PRESTO: PREFIX-ALIGNED TREE DRAFTING FOR DIFFUSION SPECULATIVE DECODING

Anonymous authors
Paper under double-blind review

ABSTRACT

Diffusion LLMs (dLLMs) have emerged as effective draft models for speculative decoding (SD), generating a block of draft tokens in a single forward pass. However, existing diffusion drafters follow a single draft path, even though dLLMs produce multiple candidates at every position, inducing a large combinatorial space of paths; exploring only a tiny fraction of it fundamentally limits acceptance length. We introduce tree-based drafting for diffusion drafters to exploit this multi-candidate structure, but find that naive tree drafting is suboptimal: diffusion marginals are inherently prefix-blind, mismatching prefix-based AR verification and yielding unreliable path ranking. We propose PRESTO (***PRE****fix-aligned* ***S****coring and priority-based* ***T****ree search for diffusion speculative dec****O****ding*), a principled framework built on two principles: (1) candidate ranking should align with the prefix-based nature of AR verification, and (2) tree construction should prioritize paths with high verification potential to maximize acceptance length. PRESTO exhibits strong generalizability across diffusion drafting paradigms, applying unchanged as a drop-in framework to both dedicated diffusion drafters and self-speculative dLLMs that draft and verify within a single forward pass. Across diverse benchmarks, PRESTO achieves average end-to-end speedups of **1.5**× over the diffusion drafter DFlash and up to **2.14**× over self-speculative dLLMs.

1 INTRODUCTION

Diffusion Large Language Models (dLLMs) have recently emerged as a promising alternative to autoregressive (AR) LLMs, introducing a new paradigm for multi-token generation (Nie et al., 2025; Ye et al., 2025a; Wu et al., 2025a; Bie et al., 2026; Cheng et al., 2025; Fu et al., 2026b). However, achieving high-quality generation in practice typically requires iterative denoising procedures and carefully designed token unmasking strategies, which partially offset the efficiency gains (Wu et al., 2025a;b). While, recent works have shown that diffusion models are particularly suited as draft models in speculative decoding (SD) (Liu et al., 2025; Li et al., 2025a; Chen et al., 2026; Yu et al., 2026). Their ability to generate multiple candidate tokens in parallel naturally favors the drafting stage of SD, enabling highly efficient draft token generation. This insight has led to a growing line of work that leverages diffusion models as drafters, including dedicated diffusion-based drafter for AR LLMs (e.g., dFlash (Chen et al., 2026)) and self-speculative dLLMs (e.g., TiDAR (Liu et al., 2025), I-DLM (Yu et al., 2026), Nemotron-Labs-Diffusion (Fu et al., 2026a)).

However, the above diffusion drafters rely on *linear (single-path) drafting*, committing to the top-1 token at each position. This is inherently suboptimal: diffusion models predict all positions in parallel from independent marginals, so multiple plausible candidates coexist at each position, and the path with the longest acceptance need not coincide with the top-1 marginal trajectory. Empirically, even strong diffusion drafters such as DFlash achieve only modest acceptance lengths under single-path drafting (Figure 2(a)). A natural remedy is to explore multiple candidate paths jointly. For AR drafter based SD methods, tree-based drafting substantially increases acceptance length (Miao et al., 2024; Cai et al., 2024; Li et al., 2024; 2025b; Svirschevski et al., 2024; Chen et al., 2025), since verification is memory-bound at low concurrency (Liu et al., 2025) and extra candidates can be verified at little additional cost. This success raises a key question:

Can we leverage tree-based drafting in diffusion drafters to further improve the end-to-end throughput of diffusion-based speculative decoding by increasing acceptance length?

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.

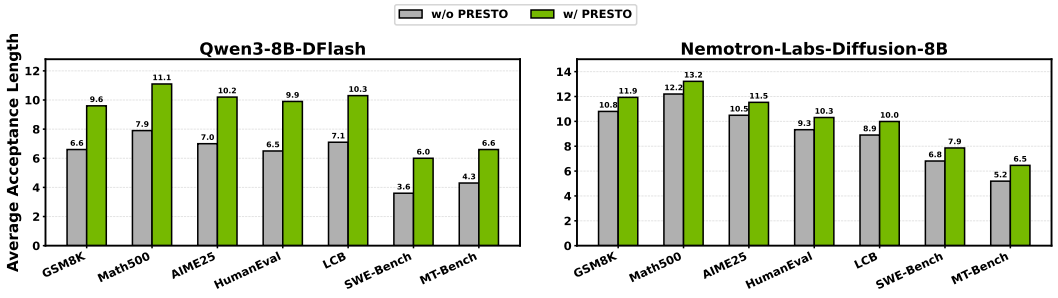


Figure 1: Acceptance length comparison of PRESTO on diffusion-based SD, including dedicated diffusion drafter SD (Qwen3-8B-DFlash) and self-speculative dLLM (Nemotron-Labs-Diffusion-8B) across diverse benchmarks. PRESTO consistently improves acceptance length over the baseline across all tasks, enabling more tokens to be accepted in each verification round.

While this direction is promising, we find that naively applying tree-based drafting to diffusion drafters is suboptimal. Diffusion drafters predict all positions of a block in parallel, producing a *marginal* distribution at each position. These marginals already capture meaningful global acceptance trends, and higher-confidence draft tokens are substantially more likely to be accepted (Figure 2(b)). However, two challenges remain. **(1) Scoring mismatch.** Because marginal scores are not conditioned on the preceding draft tokens, they cannot reflect how earlier token choices affect downstream acceptance, as a token that is likely in isolation may become unlikely given the chosen prefix (Figure 2(c)). This creates a fundamental mismatch between draft scoring and prefix-based AR verification. **(2) Budget allocation.** Even with reliable scores, the candidate space grows combinatorially with block size, while the verification budget is limited. An ideal tree drafting method should therefore maximize the expected acceptance length under a given tree budget.

To address these challenges, we propose PRESTO, a principled tree-based diffusion drafting framework via *prefix-aligned scoring* and *priority-based tree search*. The key principles behind PRESTO are that (1) candidate ranking should align with the prefix-based nature of AR verification, and (2) tree construction should prioritize candidate paths with high verification potential to maximize acceptance length. We minimally adjust the diffusion marginal with a prefix-dependent correction. We find that this correction does not require an expensive model: a lightweight, plug-and-play n-gram built on the target model's own generations suffices, as it directly approximates what the target will accept while being training-free and requiring no extra model forwards. The resulting score guides a priority-based expansion that allocates the tree budget to high-potential paths to maximize the expected acceptance length. More crucially, PRESTO exhibits generalizability across different diffusion drafting paradigms, applying unchanged to both dedicated diffusion drafters for AR targets and self-speculative dLLMs that draft and verify within a single forward pass. We thus summarize our contributions as follows:

- **We identify two key challenges in tree-based diffusion drafting.** Diffusion marginals provide informative acceptance signals, yet they are not conditioned on preceding draft tokens, creating a fundamental mismatch with prefix-based AR verification. We further show that, since the candidate space grows combinatorially, how to allocate a limited tree budget is critical to acceptance length.
- **We propose prefix-aligned scoring with a target-aligned n-gram.** We introduce a minimal, prefix-dependent correction to diffusion marginals, and show that a lightweight, plug-and-play n-gram built on the target's own generations suffices to provide this correction, preserving the drafter's global signal while aligning with prefix-based verification.
- **We propose priority-based tree search to maximize acceptance length under a fixed budget.** We formulate tree construction as maximizing the expected acceptance length under a given node budget, and solve it via priority-based expansion guided by prefix-aligned scores, which allocates the budget to the branches that contribute most to acceptance.
- **PRESTO is agnostic to the diffusion drafting paradigm.** PRESTO serves as a drop-in replacement for linear drafting in both dedicated diffusion drafters and self-speculative dLLMs, with a single shared hyperparameter. Across diverse benchmarks, it achieves average end-to-end speedups of **1.5**$\times$ over DFlash and up to **2.14**$\times$ over self-speculative dLLMs.

2 BACKGROUND AND PRELIMINARIES

2.1 TREE-BASED SPECULATIVE DECODING

Speculative decoding accelerates autoregressive generation of a target LLM p_T by leveraging a lightweight draft model. To improve acceptance, the draft model can construct a token tree encoding multiple candidate sequences. The target LLM verifies candidates in a *prefix-based* manner: starting from the root, it accepts tokens sequentially until a mismatch occurs.

Acceptance factorization. For a candidate path $P = (x_1, \ldots, x_k)$, let $a_i \in \{0, 1\}$ denote the indicator that the i-th token is accepted. Acceptance proceeds left-to-right, so by the chain rule

$$\Pr(P \text{ accepted}) = \prod_{i=1}^{k} \Pr\big(a_i = 1 \;\big|\; a_{<i} = 1,\, x_{\le i}\big), \tag{1}$$

where $a_{<i} = 1$ denotes the event that all earlier tokens have been accepted. Each factor depends on the realized prefix $x_{<i}$, making verification intrinsically prefix-conditional.

Tree construction objective. Let $\mathcal{B}_B = \{\mathcal{T} : |\mathcal{T}| \le B\}$ denote token trees with at most B nodes, and let $\alpha_{\mathcal{T}}(P)$ denote the longest prefix of P contained in $\mathcal{T}$. The ideal objective is

$$\mathcal{T}^* \in \arg\max_{\mathcal{T} \in \mathcal{B}_B} \mathbb{E}_{P \sim p_T}[\alpha_{\mathcal{T}}(P)]. \tag{2}$$

Direct optimization is infeasible, as evaluating p_T requires target-model forward passes. We use a surrogate distribution $\tilde{p}$ over candidate paths. A standard decomposition (Appendix A) yields

$$\mathbb{E}_{P \sim \tilde{p}}[\alpha_{\mathcal{T}}(P)] = \sum_{u \in \mathcal{T}} \tilde{p}(u), \tag{3}$$

where $\tilde{p}(u)$ denotes the probability that prefix u matches the sampled path. This reduces tree construction to selecting prefixes with large probability mass. Since $\tilde{p}(u)$ factorizes along the prefix,

$$\log \tilde{p}(u) = \sum_{i=1}^{|u|} \log \tilde{p}(u_i \mid u_{<i}), \tag{4}$$

providing an additive scoring rule that enables priority-based expansion.

2.2 AUTOREGRESSIVE VS. DIFFUSION SURROGATE DISTRIBUTIONS

Since verification is prefix-conditional (Eq. equation 1), a surrogate $\tilde{p}$ can faithfully rank paths only if its conditionals $\tilde{p}(u_i \mid u_{<i})$ genuinely depend on the realized prefix. We call such a surrogate *prefix-aligned* (formal definition in Appendix B). Existing tree drafting methods for AR drafters implicitly assume autoregressive prefix-conditioned scores (Miao et al., 2024; Li et al., 2024; Sun et al., 2024; Chen et al., 2025). This assumption fundamentally breaks under diffusion drafting. An AR draft model p_D defines $\tilde{p}_{\text{AR}}(P) := \prod_{i=1}^{k} p_D(x_i \mid x_{<i})$. Each conditional is a genuine function of the prefix, so $\tilde{p}_{\text{AR}}$ is prefix-aligned and provides an effective surrogate score.

Diffusion drafting yields a prefix-blind marginal. A diffusion drafter produces position-wise marginals $q_1, \ldots, q_k$, where each $q_i(x_i)$ does not condition on $x_{<i}$.[1] The factorized surrogate is thus

$$\tilde{p}_{\text{diff}}(P) := \prod_{i=1}^{k} q_i(x_i). \tag{5}$$

Since $\tilde{p}_{\text{diff}}(x_i \mid x_{<i}) = q_i(x_i)$ regardless of $x_{<i}$, the marginal surrogate violates Definition B.1. This mismatch with prefix-based verification is *intrinsic* to the marginal factorization, and while it does not preclude effective drafting, it can degrade path ranking (Sec 3).

3 OBSERVATIONS

We analyze diffusion-based draft scoring using Qwen3-4B-DFlash (block size 16), with EAGLE's tree expansion (Li et al., 2024; 2025b) as the default tree construction. For each candidate token, we

[1] Dependence on the prompt and previously generated context, encoded in a shared latent, is suppressed for notational clarity.

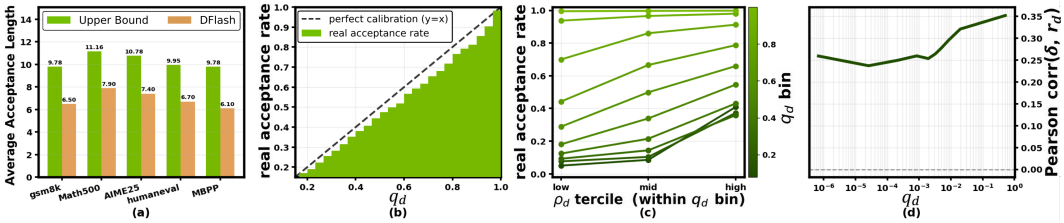


Figure 2: (a) Linear drafting leaves room for improvement via multi-path exploration. (b) The predicted confidence from q_d closely matches the empirical acceptance rate. (c) Within fixed q_d bins, empirical acceptance rates increase from low to high ρ_d terciles, indicating that prefix-conditioned compatibility captures predictive signal beyond the marginal confidence q_d. (d) ρ_d provides a correction direction aligned with the error of q_d, although it remains noisy (correlation < 1).

record its diffusion marginal q_d, its prefix-conditioned score ρ_d (instantiated with an n-gram model to reflect prefix-based AR verification), and whether it is accepted. Unless otherwise specified, results are aggregated across datasets.

Observation 1: Linear drafting leaves room for improvement via multi-path exploration. We compare linear drafting against an empirical upper bound of acceptance length. To estimate the upper bound, we retain the top-10 candidates per position and perform position-wise oracle matching: we check whether the ground-truth token at each position falls within its top-10 set, then take the longest contiguous prefix covered by these matches. Figure 2(a) shows a substantial gap between linear drafting and this upper bound across all tasks, indicating that a significant portion of the proposal space remains unexplored.

Observation 2: Marginal confidence correlates with overall acceptance. We further examine whether the diffusion draft model provides reliable estimates of acceptance. As shown in Figure 2(b), the predicted confidence (computed as the exponentiated cumulative log-probability along a path) closely matches the empirical acceptance rate. This indicates that q_d provides an informative estimate of global acceptance likelihood. At the same time, the calibration curve consistently remains below the ideal $y = x$ line, indicating that marginal confidence alone is still imperfect and does not fully characterize verifier acceptance.

Observation 3: Marginal signals are insufficient but can be improved by prefix conditioning. Despite this global calibration, q_d alone is insufficient for token-level ranking: Figure 2(c) shows that tokens with similar q_d yield markedly different acceptance rates when grouped by ρ_d. Yet the additional signal is directionally informative: Figure 2(d) shows the correction term $\delta = \log \rho_d - \log q_d$ positively correlates with the true residual $r_q = \log p_T - \log q_d$, so ρ_d provides a noisy but directionally aligned correction.

Insights. q_d provides a well-calibrated estimate of acceptance rate, but is insufficient for accurate path ranking. The prefix-conditioned signal ρ_d offers complementary information that aligns with the error of q_d, but is itself noisy. Together, these observations suggest that *an effective scoring should preserve q_d while incorporating a controlled correction based on ρ_d.*

4 PRESTO

We now introduce PRESTO, a tree-based diffusion drafting framework that augments diffusion drafter with prefix-aligned information, enabling more effective candidate exploration under prefix-conditioned AR verification. In Sec 4.1, we introduce prefix-aligned scoring that addresses the aforementioned structural mismatch. In Sec 4.2, we describe tree drafting via priority-based tree search. In Sec 4.3, we elaborate how to extend tree drafting to self-speculative dLLMs.

4.1 Prefix-Aligned Scoring via Minimal Correction

From surrogate distribution to scoring function. Sec. 2 casts tree construction as ranking prefixes by $\log \tilde{p}(u)$ (Eq. 4). For diffusion drafting, the marginal surrogate $\tilde{p}_{\text{diff}}(P) = \prod_i q_i(x_i)$ is prefix-blind (Eq. 5), which mismatches prefix-based verification. Guided by Sec. 3, we seek a prefix-aligned surrogate $\tilde{p}^\star$ that keeps the calibrated drafter signal and adds a prefix-conditioned correction.

Prefix-aligned surrogate. Let $q_d(t)$ denote the diffusion drafter's marginal at position d, and let $\rho_d(t \mid c_d)$ denote a tractable prefix-conditioned signal, where $c_d = (t_1, \ldots, t_{d-1})$ is the within-block prefix. We define the prefix-aligned conditional as a multiplicative combination:

$$p_d^\star(t \mid c_d) \propto q_d(t)\, \rho_d(t \mid c_d)^{\lambda_d}, \tag{6}$$

where $\lambda_d \geq 0$ controls the strength of the prefix correction. The joint surrogate distribution is thus

$$\tilde{p}^\star(P) := \prod_{d=1}^{k} p_d^\star(t_d \mid c_d), \tag{7}$$

which is prefix-aligned (Definition B.1) whenever $\lambda_d > 0$ and ρ_d depends nontrivially on c_d. Eq. 7 thereby provides a candidate surrogate that addresses the structural mismatch of $\tilde{p}_{\text{diff}}$. This form can also be interpreted as the solution to a KL-regularized objective that balances closeness to q_d with alignment to ρ_d (see Appendix C). We thus define the token-level score as $s_{d,t}(c_d) = \log q_d(t) + \lambda_d \log \rho_d(t \mid c_d)$, and the path score

$$S(P) = \sum_{d=1}^{k} s_{d,t_d}(c_d), \qquad c_d = (t_1, \ldots, t_{d-1}). \tag{8}$$

Eq. 8 decomposes additively over depth, supporting incremental priority-based tree expansion.

Target-generated n-gram. Ideally $\rho_d = p_T(t \mid c_d)$, the quantity the correction aims to recover (Sec. 3), but querying it per node needs costly target forwards. We therefore instantiate ρ_d with an n-gram model built entirely on a corpus $\mathcal{D}_T$ of text generated by the target model itself:

$$\rho_d(t \mid c_d) = \frac{C(\bar{c}_d, t)}{C(\bar{c}_d)}, \tag{9}$$

where $\bar{c}_d$ denotes the last $n-1$ tokens before position d (including committed context), $C(\bar{c}_d, t)$ is the number of times the n-gram $(\bar{c}_d, t)$ occurs in $\mathcal{D}_T$, and $C(\bar{c}_d)$ is the number of occurrences of the prefix $\bar{c}_d$. Because the counts come from the target's own outputs, Eq. 9 approximates p_T over short contexts. The n-gram model can therefore be regarded as a lightweight proxy that makes a target-conditioned prefix signal affordable. The full construction is detailed in Appendix D.

4.2 Priority-Based Tree Construction

From the surrogate objective to priority-based expansion. By Sec. 2, maximizing Eq. 3 amounts to selecting the B prefixes u with the largest $\log \tilde{p}^\star(u) = S(u)$, where S is the path score in Eq. 8. Exhaustive enumeration is infeasible, since the search space grows exponentially with depth and vocabulary size. Because S decomposes additively over depth, high-scoring partial prefixes tend to remain competitive after expansion. We therefore adopt greedy priority-based expansion: maintain a frontier of expandable nodes and repeatedly expand the highest-scoring one.

We identify each tree node v with the unique path from the root to v, denoted $P(v) = (t_1, \ldots, t_{d(v)})$, where $d(v) := |P(v)|$ is the depth of v. The cumulative path score is $S(v) := S(P(v)) = \sum_{i=1}^{d(v)} s_{i,t_i}(c_i)$ with $c_i = (t_1, \ldots, t_{i-1})$. The root has $d(\text{root}) = 0, S(\text{root}) = 0$. At each iteration, the algorithm:

1. Selects a frontier node $v^\star$ according to a search policy π;
2. Expands $v^\star$ by adding its top-b children, ranked by the token score $s_{d(v^\star)+1,t}(c_{d(v^\star)+1})$ at depth $d(v^\star)+1$;
3. Updates the frontier with the newly added children, removing $v^\star$ from the expandable set.

The iteration terminates when $|\mathcal{T}| = B$. Because S is additive over depth, each child's cumulative score is computed incrementally from its parent's, avoiding redundant computation.

We consider two widely used instantiations of priority-based expansion under the path score S: beam search with global retention and best-first search (BFS) (pseudocode is provided in Appendix E). Both prioritize high-S partial paths but differ in how the expansion budget is allocated across depth: BFS maintains a global priority over the entire frontier, while beam search restricts expansion to the top-W in-beam nodes per depth and retains out-of-beam nodes for the final top-B selection.

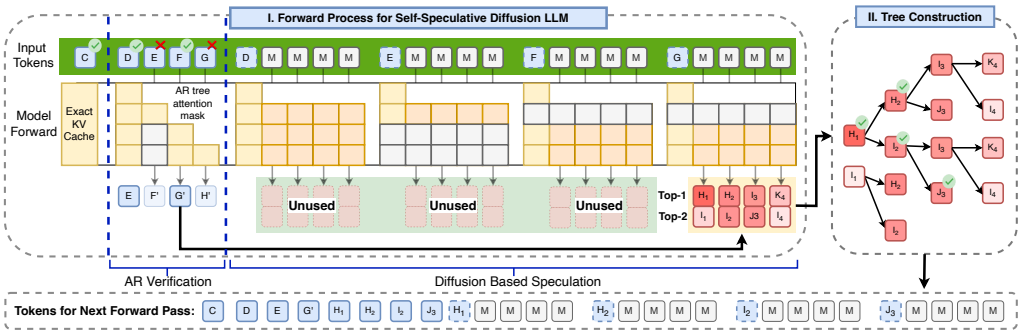


Figure 3: Application of PRESTO to hybrid dLLMs under quadratic self-speculative decoding with a tree budget of 4 drafting tokens. The detailed process is provided in Appendix F.

BFS vs. Beam Search. Table 1 reports acceptance length under the two policies across budget sizes. At small to moderate budgets ($B \leq 512$), best-first and beam search yield similar acceptance length, as beam search's per-depth budget is sufficient to cover most high-priority expansions that best-first search would select under global prioritization. At $B = 1024$, best-first outperforms beam search by a small margin: with a larger budget, the preferred expansion pattern can become less uniform across depth, which a global priority handles more flexibly than a fixed per-depth beam. In practice, verification shifts from memory-bound to compute-bound once B exceeds a hardware-dependent threshold, after which forward-pass cost grows linearly with B and erodes the speedup. We therefore operate at $B \leq 512$, which sits within the memory-bound regime where the two policies are comparable, and adopt beam search as the default search strategy.

Table 1: Acceptance length comparison.

	Qwen3-4B-DFlash		Qwen3-8B-DFlash	
Budget	BFS	Beam	BFS	Beam
$B = 128$	9.21	9.27	9.25	9.28
$B = 256$	9.55	9.54	9.60	9.62
$B = 512$	9.81	9.79	9.90	9.88
$B = 1024$	9.98	9.85	10.10	9.93

4.3 Extending Tree Construction to Hybrid dLLMs

PRESTO can be naturally extended to quadratic self-speculative (SS) decoding in hybrid dLLMs (Liu et al., 2025; Fu et al., 2026a), as illustrated in Figure 3. Quadratic SS parallelizes autoregressive verification and diffusion drafting within a single forward pass: previously drafted tokens are verified autoregressively while new candidates are simultaneously generated at trailing mask positions via diffusion decoding. PRESTO preserves this verify-and-draft pipeline without extra model forwards: in each pass, the committed prefix reuses the exact KV cache to AR-verify the previous draft tree, while the trailing mask slots attached to speculative branches are simultaneously denoised for next-round drafting. After verification, only the mask slots on the longest accepted path remain valid. Those on rejected branches are discarded, since their predictions were conditioned on prefixes no longer in the committed context. PRESTO then extracts top-k diffusion candidates from the valid slots, organizes them into a new draft tree via prefix-aligned scoring and priority-based tree expansion, and flattens the tree into draft tokens verified in the next pass through AR tree attention.

5 Empirical Validation

5.1 Experiment Setup

Models and Tasks. We evaluate PRESTO on: (i) SD with a separate diffusion drafter and AR target: dFlash with Qwen3-4B/8B and Qwen3-Coder-30B-A3B; and (ii) hybrid dLLM: Nemotron-Labs-Diffusion-8B (NLD-8B) at both linear SS and quadratic SS. Evaluation spans: **math** (GSM8K Cobbe et al. (2021), MATH Lightman et al. (2023), AIME24 Art of Problem Solving (2024), AIME25 MAA), **code** (HumanEval Chen et al. (2021), MBPP Austin et al. (2021), LiveCodeBench Jain et al. (2024)), and **chat** (MT-Bench Zheng et al. (2023), Alpaca Taori et al. (2023)). We report average acceptance length (τ) and throughput (token/s) speedup over the AR baseline.

Implementations. For dFlash and NLD under quadratic self-speculation, we use a PyTorch implementation with FlexAttention (Dong et al., 2024) for verification with tree-structured attention

Table 2: Decoding speedup over baseline and average acceptance length (τ) on Qwen3 models with thinking mode disabled and a maximum of 2048 generated tokens.

Model	Method	MATH								CODE								CHAT				*Avg.*	
		GSM8K		MATH-500		AIME24		AIME25		HumanEval		MBPP		LCB		SWE-Bench		MT-Bench		Alpaca		*Avg.*	
Temperature = 0		Sp.	τ	Sp.	τ	Sp.	τ	Sp.	τ	Sp.	τ	Sp.	τ	Sp.	τ	Sp.	τ	Sp.	τ	Sp.	τ	Sp.	τ
4B	dFlash	5.1×	6.5	6.5×	7.9	6.5×	7.4	6.2×	7.4	5.6×	6.7	5.0×	6.1	5.9×	6.9	3.2×	3.6	2.9×	4.4	2.4×	3.1	4.9×	6.0
	PRESTO	7.9×	9.4	9.2×	10.9	8.7×	10.5	8.8×	10.3	8.1×	9.9	7.9×	9.3	8.4×	10.0	5.3×	5.9	4.9×	6.7	3.9×	5.0	7.3×	8.8
	× **Gain**	**1.55**×		**1.42**×		**1.34**×		**1.42**×		**1.45**×		**1.58**×		**1.42**×		**1.66**×		**1.69**×		**1.62**×		**1.48**×	
8B	dFlash	5.0×	6.6	6.3×	7.9	6.2×	7.5	6.0×	7.0	5.4×	6.5	4.8×	6.0	5.7×	7.1	3.3×	3.6	2.8×	4.3	2.4×	3.1	4.8×	6.0
	PRESTO	8.0×	9.6	9.4×	11.1	8.5×	10.7	8.7×	10.2	8.5×	9.9	7.7×	9.4	8.5×	10.3	5.0×	6.0	4.8×	6.6	3.9×	5.0	7.3×	8.9
	× **Gain**	**1.60**×		**1.49**×		**1.37**×		**1.45**×		**1.57**×		**1.60**×		**1.49**×		**1.52**×		**1.71**×		**1.62**×		**1.52**×	
30B	dFlash	3.1×	5.2	4.1×	5.6	2.8×	5.3	2.7×	5.1	5.6×	8.0	5.6×	7.2	3.6×	6.2	3.2×	3.6	2.2×	3.5	1.8×	2.2	3.5×	5.2
	PRESTO	4.1×	7.9	5.4×	8.0	3.6×	8.0	3.6×	7.9	7.0×	10.9	7.5×	9.9	4.7×	8.5	4.6×	5.6	3.0×	5.2	2.5×	3.4	4.6×	7.5
	× **Gain**	**1.32**×		**1.32**×		**1.29**×		**1.33**×		**1.25**×		**1.34**×		**1.31**×		**1.44**×		**1.36**×		**1.39**×		**1.33**×	
Temperature = 1		Sp.	τ	Sp.	τ	Sp.	τ	Sp.	τ	Sp.	τ	Sp.	τ	Sp.	τ	Sp.	τ	Sp.	τ	Sp.	τ	Sp.	τ
4B	dFlash	4.7×	6.0	5.2×	6.6	3.8×	5.0	3.9×	4.9	4.8×	6.0	4.4×	5.6	5.0×	6.6	2.5×	3.1	2.7×	4.1	2.2×	3.0	3.9×	5.1
	PRESTO	7.2×	9.1	7.6×	9.8	6.0×	7.7	6.1×	7.7	7.6×	9.3	7.0×	9.0	6.7×	9.4	4.2×	5.0	4.3×	6.2	3.7×	4.7	6.0×	7.8
	× **Gain**	**1.53**×		**1.46**×		**1.58**×		**1.56**×		**1.58**×		**1.59**×		**1.34**×		**1.68**×		**1.59**×		**1.68**×		**1.54**×	
8B	dFlash	4.8×	6.0	5.0×	6.6	3.9×	5.1	3.8×	5.0	4.4×	5.4	4.1×	5.2	5.2×	6.8	2.3×	2.8	2.6×	3.8	2.1×	2.9	3.8×	5.0
	PRESTO	7.4×	9.0	7.4×	9.6	6.3×	7.8	6.5×	7.7	7.2×	8.6	6.9×	8.6	7.0×	9.8	4.0×	4.7	4.2×	5.9	3.6×	4.7	6.0×	7.6
	× **Gain**	**1.54**×		**1.48**×		**1.62**×		**1.71**×		**1.64**×		**1.68**×		**1.35**×		**1.74**×		**1.62**×		**1.71**×		**1.58**×	
30B	dFlash	4.1×	5.1	4.2×	5.3	3.3×	4.3	3.3×	4.2	5.8×	7.7	5.6×	7.1	3.8×	5.6	2.7×	3.2	2.1×	3.2	1.8×	2.1	3.7×	4.8
	PRESTO	4.5×	7.8	5.5×	7.7	3.7×	6.7	3.4×	6.5	6.8×	10.2	7.9×	9.6	4.3×	8.0	4.4×	4.9	3.0×	5.0	2.6×	3.3	4.6×	7.0
	× **Gain**	**1.10**×		**1.31**×		**1.12**×		**1.03**×		**1.17**×		**1.41**×		**1.13**×		**1.63**×		**1.43**×		**1.44**×		**1.26**×	

mask (Miao et al., 2024; Cai et al., 2024; Li et al., 2024). For NLD under linear self-speculation, we integrate PRESTO into the official SGLang implementation with FlashInfer attention (Ye et al., 2025b). For the n-gram model, we set n=3. Unless otherwise specified, we set $\lambda_d = 0.2$, batch size to 1, and use beam search with global retention, using beam width 10 and top-k expansion ($k = 10$) per position (See Appendix G for more details). We evaluate PRESTO under both greedy ($T = 0$) and probabilistic ($T = 1$) sampling. For dFlash, we sweep tree budgets $\mathcal{T} \in \{128, 256, 512\}$ and report the best-performing configuration in terms of end-to-end throughput. For NLD, we use a fixed tree budget $\mathcal{T}$ of 32 same as its speculative block size. All experiments run on NVIDIA B200 GPUs.

5.2 Experiment Results

PRESTO on dFlash. As illustrated in Table 2, PRESTO consistently achieves substantial improvements in τ across all models and tasks, which directly translates into higher end-to-end throughput speedup. For example, on Qwen3-4B with $T = 0$, τ increases from 7.4 to 10.3 on AIME25, while on code benchmarks it improves from 6.9 to 10.0 on LCB, leading to speedup gains across all settings. The largest improvements are observed on math and code benchmarks, where τ typically increases by $+2$ to $+3$ tokens. These trends remain stable across model scales and decoding regimes. For Qwen3-8B and under probabilistic sampling, PRESTO continues to deliver comparable relative gains.

Table 3: Decoding speedup over baseline, average acceptance length (τ), and absolute throughput on NLD-8B at linear self speculation decoding mode with a maximum of 2048 generated tokens.

Model	Method	MATH								CODE								CHAT				*Avg.*		
		GSM8K		MATH-500		AIME24		AIME25		HumanEval		MBPP		LCB		SWE-Bench		MT-Bench		Alpaca		*Avg.*		
Temperature = 0		Sp.	τ	Sp.	τ	Sp.	τ	Sp.	τ	Sp.	τ	Sp.	τ	Sp.	τ	Sp.	τ	Sp.	τ	Sp.	τ	Sp.	τ	Thr.
8B	vanilla	5.7×	10.8	6.4×	12.2	5.7×	11.0	5.5×	10.5	5.0×	9.3	4.4×	8.2	4.7×	9.0	3.7×	6.8	2.8×	5.2	2.5×	4.6	4.6×	8.8	642
	PRESTO	6.0×	11.9	6.7×	13.2	6.1×	12.2	5.8×	11.5	5.2×	10.3	4.6×	9.1	4.9×	10.0	4.0×	7.9	3.2×	6.5	2.9×	5.8	4.9×	9.9	686
	× **Gain**	**1.05**×		**1.04**×		**1.07**×		**1.06**×		**1.04**×		**1.04**×		**1.04**×		**1.08**×		**1.14**×		**1.16**×		**1.06**×		
Temperature = 1		Sp.	τ	Sp.	τ	Sp.	τ	Sp.	τ	Sp.	τ	Sp.	τ	Sp.	τ	Sp.	τ	Sp.	τ	Sp.	τ	Sp.	τ	Thr.
8B	vanilla	2.9×	5.6	2.9×	5.7	2.5×	5.0	2.6×	4.9	2.3×	4.6	2.1×	4.2	2.0×	4.0	1.4×	2.9	1.4×	2.7	1.2×	2.3	2.1×	4.2	290
	PRESTO	3.1×	6.1	3.3×	7.1	2.9×	5.7	2.9×	6.4	2.7×	5.5	2.6×	5.0	2.3×	4.8	1.8×	3.6	1.8×	3.5	1.5×	3.2	2.5×	5.1	339
	× **Gain**	**1.07**×		**1.14**×		**1.16**×		**1.12**×		**1.17**×		**1.24**×		**1.15**×		**1.29**×		**1.29**×		**1.25**×		**1.17**×		

PRESTO on NLD. As shown in Table 3, PRESTO consistently improves both throughput and average acceptance length for NLD under linear self-speculative decoding on SGLang. Under greedy decoding, PRESTO increases average throughput from $4.6\times$ to $4.9\times$ while improving average accep-

Table 4: Decoding speedup over baseline, average acceptance length (τ), and absolute throughput on NLD-8B at quadratic self speculation decoding mode with a maximum of 2048 generated tokens.

Model	Method	MATH								CODE								CHAT				*Avg.*		
		GSM8K		MATH-500		AIME24		AIME25		HumanEval		MBPP		LCB		SWE-Bench		MT-Bench		Alpaca		*Avg.*		
Temperature = 0		Sp.	τ	Sp.	τ	Sp.	τ	Sp.	τ	Sp.	τ	Sp.	τ	Sp.	τ	Sp.	τ	Sp.	τ	Sp.	τ	Sp.	τ	Thr.
8B	vanilla	2.6×	4.7	2.7×	6.8	3.2×	5.3	3.3×	5.7	3.2×	4.8	3.1×	4.5	2.4×	4.7	1.8×	3.5	2.1×	3.0	1.8×	2.7	2.6×	4.7	94
	PRESTO	5.3×	6.5	4.4×	7.5	3.5×	6.1	3.7×	6.4	3.7×	5.8	3.7×	5.4	3.1×	5.6	2.7×	4.7	2.8×	4.2	2.3×	3.6	3.5×	5.6	127
	× Gain	**2.04×**		**1.63×**		**1.09×**		**1.12×**		**1.16×**		**1.19×**		**1.29×**		**1.50×**		**1.33×**		**1.28×**		**1.35×**		
Temperature = 1		Sp.	τ	Sp.	τ	Sp.	τ	Sp.	τ	Sp.	τ	Sp.	τ	Sp.	τ	Sp.	τ	Sp.	τ	Sp.	τ	Sp.	τ	Thr.
8B	vanilla	1.4×	3.0	1.8×	3.4	1.5×	2.6	1.7×	3.0	1.7×	2.6	1.6×	2.4	1.2×	2.3	1.0×	1.8	1.2×	1.7	1.0×	1.5	1.4×	2.4	51
	PRESTO	4.1×	5.7	4.0×	6.6	2.9×	5.1	3.3×	5.7	3.2×	4.9	3.0×	4.4	2.8×	4.8	2.2×	3.7	2.3×	3.5	2.0×	3.0	3.0×	4.7	109
	× Gain	**2.93×**		**2.22×**		**1.93×**		**1.94×**		**1.88×**		**1.88×**		**2.33×**		**2.20×**		**1.92×**		**2.00×**		**2.14×**		

tance length from 8.8 to 9.9, with consistent gains observed on all benchmarks. Under probabilistic sampling, the benefits become more pronounced: average end-to-end throughput improves from $2.1\times$ to $2.5\times$, while average acceptance length increases from 4.2 to 5.1. These results demonstrate that the benefits of PRESTO persist even within a production-grade, highly optimized inference engine.

PRESTO on NLD at quadratic self speculation mode. As Table 4 shows, across all benchmarks and decoding settings, PRESTO consistently outperforms baseline in both end-to-end throughput and average acceptance length. Under greedy decoding, PRESTO improves average speedup from $2.6\times$ to $3.5\times$ on NLD-8B. The largest gains are observed on math benchmarks, where PRESTO achieves up to $2.04\times$ on GSM8k for NLD-8B. Under probabilistic sampling, PRESTO raises average end-to-end speedup from $1.4\times$ to $3.0\times$ and nearly doubles acceptance length from 2.4 to 4.7. These positive results indicate the significant benefits of applying PRESTO to quadratic self-speculation.

6 Discussions and Ablations

Lossless Tree Drafting. PRESTO is lossless by construction. It inherits verification unchanged from the dFlash and NLD. At $T = 0/1$, a drafted token is committed only if it exactly matches the target's prediction. That prediction is computed independently of the draft tree, so the tree drafting cannot alter the output distribution. More detailed discussion and empirical results are in Appendix H.

Effectiveness of prefix-conditioned signal. We first ablate the contribution of the prefix-conditioned compatibility term ρ_d by comparing our full scoring function with only using the diffusion marginal confidence q_d. As shown in Figure 4 top, incorporating ρ_d consistently improves the average acceptance length across tasks, demonstrating that effectiveness of prefix-conditioned compatibility.

Sensitivity of λ_d. Figure 4 middle shows that moderate λ_d consistently achieves the best acceptance length across different datasets and tree token budgets, with $\lambda_d \approx 0.2$ yielding the most stable gains. Small λ_d underutilizes the prefix-conditioned signal, while overly large values overly suppress the diffusion confidence q_d, leading to substantial performance degradation.

Entropy Adaptive λ_d vs. Fixed λ_d. We further explore an entropy-adaptive λ_d that adjusts the balance between q_d and ρ_d based on the diffusion drafter's uncertainty: confident predictions rely more on q_d, while uncertain ones up-weight ρ_d (details in Appendix I). As shown in Figure 4 top, it yields only marginal gains, suggesting a simple fixed λ_d already captures most of the benefit.

Overhead Analysis. As shown in Figure 4 bottom left, target-model verification dominates the overall decoding cost across all model scales, accounting for 81%–97% of the total latency. In contrast, tree-related operations contribute only 0.7%–4% of the per-step latency.

Ablation on the n-gram model. We first vary the n-gram order on Qwen3-4B with the dFlash drafter. Acceptance length saturates at $n = 3$: raising it to $n = 5$ adds at most 0.05 tokens but grows the trie from 4.5 GB to 22 GB, so we adopt $n = 3$. We further test a richer prefix signal by replacing the n-gram with Qwen3-0.6B, which changes acceptance length by at most 0.02 tokens while requiring extra forward passes. The lightweight n-gram thus suffices (details in Appendix J).

More analysis and ablations can be found in Appendix K.

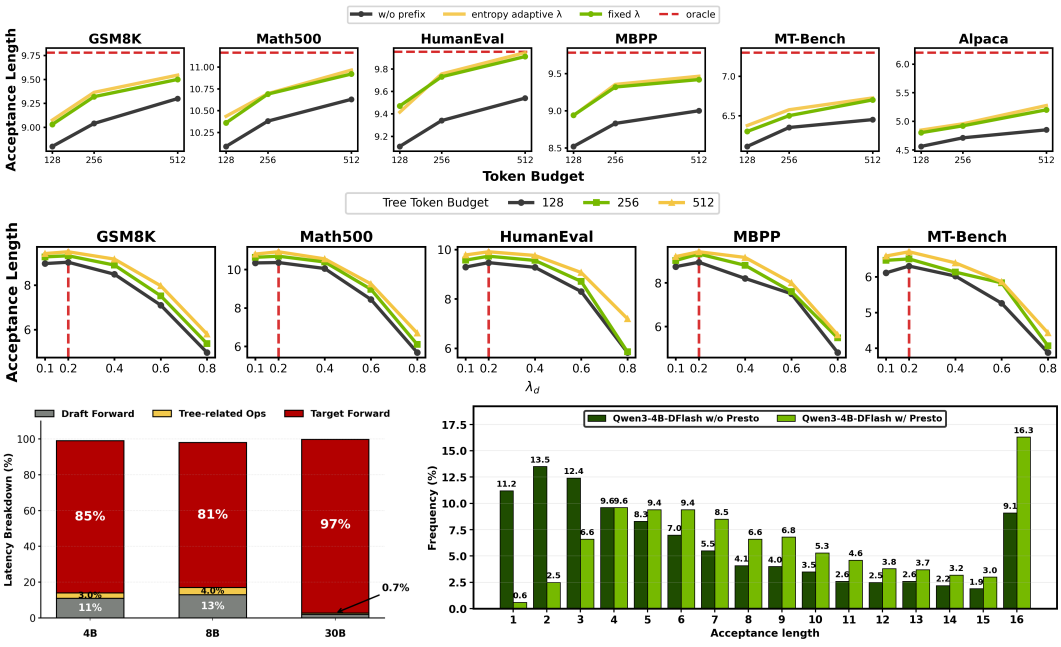


Figure 4: Qwen3-4B-DFlash, block size 16. **Top**: Adding ρ_d consistently improves acceptance length across datasets and tree budgets. **Middle**: Sensitivity to λ_d; moderate values work best, with $\lambda_d \approx 0.2$ most stable. **Bottom left**: Per-step latency breakdown across model scales; target verification dominates, and tree overhead is negligible. **Bottom right**: Acceptance-length distribution on GSM8K; PRESTO shifts mass toward longer prefixes and full-block acceptance.

7 Related Work

Speculative decoding accelerates AR LLM inference by drafting tokens with a fast drafter and verifying them in parallel with the target model (Leviathan et al., 2023; Chen et al., 2023). Follow-ups replace the external drafter with extra prediction heads (Cai et al., 2024) or feature-level drafting (Li et al., 2025c; 2024; 2025b), and tree drafting (Miao et al., 2024; Cai et al., 2024) verifies multiple continuations in one pass, evolving from static to adaptive trees that maximize expected acceptance length (Chen et al., 2025; Li et al., 2024). However, these drafters remain autoregressive, so drafting latency grows with the number of drafted tokens. dLLMs (Nie et al., 2025; Ye et al., 2025a; Bie et al., 2026; Cheng et al., 2025) generate in parallel but generally trail AR models in quality. Diffusion-based speculative decoding combines parallel drafting with lossless AR verification: TiDAR (Liu et al., 2025) jointly trains diffusion and AR objectives for self-speculation; dFlash (Chen et al., 2026) drafts a block in one forward pass with a lightweight block-diffusion drafter conditioned on target KV features; DiffuSpec (Li et al., 2025a) and SpecDiff-2 (Sandler et al., 2025) use large dLLM drafters, whose cost often offsets the speedup in real-world serving. Concurrently, DDTree (Ringel & Romano, 2026) builds trees by maximizing expected accepted tokens under the drafter's marginals with best-first expansion. However, DDTree's objective is prefix-blind, and targets only the separate-drafter setting. In contrast, PRESTO ranks paths with a unified prefix-aligned score and generalizes to self-speculative dLLMs, outperforming DDTree (Appendix K). More related work is in Appendix L.

8 Conclusion

In this work, we propose PRESTO, a principled framework for tree-based speculative decoding with diffusion language models. We identify a fundamental mismatch between diffusion draft scoring and prefix-based autoregressive verification: while diffusion probabilities provide strong marginal plausibility signals, they are inherently prefix-blind and therefore insufficient for reliable path ranking in tree-based drafting. To address this issue, we introduce a prefix-aligned scoring mechanism together with a priority-based tree search strategy, enabling effective exploration of high-quality candidate paths during diffusion drafting.

AI USE STATEMENT

In this work, we used generative AI tools to help formulate mathematical claims and to assist in the writing of proofs. We have not used generative AI tools to generate synthetic data sets, develop theoretical models or conceptual frameworks, provide critical ingredients for proving mathematical claims, propose or refine hypotheses, design or provide feedback on research methodology or experiments, implement methods, assist with translation, clean or reformat datasets, support qualitative and thematic data analysis, or interpret results. Additionally, we used generative AI tools to draft parts of the paper and to edit the paper to improve readability.

We have reviewed all AI-assisted work. AI tools were used to help state mathematical claims precisely and to write out proofs; the key ideas underlying the proofs were developed by the authors. Every AI-assisted statement and proof was checked line by line for correctness by two authors, and we verified that the assumptions and conclusions of each claim are accurately stated. All AI-drafted or AI-edited text was reviewed and revised by the authors to ensure technical accuracy and to avoid introducing unsupported claims. We take responsibility for the final content of this work, including text, claims, or artifacts produced with the aid of generative AI.

REFERENCES

Art of Problem Solving. Aime problems and solutions. https://artofproblemsolving.com/wiki/index.php/AIME, 2024. Accessed: 2025-04-20.

Jacob Austin, Augustus Odena, Maxwell Nye, Maarten Bosma, Henryk Michalewski, David Dohan, Ellen Jiang, Carrie Cai, Michael Terry, Quoc Le, and Charles Sutton. Program synthesis with large language models, 2021. URL https://arxiv.org/abs/2108.07732.

Tiwei Bie, Maosong Cao, Xiang Cao, Bingsen Chen, Fuyuan Chen, Kun Chen, Lun Du, Daozhuo Feng, Haibo Feng, Mingliang Gong, Zhuocheng Gong, Yanmei Gu, Jian Guan, Kaiyuan Guan, Hongliang He, Zenan Huang, Juyong Jiang, Zhonghui Jiang, Zhenzhong Lan, Chengxi Li, Jianguo Li, Zehuan Li, Huabin Liu, Lin Liu, Guoshan Lu, Yuan Lu, Yuxin Ma, Xingyu Mou, Zhenxuan Pan, Kaida Qiu, Yuji Ren, Jianfeng Tan, Yiding Tian, Zian Wang, Lanning Wei, Tao Wu, Yipeng Xing, Wentao Ye, Liangyu Zha, Tianze Zhang, Xiaolu Zhang, Junbo Zhao, Da Zheng, Hao Zhong, Wanli Zhong, Jun Zhou, Junlin Zhou, Liwang Zhu, Muzhi Zhu, and Yihong Zhuang. Llada2.1: Speeding up text diffusion via token editing, 2026. URL https://arxiv.org/abs/2602.08676.

Thorsten Brants, Ashok C. Popat, Peng Xu, Franz J. Och, and Jeffrey Dean. Large language models in machine translation. In Jason Eisner (ed.), *Proceedings of the 2007 Joint Conference on Empirical Methods in Natural Language Processing and Computational Natural Language Learning (EMNLP-CoNLL)*, pp. 858–867, Prague, Czech Republic, June 2007. Association for Computational Linguistics. URL https://aclanthology.org/D07-1090/.

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, 2024. URL https://arxiv.org/abs/2401.10774.

Charlie Chen, Sebastian Borgeaud, Geoffrey Irving, Jean-Baptiste Lespiau, Laurent Sifre, and John Jumper. Accelerating large language model decoding with speculative sampling, 2023. URL https://arxiv.org/abs/2302.01318.

Jian Chen, Yesheng Liang, and Zhijian Liu. Dflash: Block diffusion for flash speculative decoding, 2026. URL https://arxiv.org/abs/2602.06036.

Mark Chen, Jerry Tworek, Heewoo Jun, Qiming Yuan, Henrique Ponde de Oliveira Pinto, Jared Kaplan, Harri Edwards, Yuri Burda, Nicholas Joseph, Greg Brockman, Alex Ray, Raul Puri, Gretchen Krueger, Michael Petrov, Heidy Khlaaf, Girish Sastry, Pamela Mishkin, Brooke Chan, Scott Gray, Nick Ryder, Mikhail Pavlov, Alethea Power, Lukasz Kaiser, Mohammad Bavarian, Clemens Winter, Philippe Tillet, Felipe Petroski Such, Dave Cummings, Matthias Plappert, Fotios Chantzis, Elizabeth Barnes, Ariel Herbert-Voss, William Hebgen Guss, Alex Nichol, Alex Paino, Nikolas Tezak, Jie Tang, Igor Babuschkin, Suchir Balaji, Shantanu Jain, William Saunders,

Christopher Hesse, Andrew N. Carr, Jan Leike, Josh Achiam, Vedant Misra, Evan Morikawa, Alec Radford, Matthew Knight, Miles Brundage, Mira Murati, Katie Mayer, Peter Welinder, Bob McGrew, Dario Amodei, Sam McCandlish, Ilya Sutskever, and Wojciech Zaremba. Evaluating large language models trained on code, 2021. URL https://arxiv.org/abs/2107.03374.

Zhuoming Chen, Avner May, Ruslan Svirschevski, Yuhsun Huang, Max Ryabinin, Zhihao Jia, and Beidi Chen. Sequoia: Scalable, robust, and hardware-aware speculative decoding, 2025. URL https://arxiv.org/abs/2402.12374.

Shuang Cheng, Yihan Bian, Dawei Liu, Linfeng Zhang, Qian Yao, Zhongbo Tian, Wenhai Wang, Qipeng Guo, Kai Chen, Biqing Qi, and Bowen Zhou. Sdar: A synergistic diffusion-autoregression paradigm for scalable sequence generation, 2025. URL https://arxiv.org/abs/2510.06303.

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, 2021. URL https://arxiv.org/abs/2110.14168.

Juechu Dong, Boyuan Feng, Driss Guessous, Yanbo Liang, and Horace He. Flex attention: A programming model for generating optimized attention kernels, 2024. URL https://arxiv.org/abs/2412.05496.

Yonggan Fu, Lexington Whalen, Abhinav Garg, Chengyue Wu, Maksim Khadkevich, Nicolai Oswald, Enze Xie, Daniel Egert, Sharath Turuvekere Sreenivas, Shizhe Diao, Chenhan Yu, Ye Yu, Weijia Chen, Sajad Norouzi, Jingyu Liu, Shiyi Lan, Ligeng Zhu, Jin Wang, Jindong Jiang, Morteza Mardani, Mehran Maghoumi, Song Han, Ante Jukic, Nima Tajbakhsh, Jan Kautz, and Pavlo Molchanov. Nemotron-labs-diffusion: A tri-mode language model unifying autoregressive, diffusion, and self-speculation decoding. May 2026a. URL https://d1qx31qr3h6wln.cloudfront.net/publications/Nemotron_Diffusion_Tech_Report.pdf?VersionId=1tm4XZATEzGV7cs51XAf.xmWupU20vYW. Technical report.

Yonggan Fu, Lexington Whalen, Zhifan Ye, Xin Dong, Shizhe Diao, Jingyu Liu, Chengyue Wu, Hao Zhang, Enze Xie, Song Han, Maksim Khadkevich, Jan Kautz, Yingyan Celine Lin, and Pavlo Molchanov. Efficient-dlm: From autoregressive to diffusion language models, and beyond in speed, 2026b. URL https://arxiv.org/abs/2512.14067.

Raja Gond, Aditya K Kamath, Ramachandran Ramjee, and Ashish Panwar. Llm-42: Enabling determinism in llm inference with verified speculation, 2026. URL https://arxiv.org/abs/2601.17768.

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, 2024. URL https://arxiv.org/abs/2403.07974.

Yaniv Leviathan, Matan Kalman, and Yossi Matias. Fast inference from transformers via speculative decoding, 2023. URL https://arxiv.org/abs/2211.17192.

Guanghao Li, Zhihui Fu, Min Fang, Qibin Zhao, Ming Tang, Chun Yuan, and Jun Wang. Diffuspec: Unlocking diffusion language models for speculative decoding, 2025a. URL https://arxiv.org/abs/2510.02358.

Yuhui Li, Fangyun Wei, Chao Zhang, and Hongyang Zhang. Eagle-2: Faster inference of language models with dynamic draft trees, 2024. URL https://arxiv.org/abs/2406.16858.

Yuhui Li, Fangyun Wei, Chao Zhang, and Hongyang Zhang. Eagle-3: Scaling up inference acceleration of large language models via training-time test, 2025b. URL https://arxiv.org/abs/2503.01840.

Yuhui Li, Fangyun Wei, Chao Zhang, and Hongyang Zhang. Eagle: Speculative sampling requires rethinking feature uncertainty, 2025c. URL https://arxiv.org/abs/2401.15077.

Hunter Lightman, Vineet Kosaraju, Yura Burda, Harri Edwards, Bowen Baker, Teddy Lee, Jan Leike, John Schulman, Ilya Sutskever, and Karl Cobbe. Let's verify step by step, 2023. URL https://arxiv.org/abs/2305.20050.

Fuliang Liu, Xue Li, Ketai Zhao, Yinxi Gao, Ziyan Zhou, Zhonghui Zhang, Zhibin Wang, Wanchun Dou, Sheng Zhong, and Chen Tian. Dart: Diffusion-inspired speculative decoding for fast llm inference, 2026. URL https://arxiv.org/abs/2601.19278.

Jingyu Liu, Xin Dong, Zhifan Ye, Rishabh Mehta, Yonggan Fu, Vartika Singh, Jan Kautz, Ce Zhang, and Pavlo Molchanov. Tidar: Think in diffusion, talk in autoregression, 2025. URL https://arxiv.org/abs/2511.08923.

MAA. American invitational mathematics examination (aime). https://maa.org/math-competitions/aime. Mathematics Competition Series; n.d.a.

Xupeng Miao, Gabriele Oliaro, Zhihao Zhang, Xinhao Cheng, Zeyu Wang, Zhengxin Zhang, Rae Ying Yee Wong, Alan Zhu, Lijie Yang, Xiaoxiang Shi, Chunan Shi, Zhuoming Chen, Daiyaan Arfeen, Reyna Abhyankar, and Zhihao Jia. Specinfer: Accelerating large language model serving with tree-based speculative inference and verification. In *Proceedings of the 29th ACM International Conference on Architectural Support for Programming Languages and Operating Systems, Volume 3*, ASPLOS '24, pp. 932–949. ACM, April 2024. doi: 10.1145/3620666.3651335. URL http://dx.doi.org/10.1145/3620666.3651335.

Shen Nie, Fengqi Zhu, Zebin You, Xiaolu Zhang, Jingyang Ou, Jun Hu, Jun Zhou, Yankai Lin, Ji-Rong Wen, and Chongxuan Li. Large language diffusion models, 2025. URL https://arxiv.org/abs/2502.09992.

Team Olmo, :, Allyson Ettinger, Amanda Bertsch, Bailey Kuehl, David Graham, David Heineman, Dirk Groeneveld, Faeze Brahman, Finbarr Timbers, Hamish Ivison, Jacob Morrison, Jake Poznanski, Kyle Lo, Luca Soldaini, Matt Jordan, Mayee Chen, Michael Noukhovitch, Nathan Lambert, Pete Walsh, Pradeep Dasigi, Robert Berry, Saumya Malik, Saurabh Shah, Scott Geng, Shane Arora, Shashank Gupta, Taira Anderson, Teng Xiao, Tyler Murray, Tyler Romero, Victoria Graf, Akari Asai, Akshita Bhagia, Alexander Wettig, Alisa Liu, Aman Rangapur, Chloe Anastasiades, Costa Huang, Dustin Schwenk, Harsh Trivedi, Ian Magnusson, Jaron Lochner, Jiacheng Liu, Lester James V. Miranda, Maarten Sap, Malia Morgan, Michael Schmitz, Michal Guerquin, Michael Wilson, Regan Huff, Ronan Le Bras, Rui Xin, Rulin Shao, Sam Skjonsberg, Shannon Zejiang Shen, Shuyue Stella Li, Tucker Wilde, Valentina Pyatkin, Will Merrill, Yapei Chang, Yuling Gu, Zhiyuan Zeng, Ashish Sabharwal, Luke Zettlemoyer, Pang Wei Koh, Ali Farhadi, Noah A. Smith, and Hannaneh Hajishirzi. Olmo 3, 2026. URL https://arxiv.org/abs/2512.13961.

Liran Ringel and Yaniv Romano. Accelerating speculative decoding with block diffusion draft trees, 2026. URL https://arxiv.org/abs/2604.12989.

Mohammad Samragh, Arnav Kundu, David Harrison, Kumari Nishu, Devang Naik, Minsik Cho, and Mehrdad Farajtabar. Your llm knows the future: Uncovering its multi-token prediction potential, 2025. URL https://arxiv.org/abs/2507.11851.

Jameson Sandler, Jacob K. Christopher, Thomas Hartvigsen, and Ferdinando Fioretto. Specdiff-2: Scaling diffusion drafter alignment for faster speculative decoding, 2025. URL https://arxiv.org/abs/2511.00606.

Ziteng Sun, Ananda Theertha Suresh, Jae Hun Ro, Ahmad Beirami, Himanshu Jain, and Felix Yu. Spectr: Fast speculative decoding via optimal transport, 2024. URL https://arxiv.org/abs/2310.15141.

Ruslan Svirschevski, Avner May, Zhuoming Chen, Beidi Chen, Zhihao Jia, and Max Ryabinin. Specexec: Massively parallel speculative decoding for interactive llm inference on consumer devices, 2024. URL https://arxiv.org/abs/2406.02532.

Rohan Taori, Ishaan Gulrajani, Tianyi Zhang, Yann Dubois, Xuechen Li, Carlos Guestrin, Percy Liang, and Tatsunori B. Hashimoto. Stanford alpaca: An instruction-following llama model. https://github.com/tatsu-lab/stanford_alpaca, 2023.

Jikai Wang, Yi Su, Juntao Li, Qingrong Xia, Zi Ye, Xinyu Duan, Zhefeng Wang, and Min Zhang. Opt-tree: Speculative decoding with adaptive draft tree structure, 2025. URL https://arxiv.org/abs/2406.17276.

Chengyue Wu, Hao Zhang, Shuchen Xue, Shizhe Diao, Yonggan Fu, Zhijian Liu, Pavlo Molchanov, Ping Luo, Song Han, and Enze Xie. Fast-dllm v2: Efficient block-diffusion llm, 2025a. URL https://arxiv.org/abs/2509.26328.

Chengyue Wu, Hao Zhang, Shuchen Xue, Zhijian Liu, Shizhe Diao, Ligeng Zhu, Ping Luo, Song Han, and Enze Xie. Fast-dllm: Training-free acceleration of diffusion llm by enabling kv cache and parallel decoding, 2025b. URL https://arxiv.org/abs/2505.22618.

Jiacheng Ye, Zhihui Xie, Lin Zheng, Jiahui Gao, Zirui Wu, Xin Jiang, Zhenguo Li, and Lingpeng Kong. Dream 7b: Diffusion large language models, 2025a. URL https://arxiv.org/abs/2508.15487.

Zihao Ye, Lequn Chen, Ruihang Lai, Wuwei Lin, Yineng Zhang, Stephanie Wang, Tianqi Chen, Baris Kasikci, Vinod Grover, Arvind Krishnamurthy, and Luis Ceze. Flashinfer: Efficient and customizable attention engine for llm inference serving, 2025b. URL https://arxiv.org/abs/2501.01005.

Yifan Yu, Yuqing Jian, Junxiong Wang, Zhongzhu Zhou, Donglin Zhuang, Xinyu Fang, Sri Yanamandra, Xiaoxia Wu, Qingyang Wu, Shuaiwen Leon Song, Tri Dao, Ben Athiwaratkun, James Zou, Fan Lai, and Chenfeng Xu. Introspective diffusion language models, 2026. URL https://arxiv.org/abs/2604.11035.

Lianmin Zheng, Wei-Lin Chiang, Ying Sheng, Siyuan Zhuang, Zhanghao Wu, Yonghao Zhuang, Zi Lin, Zhuohan Li, Dacheng Li, Eric P. Xing, Hao Zhang, Joseph E. Gonzalez, and Ion Stoica. Judging llm-as-a-judge with mt-bench and chatbot arena, 2023. URL https://arxiv.org/abs/2306.05685.

A Prefix-Mass Decomposition of the Tree Objective

Let $\mathcal{V}$ denote the token vocabulary and let $\mathcal{T}$ be a prefix-closed tree whose nodes are finite token sequences. Define

$$\mathcal{T}^+ := \mathcal{T} \setminus \{\emptyset\}$$

as the set of non-root nodes. For each node $u = (u_1, \ldots, u_{|u|}) \in \mathcal{T}^+$, let $|u|$ denote its depth.

For a sampled path

$$P = (x_1, \ldots, x_k) \sim \tilde{p},$$

define the length-d prefix

$$P_{\le d} := (x_1, \ldots, x_d).$$

The accepted-prefix length under tree $\mathcal{T}$ is

$$\alpha_{\mathcal{T}}(P) := \max\{d : P_{\le d} \in \mathcal{T}\}.$$

Since $\mathcal{T}$ is prefix-closed, the event $P_{\le d} \in \mathcal{T}$ implies $P_{\le j} \in \mathcal{T}$ for all $j \le d$. Hence,

$$\alpha_{\mathcal{T}}(P) = \sum_{d=1}^{k} \mathbf{1}\{P_{\le d} \in \mathcal{T}\}.$$

Equivalently,

$$\alpha_{\mathcal{T}}(P) = \sum_{u \in \mathcal{T}^+} \mathbf{1}\{P_{\le |u|} = u\}.$$

Taking expectation with respect to $P \sim \tilde{p}$ and applying linearity of expectation,

$$\mathbb{E}_{P\sim\tilde{p}}[\alpha_{\mathcal{T}}(P)] = \sum_{u \in \mathcal{T}^+} \Pr_{P\sim\tilde{p}}(P_{\le |u|} = u).$$

Define the surrogate prefix mass

$$\tilde{p}(u) := \Pr_{P\sim\tilde{p}}(P_{\le |u|} = u).$$

Then

$$\mathbb{E}_{P\sim\tilde{p}}[\alpha_{\mathcal{T}}(P)] = \sum_{u \in \mathcal{T}^+} \tilde{p}(u).$$

If the root node is excluded by convention, then $\mathcal{T}^+ = \mathcal{T}$, yielding Eq. equation 3.

B Formal Definition of Prefix-Aligned Surrogates

Definition B.1 (Prefix-Aligned Surrogate)**.** *A surrogate distribution $\tilde{p}$ over $\mathcal{V}^k$ is* prefix-aligned *if there exists some position $i \ge 2$ and prefixes $u_{<i} \ne u'_{<i}$ such that*

$$\tilde{p}(u_i \mid u_{<i}) \ne \tilde{p}(u_i \mid u'_{<i}). \tag{10}$$

This is a necessary condition for faithfully ranking paths under prefix-based verification. A prefix-blind surrogate $\tilde{p}(u) = \prod_i \tilde{p}_i(u_i)$ assigns each token the same score regardless of the realized prefix, and thus cannot capture the prefix-conditioned structure in Eq. equation 1. The AR surrogate $\tilde{p}_{\text{AR}}$ satisfies this definition, whereas the diffusion marginal surrogate $\tilde{p}_{\text{diff}}$ (Eq. equation 5) violates it.

C KL-Regularized Derivation of the Prefix-Aligned Surrogate

We consider the variational objective

$$\min_{p \in \Delta(\mathcal{V})} \ \mathrm{KL}(p \,\|\, q_d(\cdot)) - \lambda_d \mathbb{E}_{t\sim p}\big[\log \rho_d(t \mid c_d)\big], \tag{11}$$

where q_d denotes the diffusion drafter marginal, $\rho_d(\cdot \mid c_d)$ is a prefix-conditioned compatibility signal, and $\Delta(\mathcal{V})$ denotes the probability simplex over the vocabulary. We assume $q_d(t) > 0$ and $\rho_d(t \mid c_d) > 0$ on the candidate support.

Expanding the KL divergence,

$$\mathrm{KL}(p\|q_d) = \sum_t p(t) \log \frac{p(t)}{q_d(t)},$$

the objective becomes

$$\mathcal{L}(p) = \sum_t p(t) \log \frac{p(t)}{q_d(t)} - \lambda_d \sum_t p(t) \log \rho_d(t \mid c_d).$$

Including the normalization constraint $\sum_t p(t) = 1$ with Lagrange multiplier μ, we obtain

$$\mathcal{J}(p) = \sum_t p(t) \log \frac{p(t)}{q_d(t)} - \lambda_d \sum_t p(t) \log \rho_d(t \mid c_d) + \mu \left(\sum_t p(t) - 1 \right).$$

Taking derivatives with respect to $p(t)$,

$$\frac{\partial \mathcal{J}}{\partial p(t)} = \log p(t) - \log q_d(t) + 1 - \lambda_d \log \rho_d(t \mid c_d) + \mu.$$

Setting the derivative to zero yields

$$\log p_d^\star(t) = \log q_d(t) + \lambda_d \log \rho_d(t \mid c_d) + C,$$

where C absorbs constants independent of t. Exponentiating both sides,

$$p_d^\star(t) \propto q_d(t)\, \rho_d(t \mid c_d)^{\lambda_d}.$$

Equivalently,

$$p_d^\star(t \mid c_d) = \frac{q_d(t) \rho_d(t \mid c_d)^{\lambda_d}}{Z_d(c_d)}, \qquad Z_d(c_d) = \sum_{t' \in \mathcal{V}} q_d(t') \rho_d(t' \mid c_d)^{\lambda_d}.$$

Thus, the optimal surrogate takes a product-of-experts form, combining the calibrated marginal signal q_d with a prefix-conditioned correction ρ_d.

D Construction of the n-gram model in PRESTO

The n-gram model used in PRESTO is a simple count-based statistical model, built by a single counting pass over a corpus and stored as a lightweight frequency trie. We disclose its construction details below.

Corpus preparation. The corpus is generated by the target model itself, seeded from a subset of the open-source Dolma3-mix dataset (Olmo et al., 2026) covering coding, math, scientific, and Wikipedia topics, with no overlap with the evaluation benchmarks. For each seed document we take its first 1,500 tokens and let the target model generate continuations at temperature 0.7. This yields 1B tokens in total, from which we build the n-gram model; a single n-gram model is shared across all tasks. Because the corpus is generated by the target model itself, so the counts it stores can be regarded as an offline estimate of the target's own local conditional distribution. The co-occurrence it measures is that of the target's output, not of language in general. At inference it is read in constant time from a frequency trie, at negligible cost. The n-gram model in PRESTO can therefore be regarded as a lightweight proxy that makes a target-conditioned prefix signal affordable.

Building process. With the generated corpus, each n-gram value is obtained by a single counting pass: sliding an order-n window over every sequence, we increment an integer frequency at the corresponding node. The value used at inference time is the maximum-likelihood relative frequency

$P(w \mid c) = \mathrm{count}(c, w)/\,\mathrm{count}(c)$, where c denotes the $(n-1)$-token context, w the candidate next token, $\mathrm{count}(c, w)$ the number of times the n-gram (c, w) appears in the corpus, and $\mathrm{count}(c)$ the total count of the context c. The build is extremely cheap, which is CPU-only and fully parallelized. The 3-gram model takes only about 5 minutes to construct.

Fallback strategy (the $p = 0$ case). We use the classical longest-match back-off strategy (Brants et al., 2007): we query the longest available context ($n-1$ tokens) and back off to shorter contexts when it is absent. If no context of any order contains the token, the prefix-aligned score reduces to the diffusion marginal alone, so scoring remains well-defined at every node and drafting is never interrupted.

E Priority-Based Tree Drafting

Algorithm 1 Beam Search with Global Retention

procedure BeamSearch(s, B, b, W, D)
  $\mathcal{P} \leftarrow \{\text{root}\}$, $\mathcal{T} \leftarrow \{\text{root}\}$
  for $d = 1$ to D **while** $|\mathcal{T}| < B$ **do**
    $\mathcal{P}_{\text{new}} \leftarrow \emptyset$
    for each $v \in \mathcal{P}$ **do**
      $\mathcal{C} \leftarrow \textsc{Expand}(v, b, s)$
      $\mathcal{P}_{\text{new}} \leftarrow \mathcal{P}_{\text{new}} \cup \mathcal{C}$
      $\mathcal{T} \leftarrow \mathcal{T} \cup \mathcal{C}$
    end for
    $\mathcal{P} \leftarrow \textsc{TopW}(\mathcal{P}_{\text{new}}, W)$
  end for
  return $\textsc{TopB}(\mathcal{T}, B)$
end procedure

Algorithm 2 Best-First Search

procedure BestFirst(s, B, b)
  $\mathcal{P} \leftarrow \{\text{root}\}$, $\mathcal{T} \leftarrow \{\text{root}\}$
  $S(\text{root}) \leftarrow 0$
  while $|\mathcal{T}| < B$ **do**
    $v^\star \leftarrow \arg\max_{u \in \mathcal{P}} S(u)$
    $\mathcal{P} \leftarrow \mathcal{P} \setminus \{v^\star\}$
    $\mathcal{C} \leftarrow \textsc{Expand}(v^\star, b, s)$
    Insert $\mathcal{C}$ into priority queue $\mathcal{P}$
    $\mathcal{T} \leftarrow \mathcal{T} \cup \mathcal{C}$
  end while
  ▷ tree construction complete
  return $\textsc{TopB}(\mathcal{T}, B)$
end procedure

F PRESTO on Hybrid dLLMs at Quadratic Self Speculation Decoding

We provide a detailed walk-through of how PRESTO is instantiated on Hybrid dLLMs at quadratic self speculation decoding mode, corresponding to Figure 3. Unlike the standard speculative decoding setup where a small drafter proposes tokens for a larger verifier, self-speculative dLLMs use *the same dLLM* as both drafter and verifier: a single forward pass simultaneously verifies previously drafted tokens and speculates new ones at trailing mask positions. PRESTO turns this single-model pipeline into a tree-based drafter without any additional model.

At the start of round r, the input sequence consists of three contiguous regions:

- a **committed prefix** of accepted tokens from previous rounds (green tokens with ✓ in Figure 3, e.g. `C, D`);
- a **verification region** containing the draft tokens proposed in round $r-1$ (e.g. `E, F, G`), some of which will be accepted and some rejected;
- a **speculation region** of mask tokens (`M`) appended to the right, at which the dLLM will produce new draft candidates.

A single forward pass over this composite input yields outputs that serve both roles below.

(I) AR verification of the previous draft tree. On the verification region, PRESTO uses the exact KV cache of the committed prefix together with an *AR tree attention mask* that restricts each draft token to attend only to its ancestors in the previous round's draft tree. This lets a single forward verify all paths of the tree in parallel: each path is checked left-to-right against the model's predictions, and the longest accepted prefix across paths is committed (`D`→`F` in Figure 3, where `E` on the sibling path and `G` on the same path are rejected). The position immediately after the last accepted token is filled with the model's own prediction (`F'`, `G'`, `H'` in Figure 3), and the accepted tokens become part of the committed prefix for round $r+1$.

(II) Diffusion-based speculation along the accepted path. The speculation region is structured as a set of mask slots attached to each draft token from the previous round, mirroring the previous

round's draft tree (Figure 3 shows two candidate paths under verification, D→E and D→F→G). The dLLM denoises all of these mask positions in parallel within the same forward, but only the mask slots hanging off the *accepted* path are used for the next round's drafting. In Figure 3, F is accepted and G is rejected, so PRESTO reads top-b candidates from the mask slots following F ($b = 2$ in the figure, e.g. $\{H_1, H_2\}$ at the first such position, $\{I_1, I_2\}$ at the next, and so on). The mask slots attached to rejected branches are produced by the same forward but discarded (marked "Unused" in Figure 3), since their predictions were conditioned on a context that is no longer part of the committed prefix.

(III) Draft tree construction. The per-position top-b candidates are assembled into a draft tree following the priority-based expansion of Section 4.2. The root corresponds to the first speculation position, and at depth d each node has up to b children drawn from the top-b candidates at depth $d+1$. Each child inherits its parent's cumulative path score S and adds the local token score $s_{d+1,t}(c)$, so scores are computed incrementally with no redundant work. The frontier is expanded under beam search with global retention until the tree contains B nodes, after which the top-B paths under S are kept (Algorithm 1). Crucially, the candidate logits at every depth come from the *same* forward pass that performed verification, so tree construction itself adds no additional dLLM forwards.

(IV) Flattening for the next forward pass. The draft tree is then linearised into the input sequence for round $r + 1$ (bottom of Figure 3). Each draft token is followed by a block of mask tokens that will host the next round's speculation, while the tree's parent–child structure is encoded in the AR tree attention mask used during verification. The result is a single input on which the next forward pass again performs verification and speculation jointly, completing the cycle.

G Choices of Beam width and top-K

We ablate the two search hyper-parameters, the top-k expansion size k and the beam width b, around the deployed setting $k = 10$, $b = 10$. All runs use the same fixed budget of 128 nodes with $\lambda = 0.2$. Results are averaged over the three tasks with a full sweep (GSM8K, HumanEval, and MT-Bench) and are reported in Table 5.

Table 5: Sensitivity to top-k and beam width b at a fixed 128-node budget ($\lambda = 0.2$), averaged over GSM8K, HumanEval, and MT-Bench. The deployed setting is shown in bold.

Model	$k5/b10$	$k10/b5$	$\mathbf{k10/b10}$	$k10/b20$	$k20/b10$
Qwen3-4B	8.10	8.13	**8.15**	8.15	8.16
Qwen3-8B	8.12	8.13	**8.22**	8.22	8.23

Compared with the deployed $k = 10$, $b = 10$ setting, raising either k or b to 20 changes τ by at most 0.01 on both models. Halving either one to 5 lowers τ by 0.02–0.10. The drop is largest for Qwen3-8B, which loses 0.10 at $k = 5$ and 0.09 at $b = 5$. Performance has therefore already saturated at $k = 10$, $b = 10$, and the cost per step stays low, so we use this as the default configuration in all main experiments.

H Losslessness of PRESTO

PRESTO is lossless by construction. PRESTO modifies only draft-tree construction and inherits its verification unchanged from the dFlash and self-speculative Nemotron-Labs-Diffusion (NLD) pipelines. At both $T = 0$ and $T = 1$, a drafted token is committed only if it exactly matches the target's prediction. That prediction is computed independently of the draft tree, so every committed token is the target's own, and the tree cannot alter the output distribution.

Empirical confirmation. We also verify this empirically by evaluating downstream quality at $T = 0$ and $T = 1$. We use Qwen3-4B/8B with dFlash, and NLD-8B for the self-speculative pipeline. The metrics are exact-match accuracy on math QA and pass@1 on code. Results are in Tables 6 and 7.

Table 6: Downstream quality of AR decoding, dFlash, and PRESTO on Qwen3-4B/8B (accuracy for math, pass@1 for code).

		$T = 0$			$T = 1$		
Dataset	Model	AR	dFlash	PRESTO	AR	dFlash	PRESTO
GSM8K	4B	88.0	88.5	89.0	89.0	89.0	89.5
	8B	91.0	90.0	90.5	90.5	89.5	89.5
MATH500	4B	66.5	67.0	66.5	65.5	65.5	66.5
	8B	69.5	70.0	69.5	68.5	66.0	69.0
HumanEval	4B	86.0	85.8	85.8	86.0	85.5	85.6
	8B	87.0	87.4	87.2	87.8	87.2	87.8
LiveCodeBench	4B	38.0	37.0	38.0	39.0	39.0	39.0
	8B	40.0	40.5	41.0	40.0	39.0	41.0

Table 7: Downstream quality of self-speculative NLD-8B without and with PRESTO.

	$T = 0$		$T = 1$	
Dataset	Self-spec.	+PRESTO	Self-spec.	+PRESTO
GSM8K	94.0	94.0	92.0	94.0
MATH500	82.0	83.0	80.0	81.0
HumanEval	87.0	87.0	87.0	86.0
LiveCodeBench	29.0	30.0	30.0	32.0

Small per-benchmark deviations appear in all settings. They are expected for lossless speculative decoding in practice and have been documented before (Gond et al., 2026); they are not specific to PRESTO. GPU kernels use different reduction orders for multi-token verification than for single-token decoding. This can flip the arg-max at near-ties and change output tokens even under fixed sampling parameters, and at $T = 1$ sampling randomness adds further variance. These effects come from the serving path, not from the drafting algorithm. Consistent with this, PRESTO stays within 1.0 point of AR on every dFlash setting, while the lossless linear-drafting dFlash baseline deviates by up to 2.5 points (MATH500, 8B, $T = 1$). On NLD, PRESTO never trails the baseline by more than 1.0 point. Overall, PRESTO empirically preserves the target model's output quality in every reported setting.

I Entropy-adaptive λ_d

In addition to a fixed interpolation coefficient λ_d, we explored an entropy-adaptive variant that scales the prefix correction by the uncertainty of the diffusion drafter. Let

$$H_d = -\sum_{w \in \mathcal{V}} q_d(w) \log q_d(w)$$

denote the entropy of the diffusion marginal at depth d. We define

$$\lambda_d(H_d) = \beta \cdot \frac{H_d}{H_d + C},$$

where $\beta > 0$ controls the maximum correction strength and $C > 0$ is a stabilising constant, and substitute this into the token-level score from Section 4:

$$s_{d,t}(c_d) = \log q_d(t) + \lambda_d(H_d)\ \log \rho_d(t \mid c_d).$$

Under this schedule, low-entropy (high-confidence) positions assign small weight to the prefix correction and rely primarily on the calibrated marginal q_d, while high-entropy positions up-weight the prefix-conditioned signal ρ_d. The fixed-λ_d variant used in our main experiments is recovered by replacing $\lambda_d(H_d)$ with a constant.

J Ablation on the n-gram Prefix Model

Sweep over the n-gram order n. We ablate the order n of the n-gram model in PRESTO on Qwen3-4B with the dFlash drafter, across five benchmarks and three tree-node budgets. Table 8 reports the

average acceptance length together with the memory footprint of the n-gram trie. Acceptance length rises sharply from $n = 1$ to $n = 3$ and then saturates under all three node budgets: going from $n = 3$ to $n = 5$ adds at most 0.05, while the trie grows by $2.7\times$ at $n = 4$ and $4.9\times$ at $n = 5$. A 2-token context ($n = 3$) therefore already captures most of the local signal that matters for acceptance, which makes $n = 3$ the best trade-off between acceptance and memory.

Table 8: Average acceptance length τ vs. n-gram order n on Qwen3-4B with the dFlash drafter, averaged over five benchmarks. The last column is the memory footprint of the n-gram trie.

n	128 nodes	256 nodes	512 nodes	Memory
1	8.57	8.85	8.98	4.7 MB
2	8.84	9.11	9.21	987 MB
3	**8.95**	**9.23**	**9.37**	**4.5 GB**
4	8.97	9.25	9.38	12 GB
5	9.00	9.26	9.41	22 GB

Table 9: Acceptance length τ with the n-gram model vs. Qwen3-0.6B as the prefix signal ρ_d (target: Qwen3-4B, 128 tree nodes).

Dataset	Target	n-gram τ	Qwen3-0.6B τ	$\Delta\tau$
GSM8K	4B	9.22	9.23	-0.01
HumanEval	4B	9.46	9.45	$+0.01$
MT-Bench	4B	6.39	6.41	-0.02

$\Delta\tau = \tau_{n\text{-gram}} - \tau_{\text{Qwen3-0.6B}}$.

Alternative prefix signal ρ_d: a small pretrained LM. We replace the n-gram model with Qwen3-0.6B and use its next-token distribution as ρ_d. This is a deliberately strong alternative. Instead of estimating the prefix conditional from surface counts, a pretrained LM scores candidates with learned representations and generalizes to contexts that never appear in the trie, so it is a richer estimator of the same quantity. Even so, the two signals perform almost identically in practice (Table 9). With Qwen3-4B as the target and a 128-node budget, acceptance length differs by at most 0.02 tokens ($\leq 0.4\%$) across three tasks. A model-based ρ_d therefore brings no significant benefit over the count-based n-gram and needs additional forward passes. The n-gram in PRESTO is thus an efficient way to provide the prefix signal: it serves as an offline estimate of the target model's own local conditional distribution, at negligible runtime cost.

K More Analysis and Discussion

K.1 Relationship between PRESTO and DDTree

PRESTO is a formulation rather than a single algorithm. It combines priority-based tree expansion under a general search policy π (Sec. 4.2) with prefix-aligned scoring, controlled by λ_d (Sec. 4.1). This lets us isolate what prefix-aligned scoring contributes by comparing against the closest prefix-blind instantiation.

DDTree as a special case of PRESTO. DDTree (Ringel & Romano, 2026) builds its draft tree by maximizing the expected number of accepted tokens under the drafter's marginal distribution, and it expands the tree with a best-first heap. In PRESTO's terms, this is the corner case $\lambda_d = 0$ with π set to best-first search (Algorithm 2). At $\lambda_d = 0$, the tree score reduces to the drafter's marginal and loses the prefix-aligned signal. This is exactly the prefix-blind objective that Sec. 2.2 identifies as structurally misaligned with prefix-based autoregressive verification. Comparing PRESTO with DDTree therefore serves as an ablation of the prefix-aligned scoring term under a matched expansion strategy. DDTree also covers only the standard setting with a separate diffusion drafter. PRESTO additionally extends prefix-aligned tree drafting to self-speculative diffusion LLMs (Sec. 4.3).

We compare the two methods at node budgets of 128, 256, and 512 on Qwen3-4B and Qwen3-8B. We report acceptance length τ and end-to-end throughput speedup (PRESTO / DDTree), averaged over GSM8K, MATH500, HumanEval, and LiveCodeBench. All other settings follow the main experiments. PRESTO outperforms DDTree at every tree budget on both models (Table 10). Acceptance length improves by $+0.29$ to $+0.35$. The throughput ratio rises monotonically with the budget on both models, giving end-to-end speedups of $1.04\times$–$1.10\times$. The expansion policy is held fixed across the two methods, so these gains come from the prefix-aligned scoring term λ_d. Aligning the tree objective with prefix-based verification yields consistently longer accepted prefixes.

Table 10: PRESTO vs. DDTree (the prefix-blind corner case $\lambda_d = 0$, best-first) across tree budgets. τ is acceptance length; speedup is the end-to-end throughput ratio PRESTO / DDTree. Results are averaged over GSM8K, MATH500, HumanEval, and LiveCodeBench.

Model	Node budget	DDTree τ	PRESTO τ	$\Delta\tau$	Speedup
Qwen3-4B	128	9.28	9.57	**+0.29**	1.06×
	256	9.53	9.87	**+0.34**	1.08×
	512	9.84	10.16	**+0.32**	1.10×
Qwen3-8B	128	9.37	9.67	**+0.30**	1.04×
	256	9.70	10.05	**+0.35**	1.07×
	512	9.99	10.30	**+0.31**	1.08×

L Related Work

L.1 Speculative Decoding

Speculative decoding (Leviathan et al., 2023; Chen et al., 2023) accelerates AR models by drafting tokens with a fast proposer and verifying them in parallel with the target model, provably preserving the target distribution. This guarantee relies on the target model possessing a well-trained verify distribution. A line of follow-up work eliminates the external drafter or improves draft quality: Medusa (Cai et al., 2024) augments the base LLM with multiple prediction heads and uses tree attention for parallel verification; the EAGLE family (Li et al., 2025c; 2024; 2025b) exploits feature-level context from the frozen target model, with EAGLE-1 predicting future hidden states to boost acceptance, EAGLE-2 introducing adaptive drafting trees, and EAGLE-3 refining training objectives to scale speedups; and SpecInfer (Miao et al., 2024) proposes tree-structured verification. Multi-token prediction (MTP) (Samragh et al., 2025) trains models to predict multiple future tokens simultaneously. Despite these advances, most existing methods rely on autoregressive drafting, which remains inherently sequential and limits attainable speedups.

L.2 Tree Construction in Speculative Decoding

A complementary line of work focuses on how draft tokens are organized for parallel verification. Medusa (Cai et al., 2024) popularized tree-structured drafting by packing multiple candidate continuations into a single token tree and verifying them in one forward pass via a topology-aware attention mask. However, this tree mask is hand-crafted and static. SpecInfer (Miao et al., 2024) introduced token-tree verification with provable distribution preservation. Subsequent work moves beyond fixed shapes: Sequoia (Chen et al., 2025) formulates tree construction as a dynamic-programming problem, jointly optimizing the tree topology and a hardware-aware tree size for a given accelerator. SpecExec (Svirschevski et al., 2024) pushes draft trees to massively parallel sizes (thousands of nodes) to amortize the cost of offloaded weights on consumer hardware based on Best First Search like tree construction algorihtm. EAGLE-2 (Li et al., 2024) observes that draft acceptance is context-dependent and uses the calibrated confidence scores of the EAGLE drafter to construct a dynamic draft tree per step via beam search like tree expension. OPT-Tree (Wang et al., 2025) formalizes the objective as maximizing the expected acceptance length and searches for the adaptive tree structure that attains it under a node budget. These methods consistently show that careful tree drafting contributes a substantial fraction of end-to-end speedup. However, they all assume an autoregressive drafter that produces tokens sequentially with conditional probabilities along each branch, and extending tree

construction to diffusion-based drafters, whose draft tokens are produced in parallel remains an open problem that our work directly addresses.

L.3 Diffusion-based Speculative Decoding

Recent work uses diffusion models as drafters in speculative decoding, combining parallel drafting with lossless AR verification. TiDAR (Liu et al., 2025) jointly trains diffusion and AR objectives in a sequence-level hybrid, though its generation is not yet lossless. DiffuSpec (Li et al., 2025a) and SpecDiff-2 (Sandler et al., 2025) use large pretrained dLLMs (*e.g.*, 7B) as drafters, with inference-time search or train–test alignment. These methods focus on improving the drafter itself, and the high drafting cost can offset their long acceptance lengths in practical serving. DFlash (Chen et al., 2026) instead uses a lightweight block-diffusion drafter conditioned on target-model features and reports over $6\times$ lossless acceleration. DART (Liu et al., 2026) predicts logits in parallel over masked positions and assembles drafts via N-gram-guided tree pruning, but it relies on a hand-tuned scoring rule that must be re-tuned for new budgets, drafters, or target models. Concurrently, DDTree (Ringel & Romano, 2026) builds draft trees by maximizing expected accepted tokens under the drafter's marginals with best-first expansion. This objective is prefix-blind, which Sec. 2.2 shows is misaligned with prefix-based AR verification, and DDTree covers only the separate-drafter setting. In contrast, PRESTO formulates tree construction as priority-based expansion under a unified prefix-aligned path score. The resulting framework generalizes across tree budgets and drafter designs, extends to self-speculative dLLMs (Sec. 4.3), and complements stronger diffusion drafters.

M Limitation

PRESTO requires a tractable prefix-aligned signal ρ_d to complement the diffusion drafter's prefix-blind marginals. In this work, we employ a lightweight n-gram model, which is empirically effective, but only captures short-range lexical compatibility. Exploring richer prefix-conditioned signals while preserving low overhead remains an important future direction. We hope this motivates further efforts to inject prefix-aligned signals into diffusion marginals. We additionally explored an entropy-adaptive variant of λ_d, but observed only marginal improvements over a fixed coefficient, suggesting that more sophisticated adaptation strategies may require learned or target-aware scheduling mechanisms.