

Look Before You Leap: A Lookahead Reasoning Quality Gate for Speculative Decoding

Hiroaki Kingetsu
Fujitsu Limited
h.kingetsu@fujitsu.com

Kenji Fukumizu
The Institute of Statistical Mathematics
Fujitsu Limited
fukumizu@ism.ac.jp

Kaoru Yokoo
Fujitsu Limited
kaoru_yokoo@fujitsu.com

Manohar Kaul
Fujitsu Limited
kaul.manohar@fujitsu.com

Abstract

We present a lookahead quality gate (verifier) for speculative decoding for reasoning or chain-of-thought language models. The gate accepts the longest reliable prefix of each k -token lookahead (block-wise) draft. Unlike token-level likelihood search, which is myopic and often rewards verbosity, or tree-level sampling methods that trade accuracy for latency, our approach works at an intermediate granularity. It uses only the base model’s hidden states to compute a geometry-based quality score for each prefix, then accepts the longest prefix whose score exceeds a quantile-calibrated threshold estimated from unlabeled prompts. The method integrates seamlessly with speculative/blockwise decoding and adds minimal runtime overhead, requiring no auxiliary heads, reward models, or finetuning. On math and science benchmarks, it improves accuracy over sampling baselines while achieving 2.6 – 7.9 faster generation.

1 Introduction

Large language models (LLMs) demonstrate impressive multi-step reasoning ability through chain-of-thought prompting (Wei et al., 2022), but their test-time control remains costly and fragile. Current approaches to steer generation fall broadly into two families. *Token-level* controls, such as beam search and heuristic decoding, optimize local likelihood, a proxy that correlates with reasoning quality only weakly and often rewards verbosity (Holtzman et al., 2020; Murray and Chiang, 2018). *Tree-/chunk-level* methods, including Self-Consistency (Wang et al., 2023), Tree-of-Thoughts (Yao et al., 2023), and Monte Carlo Tree Search (MCTS, Chen et al., 2024), improve robustness via multi-sample exploration but incur substantial latency and engineering overhead.

This paper asks whether multi-token drafts can be verified during decoding, without extra training,

to retain the throughput of speculative/blockwise generation while improving reasoning reliability.

We focus on the draft frameworks that generate k -token lookahead drafts across m branches before accepting them. While speculative/blockwise decoding greatly improves throughput, the current acceptance criteria, typically based on probability checks such as logit agreement and rejection sampling, prioritize speed rather than reasoning quality. As a result, high-likelihood yet flawed spans can still pass verification (Li et al., 2024; He et al., 2024b; Bhendawade et al., 2024). Our goal is a verifier that (i) operates at an intermediate granularity between token and tree, (ii) is training-free and easy to calibrate, and (iii) composes seamlessly with probability checks without auxiliary heads or models.

Approach. We propose a *lookahead quality gate*—a verifier that self-evaluates each k -token draft, or *lookahead*. For each lookahead, the gate summarizes the base model’s hidden states over the k drafted tokens, computes a geometry-based internal similarity score derived from Chain-of-Embedding (CoE, Wang et al., 2025a), aggregates evidence across each prefix, and accepts the *longest* prefix whose aggregated score exceeds a *quantile-calibrated* threshold. The threshold is decided from unlabeled prompts. The gate integrates directly into speculative/blockwise decoding, preserves parallelism, and adds only a minor verification overhead. Our design leverages internal representations known to correlate with factuality and reasoning consistency (Chuang et al., 2024; Wang et al., 2025a) and complements process-/content-level verifiers (Vacareanu et al., 2024; Ling et al., 2023; He et al., 2024a) without additional models or training. The overall workflow is illustrated in Figure 1.

Advantage. Token-level signals are noisy and myopic, while tree-level verifications are expen-

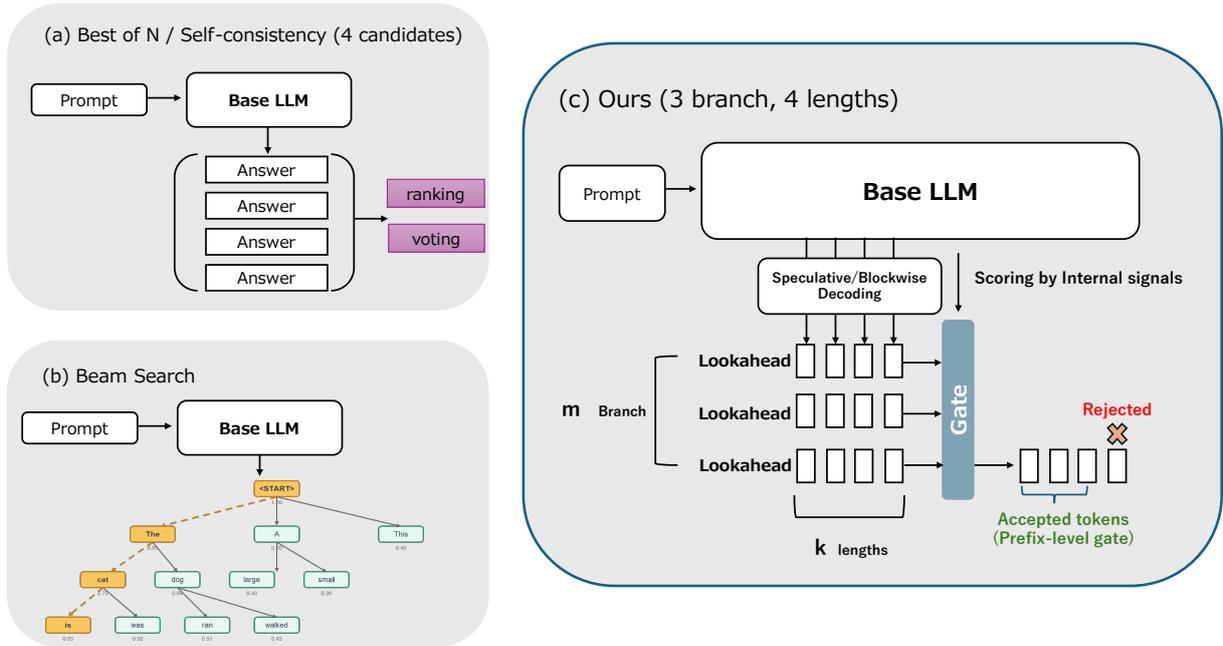


Figure 1: Method overview. (a) Best-of- N / Self-Consistency: sample multiple chains and aggregate. (b) Beam search: expand hypotheses by likelihood. (c) Our approach: verify k -token lookaheads and accept only the longest trustworthy prefix.

sive (Wang et al., 2023; Yao et al., 2023). Our proposed gate operating on short drafted windows captures medium-range coherence, which is known to correlate with reasoning reliability (Chuang et al., 2024; Wang et al., 2025a). The quantile calibration provides an acceptance-rate control from unlabeled data, yielding stable behavior under fixed compute. Combined with probability checks (Leviathan et al., 2023; Li et al., 2024), the gate rejects high-likelihood but low-quality drafts without sacrificing throughput.

Contributions. This work makes the following contributions.

1. We propose a lookahead-level verifier that uses base-model hidden states to accept reliable multi-token prefixes in draft-verify frameworks, complementing speculative/blockwise methods.
2. We design a practical acceptance policy (prefix-wise aggregation + quantile-calibrated threshold) that integrates seamlessly with speculative draft-verify frameworks.

Empirical overview. On GSM8K-100, MATH500-100, and GPQA-diamond-100, our method improves accuracy over Self-Consistency by +0.8–4.2pp while achieving 2.6–7.9 \times faster end-to-end generation (60–88% cost reduction). On challenging competition

problems (AIME2024/2025), Self-Consistency achieves higher accuracy, while our method remains competitive with shorter outputs.

2 Related Work

Token-level search. A widely used decoding strategy is beam search, which ranks partial hypotheses by cumulative likelihood with penalties or normalization. However, likelihood is a local proxy; it correlates poorly with step-wise correctness and often rewards verbosity (Wu et al., 2016; Holtzman et al., 2020). In reasoning settings, this proxy mismatch can prune correct branches prematurely: intermediate arithmetic steps or symbolic tokens often have lower likelihood than fluent but incorrect narratives, so beam search drops the correct branch before it reaches the final answer. Although there are beam search variants guided by self-evaluation, which alleviate the issue by adding auxiliary scorers (Xie et al., 2023), they require additional model passes and bespoke scoring heads, effectively turning the decoder into a verifier-assisted system with higher implementation and compute overhead. Our approach eschews likelihood ranking at the token level and instead accepts the longest acceptable k -step lookahead prefix using a calibrated CoE score.

Search over intermediate reasoning. *Chain-of-Thought* (CoT) prompting (Wei et al., 2022), *Self-Consistency* (SC, Wang et al., 2023), and *Tree-of-Thoughts* (Yao et al., 2023) expand the search space by sampling or branching intermediate reasoning steps and then aggregating them. Although improving robustness, these methods introduce substantial latency by requiring many long reasoning chains or complex coordination, resulting in slowdowns of 10–20× in some cases (Chen et al., 2024).

Speculative and lookahead decoding. *Speculative decoding* uses draft–verify pipelines, which accelerate text generation by proposing multiple draft tokens in advance and verifying them with the target model (Leviathan et al., 2023; Chen et al., 2023). Recent approaches aim at practical speedups by acceptance with probability consistency or by multi-token prediction heads (e.g. Medusa, Cai et al., 2024). Further extensions include feature- or latent-level variants (e.g. EAGLE, Li et al., 2024) and tree-structured verification (e.g., SpecInfer, Miao et al., 2024). *Lookahead decoding* accepts multiple future tokens in a single step, thereby reducing sequential dependence, while preserving exact distributional correctness (Fu et al., 2024). Across these lines, the *acceptance signal* is primarily based on probability consistency.

Reward models and verifier-in-the-loop. Process reward models (PRMs) and outcome reward models (ORMs) train separate evaluators to score intermediate steps or final answers, which can improve reasoning but incur substantial costs in additional data, training, and inference passes (Lightman et al., 2024). Prompt-based self-evaluation (e.g., Self-Refine) replaces learned verifiers with meta-prompts, but requires careful orchestration and can be brittle across domains.

Internal signals for test-time control. Entropy-based early stopping (HALT-CoT, Laouach, 2025) and layer-contrast decoding (DoLa, Chuang et al., 2024) exploit internal uncertainty/logit contrasts for test-time decisions. Chain-of-Embedding (CoE, Wang et al., 2025a) introduced a geometry-based metric that tracks layer-wise representation trajectories; smoother, more consistent paths correlate with coherent reasoning.

Our positioning. Compared with likelihood-driven token search, our approach replaces local likelihood proxies with a *multi-token* signal

based on CoE that evaluates short prefixes directly from the model’s hidden states. In contrast to tree-based or self-consistency methods, the approach avoids expensive re-generation and reliance on external judges, while preserving the robustness benefits of multi-branch drafting. Relative to speculative/blockwise accelerators, our verifier integrates probability matching with an internal content-quality gate, accepting the longest admissible prefix per step to balance speed and reliability. Together, these choices yield a practical, training-free verifier that enhances the accuracy–efficiency trade-off and can be seamlessly incorporated into existing draft–verify pipelines. For completeness, detailed latency and ablation comparisons across speculative variants as well as oracle analyses of the CoE signal are provided in Appendices A and B.

3 Method

3.1 Preliminaries: Draft–Verify Framework

Let the prompt (input sequence) be denoted by $x_{1:T} = (x_1, \dots, x_T)$, and let T denote the total generated sequence length. At each decoding step the draft model \mathcal{M}_d proposes m branches of length k , and the verifier \mathcal{M}_v evaluates every prefix of each branch. We adopt the self-verification setting $\mathcal{M}_d = \mathcal{M}_v$, so the verifier reuses the hidden states computed during drafting without any additional forward passes. We consider a draft–verify framework in which, given input or prompt, a draft model \mathcal{M}_d proposes m branches of length k each and a verifier \mathcal{M}_v (which we refer to as a quality gate) uses its internal hidden states to score each prefix of every branch. This paper adopts the *self-verification* setting $\mathcal{M}_d = \mathcal{M}_v$. Many recent speculative/blockwise methods employ a single base model for both drafting and verification, using multi-token heads or internal states, such as Medusa (Cai et al., 2024), EAGLE (Li et al., 2024), and LayerSkip (Elhoushi et al., 2024), or tree-based verification (SpecInfer, Miao et al., 2024) and lookahead decoding (Fu et al., 2024).

3.2 CoE Gating

Before formalizing the CoE score, we describe the lookahead hidden-state trajectory that CoE evaluates. We refer to each k -token branch as a **lookahead**. For each lookahead, let $z_\ell^{(u)} \in \mathbb{R}^D$ denote model’s hidden state at layer $\ell \in \{0, \dots, L\}$ for token position $u \in \{1, \dots, k\}$. To evaluate a pre-

fix of length $t \in \{1, \dots, k\}$ in a lookahead, we compute the token-averaged representation at each layer: $h_\ell^{(t)} = \frac{1}{t} \sum_{u=1}^t z_\ell^{(u)}$, forming the depth-wise trajectory $H^{(t)} = (h_0^{(t)}, \dots, h_L^{(t)})$. Token-averaging stabilizes CoE computation by reducing variance from outlier tokens (e.g., rare symbols, numerals) that would otherwise dominate the layer-wise geometry.

The *Chain-of-Embedding* (CoE) metric (Wang et al., 2025a) quantifies how hidden representations evolve across the model’s layers. Intuitively, coherent reasoning induces smooth and consistent trajectories, whereas incorrect reasoning tends to yield erratic or stagnant paths, as shown in (Wang et al., 2025a). Following that work, we use the layerwise change magnitude M_ℓ and angular consistency A_ℓ ; M_ℓ captures the magnitude of representation change between consecutive layers, while A_ℓ is the angle of the direction change. More formally, for a prefix of length t , we define

$$M_\ell^{(t)} = \|h_{\ell+1}^{(t)} - h_\ell^{(t)}\|_2, \quad (1)$$

$$A_\ell^{(t)} = \arccos\left(\frac{\langle h_{\ell+1}^{(t)}, h_\ell^{(t)} \rangle}{\|h_{\ell+1}^{(t)}\|_2 \|h_\ell^{(t)}\|_2}\right), \quad (2)$$

$$Z_{\text{mag}}^{(t)} = \|h_L^{(t)} - h_0^{(t)}\|_2. \quad (3)$$

CoE-C (complex combination). The CoE-C score for prefix t , proposed in (Wang et al., 2025a), aggregates layer-wise changes into a single scalar via complex-valued arithmetic, expressing the magnitude and angle effectively as different components:

$$\bar{C}^{(t)} = \frac{1}{L} \sum_{\ell=0}^{L-1} \frac{M_\ell^{(t)}}{Z_{\text{mag}}^{(t)}} \cos A_\ell^{(t)}, \quad (4)$$

$$\bar{S}^{(t)} = \frac{1}{L} \sum_{\ell=0}^{L-1} \frac{M_\ell^{(t)}}{Z_{\text{mag}}^{(t)}} \sin A_\ell^{(t)}, \quad (5)$$

$$\text{CoE-C}(H^{(t)}) = \sqrt{(\bar{C}^{(t)})^2 + (\bar{S}^{(t)})^2}. \quad (6)$$

While the original work also proposes a real-valued variant (CoE-R), our experiments find CoE-C to be more stable for prefix verification.

Contrast to probability-matching verification. Conventional draft–verify frameworks primarily *accept by probability matching*. In contrast, we provide a *content-quality gate* using *hidden-state representations* (CoE) that can complement that signal. This combination targets false acceptances of high-likelihood yet incorrect reasoning fragments in lookaheads.

Prefix-level gate. The original CoE computes sentence-level features; we adapt it to *prefix-level gate* by forming token-averaged states over the drafted tokens only. This averaging is not merely cosmetic: in pilot experiments without averaging, a few high-variance tokens (e.g., numerals, symbols, rare subwords) dominated the layerwise geometry, causing the gate to reject all candidates and stalling decoding. Averaging dampens these outliers while preserving the CoE trajectory, enabling stable per-prefix acceptance in speculative/blockwise decoding.

More formally, for each branch, we compute $\{s_t\}_{t=1}^k$, where $s_t = \text{CoE-C}(H^{(t)})$ in (6). We optionally form an aggregated prefix score a_t by

$$a_t^{\text{step}} = s_t, \quad (7)$$

$$a_t^{\text{mean}} = \frac{1}{t} \sum_{u=1}^t s_u, \quad (8)$$

$$a_t^{\text{wmean}} = \frac{\sum_{u=1}^t w_{u,t} s_u}{\sum_{u=1}^t w_{u,t}}, \quad w_{u,t} \in \{1, u/t\}, \quad (9)$$

then compare a_t to a threshold γ .

3.3 Calibration

We calibrate the threshold γ to control a target acceptance rate α using *unlabeled* prompts. Let f denote the chosen per-prefix score (s_t or a_t). We assume that calibration and test prompts are sampled i.i.d. from the same population. Under this assumption we set

$$\gamma = \text{Quantile}_{1-\alpha}^{\text{emp}}(\{f_i\}_{i=1}^n)$$

yields acceptance that tracks the target α on new inputs on average, typically with slight conservatism for small n . In practice, a few dozen calibration prompts ($n \approx 20\text{--}50$) suffice to stabilize γ . We found a fixed α to realize comparable acceptance across nearby reasoning domains, with re-calibration advisable for distant ones. This quantile rule is training-free and adds no extra forward passes (scores are computed from cached hidden states). We report acceptance– α sweeps in Fig. 2.

3.4 Acceptance policy and reverse scan

Terminology. We distinguish two notions that have different implications for streaming and caching: *Prefix verification* means a decision rule that evaluates only the first t tokens of a candidate $y_{1:k}$ to decide whether to accept the prefix

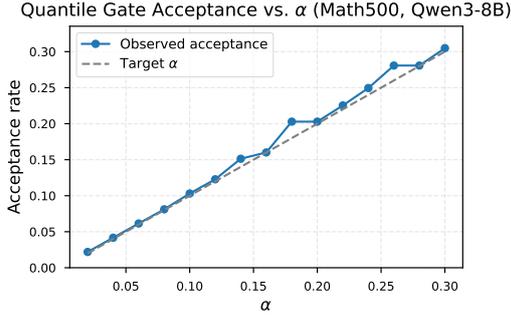


Figure 2: Observed acceptance rate when sweeping α on MATH500 (Qwen3-8B-Thinking). The empirical mass sits just below the diagonal $y = \alpha$, matching the conformal upper-bound intuition.

$y_{1:t}$. *Prefix-closed verification* further requires that if $y_{1:t}$ is accepted then every shorter prefix $y_{1:t'}$ with $t' < t$ is also accepted. The latter ensures monotonic commitment (once tokens are accepted, they cannot be rejected later), which is essential for streaming applications but may sacrifice optimality—a high-quality 10-token prefix might be rejected if tokens 3-5 have low scores, even though the overall prefix is strong.

Our acceptance rule. Given a per-prefix score a_t computed *using only* tokens up to t , and a calibrated threshold γ , we select the

$$r^* = \max\{t \in \{1, \dots, k\} : a_t \geq \gamma\}.$$

We then return the longest acceptable prefix $y_{1:r^*}$ (with $r^* = 0$ if the set is empty). This constitutes *prefix verification* but is not necessarily *prefix-closed* since $\{t : a_t \geq \gamma\}$ need not be an initial segment of $\{1, \dots, k\}$.

Implementation detail (reverse scan). We implement a reverse scan from $t=k$ down to 1 and return the first t such that $a_t \geq \gamma$. This is equivalent to the rule above and returns the unique longest acceptable lookahead.

3.5 Implementation and computational cost

Arithmetic complexity. In the draft phase, the model generates m branches of length k via standard forward passes, which naturally produce hidden states $\{z_\ell^{(u)}\}$ at all layers $\ell \in \{0, \dots, L\}$ for each token position u . Our verifier *reuses* these already-computed states to extract CoE scores, requiring no additional model forward passes—only lightweight post-processing (layer-wise norms and cosines). This yields $\mathcal{O}(mkLD)$ arithmetic overhead per iteration, negligible compared to the

$\mathcal{O}(mk \cdot V \cdot D)$ cost of the draft pass itself (where V is vocabulary size).

Memory footprint. KV cache management requires $\mathcal{O}(T)$ for the committed target path plus $\mathcal{O}(mk)$ for branch states during each iteration. Under monotonic acceptance (no rollback), peak memory scales linearly with output length, matching standard autoregressive generation. By contrast, Self-Consistency with N parallel chains incurs $\mathcal{O}(N \cdot T)$ peak KV unless chains are regenerated sequentially (sacrificing cache reuse).

Runtime overhead. Empirical profiling on Qwen3-8B-Thinking (GSM8K-100, A100 GPU) shows that verification time is $\rho \approx 0.12$ of drafting time on average (median 0.120, 95th percentile 0.135), confirming lightweight integration. The verifier stays within the compute-parity envelope and avoids cross-model calls. Full profiling details and cache scheduling strategies are provided in Appendix G.

4 Experiments

4.1 Tasks and Benchmarks

We evaluate our method on **GSM8K-100** (first 100 problems of GSM8K) (Cobbe et al., 2021), **MATH500-100** (first 100 of MATH500) (Hendrycks et al., 2021), **GPQA-diamond-100** (first 100 of GPQA-diamond) (Rein et al., 2024), **AIME2024** (Mathematical Association of America, 2024) and **AIME2025** (Mathematical Association of America, 2025). Subsampling the leading 100 follows compute-conscious prior work (Yao et al., 2023) and allows reusing the same calibration pools and ablations across runs. All prompts are single-turn and do not use external tools.

4.2 Models

We evaluate **Meta-Llama-3.1-8B-Instruct** (Grattafiori et al., 2024) and two sizes of **Qwen3** (Yang et al., 2025a). In these experiments, we use the native reasoning-enabled modes (Qwen3-8B-Thinking, Qwen3-32B-Thinking), which emit an internal reasoning segment wrapped in `<think>` tags before yielding the final answer. This improves accuracy by enabling longer, explicit reasoning. See Appendix J for examples. For Meta-Llama-3.1-8B-Instruct we adopt CoT prompting to elicit explicit reasoning.

4.3 Methods

Ours. We instantiate the gate with CoE (Section 3), using token-averaged *lookahead* states per layer, CoE-C for scoring, and aggregation (wmean). Unless stated otherwise, we set lookahead length $k=10$, branches $m=4$, and sampling parameters temperature $\tau=0.6$, top- $p=0.95$, top- $k=30$. Calibration uses 50 unlabeled prompts held out from each benchmark (GSM8K-100, MATH500-100, GPQA-diamond-100) to set a domain-specific quantile threshold at risk level $\alpha=0.80$. For AIME, where the total dataset size is only 60 problems (30 per year), we perform cross-calibration between AIME2024 and AIME2025: the threshold for AIME2024 is calibrated on holdout prompts from AIME2025, and vice versa. We measure wall-clock latency and accuracy results on an A100 GPU (40GB).

Baselines. We compare against compute-matched decoding: (1) **Pass@1**; (2) **Best-of- N (LL)** with length-normalized log-probability re-ranking; (3) **Best-of- N (CoE)** using the CoE score; (4) **Self-Consistency** (Wang et al., 2023): majority vote over N samples. Verifier-guided beam variants and reward-model-assisted decoding are complementary to our gate and are treated as out of scope for baseline comparisons.

4.4 Compute Parity

To ensure fair comparison, we follow a *token-equivalent* budget. One gate cycle drafts mk tokens and runs verification on cached states. Let ρ denote the wall-clock ratio between verification and drafting within the cycle. We set baseline sample count N such that the expected number of decoded tokens satisfies

$$\mathbb{E}[L] \cdot N \approx mk(1 + \rho),$$

where L is the average decoded length under the baseline. In our setup, this yields $N=4$ for multi-sample baselines. Measurement details for ρ and full profiles are deferred to Appendix H (Table 4).

4.5 Metrics and Protocol

Our primary evaluation metrics are **Accuracy** and **Token length**. All runs use identical prompts and fixed seeds. We evaluate each method on 16 independently sampled completions per prompt. For multi-sample baselines (BoN, SC) with $N=4$ (determined by compute parity in Section 4.4), we report accuracy by averaging over random N -subsets

drawn from the 16 samples to ensure statistical robustness. Token length counts all newly generated tokens and *includes* the `<think>` segments for models that emit them.

4.6 Main Results: Accuracy and Token Length

As detailed below, on **GSM8K-100**, **MATH500-100**, and **GPQA-diamond-100**, our lookahead quality gate consistently *improves* accuracy over baselines (including Self-Consistency) while *significantly reducing* token length under compute parity. On the more challenging **AIME2024/2025**, Self-Consistency achieves the highest accuracy, while our method remains competitive and delivers considerably shorter outputs.

- **Accuracy.** On GSM8K-100, MATH500-100, and GPQA-diamond-100 (Qwen3-8B-Thinking), our method achieves 98.0%, 92.0%, and 61.0% accuracy respectively, outperforming Self-Consistency by +0.8%, +0.5%, and +2.7%. These gains demonstrate that internal-state verification effectively identifies high-quality reasoning prefixes. On AIME2024/2025, Self-Consistency achieves higher accuracy (80.3%/72.8% vs. 77.8%/70.7%), suggesting that broader parallel search remains beneficial on extremely challenging competition problems where diverse solution strategies are critical.
- **Token length.** Our method delivers substantially shorter outputs: on Qwen3-8B-Thinking, we reduce tokens by 2%–12% on GSM8K/MATH/GPQA while maintaining or improving accuracy. Even on AIME2024/2025, where Self-Consistency attains the highest accuracy, our method remains competitive and tends to produce shorter outputs under the same token-equivalent budget. Empirically, lookahead-prefix CoE stabilizes acceptance, avoiding token-level oscillations that inflate length.

Acceptance Policy and Metric Score (ablation summary). We ablate two orthogonal design choices on MATH500-100 (Qwen3-8B-Thinking): (i) prefix verification vs. longest-only acceptance, and (ii) CoE vs. entropy vs. log-probability scoring. Prefix verification with reverse scan improves accuracy by +1.0pp over longest-only (92.0% vs. 91.0%). Among scoring objectives, CoE achieves

Table 1: Main results: **Accuracy(%) / Token Length** across different models and datasets. All methods are evaluated under compute parity. Accuracy and token lengths are averaged across multiple seeds where applicable. BoN (LL) = Best-of-N with length-normalized log-probability; SC = Self-Consistency. **Bold** denotes the best, and underline denotes the second best.

Method	GSM8K-100			MATH500-100			GPQA-diamond-100		
	Llama3.1 8B	Qwen3 8B	Qwen3 32B	Llama3.1 8B	Qwen3 8B	Qwen3 32B	Llama3.1 8B	Qwen3 8B	Qwen3 32B
		Thinking	Thinking		Thinking	Thinking		Thinking	Thinking
Pass@1	74.4 / 210	96.7 / 2602	97.0 / 1519	46.3 / 1611	90.9 / 5066	91.3 / 4047	24.1 / 2200	57.0 / 9887	70.3 / 5758
BoN (LL)	75.8 / 307	96.9 / 2306	96.7 / 1881	48.2 / 3961	90.8 / 5305	90.5 / 4498	23.4 / 3619	54.0 / 12369	<u>71.4</u> / 6063
BoN (CoE)	74.5 / 322	96.5 / 2719	96.7 / 1722	42.1 / 3845	91.4 / 4014	90.6 / 3891	<u>25.3</u> / 3533	55.3 / 10646	71.0 / 5698
SC	<u>81.3</u> / 210	<u>97.2</u> / 2286	<u>97.4</u> / 1517	<u>51.9</u> / 1182	<u>91.5</u> / 5029	<u>92.0</u> / 3915	23.7 / 2012	<u>58.3</u> / 9737	73.0 / 5276
Ours	85.5 / 228	98.0 / 2235	98.0 / 1549	55.0 / 1405	92.0 / 4666	93.0 / 4214	26.0 / 2404	61.0 / 8535	70.5 / 5197

Method	AIME2024			AIME2025		
	Llama3.1 8B	Qwen3 8B	Qwen3 32B	Llama3.1 8B	Qwen3 8B	Qwen3 32B
		Thinking	Thinking		Thinking	Thinking
Pass@1	0.5 / 5018	75.8 / 13968	80.4 / 13507	5.0 / 3629	67.3 / 18062	67.9 / 14080
BoN (LL)	0.2 / 9091	69.3 / 18553	77.1 / 16977	2.8 / 8815	63.3 / 20116	64.2 / 17776
BoN (CoE)	0.3 / 9079	71.6 / 18681	79.3 / 12212	3.1 / 8468	60.0 / 21093	62.5 / 17789
SC	0.4 / 3483	80.3 / 13828	83.4 / 11839	5.2 / 4065	72.8 / 17393	83.3 / 14721
Ours	0.3 / 4661	<u>77.8</u> / 15569	<u>83.3</u> / 11251	3.3 / 3706	<u>70.7</u> / 18724	<u>70.6</u> / 14764

Note: “Qwen3-8B/32B-Thinking” = reasoning-enabled modes (outputs <think> segments). All token counts include <think> content where applicable. See Appendix F for accuracy standard deviations.

92.0% accuracy with $\sim 1.5\times$ faster wall-clock time than log-probability (183.3s vs. 272.3s) and +1.3pp higher accuracy. Full results and analysis appear in Appendix B.

4.7 Analysis

Acceptance analysis. Across benchmarks, the verifier is most selective on easier GSM8K/MATH (lower draft/token acceptance and shorter accepted prefixes) and more permissive on harder GPQA/AIME (longer prefixes). Acceptance correlates weakly but positively with correctness; details and full plots appear in Appendix D.

Calibration sensitivity. Our calibration uses 50 unlabeled prompts per model. Empirically, thresholds calibrated on one benchmark (e.g., GSM8K) transfer reasonably to similar reasoning tasks (e.g., MATH500 and GSM8K) without re-calibration, as evidenced by consistent accuracy in Table 1. Lower risk ($\alpha\downarrow$) tightens acceptance and shortens outputs; slightly higher risk ($\alpha\uparrow$) increases acceptance.

Accept scan direction. We adopt reverse scanning, which selects the longest prefix whose aggregated score exceeds the threshold. This choice preserves accuracy while avoiding the premature truncation introduced by forward scanning, which

terminates at the first threshold violation and therefore shortens the accepted prefix.

4.8 Efficiency and Latency

We measure wall-clock latency on an A100 GPU (40GB) and compare our method against Self-Consistency under matched compute budgets (Figure 3).

Latency per problem. On Qwen3-8B-Thinking, our method achieves substantial speedups across all benchmarks compared to Self-Consistency: **GSM8K-100** (218s vs. 572s, $2.6\times$ faster), **MATH500-100** (328s vs. 1181s, $3.6\times$ faster), **GPQA-diamond** (254s vs. 2004s, $7.9\times$ faster). Self-Consistency requires many long samples to achieve high accuracy, while our verifier achieves comparable or better accuracy with fewer iterations by selectively accepting high-quality prefixes early.

Throughput. In terms of throughput (problems/hour), our method processes 16.5 GSM8K problems/hour (vs. 6.3 for SC), 11.0 MATH500 problems/hour (vs. 3.0), 14.2 GPQA problems/hour (vs. 1.8), and 5.6 AIME problems/hour (vs. 1.0). This translates to end-to-end inference cost reductions of 60–88% compared to Self-Consistency at matched accuracy levels.

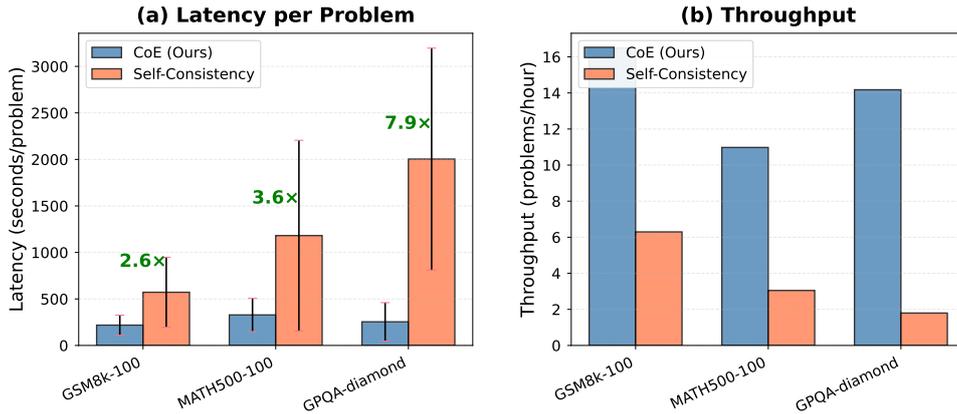


Figure 3: Efficiency comparison on Qwen3-8B-Thinking. (a) Latency per problem: our method (CoE) achieves 2.6–7.9× speedups over Self-Consistency (SC) across all benchmarks. Speedup factors are shown in green. (b) Throughput: our method processes 2.6–8× more problems per hour than SC, translating to 60–88% cost reduction at matched accuracy.

5 Conclusion

Summary of contributions. We presented the *lookahead quality gate*, a training-free lookahead verifier for speculative decoding with the following key components: (1) a *lookahead-level verification framework* that operates at intermediate granularity between token and tree search via token-averaged internal representation metrics; (2) a *quantile-calibrated acceptance policy* that provides statistically grounded precision control from unlabeled prompts.

The method integrates seamlessly with speculative/blockwise decoding, reuses draft hidden states (no extra forward passes), and achieves 2.6–7.9× latency reductions while improving accuracy on math and science benchmarks.

Key differences. Self-Consistency and MCTS improve robustness by broadening search but incur large compute; neither leverages internal representations. Speculative/blockwise accelerators improve throughput but primarily rely on token/logit agreement. Our gate instead reads *hidden-state geometry* (CoE) and admits multi-token prefixes when evidence is strong, composing with probability checks while providing calibrated quality control via a quantile threshold γ at risk level α .

We presented a practical, calibrated, geometry-guided verifier that accepts multi-token prefixes when internal evidence is strong. On standard reasoning benchmarks (GSM8K, MATH500, GPQA-diamond), our method improves accuracy over Self-Consistency while reducing token length under compute parity; on extremely challenging competition problems (AIME), Self-Consistency re-

mains superior in accuracy, whereas our method is competitive and yields shorter outputs. In our setup, this translates into substantial latency and throughput gains (see Fig. 3) under the same token-equivalent budget. The method is training-free, plug-and-play with speculative/blockwise decoding, and calibrated via unlabeled quantiles; it reuses draft hidden states (no extra forward passes) and stays within the compute-parity envelope.

6 Limitations

Our approach requires access to hidden states (closed APIs may preclude this) and shows its largest gains on math and science style reasoning. Calibration is *dataset-/domain-sensitive*: we observe partial transfer across nearby math domains, while distant domains typically benefit from recalibration and a careful choice of α . Our acceptance rule computes a_t cumulatively over the prefix $y_{1:t}$ (e.g., averaging segment scores), so a long prefix may receive a low aggregate score if an earlier segment is poor, even if later segments are strong. This is an inherent trade-off of holistic scoring: prefix-level averaging can be dragged down by a single weak segment, whereas a prefix-closed rule would commit good early segments before encountering the weak one. For streaming settings where monotonic commitment is essential, prefix-closedness can be enforced via a monotone aggregator (e.g., $a_t = \min_{u \leq t} s_u$) or by using non-decreasing per-step thresholds $\{\gamma_t\}$. Finally, we do not claim novelty for CoE itself; our contribution is in *how* to adapt and deploy it for prefix-level verification in draft–verify loops (averaging, numerical

stability, and quantile calibration). We used AI assistance for proofreading and translation of the manuscript.

Acknowledgements

KF has been supported in part by JST CREST JP-MJCR2015.

References

- Nikhil Bhendawade, Irina Belousova, Qichen Fu, Henry Mason, Mohammad Rastegari, and Mahyar Najibi. 2024. [Speculative streaming: Fast LLM inference without auxiliary models](#). *Preprint*, arXiv:2402.11131.
- Tianle Cai, Yuhong Li, Zhengyang Geng, Hongwu Peng, Jason D. Lee, Deming Chen, and Tri Dao. 2024. [Medusa: Simple LLM inference acceleration framework with multiple decoding heads](#). In *Proceedings of the 41st International Conference on Machine Learning (ICML)*, volume 235 of *Proceedings of Machine Learning Research*, pages 5209–5235. PMLR.
- Charlie Chen, Sebastian Borgeaud, Geoffrey Irving, Jean-Baptiste Lespiau, Laurent Sifre, and John Jumper. 2023. [Accelerating large language model decoding with speculative sampling](#). *Preprint*, arXiv:2302.01318.
- Ziru Chen, Michael White, Ray Mooney, Ali Payani, Yu Su, and Huan Sun. 2024. [When is tree search useful for LLM planning? it depends on the discriminator](#). In *Proceedings of the 62nd Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, pages 13659–13678.
- Yung-Sung Chuang, Yujia Xie, Hongyin Luo, Yoon Kim, James R. Glass, and Pengcheng He. 2024. [DoLa: Decoding by contrasting layers improves factuality in large language models](#). In *The Twelfth International Conference on Learning Representations (ICLR)*.
- Karl Cobbe, Vineet Kosaraju, Mohammad Bavarian, Mark Chen, Heewoo Jun, Lukasz Kaiser, Matthias Plappert, Jerry Tworek, Jacob Hilton, Reiichiro Nakano, Christopher Hesse, and John Schulman. 2021. [Training verifiers to solve math word problems](#). *Preprint*, arXiv:2110.14168. Introduces GSM8K.
- Mostafa Elhoushi, Akshat Shrivastava, Diana Liskovich, Basil Hosmer, Bram Wasti, Liangzhen Lai, Anas Mahmoud, Bilge Acun, Saurabh Agarwal, Ahmed Roman, Ahmed A. Aly, Beidi Chen, and Carole-Jean Wu. 2024. [LayerSkip: Enabling early exit inference and self-speculative decoding](#). In *Proceedings of the 62nd Annual Meeting of the Association for Computational Linguistics (ACL)*, pages 12622–12642. Association for Computational Linguistics.
- 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 (ICML)*, volume 235 of *Proceedings of Machine Learning Research*, pages 13682–13702. PMLR.
- Fabian Gloeckle, Badr Youbi Idrissi, Baptiste Roziere, David Lopez-Paz, and Gabriel Synnaeve. 2024. [Better & faster large language models via multi-token prediction](#). In *Proceedings of the 41st International Conference on Machine Learning (ICML)*, volume 235 of *Proceedings of Machine Learning Research*, pages 15706–15734.
- Aaron Grattafiori, Abhimanyu Dubey, and others. 2024. [The Llama 3 herd of models](#). *arXiv preprint arXiv:2407.21783*.
- Mingqian He, Yongliang Shen, Wenqi Zhang, Zeqi Tan, and Weiming Lu. 2024a. [Advancing process verification for large language models via tree-based preference learning](#). In *Proceedings of the 2024 Conference on Empirical Methods in Natural Language Processing (EMNLP)*, pages 2086–2099.
- Zhenyu He, Zexuan Zhong, Tianle Cai, Jason D. Lee, and Di He. 2024b. [REST: Retrieval-based speculative decoding](#). In *Proceedings of the 2024 Conference of the North American Chapter of the Association for Computational Linguistics (NAACL)*, pages 5322–5338. Association for Computational Linguistics.
- 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 *Proceedings of the NeurIPS 2021 Track on Datasets and Benchmarks*.
- Ari Holtzman, Jan Buys, Li Du, Maxwell Forbes, and Yejin Choi. 2020. [The curious case of neural text degeneration](#). In *Proceedings of the International Conference on Learning Representations (ICLR)*.
- Yassir Laaouach. 2025. [HALT-CoT: Model-agnostic early stopping for chain-of-thought reasoning via answer entropy](#). In *MusIML Workshop at ICML 2025*. Workshop paper.
- Yaniv Leviathan, Matan Kalman, and Yossi Matias. 2023. [Fast inference from transformers via speculative decoding](#). In *Proceedings of the 40th International Conference on Machine Learning (ICML)*, volume 202 of *Proceedings of Machine Learning Research*, pages 19274–19286. PMLR.
- Yaniv Leviathan, Matan Kalman, and Yossi Matias. 2024. [Looking back at speculative decoding](#). *Google Research Blog*.
- Yuhui Li, Fangyun Wei, Chao Zhang, and Hongyang Zhang. 2024. [EAGLE: Speculative sampling requires rethinking feature uncertainty](#). In *Proceedings of the*

- 41st International Conference on Machine Learning (ICML), volume 235 of *Proceedings of Machine Learning Research*, pages 27916–27934. PMLR.
- Hunter Lightman, Vineet Kosaraju, Yuri Burda, Harrison Edwards, Bowen Baker, Teddy Lee, Jan Leike, John Schulman, Ilya Sutskever, and Karl Cobbe. 2024. **Let’s verify step by step**. In *Proceedings of the International Conference on Learning Representations (ICLR)*. PRM800K dataset: <https://github.com/openai/prm800k>.
- Zhan Ling, Yunhao Fang, Xuanlin Li, Zhiao Huang, Mingu Lee, Roland Memisevic, and Hao Su. 2023. **Deductive verification of chain-of-thought reasoning**. In *Thirty-seventh Conference on Neural Information Processing Systems (NeurIPS)*.
- Mathematical Association of America. 2024. American invitational mathematics examination (AIME) 2024. <https://maa.org/maa-invitational-competitions>. Official contest problems used as evaluation set.
- Mathematical Association of America. 2025. American invitational mathematics examination (AIME) 2025. <https://maa.org/maa-invitational-competitions>. Official contest problems used as evaluation set.
- Xupeng Miao, Gabriele Oliaro, Zhihao Zhang, Xinhao Cheng, Zeyu Wang, Rae Ying Yee Wong, Alan Zhu, Lijie Yang, Xiaoxiang Shi, Chunan Shi, Zhuoming Chen, Daiyaan Arfeen, Reyna Abhyankar, and Zhihao Jia. 2024. **SpecInfer: Accelerating generative large language model serving with tree-based speculative inference and verification**. In *Proceedings of the 29th International Conference on Architectural Support for Programming Languages and Operating Systems (ASPLOS 2024)*, volume 3, pages 932–949. ACM.
- Kenton Murray and David Chiang. 2018. **Correcting length bias in neural machine translation**. In *Proceedings of the Third Conference on Machine Translation (WMT18)*, pages 212–223.
- David Rein, Betty Li Hou, Asa Cooper Stickland, Jackson Petty, Richard Yuanzhe Pang, Julien Dirani, Julian Michael, and Samuel R. Bowman. 2024. **GPQA: A graduate-level Google-proof Q&A benchmark**. In *Proceedings of the Conference on Language Modeling (COLM)*. Includes *diamond* subset.
- Junhan Shi, Yijia Zhu, Zhenning Shi, Dan Zhao, Qing Li, and Yong Jiang. 2025. **SpecCoT: Accelerating chain-of-thought reasoning through speculative exploration**. In *ES-FoMo III: 3rd Workshop on Efficient Systems for Foundation Models*.
- Hanshi Sun, Momin Haider, Ruiqi Zhang, Huitao Yang, Jiahao Qiu, Ming Yin, Mengdi Wang, Peter Bartlett, and Andrea Zanette. 2024. **Fast best-of-N decoding via speculative rejection**. In *The Thirty-eighth Annual Conference on Neural Information Processing Systems (NeurIPS)*.
- Robert Vacareanu, Anurag Pratik, Evangelia Spiliopoulou, Zheng Qi, Giovanni Paolini, Neha Anna John, Jie Ma, Yassine Benajiba, and Miguel Ballesteros. 2024. **General purpose verification for chain of thought prompting**. *Preprint*, arXiv:2405.00204.
- Xuezhi Wang, Jason Wei, Dale Schuurmans, Quoc V. Le, Ed H. Chi, and Denny Zhou. 2023. **Self-consistency improves chain of thought reasoning in language models**. *Preprint*, arXiv:2203.11171.
- Yiming Wang, Pei Zhang, Baosong Yang, Derek F. Wong, and Rui Wang. 2025a. **Latent space chain-of-embedding enables output-free LLM self-evaluation**. In *The Thirteenth International Conference on Learning Representations (ICLR)*.
- Zhihai Wang, Jie Wang, Jilai Pan, Xilin Xia, Huiling Zhen, Mingxuan Yuan, Jianye HAO, and Feng Wu. 2025b. **Accelerating large language model reasoning via speculative search**. In *Forty-second International Conference on Machine Learning*.
- Jason Wei, Xuezhi Wang, Dale Schuurmans, Maarten Bosma, Brian Ichter, Fei Xia, Ed Chi, Quoc Le, and Denny Zhou. 2022. **Chain-of-thought prompting elicits reasoning in large language models**. *Preprint*, arXiv:2201.11903.
- Yonghui Wu, Mike Schuster, Zhifeng Chen, and 1 others. 2016. **Google’s neural machine translation system: Bridging the gap between human and machine translation**. *arXiv preprint arXiv:1609.08144*.
- Hanyu Xia, Ming Zhou, and Yunxin Wang. 2024. **A comprehensive survey of speculative decoding**. In *Findings of ACL*.
- Yuxi Xie, Kenji Kawaguchi, Yiran Zhao, Xu Zhao, Min-Yen Kan, Junxian He, and Qizhe Xie. 2023. **Self-evaluation guided beam search for reasoning**. In *Advances in Neural Information Processing Systems (NeurIPS)*.
- An Yang, Anfeng Li, Baosong Yang, Pei Zhang, Jingren Zhou, and 1 others. 2025a. **Qwen3 technical report**. *arXiv preprint arXiv:2505.09388*. Includes *Qwen3-8B-Thinking*, *Qwen3-32B-Thinking*.
- Van Yang, Xiang Yue, Vipin Chaudhary, and Xiaotian Han. 2025b. **Speculative thinking: Enhancing small-model reasoning with large model guidance at inference time**. In *Second Conference on Language Modeling*.
- Shunyu Yao, Dian Yu, Jeffrey Zhao, Izhak Shafran, Thomas L. Griffiths, Yuan Cao, and Karthik R Narasimhan. 2023. **Tree of thoughts: Deliberate problem solving with large language models**. In *Thirty-seventh Conference on Neural Information Processing Systems (NeurIPS)*.

A Speculative Decoding: Recent Trends and Positioning

This appendix expands the discussion in Section 2: (i) classical lossless speculative decoding and speculative sampling; (ii) multi-token prediction heads (e.g., Medusa) and feature/latent-level drafting (e.g., EAGLE); (iii) tree-based verification (SpecInfer) and exact lookahead; (iv) thought-level speculation (e.g., SCoT / Speculative Search). We organize systems by acceptance signal (probability matching vs. content evaluation), added training (aux heads vs. training-free), and unit of acceptance (token, block, thought), with concrete latency/throughput reports where available.

Classical draft–verify frameworks. The seminal “speculative decoding” (Leviathan et al., 2023) and the concurrent “speculative sampling” (Chen et al., 2023) accelerate generation by letting a fast draft model propose multiple tokens and having the target model verify them with a rejection-sampling style correction, provably preserving the target distribution. Subsequent surveys systematize drafter choice and verification rules, and report 2–3.5× speedups without quality loss across diverse settings (Xia et al., 2024; Leviathan et al., 2024).

Multi-token prediction heads (self-contained drafting). Instead of a separate drafter, *multi-head* decoders such as Medusa attach auxiliary heads to predict several future tokens and verify a small tree each step; this yields large practical speedups with lightweight fine-tuning and no external model (Cai et al., 2024). Related multi-token objectives further improve throughput under compute parity (Gloeckle et al., 2024). Our verifier differs by using training-free internal-state signals to gate prefixes, and thus can plug into such block-wise pipelines.

Feature-/latent-level speculative decoding. EAGLE moves drafting from the token space to the *feature* (second-to-top layer) space and resolves feature uncertainty to enable lossless verification, reporting 2.7–3.5× latency speedups on large chat models; follow-ups further optimize tree depth (Li et al., 2024). These methods still accept/reject by probability consistency, whereas we introduce a *content-quality* gate derived from hidden-state geometry.

Tree-based speculative verification. System-oriented methods like SpecInfer organize drafter

outputs as a token tree and verify entire paths in parallel, substantially improving device utilization; production-style implementations demonstrate end-to-end serving speedups (Miao et al., 2024). Our prefix verifier is orthogonal: it can act as the acceptance criterion within such token trees.

Self-speculative / early-exit with shared compute. LayerSkip demonstrates “self-speculative” decoding by exiting early at shallow layers and using deeper layers as a built-in verifier, reducing memory footprint versus two-model setups and showing up to $\sim 2\times$ speedups across tasks (Elhoushi et al., 2024). Our approach shares the *self-evaluated* spirit, but bases acceptance on representation geometry instead of early-exit logits.

Lookahead and exact parallelization (non-drafter). Lookahead Decoding reduces serial dependencies by speculatively evaluating multiple future steps with exactness guarantees and no auxiliary model, reporting up to $1.8\times$ single-GPU speedups and larger multi-GPU gains (Fu et al., 2024). While methodologically distinct from draft–verify, it shares the goal of *accepting multiple tokens per step*; our verifier can serve as a content-quality gate within blockwise acceptance.

Reasoning-level (thought-level) speculative methods. Very recent work extends speculation *beyond tokens* to reasoning steps: Speculative Search coordinates a small model at both thought and token levels to accelerate tree-search reasoning, and Speculative Thinking guides a small reasoner via a large judge at inference time (Wang et al., 2025b; Yang et al., 2025b). SpecCoT performs thought-level speculative exploration by having a small model draft multiple candidate reasoning steps in parallel and a large model verify or reject them step-wise, achieving 1.7–4.1× latency reductions while maintaining accuracy comparable to standard large-model inference (Shi et al., 2025). Fast Best-of-N Decoding via Speculative Rejection extends this line by coupling best-of-N alignment with speculative acceptance, achieving 16–32× lower sampling cost while matching reward-model quality (Sun et al., 2024). These methods still rely on external evaluation of text-level drafts, while our verifier keeps the *judge internal* by reading base-model hidden states.

Positioning of our method. Across these lines, verification typically hinges on *probability matching* (token/logit agreement) or ad-

ditional evaluators. We instead perform *self-evaluated, geometry-driven* acceptance of multi-token prefixes using only the base model’s hidden states—complementary to probability-based acceptance and interoperable with both token-level (Medusa/SpecInfer) and step-level (SCoT) speculative stacks.

B Acceptance Policy and Metric Score Ablations (Details)

Scope. We ablate two orthogonal axes of the draft–verify framework: (i) the *score objective* used to rank candidates (CoE geometry vs. entropy vs. log-probability), and (ii) the *acceptance policy* (our quantile-calibrated *prefix verification* with reverse scan vs. a *longest-only* policy that never truncates within the k -token block). All comparisons use token-equivalent budgets and include <think> tokens.

Setup. Dataset: MATH500-100. Model: Qwen3-8B-Thinking. Unless noted, hyperparameters follow Section 3: lookahead $k=10$, branches $m=4$, temperature $\tau = 0.6$, top- $k=30$, top- $p=0.95$, max new tokens 30000, reverse scan, prefix-wise weighted mean (wmean) aggregation, and a calibrated (quantile) threshold.

B.1 Acceptance policy: prefix verification vs. longest-only

We compare two acceptance policies under the CoE objective. **prefix verification** applies a quantile-calibrated threshold with reverse scan and may truncate within the k -token lookahead. **longest-only** always commits the full k tokens of the top-scoring branch without truncation.

Table 2: CoE under two acceptance policies. Δ denotes accuracy difference (percentage points) relative to fixed-length full-commit.

	Accuracy (%)	Δ
CoE (prefix verification)	92.0	+1.0
CoE (longest only)	91.0	

Summary. Our prefix verification filters low-quality within-block segments and thus shortens outputs while improving precision on harder instances. The longest-only policy prioritizes throughput but forgoes this fine-grained control.

B.2 Score objectives

Here we keep calibrated *prefix verification* (reverse scan) and swap the scoring objective.

Score objectives. We compare:

- **CoE:** hidden-state geometry (CoE-C).
- **Entropy:** per-position predictive entropy from logits.
- **LL:** length-normalized log-probability of generated tokens.

Let $\ell_u \in \mathbb{R}^{|V|}$ be the verification logits at position u and $p_{u,v} := \text{softmax}(\ell_u)_v$. For a drafted prefix $y_{1:t}$ we define the following scores (higher is better):

$$H_u := - \sum_{v \in V} p_{u,v} \log p_{u,v},$$

$$s_t^{\text{ent}} := - \frac{1}{t} \sum_{u=1}^t H_u,$$

$$s_t^{\text{lp}} := \frac{1}{t} \sum_{u=1}^t \log p_u(y_u).$$

Table 3: Score objectives with *lookahead*. Means \pm std.

	Accuracy (%)	Time (s)	Tokens
Entropy	90.0 \pm 0.8	256.8 \pm 32.0	4,673.8 \pm 318.9
LL	90.7 \pm 2.1	272.3 \pm 21.2	5,182.7 \pm 172.7
Ours (CoE w/ lookahead)	92.0	183.3 \pm 9.8	4,666.8 \pm 89.7

Summary. **CoE is $\sim 1.5\times$ faster than log-prob and +1.3 pp higher in accuracy.** Entropy is slower than CoE and does not improve accuracy over log-prob in this setting. Prefix verification tends to shorten outputs while maintaining accuracy.

Why CoE is faster. CoE avoids the LM-head projection and full-vocabulary softmax required by entropy/log-prob. For Qwen3-8B (vocab $\approx 152k$, hidden 3584), projecting $m \times k$ positions per iteration dominates FLOPs and memory traffic; CoE only reads hidden states. Empirically, using logits increased per-iteration time by $\sim 40\text{--}50\%$ in our code path, consistent with the observed wall-clock gap (LL 272.3 s vs. CoE 183.3 s; $\approx 1.49\times$ faster).

B.3 Aggregation policy

In preliminary experiments on GSM8K-100 and MATH500-100, we compared three aggregation strategies for combining per-prefix CoE scores: step (no aggregation, $a_t = s_t$), mean (arithmetic mean over prefix, $a_t = \frac{1}{t} \sum_{u=1}^t s_u$), and wmean

(weighted mean with later steps upweighted, $a_t = \frac{\sum_{u=1}^t w_{u,t} s_u}{\sum_{u=1}^t w_{u,t}}$ where $w_{u,t} \in \{1, u/t\}$).

We find mean to be a strong default across benchmarks. wmean (later steps upweighted) can further shorten outputs on GSM8K-100 without hurting accuracy, while step is the most conservative, producing longer outputs with lower variance across runs. These preliminary findings informed our choice of wmean for the main experiments reported in Section 4.

C Prefix Acceptance Pseudocode

Algorithm 1 details the acceptance routine that selects the longest acceptable prefix from m branches. Given branches Y , cached hidden states Z , a calibrated threshold γ , an aggregation function (Agg), and a scan direction (rev/fwd), the algorithm computes per-prefix CoE scores, aggregates them according to the chosen policy (Section 3), and performs a reverse or forward scan to return the branch whose aggregated score a_t exceeds γ at the longest prefix length. CoE scoring itself is described in Section 3.

D Acceptance Analysis

To understand the verifier’s behavior across benchmarks, we analyze acceptance rates and accepted prefix length distributions on Qwen3-8B-Thinking (Figure 4).

Threshold transfer analysis across benchmarks.

To analyze how the verifier behaves under distributional shift, we conduct an additional experiment where a single threshold γ calibrated on MATH500-100 is applied to all benchmarks without re-calibration (separate from the main results in Table 1, which use domain-specific thresholds). The draft acceptance rate ranges from 23.2% on GSM8K-100 to 97.5% on GPQA-diamond, with MATH500-100 (33.8%) and AIME (74%) in between. Token acceptance rates follow the same ordering: GPQA-diamond (24.4%) > AIME2024/2025 (18.5%) > MATH500 (8.9%) > GSM8K (6.7%).

These differences reflect *cross-domain variations in CoE score distributions* under a fixed threshold rather than adaptive verifier behavior. Notably, GSM8K exhibits the lowest acceptance rate (23.2%), comparable to the calibration domain MATH500 (33.8%), suggesting *similar CoE score distributions* between these two math rea-

Algorithm 1 Prefix Acceptance Python-like Pseudocode

```

1 def Accept(Y, Z, gamma, Agg, scan):
2     # Y: drafted branches, Z: cached hidden
   ↪ states
3     # gamma: threshold; Agg: prefix aggregator
4     # scan in {REV, FWD}
5     S = ComputeCoEScores(Z) # per-prefix CoE
   ↪ scores
6     candidates = []
7
8     for j, branch in enumerate(Y):
9         a = AggregatePrefixes(S[j], Agg)
10        r = 0
11        k = len(branch)
12
13        if scan == "REV":
14            for t in range(k, 0, -1):
15                if a[t] >= gamma:
16                    r = t
17                    break
18        elif scan == "FWD":
19            for t in range(1, k + 1):
20                if a[t] >= gamma:
21                    r = t
22            else:
23                break
24
25        if r > 0:
26            candidates.append((a[r], r, j))
27
28    if not candidates:
29        return GreedyFallback(Y)
30
31    return SelectBest(candidates, Y)

```

soning tasks. This distributional similarity explains successful threshold transfer: the MATH500-calibrated threshold remains appropriately selective on GSM8K without re-calibration. In contrast, GPQA-diamond’s high acceptance (97.5%) indicates a substantial distributional shift where most drafted reasoning produces CoE scores above the MATH500-derived threshold. AIME (74%) shows intermediate behavior.

Accepted prefix lengths reflect reasoning granularity.

Average accepted prefix lengths range from 2.66 tokens (GSM8K) to 9.78 tokens (GPQA-diamond), with substantial variance within each benchmark ($\sigma = 1.2$ – 1.9 tokens). Longer accepted prefixes on GPQA and AIME indicate that the verifier commits to longer reasoning blocks when internal evidence is consistently strong, while shorter prefixes on GSM8K/MATH500 reflect more conservative, fine-grained verification. The distribution is right-skewed on all benchmarks, with occasional long acceptances (up to 10 tokens) when branch quality is exceptionally high.

Table 4: Verification overhead relative to drafting. Statistics are computed over all profiled iterations; $\rho \equiv t_{\text{verify}}/t_{\text{draft}}$.

Metric	Main run	Select-worst ablation
Iterations	45,960	19,989
Draft (ms)	206.6	191.7
Verify (ms)	25.3	24.0
$\bar{\rho}$	0.123	0.125
$\tilde{\rho}$	0.120	0.120
ρ_{95}	0.135	0.153

Relationship with correctness. We observe weak but consistent positive correlations between acceptance rate and correctness across benchmarks (Figure 4c,d). On GSM8K and MATH500, correct instances show slightly higher acceptance rates and longer prefixes than incorrect ones, suggesting that the CoE score partially captures reasoning quality. However, on GPQA and AIME, the distributions largely overlap, indicating that internal-state verification alone does not guarantee correctness—the verifier filters out clearly poor drafts but cannot fully distinguish subtle reasoning errors. This motivates combining lookahead with majority voting (self-consistency) to further improve robustness.

E Verification Profiling Supplement

Table 4 summarises per-iteration profiling for the production configuration described in the main text. We report the mean ($\bar{\rho}$), median ($\tilde{\rho}$), and 95th percentile (ρ_{95}) of the verification-to-draft wall-clock ratio over iterations where the draft time is positive. Both profiles use Qwen3-8B-Thinking on GSM8K-100; the second row enforces a select-worst ablation that always picks the lowest-scoring candidate.

F Accuracy Variance Tables

Tables 5 and 6 list the accuracy standard deviations corresponding to Table 1.

Table 5: Accuracy Standard Deviation (GSM8K / MATH500 / GPQA)

Method	GSM8K-100			MATH500-100			GPQA-diamond-100		
	Llama3.1 8B	Qwen3 8B	Qwen3 32B	Llama3.1 8B	Qwen3 8B	Qwen3 32B	Llama3.1 8B	Qwen3 8B	Qwen3 32B
Pass@1	±2.3	±0.9	±0.8	±2.9	±1.1	±1.0	±3.4	±3.1	±3.8
BoN (LL)	±1.9	±0.8	±0.5	±2.3	±1.1	±1.0	±2.7	±3.0	±3.3
BoN (CoE)	±2.1	±1.0	±0.6	±2.5	±0.8	±0.9	±3.0	±3.0	±3.6
SC	±1.7	±0.6	±0.5	±2.4	±0.9	±0.8	±2.9	±2.5	±2.6
Ours	±1.8	±0.5	±0.6	±2.0	±0.7	±0.6	±3.0	±3.0	±3.0

Table 6: Accuracy Standard Deviation (AIME2024 / AIME2025)

Method	AIME2024			AIME2025		
	Llama3.1 8B	Qwen3 8B	Qwen3 32B	Llama3.1 8B	Qwen3 8B	Qwen3 32B
Pass@1	±3.2	±4.1	±3.8	±1.4	±4.2	±5.3
BoN (LL)	±2.3	±4.6	±4.7	±0.6	±4.7	±0.0
BoN (CoE)	±2.3	±4.1	±3.5	±0.9	±4.5	±1.7
SC	±3.1	±2.4	±2.5	±1.3	±2.2	±1.9
Ours	±3.3	±2.4	±3.0	±1.0	±4.0	±3.0

G Complexity and Implementation Details

Asymptotic cost. Let m be the number of branches, k the lookahead length, L the number of layers, D the hidden width, and T the emitted length. The gate reuses hidden states already computed for drafting and adds only lightweight geometry on top, yielding an extra arithmetic cost of

$$\text{verify-FLOPs} = \mathcal{O}(m k L D),$$

with no additional forward passes and negligible parameter I/O.

Peak memory (KV and activations). Under a monotonic commit schedule (no rollback), the target path’s KV cache grows linearly with T , while branch caches exist only during one draft–verify iteration:

$$\text{peak-KV} = \mathcal{O}(T) + \mathcal{O}(m k).$$

This holds regardless of whether the acceptance rule is prefix-closed, as long as acceptance decisions are made once per iteration and the accepted block is appended atomically. By contrast, running N independent chains concurrently for Self-Consistency (SC) incurs $\mathcal{O}(N T)$ peak KV, or loses cache benefits if regenerated sequentially.

Cache reuse and scheduling. Within each iteration, we tile the target prefix cache to spawn m branches of length k , discard branch caches after verification, and commit only the accepted prefix $y_{1:r^*}$ to the target path. A reverse scan finds the longest acceptable prefix; the target KV advances monotonically and never rewinds.

Assumptions and edge cases. The bounds above assume (i) cached verification (no full-sequence fallback) and (ii) single-shot commits per iteration. Streaming with intermediate commits is compatible if the commit is atomic per iteration; rollback

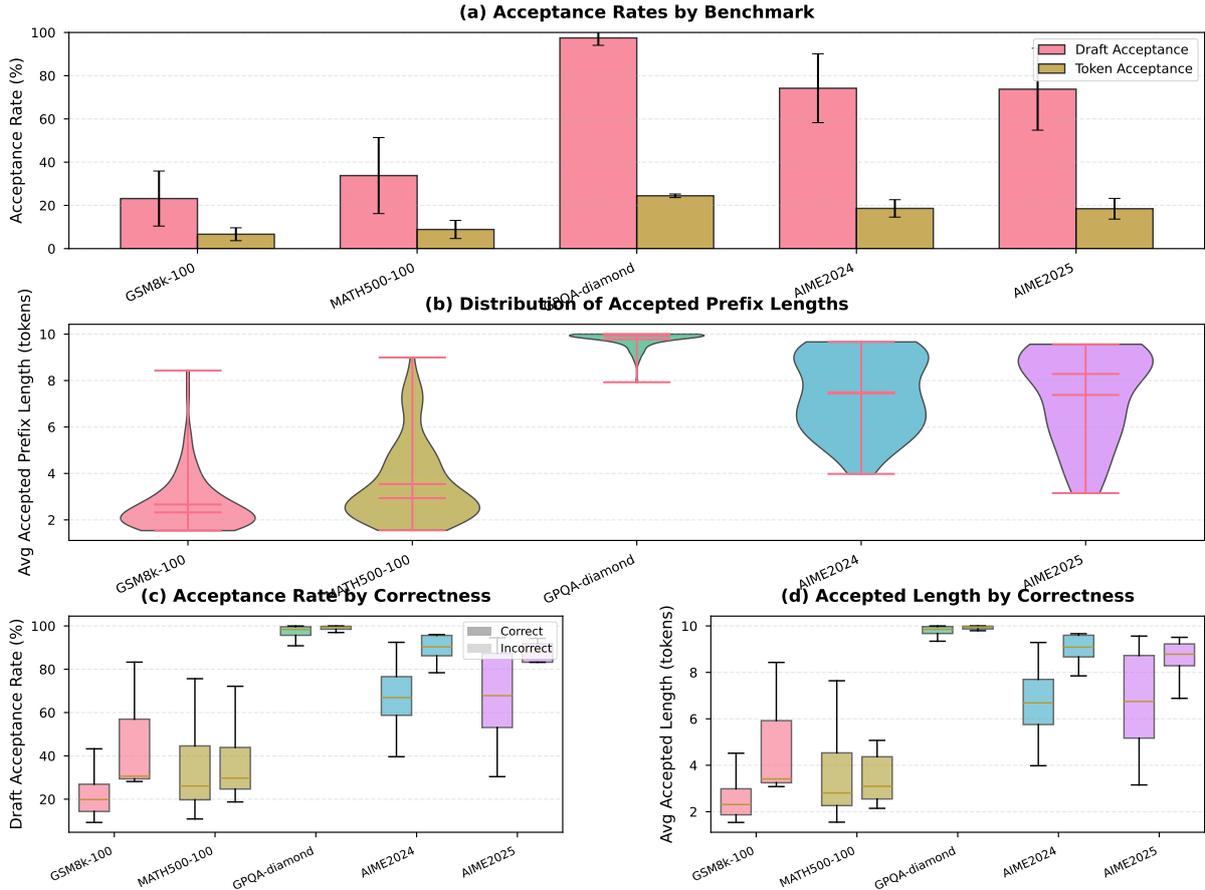


Figure 4: Acceptance analysis on Qwen3-8B-Thinking across five benchmarks. (a) Acceptance rates: draft acceptance (fraction of branches accepted) and token acceptance (fraction of drafted tokens accepted). (b) Distribution of accepted prefix lengths (violin plots show median, mean, and full distribution). (c,d) Acceptance rate and prefix length by correctness: correct instances (dark) vs. incorrect instances (light). The verifier is most selective on easier benchmarks (GSM8K) and accepts longer prefixes on challenging problems (GPQA, AIME).

would raise peak KV and is outside our scope. Implementation choices (FlashAttention v2, dynamic cache fallbacks, tokenizer templates) and the profiling summary are detailed in Appendix H.

H Implementation Notes

Profiling summary. On A100 (Qwen3-8B-Thinking, GSM8K-100), we measure the *relative verification overhead*

$$\rho = \frac{\text{time}(\text{verify})}{\text{time}(\text{draft})}$$

to be $\rho \approx 0.12$ on average (median 0.120, 95th percentile 0.135). Even under stress (frequent fallbacks), the 95th percentile remains ≈ 0.15 .

- **Hidden-state path.** Verification states are taken from `inputs_embeds` when supported, falling back to full-sequence computation only when required by the backend.

- **Cache policy.** Static KV caches are reused for the target path; we degrade gracefully when frameworks expose only dynamic caches.
- **Attention backend.** FlashAttention v2 is enabled whenever available and we fall back to standard kernels otherwise.
- **Tokenizer templates.** Chat models such as Qwen3 expose a native `<think>` template in their thinking mode to separate reasoning from answers.

I Comparison with Best-of-N + PRM Baseline

We additionally compare against Best-of-N with a trained Process Reward Model (PRM), a strong verifier-guided baseline, and select the highest-scoring candidate per problem. We use Qwen2.5-Math-PRM-7B as the verifier with $N=4$ candi-

dates, since no Qwen3-native PRM is currently available.

Table 7: Comparison of CoE vs. Best-of-4 + PRM (Qwen3 series). Benchmark names denote their 100-sample subsets (e.g., GSM8K means GSM8K-100). Bold indicates the better result.

Model	Method	GSM8K	MATH	GPQA	AIME24	AIME25
8B	Ours	98.0	92.0	61.0	77.8	70.7
	PRM	97.0	90.0	57.0	66.7	73.3
32B	Ours	98.0	93.0	70.5	83.3	70.6
	PRM	98.0	91.0	71.1	76.7	63.3

Discussion. The methods differ in their requirements:

- **Our method** is training-free and uses only the base model. It reuses hidden states already produced during generation and adds lightweight geometric operations, with verification overhead $\rho \approx 0.12$ relative to drafting time.
- **Best-of-N + PRM** requires an additional trained verifier model (7B parameters in our case) and extra forward passes per problem, increasing both compute cost and deployment complexity.

These results demonstrate that our proposed method is a lightweight alternative to PRM-based verification, competitive on the MATH benchmark while requiring no additional training or external models.

J Experimental Prompts

For reproducibility, we provide the prompt formats used in our experiments. When running Qwen3-8B/32B in thinking mode (denoted Qwen3-8B/32B-Thinking), we enable the native `<think>...</think>` template, which automatically wraps reasoning without additional instructions.

J.1 Prompt Format

All benchmarks use a simple conversational format with no system prompt. We leverage Qwen3-series native rationale template, which structures responses with explicit reasoning followed by a final answer. The model generates:

1. A `<think>` section containing step-by-step reasoning
2. A `</think>` closing tag

3. A final answer section with the solution

The prompt structure is minimal:

User Prompt Format

```
Please reason step by step, and put your final answer within
\boxed{ }.
<question>
```

J.2 Benchmark-Specific Examples

J.2.1 GSM8K

User Prompt

Janet’s ducks lay 16 eggs per day. She eats three for breakfast every morning and bakes muffins for her friends every day with four. She sells the remainder at the farmers’ market daily for \$2 per fresh duck egg. How much in dollars does she make every day at the farmers’ market?

Model Response

```
<think>
Let me break this down step by step. Janet starts with 16
eggs. She eats 3 for breakfast, leaving 13 eggs. Then she
uses 4 for muffins, leaving 9 eggs. She sells these 9 eggs
at $2 each, so  $9 \times 2 = 18$ .
</think>
She makes $18 per day.
```

J.2.2 MATH500

User Prompt

Convert the point $(0, 3)$ in rectangular coordinates to polar coordinates. Enter your answer in the form (r, θ) , where $r > 0$ and $0 \leq \theta < 2\pi$.

Model Response

```
<think>
To convert from rectangular to polar coordinates, I use
 $r = \sqrt{x^2 + y^2}$  and  $\theta = \arctan(y/x)$ . Here  $x = 0$  and
 $y = 3$ , so  $r = \sqrt{0^2 + 3^2} = 3$ . For the angle, since  $x = 0$ 
and  $y > 0$ , the point lies on the positive  $y$ -axis, which
corresponds to  $\theta = \pi/2$ .
</think>
The answer is  $(3, \frac{\pi}{2})$ .
```

J.2.3 GPQA-diamond.

User Prompt

Among the following exoplanets, which one has the highest density?

- a) An Earth-mass and Earth-radius planet.
- b) A planet with 2 Earth masses and a density of approximately 5.5 g/cm^3 .
- c) A planet with the same composition as Earth but 5 times more massive than Earth.
- d) A planet with the same composition as Earth but half the mass of Earth.

A. a B. b C. c D. d

Model Response

<think>

Density is mass divided by volume. For option (a), Earth-mass and Earth-radius gives Earth's density ($\sim 5.5 \text{ g/cm}^3$). Option (b) explicitly states 5.5 g/cm^3 . For option (c), if composition is the same but mass is $5\times$ larger, the radius increases by $5^{1/3} \approx 1.71$, so volume increases by $5^{1/3 \times 3} = 5$. Thus density remains $\sim 5.5 \text{ g/cm}^3$. For option (d), half the mass means radius decreases by $0.5^{1/3} \approx 0.79$, so volume becomes 0.5 , and density stays $\sim 5.5 \text{ g/cm}^3$. Wait, that doesn't help distinguish them. Let me reconsider... Actually, for Earth-like composition, more massive planets compress more under gravity, increasing density. So option (c) with $5\times$ Earth's mass will have higher density than Earth.

</think>

The answer is .

J.2.4 AIME.

User Prompt

Among the 900 residents of Aimeville, there are 195 who own a diamond ring, 367 who own a set of golf clubs, and 562 who own a garden spade. In addition, each of the 900 residents owns a bag of candy hearts. There are 437 residents who own exactly two of these things, and 234 residents who own exactly three of these things. Find the number of residents of Aimeville who own all four of these things.

Model Response

<think>

Let me use inclusion-exclusion principle. Let D , G , S , and C denote diamond ring, golf clubs, garden spade, and candy hearts respectively. We know $|C| = 900$ (everyone has candy). We have $|D| = 195$, $|G| = 367$, $|S| = 562$. Let n_k be the number of people owning exactly k items. We're given $n_2 = 437$ and $n_3 = 234$. We want to find n_4 .

Since everyone has candy, the question reduces to finding how many own all of $\{D, G, S, C\}$. Using the counting equations and the constraint that $\sum n_k = 900$, we can set up: $n_0 + n_1 + n_2 + n_3 + n_4 = 900$, which gives $n_0 + n_1 + 437 + 234 + n_4 = 900$, so $n_0 + n_1 + n_4 = 229$. Additionally, counting total items: $1 \cdot n_1 + 2 \cdot n_2 + 3 \cdot n_3 + 4 \cdot n_4 = 195 + 367 + 562 + 900 = 2024$. Substituting known values and solving yields $n_4 = 108$.

</think>

The answer is .