REVISITING LOSSY VERIFICATION IN SPECULATIVE DECODING: MECHANISMS, TRADE-OFFS, AND FAILURE MODES

Anonymous authors
Paper under double-blind review

ABSTRACT

Speculative Decoding (SD) accelerates large language model inference by allowing a lightweight draft model to propose tokens that are subsequently verified in parallel by a larger target model. Recent approaches introduce lossy verification schemes to further improve efficiency by relaxing strict distributional matching. Yet such relaxation silently rewrites the decoding distribution, and the resulting acceleration can come at the cost of unstable, sometimes severely degraded generation quality. In this work, we present a principled analysis of the distributions induced by lossy verification methods. We show that many seemingly distinct approaches differ only superficially and can be unified into two categories: truncation-based verification and collaborative verification. We further construct a diagnostic evaluation framework across curated benchmarks. For truncation-based methods, we identify a fundamental pitfall: performance can degrade significantly compared to the true truncation sampling baseline due to distributional distortion. For collaborative verification, we reveal that well-designed relaxation principles, namely overshoot suppression and supervision quality, matter far more than the linear interpolation between draft and target. Our code is available at `https://anonymous.4open.science/r/Fast-HSD-E6AD`.

1 INTRODUCTION

Auto-regressive Large Language Models (LLMs) Achiam et al. (2023); Touvron et al. (2023); Bai et al. (2023) are at the forefront of the AI revolution. Despite their strong performance, their non-parallelizable inference presents a significant efficiency bottleneck, particularly for long-context generation in test-time scaling OpenAI et al. (2026); Guo et al. (2025), LLM agents Yao et al. (2022); Schick et al. (2023); Significant-Gravitas (2023), and multimodal reasoning Peng et al. (2025). Speculative Decoding (SD) Leviathan et al. (2023) mitigates this challenge by having a lightweight draft model generate candidate tokens, which are then verified by the target model in parallel, reducing expensive forward passes while ensuring the generated distribution matches that of the target model.

While theoretical advances have pushed the limits of SD Sun et al. (2024); Zhou et al. (2026), the strict requirements for distribution matching continue to limit the potential for further speedup. Lossy verification methods Cai et al. (2024); Narasimhan et al. (2024); Zhou et al. (2024); Fu et al. (2025); Leviathan et al. (2023) relax these requirements for greater acceleration, but existing work overstates their advantages under curated settings (selective hyperparameters or easy benchmarks). This obscures the true speed–quality trade-offs arising from distortion of the target distribution and leaves no clear comparison across methods, hampering the broader adoption of these methods and further advancements in the field.

In this work, we present a principled analysis of the distributions induced by lossy verification methods, resulting in a precise categorization of them. This analysis reveals that the apparent differences between methods are largely superficial, with most falling into two distinct categories.

First, methods such as typical acceptance in Medusa Cai et al. (2024) and SpecCascade Narasimhan et al. (2024) accept draft tokens as long as they fall within the allowed set defined by truncation-based sampling methods, specifically η-sampling Hewitt et al. (2022) and min-p sampling Nguyen et al. (2024). We refer to these as *truncation-based verification* methods. Second, we show that

Reviewers: please read the Reviewer Guidelines (iclr.cc/Conferences/2027/ReviewerGuidelines) and the AI Policy for Reviewers (iclr.cc/Conferences/2027/AIPolicyForReviewers). If you used AI to expand, edit, or polish your review, please provide the input text to the LLM. Better still, consider skipping the LLM and submitting your original text: we, and the authors, are much more interested in your unedited thoughts than in what an LLM has to say. AI-assisted or not, you are putting your name and reputation behind your review: LLM-generated falsehoods, hallucinations or misrepresentations are subject to disciplinary action, which may include desk-rejecting all papers you have authored.

interpolation-based, lenience-based, and judge-based relaxations essentially use weighted linear combinations of the target and draft distributions to relax the standard acceptance rule, as summarized in Figure 2. Following the terminology of CoS Fu et al. (2025), we group these as *collaborative verification* methods.

We first examine truncation-based verification methods against their corresponding baselines, i.e., directly using the same truncation sampling strategies for the target model. Our experiments show that the seemingly comparable performance reported in prior work Cai et al. (2024); Narasimhan et al. (2024) largely arises from improved baseline performance induced by their adopted truncation sampling methods. Moreover, the performance gap between truncation-based verification and the true baseline grows with task difficulty, as shown in Figure 1 for SpecCascade. As harder tasks better reflect real-world scenarios, this widening gap highlights potential pitfalls of truncation-based verification in practice. This pitfall is further amplified when lossy verification methods are integrated into the EAGLE-3 speculative decoding system Li et al. (2025), where such a performance gap becomes even more pronounced.

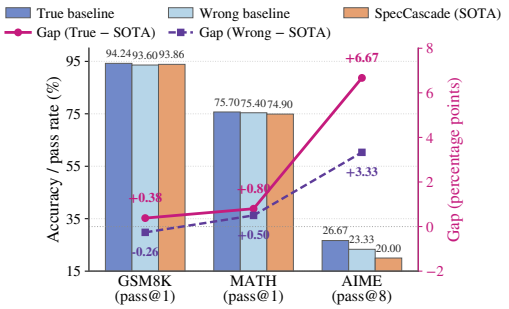


Figure 1: The accuracy gap between the lossless baseline and truncation-based verification grows with task difficulty (+0.38 pp on GSM8K to +6.67 pp on AIME). **True baseline**: target model with min-p sampling; **Wrong baseline**: target model without min-p.

We next turn to collaborative verification, where we identify a key principle for a better speed–quality trade-off: existing methods interpolate the draft and target distributions either uniformly in interpolation-based relaxation, adaptively in lenience-based relaxation, or through learned, context-conditioned decisions in judge-based relaxation. We find that interpolation is not what preserves generation quality: selectively suppressing the draft at overshoot tokens or a well-trained judge is sufficient.

In summary, we make these contributions:

- **Unified Framework**: Unifying seemingly different lossy verification methods into two categories: truncation-based verification and collaborative verification, by reviewing their underlying mechanisms.
- **Empirical Pitfalls**: Identifying a key pitfall in truncation-based verification: distributional distortion can significantly degrade performance relative to the true truncation sampling baseline.
- **Governing Principle**: In collaborative verification, instead of simple linear interpolation between the draft and target probability, overshoot suppression or a well-trained judge as a surrogate acceptance rule can significantly improve the efficiency-performance trade-off.

2 Preliminaries

We first introduce the speculative decoding framework and its lossless verification mechanism (Section 2.1), which forms the foundation for many lossy methods. We then review two decoding techniques: collaborative decoding (Section 2.2) and truncation sampling (Section 2.3). As shown in Section 3, these techniques underlie the two categories of lossy verification.

2.1 Speculative Decoding

Let p and q denote the next-token distributions of the target and draft models, respectively, over a shared vocabulary $\mathcal{V}$. In Speculative decoding (Leviathan et al., 2023), each draft token x sampled from q is accepted with probability:

$$h(x) = \min\left(1,\ \frac{p(x)}{q(x)}\right). \tag{1}$$

If the draft token x is rejected, a replacement token is resampled from the residual distribution $\big(p(x) - \min\{p(x),\, q(x)\}\big)_+$, renormalized over $\mathcal{V}$, ensuring the generated distribution equals the target p exactly. Follow-up work increases acceptance rates by proposing multiple candidates or

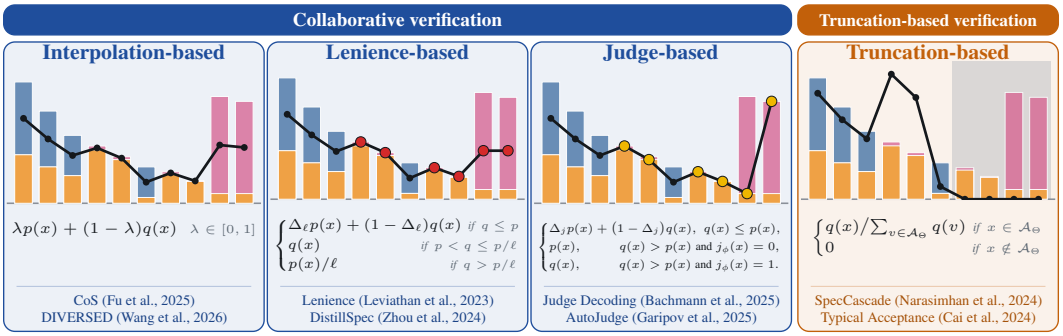


Figure 2: Mechanism-level unification of lossy verification. Each panel plots the same target/draft pair over ten tokens and the distribution r the method actually samples from. Legend: target $p(x)$, draft $q(x)$, overlap $\min(p, q)$, resulting distribution, and tokens outside the allowed set $\mathcal{A}_\Theta$. Markers flag the tokens where a method departs from plain interpolation: lenience and judge. See Section A for detailed derivations.

tree-structured drafts (Sun et al., 2023; Miao et al., 2024; Yang et al., 2024; Li et al., 2025; Fan et al., 2026), while block-level verification extends lossless guarantees to sequences of tokens (Sun et al., 2024; Zhou et al., 2026). As lossless verification approaches its theoretical limits, lossy approaches achieve further speedup by modifying the acceptance criterion: some relax the reference distribution in Eq. equation 1 (Leviathan et al., 2023; Zhou et al., 2024; Fu et al., 2025), while others gate acceptance via a set-membership criterion derived from truncation sampling (Cai et al., 2024; Narasimhan et al., 2024). Despite these advances, the underlying mechanisms of lossy methods and their speed–quality trade-offs remain poorly understood, which we address in Section 3.

2.2 Collaborative Decoding

Collaborative decoding combines the predictions of multiple models to form a reshaped next-token distribution. Using the target distribution p and draft distribution q as defined in Section 2.1, and letting $\lambda \in [0, 1]$ denote an interpolation coefficient, **Weighted Ensembling (WE)** Huang et al. (2024); Yao et al. (2024) computes the next-token probability as a convex mixture:

$$p_{\mathrm{WE}}(x) = \lambda\, p(x) + (1 - \lambda)\, q(x). \tag{2}$$

Contrastive Decoding (CD) Li et al. (2023); O'Brien & Lewis (2023) instead reweights p using a contrastive factor derived from q:

$$p_{\mathrm{CD}}(x) \;=\; \frac{p(x)/q(x)^\lambda}{\sum_{v \in \mathcal{V}} p(v)/q(v)^\lambda}. \tag{3}$$

Whereas standard speculative decoding aims to match the target distribution, **Collaborative Decoding via Speculation (CoS)** Fu et al. (2025) instead uses one of these combined distributions as the verification target, yielding additional speedup.

2.3 Truncation Sampling

Truncation sampling restricts the vocabulary at each decoding step to an allowed set $\mathcal{A}_\Theta \subseteq \mathcal{V}$ determined by a truncation strategy Θ, discarding low-probability tokens to reduce the risk of incoherent outputs. Let $Z_\Theta(p) = \sum_{v \in \mathcal{A}_\Theta} p(v)$ denote normalization constant. Resulting distribution is

$$p_\Theta(x) = \begin{cases} \dfrac{p(x)}{Z_\Theta(p)}, & x \in \mathcal{A}_\Theta, \\ 0, & x \notin \mathcal{A}_\Theta. \end{cases} \tag{4}$$

Two representative strategies, **min-p sampling** Nguyen et al. (2024) and **η-sampling** Hewitt et al. (2022), are described below:

Min-p sampling adapts to the top candidate's confidence by applying a dynamic cutoff relative to the peak probability, where $p_{\text{base}} \in (0, 1)$:

$$\mathcal{A}_{\text{min-p}} = \{x \in \mathcal{V} : p(x) \geq p_{\text{base}} \cdot \max_{v \in \mathcal{V}} p(v)\} \tag{5}$$

η-sampling adapts to model uncertainty by removing tokens below a threshold derived from the distribution entropy $H(p) = -\sum_{v \in \mathcal{V}} p(v) \log p(v)$, where $\varepsilon, \delta > 0$:

$$\mathcal{A}_{\eta} = \{x \in \mathcal{V} : p(x) \geq \min\left(\varepsilon,\, \delta e^{-H(p)}\right)\} \tag{6}$$

In Section 3, we show that state-of-the-art lossy verification methods Narasimhan et al. (2024); Cai et al. (2024) essentially accept a draft token as long as it falls within the allowed set defined by one of these truncation strategies.

3 Unifying Lossy Verification

In this section, we show that existing lossy verification methods fall into two paradigms: **collaborative verification**, which relaxes verification by matching the combined distribution from collaborative decoding, and **truncation-based verification**, which accepts draft tokens based on the allowed sets induced by truncation sampling.

3.1 Collaborative Verification

Collaborative verification relaxes the acceptance criterion so that the generated distribution becomes a combination of the draft and target distributions, improving decoding speed at the cost of degraded performance. This view covers both (i) methods that explicitly construct the combined distribution Fu et al. (2025), (ii) methods whose acceptance rule implicitly induces such a distribution Leviathan et al. (2023), (iii) methods whose learned verifiers selectively retain otherwise rejected draft tokens Bachmann et al. (2025).

Interpolation-based relaxation modifies the acceptance probability of the lossless verification to:

$$h(x) \;=\; \min\left(1, \frac{\lambda\, p(x) + (1-\lambda)\, q(x)}{q(x)}\right). \tag{7}$$

Under this rule, the resulting token distribution becomes a convex mixture of the target and draft distributions (see Figure 2):

$$P(\text{generate } x) = \lambda\, p(x) + (1-\lambda)\, q(x). \tag{8}$$

We show that CoS Fu et al. (2025), DIVERSED Wang et al. (2026), and Fuzzy SD Holsman et al. (2025) share the same interpolation-based relaxation form and differ primarily in how the mixture coefficient is determined (see Sections A.1 and A.2 for details).

Lenience-based relaxation modifies the acceptance rule with a lenience factor $\ell \in (0, 1]$:

$$h(x) \;=\; \min\left(1, \frac{p(x)}{\ell\, q(x)}\right). \tag{9}$$

The induced distribution admits the following decomposition (see Section A.3 for the proof):

$$P(\text{generate}\, x) = \begin{cases} \Delta_\ell p(x) + (1-\Delta_\ell) q(x), & q(x) \leq p(x), \\ q(x), & p(x) < q(x) \leq p(x)/\ell, \\ p(x)/\ell, & q(x) \geq p(x)/\ell. \end{cases} \tag{10}$$

where $\Delta_\ell = \left(\frac{1}{2}\sum_{v \in \mathcal{V}} |q(v) - p(v)/\ell| + \frac{1}{2} - \frac{1}{2\ell}\right) \Big/ \left(\frac{1}{2}\sum_{v \in \mathcal{V}} |q(v) - p(v)|\right)$.

This decomposition reveals an adaptive correction mechanism: when the draft *underestimates* the target ($q \leq p$), interpolation is applied; moderate overestimation ($p < q \leq p/\ell$) is left unchanged (pure q); and only the *severe* overshoot region ($q \geq p/\ell$) is corrected (see Figure 2).

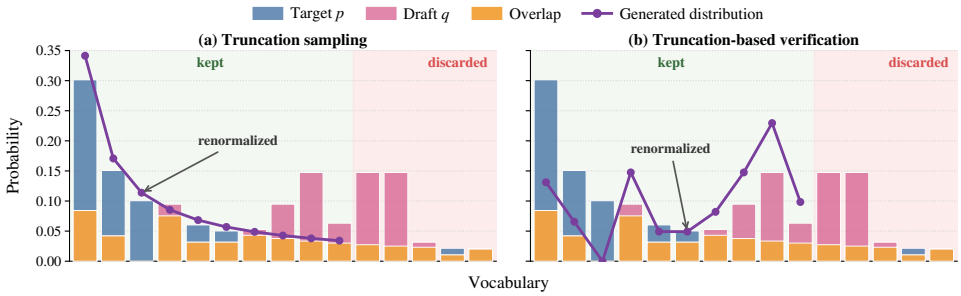


Figure 3: Comparison of distributions induced by truncation sampling and truncation-based verification. Tokens are sorted by decreasing target probability. Blue and pink bars denote the target distribution p and draft distribution q, respectively; orange denotes their overlap. The purple line shows the output distribution.

Intuitively, the ceiling p/ℓ term in Equation (10) caps the generation probability when the draft distribution is overconfident. This ceiling prevents the draft from dominating the output at tokens where it assigns probability far beyond what the target distribution p would allow, even after relaxation by the lenience factor ℓ. Separately, $\frac{1}{2}\sum_{v\in\mathcal{V}}|q(v)-p(v)|$ is the total variation distance between q and p, which measures the total mismatched probability mass between the draft and target distributions. We provide a detailed interpretation in Section B.

We reveal that Lenience (Leviathan et al., 2023) adopts the linear relaxation p/ℓ. DistillSpec (Zhou et al., 2024) extends this framework by exploring alternative relaxation forms, including p/ℓ^2 and p^ℓ.

Judge-based relaxation relaxes standard speculative decoding with a learned judgment signal. Let $s_\phi(x) \in \mathbb{R}$ denote the confidence that a learned judge assigns to draft token x, and let $j_\phi(x) = \mathbb{1}[s_\phi(x) \geq \delta] \in \{0,1\}$ indicate whether the judge predicts that x should be accepted, where $\delta \in [0,1]$. Combining with Equation (1) gives

$$h(x) = \begin{cases} 1, & q(x) \leq p(x), \\ \dfrac{p(x)}{q(x)} + \left(1 - \dfrac{p(x)}{q(x)}\right) j_\phi, & q(x) > p(x). \end{cases} \tag{11}$$

Thus, the resulting token distribution is:

$$P(\text{generate}\, x) = \begin{cases} \Delta_j p(x) + \big(1-\Delta_j\big)q(x), & q(x) \leq p(x), \\ p(x), & q(x) > p(x) \text{ and } j_\phi(x) = 0, \\ q(x), & q(x) > p(x) \text{ and } j_\phi(x) = 1. \end{cases} \tag{12}$$

where $\Delta_j = \big(\sum_{v\in\mathcal{V}}\big(q(v)-p(v)\big)_+\big(1-j(v)\big)\big)\big/\big(\sum_{v\in\mathcal{V}}\big(p(v)-q(v)\big)_+\big)$. see Section F.1 for detailed derivation.

We show that Judge Decoding (Bachmann et al., 2025) and AutoJudge (Garipov et al., 2025) share the same judge-based relaxation form and differ in how the judge is trained (see Section F.2 for detailed experiment implementation).

In summary, collaborative verification methods differ primarily in how they handle draft-model overestimation ($q(x) > p(x)$). Interpolation-based relaxation continues to blend $p(x)$ and $q(x)$ using the same form in the underestimation region. Lenience-based relaxation retains $q(x)$ under moderate overestimation but caps severe overshoot at $p(x)/\ell$. Judge-based relaxation instead uses a learned judgement signal to determine whether the overestimated draft contribution should be retained.

3.2 Truncation-based Verification

We formally define truncation-based verification. Within this framework, we distinguish two principal approaches: **SpecCascade** Narasimhan et al. (2024) and **typical acceptance** Cai et al. (2024).

Definition 1 (Truncation-based verification)**.** Let $\mathcal{A}_\Theta \subseteq \mathcal{V}$ denote the allowed set of a truncation strategy Θ. Truncation-based verification replaces the acceptance rule in Equation (1) with $h(x) = \mathbb{1}\{x \in \mathcal{A}_\Theta\}$: a draft token is accepted if it lies in $\mathcal{A}_\Theta$, and tokens outside this set are rejected.

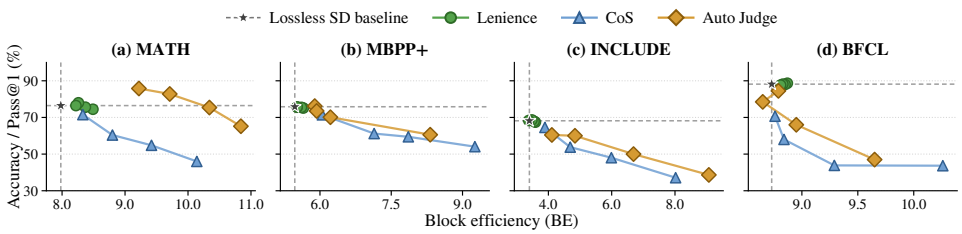


Figure 4: Efficiency and task performance tradeoff for collaborative verification across four benchmarks (Qwen2.5-72B/0.5B-Instruct-GPTQ-Int8). Each marker represents one hyperparameter setting, and lines connect settings in sweep order. Full results are in Section H; the Llama pair and extended analysis are in Section D.

The resulting distribution is the renormalized draft distribution restricted to $\mathcal{A}_\Theta$, where $Z_\Theta(q) = \sum_{v \in \mathcal{A}_\Theta} q(v)$ is the normalization constant:

$$P(\text{generate } x) = \begin{cases} q(x)/Z_\Theta(q), & x \in \mathcal{A}_\Theta, \\ 0, & x \notin \mathcal{A}_\Theta. \end{cases} \tag{13}$$

Specifically, SpecCascade (Narasimhan et al., 2024) and Medusa (Cai et al., 2024) are examples of truncation-based verification, with allowed sets defined by min-p (Equation (5)) and η-sampling (Equation (6)), respectively. Their reported gains over standard speculative decoding conflate the effect of verification with the effect of truncation sampling itself. When evaluated against the appropriately truncated target, the actual performance gap becomes clear, as shown in Section 5.

Having characterized existing lossy verification into collaborative and truncation-based verification, we now evaluate both paradigms empirically. As foreshadowed in the introduction, benchmark difficulty is itself diagnostic: easy tasks such as GSM8K mask the distortions introduced by lossy verification We therefore evaluate on four harder benchmarks spanning distinct domains: MATH Hendrycks et al. (2021) for mathematical reasoning, MBPP+ Liu et al. (2023) for code generation, INCLUDE (Romanou et al., 2024) for multilingual understanding, and BFCL (Patil et al., 2025) for agentic tool use. For efficiency we report Block Efficiency (BE), the number of tokens accepted per step, and Decoding Speed (DS), tokens per second; for task quality we report Accuracy on MATH, INCLUDE, and BFCL and Pass@1 on MBPP+. Configurations are given in Section C.

4 Identifying the Key Factor in Collaborative Verification

As analyzed in Section 3.1, Lenience-based, interpolation-based, and judge-based relaxation all fall under the broader category of collaborative verification. We use Lenience Leviathan et al. (2023), CoS Fu et al. (2025), and AutoJudge Garipov et al. (2025) as representative methods of these three subclasses, respectively.

4.1 Overshoot Suppression Improves Performance

Both CoS and AutoJudge exhibit a clear efficiency-performance tradeoff once relaxation becomes aggressive (see Figure 4). As illustrated by Equations (8) and (12), the generated distribution increasingly leans toward the draft distribution, and the relaxed acceptance rule leads to degraded task performance. In contrast, we observe that lenience-based relaxation maintains strong task performance while achieving slightly better efficiency than the baseline (Figure 4). Since $q(x)$ is effectively the worst-case of overestimation region ($q > p$), the gain from lenience must come from either the adaptive interpolation in the underestimated region or the thresholding in the overshooting region.

To disentangle these contributions, we further conduct an ablation study on MBPP+ to investigate the key factor behind the effectiveness of lenience-based relaxation. We selectively replace the generated distribution of CoS in the $q(x) < p(x)$ and $q(x) > p(x)/\lambda$ regions with either ceiling the overshoot region or the adaptive interpolation in the underestimated region, and then measure the resulting

changes in efficiency and task performance tradeoff. As shown in Table 1, introducing adaptive interpolation in the undershoot region has similar severe tradeoff as CoS. In contrast, ceiling the overshooting region alone achieves task performance comparable to lossless verification, suggesting that ceiling is its primary contribution.

In summary, the ablation identifies the overshoot region, not the underestimated region, as the source of lenience-based relaxation's effectiveness: with the overshoot ceiling alone, task performance matches lossless verification while efficiency still improves, whereas the adaptive interpolation in the underestimated region inherits CoS's severe trade-off. This aligns with recent findings that low-quality generations arise primarily from overshoot tokens Zhou et al. (2025); Yue et al. (2025); Fan et al. (2026), and points to overshoot suppression, rather than interpolation, as the key design principle for effective collaborative verification.

4.2 A Well-Trained Judge Is Promising

AutoJudge achieves the best trade-off among the collaborative methods. It outperforms CoS over most of the sweep on every benchmark, and on MATH it beats even the lossless baseline across a wide BE range for both model pairs Figures 4 and 8b. Its weakness is stability: the sweep is non-monotone on some benchmarks, and the margin over CoS varies by domain. We attribute this pattern to the judge's supervision rather than to its resulting distribution Equation (12). We train a single judge on GSM8K and LiveCodeBench mismatches according to the official release (Section F.2). GSM8K contributes 7,473 problems against LiveCodeBench's 880, so the training data is mostly mathematical reasoning, and MATH is where the judge beats CoS by the widest margin.

Table 1: Ablation on collaborative verification (MBPP+). BE: block efficiency.

Yielded rule	λ	BE↑	P@1↑
$\begin{cases} q + \Delta_\ell(p-q), & q \le p \\ \lambda p + (1-\lambda)q, & q > p \end{cases}$	0.2	9.15	50.26
	0.4	7.93	56.61
	0.6	6.80	60.85
	0.8	5.95	66.14
$\begin{cases} \lambda p + (1-\lambda)q, & q \le p/\lambda \\ p/\lambda, & q > p/\lambda \end{cases}$	0.2	5.59	75.93
	0.4	5.57	75.13
	0.6	5.54	75.66
	0.8	5.54	75.13

To test this directly, we sweep the training data mixture from math-dominated (9:1) to code-dominated (1:9). As shown in Table 2, interestingly, the math-dominated judge outperforms the code-dominated one on every benchmark, including MBPP+: upweighting LiveCodeBench does not improve the one benchmark closest to it. The math-heavy judges also stay usable across all four benchmarks, whereas the code-heavy ones trade task performance for acceptance.

In conclusion, the mixture sweep isolates supervision quality as what governs judge behavior, which makes judge-based relaxation promising. Unlike interpolation-based and lenience-based relaxation, whose induced distributions are fixed in closed form, the judge's ceiling is set by its supervision and moves as the supervision improves. With sufficient supervision it already beats lossless SD on both efficiency and task performance, so improving judge supervision is a concrete direction for extending that result beyond a single domain.

Table 2: Mixture-ratio sweep. Each cell averages over the acceptance thresholds $\delta \in \{0.2, 0.4, 0.6, 0.8\}$.

Mix	MATH		MBPP+		INCLUDE		BFCL	
	BE	Acc	BE	P@1	BE	Acc	BE	Acc
9:1	8.62	**83.10**	5.82	**76.72**	3.90	**62.96**	8.89	**88.00**
7:3	9.67	73.95	6.65	69.84	6.50	59.09	9.39	77.38
5:5	9.67	74.55	7.60	60.12	7.12	47.50	9.53	77.00
3:7	9.71	73.40	6.66	71.63	6.98	50.91	9.49	82.50
1:9	10.55	64.45	8.34	57.87	8.89	43.52	9.92	63.00

5 Revealing the Pitfall in Truncation-Based Verification

We re-evaluate two representative truncation-based verification methods, i.e., typical acceptance (Cai et al., 2024) and SpecCascade (Narasimhan et al., 2024), against their matched truncation sampling baselines (η-sampling and Min-p sampling, respectively) directly applied on the target model with lossless verification (Leviathan et al., 2023). This setup isolates whether the reported gains stem from the verification mechanism itself or from the underlying truncation sampling. Single-draft

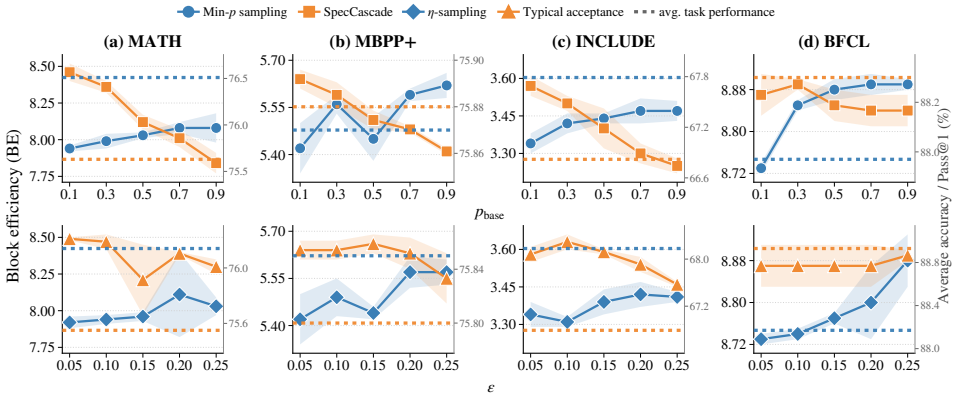


Figure 5: Block efficiency and task performance of truncation-based methods using the Qwen2.5-72B and Qwen2.5-0.5B pair across four benchmarks. *Top row*: Min-p sampling and SpecCascade. *Bottom row*: η-sampling and typical acceptance. Solid lines report mean block efficiency (BE). Colored dotted horizontal lines report accuracy or Pass@1, averaged over hyperparameter settings.

experiments are conducted within the standard SD framework Leviathan et al. (2023), and multi-draft experiments within EAGLE-3 Li et al. (2025).

5.1 Standard SD

Across multiple benchmarks (Figure 5), truncation-based verification exhibits a clear performance gap relative to its matched baseline under standard speculative decoding. Min-p sampling and η-sampling outperform SpecCascade and typical acceptance respectively. As formalized in Definition 1, this degradation arises because tokens within the allowed set are always accepted, producing a distribution that mirrors the draft model rather than the target distribution (see Figure 3). Overall, these results reinforce our claim that the apparent gains of truncation-based verification are largely driven by truncation itself, while the verification mechanism can further distort the decoding distribution and reduce task performance.

Beyond the performance gaps, we investigate truncation sampling effect in Lemma 1. It decomposes the truncation-induced change in acceptance probability into a non-negative *gain* term, accumulated over the retained support A, and a *loss* term, accumulated over the discarded region at the tail (proof see Section G). $\Delta\mathrm{BE}$ thus determines whether truncation sampling hurts SD efficiency.

Lemma 1 (Truncation sampling efficiency effect)**.** *Denote allowed set $\mathcal{A}_\Theta \subseteq \mathcal{V}$ determined by a truncation strategy Θ, $z = \sum_{v\in\mathcal{A}_\Theta} p(v)$. The change in single-token acceptance probability induced by truncation sampling satisfies*

$$\Delta\mathrm{BE} = \sum_{x\in\mathcal{A}_\Theta} \min\left\{(q(x)-p(x))_+,\ (\tfrac{1}{z}-1)p(x)\right\} - \sum_{x\notin\mathcal{A}_\Theta} \min\big(p(x),q(x)\big). \tag{14}$$

To make this concrete, Figure 6 plots the two terms from Lemma 1 across a range of distribution pairs. Under min-p sampling, the gain dominates at smaller p_{base}, but the margin narrows as p_{base} approaches 0.9, where the discarded mass begins to outweigh the redistributed gain. Under η-sampling, $\Delta\mathrm{BE} > 0$ uniformly across all distribution pairs and ϵ, and the net gain grows monotonically with ϵ despite wider confidence bands. This indicates that η-sampling reliably converts mass from the truncated tail into additional acceptance probability on the retained support, whereas min-p does not.

The contrasting trends in Figure 5 further show that the benchmark experiments are consistent with both the theoretical analysis in Lemma 1 and the simulation in Figure 6. The lemma predicts that the efficiency effect of truncation depends on the balance between the gain and the loss. This balance is reflected in the simulation: min-p sampling yields positive gains at moderate thresholds but becomes less favorable as p_{base} approaches 0.9, whereas η-sampling maintains positive $\Delta\mathrm{BE}$ across the tested range. The benchmark results follow the same pattern: min-p sampling rises to a peak and then declines, while η-sampling increases more consistently, with only minor deviations. Overall, these aligned trends support the theoretical decomposition and the controlled simulation.

Table 3: Evaluation results using LLaMA-3.1 8B and its official released pair under EAGLE-3, averaged across the parameter grid for each method. **Bold** denotes the better task performance within each matched pair. Gray denotes the EAGLE-3 baseline, which is not directly comparable.

Method	MATH			MBPP+			INCLUDE			BFCL		
	BE	DS	Acc (%)	BE	DS	Pass@1 (%)	BE	DS	Acc (%)	BE	DS	Acc (%)
EAGLE-3 (Li et al., 2025)	3.76	140.21	73.20	4.70	167.83	59.30	0.67	45.72	35.50	2.57	85.27	86.00
Min-p sampling (Nguyen et al., 2024)	3.98	143.22	**76.88**	4.90	168.57	**61.80**	0.56	45.80	**37.00**	2.58	84.26	**86.60**
SpecCascade (Narasimhan et al., 2024)	4.22	152.74	76.16	5.00	175.91	60.74	0.78	48.13	33.27	2.59	85.20	85.40
η-sampling (Hewitt et al., 2022)	3.93	140.05	**76.12**	4.87	165.61	**62.17**	0.55	45.33	**36.73**	2.58	79.77	**86.30**
Typical acceptance (Cai et al., 2024)	4.61	162.73	69.84	5.20	180.10	56.88	1.08	55.54	27.91	2.64	85.61	81.40

5.2 EAGLE-3

Beyond standard SD, EAGLE-3 is of particular interest because it operates over multiple drafts arranged in a tree. We are therefore intrigued to investigate whether lossy verification behaves differently in this setting. Theorem 1 answers this by characterizing the divergence from the matched truncation sampling target under both schemes.

Theorem 1 (Draft-target divergence under tree verification)**.** *For a strategy Θ with allowed set $\mathcal{A}_\Theta$, let $Z_\Theta(p) = \sum_{x\in\mathcal{A}_\Theta} p(x)$ and $Z_\Theta(q) = \sum_{x\in\mathcal{A}_\Theta} q(x)$ denote the retained target and draft mass. The KL divergence between the distributions generated by truncation-based verification and truncation sampling is*

$$\mathrm{KL}_{\mathrm{SD}} = \mathbb{E}\Big[\sum_{x\in\mathcal{A}_\Theta} \frac{q(x)}{Z_\Theta(q)} \log \frac{q(x)Z_\Theta(p)}{Z_\Theta(q)p(x)}\Big], \quad \mathrm{KL}_{\mathrm{EAGLE}} = \mathbb{E}\Big[\log \frac{Z_\Theta(p)}{p(x)}\Big]$$

for standard SD and EAGLE-3 respectively, and these behave differently as the draft improves: $\lim_{q\to p} \mathrm{KL}_{\mathrm{SD}} = 0$*, whereas* $\mathrm{KL}_{\mathrm{EAGLE}} > 0$.

In particular, $\mathrm{KL}_{\mathrm{SD}} < \mathrm{KL}_{\mathrm{EAGLE}}$ for all q sufficiently close to p: improving draft drives per-token distortion to zero under standard SD but not under EAGLE-3. We therefore expect task-performance gap between truncation-based verification and truncation sampling larger under EAGLE-3.

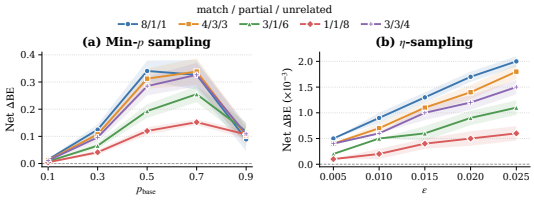


Figure 6: Net ΔBE under different draft–target alignment ratios (matching / partially overlapping / unrelated tokens). (a) Min-p: gain peaks at intermediate p_{base} and collapses at 0.9. (b) η-sampling: gain grows monotonically with ϵ.

Consistent with the theoretical prediction, under EAGLE-3, where the divergence in Theorem 1 does not vanish, the average performance gap to the matched truncation sampling baseline widens for both SpecCascade and typical acceptance (Table 3), compared with standard SD (Figure 5). The degradation is severe enough that typical acceptance falls below the EAGLE-3 baseline on all four benchmarks, and SpecCascade on two, while the matched truncation sampling baselines stay above it. What the truncation-based rules buy in return is a small block-efficiency gain (Section G), which does not offset the distortion.

6 Conclusion

We provide a unified view of lossy verification in speculative decoding, showing that existing methods fall into two paradigms: truncation-based verification and collaborative verification. Our analysis reveals a critical pitfall of truncation-based methods, where performance can collapse relative to the true baseline due to distributional distortion, especially in EAGLE-3, and identifies the importance of controlling overshoot in lossy verification to preserve generation quality. These findings offer principled guidance for speculative decoding that enable faster, quality-preserving LLM inference.

REFERENCES

Josh Achiam, Steven Adler, Sandhini Agarwal, Lama Ahmad, Ilge Akkaya, Florencia Leoni Aleman, Diogo Almeida, Janko Altenschmidt, Sam Altman, Shyamal Anadkat, et al. Gpt-4 technical report. *arXiv preprint arXiv:2303.08774*, 2023.

Gregor Bachmann, Sotiris Anagnostidis, Albert Pumarola, Markos Georgopoulos, Artsiom Sanakoyeu, Yuming Du, Edgar Schönfeld, Ali Thabet, and Jonas K Kohler. Judge decoding: Faster speculative sampling requires going beyond model alignment. In *Proceedings of the International Conference on Learning Representations (ICLR)*, 2025.

Jinze Bai, Shuai Bai, Yunfei Chu, Zeyu Cui, Kai Dang, Xiaodong Deng, Yang Fan, Wenbin Ge, Yu Han, Fei Huang, et al. Qwen technical report. *arXiv preprint arXiv:2309.16609*, 2023.

Tianle Cai, Yuhong Li, Zhengyang Geng, Hongwu Peng, Jason D Lee, Deming Chen, and Tri Dao. Medusa: Simple llm inference acceleration framework with multiple decoding heads. In *Proceedings of the International Conference on Machine Learning (ICML)*, 2024.

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. Training verifiers to solve math word problems. *arXiv preprint arXiv:2110.14168*, 2021. URL https://arxiv.org/abs/2110.14168.

Jiaming Fan, Daming Cao, Xiangzhong Luo, Jiale Fu, Chonghan Liu, and Xu Yang. Flatter tokens are more valuable for speculative draft model training. *arXiv preprint arXiv:2601.18902*, 2026.

Elias Frantar, Saleh Ashkboos, Torsten Hoefler, and Dan Alistarh. Gptq: Accurate post-training quantization for generative pre-trained transformers. *arXiv preprint arXiv:2210.17323*, 2022.

Jiale Fu, Yuchu Jiang, Junkai Chen, Jiaming Fan, Xin Geng, and Xu Yang. Fast large language model collaborative decoding via speculation. *arXiv preprint arXiv:2502.01662*, 2025.

Roman Garipov, Fedor Velikonivtsev, Ivan Ermakov, Ruslan Svirschevski, Vage Egiazarian, and Max Ryabinin. AutoJudge: Judge decoding without manual annotation. In *Advances in Neural Information Processing Systems*, 2025. URL https://arxiv.org/abs/2504.20039.

Daya Guo, Dejian Yang, Haowei Zhang, Junxiao Song, Ruoyu Zhang, Runxin Xu, Qihao Zhu, Shirong Ma, Peiyi Wang, Xiao Bi, et al. Deepseek-r1: Incentivizing reasoning capability in llms via reinforcement learning. *arXiv preprint arXiv:2501.12948*, 2025.

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

John Hewitt, Christopher D Manning, and Percy Liang. Truncation sampling as language model desmoothing. *arXiv preprint arXiv:2210.15191*, 2022.

Maximilian Holsman, Yukun Huang, and Bhuwan Dhingra. Fuzzy speculative decoding for a tunable accuracy–runtime tradeoff. *arXiv preprint arXiv:2502.20704*, 2025. URL https://arxiv.org/abs/2502.20704.

Yichong Huang, Xiaocheng Feng, Baohang Li, Yang Xiang, Hui Wang, Ting Liu, and Bing Qin. Ensemble learning for heterogeneous large language models with deep parallel collaboration. *Advances in Neural Information Processing Systems (NeurIPS)*, 2024.

Naman Jain, King Han, Alex Gu, Wen-Ding Li, Fanjia Yan, Tianjun Zhang, Sida Wang, Armando Solar-Lezama, Koushik Sen, and Ion Stoica. LiveCodeBench: Holistic and contamination free evaluation of large language models for code. In *The Thirteenth International Conference on Learning Representations (ICLR)*, 2025. URL https://openreview.net/forum?id=chfJJYC3iL.

Yaniv Leviathan, Matan Kalman, and Yossi Matias. Fast inference from transformers via speculative decoding. In *Proceedings of the International Conference on Machine Learning (ICML)*, 2023.

Xiang Lisa Li, Ari Holtzman, Daniel Fried, Percy Liang, Jason Eisner, Tatsunori B Hashimoto, Luke Zettlemoyer, and Mike Lewis. Contrastive decoding: Open-ended text generation as optimization. In *Proceedings of the 61st Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, 2023.

Yuhui Li, Fangyun Wei, Chao Zhang, and Hongyang Zhang. Eagle-3: Scaling up inference acceleration of large language models via training-time test. *arXiv preprint arXiv:2503.01840*, 2025.

Jiawei Liu, Chunqiu Steven Xia, Yuyao Wang, and Lingming Zhang. Is your code generated by chatgpt really correct? rigorous evaluation of large language models for code generation. *Advances in Neural Information Processing Systems (NeurIPS)*, 2023.

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

Harikrishna Narasimhan, Wittawat Jitkrittum, Ankit Singh Rawat, Seungyeon Kim, Neha Gupta, Aditya Krishna Menon, and Sanjiv Kumar. Faster cascades via speculative decoding. *arXiv preprint arXiv:2405.19261*, 2024.

Minh Nhat Nguyen, Andrew Baker, Clement Neo, Allen Roush, Andreas Kirsch, and Ravid Shwartz-Ziv. Turning up the heat: Min-p sampling for creative and coherent llm outputs. *arXiv preprint arXiv:2407.01082*, 2024.

Sean O'Brien and Mike Lewis. Contrastive decoding improves reasoning in large language models. *arXiv preprint arXiv:2309.09117*, 2023.

OpenAI, :, Aaron Jaech, Adam Kalai, Adam Lerer, Adam Richardson, Ahmed El-Kishky, Aiden Low, Alec Helyar, Aleksander Madry, Alex Beutel, Alex Carney, Alex Iftimie, Alex Karpenko, Alex Tachard Passos, Alexander Neitz, Alexander Prokofiev, Alexander Wei, Allison Tam, Ally Bennett, Ananya Kumar, Andre Saraiva, Andrea Vallone, Andrew Duberstein, Andrew Kondrich, Andrey Mishchenko, Andy Applebaum, Angela Jiang, Ashvin Nair, Barret Zoph, Behrooz Ghorbani, Bohan Zhang, Ben Rossen, Benjamin Sokolowsky, Boaz Barak, Bob McGrew, Borys Minaiev, Botao Hao, Bowen Baker, Brandon Houghton, Brandon McKinzie, Brydon Eastman, Camillo Lugaresi, Cary Bassin, Cary Hudson, Chak Ming Li, Charles de Bourcy, Chelsea Voss, Chen Shen, Chong Zhang, Chris Koch, Chris Orsinger, Christopher Hesse, Claudia Fischer, Clive Chan, Dan Roberts, Daniel Kappler, Daniel Levy, Daniel Selsam, David Dohan, David Farhi, David Mely, David Robinson, Dimitris Tsipras, Doug Li, Dragos Oprica, Eben Freeman, Eddie Zhang, Edmund Wong, Elizabeth Proehl, Enoch Cheung, Eric Mitchell, Eric Wallace, Erik Ritter, Evan Mays, Fan Wang, Felipe Petroski Such, Filippo Raso, Florencia Leoni, Foivos Tsimpourlas, Francis Song, Fred von Lohmann, Freddie Sulit, Geoff Salmon, Giambattista Parascandolo, Gildas Chabot, Grace Zhao, Greg Brockman, Guillaume Leclerc, Hadi Salman, Haiming Bao, Hao Sheng, Hart Andrin, Hessam Bagherinezhad, Hongyu Ren, Hunter Lightman, Hyung Won Chung, Ian Kivlichan, Ian O'Connell, Ian Osband, Ignasi Clavera Gilaberte, Ilge Akkaya, Ilya Kostrikov, Ilya Sutskever, Irina Kofman, Jakub Pachocki, James Lennon, Jason Wei, Jean Harb, Jerry Twore, Jiacheng Feng, Jiahui Yu, Jiayi Weng, Jie Tang, Jieqi Yu, Joaquin Quiñonero Candela, Joe Palermo, Joel Parish, Johannes Heidecke, John Hallman, John Rizzo, Jonathan Gordon, Jonathan Uesato, Jonathan Ward, Joost Huizinga, Julie Wang, Kai Chen, Kai Xiao, Karan Singhal, Karina Nguyen, Karl Cobbe, Katy Shi, Kayla Wood, Kendra Rimbach, Keren Gu-Lemberg, Kevin Liu, Kevin Lu, Kevin Stone, Kevin Yu, Lama Ahmad, Lauren Yang, Leo Liu, Leon Maksin, Leyton Ho, Liam Fedus, Lilian Weng, Linden Li, Lindsay McCallum, Lindsey Held, Lorenz Kuhn, Lukas Kondraciuk, Lukasz Kaiser, Luke Metz, Madelaine Boyd, Maja Trebacz, Manas Joglekar, Mark Chen, Marko Tintor, Mason Meyer, Matt Jones, Matt Kaufer, Max Schwarzer, Meghan Shah, Mehmet Yatbaz, Melody Y. Guan, Mengyuan Xu, Mengyuan Yan, Mia Glaese, Mianna Chen, Michael Lampe, Michael Malek, Michele Wang, Michelle Fradin, Mike McClay, Mikhail Pavlov, Miles Wang, Mingxuan Wang, Mira Murati, Mo Bavarian, Mostafa Rohaninejad, Nat McAleese, Neil Chowdhury, Neil Chowdhury, Nick Ryder, Nikolas Tezak, Noam Brown, Ofir Nachum, Oleg Boiko, Oleg Murk, Olivia Watkins, Patrick Chao, Paul Ashbourne, Pavel Izmailov, Peter Zhokhov, Rachel Dias, Rahul Arora,

Randall Lin, Rapha Gontijo Lopes, Raz Gaon, Reah Miyara, Reimar Leike, Renny Hwang, Rhythm Garg, Robin Brown, Roshan James, Rui Shu, Ryan Cheu, Ryan Greene, Saachi Jain, Sam Altman, Sam Toizer, Sam Toyer, Samuel Miserendino, Sandhini Agarwal, Santiago Hernandez, Sasha Baker, Scott McKinney, Scottie Yan, Shengjia Zhao, Shengli Hu, Shibani Santurkar, Shraman Ray Chaudhuri, Shuyuan Zhang, Siyuan Fu, Spencer Papay, Steph Lin, Suchir Balaji, Suvansh Sanjeev, Szymon Sidor, Tal Broda, Aidan Clark, Tao Wang, Taylor Gordon, Ted Sanders, Tejal Patwardhan, Thibault Sottiaux, Thomas Degry, Thomas Dimson, Tianhao Zheng, Timur Garipov, Tom Stasi, Trapit Bansal, Trevor Creech, Troy Peterson, Tyna Eloundou, Valerie Qi, Vineet Kosaraju, Vinnie Monaco, Vitchyr Pong, Vlad Fomenko, Weiyi Zheng, Wenda Zhou, Wenting Zhan, Wes McCabe, Wojciech Zaremba, Yann Dubois, Yinghai Lu, Yining Chen, Young Cha, Yu Bai, Yuchen He, Yuchen Zhang, Yunyun Wang, Zheng Shao, and Zhuohan Li. Openai o1 system card, 2026. URL https://arxiv.org/abs/2412.16720.

Shishir G. Patil, Huanzhi Mao, Charlie Cheng-Jie Ji, Fanjia Yan, Vishnu Suresh, Ion Stoica, and Joseph E. Gonzalez. The berkeley function calling leaderboard (bfcl): From tool use to agentic evaluation of large language models. In *Forty-second International Conference on Machine Learning*, 2025.

Yingzhe Peng, Gongrui Zhang, Miaosen Zhang, Zhiyuan You, Jie Liu, Qipeng Zhu, Kai Yang, Xingzhong Xu, Xin Geng, and Xu Yang. Lmm-r1: Empowering 3b lmms with strong reasoning abilities through two-stage rule-based rl, 2025. URL https://arxiv.org/abs/2503.07536.

Angelika Romanou, Negar Foroutan, Anna Sotnikova, Zeming Chen, Sree Harsha Nelaturu, Shivalika Singh, Rishabh Maheshwary, Micol Altomare, Mohamed A Haggag, Alfonso Amayuelas, et al. Include: Evaluating multilingual language understanding with regional knowledge. *arXiv preprint arXiv:2411.19799*, 2024.

Timo Schick, Jane Dwivedi-Yu, Roberto Dessì, Roberta Raileanu, Maria Lomeli, Eric Hambro, Luke Zettlemoyer, Nicola Cancedda, and Thomas Scialom. Toolformer: Language models can teach themselves to use tools. *Advances in Neural Information Processing Systems (NeurIPS)*, 2023.

Significant-Gravitas. Autogpt, 2023. URL https://github.com/Significant-Gravitas/AutoGPT. Accessed: 2026-01-22.

Ziteng Sun, Ananda Theertha Suresh, Jae Hun Ro, Ahmad Beirami, Himanshu Jain, and Felix Yu. Spectr: Fast speculative decoding via optimal transport. *Advances in Neural Information Processing Systems (NeurIPS)*, 2023.

Ziteng Sun, Uri Mendlovic, Yaniv Leviathan, Asaf Aharoni, Ahmad Beirami, Jae Hun Ro, and Ananda Theertha Suresh. Block verification accelerates speculative decoding. *arXiv preprint arXiv:2403.10444*, 2024.

Hugo Touvron, Louis Martin, Kevin Stone, Peter Albert, Amjad Almahairi, Yasmine Babaei, Nikolay Bashlykov, Soumya Batra, Prajjwal Bhargava, Shruti Bhosale, et al. Llama 2: Open foundation and fine-tuned chat models. *arXiv preprint arXiv:2307.09288*, 2023.

Ziyi Wang, Siva Rajesh Kasa, Ankith M S, Santhosh Kumar Kasa, Jiaru Zou, Sumit Negi, Ruqi Zhang, Nan Jiang, and Qifan Song. DIVERSED: Relaxed speculative decoding via dynamic ensemble verification. In *Proceedings of the 29th International Conference on Artificial Intelligence and Statistics*, 2026. URL https://arxiv.org/abs/2604.07622.

An Yang, Baosong Yang, Beichen Zhang, Binyuan Hui, Bo Zheng, Bowen Yu, Chengyuan Li, Dayiheng Liu, Fei Huang, Haoran Wei, Huan Lin, Jian Yang, Jianhong Tu, Jianwei Zhang, Jianxin Yang, Jiaxi Yang, Jingren Zhou, Junyang Lin, Kai Dang, Keming Lu, Keqin Bao, Kexin Yang, Le Yu, Mei Li, Mingfeng Xue, Pei Zhang, Qin Zhu, Rui Men, Runji Lin, Tianhao Li, Tingyu Xia, Xingzhang Ren, Xuancheng Ren, Yang Fan, Yang Su, Yichang Zhang, Yu Wan, Yuqiong Liu, Zeyu Cui, Zhenru Zhang, and Zihan Qiu. Qwen2.5 technical report. *arXiv preprint arXiv:2412.15115*, 2025.

Sen Yang, Shujian Huang, Xinyu Dai, and Jiajun Chen. Multi-candidate speculative decoding. *arXiv preprint arXiv:2401.06706*, 2024.

Shunyu Yao, Jeffrey Zhao, Dian Yu, Nan Du, Izhak Shafran, Karthik R Narasimhan, and Yuan Cao. React: Synergizing reasoning and acting in language models. In *Proceedings of the International Conference on Learning Representations (ICLR)*, 2022.

Yuxuan Yao, Han Wu, Mingyang Liu, Sichun Luo, Xiongwei Han, Jie Liu, Zhijiang Guo, and Linqi Song. Determine-then-ensemble: Necessity of top-k union for large language model ensembling. *arXiv preprint arXiv:2410.03777*, 2024.

Yang Yue, Zhiqi Chen, Rui Lu, Andrew Zhao, Zhaokai Wang, Yang Yue, Shiji Song, and Gao Huang. Does reinforcement learning really incentivize reasoning capacity in llms beyond the base model? In *Advances in Neural Information Processing Systems (NeurIPS)*, 2025.

Yongchao Zhou, Kaifeng Lyu, Ankit Singh Rawat, Aditya Krishna Menon, Afshin Rostamizadeh, Sanjiv Kumar, Jean-François Kagy, and Rishabh Agarwal. Distillspec: Improving speculative decoding via knowledge distillation. In *Proceedings of the International Conference on Learning Representations (ICLR)*, 2024.

Yuxuan Zhou, Margret Keuper, and Mario Fritz. Balancing diversity and risk in llm sampling: How to select your method and parameter for open-ended text generation. In *Proceedings of the Annual Meeting of the Association for Computational Linguistics (ACL)*, 2025.

Yuxuan Zhou, Fei Huang, Heng Li, Fengyi Wu, Tianyu Wang, Jianwei Zhang, Junyang Lin, and Zhi-Qi Cheng. Overcoming joint intractability with lossless hierarchical speculative decoding. *The Fourteenth International Conference on Learning Representations*, 2026.

A Detailed Derivations for Collaborative Verification Methods

This section derives the generation distributions induced by Lenience, Judge-Based Relaxation, DIVERSED and Fuzzy Speculative Decoding (Fuzzy SD). For DIVERSED and Fuzzy SD, the resulting distribution can be interpreted as a context-dependent combination of the target distribution p and draft distribution q.

A.1 DIVERSED

DIVERSED (Wang et al., 2026) generalizes the static ensemble used by CoS by predicting a task- and context-dependent ensemble weight. At decoding step t, the method computes

$$\lambda_t = f(h_t^q, h_t^p), \qquad \lambda_t \in [0,1], \tag{15}$$

where h_t^q and h_t^p are representations produced by the draft and target models. For brevity, we omit the time index and write $\lambda = \lambda_t$.

DIVERSED constructs the context-dependent ensemble distribution

$$\nu(x) = \lambda p(x) + \big(1-\lambda\big) q(x). \tag{16}$$

The draft token is then verified against ν, rather than directly against p, using

$$h(x) = \min\left(1, \frac{\nu(x)}{q(x)}\right). \tag{17}$$

Upon rejection, the corresponding speculative-sampling correction distribution is

$$r(x) = \frac{\big(\nu(x) - q(x)\big)_+}{\sum_{v\in\mathcal{V}} \big(\nu(v) - q(v)\big)_+}. \tag{18}$$

Let

$$R = 1 - \sum_{v\in\mathcal{V}} q(v)h(v) \tag{19}$$

denote the rejection probability. Since

$$q(x)h(x) = \min(q(x), \nu(x)), \tag{20}$$

we have

$$R = \sum_{v \in \mathcal{V}} \big(q(v) - \nu(v)\big)_+. \tag{21}$$

Because both q and ν are normalized distributions,

$$\sum_{v \in \mathcal{V}} \big(q(v) - \nu(v)\big)_+ = \sum_{v \in \mathcal{V}} \big(\nu(v) - q(v)\big)_+. \tag{22}$$

Substituting Equations (17) and (18) gives

$$\begin{aligned} P(\text{generate } x) &= \min(q(x), \nu(x)) + Rr(x) && (23)\\ &= \min(q(x), \nu(x)) + \big(\nu(x) - q(x)\big)_+ && (24)\\ &= \nu(x). && (25) \end{aligned}$$

Therefore,

$$\boxed{P(\text{generate } x) = \lambda p(x) + \big(1 - \lambda\big) q(x).} \tag{26}$$

DIVERSED thus belongs directly to collaborative verification. Unlike CoS, which uses a fixed global coefficient, DIVERSED predicts λ from the current task and decoding context. However, at each individual decoding position, the coefficient remains shared across all vocabulary tokens.

A.2 Fuzzy Speculative Decoding

Fuzzy SD (Holsman et al., 2025) makes a deterministic acceptance decision based on the divergence between the full draft and target distributions at the current position. Define the context-level gate

$$g_T = \mathbb{I}\left[D(p(\cdot), q(\cdot)) < T\right], \qquad g_T \in \{0, 1\}, \tag{27}$$

where D is a distributional divergence and T is a user-specified threshold.

Because g_T depends on the complete pair of distributions rather than on the sampled token, the acceptance probability is

$$h_T(x) = g_T \qquad \text{for every } x \in \mathcal{V}. \tag{28}$$

When $g_T = 1$, the divergence is below the threshold and every candidate drawn from q is accepted. Therefore,

$$P(\text{generate } x \mid g_T = 1) = q(x). \tag{29}$$

When $g_T = 0$, the candidate is rejected and Fuzzy SD uses the target decoding distribution for correction. Thus,

$$P(\text{generate } x \mid g_T = 0) = p(x). \tag{30}$$

Equivalently, substituting $h_T(x) = g_T$ and $r(x) = p(x)$ yields

$$\begin{aligned} P(\text{generate } x) &= g_T q(x) + \left(1 - \sum_{v \in \mathcal{V}} g_T q(v)\right) p(x) && (31)\\ &= g_T q(x) + (1 - g_T) p(x), && (32) \end{aligned}$$

because $\sum_v q(v) = 1$. Hence,

$$\boxed{P(\text{generate } x) = (1 - g_T) p(x) + g_T q(x).} \tag{33}$$

Fuzzy SD can therefore be interpreted as a binary, context-dependent collaborative verification method:

$$P(\text{generate } x) = \begin{cases} q(x), & D(p(\cdot), q(\cdot)) < T, \\ p(x), & D(p(\cdot), q(\cdot)) \geq T. \end{cases} \tag{34}$$

Unlike CoS and DIVERSED, Fuzzy SD does not use a soft interpolation coefficient in $(0, 1)$. Instead, it selects one endpoint of the p–q interpolation according to the current distributional discrepancy.

A.3 LENIENCE

In tokenwise speculative sampling (Leviathan et al., 2023), each token x is drafted from $q(x)$ and verified against $p(x)$. It is accepted with probability $h(x) = \min\{1,\, p(x)/\ell q(x)\}$, or rejected and replaced from $P_{\text{res}}(x)$. Thus the yield probability is:

$$
\begin{aligned}
&P(\text{generate } x) = P(\text{draft and accept } x) + P(\text{draft and reject } v,\ \text{resampled } x) \\
&= q(x)h(x) + \Big[\sum_{v\in\mathcal{V}} q(v)(1-h(v))\Big] \times \frac{p(x) - \min\{p(x), q(x)\}}{\sum_{v\in\mathcal{V}}\big(p(v) - \min\{p(v), q(v)\}\big)} \\
&= \min\Big\{q(x),\, \frac{p(x)}{\ell}\Big\} + \frac{\sum_{v\in\mathcal{V}}\big(q(v) - \min\{q(v),\, p(v)/\ell\}\big)}{\sum_{v\in\mathcal{V}}\big(q(v) - \min\{q(v),\, p(v)\}\big)} \times \Big[p(x) - \min\{p(x), q(x)\}\Big] \\
&= \min\Big\{q(x),\, \frac{p(x)}{\ell}\Big\} + \Delta_\ell\Big[p(x) - \min\{p(x), q(x)\}\Big] \\
&= \begin{cases} q(x) + \Delta_\ell(p(x) - q(x)), & q(x) \le p(x), \\ q(x), & p(x) \le q(x) \le p(x)/\ell, \\ p(x)/\ell, & q(x) \ge p(x)/\ell. \end{cases}
\end{aligned}
$$

Derivation for Δ_ℓ

$$
\Delta_\ell = \frac{\sum_{v\in\mathcal{V}}\left(q(v) - \min\left\{q(v),\, \frac{p(v)}{\ell}\right\}\right)}{\sum_{v\in\mathcal{V}}\left(q(v) - \min\left\{q(v),\, p(v)\right\}\right)}. \tag{35}
$$

Using the identity

$$
a - \min\{a, b\} = \frac{|a-b| + (a-b)}{2}, \tag{36}
$$

the numerator becomes

$$
\sum_{v\in\mathcal{V}}\left(q(v) - \min\left\{q(v),\, \frac{p(v)}{\ell}\right\}\right) = \frac{1}{2}\sum_{v\in\mathcal{V}}\left|q(v) - \frac{p(v)}{\ell}\right| + \frac{1}{2}\sum_{v\in\mathcal{V}}\left(q(v) - \frac{p(v)}{\ell}\right). \tag{37}
$$

Since p and q are probability distributions on $\mathcal{V}$,

$$
\sum_{v\in\mathcal{V}} q(v) = 1, \qquad \sum_{v\in\mathcal{V}} p(v) = 1, \tag{38}
$$

we have

$$
\sum_{v\in\mathcal{V}}\left(q(v) - \frac{p(v)}{\ell}\right) = 1 - \frac{1}{\ell}. \tag{39}
$$

Therefore,

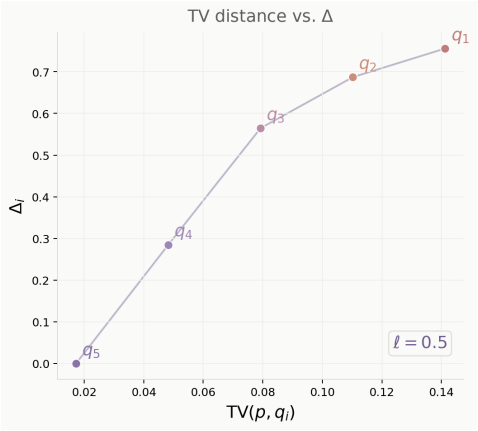
$$
\sum_{v\in\mathcal{V}}\left(q(v) - \min\left\{q(v),\, \frac{p(v)}{\ell}\right\}\right) = \frac{1}{2}\sum_{v\in\mathcal{V}}\left|q(v) - \frac{p(v)}{\ell}\right| + \frac{1}{2} - \frac{1}{2\ell}. \tag{40}
$$

Similarly, the denominator satisfies

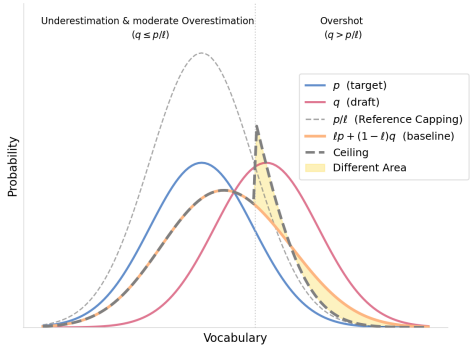
$$
\sum_{v\in\mathcal{V}}\left(q(v) - \min\left\{q(v),\, p(v)\right\}\right) = \frac{1}{2}\sum_{v\in\mathcal{V}}|q(v) - p(v)| + \frac{1}{2}\sum_{v\in\mathcal{V}}\left(q(v) - p(v)\right). \tag{41}
$$

Again using

$$
\sum_{v\in\mathcal{V}} q(v) = \sum_{v\in\mathcal{V}} p(v) = 1, \tag{42}
$$



(a) Adaptive interpolation.



(b) Overshoot ceiling.

Figure 7: Two views of lenience-based relaxation. Fig.a illustrates how the interpolation strength adapts to the draft–target divergence on a per-token basis. Fig.b contrasts a global linear mixture with a pointwise ceiling at p/ℓ, highlighting the asymmetric treatment of overshoot.

we obtain

$$\sum_{v \in \mathcal{V}} (q(v) - p(v)) = 0, \tag{43}$$

and hence

$$\sum_{v \in \mathcal{V}} (q(v) - \min\{q(v), p(v)\}) = \frac{1}{2} \sum_{v \in \mathcal{V}} |q(v) - p(v)|. \tag{44}$$

Substituting these two expressions back into Δ_ℓ yields

$$\Delta_\ell = \frac{\frac{1}{2} \sum_{v \in \mathcal{V}} \left| q(v) - \frac{p(v)}{\ell} \right| + \frac{1}{2} - \frac{1}{2\ell}}{\frac{1}{2} \sum_{v \in \mathcal{V}} |q(v) - p(v)|}. \tag{45}$$

Thus,

$$\boxed{P(\text{generate } x) = \begin{cases} \Delta_\ell p(x) + (1 - \Delta_\ell) q(x), & q(x) \le p(x), \\ q(x), & p(x) < q(x) \le p(x)/\ell, \\ p(x)/\ell, & q(x) \ge p(x)/\ell. \end{cases}} \tag{46}$$

B Lenience-based Relaxation Interpretation

Lenience-based relaxation trades a controlled amount of distributional distortion for a higher acceptance rate, governed by the lenience parameter ℓ. Figure 7 examines two facets of this trade-off: *how much* relaxation to apply, and *where* it takes effect.

Adaptive interpolation. Figure 7a plots the interpolation strength Δ_i against the draft–target divergence $\mathrm{TV}(p, q_i)$. Rather than applying a fixed amount of leniency, the relaxation adapts to local disagreement: when the draft already agrees with the target (q_5, small TV) we have $\Delta_i \approx 0$ and almost no relaxation is applied, whereas a more divergent draft (q_1) induces a larger Δ_i. The concave profile further shows that the additional leniency saturates rather than growing without bound as divergence increases, so relaxation is spent precisely where draft and target diverge and withheld where they coincide.

Table 4: Parameter counts of the target and draft models used in the experiments. For EAGLE-3, the draft-model count denotes the additional parameters of the EAGLE-3 draft head.

Decoding Framework	Model Pair	Role	Model	Parameters
Tokenwise SD	Qwen2.5	Target	`Qwen/Qwen2.5-72B-Instruct-GPTQ-Int8`	72.7B
		Draft	`Qwen/Qwen2.5-0.5B-Instruct-GPTQ-Int8`	0.49B
	LLaMA	Target	`meta-llama/Llama-3.1-8B-Instruct`	8B
		Draft	`meta-llama/Llama-3.2-1B-Instruct`	1.23B
EAGLE-3	LLaMA	Target	`meta-llama/Llama-3.1-8B-Instruct`	8B
		Draft	`yuhuili/EAGLE3-LLaMA3.1-Instruct-8B`	$\sim$0.4B
	Qwen3	Target	`Qwen/Qwen3-8B`	8.2B
		Draft	`Tengyunw/qwen3_8b_eagle3`	$\sim$0.4B

Overshoot ceiling. Figure 7b contrasts a global linear mixture $\ell p + (1-\ell)q$, which blends draft and target uniformly across the vocabulary, with a pointwise ceiling at p/ℓ (reference capping), which accepts the draft wherever it stays below the ceiling and clips it only where it exceeds it. Two regimes emerge: for $q \le p/\ell$ the draft lies under the ceiling (underestimation or moderate overestimation) and is accepted, while for $q > p/\ell$ the draft *overshoots* the relaxed target and is capped. The shaded region marks exactly where the two schemes disagree—the overshoot mass that the ceiling clips but the linear baseline retains—highlighting the asymmetric treatment of overshoot.

C Experimental Setup

Models. We evaluate tokenwise speculative decoding using two target–draft model pairs. For the Qwen2.5 experiments in Table 7, we use the instruction-tuned, 8-bit GPTQ-quantized (Frantar et al., 2022) models `Qwen/Qwen2.5-72B-Instruct-GPTQ-Int8` and `Qwen/Qwen2.5-0.5B-Instruct-GPTQ-Int8` as the target and draft models, respectively (Yang et al., 2025). For the LLaMA experiments in Table 8, we use `meta-llama/Llama-3.1-8B-Instruct` as the target model and `meta-llama/Llama-3.2-1B-Instruct` as the draft model.

For the EAGLE-3 experiments (Li et al., 2025), we consider two target–draft configurations. Table 9 uses `meta-llama/Llama-3.1-8B-Instruct` as the target model and `yuhuili/EAGLE3-LLaMA3.1-Instruct-8B` as the corresponding EAGLE-3 draft model. Table 10 uses `Qwen/Qwen3-8B` as the target model and `Tengyunw/qwen3_8b_eagle3` as the EAGLE-3 draft model.

Artifact Licensing. We release only our source code and experimental scripts. Our implementation builds on publicly available open-source models, which remain subject to their original licenses and are not redistributed as part of our release. Users are required to obtain the corresponding model weights from their original repositories and comply with the associated license terms. We preserve all required attribution and license notices for reused components, and release only code that we have the right to distribute.

Computation hours. For the Qwen2.5 experiments in Table 7, it takes ~ 5000 A100 hours. For the LLaMA experiments in Table 8, it takes ~ 200 A6000 hours. For the EAGLE-3 experiments in Table 9, it takes ~ 100 A6000 hours. For the EAGLE-3 experiments in Table 10, it takes ~ 100 A6000 hours.

Benchmarks and Metrics. We evaluate all methods on four benchmarks covering complementary generation capabilities: MATH Hendrycks et al. (2021) for mathematical reasoning, MBPP+ Liu et al. (2023) for code generation, INCLUDE for multilingual knowledge, and BFCL (Patil et al., 2025) for tool use. We use the first 500 examples from MATH Hendrycks et al. (2021), the full MBPP+ Liu et al. (2023) test set containing 378 examples, five examples per language from INCLUDE (Romanou et al., 2024) for a total of 220 examples, and 200 calls from the `parallel_multiple` split of BFCL-v3 (Patil et al., 2025). Generation quality is measured using accuracy for MATH, INCLUDE, and

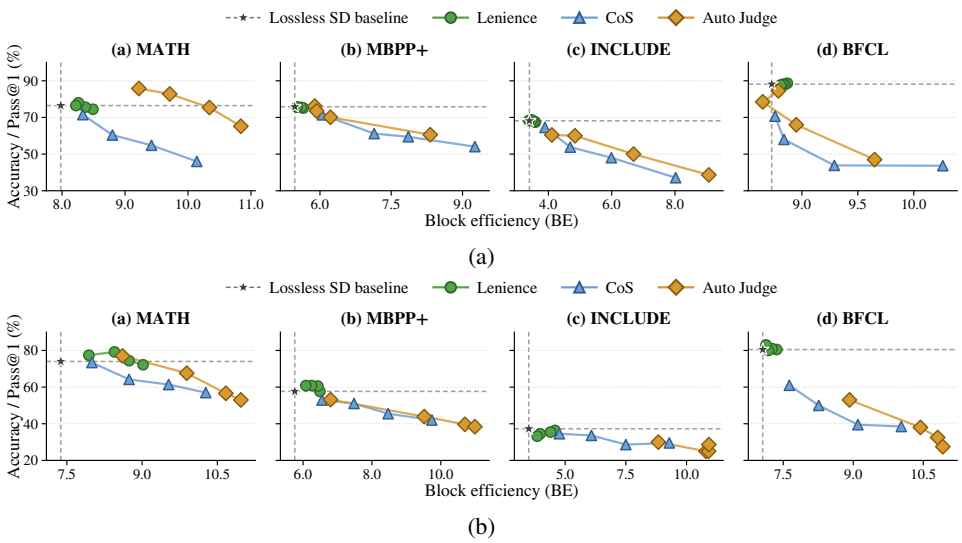


Figure 8: Efficiency and task performance tradeoff for collaborative verification across four benchmarks. Figure 8a uses Qwen2.5-72B-Instruct-GPTQ-Int8 as the target model and Qwen2.5-0.5B-Instruct-GPTQ-Int8 as the draft model, while Figure 8b uses Llama-3.1-8B-Instruct as the target model and Llama-3.2-1B-Instruct as the draft model.

BFCL, and Pass@1 for MBPP+. Efficiency is measured using block efficiency (BE) and decoding speed (DS), where DS is reported in generated tokens per second.

Decoding Configuration. The EAGLE-3 experiments in Tables 9 and 10 use a sampling temperature of $T = 0.7$ and a block size of 7. For each relaxation method, we sweep the corresponding verification parameter over the values reported in the tables. In the overshoot-capping ablation in Table 5, results are averaged over $\ell \in 0.2, 0.4, 0.6, 0.8$, with $p_{\text{base}} = 0.5$ used to define the truncation set $\mathcal{A}_\Theta$.

Hardware Platform. The experiments in Figure 1 are conducted on a single NVIDIA H200 GPU with 140GB of memory. The EAGLE-3 experiments reported in Tables 9 and 10 are conducted on a single NVIDIA RTX A6000 GPU with 48 GB of memory. All remaining verification and ablation experiments are conducted on two NVIDIA A100 GPUs, each with 80 GB of memory.

D Extended Analysis of the Collaborative Verification Trade-off

Figure 8a reports the Qwen2.5-72B/0.5B pair; Figure 8b repeats the comparison with LLaMA-3.1-8B/3.2-1B. Each curve sweeps the relaxation hyperparameter of one method (the lenience factor ℓ, the interpolation weight λ of CoS, and the judge threshold δ of AutoJudge). We organize the discussion around three observations.

Lenience preserves task performance across both model pairs. Across all four benchmarks and both model pairs, lenience stays within noise of the lossless baseline in accuracy, in several cases marginally above it, while moving block efficiency by only a small margin (roughly $+0.2$ to $+1.5$ BE). Its points are also tightly clustered: sweeping ℓ over its usable range does not spread them out the way λ and the judge threshold δ spread the other two methods. Both properties trace to the same source. The MBPP+ ablation in Table 1 isolates it: removing the overshoot ceiling while retaining the adaptive interpolation coefficient recovers CoS-like degradation, whereas retaining the ceiling alone preserves the flat segment. Capping the yield of an overshooting token bounds the pointwise deviation from p, which is what protects accuracy, and the same cap limits how much acceptance mass can be recovered, which is why the efficiency gain is modest.

CoS degrades smoothly and monotonically with block efficiency. CoS traces a near-linear descent on every benchmark. This follows directly from Equation (8): the induced distribution is a convex combination of p and q with a coefficient that grows with λ uniformly over the vocabulary, so the generated distribution drifts toward the draft at a rate set by a single free scalar. Nothing in the rule distinguishes tokens whose substitution is harmless from tokens that change the answer, and the empirical curves inherit that uniformity: the trade-off is predictable, tunable, and strictly unfavorable.

The judge's apparent advantage is a domain effect, not an acceptance-rule effect. On MATH, AutoJudge dominates CoS at matched block efficiency for both model pairs, and with the Qwen pair it sits *above* the lossless baseline over a wide BE range. On the remaining three benchmarks the two curves largely interleave, and both fall well below the baseline once relaxation becomes aggressive. The size of the MATH gain is also model-dependent: under the Qwen pair the judge clears the lossless baseline, while under LLaMA-3.1-8B/3.2-1B the same sweep tracks at or slightly below it, though still above CoS. On BFCL with the Qwen pair the sweep is non-monotone, with at least one operating point scoring well below a neighbor at comparable block efficiency.

We attribute this pattern to the judge's supervision rather than to its acceptance rule. Our judge is a single head trained on the union of mined GSM8K and LiveCodeBench mismatches (Section F.2), with GSM8K contributing roughly ten times as many source problems, so the mixture is dominated by mathematical reasoning. MATH is the one benchmark that matches the dominant component of that mixture, and it is the one benchmark where the judge separates from CoS. The residual model dependence is consistent with the mined labels being derived from Qwen2.5 target completions, which aligns the judge with one of the two verified models but not the other. The non-monotonicity on BFCL is consistent with δ thresholding a learned score whose calibration shifts across tasks, unlike ℓ and λ, whose effects on the induced distribution are fixed in closed form by Equations (8) and (10).

E Detailed Analysis in Overshoot Analysis in Lenience-based Relaxation

The ablation in Table 5 supports our claim that adaptive interpolation is not the primary source of improvement in lenience-based relaxation: removing the adaptive interpolation while retaining the p/ℓ cap preserves most of the benefit. The overshoot region is precisely where the target and draft distributions disagree most strongly, and the draft tends to over-allocate probability to low-quality tokens there. Capping at p/ℓ in this region directly suppresses this failure mode.

We further investigate the overlap between the overshoot region and the set of tokens discarded by min-p sampling, motivated by the intuition that accepting more tokens from the discarded set can improve efficiency. This suggests combining the two mechanisms: use the min-p allowed set $\mathcal{A}_\Theta$ to gate acceptance, and apply the p/ℓ cap to control overshoot outside it. Concretely, tokens in $\mathcal{A}_\Theta$ are yielded with draft probability $q(x)$, while tokens outside are yielded with $\min\{q(x),\, p(x)/\ell\}$, so the rule relies on the draft distribution except in the overlap between the overshoot region and the discarded set. As shown in the last row of Table 5, this rule delivers a 3.7% gain in BE over lossless SD while matching its Pass@1 exactly, outperforming both SpecCascade and lenience-based relaxation. The discarded set thus appears to be a promising target for further acceleration at negligible quality cost.

F Extended Analysis for Judge-Based Relaxation

F.1 Judge-Based Relaxation Derivation

Judge-based relaxation augments standard speculative verification with a binary judgment signal $j_\phi(x) \in \{0, 1\}$. When $q(x) > p(x)$, standard speculative verification accepts the draft token with probability $p(x)/q(x)$. If standard verification would reject the token, the judge retains it whenever $j_\phi(x) = 1$. Therefore, the effective acceptance probability is

$$h(x) = \begin{cases} 1, & q(x) \le p(x), \\ \dfrac{p(x)}{q(x)} + \left(1 - \dfrac{p(x)}{q(x)}\right) j_\phi(x), & q(x) > p(x). \end{cases} \tag{47}$$

Table 5: Effect of truncation and overshoot capping on MBPP+, averaged over $\ell \in \{0.2, 0.4, 0.6, 0.8\}$. $p_{\text{base}} = 0.5$ is used in $\mathcal{A}_\Theta$.

Yielded rule	**Avg. BE**	**Avg. Pass@1 (%)**
$P(\text{generate } x) = p(x)$	5.42	75.66
$P(\text{generate } x) = \begin{cases} q(x), & x \in \mathcal{A}_\Theta, \\ 0, & \text{otherwise}, \end{cases}$	5.51 (+1.7%)	74.87 (-1.0%)
$P(\text{generate } x) = \begin{cases} \Delta\, p(x) + (1-\Delta)\, q(x), & q(x) \le p(x), \\ q(x), & p(x) < q(x) \le p(x)/\ell, \\ p(x)/\ell, & q(x) \ge p(x)/\ell, \end{cases}$	5.53 (+2.0%)	75.22 (-0.6%)
$P(\text{generate } x) = \begin{cases} q(x), & q(x) \le p(x)/\ell, \\ p(x)/\ell, & q(x) \ge p(x)/\ell, \end{cases}$	5.58 (+3.0%)	75.33 (-0.4%)
$P(\text{generate } x) = \begin{cases} q(x), & x \in \mathcal{A}_\Theta, \\ \min\{q(x), p(x)/\ell\}, & \text{otherwise} \end{cases}$	**5.62 (+3.7%)**	**75.66 (-0.0%)**

As in standard speculative sampling, a rejected proposal is resampled from the residual distribution

$$r(x) = \frac{\big(p(x) - q(x)\big)_+}{\sum_{v \in \mathcal{V}} \big(p(v) - q(v)\big)_+}. \tag{48}$$

The resulting probability of generating token x is

$$\begin{aligned} P(\text{generate } x) &= P(\text{draft and accept } x) + P(\text{reject})\, r(x) \\ &= q(x)h(x) + \left[\sum_{v \in \mathcal{V}} q(v)\big(1 - h(v)\big)\right] r(x). \end{aligned} \tag{49}$$

We first simplify the total rejection probability. For tokens satisfying $q(v) \le p(v)$, we have $h(v) = 1$, and hence they contribute no rejected mass. For $q(v) > p(v)$,

$$\begin{aligned} q(v)\big(1 - h(v)\big) &= q(v)\left[1 - \frac{p(v)}{q(v)} - \left(1 - \frac{p(v)}{q(v)}\right) j(v)\right] \\ &= \big(q(v) - p(v)\big)\big(1 - j(v)\big). \end{aligned} \tag{50}$$

Therefore,

$$R_j \coloneqq P(\text{reject}) = \sum_{v \in \mathcal{V}} \big(q(v) - p(v)\big)_+ \big(1 - j(v)\big). \tag{51}$$

Let

$$R = \sum_{v \in \mathcal{V}} \big(p(v) - q(v)\big)_+. \tag{52}$$

Since p and q are normalized distributions,

$$\sum_{v \in \mathcal{V}} \big(q(v) - p(v)\big)_+ = \sum_{v \in \mathcal{V}} \big(p(v) - q(v)\big)_+ = R. \tag{53}$$

Define

$$\Delta_j = \frac{R_j}{R} = \frac{\sum_{v \in \mathcal{V}} \big(q(v) - p(v)\big)_+ \big(1 - j(v)\big)}{\sum_{v \in \mathcal{V}} \big(p(v) - q(v)\big)_+}. \tag{54}$$

Because $j(v) \in \{0, 1\}$,

$$0 \le \big(1 - j(v)\big) \le 1, \tag{55}$$

which implies

$$0 \le R_j \le R \qquad \Longrightarrow \qquad \beta \in [0,1]. \tag{56}$$

We now derive the resulting distribution in the two regions.

Draft-underestimation region: $q(x) \le p(x)$. In this region, the draft token is always accepted, so

$$q(x)h(x) = q(x). \tag{57}$$

Moreover,

$$r(x) = \frac{p(x) - q(x)}{R}. \tag{58}$$

Substituting into Equation (49) gives

$$\begin{aligned} P(\text{generate}\, x) &= q(x) + R_j \frac{p(x) - q(x)}{R} \\ &= q(x) + \Delta_j \big(p(x) - q(x)\big) \\ &= \Delta_j p(x) + (1 - \Delta_j) q(x). \end{aligned} \tag{59}$$

Draft-overshoot region: $q(x) > p(x)$. In this region, the residual distribution assigns zero probability to x, since

$$\big(p(x) - q(x)\big)_+ = 0. \tag{60}$$

Thus, the generated probability consists only of accepted draft mass:

$$\begin{aligned} P(\text{generate}\, x) &= q(x)h(x) \\ &= q(x) \left[\frac{p(x)}{q(x)} + \left(1 - \frac{p(x)}{q(x)}\right) j_\phi(x) \right] \\ &= p(x) + \big(q(x) - p(x)\big) j_\phi(x) \\ &= \big(1 - j_\phi(x)\big) p(x) + j_\phi(x) q(x). \end{aligned} \tag{61}$$

Combining the two regions, the induced distribution is

$$P(\text{generate}\, x) = \begin{cases} \Delta_j p(x) + \big(1 - \Delta_j\big) q(x), & q(x) \le p(x), \\ p(x), & q(x) > p(x) \text{ and } j_\phi(x) = 0, \\ q(x), & q(x) > p(x) \text{ and } j_\phi(x) = 1. \end{cases} \tag{62}$$

where Δ_j is given by Equation (54).

Although all three methods introduce a learned judge to accept target–draft mismatches, they differ primarily in how the judge supervision is obtained. Judge Decoding (Bachmann et al., 2025) constructs a manually curated training set to teach a lightweight verifier whether a draft continuation remains valid despite differing from the target output. AutoJudge (Garipov et al., 2025) removes the need for manual annotation by automatically identifying task-critical mismatches through counterfactual correction and downstream outcome evaluation, but consequently relies on tasks with verifiable quality signals. Thus, both methods share the same judge-based relaxation mechanism at inference time, while progressing from manually supervised, to outcome-supervised, to target-derived self-supervised judge training.

F.2 Judge-based Relaxation Implementation

We instantiate judge-based relaxation with AutoJudge (Garipov et al., 2025), using the authors' public implementation.[1] Both models are frozen: `Qwen2.5-72B-Instruct-GPTQ-Int8` as the target model and `Qwen2.5-0.5B-Instruct-GPTQ-Int8` as the draft model.

AutoJudge labels a draft/target mismatch as *unimportant* if substituting the draft token and letting the target model complete the response preserves the final answer, and as *important* otherwise. The search is semi-greedy: it proceeds from the earliest mismatch, keeps accepted substitutions in the

[1] `https://github.com/garipovroma/autojudge`

Table 6: Mixture-ratio sweep, Qwen2.5-72B / 0.5B, thresholds matched at the 20th/40th/60th/80th percentile of each head's validation scores. Single run per cell.

Mix	Param	MATH		MBPP+		INCLUDE		BFCL	
		BE	Acc (%)	BE	Pass@1 (%)	BE	Acc (%)	BE	Acc (%)
9:1	0.2	8.59	84.20	5.81	76.72	3.76	55.91	8.88	88.00
	0.4	8.67	82.40	5.81	76.72	3.89	65.91	8.94	88.50
	0.6	8.47	83.80	5.81	76.72	3.93	60.00	8.84	87.50
	0.8	8.76	82.00	5.83	76.72	4.00	70.00	8.88	88.00
7:3	0.2	8.51	82.60	5.81	76.72	3.90	70.00	8.78	88.50
	0.4	9.04	82.40	5.82	76.19	4.56	70.00	8.94	87.00
	0.6	10.17	68.20	6.52	70.90	7.33	48.18	9.75	82.00
	0.8	10.97	62.60	8.46	55.56	10.22	48.18	10.09	52.00
5:5	0.2	8.95	78.40	6.20	69.31	5.73	54.09	8.70	88.00
	0.4	9.31	74.40	6.93	61.90	6.29	45.91	9.14	84.00
	0.6	9.98	76.60	7.83	58.20	7.73	40.00	9.97	70.00
	0.8	10.45	68.80	9.42	51.06	8.74	50.00	10.30	66.00
3:7	0.2	8.58	80.40	5.86	76.46	4.10	61.82	8.86	88.00
	0.4	9.22	82.60	5.93	77.78	5.46	60.00	9.69	88.00
	0.6	10.08	78.40	6.58	72.22	7.85	51.82	9.44	84.00
	0.8	10.94	52.20	8.26	60.05	10.49	30.00	9.98	70.00
1:9	0.2	9.73	72.60	6.28	70.90	6.16	44.09	8.70	76.00
	0.4	10.57	64.40	7.63	60.32	8.77	58.18	9.96	62.00
	0.6	10.90	52.80	8.94	53.17	10.05	41.82	10.42	58.00
	0.8	10.99	68.00	10.51	47.09	10.57	30.00	10.61	56.00

running prefix, and re-derives the remaining mismatches after each accepted swap, so that labels reflect the joint effect of substitutions rather than each token in isolation. We run this procedure on the 7,473-problem GSM8K (Cobbe et al., 2021) training split, with answer equivalence defined by the extracted final number, and on the 880 problems of the LiveCodeBench (Jain et al., 2025) `code_generation_lite` `release_v5` split, with equivalence defined by the benchmark's test suite. Unlike the original setup, which trains one classifier per task, we train a *single* judge on the union of both sources, since our evaluation spans four benchmarks rather than the one the judge was mined on.

F.3 Extended Analysis of Judge-based Verification

Table 6 sweeps the composition of the judge's mining data (GSM8K:LiveCodeBench, from math-dominated 9:1 to code-dominated 1:9) against the acceptance threshold.

Source quality dominates the efficiency-performance tradeoff. Degradation from 9:1 to 1:9 at 0.8 orders as MATH (-14.0), MBPP+ (-29.6), BFCL (-32.0), INCLUDE (-40.0), consistent with proximity to GSM8K, the source that survives at high math ratios.

G Block Efficiency and Distributional Gap under Tree Verification

This section proves Lemma 1 (Section G.1), and proves the distributional gap of Theorem 1 and the block efficiency of Proposition 1 through a per-position analysis (Lemma 2, Corollary 1). Throughout, p, q, p_Θ, z, Z_Θ, and $\mathcal{A}_\Theta$ refer to the current decoding step and are conditioned on all previously generated tokens, as in Section 2.3.

G.1 Proof of Truncation sampling efficiency effect

Proof. Fix a decoding step and condition on the prefix. Under standard speculative decoding with draft q and reference distribution $\tilde{p}$, the single-token acceptance probability is

$$A(\tilde{p}, q) = \sum_{x \in \mathcal{V}} q(x) \min\Big\{1, \frac{\tilde{p}(x)}{q(x)}\Big\} = \sum_{x \in \mathcal{V}} \min\big(\tilde{p}(x), q(x)\big). \tag{63}$$

Without truncation $\tilde{p} = p$; with truncation strategy Θ applied to the target, $\tilde{p} = p_\Theta$, i.e. $p_\Theta(x) = p(x)/z$ for $x \in \mathcal{A}_\Theta$ and 0 otherwise, where $z = \sum_{v \in \mathcal{A}_\Theta} p(v)$. The draft distribution is unmodified. Hence

$$\Delta\mathrm{BE} = A(p_\Theta, q) - A(p, q) = \sum_{x \in \mathcal{A}_\Theta} \Big[\min\big(\tfrac{p(x)}{z}, q(x)\big) - \min\big(p(x), q(x)\big)\Big] - \sum_{x \notin \mathcal{A}_\Theta} \min\big(p(x), q(x)\big), \tag{64}$$

since p_Θ vanishes off $\mathcal{A}_\Theta$.

It remains to evaluate the bracketed term for $x \in \mathcal{A}_\Theta$. Because $z \le 1$ we have $p(x)/z \ge p(x)$, and there are two cases.

Case $q(x) \le p(x)$. Then $\min(p(x), q(x)) = q(x)$ and $\min(p(x)/z, q(x)) = q(x)$, so the bracketed term is 0.

Case $q(x) > p(x)$. Then $\min(p(x), q(x)) = p(x)$, and

$$\min\Big(\frac{p(x)}{z}, q(x)\Big) - p(x) = \min\Big(\frac{p(x)}{z} - p(x),\ q(x) - p(x)\Big) = \min\Big(\Big(\frac{1}{z} - 1\Big)p(x),\ q(x) - p(x)\Big). \tag{65}$$

Both cases are covered by $\min\{(q(x) - p(x))_+,\ (\frac{1}{z} - 1)p(x)\}$, which vanishes when $q(x) \le p(x)$. Substituting into Equation equation 64 gives

$$\Delta\mathrm{BE} = \sum_{x \in \mathcal{A}_\Theta} \min\Big\{\big(q(x) - p(x)\big)_+,\ \Big(\frac{1}{z} - 1\Big)p(x)\Big\} - \sum_{x \notin \mathcal{A}_\Theta} \min\big(p(x), q(x)\big), \tag{66}$$

as claimed. □

G.2 Tree verification as implemented

EAGLE-3 drafts a token tree and verification accepts a single root-to-leaf path (Li et al., 2025). Greedy drafting fixes a deterministic candidate set $x^{(1)}, \ldots, x^{(D)}$ at each position (D fixed by the drafting configuration), examined in this order, with draft probabilities playing no role in verification. With a truncation warper active the verifier scores candidates by the truncated target p_Θ (Equation (13)), accepting each iff $r \le h(x^{(i)})$, $r \sim \mathrm{Unif}[0, 1]$:

- **Truncation sampling + SD (lossless).**

$$h(x) = \min\Big\{p_\Theta(x)/q,\ 1\Big\}\Big|_{q=1} = p_\Theta(x), \tag{67}$$

  with zero-and-renormalize on rejection, reproducing p_Θ at each position (Lemma 2(i)).
- **SpecCascade.**

$$h(x) = \mathbf{1}\Big[p_\Theta(x) \ge p_{\text{base}} \max_v p_\Theta(v)\Big] = \mathbf{1}[x \in \mathcal{A}_{\text{min-}p}]. \tag{68}$$

- **Typical acceptance.**

$$h(x) = \min\Big\{p_\Theta(x)/\tau,\ 1\Big\} = \mathbf{1}[x \in \mathcal{A}_\eta], \qquad \tau = \min\big(\varepsilon,\ \sqrt{\varepsilon}\, e^{-H(p_\Theta)}\big). \tag{69}$$

Equations 68 and 69 are the indicator of Definition 1 with $\Theta = \text{min-}p$ (Equation (5)) and $\Theta = \eta$ (Equation (6), $\delta = \sqrt{\varepsilon}$): off-set candidates are rejected and in-set candidates accepted with probability one. If a candidate is accepted the walk descends into its subtree; otherwise the path terminates with a bonus token from p_Θ. The accepted length L counts accepted drafted tokens only (the bonus excluded) and equals the block efficiency in Tables 9 and 10.

G.3 PER-POSITION ANALYSIS

Write $a(C) = \Pr[\text{some } x \in C \text{ accepted}]$ for the per-position acceptance probability of a rule, given the prefix.

Lemma 2. ***Per-position generation and acceptance*** *Let $C = \{x^{(1)}, \dots, x^{(D)}\}$ be the candidates at the current position, examined in order, with $p_\Theta(C) = \sum_{x \in C} p_\Theta(x)$. Under Section G.2:*

(i) Truncation sampling + SD emits p_Θ exactly, with

$$\Pr[\textit{accept } x^{(i)}] = p_\Theta(x^{(i)}), a^{\text{lossless}}(C) = p_\Theta(C). \tag{70}$$

(ii) Any rule with $\tilde{p} = p_\Theta$ accepts only tokens in $C \cap \mathcal{A}_\Theta$, with

$$\begin{aligned} a^{\text{lossy}}(C) = \mathbf{1}[C \cap \mathcal{A}_\Theta \neq \varnothing] \textit{ (SpecCasc.)}; a^{\text{lossy}}(C) \in (0, 1], \textit{ with } >0 \\ \iff C \cap \mathcal{A}_\Theta \neq \varnothing \textit{ (typical acc.).} \end{aligned} \tag{71}$$

Proof. (i) Initialize $\tilde{p}_1 = p_\Theta$; the zero-and-renormalize update gives $\tilde{p}_i = p_\Theta(\cdot \mid \mathcal{V} \setminus \{x^{(1)}, \dots, x^{(i-1)}\})$. Then

$$\begin{aligned} \Pr[\text{reach } x^{(i)}] &= \prod_{k=1}^{i-1} \big(1 - \tilde{p}_k(x^{(k)})\big) \\ &= \prod_{k=1}^{i-1} \frac{1 - \sum_{l \le k} p_\Theta(x^{(l)})}{1 - \sum_{l \le k-1} p_\Theta(x^{(l)})} \\ &= 1 - \sum_{l<i} p_\Theta(x^{(l)}), \\ \Pr[\text{accept } x^{(i)}] &= \Pr[\text{reach } x^{(i)}] \cdot \tilde{p}_i(x^{(i)}) \\ &= \Big(1 - \textstyle\sum_{l<i} p_\Theta(x^{(l)})\Big) \times \dfrac{p_\Theta(x^{(i)})}{1 - \sum_{l<i} p_\Theta(x^{(l)})} \\ &= p_\Theta(x^{(i)}), \\ \Rightarrow a^{\text{lossless}}(C) &= \sum_i p_\Theta(x^{(i)}) = p_\Theta(C) \end{aligned}$$

. On all-reject,

$$\Pr[\text{emit } x] = \big(1 - p_\Theta(C)\big) \frac{p_\Theta(x)}{1 - p_\Theta(C)} = p_\Theta(x),$$

for $x \notin C$, so the position emits p_Θ on all of $\mathcal{V}$.

(ii) With $\tilde{p} = p_\Theta$, Equation (68) and Equation (69) give

$$x \notin \mathcal{A}_\Theta \;\Rightarrow\; \tilde{p}(x) = 0 \;\Rightarrow\; h(x) = 0,$$

so accepted tokens lie in $C \cap \mathcal{A}_\Theta$. SpecCascade has $h \equiv 1$ on $\mathcal{A}_\Theta$ ($\Rightarrow$ first in-set candidate accepted surely); typical acceptance has $h(x) = \min\{p_\Theta(x)/\tau, 1\} > 0$ on $\mathcal{A}_\Theta$. This is Equation (71). □

Corollary 1. ***Tree verification accepts at least as often*** *At every position,*

$$a^{\text{lossless}}(C) = p_\Theta(C) = \frac{\sum_{v \in C \cap \mathcal{A}_\Theta} p(v)}{z} \le \mathbf{1}[C \cap \mathcal{A}_\Theta \neq \varnothing] = a^{\text{lossy}}_{\text{SpecCascade}}(C), \tag{72}$$

strict whenever $0 < p_\Theta(C) < 1$; for typical acceptance $a^{\text{lossless}} \le a^{\text{lossy}}$ holds accumulated along the path.

Proof. By Lemma 2, $a^{\text{lossless}}(C) = p_\Theta(C) \in [0, 1]$ and $p_\Theta(C) > 0 \iff C \cap \mathcal{A}_\Theta \neq \varnothing$, so $p_\Theta(C) \le \mathbf{1}[C \cap \mathcal{A}_\Theta \neq \varnothing]$, with equality iff $p_\Theta(C) \in \{0, 1\}$. □

G.4 Proof of Draft-target divergence under tree verification theorem

Proof. Write $p_\Theta(x) = p(x)/Z_\Theta(p)$ for $x \in \mathcal{A}_\Theta$ (zero otherwise), the truncated target; by Lemma 2(i) truncation sampling emits p_Θ at every position.

Statement 1 ($\mathrm{KL_{SD}}$*).* Truncation-based verification under standard SD generates $q/Z_\Theta(q)$ on $\mathcal{A}_\Theta$ (Equation (13)). The per-token term is

$$\begin{aligned} D_{\mathrm{KL}}\big(q/Z_\Theta(q) \,\big\|\, p_\Theta\big) &= \sum_{x\in\mathcal{A}_\Theta} \frac{q(x)}{Z_\Theta(q)} \log \frac{q(x)/Z_\Theta(q)}{p(x)/Z_\Theta(p)} \\ &= \sum_{x\in\mathcal{A}_\Theta} \frac{q(x)}{Z_\Theta(q)} \log \frac{q(x)\, Z_\Theta(p)}{Z_\Theta(q)\, p(x)}, \end{aligned}$$

the $\mathrm{KL_{SD}}$ of Theorem 1, finite since $p > 0$ on $\mathcal{A}_\Theta$, and

$$q = p \;\Rightarrow\; Z_\Theta(q) = Z_\Theta(p) \Rightarrow\; q/Z_\Theta(q) = p_\Theta \;\Rightarrow\; D_{\mathrm{KL}} = 0,$$

so $\mathrm{KL_{SD}} \to 0$ as $q \to p$ (Theorem 1).

Statement 2 ($\mathrm{KL_{EAGLE}}$*).* By Lemma 2(ii) the accepted token is the first candidate in $\mathcal{A}_\Theta$, deterministic given the prefix, with $p_\Theta(x) = p(x)/Z_\Theta(p) > 0$ and the accept test governed by p_Θ, not q. The position therefore emits the point mass at x, whose divergence from p_Θ is

$$\log \frac{1}{p_\Theta(x)} = \log \frac{Z_\Theta(p)}{p(x)} \;=\; \mathrm{KL_{EAGLE}},$$

for SpecCascade and typical acceptance alike. Finally,

$$|\mathcal{A}_\Theta| > 1 \;\Rightarrow\; p_\Theta(x) \le \max_{v\in\mathcal{A}_\Theta} p_\Theta(v) < 1 \Rightarrow\; \log \frac{Z_\Theta(p)}{p(x)} \ge \log \frac{1}{\max_{v\in\mathcal{A}_\Theta} p_\Theta(v)} > 0,$$

for every draft and uniformly in q, so $\mathrm{KL_{EAGLE}} > 0$ does not vanish as $q \to p$ (Theorem 1). □

G.5 Block Efficiency Analysis

Proposition 1. ***Block efficiency of tree verification*** *With L as above,*

$$\mathrm{BE} = \mathbb{E}[L] = \sum_{d\ge 1} \prod_{j=1}^{d} \Pr[L \ge j \mid L \ge j-1], \tag{73}$$

In Equation (73), every factor is at least as large under truncation-based verification as under truncation sampling + SD; hence its block efficiency is weakly higher. *(See Section G for the proof.)* However, the slight gain in BE is at the cost of even more severe performance degradation under multi-draft setting.

Proof of Proposition 1.

$$\mathrm{BE} = \mathbb{E}[L] = \sum_{d\ge 1} \Pr[L \ge d] = \sum_{d\ge 1} \prod_{j=1}^{d} \Pr[L \ge j \mid L \ge j-1],$$

by Equation (73), each factor being $a(C)$ of Lemma 2. By Corollary 1, $a^{\mathrm{lossy}} \ge a^{\mathrm{lossless}}$ (per-position and strict for SpecCascade when $0 < p_\Theta(C) < 1$; accumulated along the path for typical acceptance), so every factor is weakly larger and BE is weakly higher. □

G.6 Qualitative Examples

We hereby include some samples revealing the failure mode of truncation-based verifications against their matched baselines.

MBPP+ task 9 — wrong sentinel return

Prompt: *Write a python function to find the minimum number of rotations (greater than 0) required to get the same string.*

Matched baseline (min-p):

```
for i in range(1, n):
if temp[i:i+n] == s: return i
return n
```
✓

SpecCascade (min-p allowed set):

```
for i in range(1, n):
if temp[i:i+n] == s: return i
return 0
```
×

Fingerprint: both versions share identical body and loop logic; they disagree only on the post-loop sentinel. The draft defaults to `0`, which violates the "greater than 0" specification; the target returns `n`, the trivial full-rotation period.

BFCL `parallel_multiple_195` — math vs. Python notation

Ground-truth slot: `calculate_area_under_curve(function="x**2",...)`

Matched baseline (min-p): `function="x**2"` ✓

SpecCascade (min-p allowed set): `function="x^2"` ×

Fingerprint: the draft writes the math-class form `x^2`; the target writes the Python form `x**2` that the BFCL grader requires. This divergence recurs 14 times across the $(p_{\text{base}}, \text{seed})$ grid, always in the same direction.

MBPP+ task 637 — function signature mishread

Prompt: *Write a function to check whether the given amount has no profit and no loss.*

Hidden tests call the function as `noprofit_noloss(cost_price, selling_price)`.

Matched baseline (min-p):

```
def noprofit_noloss(cost_price, selling_price):
return cost_price == selling_price
```
✓

SpecCascade (min-p allowed set):

```
def noprofit_noloss(amount):
return amount == 0
```
×

Fingerprint: the draft latches onto the salient word *"amount"* in the prompt and emits a plausible-looking single-argument signature; the target reads the two-quantity nature of the problem and emits the matching two-argument signature.

H Extended Results

I Use of AI Assistants.

We used OpenAI Codex and Anthropic Claude to assist with language editing, code development, and debugging. All AI-generated text and code suggestions were reviewed, revised, and verified by the authors. The authors remain fully responsible for the implementation, experimental results, analysis, and final manuscript.

Table 7: Verification results across four benchmarks: MATH, MBPP+, INCLUDE, and BFCL. The target and draft models are Qwen/Qwen2.5-72B-Instruct-GPTQ-Int8 and Qwen/Qwen2.5-0.5B-Instruct-GPTQ-Int8, respectively.

Method	Param	MATH			MBPP+			INCLUDE			BFCL		
		BE	DS	Acc (%)	BE	DS	Pass@1 (%)	BE	DS	Acc (%)	BE	DS	Acc (%)
SD Baseline Leviathan et al. (2023)	–	$7.98_{\pm 0.02}$	$4.67_{\pm 0.02}$	$76.47_{\pm 1.53}$	$5.47_{\pm 0.06}$	$4.06_{\pm 0.03}$	$75.84_{\pm 0.40}$	$3.40_{\pm 0.03}$	$4.63_{\pm 0.01}$	$68.18_{\pm 0.00}$	$8.73_{\pm 0.01}$	$6.86_{\pm 0.00}$	$88.17_{\pm 0.58}$
Min-p Smpl. (Nguyen et al., 2024) + SD	0.1	$7.94_{\pm 0.02}$	$4.65_{\pm 0.02}$	$76.27_{\pm 1.53}$	$5.42_{\pm 0.08}$	$3.89_{\pm 0.14}$	$75.57_{\pm 0.40}$	$3.34_{\pm 0.04}$	$4.63_{\pm 0.01}$	$68.79_{\pm 0.69}$	$8.73_{\pm 0.01}$	$6.97_{\pm 0.18}$	$88.17_{\pm 0.58}$
	0.3	$7.99_{\pm 0.05}$	$4.64_{\pm 0.01}$	$76.67_{\pm 0.83}$	$5.56_{\pm 0.03}$	$3.90_{\pm 0.14}$	$76.19_{\pm 0.46}$	$3.42_{\pm 0.04}$	$4.62_{\pm 0.01}$	$68.18_{\pm 2.08}$	$8.85_{\pm 0.01}$	$6.94_{\pm 0.21}$	$88.33_{\pm 1.04}$
	0.5	$8.03_{\pm 0.02}$	$4.64_{\pm 0.01}$	$76.07_{\pm 1.17}$	$5.45_{\pm 0.07}$	$3.89_{\pm 0.14}$	$75.57_{\pm 0.31}$	$3.44_{\pm 0.02}$	$4.61_{\pm 0.02}$	$68.48_{\pm 1.46}$	$8.88_{\pm 0.02}$	$6.97_{\pm 0.19}$	$87.67_{\pm 0.29}$
	0.7	$8.08_{\pm 0.04}$	$4.64_{\pm 0.01}$	$76.27_{\pm 0.42}$	$5.59_{\pm 0.02}$	$3.88_{\pm 0.13}$	$76.01_{\pm 0.31}$	$3.47_{\pm 0.05}$	$4.63_{\pm 0.01}$	$66.52_{\pm 1.14}$	$8.89_{\pm 0.02}$	$6.96_{\pm 0.19}$	$87.67_{\pm 0.29}$
	0.9	$8.08_{\pm 0.10}$	$4.65_{\pm 0.01}$	$77.27_{\pm 1.27}$	$5.62_{\pm 0.04}$	$3.88_{\pm 0.14}$	$76.01_{\pm 0.15}$	$3.47_{\pm 0.04}$	$4.62_{\pm 0.01}$	$66.97_{\pm 1.31}$	$8.89_{\pm 0.01}$	$6.97_{\pm 0.18}$	$88.00_{\pm 0.00}$
SpecCascade (Narasimhan et al., 2024)	0.1	$8.46_{\pm 0.06}$	$4.53_{\pm 0.22}$	$76.87_{\pm 0.46}$	$5.64_{\pm 0.03}$	$3.89_{\pm 0.14}$	$76.46_{\pm 0.26}$	$3.57_{\pm 0.04}$	$4.63_{\pm 0.01}$	$64.85_{\pm 1.14}$	$8.87_{\pm 0.04}$	$6.83_{\pm 0.05}$	$89.00_{\pm 0.50}$
	0.3	$8.36_{\pm 0.04}$	$4.65_{\pm 0.01}$	$75.47_{\pm 0.70}$	$5.59_{\pm 0.04}$	$3.89_{\pm 0.14}$	$75.40_{\pm 0.26}$	$3.50_{\pm 0.03}$	$4.60_{\pm 0.01}$	$66.67_{\pm 3.44}$	$8.89_{\pm 0.01}$	$6.81_{\pm 0.02}$	$88.67_{\pm 0.29}$
	0.5	$8.12_{\pm 0.05}$	$4.66_{\pm 0.01}$	$74.87_{\pm 0.42}$	$5.51_{\pm 0.03}$	$3.90_{\pm 0.12}$	$74.87_{\pm 0.70}$	$3.40_{\pm 0.08}$	$4.61_{\pm 0.02}$	$68.33_{\pm 3.03}$	$8.85_{\pm 0.03}$	$6.88_{\pm 0.03}$	$88.00_{\pm 0.00}$
	0.7	$8.01_{\pm 0.06}$	$4.66_{\pm 0.01}$	$75.47_{\pm 0.70}$	$5.48_{\pm 0.01}$	$3.90_{\pm 0.14}$	$76.46_{\pm 0.53}$	$3.30_{\pm 0.04}$	$4.60_{\pm 0.01}$	$68.33_{\pm 1.05}$	$8.84_{\pm 0.03}$	$6.82_{\pm 0.03}$	$87.67_{\pm 0.29}$
	0.9	$7.84_{\pm 0.07}$	$4.66_{\pm 0.01}$	$75.47_{\pm 1.40}$	$5.41_{\pm 0.01}$	$3.89_{\pm 0.14}$	$76.19_{\pm 0.26}$	$3.25_{\pm 0.03}$	$4.62_{\pm 0.01}$	$65.91_{\pm 0.45}$	$8.84_{\pm 0.03}$	$5.28_{\pm 0.04}$	$88.17_{\pm 0.58}$
η Smpl. (Hewitt et al., 2022) + SD	0.05	$7.92_{\pm 0.04}$	$4.63_{\pm 0.01}$	$76.67_{\pm 1.85}$	$5.42_{\pm 0.08}$	$3.89_{\pm 0.14}$	$75.57_{\pm 0.40}$	$3.34_{\pm 0.05}$	$4.61_{\pm 0.02}$	$67.73_{\pm 1.82}$	$8.73_{\pm 0.01}$	$6.95_{\pm 0.18}$	$88.17_{\pm 0.58}$
	0.10	$7.94_{\pm 0.03}$	$4.64_{\pm 0.01}$	$76.47_{\pm 1.03}$	$5.49_{\pm 0.06}$	$3.88_{\pm 0.15}$	$76.54_{\pm 0.61}$	$3.31_{\pm 0.02}$	$4.62_{\pm 0.02}$	$68.79_{\pm 0.26}$	$8.74_{\pm 0.01}$	$6.82_{\pm 0.02}$	$88.17_{\pm 0.58}$
	0.15	$7.96_{\pm 0.03}$	$4.64_{\pm 0.02}$	$75.87_{\pm 2.21}$	$5.44_{\pm 0.02}$	$3.89_{\pm 0.15}$	$76.01_{\pm 0.31}$	$3.39_{\pm 0.05}$	$4.60_{\pm 0.03}$	$67.27_{\pm 0.45}$	$8.77_{\pm 0.01}$	$6.79_{\pm 0.05}$	$88.17_{\pm 0.58}$
	0.20	$8.11_{\pm 0.29}$	$4.51_{\pm 0.21}$	$76.27_{\pm 0.70}$	$5.57_{\pm 0.05}$	$3.88_{\pm 0.12}$	$76.01_{\pm 0.15}$	$3.42_{\pm 0.05}$	$4.61_{\pm 0.01}$	$68.33_{\pm 1.84}$	$8.80_{\pm 0.07}$	$6.80_{\pm 0.02}$	$88.17_{\pm 0.58}$
	0.25	$8.03_{\pm 0.06}$	$4.50_{\pm 0.21}$	$75.40_{\pm 1.60}$	$5.57_{\pm 0.04}$	$3.87_{\pm 0.14}$	$75.13_{\pm 0.26}$	$3.41_{\pm 0.02}$	$4.63_{\pm 0.03}$	$68.79_{\pm 1.60}$	$8.88_{\pm 0.05}$	$6.72_{\pm 0.07}$	$88.17_{\pm 0.58}$
Typical Acceptance Cai et al. (2024)	0.05	$8.49_{\pm 0.01}$	$4.67_{\pm 1.93}$	$76.60_{\pm 2.08}$	$5.64_{\pm 0.03}$	$3.89_{\pm 0.14}$	$76.46_{\pm 1.06}$	$3.58_{\pm 0.03}$	$4.62_{\pm 0.01}$	$66.52_{\pm 1.31}$	$8.87_{\pm 0.04}$	$6.84_{\pm 0.02}$	$89.00_{\pm 0.50}$
	0.10	$8.47_{\pm 0.05}$	$4.66_{\pm 0.00}$	$75.20_{\pm 1.11}$	$5.64_{\pm 0.03}$	$3.89_{\pm 0.13}$	$76.46_{\pm 1.06}$	$3.63_{\pm 0.03}$	$4.61_{\pm 0.01}$	$66.52_{\pm 1.05}$	$8.87_{\pm 0.04}$	$6.85_{\pm 0.01}$	$89.00_{\pm 0.50}$
	0.15	$8.21_{\pm 0.24}$	$4.65_{\pm 0.03}$	$75.73_{\pm 2.12}$	$5.66_{\pm 0.03}$	$3.88_{\pm 0.14}$	$75.66_{\pm 0.53}$	$3.59_{\pm 0.01}$	$4.62_{\pm 0.02}$	$66.06_{\pm 1.31}$	$8.87_{\pm 0.04}$	$6.85_{\pm 0.03}$	$89.00_{\pm 0.50}$
	0.20	$8.39_{\pm 0.01}$	$4.65_{\pm 0.00}$	$74.67_{\pm 1.55}$	$5.63_{\pm 0.05}$	$3.90_{\pm 0.14}$	$75.40_{\pm 0.26}$	$3.54_{\pm 0.03}$	$4.60_{\pm 0.01}$	$68.79_{\pm 0.69}$	$8.87_{\pm 0.04}$	$6.84_{\pm 0.02}$	$89.00_{\pm 0.50}$
	0.25	$8.30_{\pm 0.05}$	$4.67_{\pm 0.01}$	$75.53_{\pm 0.31}$	$5.55_{\pm 0.08}$	$3.90_{\pm 0.14}$	$75.04_{\pm 0.15}$	$3.46_{\pm 0.01}$	$4.63_{\pm 0.01}$	$66.06_{\pm 0.26}$	$8.89_{\pm 0.01}$	$6.78_{\pm 0.04}$	$88.67_{\pm 0.29}$
Lenience Leviathan et al. (2023)	0.2	$8.49_{\pm 0.02}$	$4.68_{\pm 0.05}$	$74.47_{\pm 0.58}$	$5.65_{\pm 0.03}$	$4.06_{\pm 0.02}$	$75.13_{\pm 0.26}$	$3.59_{\pm 0.04}$	$4.52_{\pm 0.04}$	$67.42_{\pm 1.39}$	$8.87_{\pm 0.01}$	$6.90_{\pm 0.02}$	$88.67_{\pm 0.29}$
	0.4	$8.37_{\pm 0.08}$	$4.66_{\pm 0.12}$	$75.60_{\pm 2.65}$	$5.61_{\pm 0.02}$	$4.06_{\pm 0.02}$	$75.49_{\pm 0.67}$	$3.55_{\pm 0.06}$	$4.53_{\pm 0.09}$	$67.88_{\pm 1.31}$	$8.85_{\pm 0.04}$	$6.87_{\pm 0.01}$	$88.00_{\pm 0.50}$
	0.6	$8.26_{\pm 0.05}$	$4.66_{\pm 0.14}$	$78.00_{\pm 1.73}$	$5.52_{\pm 0.07}$	$4.06_{\pm 0.03}$	$75.57_{\pm 0.40}$	$3.48_{\pm 0.02}$	$4.50_{\pm 0.18}$	$68.64_{\pm 1.82}$	$8.83_{\pm 0.02}$	$6.84_{\pm 0.02}$	$88.17_{\pm 0.29}$
	0.8	$8.22_{\pm 0.07}$	$4.65_{\pm 0.23}$	$76.47_{\pm 1.15}$	$5.52_{\pm 0.04}$	$4.06_{\pm 0.02}$	$75.75_{\pm 0.40}$	$3.38_{\pm 0.03}$	$4.49_{\pm 0.11}$	$68.33_{\pm 0.26}$	$8.81_{\pm 0.05}$	$6.87_{\pm 0.02}$	$87.67_{\pm 0.29}$
CoS Fu et al. (2025)	0.2	$8.33_{\pm 0.00}$	$4.68_{\pm 0.01}$	$71.53_{\pm 2.12}$	$6.05_{\pm 0.13}$	$4.10_{\pm 0.01}$	$71.43_{\pm 1.06}$	$3.89_{\pm 0.04}$	$4.61_{\pm 0.02}$	$64.55_{\pm 1.98}$	$8.76_{\pm 0.13}$	$6.79_{\pm 0.04}$	$70.67_{\pm 3.40}$
	0.4	$8.80_{\pm 0.04}$	$4.67_{\pm 0.00}$	$60.40_{\pm 2.84}$	$7.14_{\pm 0.09}$	$4.13_{\pm 0.01}$	$61.20_{\pm 3.29}$	$4.69_{\pm 0.07}$	$4.58_{\pm 0.03}$	$53.79_{\pm 2.10}$	$8.84_{\pm 0.15}$	$6.69_{\pm 0.03}$	$58.00_{\pm 4.50}$
	0.6	$9.42_{\pm 0.01}$	$4.67_{\pm 0.01}$	$54.80_{\pm 0.72}$	$7.86_{\pm 0.02}$	$4.20_{\pm 0.02}$	$59.44_{\pm 0.85}$	$5.99_{\pm 0.03}$	$4.53_{\pm 0.01}$	$48.03_{\pm 1.89}$	$9.29_{\pm 0.16}$	$6.69_{\pm 0.03}$	$43.83_{\pm 3.79}$
	0.8	$10.14_{\pm 0.11}$	$4.65_{\pm 0.01}$	$46.00_{\pm 2.12}$	$9.26_{\pm 0.04}$	$4.26_{\pm 0.00}$	$54.06_{\pm 0.31}$	$8.02_{\pm 0.05}$	$4.49_{\pm 0.01}$	$37.12_{\pm 2.05}$	$10.26_{\pm 0.07}$	$6.75_{\pm 0.01}$	$43.67_{\pm 1.89}$
Auto Judge Garipov et al. (2025)	0.2	$9.22_{\pm 0.02}$	$4.67_{\pm 0.12}$	$85.80_{\pm 0.20}$	$5.89_{\pm 0.03}$	$4.09_{\pm 0.02}$	$76.19_{\pm 0.26}$	$4.11_{\pm 0.11}$	$4.58_{\pm 0.03}$	$60.45_{\pm 0.45}$	$8.79_{\pm 0.05}$	$6.81_{\pm 0.02}$	$84.50_{\pm 0.50}$
	0.4	$9.71_{\pm 0.01}$	$4.65_{\pm 0.02}$	$82.80_{\pm 0.20}$	$5.93_{\pm 0.04}$	$4.08_{\pm 0.01}$	$73.54_{\pm 0.26}$	$4.84_{\pm 0.07}$	$4.57_{\pm 0.05}$	$60.00_{\pm 0.45}$	$8.65_{\pm 0.04}$	$6.78_{\pm 0.03}$	$78.50_{\pm 0.50}$
	0.6	$10.34_{\pm 0.11}$	$4.69_{\pm 0.03}$	$75.40_{\pm 0.40}$	$6.22_{\pm 0.09}$	$4.07_{\pm 0.03}$	$70.11_{\pm 0.53}$	$6.69_{\pm 0.08}$	$4.55_{\pm 0.02}$	$50.00_{\pm 0.91}$	$8.95_{\pm 0.10}$	$6.76_{\pm 0.02}$	$66.00_{\pm 0.50}$
	0.8	$10.84_{\pm 0.13}$	$4.64_{\pm 0.01}$	$65.20_{\pm 0.40}$	$8.32_{\pm 0.03}$	$4.06_{\pm 0.02}$	$60.58_{\pm 0.53}$	$9.07_{\pm 0.05}$	$4.52_{\pm 0.04}$	$38.64_{\pm 0.91}$	$9.65_{\pm 0.07}$	$6.74_{\pm 0.01}$	$47.00_{\pm 1.00}$

Table 8: Verification results across four benchmarks: MATH, MBPP+, INCLUDE, and BFCL. The target and draft models are `meta-llama/Llama-3.1-8B-Instruct` and `meta-llama/Llama-3.2-1B-Instruct`, respectively. **Bold** denotes the best value in each column, and underline denotes the second-best distinct value.

Method	Param	MATH			MBPP+			INCLUDE			BFCL		
		BE	DS	Acc (%)	BE	DS	Pass@1 (%)	BE	DS	Acc (%)	BE	DS	Acc (%)
SD Baseline Leviathan et al. (2023)	–	7.38	67.34	74.00	5.76	61.36	57.67	3.50	54.38	37.27	7.06	40.47	80.50
Min-p Sampling (Nguyen et al., 2024) + SD	0.1	7.55	70.64	74.20	6.00	63.38	60.32	4.07	69.89	43.18	7.37	41.64	80.50
	0.3	7.62	71.19	76.80	5.96	64.30	58.99	4.57	70.20	37.73	7.34	41.03	81.50
	0.5	7.65	71.31	74.20	6.00	64.15	58.20	4.55	70.29	**44.09**	7.20	40.89	83.00
	0.7	7.67	71.62	77.00	6.10	63.91	**60.85**	4.64	70.25	41.36	7.27	41.36	**84.00**
	0.9	7.77	71.14	75.80	6.18	63.31	60.58	4.92	69.79	42.27	7.36	40.41	83.50
SpecCascade Narasimhan et al. (2024)	0.1	9.23	72.22	72.00	6.96	64.37	59.26	5.79	71.35	36.82	7.74	40.95	81.00
	0.3	8.27	72.19	75.20	6.26	64.50	60.32	4.70	71.53	36.82	7.50	40.49	83.00
	0.5	7.71	73.13	76.80	6.14	65.16	59.26	4.60	71.33	40.45	7.36	41.90	81.00
	0.7	7.30	72.85	76.60	5.93	65.31	59.79	4.44	71.19	37.73	7.31	41.04	82.00
	0.9	6.96	73.19	76.20	5.91	65.31	59.79	4.13	71.78	38.18	7.24	41.92	77.50
η Smpl. (Hewitt et al., 2022) + SD	0.05	9.09	84.16	71.20	6.79	70.08	58.20	4.75	73.42	37.73	7.37	40.12	79.00
	0.10	8.92	82.36	70.60	6.71	71.30	58.99	4.63	72.59	34.55	7.28	39.67	78.50
	0.15	8.52	78.46	76.60	6.36	66.13	58.20	4.35	64.32	35.00	7.21	38.27	81.00
	0.20	8.19	74.81	76.40	6.29	65.40	58.99	4.25	64.10	32.73	7.24	40.24	81.50
	0.25	8.00	73.05	75.20	6.10	62.91	60.58	4.18	63.73	34.09	7.19	40.06	82.50
Typical Acceptance Cai et al. (2024)	0.05	7.51	69.34	75.60	5.96	62.26	57.67	4.04	61.41	39.55	7.41	40.42	80.00
	0.10	7.53	68.73	72.00	6.00	61.88	58.20	4.10	63.53	38.64	7.32	41.92	80.50
	0.15	7.68	72.12	75.60	6.01	61.98	60.58	4.29	63.42	39.09	7.31	38.86	81.00
	0.20	7.66	71.93	75.40	5.96	61.46	60.32	4.28	64.27	32.73	7.33	42.97	82.00
	0.25	7.66	72.93	74.20	5.99	60.78	58.99	4.34	65.19	34.09	7.30	39.81	**84.00**
Lenience Leviathan et al. (2023)	0.2	9.02	83.33	72.20	6.49	67.02	57.67	4.58	70.85	36.36	7.36	40.14	80.50
	0.4	8.74	88.93	74.40	6.43	69.39	60.58	4.40	68.20	35.45	7.26	40.98	81.00
	0.6	8.45	78.05	**79.20**	6.25	65.39	**60.85**	3.97	60.24	34.55	7.18	39.14	80.00
	0.8	7.94	72.52	77.40	6.08	64.72	**60.85**	3.85	59.71	33.18	7.13	38.86	83.00
CoS Fu et al. (2025)	0.2	8.00	73.08	73.40	6.55	67.64	52.91	4.76	71.59	34.55	7.63	42.45	61.00
	0.4	8.74	80.73	64.20	7.48	77.39	51.06	6.08	91.72	33.64	8.26	45.17	50.00
	0.6	9.53	87.25	61.40	8.47	89.77	45.50	7.50	112.38	28.64	9.10	49.87	39.50
	0.8	10.27	94.59	57.00	9.74	101.28	42.06	9.29	138.47	29.55	10.03	55.27	38.50
Auto Judge Garipov et al. (2025)	0.2	8.61	73.94	77.00	6.80	74.78	53.17	8.84	67.68	30.00	8.92	47.03	53.00
	0.4	9.89	71.86	67.60	9.52	74.89	43.92	10.80	67.36	25.00	10.44	46.89	38.00
	0.6	10.67	72.22	56.60	10.70	73.76	39.68	10.91	66.71	25.00	10.81	46.27	32.50
	0.8	10.97	72.36	53.00	10.99	72.85	38.36	10.92	66.99	28.64	10.92	45.52	27.50

Table 9: EAGLE-3 speculative decoding results across four benchmarks (LLaMA-3.1 8B, temperature = 0.7, block size= 7). **Bold** marks the best value per column across all methods; underline marks the second best.

Method	Param	MATH			MBPP+			INCLUDE			BFCL		
		BE	DS	Acc (%)	BE	DS	Pass@1 (%)	BE	DS	Acc (%)	BE	DS	Acc (%)
Baseline	—	3.76	140.21	73.20	4.70	167.83	59.26	0.67	45.72	35.45	2.57	85.27	86.00
Lenience Leviathan et al. (2023)	0.2	4.49	161.29	71.20	5.17	181.95	61.11	0.86	50.61	30.91	2.63	**87.31**	83.50
	0.4	4.34	157.03	76.20	5.02	177.13	59.79	0.80	48.93	32.73	2.61	86.14	85.50
	0.6	4.19	152.85	76.00	4.96	175.43	59.52	0.73	47.03	34.55	2.59	85.77	86.00
	0.8	4.07	149.39	75.20	4.92	174.25	59.79	0.72	46.73	33.64	2.59	85.56	84.00
Min-p Sampling (Nguyen et al., 2024) + SD	0.1	3.89	140.72	75.20	4.84	166.79	61.90	0.54	45.34	40.00	2.58	84.45	86.00
	0.3	3.96	142.85	**77.60**	4.89	168.29	60.32	0.55	45.65	36.82	2.58	84.13	86.00
	0.5	3.98	143.42	77.40	4.91	168.98	60.85	0.56	45.91	35.91	2.58	84.09	86.50
	0.7	4.00	143.88	77.40	4.92	169.23	62.43	0.56	45.96	35.91	2.59	84.21	**87.50**
	0.9	**4.05**	**145.24**	76.80	**4.93**	**169.58**	63.49	**0.57**	**46.13**	36.36	2.59	84.42	87.00
SpecCascade Narasimhan et al. (2024)	0.1	4.47	159.83	73.00	5.16	180.81	59.79	0.92	51.76	30.91	2.62	85.82	84.50
	0.3	4.28	154.42	77.40	5.03	176.79	**61.38**	0.79	48.26	31.82	2.59	85.18	84.50
	0.5	4.18	151.56	**77.60**	4.99	175.66	60.85	0.76	47.43	31.82	2.58	85.03	86.50
	0.7	4.12	149.71	76.20	4.93	173.70	**61.38**	0.73	46.74	**36.82**	2.58	84.99	85.50
	0.9	4.06	148.20	76.60	4.89	172.60	60.32	0.72	46.45	35.00	2.58	85.00	86.00
η Smpl. (Hewitt et al., 2022) + SD	0.05	3.85	137.92	75.20	4.84	164.79	59.79	0.55	45.42	35.00	2.58	83.63	**87.50**
	0.10	3.90	139.33	77.00	4.86	165.32	61.11	0.54	45.06	**41.82**	2.59	83.92	86.00
	0.15	3.93	140.21	77.20	4.88	165.95	61.90	0.54	45.13	35.00	2.59	83.91	86.50
	0.20	3.96	140.97	76.00	4.89	166.23	**64.29**	0.54	45.14	34.09	2.57	83.54	85.00
	0.25	3.99	141.83	75.20	4.88	165.77	63.76	0.57	45.92	37.73	2.58	83.85	86.50
Typical Acceptance Cai et al. (2024)	0.05	**4.82**	**168.77**	66.00	**5.29**	**182.86**	55.29	**1.24**	**59.80**	29.55	**2.66**	86.11	78.00
	0.10	4.66	164.19	68.20	5.21	180.54	56.08	1.15	57.20	29.55	2.65	85.85	79.50
	0.15	4.58	161.86	72.20	5.16	178.87	57.41	1.05	54.76	25.91	2.64	85.65	83.00
	0.20	4.54	160.65	69.80	5.16	178.94	58.47	0.98	52.73	27.73	2.63	85.40	83.50
	0.25	4.46	158.17	73.00	5.17	179.31	57.14	1.00	53.21	26.82	2.62	85.05	83.00

Table 10: EAGLE-3 speculative decoding results across four benchmarks (Qwen3-8B, temperature = 0.7, block size= 7). **Bold** marks the best value per column across all methods; underline marks the second best.

Method	Param	MATH			MBPP+			INCLUDE			BFCL		
		BE	DS	Acc (%)	BE	DS	Pass@1 (%)	BE	DS	Acc (%)	BE	DS	Acc (%)
Baseline	—	4.54	162.99	81.60	4.68	163.24	51.32	1.81	117.06	46.36	2.94	95.97	81.50
Lenience Leviathan et al. (2023)	0.2	4.92	177.76	82.80	5.19	179.13	48.68	1.96	128.99	46.82	2.99	97.64	79.50
	0.4	4.84	175.13	83.20	5.01	174.05	51.85	1.96	126.66	47.73	2.98	97.83	80.50
	0.6	4.74	170.87	81.20	4.93	171.24	53.97	1.87	124.63	44.09	2.97	96.88	79.00
	0.8	4.69	169.67	85.20	4.83	166.80	52.38	1.86	117.87	44.55	2.95	96.50	82.00
Min-p Sampling (Nguyen et al., 2024) + SD	0.1	4.60	166.01	84.80	4.75	165.56	53.97	1.90	127.20	45.91	2.96	97.14	85.00
	0.3	4.65	168.24	84.60	4.75	164.18	52.12	1.81	123.75	47.27	2.96	96.80	85.50
	0.5	4.66	167.17	85.60	4.81	167.42	53.17	1.86	123.60	41.36	2.96	96.47	85.50
	0.7	4.64	165.93	84.80	4.77	164.67	55.82	1.88	133.99	45.00	2.96	96.50	85.50
	0.9	4.66	166.33	85.40	4.75	165.85	55.82	1.87	137.57	43.64	2.96	97.15	85.50
SpecCascade Narasimhan et al. (2024)	0.1	4.94	177.01	83.00	5.19	179.58	49.74	1.97	127.06	43.64	2.98	97.31	80.50
	0.3	4.77	172.16	83.40	4.95	171.89	53.17	1.92	137.64	44.09	2.97	97.45	79.50
	0.5	4.72	169.82	84.20	4.82	168.04	53.97	1.89	121.72	45.00	2.96	96.80	79.00
	0.7	4.69	169.53	85.20	4.78	167.23	52.65	1.83	133.27	47.27	2.95	96.82	80.00
	0.9	4.66	168.22	83.60	4.75	164.58	57.14	1.85	125.19	43.18	2.94	96.36	78.50
η Smpl. (Hewitt et al., 2022) + SD	0.05	4.59	165.41	84.40	4.69	162.52	54.50	1.80	129.36	50.00	2.95	96.80	84.50
	0.10	4.63	165.44	84.60	4.76	166.15	55.82	1.82	126.64	51.36	2.96	96.79	85.50
	0.15	4.64	167.65	**86.00**	4.73	164.01	54.76	1.97	140.23	49.55	2.96	96.61	84.50
	0.20	4.64	167.49	84.40	4.73	164.25	57.41	1.86	130.50	48.18	2.96	96.54	**86.00**
	0.25	4.64	167.67	85.00	4.79	167.47	50.79	1.84	118.36	43.64	2.95	96.86	84.50
Typical Acceptance Cai et al. (2024)	0.05	**4.98**	**179.56**	80.40	**5.23**	**181.43**	49.21	2.29	**165.55**	46.82	2.98	97.49	80.50
	0.10	4.87	175.36	83.40	5.16	178.41	49.21	**2.39**	150.90	**57.73**	**3.03**	**99.11**	79.00
	0.15	4.86	175.56	85.20	4.90	171.26	**70.11**	2.19	143.56	52.73	3.02	98.39	79.00
	0.20	4.81	173.48	84.60	5.07	176.31	66.67	1.90	118.54	50.45	2.94	96.28	84.50
	0.25	4.80	173.41	84.00	4.93	170.26	69.31	1.90	138.16	46.82	3.01	98.32	77.00