PRESTO: Prefix-Aligned Tree Drafting for Diffusion Speculative Decoding

Anonymous Authors[1]

Abstract

Diffusion Large Language Models (dLLMs) have recently emerged as a promising alternative to autoregressive (AR) LLMs, offering parallel token generation. Recent works have shown that dLLMs are particularly effective as draft models in speculative decoding (SD), where they can efficiently propose multiple candidate tokens in parallel. However, existing diffusion-based drafting methods primarily rely on linear drafting, despite diffusion models simultaneously producing multiple candidate tokens at all positions that can be combined into many possible paths. A natural solution is to extend tree-based drafting to diffusion models, enabling the exploration of diverse candidate paths. However, we find that applying naive tree-based drafting is suboptimal due to a fundamental mismatch between diffusion draft confidence and prefix-based AR verification: diffusion marginals are inherently prefix-blind, leading to unreliable path ranking. To this end, we propose PRESTO, a principled framework for tree-based diffusion drafting via *prefix-aligned scoring* and *priority-based tree search*. A key principle behind our framework is that candidate ranking should align with the prefix-based nature of AR verification. Guided by this, we design a prefix-aligned surrogate score to prioritize high-quality candidate paths during tree expansion for diffusion drafter. PRESTO is a general tree drafting framework applicable to both individual diffusion drafter SD and self-speculative dLLMs. Extensive experiments show that PRESTO achieves up to an average of **1.5**$\times$ speedup on the state-of-the-art individual diffusion drafter SD and an average of **1.95**$\times$ on self-speculative diffusion LLMs across diverse benchmarks, thereby unlocking the full potential of tree-based speculative decoding for diffusion drafting.

[1]Anonymous Institution, Anonymous City, Anonymous Region, Anonymous Country. Correspondence to: Anonymous Author <anon.email@domain.com>.

Preliminary work. Under review by the International Conference on Machine Learning (ICML). Do not distribute.

1. Introduction

Diffusion Large Language Models (dLLMs) have recently emerged as a promising alternative to autoregressive (AR) LLMs, enabling parallel token generation (Nie et al., 2025; Ye et al., 2025; Wu et al., 2025a; Bie et al., 2026; Cheng et al., 2025; Fu et al., 2026b). However, high-quality generation typically requires iterative denoising and carefully designed unmasking strategies, which partially offset their efficiency gains (Wu et al., 2025a;b). This has motivated a growing line of work that instead uses diffusion models as draft models in speculative decoding (SD) (Liu et al., 2025; Li et al., 2025a; Chen et al., 2026; Yu et al., 2026), where their parallel generation naturally fits the proposal stage. Existing approaches include diffusion-based drafting 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, existing methods primarily rely on *linear drafting*, where tokens are generated along a single trajectory and verified sequentially. This is inherently suboptimal for diffusion: due to its non-autoregressive nature, multiple plausible candidates coexist, and the ground-truth continuation may lie outside the top-1 trajectory. Empirically, even strong diffusion drafters in dFlash yield only modest acceptance length under single-path drafting (Figure 2(a)).

A natural direction is to jointly explore multiple candidate paths. In AR settings, tree-based drafting substantially increases acceptance length by expanding candidates in parallel (Miao et al., 2024; Cai et al., 2024; Li et al., 2024; 2025b; Svirschevski et al., 2024; Chen et al., 2025). Since the target model is typically not compute-bound during verification (Liu et al., 2025), spending more drafting compute on a richer candidate set yields more accepted tokens per step. This naturally raises the question:

Can we leverage tree-based drafting in diffusion drafters to improve acceptance length and push parallel decoding throughput even further?

While promising, we find that naively applying tree-based drafting to diffusion models is suboptimal. Diffusion drafters produce position-wise distributions in parallel without strict prefix dependencies, enabling flexible tree construction but yielding marginal rather than prefix-

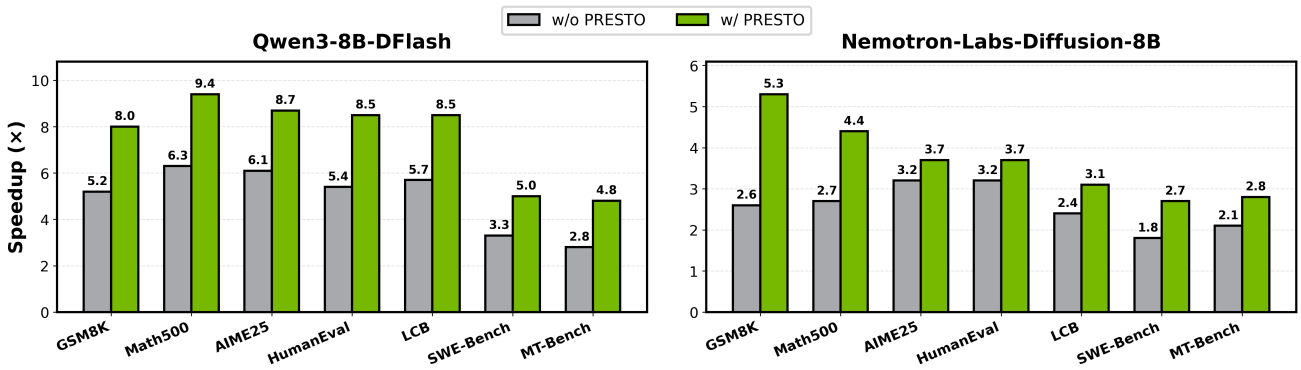

Figure 1. Speedup comparison of PRESTO on diffusion-based speculative decoding, including individual diffusion drafter SD (dFlash) and self-speculative diffusion LLM (Nemotron-Labs-Diffusion), across diverse benchmarks. Overall, PRESTO achieves consistent improvements, with up to 1.7× speedup on Qwen3-8B-DFlash and over 2.0× speedup on Nemotron-Labs-Diffusion-8B.

conditioned scores. These scores cannot capture how earlier token choices affect downstream acceptance (e.g., a token likely in isolation may become unlikely given the prefix), creating a fundamental mismatch with prefix-based AR verification process.

We propose PRESTO, a principled tree-based diffusion drafting framework via *prefix-aligned scoring* and *priority-based tree search*. Our key idea is to minimally adjust the diffusion marginal with a prefix-dependent correction, so that the score better reflects sequential verification behavior. This score then guides a priority-based expansion strategy that allocates computation to high-quality paths during tree construction. As a result, PRESTO significantly improves acceptance length and throughput on both diffusion-based drafting for AR LLMs and self-speculative dLLMs. Our contributions are:

- **We identify a fundamental mismatch between diffusion draft scoring and prefix-based AR verification.** Diffusion probabilities are inherently prefix-blind, leading to unreliable path ranking and suboptimal acceptance length in tree-based drafting.
- **We design prefix-aligned scoring to resolve this mismatch** through a principled, minimal correction that augments the diffusion marginal with prefix-dependent signals, remaining faithful to the diffusion model while compatible with prefix-based verification.
- **We propose PRESTO, a tree-based diffusion drafting framework** that combines prefix-aligned scoring with priority-based tree expansion. Across diverse benchmarks and model sizes, PRESTO achieves up to **1.5**× average speedup on dFlash and **1.95**× on Nemotron-Labs-Diffusion.
- **We are the first to extend tree-based drafting to self-speculative dLLMs.** As a drop-in replacement for linear drafting in the parallelized verify-and-draft pipeline, PRESTO consistently improves acceptance length and throughput.

2. Preliminaries

2.1. Tree-based Speculative Decoding

Speculative decoding accelerates a target LLM p_T via a lightweight draft model. To improve acceptance, the drafter constructs a *token tree* encoding multiple candidate sequences, which the target verifies in a *prefix-based* manner: starting from the root, tokens are accepted sequentially until a mismatch occurs.

Acceptance factorization. For a candidate path $P = (x_1, \ldots, x_k)$, let $a_i \in \{0, 1\}$ indicate whether the i-th token is accepted. By the chain rule,

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

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

Tree construction objective. Let $\mathcal{B}_B = \{\mathcal{T} : |\mathcal{T}| \le B\}$ and let $\alpha_{\mathcal{T}}(P)$ denote the longest prefix of P contained in $\mathcal{T}$. The ideal objective $\mathcal{T}^* \in \arg\max_{\mathcal{T} \in \mathcal{B}_B} \mathbb{E}_{P \sim p_T}[\alpha_{\mathcal{T}}(P)]$ is infeasible since evaluating p_T requires target forward passes. Using a surrogate $\tilde{p}$ over paths, a standard decomposition (Appendix A) gives

$$\mathbb{E}_{P \sim \tilde{p}}[\alpha_{\mathcal{T}}(P)] = \sum_{u \in \mathcal{T}} \tilde{p}(u), \quad (2)$$

where $\tilde{p}(u)$ is the probability that prefix u matches the sampled path. This reduces tree construction to selecting prefixes with large probability mass, with an additive scoring rule

$$\log \tilde{p}(u) = \sum_{i=1}^{|u|} \log \tilde{p}(u_i \mid u_{<i}). \quad (3)$$

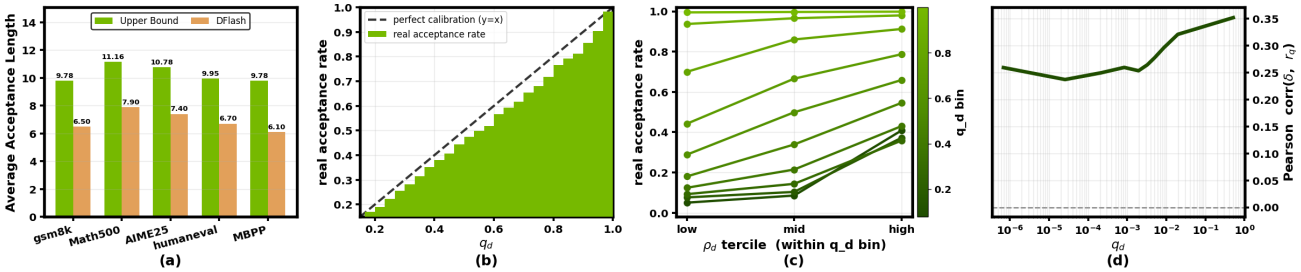

Figure 2. (a) Linear drafting leaves room for improvement via multi-path exploration. (b) The predicted confidence from q_d closely matches the empirical acceptance rate. (c) Within fixed q_d bins, acceptance rates increase from low to high ρ_d terciles, indicating that prefix-conditioned compatibility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

2.2. Prefix-Aligned Surrogate Distributions

Definition 2.1 (Prefix-Aligned Surrogate)**.** *A surrogate $\tilde{p}$ over $\mathcal{V}^k$ is* prefix-aligned *if there exists $i \geq 2$ and $u_{<i} \neq u'_{<i}$ such that*

$$\tilde{p}(u_i \mid u_{<i}) \neq \tilde{p}(u_i \mid u'_{<i}). \tag{4}$$

The nontriviality clause is essential: under an independence assumption, any joint distribution admits a trivial prefix-blind factorization $\tilde{p}(u) = \prod_i \tilde{p}_i(u_i)$, which assigns identical per-position contributions regardless of the realized prefix and therefore fails to capture the prefix-conditioned structure of Eq. (1).

2.3. Autoregressive vs. Diffusion Surrogate Distributions

AR drafting yields a prefix-aligned surrogate. 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 induced factorized surrogate is

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

Since $\tilde{p}_{\text{diff}}(x_i \mid x_{<i}) = q_i(x_i)$ regardless of $x_{<i}$, the marginal surrogate violates Definition 2.1. 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 (Section 3).

3. Observations

We use Qwen3-4B-DFlash as a representative model to analyze diffusion drafter behavior; detailed settings are in Appendix B.

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

Observation 1: Linear drafting leaves room for improvement via multi-path exploration. We compare linear drafting against an empirical upper bound from exhaustive multi-path exploration: at each position we take the top-10 candidates from the diffusion marginal, enumerate all combinations into a candidate tree, and brute-force search for the longest target-accepted prefix. Figure 2(a) shows a substantial gap between linear drafting and this upper bound across all tasks.

Observation 2: Marginal confidence is well-calibrated for overall 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 a well-calibrated estimate of global acceptance rate.

Observation 3: Marginal signals are insufficient but correctable. 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 is well-calibrated globally but insufficient for path ranking; ρ_d offers complementary, prefix-conditioned signal aligned with the error of q_d but itself noisy. Together, these suggest that *effective scoring should preserve q_d while incorporating a controlled correction from ρ_d.*

4. PRESTO

We now introduce PRESTO, a tree-based drafting framework that augments diffusion drafter with prefix-aligned information. In Section 4.1, we introduce prefix-aligned scoring that addresses the aforementioned structural mismatch. In Section 4.2, we describe tree drafting via priority-based tree search. In Section 4.3, we elaborate how to extend tree drafting to self-speculative dLLMs.

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**
      $\mathcal{C} \leftarrow \text{EXPAND}(v, b, s)$
      $\mathcal{P}_{\text{new}} \leftarrow \mathcal{P}_{\text{new}} \cup \mathcal{C}$
      $\mathcal{T} \leftarrow \mathcal{T} \cup \mathcal{C}$
    end for
    $\mathcal{P} \leftarrow \text{TOPW}(\mathcal{P}_{\text{new}}, W)$
  end for
  return $\text{TOPB}(\mathcal{T}, B)$
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**
    $v^\star \leftarrow \arg\max_{u \in \mathcal{P}} S(u)$
    $\mathcal{P} \leftarrow \mathcal{P} \setminus \{v^\star\}$
    $\mathcal{C} \leftarrow \text{EXPAND}(v^\star, b, s)$
    Insert $\mathcal{C}$ into priority queue $\mathcal{P}$
    $\mathcal{T} \leftarrow \mathcal{T} \cup \mathcal{C}$
  end while
  ▷ tree construction complete
  return $\text{TOPB}(\mathcal{T}, B)$
end procedure

4.1. Prefix-Aligned Scoring via Minimal Correction

From surrogate distribution to scoring function. Section 2 reduces tree construction to ranking prefixes by $\log \tilde{p}(u)$ (Eq. (3)). For diffusion drafting, the natural marginal surrogate $\tilde{p}_{\text{diff}}(P) = \prod_i q_i(x_i)$ is prefix-blind (Eq. (5)), creating a mismatch with prefix-based verification. We therefore seek a prefix-aligned surrogate $\tilde{p}^\star$ that preserves the calibrated drafter signal while incorporating a prefix-conditioned correction in terms of Section 3.

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 induced joint surrogate distribution is

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

which is prefix-aligned (Definition 2.1) 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). For simplicity, we use the unnormalized log-score induced by Eq. (6). Specifically, we 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.

4.2. Priority-Based Tree Construction

From the surrogate objective to priority-based expansion. Section 2 shows that maximizing the surrogate objective in Eq. (2) reduces to selecting the top-B prefixes by their log-probabilities under $\tilde{p}^\star$. Equivalently, the optimal tree consists of the B prefixes u with the largest $\log \tilde{p}^\star(u) = S(u)$, where S is the path score in Eq. (8). However, exhaustively expanding and ranking all candidate continuations is computationally infeasible due to the exponentially growing search space and large vocabulary branching factor. Instead, since the path score decomposes additively over depth, high-scoring partial prefixes are more likely to remain among the top-ranked candidates after further expansion. This naturally motivates a greedy priority-based expansion strategy: maintain a frontier of expandable nodes and iteratively expand the highest-scoring node.

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 (Algorithm 1) and best-first search (BFS) (Algorithm 2). 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. Beam search can thus be viewed as a practical approximation to BFS with a fixed-width active beam.

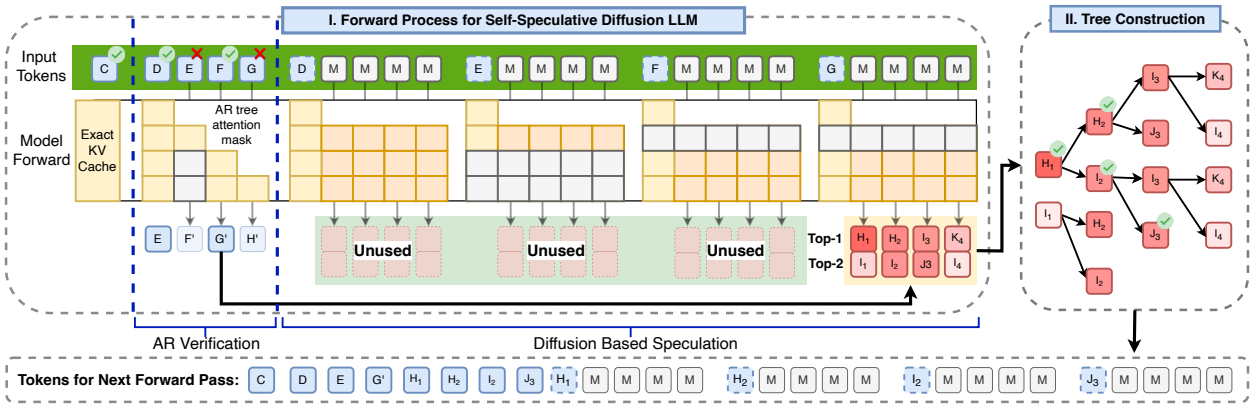

Figure 3. Application of PRESTO on Self-speculative dLLMs, PRESTO is the first to explore tree-based drafting for self speculative dLLMs. The detailed process is elaborated in Appendix D.

Table 1. Accept length comparison between BFS and Beam Search.

	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

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 suffices to cover the shallow trees that best-first would expand under global priority. At $B = 1024$, best-first outperforms beam search by a small margin: with a larger budget, the optimal 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 $B \leq 512$, where the two policies are comparable, and adopt beam search as the default.

4.3. Extending Tree Construction to Self-speculative Decoding

The above tree-construction algorithm can be extended to support self-speculative decoding in hybrid autogressive-diffusion models (Liu et al., 2025; Fu et al., 2026a), as shown in Figure 3. Similar to the literature (Liu et al., 2025), we parallelize the autoregressive-based verification and diffusion-based drafting in a single model forward process. The key difference in our approach is that we replace the linear draft with a tree draft constructed by PRESTO. As detailed in Section 5.2, this replacement has consistently led to improved average acceptance length and throughput across different datasets and models.

5. Empirical Validation

We show that PRESTO significantly improves the decoding efficiency of diffusion-based speculative decoding across both standard and self-speculative diffusion SD.

5.1. Experiment Setup

Models, Tasks, and Metrics. We evaluate PRESTO on two settings: (i) standard SD with a separate diffusion drafter and AR target: dFlash with Qwen3-4B, Qwen3-8B, and Qwen3-Coder-30B-A3B backbones; and (ii) self-speculative dLLM: Nemotron-Labs-Diffusion at 3B and 8B. Evaluation spans three task categories: **mathematical reasoning** (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 **conversation** (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 on NVIDIA B200 GPUs.

Implementation. We implement tree-structured decoding with FlexAttention and use an n-gram model (Liu et al., 2026) ($n = 3$) as the prefix-dependent signal. Unless otherwise specified, we set $\lambda_d = 0.2$, batch size 1, beam search with width 10 and per-position top-k expansion ($k = 10$), and evaluate at temperatures 0 and 1. For dFlash we sweep tree budgets $\{128, 256, 512\}$ and report the best; for Nemotron-Labs-Diffusion we use a fixed budget of 16.

5.2. Experiment Results

Main Results. As shown in Tables 2 and 3, PRESTO consistently improves average acceptance length τ and throughput across both dFlash and Nemotron-Lab-Diffusion (NLD), across all models and tasks. On Qwen3-4B-DFlash with $T = 0$, τ rises from 7.9 to 10.9 on MATH-500, 7.4 to 10.3 on AIME25, 6.9 to 10.0 on LCB, and 6.7 to 9.9 on Hu-

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×**	**1.45×**	**1.42×**	**1.38×**	**1.34×**	**1.42×**	**1.42×**	**1.39×**	**1.45×**	**1.48×**	**1.58×**	**1.52×**	**1.42×**	**1.45×**	**1.66×**	**1.64×**	**1.69×**	**1.52×**	**1.62×**	**1.61×**	**1.48×**	**1.47×**
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×**	**1.45×**	**1.49×**	**1.41×**	**1.37×**	**1.43×**	**1.45×**	**1.46×**	**1.57×**	**1.52×**	**1.60×**	**1.57×**	**1.49×**	**1.45×**	**1.52×**	**1.67×**	**1.71×**	**1.53×**	**1.62×**	**1.61×**	**1.52×**	**1.49×**
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×**	**1.52×**	**1.32×**	**1.43×**	**1.29×**	**1.51×**	**1.33×**	**1.55×**	**1.25×**	**1.36×**	**1.34×**	**1.38×**	**1.31×**	**1.37×**	**1.44×**	**1.56×**	**1.36×**	**1.49×**	**1.39×**	**1.55×**	**1.33×**	**1.45×**

Table 3. Decoding speedup over baseline and average τ on Nemotron-Labs-Diffusion models 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
3B	vanilla	4.1×	5.81	3.7×	6.68	4.0×	6.28	3.8×	6.33	2.1×	4.61	3.0×	4.36	1.6×	4.77	1.8×	3.37	2.1×	3.15	1.9×	2.65	2.8×	4.80
	PRESTO	4.8×	6.81	5.3×	7.71	4.3×	6.92	4.7×	7.36	4.0×	5.57	3.7×	5.44	3.3×	5.71	2.9×	4.03	2.6×	3.85	2.2×	3.39	3.8×	5.68
	× Gain	**1.17×**	**1.17×**	**1.43×**	**1.15×**	**1.08×**	**1.10×**	**1.24×**	**1.16×**	**1.90×**	**1.21×**	**1.23×**	**1.25×**	**2.06×**	**1.20×**	**1.61×**	**1.20×**	**1.24×**	**1.22×**	**1.16×**	**1.28×**	**1.36×**	**1.18×**
8B	vanilla	2.6×	5.92	2.7×	6.83	3.2×	5.31	3.3×	5.65	3.2×	4.84	3.1×	4.47	2.4×	4.74	1.8×	3.47	2.1×	3.03	1.8×	2.67	2.6×	4.69
	PRESTO	5.3×	6.50	4.4×	7.48	3.5×	6.05	3.7×	6.40	3.7×	5.78	3.7×	5.38	3.1×	5.56	2.7×	4.67	2.8×	4.23	2.3×	3.57	3.5×	5.56
	× Gain	**2.04×**	**1.10×**	**1.63×**	**1.10×**	**1.09×**	**1.14×**	**1.12×**	**1.13×**	**1.16×**	**1.19×**	**1.19×**	**1.20×**	**1.29×**	**1.17×**	**1.50×**	**1.35×**	**1.33×**	**1.40×**	**1.28×**	**1.34×**	**1.35×**	**1.19×**

manEval, with the largest gains ($+2$ to $+3$ tokens) on math and code; stronger drafters (Qwen3-8B) yield comparable relative gains, indicating that PRESTO pushes dFlash's acceptance length closer to its upper bound. The same trends hold on NLD: PRESTO lifts average speedup from $2.8\times$ to $3.8\times$ on NLD-3B and $2.6\times$ to $3.5\times$ on NLD-8B with consistent acceptance-length gains. Results under stochastic decoding ($T = 1$), where the gains become substantially larger due to PRESTO 's ability to preserve multiple plausible continuations under uncertainty, are reported in Appendix E.

5.3. Discussions and Ablations

Effectiveness of prefix-conditioned signal. We ablate the prefix-conditioned compatibility term ρ_d by comparing our full scoring function against using only the diffusion marginal q_d. As shown in Figure 4 (top) in Appendix F, incorporating ρ_d consistently improves average acceptance length across tasks, confirming its effectiveness.

Sensitivity to λ_d. As shown in Figure 4 (middle) in Appendix F, moderate λ_d achieves the best acceptance length across datasets and tree budgets, with $\lambda_d \approx 0.2$ yielding the most stable gains. Small values underutilize the prefix-conditioned signal, while overly large values suppress q_d and substantially degrade performance.

More detailed discussions and ablations including tree operations overhead and distribution of acceptance length are in Appendix F.

6. Related Work

Speculative decoding accelerates AR LLMs by drafting candidates and verifying them in parallel (Leviathan et al., 2023; Chen et al., 2023), with follow-ups using prediction heads (Cai et al., 2024) or feature-level drafting (Li et al., 2025c; 2024; 2025b). Tree-structured drafting further packs multiple candidates per forward (Miao et al., 2024; Cai et al., 2024), with adaptive variants improving over static shapes (Chen et al., 2025; Li et al., 2024; Svirschevski et al., 2024; Wang et al., 2025). Yet AR drafters cap speedup by drafter latency. Diffusion LLMs (Nie et al., 2025; Ye et al., 2025; Wu et al., 2025a; Bie et al., 2026; Cheng et al., 2025; Fu et al., 2026b) enable parallel generation, motivating diffusion-based SD: TiDAR (Liu et al., 2025) achieves self-speculation via joint diffusion-AR training; dFlash (Chen et al., 2026) pairs a block-diffusion drafter with target-conditioned KV injection; DiffuSpec (Li et al., 2025a) and SpecDiff-2 (Sandler et al., 2025) use large dLLMs as drafters. All remain linear, leaving tree-drafting gains unrealized. See Appendix G for details.

7. Conclusion

We propose PRESTO, a tree-based SD framework for diffusion LLMs. We identify that diffusion probabilities are inherently prefix-blind, causing unreliable path ranking under prefix-based AR verification. PRESTO addresses this via prefix-aligned scoring and priority-based tree search, enabling effective exploration of high-quality candidate paths.

Impact Statement

This work aims to improve the efficiency and accessibility of large language model inference by accelerating speculative decoding through tree-based diffusion drafting. By increasing throughput while preserving generation quality losslessly, PRESTO reduces the computational cost and latency of serving high-capacity language models, potentially broadening access to advanced AI capabilities for researchers and practitioners with limited hardware resources.

References

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

Austin, J., Odena, A., Nye, M., Bosma, M., Michalewski, H., Dohan, D., Jiang, E., Cai, C., Terry, M., Le, Q., and Sutton, C. Program synthesis with large language models, 2021. URL https://arxiv.org/abs/2108.07732.

Bie, T., Cao, M., Cao, X., Chen, B., Chen, F., Chen, K., Du, L., Feng, D., Feng, H., Gong, M., Gong, Z., Gu, Y., Guan, J., Guan, K., He, H., Huang, Z., Jiang, J., Jiang, Z., Lan, Z., Li, C., Li, J., Li, Z., Liu, H., Liu, L., Lu, G., Lu, Y., Ma, Y., Mou, X., Pan, Z., Qiu, K., Ren, Y., Tan, J., Tian, Y., Wang, Z., Wei, L., Wu, T., Xing, Y., Ye, W., Zha, L., Zhang, T., Zhang, X., Zhao, J., Zheng, D., Zhong, H., Zhong, W., Zhou, J., Zhou, J., Zhu, L., Zhu, M., and Zhuang, Y. Llada2.1: Speeding up text diffusion via token editing, 2026. URL https://arxiv.org/abs/2602.08676.

Cai, T., Li, Y., Geng, Z., Peng, H., Lee, J. D., Chen, D., and Dao, T. Medusa: Simple llm inference acceleration framework with multiple decoding heads, 2024. URL https://arxiv.org/abs/2401.10774.

Chen, C., Borgeaud, S., Irving, G., Lespiau, J.-B., Sifre, L., and Jumper, J. Accelerating large language model decoding with speculative sampling, 2023. URL https://arxiv.org/abs/2302.01318.

Chen, J., Liang, Y., and Liu, Z. Dflash: Block diffusion for flash speculative decoding, 2026. URL https://arxiv.org/abs/2602.06036.

Chen, M., Tworek, J., Jun, H., Yuan, Q., de Oliveira Pinto, H. P., Kaplan, J., Edwards, H., Burda, Y., Joseph, N., Brockman, G., Ray, A., Puri, R., Krueger, G., Petrov, M., Khlaaf, H., Sastry, G., Mishkin, P., Chan, B., Gray, S., Ryder, N., Pavlov, M., Power, A., Kaiser, L., Bavarian, M., Winter, C., Tillet, P., Such, F. P., Cummings, D., Plappert, M., Chantzis, F., Barnes, E., Herbert-Voss, A., Guss, W. H., Nichol, A., Paino, A., Tezak, N., Tang, J., Babuschkin, I., Balaji, S., Jain, S., Saunders, W., Hesse, C., Carr, A. N., Leike, J., Achiam, J., Misra, V., Morikawa, E., Radford, A., Knight, M., Brundage, M., Murati, M., Mayer, K., Welinder, P., McGrew, B., Amodei, D., McCandlish, S., Sutskever, I., and Zaremba, W. Evaluating large language models trained on code, 2021. URL https://arxiv.org/abs/2107.03374.

Chen, Z., May, A., Svirschevski, R., Huang, Y., Ryabinin, M., Jia, Z., and Chen, B. Sequoia: Scalable, robust, and hardware-aware speculative decoding, 2025. URL https://arxiv.org/abs/2402.12374.

Cheng, S., Bian, Y., Liu, D., Zhang, L., Yao, Q., Tian, Z., Wang, W., Guo, Q., Chen, K., Qi, B., and Zhou, B. Sdar: A synergistic diffusion-autoregression paradigm for scalable sequence generation, 2025. URL https://arxiv.org/abs/2510.06303.

Cobbe, K., Kosaraju, V., Bavarian, M., Chen, M., Jun, H., Kaiser, L., Plappert, M., Tworek, J., Hilton, J., Nakano, R., Hesse, C., and Schulman, J. Training verifiers to solve math word problems, 2021. URL https://arxiv.org/abs/2110.14168.

Fu, Y., Whalen, L., Garg, A., Wu, C., Khadkevich, M., Oswald, N., Xie, E., Egert, D., Sreenivas, S. T., Diao, S., Yu, C., Yu, Y., Chen, W., Norouzi, S., Lan, S., Zhu, L., Wang, J., Jiang, J., Mardani, M., Maghoumi, M., Han, S., Jukic, A., Tajbakhsh, N., Kautz, J., and Molchanov, P. Nemotron-labs-diffusion: A tri-mode language model unifying autoregressive, diffusion, and self-speculation decoding. *arXiv preprint*, May 2026a.

Fu, Y., Whalen, L., Ye, Z., Dong, X., Diao, S., Liu, J., Wu, C., Zhang, H., Xie, E., Han, S., Khadkevich, M., Kautz, J., Lin, Y. C., and Molchanov, P. Efficient-dlm: From autoregressive to diffusion language models, and beyond in speed, 2026b. URL https://arxiv.org/abs/2512.14067.

Jain, N., Han, K., Gu, A., Li, W.-D., Yan, F., Zhang, T., Wang, S., Solar-Lezama, A., Sen, K., and Stoica, I. Livecodebench: Holistic and contamination free evaluation of large language models for code, 2024. URL https://arxiv.org/abs/2403.07974.

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

Li, G., Fu, Z., Fang, M., Zhao, Q., Tang, M., Yuan, C., and Wang, J. Diffuspec: Unlocking diffusion language models for speculative decoding, 2025a. URL https://arxiv.org/abs/2510.02358.

Li, Y., Wei, F., Zhang, C., and Zhang, H. Eagle-2: Faster inference of language models with dynamic draft trees, 2024. URL https://arxiv.org/abs/2406.16858.

Li, Y., Wei, F., Zhang, C., and Zhang, H. Eagle-3: Scaling up inference acceleration of large language models via training-time test, 2025b. URL https://arxiv.org/abs/2503.01840.

Li, Y., Wei, F., Zhang, C., and Zhang, H. Eagle: Speculative sampling requires rethinking feature uncertainty, 2025c. URL https://arxiv.org/abs/2401.15077.

Lightman, H., Kosaraju, V., Burda, Y., Edwards, H., Baker, B., Lee, T., Leike, J., Schulman, J., Sutskever, I., and Cobbe, K. Let's verify step by step, 2023. URL https://arxiv.org/abs/2305.20050.

Liu, F., Li, X., Zhao, K., Gao, Y., Zhou, Z., Zhang, Z., Wang, Z., Dou, W., Zhong, S., and Tian, C. Dart: Diffusion-inspired speculative decoding for fast llm inference, 2026. URL https://arxiv.org/abs/2601.19278.

Liu, J., Dong, X., Ye, Z., Mehta, R., Fu, Y., Singh, V., Kautz, J., Zhang, C., and Molchanov, P. 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.

Miao, X., Oliaro, G., Zhang, Z., Cheng, X., Wang, Z., Zhang, Z., Wong, R. Y. Y., Zhu, A., Yang, L., Shi, X., Shi, C., Chen, Z., Arfeen, D., Abhyankar, R., and Jia, Z. 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.

Nie, S., Zhu, F., You, Z., Zhang, X., Ou, J., Hu, J., Zhou, J., Lin, Y., Wen, J.-R., and Li, C. Large language diffusion models, 2025. URL https://arxiv.org/abs/2502.09992.

Samragh, M., Kundu, A., Harrison, D., Nishu, K., Naik, D., Cho, M., and Farajtabar, M. Your llm knows the future: Uncovering its multi-token prediction potential, 2025. URL https://arxiv.org/abs/2507.11851.

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

Svirschevski, R., May, A., Chen, Z., Chen, B., Jia, Z., and Ryabinin, M. Specexec: Massively parallel speculative decoding for interactive llm inference on consumer devices, 2024. URL https://arxiv.org/abs/2406.02532.

Taori, R., Gulrajani, I., Zhang, T., Dubois, Y., Li, X., Guestrin, C., Liang, P., and Hashimoto, T. B. Stanford alpaca: An instruction-following llama model. https://github.com/tatsu-lab/stanford_alpaca, 2023.

Wang, J., Su, Y., Li, J., Xia, Q., Ye, Z., Duan, X., Wang, Z., and Zhang, M. Opt-tree: Speculative decoding with adaptive draft tree structure, 2025. URL https://arxiv.org/abs/2406.17276.

Wu, C., Zhang, H., Xue, S., Diao, S., Fu, Y., Liu, Z., Molchanov, P., Luo, P., Han, S., and Xie, E. Fast-dllm v2: Efficient block-diffusion llm, 2025a. URL https://arxiv.org/abs/2509.26328.

Wu, C., Zhang, H., Xue, S., Liu, Z., Diao, S., Zhu, L., Luo, P., Han, S., and Xie, E. Fast-dllm: Training-free acceleration of diffusion llm by enabling kv cache and parallel decoding, 2025b. URL https://arxiv.org/abs/2505.22618.

Ye, J., Xie, Z., Zheng, L., Gao, J., Wu, Z., Jiang, X., Li, Z., and Kong, L. Dream 7b: Diffusion large language models, 2025. URL https://arxiv.org/abs/2508.15487.

Yu, Y., Jian, Y., Wang, J., Zhou, Z., Zhuang, D., Fang, X., Yanamandra, S., Wu, X., Wu, Q., Song, S. L., Dao, T., Athiwaratkun, B., Zou, J., Lai, F., and Xu, C. Introspective diffusion language models, 2026. URL https://arxiv.org/abs/2604.11035.

Zheng, L., Chiang, W.-L., Sheng, Y., Zhuang, S., Wu, Z., Zhuang, Y., Lin, Z., Li, Z., Li, D., Xing, E. P., Zhang, H., Gonzalez, J. E., and Stoica, I. 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. (2).

B. Experimental Setup for Observations

In this section, we describe the experimental setup used in Section 3 to analyze the behavior of the diffusion drafter.

Model and tasks. We use Qwen3-4B-DFlash as a representative model and aggregate results across all datasets in our main evaluation (math, code, and chat).

Tree construction. We employ EAGLE-2/3 Beam Search (Li et al., 2024; 2025b) as the default tree search algorithm. For each candidate token, we record its draft probability q_d, prefix-conditioned score ρ_d, and acceptance outcome. The prefix-conditioned score ρ_d is instantiated via an 3-gram model that captures prefix-conditioned compatibility.

Empirical upper bound (Observation 1). To estimate the upper bound on acceptance length achievable by multi-path exploration, at each position we select the top-10 candidates from the diffusion marginal and enumerate all combinations into a candidate tree. We then brute-force search this tree to identify the longest prefix accepted by the target verifier. The gap between linear drafting and this bound (Figure 2(a)) quantifies the headroom available to tree-based methods.

C. KL-Regularized Derivation of the Prefix-Aligned Surrogate

We consider the variational objective

$$\min_{p \in \Delta(\mathcal{V})} \ \mathrm{KL}(p \,\|\, q_d(\cdot)) - \lambda_d \mathbb{E}_{t \sim p}\big[\log \rho_d(t \mid c_d)\big], \tag{9}$$

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. PRESTO on Self-Speculative Diffusion LLMs

We provide a detailed walk-through of how PRESTO is instantiated on self-speculative diffusion LLMs (dLLMs), 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.

E. Results under Stochastic Decoding ($T = 1$)

Table 4. Decoding speedup over baseline and average acceptance length (τ) on Qwen3 models with thinking mode disabled and a maximum of 2048 generated tokens when T=1.

Model	Method	MATH								CODE								CHAT				*Avg.*	
		GSM8K		MATH-500		AIME24		AIME25		HumanEval		MBPP		LCB		SWE-Bench		MT-Bench		Alpaca		*Avg.*	
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×**	**1.52×**	**1.46×**	**1.48×**	**1.58×**	**1.54×**	**1.56×**	**1.57×**	**1.58×**	**1.55×**	**1.59×**	**1.61×**	**1.34×**	**1.42×**	**1.68×**	**1.61×**	**1.59×**	**1.51×**	**1.68×**	**1.57×**	**1.54×**	**1.53×**
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×**	**1.50×**	**1.48×**	**1.45×**	**1.62×**	**1.53×**	**1.71×**	**1.54×**	**1.64×**	**1.59×**	**1.68×**	**1.65×**	**1.35×**	**1.44×**	**1.74×**	**1.68×**	**1.62×**	**1.55×**	**1.71×**	**1.62×**	**1.58×**	**1.54×**
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×**	**1.53×**	**1.31×**	**1.45×**	**1.12×**	**1.56×**	**1.03×**	**1.55×**	**1.17×**	**1.32×**	**1.41×**	**1.35×**	**1.13×**	**1.43×**	**1.63×**	**1.53×**	**1.43×**	**1.56×**	**1.44×**	**1.57×**	**1.26×**	**1.46×**

Table 5. Decoding speedup over baseline and average τ on Nemotron-Labs-Diffusion models and a maximum of 2048 generated tokens when T=1.

Model	Method	MATH								CODE								CHAT				*Avg.*	
		GSM8K		MATH-500		AIME24		AIME25		HumanEval		MBPP		LCB		SWE-Bench		MT-Bench		Alpaca		*Avg.*	
Temperature = 1		Sp.	τ	Sp.	τ	Sp.	τ	Sp.	τ	Sp.	τ	Sp.	τ	Sp.	τ	Sp.	τ	Sp.	τ	Sp.	τ	Sp.	τ
3B	vanilla	2.3×	3.82	3.1×	4.15	2.2×	3.29	1.9×	3.50	2.2×	3.04	1.8×	2.74	2.1×	3.03	1.3×	1.96	1.5×	2.05	1.2×	1.78	2.0×	2.94
	PRESTO	4.4×	6.29	4.8×	7.03	4.0×	6.52	4.0×	6.31	3.4×	5.22	3.3×	4.82	3.5×	5.28	2.3×	3.70	2.4×	3.58	2.1×	3.12	3.4×	5.19
	× Gain	**1.91×**	**1.65×**	**1.55×**	**1.69×**	**1.82×**	**1.98×**	**2.11×**	**1.80×**	**1.55×**	**1.72×**	**1.83×**	**1.76×**	**1.67×**	**1.74×**	**1.77×**	**1.89×**	**1.60×**	**1.75×**	**1.75×**	**1.75×**	**1.70×**	**1.77×**
8B	vanilla	1.4×	2.97	1.8×	3.41	1.5×	2.58	1.7×	3.00	1.7×	2.61	1.6×	2.39	1.2×	2.28	1.0×	1.77	1.2×	1.73	1.0×	1.53	1.4×	2.43
	PRESTO	4.1×	5.65	4.0×	6.59	2.9×	5.10	3.3×	5.70	3.2×	4.94	3.0×	4.42	2.8×	4.75	2.2×	3.68	2.3×	3.53	2.0×	2.99	3.0×	4.73
	× Gain	**2.93×**	**1.90×**	**2.22×**	**1.93×**	**1.93×**	**1.98×**	**1.94×**	**1.90×**	**1.88×**	**1.89×**	**1.88×**	**1.85×**	**2.33×**	**2.08×**	**2.20×**	**2.08×**	**1.92×**	**2.04×**	**2.00×**	**1.95×**	**2.14×**	**1.95×**

PRESTO 's gains become substantially larger under stochastic decoding. Tables 4 and 5 report decoding speedup and average acceptance length at $T = 1$ on dFlash and Nemotron-Labs-Diffusion (NLD). On dFlash, PRESTO achieves an average $1.54\times$, $1.58\times$, and $1.26\times$ throughput improvement on Qwen3-4B, 8B, and 30B respectively, with consistent acceptance-length gains across all benchmarks. The improvements are even more pronounced on NLD, where vanilla self-speculative decoding suffers severe degradation under stochasticity: throughput drops to $2.0\times$ on NLD-3B and $1.4\times$ on NLD-8B, well below their $T = 0$ levels. PRESTO lifts these to $3.4\times$ and $3.0\times$, yielding $1.70\times$ and $2.14\times$ relative gains. This stronger improvement under $T = 1$ reflects PRESTO 's core advantage: stochastic decoding amplifies trajectory uncertainty, which linear drafting cannot accommodate, while PRESTO 's tree-structured drafting preserves multiple plausible paths and recovers acceptance length that linear drafting loses.

F. More Discussions and Ablations

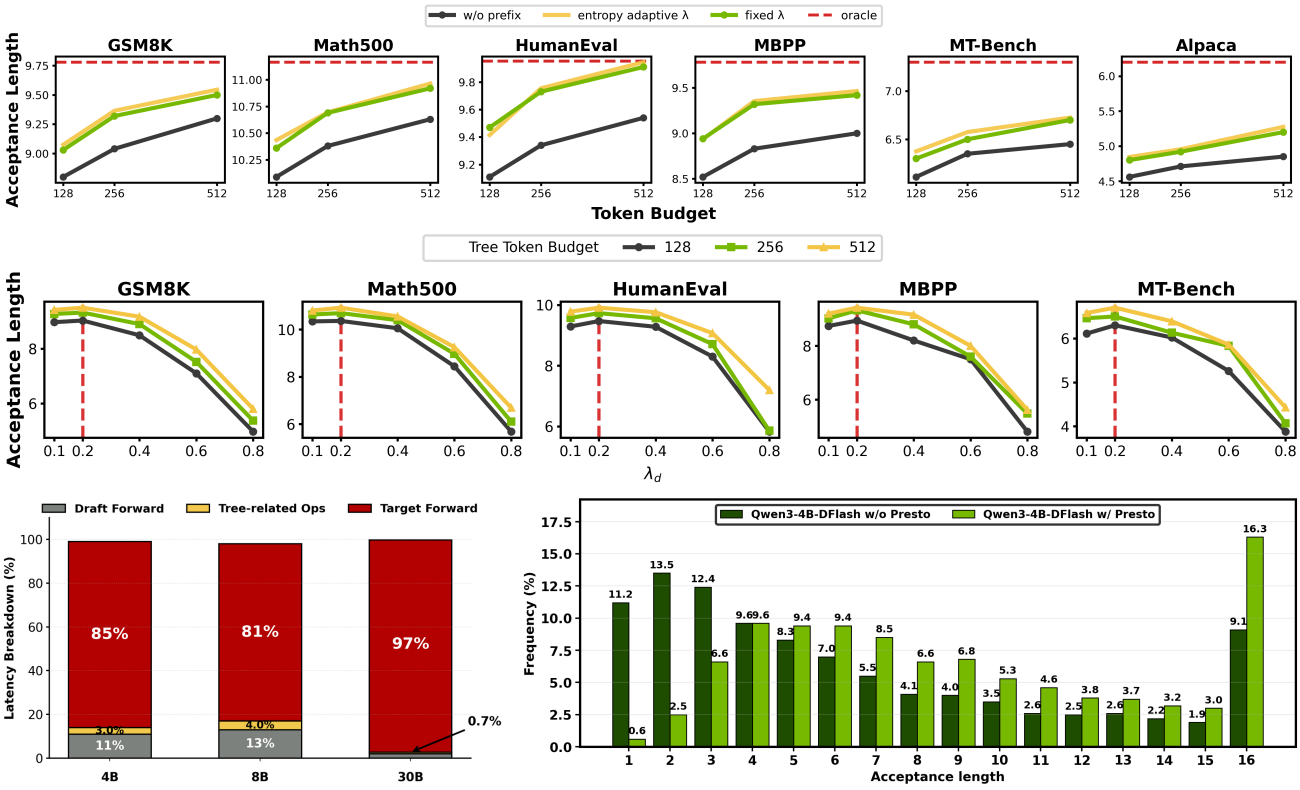

Figure 4. **Top**: prefix-conditioned signals consistently improve acceptance length. **Middle**: effect of λ_d under different datasets and tree token budgets. **Bottom left**: Per-step latency breakdown across different model scales. **Bottom right**: Comparison of accepted lengths distribution on GSM8K. Note that all results are obtained from Qwen3-4B-DFlash with diffusion block size being 16.

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. Beyond the fixed interpolation coefficient λ_d used in our main experiments, we explore an entropy-adaptive variant that scales the prefix correction by the diffusion drafter's uncertainty: confident predictions rely more on q_d, while uncertain ones up-weight ρ_d. Letting $H_d = -\sum_{w \in \mathcal{V}} q_d(w) \log q_d(w)$ 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 $C > 0$ stabilises the ratio. Substituting into the token-level score from Section 4 gives

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

which recovers the fixed-λ_d variant when $\lambda_d(H_d)$ is replaced by a constant. Empirically, this entropy-adaptive schedule yields only marginal gains over the fixed variant (Figure 4, top), suggesting that 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, which is almost negligible compared with the cost of target-model verification.

Acceptance Length Distribution Analysis. Figure 4 bottom right shows that PRESTO consistently shifts the acceptance-length distribution toward longer accepted prefixes. Compared to vanilla diffusion drafting, short acceptance cases are substantially reduced, while long-prefix and full-block acceptance become significantly more frequent.

G. Detailed Related work

G.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.

G.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. 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.

G.3. Diffusion-based Speculative Decoding

Recent work explores using diffusion models as drafters within speculative decoding, combining the parallelism of diffusion drafting with the quality guarantee of AR verification. TiDAR (Liu et al., 2025) jointly trains diffusion and autoregressive objectives in a sequence-level hybrid, enabling parallel "thinking" via diffusion and sequential "talking" via autoregressive decoding, though final generation quality is not yet lossless. DiffuSpec (Li et al., 2025a) and SpecDiff-2 (Sandler et al., 2025) employ large pretrained dLLMs as speculative drafters, with inference-time search or train–test alignment to improve acceptance. However, these approaches rely on massive drafters (*e.g.*, 7B parameters), incurring substantial memory and latency overhead. While they achieve long acceptance lengths, the high drafting cost often offsets the practical speedups in real-world serving scenarios. Recently, DFlash (Chen et al., 2026) proposes to employ a lightweight block-diffusion drafter conditioned on context features extracted from the target model, generating an entire block of draft tokens in a single forward pass and reporting over $6\times$ lossless acceleration. Similarly, DART (Liu et al., 2026) performs parallel logit prediction over multiple masked positions and assembles drafts via N-gram-guided tree pruning. However, its tree construction relies on a fixed, hand-tuned scoring rule with hard-coded hyperparameters, which leaves no principled way to adapt the tree shape to different budgets, drafters, or target models without re-tuning. In contrast, PRESTO formulates tree construction as priority-based expansion under a unified path score, yielding a single algorithmic framework that extends across budgets and drafter designs.

H. Limitations

PRESTO requires a tractable prefix-dependent signal ρ_d to correct the diffusion drafter's prefix-blind marginals; we use a 3-gram model, which is cheap and effective but captures only short-range lexical compatibility, and richer choices remain to be explored under the constraint that ρ_d stays cheap enough not to erode the speedup. Moreover, our entropy-adaptive variant of λ_d yielded only marginal gains, suggesting that more sophisticated adaptation may require a learned schedule. Last, characterising PRESTO across larger batch sizes, longer contexts, and a broader hardwares is also meaningful, which we leave to future work.