Accelerating Speculative Reasoning with Internal Probing

Xiangchen Song[1,2*] **Saket Dingliwal**[2] **Sai Muralidhar Jayanthi**[2] **Vivek Govindan**[2]
Khiem Pham[2,3] **Shobha Vasudevan**[2] **Beidi Chen**[1,2] **Sravan Babu Bodapati**[2]
Aram Galstyan[2,4]
[1]Carnegie Mellon University [2]Amazon [3]Cornell University
[4]University of Southern California
xiangchs@cs.cmu.edu

Abstract

The widespread deployment of large reasoning models (LRMs) is hindered by the prohibitive inference costs of generating lengthy reasoning chains. To address this, speculative reasoning uses smaller and faster models to draft candidate reasoning steps and then use larger models to verify their quality. Unlike existing methods that rely on coarse verbal feedback from larger models to judge the reasoning step quality, we proposed JudgeReason which introduces a lightweight "Judge Probe" that extracts fine-grained evaluation signals directly from the target model's hidden states. This probe efficiently evaluates draft reasoning steps and facilitates a best-of-N search to improve the acceptance rate. Experimental results on the MATH and GPQA datasets demonstrate that JudgeReason significantly improves reasoning success rates while maintaining high computational efficiency, achieving a superior Pareto frontier compared to existing speculative reasoning approaches.

1 Introduction

Large language models (LLMs) have demonstrated remarkable capabilities in complex reasoning tasks [16, 6, 4]. However, deploying these large reasoning models (LRMs) faces a significant bottleneck: the substantial computational cost of generating long, multi-step reasoning chains. This high cost poses a major obstacle to their widespread adoption in resource-constrained environments.

Speculative decoding techniques [7, 2, 9] accelerate inference by using a smaller "draft" model to propose tokens that are subsequently verified by a larger "target" model. However, these token-level approaches are not optimized for the unique structure of reasoning tasks. Reasoning often involves a hierarchical process where complex problems are decomposed into semantically coherent sub-steps.

This structural property creates an opportunity for more efficient acceleration. Instead of verifying individual tokens, we can evaluate entire reasoning steps, enabling verification at a coarser, more semantically meaningful level. Recent speculative reasoning methods have explored this idea [10, 14, 3]. However, existing approaches have notable limitations. They either rely on verbal feedback elicited through dedicated prompts from the larger model [10], which incurs significant computational overhead and can be inconsistent, or depend on token-level verification with external reward models [8, 15], which often fails to capture the semantic coherence of a full reasoning step.

Inspired by recent work [1] on using hidden states to accelerate speculative decoding, we observe that an LRM's internal representations contain rich information about the quality of a reasoning step. This

*Work was done while interning at Amazon

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

motivates **JudgeReason**, a framework that probes the hidden states of the target model to generate fine-grained evaluation scores for draft reasoning steps. Our approach introduces a lightweight "Judge Probe" that operates directly on the target model's hidden states, evaluating a draft step without requiring the target model to generate any new tokens.

Our contributions are threefold. **First**, we introduce a novel speculative reasoning framework that leverages hidden state information for efficient, step-wise evaluation. **Second**, we design a lightweight and efficient judge probe architecture that provides fine-grained quality scores, enabling more accurate filtering and selection of draft steps. **Third**, we conduct comprehensive experiments on challenging reasoning benchmarks, demonstrating that JudgeReason achieves a superior trade-off between accuracy and computational efficiency compared to existing methods.

2 Related Work

Speculative Decoding. This class of methods accelerates LLM inference by using a smaller draft model to generate token sequences that are then verified in parallel by a larger target model [7, 2]. These methods require exact distribution matching between the draft and target models, a constraint that is often too strict for reasoning tasks where semantic equivalence is more critical than token-level identity.

Speculative Reasoning. Extending speculative decoding, these approaches operate at the level of reasoning steps rather than tokens. For instance, SpecReason [10] employs verbal feedback from the target model to evaluate draft steps. Other methods explore alternative verification strategies [14, 3]. However, these approaches can incur substantial overhead from generating feedback or may not fully capture the rich semantic information available within the target model's internal representations.

Probing and Model Interpretability. A large body of work has shown that a transformer's hidden states encode rich linguistic and semantic information [12, 13]. More recently, studies have demonstrated that hidden states can be used to predict model confidence and uncertainty [5]. Our work builds on these insights, proposing to extract signals directly from the target model's representations to evaluate the quality of a generated reasoning step.

3 Methodology

3.1 Problem Formulation

In speculative reasoning, we aim to accelerate inference for a large target model, M_T, by using a smaller draft model, M_D, to generate candidate reasoning steps. The key distinctions between speculative decoding and speculative reasoning lie in the properties they preserve and their verification granularity, as summarized in Table 1.

Table 1: A comparison of key aspects between speculative decoding and speculative reasoning.

Aspect	**Speculative Decoding**	**Speculative Reasoning**
Preserved Property	Conditional Probability	Reasoning Ability
Alignment Granularity	Token-level	Step-level
Verification Method	Deterministic	Probabilistic / Score-based

Given a problem x and the reasoning context generated so far, c, the draft model M_D produces a candidate reasoning step s. The core challenge is to efficiently verify whether s adheres to the reasoning quality of M_T without requiring M_T to generate its own step for a direct comparison.

3.2 The JudgeReason Framework

Our framework comprises three primary stages: (1) draft step generation, (2) hidden state extraction, and (3) judge probe evaluation. If a draft step is rejected, a fourth stage, (4) target step generation, is invoked.

Draft Step Generation. The draft model M_D generates a candidate reasoning step: $s = M_D(x \oplus c)$, where $\oplus$ denotes concatenation. To improve the acceptance rate, we can generate N candidate steps in parallel using sampling: $\{s_1, s_2, \ldots, s_N\}$, each conditioned on the shared context c.

Hidden State Extraction. We structure the generation process such that reasoning steps are delimited by a specific token sequence (e.g., '\n\n'). For each candidate step s_i, we concatenate it with the prior context c and feed the full sequence to the target model M_T in a single forward pass. We then extract the hidden state h_i from the final transformer layer at the position of the concluding delimiter token. This process efficiently yields an internal representation of how M_T perceives the draft step s_i.

Judge Probe Evaluation. The extracted hidden state h_i is passed to the Judge Probe, a lightweight model that maps the hidden state to a scalar quality score: $\text{score}_i = f_{\text{probe}}(h_i)$. The probe is implemented as a linear layer, $f_{\text{probe}}(h) = W \cdot h + b$, where $W \in \mathbb{R}^{1 \times d}$ and $b \in \mathbb{R}$, with d being the hidden dimension of M_T. A draft step is accepted if its score exceeds a predefined threshold τ:

$$\text{accept}(s_i) = \begin{cases} \text{True} & \text{if score}_i \geq \tau \\ \text{False} & \text{otherwise} \end{cases}$$

In the multi-draft case (best-of-N), we select the candidate with the highest score, provided it is above the threshold τ. If no candidate meets the threshold, we revert to the target model M_T to generate the next reasoning step autoregressively.

3.3 Data Collection and Probe Training

To train the Judge Probe, we collect data from two common reasoning benchmarks: **MATH** [4] and **GPQA** [11]. We generate training examples by running the SpecReason [10] baseline on a training split of 80 questions from each dataset. We record tuples of `(hidden_state, score)`, where the score is derived from the target model's verbal feedback (on a 0–9 scale) for a given draft step. This process yields approximately 20,000 training examples. We train the linear probe to predict these scores using a mean squared error (MSE) loss with the Adam optimizer, a learning rate of 1×10^{-4}, and a batch size of 512.

4 Experiments

Experimental Setup. We evaluate on the held-out test sets of the MATH and GPQA benchmarks, ensuring no overlap with the probe's training data. Following prior work [10], we use `deepseek-ai/DeepSeek-R1-Distill-Qwen-1.5B` as the draft model (M_D) and `Qwen/QwQ-32B` as the target model (M_T). We compare JudgeReason against three baselines: the standalone **Draft Model**, the standalone **Target Model**, **SpecReason** [10] (which uses verbal feedback), and a **Random Rejection** baseline that accepts draft steps with a fixed probability. We report three key metrics: **Pass@1** (success rate, averaged over 4 runs), **Throughput** (questions per second), and **Acceptance Rate**.

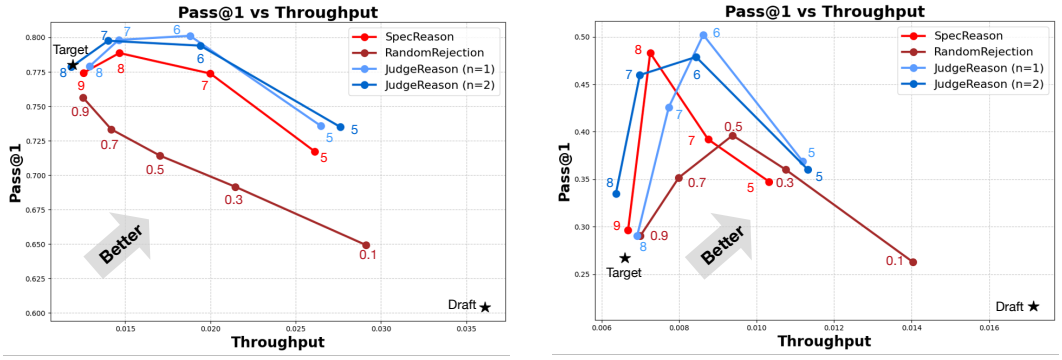
Main Results. Table 2 shows that JudgeReason consistently outperforms all baselines. For SpecReason and JudgeReason, we report results at thresholds that yield a strong Pass@1 score. On MATH, JudgeReason with a single draft step ($N = 1$) achieves the highest Pass@1 of any method, including the standalone target model. Using two draft steps ($N = 2$) further boosts throughput and the acceptance rate. On GPQA, JudgeReason ($N = 1$) achieves the best Pass@1 and throughput simultaneously. The Pareto curves in Figure 1 clearly show that JudgeReason dominates other speculative methods, offering a superior trade-off between reasoning accuracy and computational speed.

4.1 The Quality Filtering Effect

Interestingly, the highest reasoning accuracy is achieved not by the target model alone, but by JudgeReason at an intermediate acceptance threshold. This phenomenon arises from a **quality filtering effect**: the Judge Probe selectively accepts only high-confidence draft steps whose quality can, on average, exceed that of the target model's typical output. This creates a hybrid system that synergistically combines the speed of the draft model with a filtered, high-quality subset of its outputs, leading to performance that can surpass either model individually. This can also be infered from Figure 2 which illustrates the relationship between among acceptance rates, pass@1 and throughput.

Table 2: Main results on the MATH and GPQA datasets. Our method, JudgeReason, demonstrates superior performance in both reasoning accuracy (Pass@1) and efficiency (Throughput). Best results for speculative methods are in **bold**.

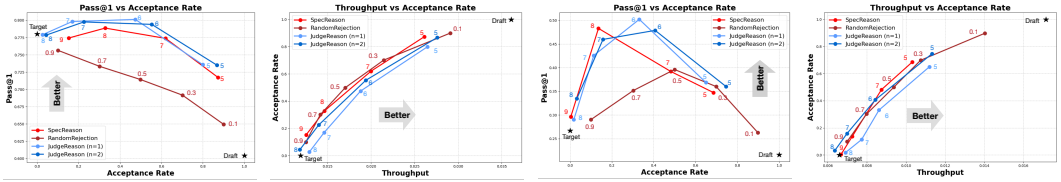
Dataset	Method	Pass@1	Throughput (q/s)	Acceptance Rate
MATH	Draft Model	0.604 ± 0.018	0.0360 ± 0.0006	1.0
	Target Model	0.780 ± 0.006	0.0119 ± 0.0002	0.0
	Random Reject ($p = 0.5$)	0.714 ± 0.014	0.0170 ± 0.0004	0.498 ± 0.002
	SpecReason	0.789 ± 0.013	0.0147 ± 0.0003	0.328 ± 0.006
	JudgeReason ($N = 1$)	$\mathbf{0.801 \pm 0.006}$	0.0188 ± 0.0002	0.474 ± 0.009
	JudgeReason ($N = 2$)	0.794 ± 0.010	$\mathbf{0.0194 \pm 0.0003}$	$\mathbf{0.553 \pm 0.007}$
GPQA	Draft Model	0.2161 ± 0.0284	0.0171 ± 0.0008	1.0
	Target Model	0.2669 ± 0.0347	0.0065 ± 0.0001	0.0
	Random Reject ($p = 0.5$)	0.3962 ± 0.0070	0.0094 ± 0.0001	0.4996 ± 0.0035
	SpecReason	0.483 ± 0.029	0.0073 ± 0.0001	0.136 ± 0.011
	JudgeReason ($N = 1$)	$\mathbf{0.502 \pm 0.025}$	$\mathbf{0.0086 \pm 0.0001}$	0.331 ± 0.006
	JudgeReason ($N = 2$)	0.479 ± 0.047	0.0084 ± 0.0001	$\mathbf{0.407 \pm 0.013}$



(a) MATH (b) GPQA

Figure 1: Pass@1 vs. Throughput. The labels are the thresholds (τ) used in each method, for Random Reject, we reported the rejected ratio instead. We can see JudgeReason (in blues) consistently defines a superior accuracy and efficiency frontier compared to baselines across both datasets.

In particular for GPQA, the significant gap between smart scoring methods such as SpecReason and JudgeReason and dummy scoring methods such as Random Reject, clearly mark the quality filtering effect. All those plots also showed how JudgeReason consistently achieves a better trade-off between accuracy and efficiency.



(a) MATH Pass@1 vs Acceptance Rate (b) MATH Throughput vs Acceptance Rate (c) GPQA Pass@1 vs Acceptance Rate (d) GPQA Throughput vs Acceptance Rate

Figure 2: Additional metric curves on MATH ((a), (b)) and GPQA ((c), (d)), showing relations among Pass@1, Acceptance Rate, and Throughput.

5 Conclusion

We introduced JudgeReason, a speculative reasoning framework that efficiently evaluates draft reasoning steps by directly probing the target model's hidden states. Our experiments demonstrate that this approach not only accelerates inference but can also improve overall reasoning accuracy

through a powerful quality filtering effect. By replacing costly verbal feedback with a lightweight and direct probe of the model's internal representations, JudgeReason achieves a new state-of-the-art Pareto frontier for accuracy and efficiency. This method offers a practical and effective path toward deploying large-scale reasoning models, with future work poised to explore probe adaptation across different model architectures and reasoning domains.

References

[1] Gregor Bachmann, Sotiris Anagnostidis, Albert Pumarola, Markos Georgopoulos, Artsiom Sanakoyeu, Yuming Du, Edgar Schönfeld, Ali Thabet, and Jonas K Kohler. Judge decoding: Faster speculative sampling requires going beyond model alignment. In *The Thirteenth International Conference on Learning Representations*, 2025.

[2] Charlie Chen, Sebastian Borgeaud, Geoffrey Irving, Jean-Baptiste Lespiau, Laurent Sifre, and John Jumper. Accelerating large language model decoding with speculative sampling. *arXiv preprint arXiv:2302.01318*, 2023.

[3] Yuanlin Chu, Bo Wang, Xiang Liu, Hong Chen, Aiwei Liu, and Xuming Hu. Ssr: Speculative parallel scaling reasoning in test-time. *arXiv preprint arXiv:2505.15340*, 2025.

[4] Dan Hendrycks, Collin Burns, Saurav Kadavath, Akul Arora, Steven Basart, Eric Tang, Dawn Song, and Jacob Steinhardt. Measuring mathematical problem solving with the math dataset. *arXiv preprint arXiv:2103.03874*, 2021.

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

[6] Takeshi Kojima, Shixiang Shane Gu, Machel Reid, Yutaka Matsuo, and Yusuke Iwasawa. Large language models are zero-shot reasoners. *Advances in neural information processing systems*, 35:22199–22213, 2022.

[7] Yaniv Leviathan, Matan Kalman, and Yossi Matias. Fast inference from transformers via speculative decoding. In *International Conference on Machine Learning*, pages 19274–19286. PMLR, 2023.

[8] Baohao Liao, Yuhui Xu, Hanze Dong, Junnan Li, Christof Monz, Silvio Savarese, Doyen Sahoo, and Caiming Xiong. Reward-guided speculative decoding for efficient llm reasoning. *arXiv preprint arXiv:2501.19324*, 2025.

[9] Xupeng Miao, Gabriele Oliaro, Zhihao Zhang, Xinhao Cheng, Zeyu Wang, Zhengxin Zhang, Rae Ying Yee Wong, Alan Zhu, Lijie Yang, Xiaoxiang Shi, et al. Specinfer: Accelerating large language model serving with tree-based speculative inference and verification. In *Proceedings of the 29th ACM International Conference on Architectural Support for Programming Languages and Operating Systems, Volume 3*, pages 932–949, 2024.

[10] Rui Pan, Yinwei Dai, Zhihao Zhang, Gabriele Oliaro, Zhihao Jia, and Ravi Netravali. Specreason: Fast and accurate inference-time compute via speculative reasoning. *arXiv preprint arXiv:2504.07891*, 2025.

[11] David Rein, Betty Li Hou, Asa Cooper Stickland, Jackson Petty, Richard Yuanzhe Pang, Julien Dirani, Julian Michael, and Samuel R Bowman. Gpqa: A graduate-level google-proof q&a benchmark. In *First Conference on Language Modeling*, 2024.

[12] Anna Rogers, Olga Kovaleva, and Anna Rumshisky. A primer in BERTology: What we know about how BERT works. *Transactions of the Association for Computational Linguistics*, 8:842–866, 2020.

[13] Ian Tenney, Dipanjan Das, and Ellie Pavlick. Bert rediscovers the classical nlp pipeline. *arXiv preprint arXiv:1905.05950*, 2019.

[14] Jikai Wang, Juntao Li, Jianye Hou, Bowen Yan, Lijun Wu, and Min Zhang. Efficient reasoning for llms through speculative chain-of-thought. *arXiv preprint arXiv:2504.19095*, 2025.

[15] Zhenglin Wang, Jialong Wu, Yilong Lai, Congzhi Zhang, and Deyu Zhou. Seed: Accelerating reasoning tree construction via scheduled speculative decoding. *arXiv preprint arXiv:2406.18200*, 2024.

[16] Jason Wei, Xuezhi Wang, Dale Schuurmans, Maarten Bosma, Fei Xia, Ed Chi, Quoc V Le, Denny Zhou, et al. Chain-of-thought prompting elicits reasoning in large language models. *Advances in neural information processing systems*, 35:24824–24837, 2022.