

# From Tokens to Steps: Verification-Aware Speculative Decoding for Efficient Multi-Step Reasoning

**Kiran Purohit\***  
IIT Kharagpur  
kiran.purohit  
@kgpian.iitkgp.ac.in

**Ramasuri Narayanam**  
Adobe Research  
rnarayanam  
@adobe.com

**Soumyabrata Pal**  
Adobe Research  
soumyabratap  
@adobe.com

## Abstract

Speculative decoding (SD) accelerates large language model inference by allowing a lightweight draft model to propose outputs that a stronger target model verifies. However, its token-centric nature allows erroneous steps to propagate. Prior approaches mitigate this using external reward models, but incur additional latency, computational overhead, and limit generalizability. We propose **SPEC-GUARD**, a verification-aware speculative decoding framework that performs *step-level* verification using only model-internal signals. At each step, SPEC-GUARD samples multiple draft candidates and selects the most consistent step, which is then validated using an ensemble of two lightweight model-internal signals: (i) an attention-based grounding score that measures attribution to the input and previously accepted steps, and (ii) a log-probability-based score that captures token-level confidence. These signals jointly determine whether a step is accepted or recomputed using the target, allocating compute selectively. Experiments across a range of reasoning benchmarks show that SPEC-GUARD improves accuracy by 3.6% while reducing latency by  $\sim 11\%$ , outperforming both SD and reward-guided SD.

## 1 Introduction

Large language models (LLMs) have demonstrated a remarkable ability to solve complex multi-step reasoning problems across domains such as mathematics and knowledge-intensive tasks (Brown et al., 2020; Team et al., 2024; Hurst et al., 2024). However, their practical deployment is constrained by high inference costs, which limit scalability and real-time applicability (Patterson et al., 2021). *Reducing inference overhead without sacrificing accuracy has therefore become a central research challenge* (Xu et al., 2024; Lin et al., 2024).

Speculative decoding (SD) (Leviathan et al., 2023) has emerged as a promising solution to accelerate inference, where a lightweight draft model generates candidate tokens, and a stronger target model verifies them. By offloading much of the token generation process to the smaller draft model, SD achieves significant latency reductions compared to decoding with the target model alone. Despite these gains, SD remains inherently token-centric, leading to critical limitations in reasoning tasks. Its strict unbiasedness requirement often rejects semantically correct draft tokens that have low probability under the target model, resulting in wasted computation and reduced efficiency (Bachmann et al., 2025; Holtzman et al., 2020). This rigidity limits speedups and makes it less effective for multi-step tasks.

Recent extensions of SD attempt to address this limitation. For example, reward-guided speculative decoding (RSD) (Liao et al., 2025) introduces external pre-trained reward models (PRMs) to verify the correctness of the draft output. Although effective in improving reliability, it incurs substantial drawbacks. First, reliance on external verifiers significantly increases latency and compute overhead. Second, pre-trained reward models are often specialized to specific tasks, making them difficult to generalize across diverse reasoning tasks.

This naturally leads to the central question driving our work: *How can we design a speculative decoding framework that maintains accuracy in multi-step reasoning tasks while remaining cost-efficient and scalable, without relying on external verifier models?*

We present **SPEC-GUARD**, a mathematically grounded, verification-aware framework for adaptive inference-time compute allocation. The key intuition behind SPEC-GUARD is:

- *Accuracy preservation*: Mitigate error propagation by ensuring trusted intermediate outputs are accepted, safeguarding correctness throughout the

\*Work done during internship at Adobe Research.

reasoning chain.

- *Efficiency*: Enable lightweight, cost-effective verification without relying on large external verifiers, reducing latency.

SPECGUARD integrates two lightweight verifiers derived directly from the model itself: (i) *Attention-based grounding verification*, which checks whether the generated step is properly grounded in the input context or previously validated steps, and (ii) *Log-probability-based verification*, which ensures confidence at the token level. These complementary signals are combined into an *ensemble verifier* that adaptively decides whether to accept draft outputs or invoke the target model. Furthermore, we introduce a novel *self-consistency selector* that identifies the most semantically consistent step from multiple sampled draft candidates. To summarize, our key contributions are:

1. We propose SPECGUARD, a novel framework that integrates model-internal verifiers with adaptive inference-time compute allocation, improving reliability without the need for external reward models.
2. We introduce a novel *self-consistency selector* that identifies the representative step from multiple sampled draft candidates.
3. Extensive experiments on various reasoning benchmarks show that SPECGUARD improves accuracy by up to 3.6% while reducing latency by  $\sim 11\%$  compared to state-of-the-art methods, making it both effective and efficient for real-world deployment.

## 2 Related Work

**Speculative Decoding.** Speculative decoding accelerates inference by letting a lightweight draft model propose tokens that a larger target model verifies in parallel (Leviathan et al., 2023; Li et al., 2024; Chen et al., 2024c, 2023; Zhang et al., 2024; Stern et al., 2018; Xia et al., 2024; Sun et al., 2024). Variants include tree-based speculation (Chen et al., 2024b; Sun et al., 2023; Fu et al., 2024; Miao et al., 2024) to increase acceptance, self-speculative decoding that leverages parts of the base model (Zhang et al., 2024; Elhoushi et al., 2024), and CTC-based drafting (Wen et al., 2024) to improve sequence quality. Methods like LayerSkip (Elhoushi et al., 2024) and Draft-on-the-Fly (Metel et al., 2024) further explore adaptive or early-exit strategies. SpecReason (Pan et al., 2025) performs

speculative reasoning with the target model as a critic that scores semantic utility. SPECGUARD, in contrast, combines multi-sample self-consistency with an ensemble verifier, enabling stronger filtering of plausible-but-ungrounded steps. RSD (Liao et al., 2025) incorporates process reward models (PRMs) to guide speculative reasoning at the step level. SPECGUARD differs by keeping the standard draft-target pipeline but replacing external verifiers with lightweight, model-internal signals for step-level evaluation.

**Reward Models on Reasoning.** Reward models are used to provide feedback for choosing the correct reasoning path (Zhou et al., 2025; Wang et al., 2024; Chen et al., 2024a). Outcome reward models (ORMs) (Dong et al., 2024; Yu et al., 2024) score final answers, while process reward models (PRMs) (Lightman et al., 2023) assess intermediate steps. Advances in reward models have increased focus on scaling test-time compute (Snell et al., 2024), enabling strategies like Best-of-N sampling (Dong et al., 2023; Cobbe et al., 2021; Brown et al., 2024), tree search (Yao et al., 2023; Qi et al., 2024; Chen et al., 2024a), and reward-guided inference such as RSD (Liao et al., 2025) or SPECS (Cemri et al., 2025). These improve reasoning quality but add latency and reliance on external verifiers. In contrast, SPECGUARD leverages an ensemble of internal confidence and grounding signals, avoiding external PRMs while improving multi-step reasoning accuracy.

## 3 Our Proposed Approach

In this section, we present SPECGUARD, a novel speculative decoding framework leveraging inference-time compute. It employs an ensemble verifier combining attention-based grounding (Section 3.1) and probability-based signals (Section 3.2). This formulation enables efficient reasoning without reliance on external verifiers while maintaining interpretability and robustness. We then describe how our approach combines the inference time compute with the ensemble-guided acceptance criteria (Section 3.3). Figure 1 outlines the high-level architectural overview of our proposed framework.

### 3.1 Attention-Based Grounding Verification

We introduce Attention-Based Grounding Verification (ABGV) as a mechanism to assess whether each output token (i.e. full reasoning step in our

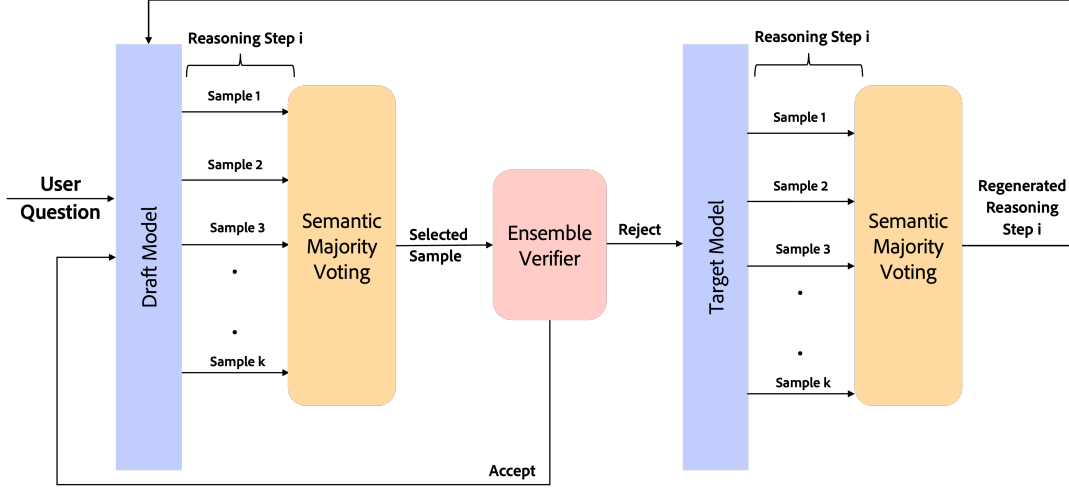


Figure 1: Architectural overview of the SPECGUARD framework

scenario) generated by a language model is sufficiently grounded in the input context or the previously generated steps. Unlike approaches that rely on external verifiers or auxiliary models, ABGV directly leverages the internal attention matrices of the model itself, enabling efficient and scalable verification. The key intuition is that a correctly grounded output should exhibit strong attention alignment with the most relevant input tokens or validated prior steps, thereby reflecting faithful attribution rather than spurious correlations.

Let an input prompt be denoted as  $x$ , and the language model generates an output sequence  $y_i = (y_{i,1}, y_{i,2}, \dots, y_{i,T})$  at each step  $i$ . At each step, the model produces multilayer multihead attention matrices for each layer  $l$  and head  $h$ :  $A^{(l,h)} \in \mathbb{R}^{(t_{\text{input}} + t_{\text{output}}) \times (t_{\text{input}} + t_{\text{output}})}$ .

To compute cumulative attribution from input tokens to an output token, we use the well-known *attention rollout* mechanism, which recursively multiplies attention matrices across layers to show the total influence of each input token on the final output. Formally, let  $A^{(l)}$  denote the attention matrix averaged over the heads in layer  $l$ . The rollout matrix  $R$  is computed as:  $R = A^{(L)} A^{(L-1)} \dots A^{(1)}$ .

For each output token  $y_{i,t}$ , the distribution over the input tokens is given by the row  $R_{y_{i,t}}$  of the rollout matrix, normalized to sum to 1. Let  $\mathcal{I}$  denote the set of input tokens (including prior reasoning steps). The grounding score for token  $y_{i,t}$  is defined as:

$$G(y_{i,t}) = \sum_{j \in \mathcal{I}} R_{y_{i,t}}[j]$$

Here,  $R_{y_{i,t}}[j]$  denotes the attention weight of the token  $y_{i,t}$  to the input token  $j$ . A higher ground-

ing score indicates a stronger reliance on the input context. We adopt a stricter criterion by taking the minimum token grounding score across the reasoning step  $y_i$ :  $G_{\text{min-step}} = \min_t G(y_{i,t})$ , which ensures that every token in the generated reasoning step  $y_i$  is sufficiently grounded, thereby preventing ungrounded tokens from being masked by averaging.

**Memory-Efficient Design:** A naive implementation would require storing attention matrices from all layers, which could become memory-intensive for larger models and longer outputs. To ensure practical scalability, ABGV employs two lightweight design choices:

- **Layer subset:** we store attention matrices from the last 3 layers, which we find sufficient for grounding quality (Figure 3 shows minimal loss in performance).
- **Head sparsification:** we discard entries below 0.01 in attention heads, reducing memory footprint with negligible effect on grounding fidelity (see Figure 3).

### 3.2 Log Probability-Based Verification

We introduce log-probability-based verification (LPBV) as a complementary mechanism to assess the reliability of the full reasoning step generated by a language model. LPBV relies on the model’s own predictive confidence, as reflected in the conditional log probability of the tokens generated. The key intuition is that faithfully generated and reliable output should be assigned a relatively high log probability under the model’s next-token distribution, while unreliable tokens are often associated

with predictions of low probability. For each token  $y_{i,t}$ , of the reasoning step  $y_i$ , the model produces a conditional probability given the input  $x$  and the prior steps:  $p(y_{i,t} | x, y_{i,<t})$ , from which we compute the logarithmic probability score:

$$L(y_{i,t}) = \log p(y_{i,t} | x, y_{i,<t})$$

At each step level, a stricter criterion is applied by taking the minimum log probability across tokens, ensuring that no token is assigned disproportionately low confidence:  $L_{\min\text{-step}} = \min_t L(y_{i,t})$ .

### 3.3 SPEC-GUARD: Verification-Aware Speculative Decoding

We propose SPEC-GUARD, an ensemble verifier-guided speculative decoding framework that augments speculative decoding with principled verification at the step level. At each reasoning step, SPEC-GUARD evaluates draft outputs using two lightweight, model-internal signals: (i) *Log Probability-Based Verification (LPBV)*, which enforces token-level confidence by measuring predictive likelihoods, and (ii) *Attention-Based Grounding Verification (ABGV)*, which ensures that every generated token is properly attributed to the input or previously validated steps via attention rollout. These complementary criteria are combined into a unified ensemble score that carries **formal guarantees**: only steps that are simultaneously confident and grounded are accepted, while uncertain steps trigger recomputation with the target model. In doing so, SPEC-GUARD mitigates error cascades (common in SD), thus improving reasoning reliability while preserving efficiency.

At step  $i$ , SPEC-GUARD proceeds as follows:

1. **Generate Draft Step:** The draft model  $m$ , samples  $k$  candidate steps  $\{\hat{y}_i^{(1)}, \dots, \hat{y}_i^{(k)}\}$  conditioned on the input prompt and previously accepted steps. To identify the most consistent candidate from these  $k$  possibilities, we propose the *self-consistency selector* (see Section 3.4), which selects the step  $\hat{y}_i^{j^*}$  that is maximally consistent with the other  $k - 1$  candidates.
2. **Compute Verification Scores:** For the selected step, the ensemble verifier computes both the logarithmic probability-based score  $L(\hat{y}_i^{j^*})$  and the grounding score  $G(\hat{y}_i^{j^*})$ . Before aggregation, both scores are scaled to a

---

#### Algorithm 1: SPEC-GUARD

---

```

1 Input: Prompt  $x$ , draft model  $m$ , target model  $M$ ,
   log prob function  $L(\cdot)$ , grounding score function
    $G(\cdot)$ , log prob range  $[\ell_{\min}, \ell_{\max}]$ , grounding range
    $[g_{\min}, g_{\max}]$ , weight  $\beta$ , acceptance threshold  $\tau$ ,
   EOS token  $s$ , max length  $N$ , samples per step  $k$ 
2 Output: Response  $y_{1:i}$ 
3 Initialize  $y_{1:0} \leftarrow ""$ 
4 for  $i \leftarrow 1$  to  $N - 1$  do
5   Sample  $k$  draft candidates
      $\{\hat{y}_i^{(1)}, \dots, \hat{y}_i^{(k)}\} \leftarrow m(x, y_{1:i-1})$ 
6   Select draft step
      $\hat{y}_i^{j^*} \leftarrow \text{Self-Consistency Selector}(\{\hat{y}_i^{(j)}\}_{j=1}^k)$ 
7   Compute min log prob  $\ell_i \leftarrow L(\hat{y}_i^{j^*})$ 
8   Compute min grounding score  $g_i \leftarrow G(\hat{y}_i^{j^*})$ 
9   Normalize:  $\tilde{\ell}_i = \frac{\ell_i - \ell_{\min}}{\ell_{\max} - \ell_{\min}}$ ,  $\tilde{g}_i = \frac{g_i - g_{\min}}{g_{\max} - g_{\min}}$ 
10  Compute ensemble verifier score:
      $r_i \leftarrow \beta \cdot \tilde{\ell}_i + (1 - \beta) \cdot \tilde{g}_i$ 
11  if  $r_i \geq \tau$  then
12    | Accept draft step  $y_i \leftarrow \hat{y}_i^{j^*}$ 
13  else
14    | Sample  $k$  target candidates
15    |  $\{y_i^{(1)}, \dots, y_i^{(k)}\} \leftarrow M(x, y_{1:i-1})$ 
16    | Select target step  $y_i^{j^*} \leftarrow$ 
17    |   Self-Consistency Selector( $\{y_i^{(j)}\}_{j=1}^k$ )
18    |  $y_i \leftarrow y_i^{j^*}$ 
19  if  $s \in y_i$  then
20    | break

```

---

comparable range using min-max normalization.

3. **Apply Acceptance Criterion:** The ensemble verifier combines normalized scores through a weighted aggregation to determine acceptance. If the criterion is satisfied,  $\hat{y}_i^{j^*}$  is accepted; otherwise, the target model  $M$  is invoked to sample  $k$  candidate steps  $\{y_i^{(1)}, \dots, y_i^{(k)}\}$  to reduce stochastic variance and improve reliability. Since even the target model may occasionally produce inconsistent reasoning, *Self-consistency selector* is again applied to select the most consistent step  $y_i^{j^*}$ .
4. **Repeat Until Termination:** This process continues until the model generates an end-of-sequence (EOS) token or the sequence reaches the maximum length  $N$ .

Algorithm 1 outlines the key steps involved in the proposed approach SPEC-GUARD. Analysis of the computational complexity of SPEC-GUARD is provided in Appendix A.3.

---

**Algorithm 2: Self-Consistency Selector**

---

**Input:** Set of  $k$  candidates  $\{y^{(1)}, \dots, y^{(k)}\}$ , sentence transformer model  $\mathcal{E}$

**Output:** Index  $j^*$  of the selected candidate

- 1 Compute embeddings for all candidates:  
 $e^{(j)} \leftarrow \mathcal{E}(y^{(j)})$  for  $j = 1 \dots k$
  - 2 Normalize embeddings so that  $\|e^{(j)}\|_2 = 1$
  - 3 Compute pairwise similarity matrix:  
 $S_{ij} \leftarrow \langle e^{(i)}, e^{(j)} \rangle$  for  $i, j = 1 \dots k$
  - 4 Apply row-wise softmax:  $\tilde{S}_{ij} = \frac{\exp(S_{ij})}{\sum_{l=1}^k \exp(S_{il})}$
  - 5 Extract diagonal scores:  $d_j \leftarrow \tilde{S}_{jj}$  for  $j = 1 \dots k$
  - 6  $j^* \leftarrow \arg \min_j d_j$ ; // Select most semantically consistent candidate
- 

### 3.4 Self-Consistency Selector to Identify Consistent Candidate

To identify the most consistent reasoning step among a set of  $k$  sampled candidates (either by draft or target), we propose the *self-consistency selector*, based on (Zhu et al., 2025). The underlying intuition is that a consistent candidate should exhibit strong agreement with the other candidates, rather than being an outlier. Formally, each candidate  $y^{(j)}$  is encoded in a normalized embedding  $e^{(j)}$  using a pre-trained sentence transformer  $\mathcal{E}$ . While embeddings could, in principle, be obtained from the model’s final hidden layer, vLLM does not expose intermediate states, so we compute them externally. The (cosine) similarities are then calculated in pairs between the candidates, yielding a similarity matrix  $S \in \mathbb{R}^{k \times k}$ , which is further normalized row-wise using softmax to obtain  $\tilde{S}$ . For each candidate  $y^{(j)}$ , we calculate its self-alignment score  $d_j = \tilde{S}_{jj}$ , which measures the degree to which the candidate aligns with itself relative to the others. Candidates that are semantically consistent with the rest of the set distribute their similarity mass across multiple peers, producing a low  $d_j$ , while outliers or less consistent candidates concentrate the similarity primarily on themselves, resulting in a high  $d_j$ . Thus, candidates with lower  $d_j$  are more representative of the set, and our approach selects the candidate with the minimum self-alignment score:  $j^* \leftarrow \arg \min_j d_j$ . Algorithm 2 describes our novel self-consistency selector.

### 3.5 Formal Guarantees

We now present formal guarantees for the proposed SPEC GUARD algorithm.

**Lemma 1** (Soundness Guarantee). *Let  $\mathcal{C}$  denote the set of correct reasoning steps,  $\tilde{\ell}_i$*

*be the logarithmic probability signal, and  $\tilde{g}_i$  be the attention-grounding signal. For any  $\alpha \in [0, 1]$ ,  $\epsilon_\ell \in [0, 1]$  and  $\epsilon_g \in [0, 1]$ , assume that  $\Pr[\tilde{\ell}_i \geq \alpha \mid y_i \in \mathcal{C}] \geq 1 - \epsilon_\ell$ ,  $\Pr[\tilde{g}_i \geq \alpha \mid y_i \in \mathcal{C}] \geq 1 - \epsilon_g$ . Then,  $\Pr[V(y_i) = \text{accept} \mid y_i \in \mathcal{C}] \geq 1 - (\epsilon_\ell + \epsilon_g)$ .*

*Proof.* Both  $\tilde{\ell}_i$  and  $\tilde{g}_i$  independently provide high probability acceptance for correct steps. Since the ensemble score satisfies  $r_i \geq \min(\tilde{\ell}_i, \tilde{g}_i)$ , the probability of rejection is bounded by the union of individual error events. Then, the total error probability is at most  $\epsilon_\ell + \epsilon_g$  and thus the lemma follows.  $\square$

**Lemma 2** (Efficiency Guarantee). *Let  $\pi_i = \Pr[V(y_i) = \text{accept}]$ . Then the expected no. of target calls  $C_T$ , is  $\mathbb{E}[C_T] = \sum_{i=1}^T (1 - \pi_i)$ . If  $\pi_i \geq \pi_{\min}$  for all  $i$ , then  $\mathbb{E}[C_T] \leq T \cdot (1 - \pi_{\min})$ .*

*Proof.* At each step  $i$ , a target call is required if and only if  $V(y_i) = \text{reject}$ . Thus, the expectation is  $\sum_i (1 - \pi_i)$ . If  $\pi_i \geq \pi_{\min}$ , the sum is bounded by  $T(1 - \pi_{\min})$ . This formalizes that higher acceptance rates directly reduce expected target calls.  $\square$

**Theorem 1** (Accuracy–Efficiency Trade-off). *Suppose correct steps satisfy Lemma 1 and the incorrect steps are rejected with probability at least  $1 - \delta$ . Then, for any sequence of length  $T$ ,  $\Pr[\text{all accepted steps are correct}] \geq (1 - \epsilon_\ell - \epsilon_g)^T \cdot (1 - \delta)^{C_T}$ .*

*Proof.* By Lemma 1, the probability of accepting only the correct steps is at least  $(1 - \epsilon_\ell - \epsilon_g)^T$ . By assumption, incorrect steps are rejected with probability at least  $1 - \delta$ , and there are  $C_T$  target calls. Thus, the lower bound of the joint probability is  $(1 - \epsilon_\ell - \epsilon_g)^T (1 - \delta)^{C_T}$ .  $\square$

These results show that SPEC GUARD provides multiplicative accuracy guarantees while bounding the expected number of target calls.

## 4 Experiments

We address the following research questions:

**RQ I.** Does SPEC GUARD provide measurable accuracy improvements on multi-step reasoning benchmarks compared to state-of-the-art methods, while mitigating error cascades?

**RQ II.** How does the number of sampled candidates per reasoning step influence both the reliability and stability of SPEC GUARD under the ensemble verification criterion?

Table 1: Accuracy on reasoning benchmarks.

Method	Target Model	Draft Model	MATH500	GSM8K	Gaokao 2023 En	Olympiad Bench
<b>Math Model, Draft and Target: Qwen2.5-Math-Instruct</b>						
Target Model	7B	-	83.0	94.7	66.8	40.6
Target-only Majority	7B	-	84.9	95.2	68.8	41.0
Draft-only Majority	-	1.5B	79.0	88.9	67.9	40.9
Best-of- $N$	-	1.5B	82.2	93.3	67.4	40.7
SD	7B	1.5B	82.4	94.2	66.3	39.4
RSD	7B	1.5B	82.4	94.4	68.5	39.6
RSD Majority	7B	1.5B	78.0	88.7	64.9	38.7
SC + LPBV	7B	1.5B	83.2	94.5	67.5	39.7
SPECGUARD GREEDY	7B	1.5B	83.6	95.6	68.8	40.7
SPECGUARD	7B	1.5B	<b>85.4</b>	<b>95.8</b>	<b>69.4</b>	<b>41.2</b>
<b>General Model, Draft and Target: Qwen2.5-Instruct</b>						
Target Model	7B	-	74.8	91.7	64.9	38.8
Draft-only Majority	-	1.5B	66.4	82.1	56.9	28.7
Best-of- $N$	-	1.5B	73.4	89.7	60.5	32.7
SD	7B	1.5B	74.8	91.6	63.1	37.1
RSD	7B	1.5B	71.4	90.1	60.5	37.6
RSD Majority	7B	1.5B	60.6	77.0	55.3	31.7
SPECGUARD GREEDY	7B	1.5B	74.9	92.0	65.5	37.8
SPECGUARD	7B	1.5B	<b>77.0</b>	<b>93.0</b>	<b>66.0</b>	<b>40.3</b>
<b>General Model, Draft: Llama-3.2-Instruct and Target: Llama-3.1-Instruct</b>						
Target Model	8B	-	48.2	83.9	40.8	14.5
Draft-only Majority	-	1B	38.0	60.2	32.2	9.5
Best-of- $N$	-	1B	48.6	74.8	40.7	14.4
SD	8B	1B	47.0	83.4	40.1	16.1
RSD	8B	1B	50.0	83.9	41.8	15.7
RSD Majority	8B	1B	36.6	61.9	30.6	12.3
SPECGUARD GREEDY	8B	1B	50.0	84.5	41.9	16.9
SPECGUARD	8B	1B	<b>51.6</b>	<b>85.1</b>	<b>43.9</b>	<b>17.2</b>

**RQ III.** Can SPECGUARD reduce inference latency compared to RSD while preserving, or even enhancing, accuracy guarantees?

#### 4.1 Experimental Setup

**Datasets and Metrics:** We conduct extensive experiments on datasets that require complex reasoning, including MATH500 (Hendrycks et al., 2021), GSM8K (Cobbe et al., 2021), GaoKao-2023-En (Liao et al., 2024), and OlympiadBench (He et al., 2024). For evaluation, we adopt the official metrics, i.e., *exact match* (EM). Detailed descriptions of the datasets can be found in Appendix A.1.

**Models:** To evaluate SPECGUARD’s effectiveness, we consider both general-purpose and math-focused LLMs, namely Qwen-2.5-Math (Yang et al., 2024), Qwen-2.5 (Qwen et al., 2025), and Llama-3 (Dubey et al., 2024). For RSD, we adopt Skywork-o1-OpenPRM (o1 Team, 2024) as the PRM.

**Baselines:** We evaluate SPECGUARD against four categories of baselines: (1) *Target model only*: the target model is used independently, which generally incurs a higher computational cost compared to SPECGUARD. (2) *Draft model with or without PRM*: This group covers inference time compute techniques that aim to maximize the performance of the draft model. Specifically, we evaluate majority voting and Best-of- $N$  (BoN) (Brown et al., 2024; Cobbe et al., 2021), where BoN selects the highest scoring response (last step) among  $N$  candidates using a PRM; beam search (Chen et al., 2024a), which employs a PRM to identify the optimal decoding trajectory; and we process Best-of- $N$ , which samples  $N$  candidate steps and chooses the one with the highest reward. (3) *Speculative decoding (SD)*: We also include speculative decoding with 7 speculative tokens for accelerating inference (Leviathan et al., 2023). (4) *RSD*: (Liao et al., 2025) leverages a PRM to score intermediate steps and

Table 2: Comparison with search-based methods on Qwen2.5-Math-Instruct. Beam Search and Process Best-of- $N$  use a 1.5B base model and a 1.5B PRM.

Method	Setting	MATH500	GSM8K
Draft Model (1.5B)	-	73.8	85.0
Process Best-of- $N$	$N = 8$	75.8	87.8
Process Best-of- $N$	$N = 16$	76.0	87.9
Beam Search	bs=4	78.2	88.9
Beam Search	bs=8	78.2	88.4
RSD (1.5B/7B/1.5B)	-	82.4	94.4
SPECGUARD GREEDY	-	83.6	95.6
SPECGUARD	maj@16	<b>85.4</b>	<b>95.8</b>

adaptively determine when to call target model.

**Setting:** We perform all experiments on NVIDIA A100 GPUs with vLLM as the serving backend. We define a reasoning step as a generation terminated by  $\backslash n \backslash n$ . For generating multiple samples, we set  $temperature = 0.7$ ,  $top\_p = 0.8$ , and  $n = 16$ . SPECGUARD refers to this multi-sample setting combined with our self-consistency selector, which chooses the most representative reasoning step. RSD Majority likewise employs this multi-sample setting, with each step scored by a PRM and the highest-scoring candidate selected. In the greedy setting where  $temperature = 0$ ,  $top\_p = 1$ , and  $n = 1$ , we refer our approach as SPECGUARD GREEDY. For both RSD and SPECGUARD, we set the threshold parameter to  $\tau = 0.7$  and we set  $\beta = 0.3$  for SPECGUARD. For details about hyperparameters refer to Appendix A.2.2. Unless stated otherwise, all models used are Qwen2.5-Math-Instruct.

## 4.2 Performance Comparison

To address **RQ1**, we evaluate SPECGUARD on a broad set of challenging reasoning benchmarks, in Table 1, and make the following observations: (1) Inference-time compute strategies such as majority voting and Best-of- $N$ , which rely on extensive draft sampling, typically underperform compared to the accuracy of a single target model. This shows the critical role of larger models in reasoning tasks, as their capabilities cannot be readily matched by smaller models even with increased computation. (2) While target-only majority voting may match SPECGUARD in accuracy, it incurs substantially higher computational cost as every reasoning step must be sampled multiple times from the target, contrary to our objective of reducing target calls. (3) Although speculative decoding (SD) is theoretically unbiased, matching target model accuracy, it often performs worse

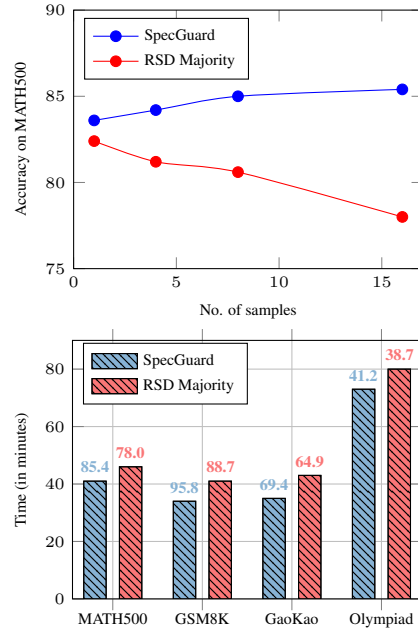


Figure 2: (Top) Varying number of samples. (Bottom) Runtime comparison (y-axis) RSD Majority vs SPECGUARD with corresponding accuracy indicated on top of bars.

in practice, primarily due to floating-point errors (Chen et al., 2023). Furthermore, when the draft model surpasses the target model, the strict unbiased nature of SD can actually degrade performance relative to the draft. (4) Reward-guided speculative decoding (RSD) addresses this by using a process reward model (PRM) to evaluate draft reasoning steps, but reliance on an external verifier adds latency and computational overhead. (5) SPECGUARD uses lightweight model-internal verifiers instead of PRMs to evaluate draft steps, and consistently outperforms both the target model and RSD across all benchmarks, demonstrating strong efficiency and performance. While LPBV captures confidence, it lacks grounding, allowing confident yet ungrounded steps to pass. In contrast, SPECGUARD achieves higher accuracy, showing ABGV is crucial for rejecting plausible but ungrounded steps. Appendix A.2.3 presents **qualitative analysis** showing that PRM often assigns high scores to incorrect draft steps, underscoring the *need for stronger verification that ensures both step-wise soundness and final-answer correctness*.

## 4.3 Comparison with Search-Based Approaches

We also compare SPECGUARD with beam search (Chen et al., 2024a) and process Best-of- $N$ , in Ta-

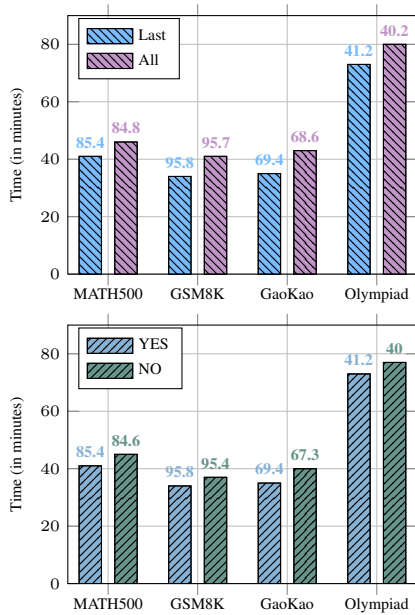


Figure 3: Runtime comparison (y-axis) with corresponding **accuracy indicated on top of bars**. (Top) changing layers. (Bottom) Sparsity in attention heads by SPEC-GUARD.

ble 2. Our method consistently outperforms both baselines. These findings reveal that when reasoning steps become complex, search-based techniques face limitations, as combinatorial growth of candidate solutions makes it difficult to reliably identify optimal paths, resulting in degraded performance. In contrast, SPEC-GUARD leverages the power of larger models to generate strong candidate solutions. In addition, the incorporation of an ensemble verifier provides step-level feedback, mitigating the challenges of difficult tasks. *This highlights that moving beyond purely search-based strategies and augmenting larger models with lightweight feedback mechanisms can deliver both higher efficiency and stronger performance, when the search space is vast or the reasoning task is highly complex.*

#### 4.4 Sample Size and Time Comparison

To investigate **RQ II**, we evaluate SPEC-GUARD under varying sample sizes per reasoning step. In Figure 2, accuracy steadily improves as more diverse candidates are explored, with gains saturating at higher counts. In contrast, RSD Majority exhibits diminishing returns and even degrades performance with larger samples, due to accumulated noise from PRM. This demonstrates that SPEC-GUARD takes advantage of additional candidate generations more effectively.

For **RQ III**, Figure 2 compares the runtime of RSD Majority and SPEC-GUARD, with the accuracy indicated on top of the bars. SPEC-GUARD consistently achieves both higher accuracy and lower latency. For example, on GSM8K, SPEC-GUARD achieves an accuracy of 95.8% in 34 minutes, compared to the accuracy of the RSD Majority 88.7% in more than 41 minutes. Together, these results confirm that our ensemble verifier-guided speculative decoding framework improves reasoning reliability while delivering superior efficiency.

#### 4.5 Ablation Studies

We perform ablation studies to analyze key design choices in SPEC-GUARD, shown in Figure 3.

**Changing Layers:** Figure 3 compares using attention from the last three layers versus aggregating across all layers. Although the latter yields marginal gains on some benchmarks, it adds runtime overhead. Leveraging only the last three layers strikes a better balance, achieving higher accuracy with lower latency. We show other variants in Appendix A.2.1.

**Sparsity in Attention Heads:** We discard entries below 0.01 in attention heads. Figure 3 shows that enforcing sparsity improves both accuracy and runtime, sharpening the verifier’s focus on relevant attention patterns and enhancing efficiency without performance loss.

### 5 Conclusion and Future Work

In this work, we propose SPEC-GUARD, an adaptive speculative decoding that improves both efficiency and accuracy in multistep reasoning. By leveraging lightweight model-internal signals for verification, along with a self-consistency selector that identifies representative reasoning step across samples, SPEC-GUARD avoids dependence on external reward models and achieves higher accuracy with reduced latency compared to state-of-the-art methods. In future, we plan to extend SPEC-GUARD by incorporating additional internal signals such as entropy-based measures and uncertainty calibration to refine verifier reliability. Another promising direction is applying SPEC-GUARD to domains beyond text reasoning, including code generation and multimodal tasks.

## 6 Limitations

Our work primarily focuses on improving inference-time efficiency and robustness for multi-step reasoning through verification-aware speculative decoding. While the proposed framework demonstrates consistent gains across a variety of reasoning benchmarks, we acknowledge several limitations related to the scope and evaluation of the current study.

Our experiments are conducted on a set of reasoning benchmarks that emphasize structured, step-by-step inference. Although these benchmarks cover a wide range of reasoning patterns, we do not explicitly evaluate performance on open-ended generation tasks or domains such as long-form dialogue and creative writing, where the notion of step-level verification may manifest differently. Extending the evaluation to such settings remains an interesting direction for future work.

Our analysis focuses on single-instance inference and does not explicitly address deployment aspects such as large-scale batching or hardware-specific optimizations. While our method reduces latency at the per-instance level, additional engineering considerations may be required to fully realize these benefits in production-scale systems. We leave these extensions to future work.

## 7 Ethical Considerations

Our approach relies on large language models for multi-step reasoning, it inherits known risks associated with LLMs, including hallucinated or factually incorrect outputs (Ji et al., 2023). In real-world deployments, such incorrect reasoning traces or final answers could be mistakenly trusted by users, potentially leading to the dissemination of misleading or incorrect information (Zhang et al., 2023; Albrecht et al., 2022). While our method aims to improve reasoning reliability by selecting the most consistent reasoning step, it does not eliminate the possibility of hallucinations, and the resulting systems should not be considered fully reliable without human oversight. The proposed approach is therefore best suited for research settings or as a supporting component in applications where appropriate validation mechanisms are in place.

Finally, our method does not rely on or introduce any private or personally identifiable data. All experiments are conducted using publicly available benchmarks. Although the underlying language models may have been pre-trained on large-scale

corpora that could contain sensitive information, our method does not explicitly or implicitly solicit such content.

## References

- Joshua Albrecht, Ellie Kitanidis, and Abraham J. Fetterman. 2022. [Despite "super-human" performance, current llms are unsuited for decisions about ethics and safety](#). *Preprint*, arXiv:2212.06295.
- G. Bachmann, S. Anagnostidis, A. Pumarola, M. Georgopoulos, A. Sanakoyeu, Y. Du, E. Schönfeld, A.K. Thabet, and J. Kohler. 2025. Judge decoding: Faster speculative sampling requires going beyond model alignment. In *13th International Conference on Learning Representations (ICLR)*.
- Bradley Brown, Jordan Juravsky, Ryan Ehrlich, Ronald Clark, Quoc V Le, Christopher Ré, and Azalia Mirhoseini. 2024. Large language models: Scaling inference compute with repeated sampling. *arXiv preprint arXiv:2407.21787*.
- Tom Brown, Benjamin Mann, Nick Ryder, Melanie Subbiah, Jared D Kaplan, Prafulla Dhariwal, Arvind Neelakantan, Pranav Shyam, Girish Sastry, Amanda Askell, and 1 others. 2020. Language models are few-shot learners. *Advances in neural information processing systems*, 33:1877–1901.
- Mert Cemri, Nived Rajaraman, Rishabh Tiwari, Xiaoxuan Liu, Kurt Keutzer, Ion Stoica, Kannan Ramchandran, Ahmad Beirami, and Ziteng Sun. 2025. Specs: Faster test-time scaling through speculative drafts. In *ES-FoMo III: 3rd Workshop on Efficient Systems for Foundation Models*.
- Charlie Chen, Sebastian Borgeaud, Geoffrey Irving, Jean-Baptiste Lespiau, Laurent Sifre, and John Jumper. 2023. Accelerating large language model decoding with speculative sampling. *arXiv preprint arXiv:2302.01318*.
- Guoxin Chen, Minpeng Liao, Chengxi Li, and Kai Fan. 2024a. Alphamath almost zero: process supervision without process. *Advances in Neural Information Processing Systems*, 37:27689–27724.
- Zhuoming Chen, Avner May, Ruslan Svirschevski, Yuhsun Huang, Max Ryabinin, Zhihao Jia, and Beidi Chen. 2024b. Sequoia: Scalable, robust, and hardware-aware speculative decoding. *arXiv preprint arXiv:2402.12374*.
- Ziyi Chen, Xiaocong Yang, Jiacheng Lin, Chenkai Sun, Kevin Chang, and Jie Huang. 2024c. Cascade speculative drafting for even faster llm inference. *Advances in Neural Information Processing Systems*, 37:86226–86242.
- Karl Cobbe, Vineet Kosaraju, Mohammad Bavarian, Mark Chen, Heewoo Jun, Lukasz Kaiser, Matthias Plappert, Jerry Tworek, Jacob Hilton, Reiichiro

- Nakano, and 1 others. 2021. Training verifiers to solve math word problems. *arXiv preprint arXiv:2110.14168*.
- Hanze Dong, Wei Xiong, Deepanshu Goyal, Yihan Zhang, Winnie Chow, Rui Pan, Shizhe Diao, Jipeng Zhang, Kashun Shum, and Tong Zhang. 2023. Raft: Reward ranked finetuning for generative foundation model alignment. *Transactions on Machine Learning Research*, 2023.
- Hanze Dong, Wei Xiong, Bo Pang, Haoxiang Wang, Han Zhao, Yingbo Zhou, Nan Jiang, Doyen Sahoo, Caiming Xiong, and Tong Zhang. 2024. Rlhf workflow: From reward modeling to online rlhf a comprehensive practical alignment recipe of iterative preference learning. *Transactions on Machine Learning Research*, 2024.
- Abhimanyu Dubey, Abhinav Jauhri, Abhinav Pandey, Abhishek Kadian, Ahmad Al-Dahle, Aiesha Letman, Akhil Mathur, Alan Schelten, Amy Yang, Angela Fan, and 1 others. 2024. The llama 3 herd of models. *arXiv e-prints*, pages arXiv–2407.
- Mostafa Elhoushi, Akshat Shrivastava, Diana Liskovich, Basil Hosmer, Bram Wasti, Liangzhen Lai, Anas Mahmoud, Bilge Acun, Saurabh Agarwal, Ahmed Roman, and 1 others. 2024. Layerskip: Enabling early exit inference and self-speculative decoding. In *Proceedings of the 62nd Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, pages 12622–12642.
- Yichao Fu, Peter Bailis, Ion Stoica, and Hao Zhang. 2024. Break the sequential dependency of llm inference using lookahead decoding. In *Proceedings of the 41st International Conference on Machine Learning*, pages 14060–14079.
- Chaoqun He, Renjie Luo, Yuzhuo Bai, Shengding Hu, Zhen Thai, Junhao Shen, Jinyi Hu, Xu Han, Yujie Huang, Yuxiang Zhang, and 1 others. 2024. Olympiadbench: A challenging benchmark for promoting agi with olympiad-level bilingual multimodal scientific problems. In *Proceedings of the 62nd Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, pages 3828–3850.
- Dan Hendrycks, Collin Burns, Saurav Kadavath, Akul Arora, Steven Basart, Eric Tang, Dawn Song, and Jacob Steinhardt. 2021. Measuring mathematical problem solving with the math dataset. In *Thirty-fifth Conference on Neural Information Processing Systems Datasets and Benchmarks Track (Round 2)*.
- Ari Holtzman, Jan Buys, Li Du, Maxwell Forbes, and Yejin Choi. 2020. The curious case of neural text degeneration. In *International Conference on Learning Representations*.
- Aaron Hurst, Adam Lerer, Adam P Goucher, Adam Perelman, Aditya Ramesh, Aidan Clark, AJ Ostrow, Akila Welihinda, Alan Hayes, Alec Radford, and 1 others. 2024. Gpt-4o system card. *arXiv preprint arXiv:2410.21276*.
- Ziwei Ji, Nayeon Lee, Rita Frieske, Tiezheng Yu, Dan Su, Yan Xu, Etsuko Ishii, Ye Jin Bang, Andrea Madotto, and Pascale Fung. 2023. *Survey of hallucination in natural language generation*. *ACM Computing Surveys*, 55(12):1–38.
- Yaniv Leviathan, Matan Kalman, and Yossi Matias. 2023. Fast inference from transformers via speculative decoding. In *International Conference on Machine Learning*, pages 19274–19286. PMLR.
- Yuhui Li, Fangyun Wei, Chao Zhang, and Hongyang Zhang. 2024. Eagle: Speculative sampling requires rethinking feature uncertainty. In *International Conference on Machine Learning*, pages 28935–28948. PMLR.
- Baohao Liao, Yuhui Xu, Hanze Dong, Junnan Li, Christof Monz, Silvio Savarese, Doyen Sahoo, and Caiming Xiong. 2025. Reward-guided speculative decoding for efficient llm reasoning. In *Forty-second International Conference on Machine Learning (ICML)*.
- Minpeng Liao, Chengxi Li, Wei Luo, Wu Jing, and Kai Fan. 2024. Mario: Math reasoning with code interpreter output—a reproducible pipeline. In *Findings of the Association for Computational Linguistics ACL 2024*, pages 905–924.
- Hunter Lightman, Vineet Kosaraju, Yuri Burda, Harrison Edwards, Bowen Baker, Teddy Lee, Jan Leike, John Schulman, Ilya Sutskever, and Karl Cobbe. 2023. Let’s verify step by step. In *The Twelfth International Conference on Learning Representations*.
- Ji Lin, Jiaming Tang, Haotian Tang, Shang Yang, Weiming Chen, Wei-Chen Wang, Guangxuan Xiao, Xingyu Dang, Chuang Gan, and Song Han. 2024. Awq: Activation-aware weight quantization for on-device llm compression and acceleration. *Proceedings of machine learning and systems*, 6:87–100.
- Michael Metel, Peng Lu, Boxing Chen, Mehdi Rezagholizadeh, and Ivan Kobzyev. 2024. Draft on the fly: Adaptive self-speculative decoding using cosine similarity. In *Findings of the Association for Computational Linguistics: EMNLP 2024*, pages 2267–2272.
- Xupeng Miao, Gabriele Oliaro, Zhihao Zhang, Xinhao Cheng, Zeyu Wang, Zhengxin Zhang, Rae Ying Yee Wong, Alan Zhu, Lijie Yang, Xiaoxiang Shi, and 1 others. 2024. 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.
- S. o1 Team. 2024. *Skywork-o1 open series*.
- Rui Pan, Yinwei Dai, Zhihao Zhang, Gabriele Oliaro, Zhihao Jia, and Ravi Netravali. 2025. Specreason: Fast and accurate inference-time compute via speculative reasoning. *arXiv preprint arXiv:2504.07891*.

- David Patterson, Joseph Gonzalez, Quoc Le, Chen Liang, Lluís-Miquel Munguia, Daniel Rothchild, David So, Maud Texier, and Jeff Dean. 2021. Carbon emissions and large neural network training. *arXiv preprint arXiv:2104.10350*.
- Zhenting Qi, MA Mingyuan, Jiahang Xu, Li Lyna Zhang, Fan Yang, and Mao Yang. 2024. Mutual reasoning makes smaller llms stronger problem-solver. In *The Thirteenth International Conference on Learning Representations*.
- Qwen, :, An Yang, Baosong Yang, Beichen Zhang, Binyuan Hui, Bo Zheng, Bowen Yu, Chengyuan Li, Dayiheng Liu, Fei Huang, Haoran Wei, Huan Lin, and Jian Yang et al. 2025. [Qwen2.5 technical report](#). Preprint, arXiv:2412.15115.
- Charlie Snell, Jaehoon Lee, Kelvin Xu, and Aviral Kumar. 2024. Scaling llm test-time compute optimally can be more effective than scaling model parameters. *arXiv preprint arXiv:2408.03314*.
- Mitchell Stern, Noam Shazeer, and Jakob Uszkoreit. 2018. Blockwise parallel decoding for deep autoregressive models. *Advances in Neural Information Processing Systems*, 31.
- Hanshi Sun, Zhuoming Chen, Xinyu Yang, Yuandong Tian, and Beidi Chen. 2024. Triforce: Lossless acceleration of long sequence generation with hierarchical speculative decoding. *CoRR*.
- Ziteng Sun, Ananda Theertha Suresh, Jae Hun Ro, Ahmad Beirami, Himanshu Jain, and Felix Yu. 2023. Spectr: Fast speculative decoding via optimal transport. *Advances in Neural Information Processing Systems*, 36:30222–30242.
- Gemini Team, Petko Georgiev, Ving Ian Lei, Ryan Burnell, Libin Bai, Anmol Gulati, Garrett Tanzer, Damien Vincent, Zhufeng Pan, Shibo Wang, and 1 others. 2024. Gemini 1.5: Unlocking multimodal understanding across millions of tokens of context. *arXiv preprint arXiv:2403.05530*.
- Peiyi Wang, Lei Li, Zhihong Shao, Runxin Xu, Damai Dai, Yifei Li, Deli Chen, Yu Wu, and Zhifang Sui. 2024. Math-shepherd: Verify and reinforce llms step-by-step without human annotations. In *Proceedings of the 62nd Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, pages 9426–9439.
- Zhuofan Wen, Shangtong Gui, and Yang Feng. 2024. Speculative decoding with ctc-based draft model for llm inference acceleration. *Advances in Neural Information Processing Systems*, 37:92082–92100.
- Heming Xia, Zhe Yang, Qingxiu Dong, Peiyi Wang, Yongqi Li, Tao Ge, Tianyu Liu, Wenjie Li, and Zhifang Sui. 2024. Unlocking efficiency in large language model inference: A comprehensive survey of speculative decoding. In *Findings of the Association for Computational Linguistics ACL 2024*, pages 7655–7671.
- Yuhui Xu, Zhanming Jie, Hanze Dong, Lei Wang, Xudong Lu, Aojun Zhou, Amrita Saha, Caiming Xiong, and Doyen Sahoo. 2024. Think: Thinner key cache by query-driven pruning. In *The Thirteenth International Conference on Learning Representations*.
- An Yang, Beichen Zhang, Binyuan Hui, Bofei Gao, Bowen Yu, Chengpeng Li, Dayiheng Liu, Jianhong Tu, Jingren Zhou, Junyang Lin, and 1 others. 2024. Qwen2. 5-math technical report: Toward mathematical expert model via self-improvement. *arXiv preprint arXiv:2409.12122*.
- Shunyu Yao, Dian Yu, Jeffrey Zhao, Izhak Shafran, Tom Griffiths, Yuan Cao, and Karthik Narasimhan. 2023. Tree of thoughts: Deliberate problem solving with large language models. *Advances in neural information processing systems*, 36:11809–11822.
- Fei Yu, Anningzhe Gao, and Benyou Wang. 2024. Ovm, outcome-supervised value models for planning in mathematical reasoning. In *Findings of the Association for Computational Linguistics: NAACL 2024*, pages 858–875.
- Jianyi Zhang, Xu Ji, Zhangchi Zhao, Xiali Hei, and Kim-Kwang Raymond Choo. 2023. [Ethical considerations and policy implications for large language models: Guiding responsible development and deployment](#). Preprint, arXiv:2308.02678.
- Jun Zhang, Jue Wang, Huan Li, Lidan Shou, Ke Chen, Gang Chen, and Sharad Mehrotra. 2024. Draft& verify: Lossless large language model acceleration via self-speculative decoding. In *Proceedings of the 62nd Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, pages 11263–11282.
- Jin Peng Zhou, Kaiwen Wang, Jonathan D Chang, Zhaolin Gao, Nathan Kallus, Kilian Q Weinberger, Kianté Brantley, and Wen Sun. 2025. q#: Provably optimal distributional rl for llm post-training. *CoRR*.
- Y. Zhu, H. Zhang, B. Wu, J. Li, Z. Zheng, P. Zhao, P. Chen, and Y. Bian. 2025. Measuring diversity in synthetic datasets. In *Forty-Second International Conference on Machine Learning*.

## A Appendix

### A.1 Datasets Description

An overview of the dataset statistics and examples are shown in Table 3.

**MATH500:** A benchmark subset curated from the MATH dataset, consisting of 500 competition-level mathematics problems spanning algebra, geometry, combinatorics, number theory, and probability. Each problem is accompanied by a detailed step-by-step solution, requiring multi-hop symbolic and logical reasoning. We use the full 500 problems as the evaluation set.

**GSM8K:** A dataset of linguistically diverse grade-school math word problems designed to test multi-step numerical reasoning. It comprises 8.5K questions, with a test set of 1,319 problems. Each question includes annotated solutions with intermediate steps, encouraging models to demonstrate faithful reasoning chains.

**Gaokao-2023-En:** Derived from the English-translated 2023 Gaokao (China’s national college entrance exam), this dataset contains high-school level math word problems with a strong emphasis on reasoning over algebra, functions, and applied mathematics. It poses particular challenges due to its formal problem style and complex solution trajectories. The evaluation set includes 385 problems.

**OlympiadBench:** A large-scale benchmark of problems drawn from global mathematics and science Olympiads, covering topics such as advanced algebra, geometry, physics, and logical reasoning. The problems are highly challenging, requiring creative multi-step reasoning far beyond routine computation. We evaluate on the test split of 675 questions.

### A.2 Additional Experiments

#### A.2.1 Changing Layers

We perform ablation studies to analyze key design choices in SPECGUARD, particularly the selection of layers used to extract internal grounding signals. Figure 4 compares four settings: using attention from (1) all layers, (2) the last three layers, (3) the middle three layers, and (4) the first three layers. The results show that the first three layers perform noticeably worse than other variants, while the middle three layers achieve higher accuracy but still lag behind the deeper layers. Both the last three layers and all layers yield strong performance, but using

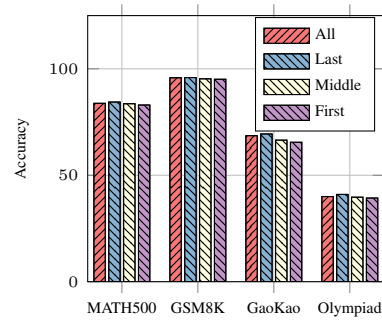


Figure 4: Changing Layers

all layers incurs higher runtime overhead. Overall, the last three layers provide the best trade-off, delivering strong accuracy with lower latency.

#### A.2.2 Tuning of $\beta$ and $\tau$

We analyze the sensitivity of our approach to two hyperparameters: the step acceptance threshold  $\tau$  for our ensemble verifier, and the weighting factor  $\beta$ , which balances the log-probability and attention grounding score when computing the ensemble score. As shown in Table 4, accuracy remains stable across different values of  $\beta$ , with  $\beta = 0.3$  performing slightly better than higher values, suggesting that a moderate weighting strikes a good balance between model confidence and grounding. Table 5 reports results for varying  $\tau$ . We find that  $\tau = 0.7$  achieves the most consistent gains across datasets, while both lower ( $\tau = 0.6$ ) and higher values ( $\tau = 0.8, 0.9$ ) lead to small drops. Overall, our method is robust to hyperparameter choices, with  $\beta = 0.3$  and  $\tau = 0.7$  serving as effective defaults across tasks.

#### A.2.3 Qualitative Analysis

Table 6, 7 presents a qualitative example of reasoning steps scored by the PRM. Each intermediate step receives a high score, leading the verifier to accept the draft-generated reasoning without intervention. However, despite this consistent acceptance, the reasoning chain ultimately produces an incorrect final answer. This illustrates a key limitation of relying solely on PRM scores: while they may capture local plausibility of individual steps, they do not guarantee global correctness of the overall solution. Such cases highlight the need for more robust evaluation mechanisms that can account for consistency across steps as well as correctness of the final outcome.

### A.3 Complexity Analysis - SPEC GUARD

We compare the computational complexity of Speculative Decoding (SD), Reward-guided Speculative Decoding (RSD), and SPEC GUARD.

Let:

- $T$  = number of reasoning steps,
- $k$  = number of draft candidates sampled per step,
- $d$  = hidden dimension of embeddings,
- $H$  = number of attention heads,
- $L$  = number of layers used in ABGV,
- $C_{\text{draft}}, C_{\text{target}}$  = per-step cost of draft and target models.

**Speculative Decoding (SD).** Complexity

$$= O(T \cdot (k \cdot C_{\text{draft}} + (1 - \pi) \cdot k \cdot C_{\text{target}})),$$

where  $\pi$  is the acceptance probability.

**Reward-guided Speculative Decoding (RSD).**

In addition to SD cost, RSD requires a pretrained reward model (PRM) verifier:

Complexity =

$$O(T \cdot (k \cdot C_{\text{draft}} + (1 - \pi) \cdot k \cdot C_{\text{target}} + C_{\text{PRM}})).$$

**SPEC GUARD.** It includes draft sampling, self-consistency selector, and ensemble verification.

$$\text{Complexity} = O\left(T \cdot (k \cdot C_{\text{draft}} + k^2 d + \tilde{L} \tilde{H} T^2 + (1 - \pi) \cdot k \cdot C_{\text{target}})\right),$$

where  $\tilde{L} \ll L$  and  $\tilde{H} \ll H$  represent the reduced number of layers and heads used in ABGV under sparsity/last-layer approximations.

It is not difficult to infer the following from the complexity analysis.

- SPEC GUARD avoids the external PRM cost  $C_{\text{PRM}}$  in RSD, reducing latency and compute.
- With practical optimizations ( $\tilde{L} \approx 3$ ,  $\tilde{H} \ll H$ ), the ABGV overhead is negligible relative to the forward draft / target passes.
- Empirically, SPEC GUARD achieves up to 11% lower runtime than RSD while improving accuracy by 1–3% in benchmarks.

### A.4 LLM Usage

Large Language Models (LLMs) were used in this work solely as general-purpose assistive tools. Specifically, they were employed in two limited capacities: (i) to aid in polishing the writing for clarity and readability, and (ii) to assist in retrieval and discovery tasks, such as identifying related work. No part of the research design, algorithm development, theoretical analysis, or experimental implementation relied on LLMs. Their role was restricted to supportive tasks.

Table 3: Overview of the Complex QA datasets used in this study.

Dataset	#Test	Example Question	Description
MATH500 (Hendrycks et al., 2021)	500	What is the smallest positive perfect cube that can be written as the sum of three consecutive integers?	multi-step arithmetic word problems
GSM8K (Cobbe et al., 2021)	1319	The red car is 40% cheaper than the blue car. The price of the blue car is \$100. How much do both cars cost?	multi-step arithmetic word problems
GaoKao-2023-En (Liao et al., 2024)	385	Suppose the universe set is $U=\{0,1,2,4,6,8\}$ . Two of its subsets are $M=\{0,4,6\}$ , $N=\{0,1,6\}$ . Find $M \cup \bar{N}$ .	multi-step arithmetic word problems
OlympiadBench (He et al., 2024)	675	A number is called Norwegian if it has three distinct positive divisors whose sum is equal to 2022. Determine smallest Norwegian number.	multi-step arithmetic word problems

Table 4: Accuracy with different  $\beta$ s. Overall,  $\beta = 0.3$  works well for different tasks.

Method	Target Model	Draft Model	Setting	MATH500	GSM8K	Gaokao 2023 En	Olympiad Bench
<b>Math Model, Draft and Target: Qwen2.5-Math-Instruct</b>							
Ours Majority	7B	1.5B	$\beta = 0.3$	85.4	95.8	69.4	41.2
Ours Majority	7B	1.5B	$\beta = 0.5$	85.0	95.7	68.5	40.4
Ours Majority	7B	1.5B	$\beta = 0.7$	84.4	95.6	65.5	40.2

Table 5: Accuracy with different  $\tau$ s. Overall,  $\tau = 0.7$  works well for different tasks.

Method	Target Model	Draft Model	Setting	MATH500	GSM8K	Gaokao 2023 En	Olympiad Bench
<b>Math Model, Draft and Target: Qwen2.5-Math-Instruct</b>							
Ours Majority	7B	1.5B	$\tau = 0.6$	83.6	93.5	67.4	39.7
Ours Majority	7B	1.5B	$\tau = 0.7$	85.4	95.8	69.4	41.2
Ours Majority	7B	1.5B	$\tau = 0.8$	84.2	94.6	68.7	40.4
Ours Majority	7B	1.5B	$\tau = 0.9$	85.1	95.6	69.2	41.0

Table 6: Qualitative analysis of reasoning steps scored by PRM. Even though all draft-generated steps receive high acceptance scores from the PRM, the final answer is still incorrect.

Reasoning Steps	PRM Scores
Step 1: Identify the number of diamonds in the chest. The treasure hunter found 175 diamonds.	0.962108
Step 2: Calculate the number of rubies. There were 35 fewer rubies than diamonds, so the number of rubies is $175 - 35 = 140$ .	0.962108
Step 3: Calculate the number of emeralds. The number of emeralds is twice the number of rubies, so the number of emeralds is $2 * 140 = 280$ .	0.962108
Step 4: Calculate total number of gems. To find the total number of gems, add the number of diamonds, rubies, and emeralds: $175 + 140 + 280 = 535$ .	0.961534
Step 5: The final answer is: <span style="border: 1px solid red; padding: 2px;">535</span>	0.962108

Table 7: Qualitative analysis of reasoning steps scored by **PRM**. *Even though all draft-generated steps receive high acceptance scores from the PRM, the final answer is still incorrect.*

*Question: Two girls each got 1/6 of the 24 liters of water. Then a boy got 6 liters of water. How many liters of water were left?*

Reasoning Steps	PRM Scores
Step 1: Calculate the amount of water each girl received. Each girl got 1/6 of 24 liters of water, which is $(1/6) * 24 = 4$ liters.	0.92192
Step 2: Calculate the total amount of water the girls received. Since there are two girls, the total amount of water they received is $2 * 4 = 8$ liters.	0.92079
Step 3: Calculate the amount of water left after the boy got 6 liters. Subtract the amount of water the boys received from the total amount of water the girls received: $8 - 6 = 2$ liters.	0.92079
Step 4: The final answer is: <code>\boxed{2}</code>	0.92079

Table 8: Qualitative analysis of reasoning steps accepted by **SPECGUARD**, where *consistent step verification leads to the correct final answer.*

*Question: A treasure hunter found a buried treasure chest filled with gems. There were 175 diamonds, 35 fewer rubies than diamonds, and twice the number of emeralds than the rubies. How many of the gems were there in the chest?*

Reasoning Steps
Step 1: First, let's identify the number of diamonds, which is given as 175.
Step 2: The number of rubies is 35 fewer than the number of diamonds, so we need to subtract 35 from 175 to find the number of rubies: $175 - 35 = 140$ rubies.
Step 3: The number of emeralds is twice the number of rubies, so we need to multiply the number of rubies by 2 to find the number of emeralds: $140 * 2 = 280$ emeralds.
Step 4: To find the total number of gems in the chest, we need to add the number of diamonds, rubies, and emeralds: $175 + 140 + 280 = 595$ gems.
Step 5: The final answer is: <code>\boxed{595}</code>

Table 9: Qualitative analysis of reasoning steps accepted by **SPECGUARD**, where *consistent step verification leads to the correct final answer.*

*Question: Two girls each got 1/6 of the 24 liters of water. Then a boy got 6 liters of water. How many liters of water were left?*

Reasoning Steps
Step 1: First, let's calculate the amount of water each girl got. Since they each got 1/6 of the 24 liters, we can calculate this as $24 * (1/6) = 4$ liters per girl.
Step 2: Since there are two girls, the total amount of water they got is $4 * 2 = 8$ liters.
Step 3: Now, let's calculate the amount of water left after the girls got their share. We subtract the amount they got from the total amount of water: $24 - 8 = 16$ liters.
Step 4: Then, the boy got 6 liters of water. We subtract this from the remaining water: $16 - 6 = 10$ liters.
Step 5: The final answer is: <code>\boxed{10}</code>