., 2016; Schuster et al., 2022; Jazbec et al., 2024) employ confidence or uncertainty measures to enable early exiting during inference. The intuition is that a model's uncertainty, typically quantified as the entropy of its next-token distribution should decrease as reason-

ing progresses. A substantial drop in entropy (i.e., a high EAT score) after reasoning indicates that the model has likely reached a confident and stable conclusion.

Other lines of work have instead focused on training verifiers to assess the correctness of model generated solutions or to determine preference between candidate responses (Cobbe and et al, 2021; Creswell and Shanahan, 2022b; Bai and et al, 2022; Zheng et al., 2023; Lightman and et al, 2023; Zhang et al., 2024; Paul et al., 2024). However, with recent advances in reasoning capabilities, LLMs can increasingly self-evaluate critiquing and refining their outputs without external verifiers often through natural language prompt templates that guide self-reflection and correction (Shinn et al., 2023; Ling et al., 2023; Madaan et al., 2023; Weng et al., 2023).

In contrast, our approach leverages implicit correctness signals encoded within the model’s own internal representations of the reasoning process. We employ a rollout-level EAT criterion, meaning entropy analysis is applied after the entire reasoning sequence (rather than step by step). Specifically, after generating a full reasoning rollout, we compute the entropy statistics namely, the mean and variance of next-token entropies at all `</think>` boundaries and use these to determine whether to accept, continue, or re-roll the reasoning sequence. A high EAT value suggests that the model has converged to a confident answer, whereas low EAT indicates lingering uncertainty or inconsistency, in which case additional rollouts may improve performance.

The procedure is as follows:

- Take a generated reasoning sequence.
- Replay it token by token through the model.
- Record next-token entropy at each `</think>` boundary.
- Compute the mean (μ) and variance (σ) of these entropies.
- Accept the rollout if stability criteria are met (e.g., $\mu < \tau_1$ and $\sigma < \tau_2$), otherwise continue or re-roll. Here τ_1 and τ_2 are hyperparameters which can be optimized on the training split of the datasets.

4 Experiments

We evaluate several large language models in this study: GPT-4o (OpenAI, 2024), and LLaMA-3.1-8B (Llama Team, 2024). Each model is tested under **few-shot Chain-of-Thought (CoT)** setting (et al., 2022; Ton et al., 2025).

The prompt is for all the aforementioned LLMs are the same, and it is (we also tested variations of the prompts and observed similar results):

```
You are given a question between the tags:
<question> and </question>.

<question> {user_question} </question>

First, think about the question and provide a
step-by-step reasoning process between the
tags:
<think> ... </think>

Finally, on a new line print only the final answer: a single number with no extra text or
formatting.
```

4.1 Setup

We evaluate our entropy-guided decoding framework on two widely used reasoning benchmarks: GSM8k (grade-school math word problems) (OpenAI, 2021) and AMC2023 (Kaggle, 2023) (American Mathematics Competition problems which are substantially harder). We used Llama3.1-8B and GPT4o for this task. We used the default hyperparameters (top_k, temperature). The model is prompted with chain-of-thought reasoning, and decoding is performed with our method. We run this using the APIs ¹ and ². For each dataset, we record:

1. Accuracy: proportion of tasks solved correctly.
2. Base accuracy: proportion of tasks solved without branching.
3. Job statistics: number of rollouts (“jobs”) required before success.
4. Success job rate: ratio of successful rollouts among total generated jobs.

¹<https://console.groq.com/docs/model/llama-3.1-8b-instant>

²<https://platform.openai.com/docs/models/gpt-4o>

Model	Data	Base Accuracy	Accuracy	Mean Jobs	Max Jobs	Mean Success Job Rate	Mean Success Depth	Mean Elapsed Time (s)
Llama3.1-8B	GSM8k	84.2	99.6	1.4	23	0.993	0.997	30.2
Llama3.1-8B	AMC	45.0	97.5	1.6	21	0.786	1.21	16.9
GPT4o	GSM8k	96.3	100	1.3	19	0.987	0.996	42.2
GPT4o	AMC	75.0	98.25	1.9	12	0.759	1.33	51.0

Table 1: Results of entropy-guided decoding method on GSM8k and AMC2023 datasets. The table reports the base accuracy (%): pass@1 of the base model, accuracy (%): accuracy of the proposed method, job statistics, mean success job rate, mean success depth and average elapsed time in seconds.

5. Success depth: the depth at which branching first reaches a correct solution.

6. Elapsed time: wall-clock latency per problem.

We summarize the performance of our entropy-guided decoding method on GSM8k (OpenAI, 2021) and AMC2023 (Kaggle, 2023) datasets. Table 1 reports the accuracy, first-trial accuracy, job statistics, success job rate, and elapsed time statistics.

4.2 GSM8k Results

On GSM8K, using the Llama3.1-8B model, our method achieves 99.6% accuracy, with 84.2% of tasks solved correctly on the first trial. On average, only 1.4 attempts were required per problem, with a mean elapsed time of 30.2 seconds, and no corrupted or failed tasks were observed. These results demonstrate that our approach handles simpler arithmetic reasoning tasks extremely efficiently.

Using the GPT4o model, our method also yields a significant increase in accuracy, requiring only 42.2 seconds on average. This further confirms that our approach is efficient for simpler arithmetic reasoning tasks.

4.3 AMC2023 Results

On AMC2023, which is significantly more challenging, our method still achieves 97.5% accuracy across 40 tasks for Llama3.1-8B and 98.25% accuracy for GPT4o. However, only 45% and 75% of the problems were solved at the first trial for both models, requiring additional exploration of vulnerable tokens. For Llama3.1-8B, the maximum number of jobs required for a single task was 21, with an average of 1.6 jobs created per task. For GPT4o, the maximum number of jobs required for a single task was 12, with an average of 1.9 jobs created per task.

Despite the increased complexity, our method successfully identified correct solutions for more tasks. Notably, correctness was consistently achieved within finite and bounded exploration budgets, demonstrating the scalability of entropy-guided rollouts. The variation in success job rate (mean ≈ 0.786 for Llama3.1-8B, and mean ≈ 0.759 for GPT4o) indicates that although some branches terminate unsuccessfully, the pool-based strategy ensures sufficient coverage of correct reasoning paths.

4.4 Perturbed datasets and ablation studies

To assess model robustness on mathematical reasoning tasks, we generated larger evaluation sets following the perturbation method introduced in (Yu et al., 2024). These datasets contain modified versions of the original benchmarks and are designed to test whether models can reliably produce correct answers for different surface forms of the same underlying program. The approach in (Yu et al., 2024) first extracts executable programs from the original math datasets and validates them using the corresponding input–output pairs, ensuring they capture the reasoning necessary to solve the text problems. GPT5 model then creates new questions by replacing the input–output pairs while preserving the program structure. We manually verify the correctness of the perturbed questions and provide the answers. Examples of the resulting perturbed dataset are shown in Table 2. Using this method, we generated 2,242 perturbed questions from GSM8K and 320 from AMC2023. We evaluate LLaMA-3.1-8B (Llama Team, 2024) and GPT-4o (OpenAI, 2024). Each model is tested using its default hyperparameters (e.g., top_k, temperature) under the few-shot chain-of-thought (CoT) prompt setting (et al., 2022). We apply our method to these perturbed datasets, with results summarized in Table 3. The findings indicate that standard LLMs

Perturbed questions

Betty is saving money for a new wallet which costs \$90. Betty has only forty percent of the money she needs. Her parents decided to give her \$10 for that purpose, and her grandparents two and a half times as much as her parents. How much more money does Betty need to buy the wallet?

Julie is reading a 110-page book. Yesterday, she was able to read 5 pages and today, she read two and a half times as many pages as yesterday. If she wants to read 45% of the remaining pages tomorrow, how many pages should she read?

Tina makes \$17.50 an hour. If she works more than 9 hours per shift, she is eligible for overtime, which is paid by your hourly wage + 0.4 times your hourly wage. If she works 9 hours every day for 4 days, how much money does she make?

Randy has 55 mango trees on his farm. He also has 7 less than 0.35 times as many coconut trees as mango trees. How many trees does Randy have in all on his farm?

Table 2: Examples of perturbed questions from the GSM8K.

struggle on the perturbed datasets, whereas our method achieves stable and robust performance.

4.5 Regeneration at Random Tokens

For comparison, we also consider an alternative strategy in which a token is selected at random, and regeneration proceeds from that point using the same limits for consistency. The results of this approach are reported in the last column of Table 3.

We further evaluate performance across different generation budgets, defined as the maximum number of allowed generated tokens, to examine how much accuracy LLaMA-3.1-8B can achieve with both the proposed method and the random approach. These results are illustrated in Figure 4.

Overall, selecting the highest-entropy token consistently outperforms random token selection under the same generation budget.

4.6 Comparison with SOTA Model

We compared the LLaMA-3.1-8B model, using our entropy-based decoding, against the state-of-the-art GPT-5 model in terms of both performance and cost on perturbed datasets.

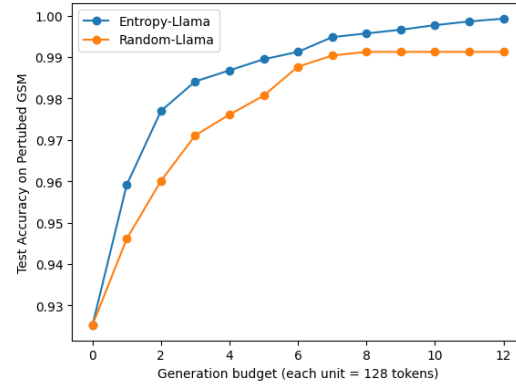


Figure 4: Test accuracy on the perturbed GSM8K dataset under different generation budgets. Each x-axis unit represents 128 tokens. Results are shown for the Llama3.1-8B model using entropy-based and random approaches.

For GPT-5, we used the hosted API³, which charges \$10 per million output tokens and \$1.25 per million input tokens. Evaluation was performed on the perturbed GSM8K and AMC2023 datasets. The distributions of input tokens, output tokens, and inference times are detailed in A.2.

For LLaMA-3.1-8B, we first assessed the cost via the hosted API⁴, priced at \$0.08 per million output tokens and \$0.05 per million input tokens. Distributions of input tokens, output tokens, and runtime are shown in A.3.

Since LLaMA-3.1-8B is open-source, we additionally ran it locally to evaluate on-premise inference costs. Full details of the setup and computations are provided in A.6.

Table 4 presents a direct comparison between GPT-5 and LLaMA-3.1-8B under our entropy-based decoding framework. Despite its smaller size, LLaMA-3.1-8B achieves comparable accuracy with GPT-5, while operating at more than an order of magnitude lower cost.

4.7 Hyperparameters and Ablation Studies

The hyperparameters in our experiments were tuned using Optuna (Akiba et al., 2019) to maximize accuracy on the GSM8K training split with LLaMA-3.1-8B. Although hyperparameters could be optimized individually for each model and dataset, we adopt a single configuration to highlight the effectiveness of our entropy-based decod-

³<https://platform.openai.com/docs/models/gpt-5>

⁴<https://console.groq.com/docs/model/llama-3.1-8b-instant>

Model	Data	Base Accuracy	Proposed Accuracy	Random Accuracy	Mean	Mean
					Elapsed Time(s) (Proposed)	Elapsed Time(s) (Random)
Llama3.1-8B	GSM8K_p	75.3	99.2	85.3	37.8	95.4
Llama3.1-8B	AMC_p	42.7	98.3	92.7	17.4	21.3
GPT4o	GSM8K_p	79.5	99.4	94.5	45.7	96.9
GPT4o	AMC_p	43.2	91.9	87.6	57.4	93.8

Table 3: Performance of LLMs on perturbed GSM8K and AMC2023 datasets. Base Accuracy refers to the percentage accuracy of the original model. Proposed Accuracy is the accuracy (%) achieved using our proposed method. Random Accuracy is the accuracy (%) obtained when regeneration is initiated from a randomly selected token. Mean Elapsed Time (Proposed) is the average time per question when regeneration uses our proposed method. Mean Elapsed Time (Random) is the average time per question when regeneration starts from a randomly chosen token.

Model	Data	Accuracy	Avg. number of input tokens	Avg. number of output tokens	cost (ϵ)
GPT5	GSM8K_p	96.6	125	508	0.52
Llama3.1-8B	GSM8K_p	99.2	3485	4323	0.05
GPT5	AMC_p	100	158	1779	1.8
Llama3.1-8B	AMC_p	98.3	3692	5285	0.06

Table 4: We compare the GPT5 model, and the LLaMA-3.1-8B model under entropy-based decoding. Performance is evaluated on the perturbed GSM8K and AMC2023 datasets. Our analysis includes accuracy, the average number of input and output tokens, and the associated API cost.

ing. The final hyperparameter values are reported in Table 7.

We also performed ablation studies by varying each hyperparameter, with results summarized in Table 8 in A.5. These results show that changes to the hyperparameters lead to only minor variations in model performance.

4.8 Discussion

The comparison between GSM8K and AMC2023, along with their perturbed versions, highlights the adaptive behavior of our framework. Unlike self-consistency, which generates a fixed number of full rollouts often resulting in redundant computation our approach dynamically allocates effort, branching only at uncertain tokens. These findings demonstrate that entropy-guided decoding achieves both efficiency and robustness: it quickly solves simpler tasks while scaling gracefully to more challenging ones by concentrating exploration on the most uncertain points.

5 Conclusions

In this work, we introduced an entropy-guided decoding framework for LLMs, aimed at overcoming the limitations of conventional strategies such as greedy search, sampling, and self-consistency.

Rather than committing prematurely to a single reasoning trajectory or generating multiple full rollouts indiscriminately, our method identifies high-entropy tokens points of uncertainty and selectively branches at these decision-critical positions. By maintaining a dynamic pool of partial rollouts, the framework enables parallel reasoning while allocating computational resources adaptively and efficiently. We also proposed a rollout-level EAT criterion, applying entropy analysis after the entire reasoning sequence, to decide whether to accept, continue, or re-roll the sequence.

Experimental results on the GSM8K and AMC2023 benchmarks, including their perturbed variants, demonstrate the robustness and effectiveness of our approach. These findings underscore the potential of uncertainty-aware decoding as a scalable and efficient approach to reasoning. By directly addressing error propagation at the token level, entropy-guided rollouts enable LLMs to solve tasks reliably and efficiently. Future work may extend this paradigm to broader domains such as code generation, scientific discovery, and multimodal reasoning, as well as explore integration with reinforcement learning signals to further refine process-level rewards. The code and perturbed datasets will be released upon acceptance.

6 Limitations

Although entropy-guided decoding improves robustness and efficiency in reasoning tasks, several limitations remain. First, the method relies on entropy as a proxy for uncertainty, which may not always correlate with true reasoning difficulty. In cases where the model is confidently wrong, such as systematic misconceptions or spurious correlations, the framework may fail to trigger branching at critical steps. Second, while adaptive branching is more efficient than generating full rollouts, the approach can still incur high computational costs on problems with many consecutive uncertain tokens. The number of spawned jobs may grow rapidly in adversarial or highly combinatorial settings, especially when no correct trajectory is easily recoverable. Third, the framework assumes access to a token-level entropy signal during inference, which may not be available or efficiently computed in all deployment environments. Integrating the method into API-restricted or latency-sensitive systems could therefore be challenging. Fourth, hyperparameter selection could be optimized using the training splits of each dataset. However, in this work, we use a single hyperparameter setting to demonstrate the effectiveness of entropy-based decoding. Finally, our evaluation focuses on mathematical reasoning benchmarks. Although the approach may generalize to domains such as code generation, scientific discovery, or multimodal reasoning, its effectiveness outside the tested datasets remains to be validated. Future work is needed to examine how domain structure, task format, and model scale affect the behavior of entropy-guided branching.

References

Takuya Akiba, Shotaro Sano, Toshihiko Yanase, Takeru Ohta, and Masanori Koyama. 2019. Optuna: A Next-Generation Hyperparameter Optimization Framework. In *Proceedings of the 25th ACM SIGKDD International Conference on Knowledge Discovery Data Mining*, pages 2623–2631.

Yuntao Bai and et al. 2022. Constitutional ai: Harmlessness from ai feedback. <https://arxiv.org/abs/2212.08073>.

Rishi Bommasani, Daniel Hudson, Ehsan Adeli, and et al. 2021. On the opportunities and risks of foundation models. *arXiv preprint arXiv:2108.07258*.

Johan Boye and Birger Moell. 2025. Large language

models and mathematical reasoning failures. *arXiv preprint arXiv:2502.11574*.

Tom B. Brown, Benjamin Mann, Nick Ryder, and et al. 2020. Language models are few-shot learners. *arXiv preprint arXiv:2005.14165*.

Aakanksha Chowdhery, Sharan Narang, Jacob Devlin, Maarten Bosma, Gaurav Mishra, Adam Roberts, Paul Barham, Hyung Won Chung, Charles Sutton, Sebastian Gehrmann, and 1 others. 2023. Palm: Scaling language modeling with pathways. *Journal of Machine Learning Research*, 24(240):1–113.

Karl Cobbe and et al. 2021. Training verifiers to solve math word problems. *arXiv preprint arXiv:2509.26522*.

Antonia Creswell and Murray Shanahan. 2022a. Faithful reasoning using large language models. *arXiv preprint arXiv:2208.14271*.

Antonia Creswell and Murray Shanahan. 2022b. Faithful reasoning using large language models. <https://arxiv.org/abs/2208.14271>.

Wei et al. 2022. Chain-of-thought prompting elicits reasoning in large language models. In *NeurIPS*.

Angela Fan, Mike Lewis, and Yann Dauphin. 2018. Hierarchical neural story generation. In *ACL*.

Yu Fei, Yasaman Razeghi, and Sameer Singh. 2025. Nudging: Inference-time alignment of llms via guided decoding. In *Proceedings of the 63rd Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, pages 12702–12739.

Alex Graves. 2016. Adaptive computation time for recurrent neural networks. *arXiv preprint arXiv:1603.08983*.

Ari Holtzman, Jan Buys, Li Du, Maxwell Forbes, and Yejin Choi. 2020a. The curious case of neural text degeneration. In *ICLR*.

Ari Holtzman, Jan Buys, Li Du, Maxwell Forbes, and Yejin Choi. 2020b. The curious case of neural text degeneration. In *International Conference on Learning Representations*.

Metod Jazbec, Alexander Timans, Tin H Veljkovic, Kaspar Sakmann, Dan Zhang, Christian A Næsseth, and Eric Nalisnick. 2024. Fast yet safe: Early-exiting with risk control. In *Advances in Neural Information Processing Systems*, volume 37, pages 129825–129854.

Kaggle. 2023. *Amc - regression final competition*.

Jungo Kasai, Keisuke Sakaguchi, Ronan Le Bras, Dragomir Radev, Yejin Choi, and Noah A. Smith. 2022. A call for clarity in beam search: How it works and when it stops. *arXiv preprint arXiv:2204.05424*.

650	Hunter Lightman and et al. 2023. Let’s verify step by step. https://arxiv.org/abs/2305.20050 .	Claude E. Shannon. 1948. A mathematical theory of communication. <i>Bell System Technical Journal</i> , 27(3):379–423.	704
651			705
652	Zhan Ling, Yunhao Fang, Xuanlin Li, Zhiao Huang, Mingu Lee, Roland Memisevic, and Hao Su. 2023. Deductive verification of chain-of-thought reasoning. <i>Advances in Neural Information Processing Systems</i> , 36:36407–36433.	Noah Shinn, Federico Cassano, Ashwin Gopinath, Karthik Narasimhan, and Shunyu Yao. 2023. Reflexion: Language agents with verbal reinforcement learning. <i>Advances in Neural Information Processing Systems</i> , 36:8634–8652.	707
653			708
654			709
655			710
656			711
657	Meta Llama Team, AI. 2024. The llama 3 herd of models .	Ola Shorinwa, Zhiting Mei, Justin Lidard, Allen Z. Ren, and Anirudha Majumdar. 2025. A survey on uncertainty quantification of large language models: Taxonomy, open research challenges, and future directions. <i>ACM Computing Surveys</i> .	712
658			713
659	GuangSheng Luo, ZhiJun Fang, JianLing Liu, and Yi-Fan Bai. 2025. Clip guided image caption decoding based on monte carlo tree search: Clip guided image caption decoding. <i>Multimedia Systems</i> .		714
660			715
661			716
662			717
663	Aman Madaan, Niket Tandon, Prakhar Gupta, Skyler Hallinan, Luyu Gao, Sarah Wiegrefe, Uri Alon, Nouha Dziri, Shrimai Prabhunoye, Yiming Yang, and 1 others. 2023. Self-refine: Iterative refinement with self-feedback. <i>Advances in Neural Information Processing Systems</i> , 36:46534–46594.	Ilya Sutskever, Oriol Vinyals, and Quoc V Le. 2014. Sequence to sequence learning with neural networks. In <i>NeurIPS</i> .	718
664			719
665			720
666			721
667			722
668			723
669	Clara Meister, Michael Forster, and Ryan Cotterell. 2021. Determinantal beam search . In <i>Proceedings of ACL</i> .	Surat Teerapittayanon, Bradley McDanel, and Hsiang-Tsung Kung. 2016. Branchynet: Fast inference via early exiting from deep neural networks. In <i>2016 23rd International Conference on Pattern Recognition (ICPR)</i> , pages 2464–2469. IEEE.	724
670			725
671			726
672	Clara Meister, Tim Vieira, and Ryan Cotterell. 2020. Best-first beam search . In <i>Proceedings of ACL</i> .	Jean-Francois Ton, Muhammad Faaiz Taufiq, and Yang Liu. 2025. Understanding chain-of-thought in llms through information theory. <i>ICML</i> .	727
673			728
674	OpenAI. 2021. Grade school math (gsm8k) dataset. https://github.com/openai/grade-school-math .	Shenzhi Wang and et al. 2022. Beyond the 80/20 rule: High-entropy minority tokens drive effective reinforcement learning for llm reasoning . <i>arXiv preprint arXiv:2506.01939</i> .	729
675			730
676			731
677	OpenAI. 2023. GPT-4 technical report. In <i>arXiv preprint arXiv:2303.08774</i> .	Xi Wang, James McInerney, Lequn Wang, and Nathan Kallus. 2025. Eat: Entropy after </think> for reasoning model early exiting . <i>arXiv preprint arXiv:2509.26522</i> .	732
678			733
679	OpenAI. 2024. Gpt-4o system card .	Xuezhi Wang, Yada Yang, and et al. 2023. Self-consistency improves chain-of-thought reasoning in language models. In <i>EMNLP</i> .	734
680	Liangming Pan, Alon Albalak, Xinyi Wang, and William Yang Wang. 2023. Logic-llm: Empowering large language models with symbolic solvers for faithful logical reasoning . <i>arXiv preprint arXiv:2305.12295</i> .		735
681			736
682			737
683			738
684			739
685	Debjit Paul, Mete Ismayilzada, Maxime Peyrard, Beatriz Borges, Antoine Bosselut, Robert West, and Boi Faltings. 2024. Refiner: Reasoning feedback on intermediate representations. In <i>Proceedings of the 18th Conference of the European Chapter of the Association for Computational Linguistics (Volume 1: Long Papers)</i> , pages 1100–1126.	Zilong Wang, Hao Zhang, Chun-Liang Li, Julian Martin Eisenschlos, Vincent Perot, Zifeng Wang, Lesly Miculicich, Yasuhisa Fujii, Jingbo Shang, Chen-Yu Lee, and Tomas Pfister. 2024. Chain-of-table: Evolving tables in the reasoning chain for table understanding . <i>arXiv preprint arXiv:2401.04398</i> .	740
686			741
687			742
688			743
689			744
690			745
691			746
692	Zexuan Qiu, Zijing Ou, Bin Wu, Jingjing Li, Aiwei Liu, and Irwin King. 2025. Entropy-based decoding for retrieval-augmented large language models. In <i>Proceedings of the 2025 Conference of the Nations of the Americas Chapter of the Association for Computational Linguistics: Human Language Technologies (Volume 1: Long Papers)</i> , pages 4616–4627.	Ruonan Wang, Runxi Wang, Yunwen Shena, Chengfeng Wua, Qinglin Zhoua, and Rohitash Chandraa. 2022. Evaluation of llms for mathematical problem solving. https://arxiv.org/pdf/2506.00309 .	747
693			748
694			749
695			750
696			751
697			752
698			753
699	Tal Schuster, Adam Fisch, Jai Gupta, Mostafa Dehghani, Dara Bahri, Vinh Tran, Yi Tay, and Donald Metzler. 2022. Confident adaptive language modeling. In <i>Advances in Neural Information Processing Systems</i> , volume 35, pages 17456–17472.	Yixuan Weng, Minjun Zhu, Fei Xia, Bin Li, Shizhu He, Shengping Liu, Bin Sun, Kang Liu, and Jun Zhao. 2023. Large language models are better reasoners with self-verification. In <i>Findings of the Association for Computational Linguistics: EMNLP 2023</i> , pages 2550–2575.	754
700			755
701			756
702			757
703			758

759 Dane A Williamson, Yangfeng Ji, and Matthew B
760 Dwyer. 2025. Syntactic blind spots: How misalign-
761 ment leads to llms’ mathematical errors. In *Proceed-*
762 *ings of The 3rd Workshop on Mathematical Natural*
763 *Language Processing (MathNLP 2025)*, pages 1–14.

764 Sam Wiseman and Alexander M Rush. 2016. Sequence-
765 to-sequence learning as beam-search optimization.
766 In *EMNLP*.

767 Jinxuan Xu, Yuqian Zhang, and Hang Liu. 2025. A
768 survey of uncertainty estimation methods on large
769 language models. <https://arxiv.org/abs/2503.00172>.

770 Yukang Yang, Declan Iain Campbell, Kaixuan Huang,
771 Mengdi Wang, Jonathan D. Cohen, and Taylor Whit-
772 tington Webb. 2025. [Emergent symbolic mechanisms](#)
773 [support abstract reasoning in large language models](#).
774 *OpenReview / ICML*.

775 Wenpeng Yin, Jessica Hay, and Dan Roth. 2019. Bench-
776 marking zero-shot text classification: Datasets, eval-
777 uation and entailment approach. In *EMNLP*.

778 Xiaodong Yu, Ben Zhou, Hao Cheng, and Dan Roth.
779 2024. Reasonagain: Using extractable symbolic pro-
780 grams to evaluate mathematical reasoning. *arXiv*
781 *preprint arXiv:2410.19056*.

782 Yunxiang Zhang, Muhammad Khalifa, Lajanugen Lo-
783 geswaran, Jaekyeom Kim, Moontae Lee, Honglak
784 Lee, and Lu Wang. 2024. Small language models
785 need strong verifiers to self-correct reasoning. In
786 *Findings of the Association for Computational Lin-*
787 *guistics ACL 2024*, pages 15637–15653.

788 Zheng Zhang. 2025. Comprehension without com-
789 petence: Architectural limits of llms in sym-
790 bolic computation and reasoning. *arXiv preprint*
791 *arXiv:2507.10624*.

792 Lianmin Zheng, Wei-Lin Chiang, Ying Sheng, Siyuan
793 Zhuang, Zhanghao Wu, Yonghao Zhuang, Zi Lin,
794 Zhuohan Li, Dacheng Li, Eric Xing, and 1 others.
795 2023. Judging llm-as-a-judge with mt-bench and
796 chatbot arena. *Advances in neural information pro-*
797 *cessing systems*, 36:46595–46623.

798 Tinghui Zhu, Kai Zhang, Jian Xie, and Yu Su. 2024.
799 [Deductive beam search: Decoding deducible ratio-](#)
800 [nale for chain-of-thought reasoning](#). *arXiv preprint*
801 *arXiv:2401.17686*.

A Appendix

A.1 Example where the base model (llama3.1-8B) fails but the proposed method succeeds

Table 5 presents a case in which the base model (Llama3.1-8B) fails, but the proposed method succeeds. Additional examples where the base model fails but our method works are provided in Table 6.

A.2 Time, input and output token distribution of the GPT5 model

We present the distributions of consumed time, input tokens, and output tokens using the GPT5 model. The results for the perturbed GSM and AMC datasets are shown in Figure 5, 8, 9, 10, 6, and 7.

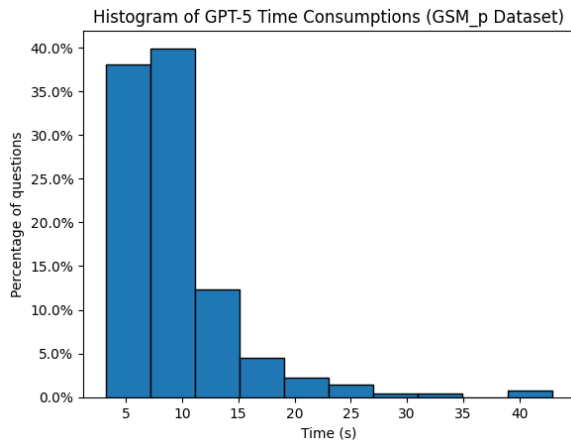


Figure 5: Consumption time (s) distribution of GPT5 model on the perturbed GSM8K dataset.

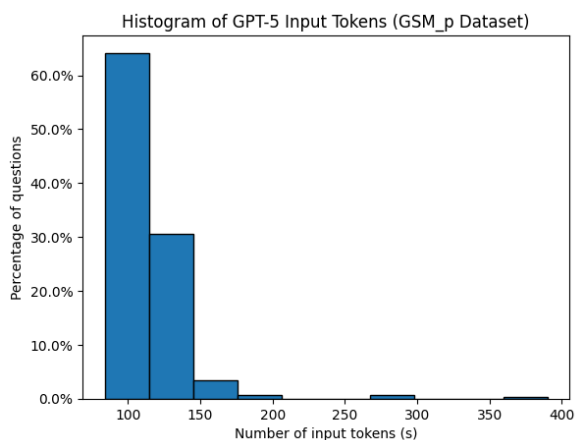


Figure 6: Distribution of the number of input tokens for the GPT5 model on the perturbed GSM8K dataset.

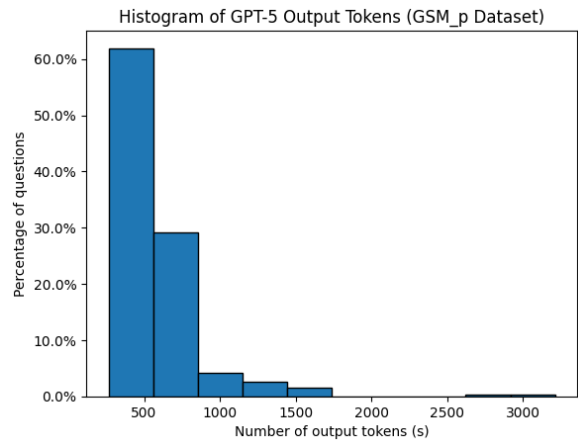


Figure 7: Distribution of the number of output tokens for the GPT5 model on the perturbed GSM8K dataset.

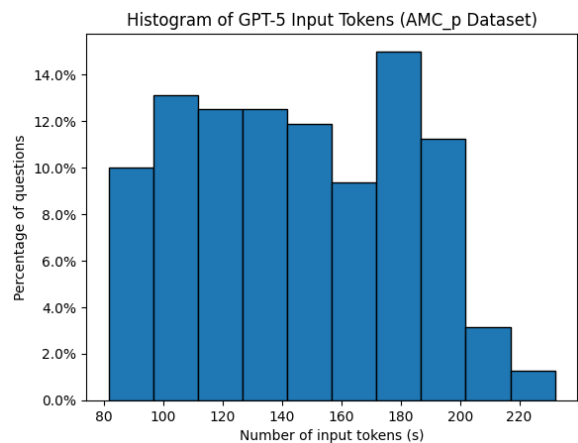


Figure 8: Distribution of the number of input tokens for the GPT5 model on the perturbed AMC dataset.

A.3 Time, input and output token distribution of the Llama3.1-8B model

We present the distributions of consumed time, input tokens, and output tokens using the Llama3.1-8B model. The results for the perturbed GSM and AMC datasets are shown in Figure 14, 15, and 16.

A.4 Parameter values

We optimized the hyperparameters using Optuna (Akiba et al., 2019), targeting the highest accuracy on the GSM8K training split using Llama3.1-8B. While dataset and model specific tuning is possible, we intentionally use one shared set of hyperparameters to demonstrate that entropy-based decoding remains effective without per-dataset optimization. The chosen values are reported in Table 7.

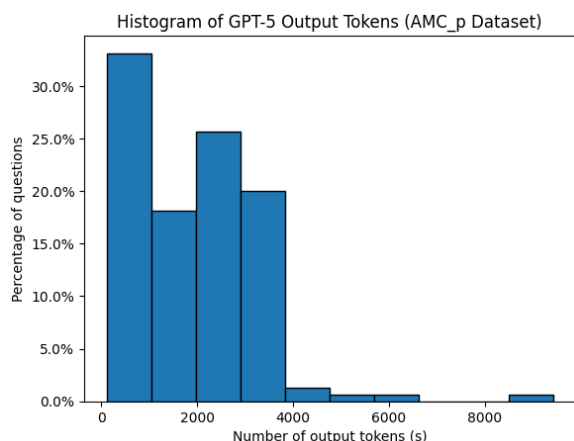


Figure 9: Distribution of the number of output tokens for the GPT5 model on the perturbed AMC dataset.

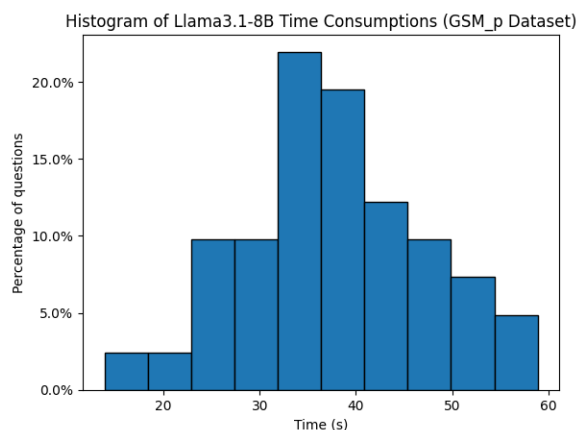


Figure 11: Consumption time (s) distribution of Llama3.1-8B model on the perturbed GSM8K dataset.

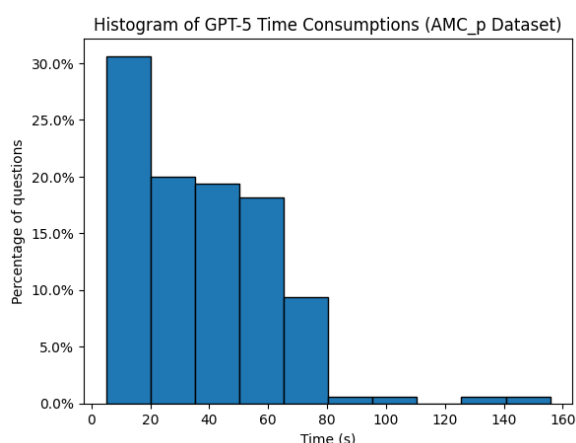


Figure 10: Consumption time (s) distribution of GPT5 model on the perturbed AMC dataset.

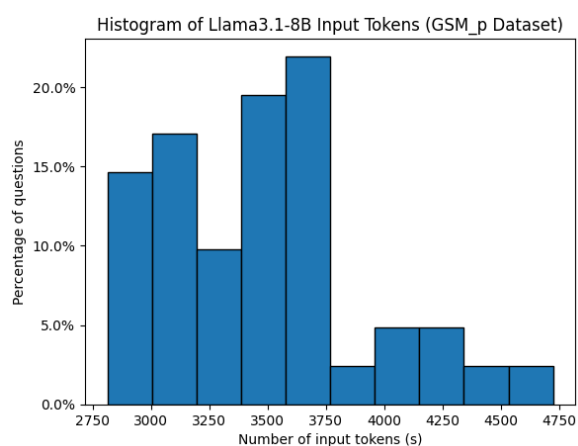


Figure 12: Distribution of the number of input tokens for the Llama3.1-8B model on the perturbed GSM8K dataset.

832 A.5 Ablation studies on parameter values

833 We independently varied each hyperparameter to
 834 analyze its effect on both accuracy and computa-
 835 tional efficiency. The results are shown in Table
 836 8.

837 A.6 Cost analysis of running LLaMA-3.1-8B 838 locally

839 Because LLaMA-3.1-8B is open-source, we addi-
 840 tionally deploy it locally to evaluate inference cost.
 841 The model is hosted on **8xA100 GPUs (40GB
 842 each)**. To estimate cost, we reference cloud GPU
 843 pricing from Lambda Labs⁵, which lists an hourly
 844 rate of **\$1.29 per A100-40GB**. Running locally re-
 845 duces end-to-end latency by approximately **90%**
 846 compared to the API. Using the average runtime
 847 per query, we multiply compute time by GPU cost
 848 to obtain per-question estimates. The resulting

cost is lower than API usage—approximately **0.03¢**
 per perturbed GSM8K question and **0.04¢** per per-
 turbed AMC2023 question.

849
 850
 851

⁵<https://lambda.ai/pricing>

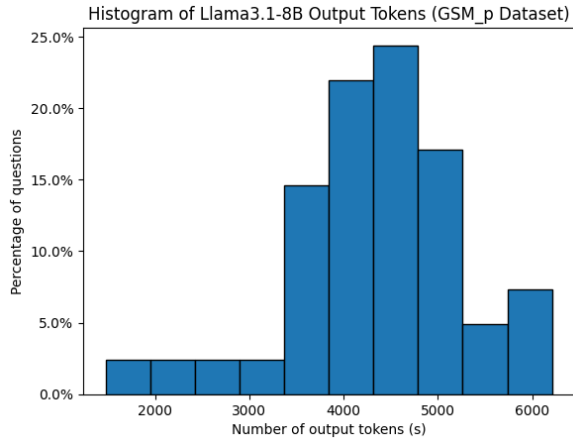


Figure 13: Distribution of the number of output tokens for the Llama3.1-8B model on the perturbed GSM8K dataset.

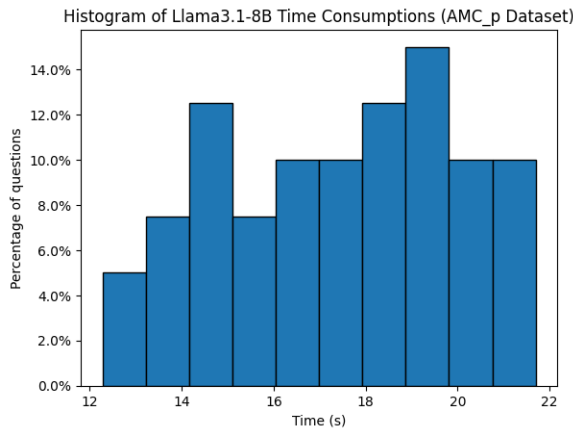


Figure 14: Consumption time (s) distribution of Llama3.1-8B model on the perturbed AMC dataset.

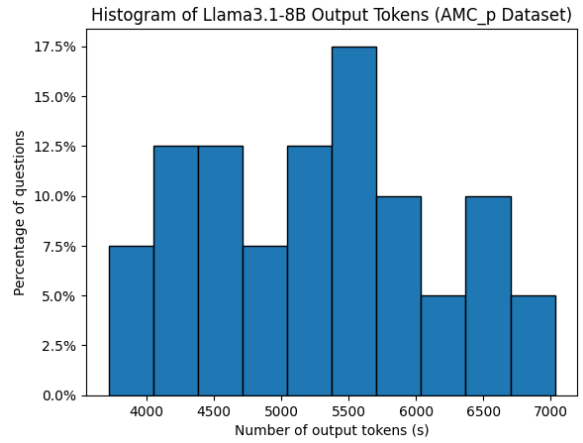


Figure 16: Distribution of the number of output tokens for the Llama3.1-8B model on the perturbed AMC dataset.

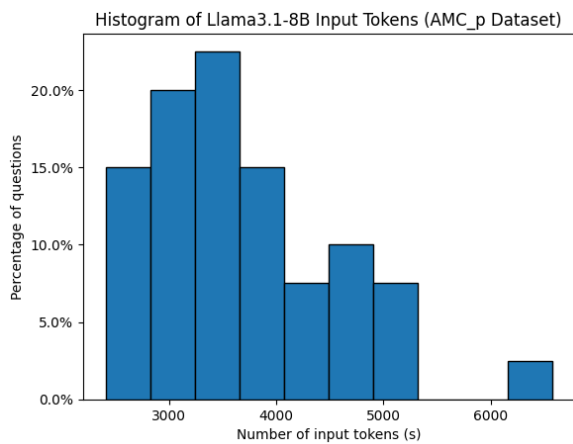


Figure 15: Distribution of the number of input tokens for the Llama3.1-8B model on the perturbed AMC dataset.

Perturbed questions	Base	Proposed
<p>Issac has to buy pens and pencils for the upcoming school year. Issac buys 108 writing utensils total. He buys 12 more than 5 times the number of pencils than pens. How many pens did Issac buy?</p>	<p>Let's define the variables: - Let x be the number of pens Issac buys. - Let y be the number of pencils Issac buys. According to the problem, he buys a total of 108 writing utensils:</p> $x + y = 108$ <p>It is also given that he buys "12 more than 5 times the number of pencils than pens". This translates to:</p> $x = 5y + 12$ <p>Now we have a system of two equations: 1. $x + y = 108$ 2. $x = 5y + 12$ Substitute the expression for x from equation 2 into equation 1:</p> $5y + 12 + y = 108$ <p>Combine the y terms:</p> $6y + 12 = 108$ <p>Subtract 12 from both sides:</p> $6y = 96$ <p>Divide by 6:</p> $y = 16$ <p>Now that we know the number of pencils, we can find the number of pens: Substitute $y = 16$ back into equation 2:</p> $x = 5(16) + 12$ $x = 80 + 12$ $x = 92$ <p>Issac buys 92 pens.</p>	<p>Let's define the variables: - Let x be the number of pens Issac buys. - Let y be the number of pencils Issac buys. According to the problem, he buys a total of 108 writing utensils:</p> $x + y = 108$ <p>It is also given that he buys "12 more than 5 times the number of pencils than pens". This translates to:</p> $y = 5x + 12$ <p>Now we have a system of two equations: 1. $x + y = 108$ 2. $y = 5x + 12$ Substitute the expression for y from equation 2 into equation 1:</p> $5x + 12 + x = 108$ <p>Combine the x terms:</p> $6x + 12 = 108$ <p>Subtract 12 from both sides:</p> $6x = 96$ <p>Divide by 6:</p> $x = 16$ <p>Issac buys 16 pens.</p>

Table 5: GSM8K example where the base model (llama3.1-8B) fails but the proposed method succeeds. x is a high entropy token.

Perturbed questions	Base	Proposed
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?	12	13
Christina is planning a birthday party and needs .75 gift bags per invited guest, because 1/4 of attendees don't show up. She invited 16 friends. Gift bags are \$2 each. How much will she spend?	18	24
Lee used to be able to run the 400-meter hurdles two seconds faster than Gerald would run the 400-meter hurdles. But Gerald changed his diet, which improved his speed by 10%. If Lee runs the 400-meter hurdles in 38 seconds, how fast can Gerald, with his improved diet, run the 400-meter hurdles, in seconds?	32.4	36
Adrien's total salary was 30 percent higher than Lylah's. Four years later, his salary had increased, and he was earning 40% more than what he was making four years ago. If Adrien's and Lylah's salary increased simultaneously, and Adrien earned \$40000 four years ago, calculate the total salary the two were receiving four years later?	96000	95200
A garden is filled with 105 flowers of various colors. There are twice as many red flowers as orange. There are five fewer yellow flowers than red. If there are 10 orange flowers, how many pink and purple flowers are there if they have the same amount and there are no other colors?	60	30
Albert wants a paintbrush that costs \$1.50, a set of paints that costs \$4.35, and a wooden easel that costs \$12.65. Albert already has \$6.50. How much more money does Albert need?	13	12
A banana tree has 100 bananas left after Raj cut some bananas from it. If Raj has eaten 70 bananas and has twice as many remaining in his basket, how many bananas were on the tree initially?	240	310
Drew is reseeding his lawn with grass seed. One bag of grass seed covers 250 square feet of lawn. His lawn is 22 feet from the house to the curb and 36 feet from side to side. He bought four bags of seed. How many extra square feet could the leftover grass seed cover after Drew reseeds his lawn?	0	208
Issac has to buy pens and pencils for the upcoming school year. Issac buys 108 writing utensils total. He buys 12 more than 5 times the number of pencils than pens. How many pens did Issac buy?	92	16

Table 6: GSM8K examples where the base model fails but the proposed method succeeds.

parameter	explanation	value
max_degree	The maximum number of entropy-heavy tokens to branch from.	3
min_degree	The minimum number of entropy-heavy tokens to branch from.	2
degree_depth_decay	How quickly branching tapers off as moving deeper.	0.6
max_mcts_depth	How deep any branch can go.	3
max_num_create_jobs	How many total branches can be explored.	32
τ_1	The mean entropy threshold for stopping	2.3
τ_2	The variance entropy threshold for stopping	9.8

Table 7: Hyperparameters used in our experiments, along with explanations and the values chosen. These values were optimized to maximize accuracy on the GSM8K train split.

Parameter	Value	Data	Accuracy	Mean Elapsed Time(s)
max_degree	2	GSM8K_p	98.1	35.2
max_degree	3	GSM8K_p	99.2	37.8
max_degree	4	GSM8K_p	98.7	36.3
max_degree	5	GSM8K_p	98.3	37.1
max_degree	2	AMC_p	97.5	18.1
max_degree	3	AMC_p	98.3	21.3
max_degree	4	AMC_p	99.1	22.7
max_degree	5	AMC_p	99.8	28.2
min_degree	1	GSM8K_p	98.5	32.9
min_degree	2	GSM8K_p	99.2	37.8
min_degree	1	AMC_p	97.8	19.7
min_degree	2	AMC_p	98.3	21.3
degree_depth_decay	0.4	GSM8K_p	98.1	36.6
degree_depth_decay	0.5	GSM8K_p	99.0	39.2
degree_depth_decay	0.6	GSM8K_p	99.2	37.8
degree_depth_decay	0.7	GSM8K_p	96.9	35.7
degree_depth_decay	0.4	AMC_p	99.2	26.7
degree_depth_decay	0.5	AMC_p	98.6	22.9
degree_depth_decay	0.6	AMC_p	98.3	21.3
degree_depth_decay	0.7	AMC_p	97.8	19.9
max_mcts_depth	2	GSM8K_p	93.5	32.4
max_mcts_depth	3	GSM8K_p	99.2	37.8
max_mcts_depthy	4	GSM8K_p	99.4	39.7
max_mcts_depth	2	AMC_p	97.9	20.2
max_mcts_depth	3	AMC_p	98.3	21.3
max_mcts_depthy	4	AMC_p	99.0	27.9
max_num_create_jobs	16	GSM8K_p	90.9	26.1
max_num_create_jobs	32	GSM8K_p	99.2	37.8
max_num_create_jobs	64	GSM8K_p	99.8	40.1
max_num_create_jobs	16	AMC_p	96.3	18.6
max_num_create_jobs	32	AMC_p	98.3	21.3
max_num_create_jobs	64	AMC_p	99.2	27.9
τ_1	2.0	GSM8K_p	98.0	36.3
τ_1	2.3	GSM8K_p	99.2	37.8
τ_1	3.0	GSM8K_p	98.8	38.1
τ_1	2.0	AMC_p	98.8	27.1.3
τ_1	2.3	AMC_p	98.3	21.3
τ_1	3.0	AMC_p	97.3	18.8
τ_2	9	GSM8K_p	98.6	39.1
τ_2	9.8	GSM8K_p	99.2	37.8
τ_2	11	GSM8K_p	97.9	36.6
τ_2	9	AMC_p	99.2	25.3
τ_2	9.8	AMC_p	98.3	21.3
τ_2	11	AMC_p	97.5	19.2

Table 8: Ablation studies on different parameter values. Performance on the perturbed GSM8K and AMC2023 dataset using Llama3.1-8B with entropy decoding. Mean Elapsed Time is the average time (s) per question when regeneration uses our proposed method.