PRESTO: Prefix-Aligned Tree Drafting for Diffusion Speculative Decoding

Zheng Wang [*1] **Zhifan Ye** [*2 3] **Qi Cheng** [3] **Yonggan Fu** [3] **Ziyan Wang** [2] **Feng Zhu** [2] **Haozhe Zhao** [1]
Pavlo Molchanov [3] **Humphrey Shi** [2 3] **Minjia Zhang** [1]

Abstract

Diffusion Large Language Models (dLLMs) have emerged as a promising alternative to autoregressive (AR) LLMs, generating tokens in parallel. This makes them effective draft models for speculative decoding (SD), producing an entire block of draft tokens in a single forward pass. Yet existing diffusion-based drafting methods rely on linear drafting, even though dLLMs emit multiple candidate tokens across positions, inducing a large combinatorial space of decoding paths. Consequently, they limit acceptance length and decoding efficiency. To exploit this multi-candidate structure, we apply tree-based drafting to diffusion drafters, enabling exploration of diverse candidate paths. However, we find that naive tree drafting is suboptimal: diffusion marginals are prefix-blind, mismatching the prefix-based AR verification and yielding unreliable path ranking. We propose PRESTO, a principled framework that extends tree-based drafting to diffusion drafters while resolving the fundamental mismatch between diffusion draft confidence and prefix-based AR verification through ***PRE**fix-aligned **S**coring* and *priority-based **T**ree search* for diffusion speculative dec**O**ding. 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. Extensive experiments show that PRESTO achieves up to an average of **1.5**× end-to-end throughput speedup on the state-of-the-art dedicated diffusion drafter SD and an average of **1.12**× on self-speculative diffusion LLMs across diverse benchmarks.

[1]University of Illinois Urbana-Champaign, USA [2]Georgia Institute of Technology, USA [3]NVIDIA, USA. Correspondence to: Zheng Wang <zheng.wang@illinois.edu>, Zhifan Ye <zhifan.ye@gatech.edu>.

Proceedings of the 43rd International Conference on Machine Learning, Seoul, South Korea. PMLR 306, 2026. Copyright 2026 by the author(s).

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., 2025a; 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 (NLD) (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 further improve the throughput of diffusion-based speculative decoding by increasing acceptance length?

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

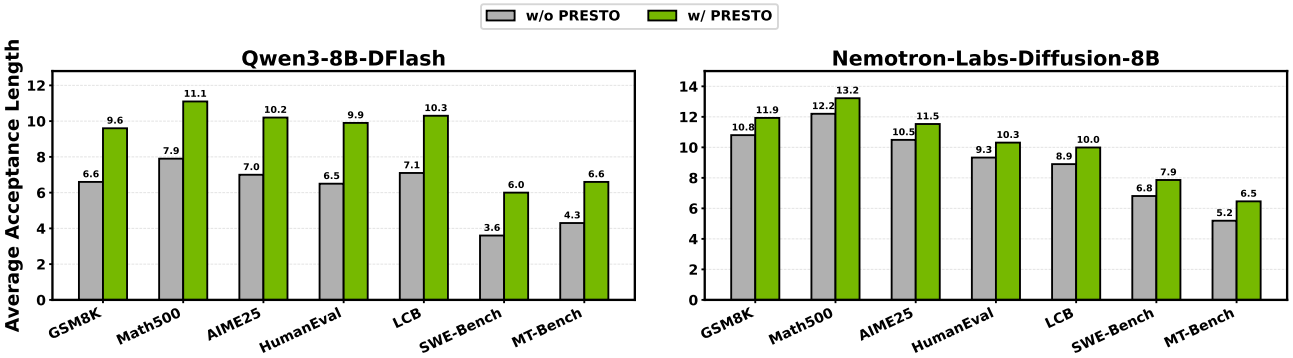

Figure 1. Acceptance length comparison of PRESTO on diffusion-based SD, including dedicated diffusion drafter SD (Qwen3-8B-DFlash) and self-speculative dLLM (Nemotron-Labs-Diffusion-8B; marginal-only scoring with no prefix-signal) across benchmarks. PRESTO consistently improves acceptance length over the baseline across all tasks, enabling more tokens to be accepted in each verification round.

but yielding marginal rather than prefix-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 (SS) 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 formalize tree-based speculative drafting for diffusion drafters.** We introduce PRESTO, a principled tree-based diffusion drafting framework that explores high-quality draft paths through priority-based tree expansion and prefix-aligned scoring.
- **PRESTO generalizes across both dedicated and self-speculative diffusion SD frameworks.** PRESTO is the first tree-based drafting framework applicable to both dedicated diffusion drafters and self-speculative diffusion LLMs, achieving an average throughput speedup of **1.5**× on dFlash and an average of **1.12**× on Nemotron-Labs-Diffusion over linear drafting.

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,

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

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 $\mathcal{B}_B = \{\mathcal{T} : |\mathcal{T}| \leq B\}$ and let $\alpha_{\mathcal{T}}(P)$ denote the longest prefix of P contained in $\mathcal{T}$ (tree budget). 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

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

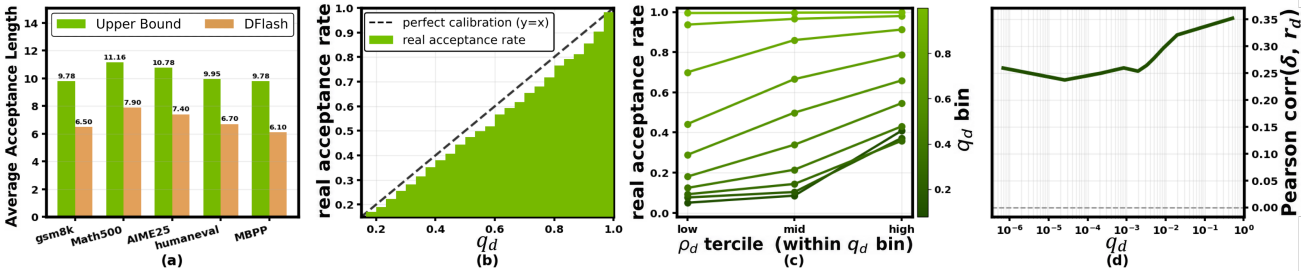

Figure 2. (a) Linear drafting leaves room for improvement via multi-path exploration. (b) The predicted confidence from q_d closely matches the empirical acceptance rate. (c) Within fixed q_d bins, acceptance rates increase from low to high ρ_d terciles, indicating that prefix-conditioned compatibility 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 correlates with 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 already captures meaningful global acceptance trends.

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 informative estimates of global acceptance, but it is 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

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

tree drafting to self-speculative dLLMs.

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 $|\mathcal{T}| = B$. As 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

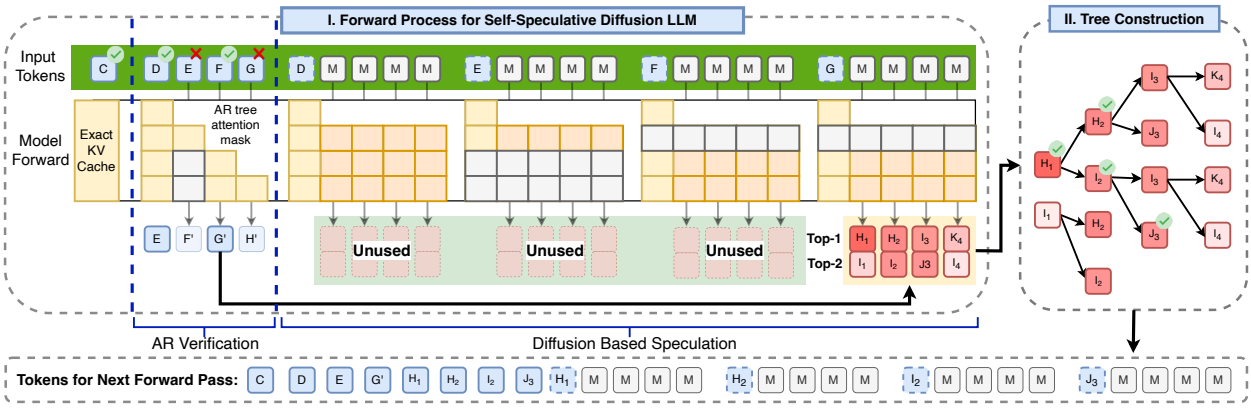

Figure 3. PRESTO is the first to explore tree-based drafting for quadratic SS dLLMs. The detailed process is in Appendix D.

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.

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, where 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 hybrid dLLMs

The above tree-construction algorithm can be naturally extended to support quadratic SS decoding in hybrid dLLMs (Liu et al., 2025; Fu et al., 2026a)[2], 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, PRESTO has consistently led to improved average acceptance length and end-to-end throughput.

[2]NLD primarily highlights a linear SS decoding mode. Applying PRESTO to this setting is identical to the dedicated diffusion drafter setting discussed earlier. Note that the NLD paper also introduces a quadratic SS variant which is explored in this work.

5. Empirical Validation

We show that PRESTO significantly improves the decoding efficiency of both standard and SS diffusion SDs.

5.1. Experiment Setup

Models, Tasks, and Metrics. We evaluate PRESTO on: (i) standard SD with a separate diffusion drafter and AR target: dFlash with Qwen3-4B/8B and Qwen3-Coder-30B-A3B; and (ii) SS dLLM: NLD-8B at both linear and quadratic SS decoding modes. We test three task categories: **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 **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. For dFlash and NLD under quadratic SS, we use a PyTorch implementation with FlexAttention (Dong et al., 2024); under linear SS, we integrate PRESTO into SGLang with FlashInfer (Ye et al., 2025b). The prefix-aligned signal is an n-gram model ($n = 3$)[3], with $\lambda_d = 0.2$, batch size 1, and beam search with global retention (width 10, top-$k = 10$ per position). We evaluate PRESTO under deterministic ($T = 0$) and stochastic ($T = 1$) decoding, sweeping tree budgets $\mathcal{T} \in \{128, 256, 512\}$ for dFlash (reporting the best end-to-end throughput) and fixing $\mathcal{T} = 32$ for Nemotron-Labs-Diffusion per its speculative block size.

[3]Following the n-gram construction in (Liu et al., 2026).

Table 2. Decoding speedup 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×**	

Table 3. Decoding speedup and average τ on NLD-8B at both linear and quadratic SS decoding mode and a maximum of 2048 generated tokens.

Decoding Mode	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
Linear	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×	8.7
	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
	× **Gain**	**1.05×**		**1.04×**		**1.07×**		**1.06×**		**1.04×**		**1.04×**		**1.04×**		**1.08×**		**1.14×**		**1.16×**		**1.07×**	
Quadratic	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
	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
	× **Gain**	**2.04×**		**1.63×**		**1.09×**		**1.12×**		**1.16×**		**1.19×**		**1.29×**		**1.50×**		**1.33×**		**1.28×**		**1.35×**	

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-Labs-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 HumanEval, 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. Under linear SS decoding, PRESTO improves average speedup from $4.6\times$ to $4.9\times$ and average acceptance length from 8.7 to 9.9, with consistent gains across all benchmarks. The improvements are also large under quadratic SS decoding, where average speedup increases from $2.6\times$ to $3.5\times$ and average acceptance length from 4.7 to 5.6. 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. 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. Extending PRESTO to production serving frameworks such as vLLM, together with characterizing its behavior under larger batch sizes remains meaningful future work.

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.

Dong, J., Feng, B., Guessous, D., Liang, Y., and He, H. Flex attention: A programming model for generating optimized attention kernels, 2024. URL https://arxiv.org/abs/2412.05496.

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., Liu, J., 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. May 2026a. URL https://d1qx31qr3h6wln.cloudfront.net/publications/Nemotron_Diffusion_Tech_Report.pdf?VersionId=1tm4XZATEzGV7cs51XAf.xmWupU20vYW. Technical report.

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, 2025a. URL https://arxiv.org/abs/2508.15487.

Ye, Z., Chen, L., Lai, R., Lin, W., Zhang, Y., Wang, S., Chen, T., Kasikci, B., Grover, V., Krishnamurthy, A., and Ceze, L. Flashinfer: Efficient and customizable attention engine for llm inference serving, 2025b. URL https://arxiv.org/abs/2501.01005.

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

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

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

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

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

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

Equivalently,

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

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

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

Define the surrogate prefix mass

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

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 achievable upper bound, at each position, we retain the top-10 candidates from the diffusion marginal distribution and perform a position-wise oracle matching analysis. Specifically, we check whether the ground-truth token at each position appears in the corresponding top-10 candidate set, and then compute the longest contiguous prefix length covered by these matches. The reported upper bound is equivalent to the result that would be obtained by exhaustive enumeration. However, we do not explicitly enumerate all 10^{16} paths. Instead, we exploit the fact that acceptance is determined by whether the ground-truth token is covered by the top-k candidate set at each position. This allows us to compute the same oracle upper bound efficiently through position-wise matching.

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. Full Experimental Results

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.

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×**	

Table 5. Decoding speedup over baseline and average acceptance length (τ) on Nemotron-Labs-Diffusion-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
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×	8.7
	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
	× **Gain**	**1.05×**		**1.04×**		**1.07×**		**1.06×**		**1.04×**		**1.04×**		**1.04×**		**1.08×**		**1.14×**		**1.16×**		**1.07×**	
Temperature = 1		Sp.	τ	Sp.	τ	Sp.	τ	Sp.	τ	Sp.	τ	Sp.	τ	Sp.	τ	Sp.	τ	Sp.	τ	Sp.	τ	Sp.	τ
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
	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
	× **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 dFlash. As illustrated in Table 4, PRESTO consistently achieves substantial improvements in the average acceptance length 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 7.9 to 10.9 on MATH-500 and from 7.4 to 10.3 on AIME25, while on code benchmarks it improves from 6.9 to 10.0 on LCB and from 6.7 to 9.9 on HumanEval, leading to corresponding 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: even for stronger drafters such as Qwen3-8B and under stochastic decoding ($T = 1$), PRESTO continues to deliver comparable relative gains. These results indicate that PRESTO effectively pushes the acceptance length of dFlash closer to its upper bound, delivering consistent and substantial end-to-end throughput speedup gains.

PRESTO on Nemotron-Labs-Diffusion (NLD). As shown in Table 5, PRESTO consistently improves both throughput and average acceptance length over the official linear self-speculative decoding implementation of Nemotron-Labs-Diffusion across all benchmarks and decoding temperatures. Notably, these experiments are conducted using the highly optimized SGLang implementation released by NLD, where substantial additional gains are already difficult to obtain. Nevertheless,

Table 6. Decoding speedup over baseline and average acceptance length (τ) on Nemotron-Labs-Diffusion-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
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
	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
	× 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
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
	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
	× Gain	**2.93×**		**2.22×**		**1.93×**		**1.94×**		**1.88×**		**1.88×**		**2.33×**		**2.20×**		**1.92×**		**2.00×**		**2.14×**	

by replacing linear drafting with tree-based exploration while preserving the original verification pipeline, PRESTO remains effective across a wide range of tasks. Under deterministic decoding ($T = 0$), PRESTO increases average end-to-end throughput from $4.6\times$ to $4.9\times$ while improving average acceptance length from 8.7 to 9.9, with consistent gains observed on all benchmarks. Under stochastic decoding ($T = 1$), where a single drafted trajectory is more likely to diverge from the target model, 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. Across individual benchmarks, PRESTO achieves up to $1.29\times$ end-to-end throughput improvement and increases acceptance length by as much as 39% on Alpaca under stochastic decoding (T=1). These results demonstrate that the benefits of tree-based drafting persist even within a production-grade, highly optimized inference engine.

PRESTO on Nemotron-Labs-Diffusion (NLD) at quadratic self speculation mode. We further evaluate PRESTO under the quadratic self-speculative decoding setting introduced in Nemotron-Labs-Diffusion. As Table 6 shows, across all benchmarks and decoding settings, PRESTO also consistently outperforms vanilla quadratic self-speculative decoding in both end-to-end throughput and average acceptance length. Under deterministic decoding ($T = 0$), PRESTO provides stable gains across math, code, and chat tasks, improving average speedup from $2.6\times$ to $3.5\times$ on NLD-8B, while also consistently increasing acceptance length from 4.7 to 5.6. The largest gains are observed on math benchmarks, where PRESTO achieves up to $2.04\times$ on GSM8k for NLD-8B. Gains are largest under stochastic decoding ($T = 1$), where vanilla single-path drafting degrades sharply from trajectory mismatch and accumulated uncertainty, while PRESTO 's tree-structured exploration retains multiple plausible continuations. On NLD-8B, this 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, reaching up to $2.9\times$ throughput and over $2\times$ acceptance length over vanilla self-speculative decoding. These positive results indicate the significant benefits of applying PRESTO to quadratic self-speculation.

We note that the throughput for dFlash and NLD quadratic self speculation can be further improved through system-level optimizations, including custom kernels, optimized KV-cache management, and efficient request scheduling. The goal here is not to fully optimize absolute throughput, but rather to provide a first-hand comparison between linear and tree-based drafting for diffusion drafters under the same native PyTorch implementation.

F. More Discussions and Ablations

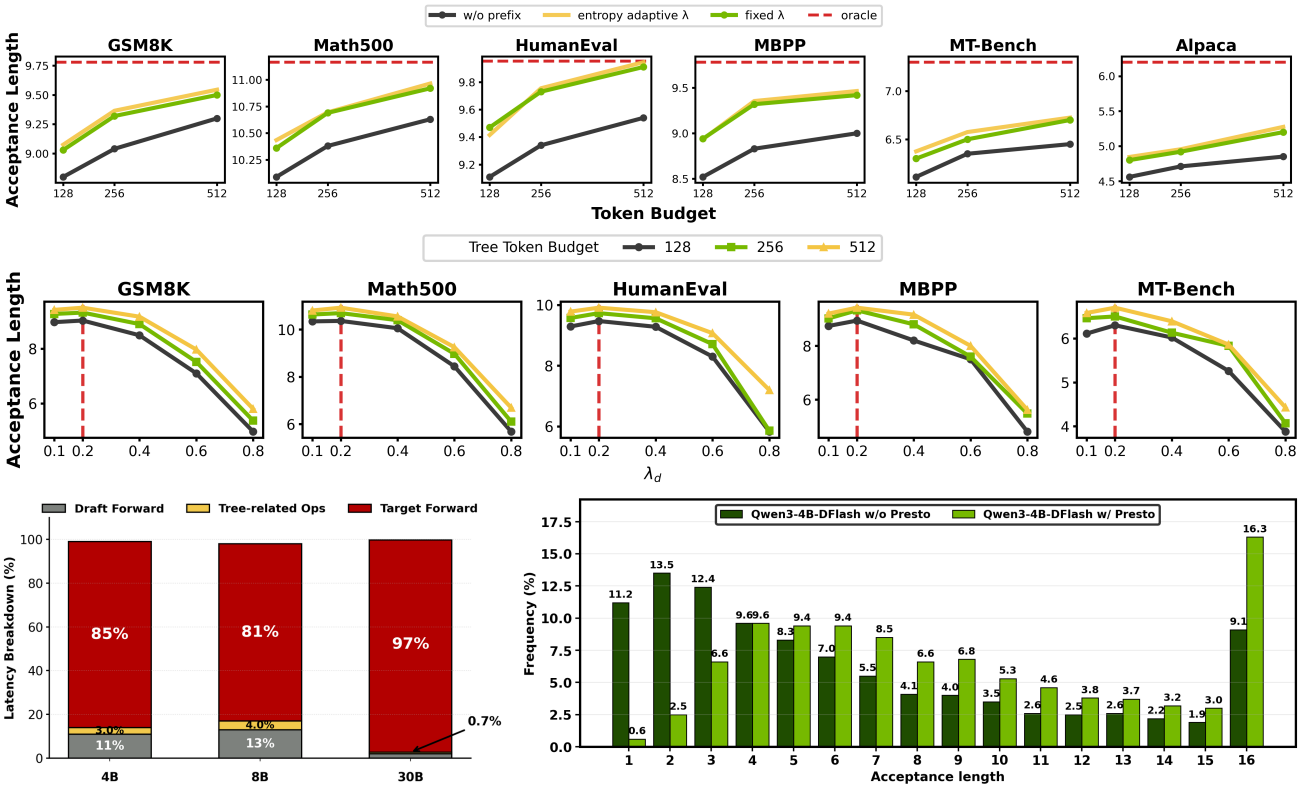

Figure 4. **Top**: prefix-conditioned signals consistently improve acceptance length. **Middle**: effect of λ_d under different datasets and tree token budgets. **Bottom left**: Per-step latency breakdown across different model scales. **Bottom right**: Comparison of accepted lengths distribution on 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 the 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. 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 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.

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 primarily focus on improving the quality or alignment of the diffusion drafter itself, often relying on massive drafters (*e.g.*, 7B parameters), which incur substantial memory and latency overhead. While they achieve long acceptance lengths, the high drafting cost can offset the practical speedups in real-world serving scenarios. In contrast, PRESTO studies how to more effectively exploit the rich multi-position candidate space produced by diffusion drafting through prefix-aware tree construction and priority-based expansion, making it complementary to stronger diffusion drafters and existing diffusion-based speculative decoding systems. 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 naturally generalizes across different tree budgets and diffusion drafter designs.