PSD: Pushing the Pareto Frontier of Diffusion LLMs via Parallel Speculative Decoding

Shengyin Sun
Huawei Technologies
sun.shengyin1@huawei.com

Yiming Li*
Huawei Technologies
li.yiming4@huawei.com

Renxi Liu
Huawei Technologies

Xinqi Li
Huawei Technologies

Hui-Ling Zhen
Huawei Technologies

Weizhe Lin
Huawei Technologies

Chen Chen
Huawei Technologies

Xianzhi Yu
Huawei Technologies

Mingxuan Yuan
Huawei Technologies

Chen Ma
City University of Hong Kong

Abstract

Diffusion large language models (dLLMs) generate text by iteratively denoising masked token sequences. Although dLLMs can predict all masked positions in parallel within each step, the large number of denoising iterations still makes inference expensive. This cost can be reduced spatially by unmasking multiple tokens per step, or temporally by collapsing multiple denoising steps into one verification call. We propose Parallel Speculative Decoding (PSD), a training-free framework that jointly improves inference along both axes. Using the confidence scores from a single forward pass, PSD selects positions to unmask via a configurable, adaptive unmasking policy and constructs multi-depth speculative drafts without extra model calls. A final batched verification pass then applies hierarchical acceptance, keeping the deepest draft that remains consistent with the updated predictions. Experiments on three dLLMs across reasoning and code generation tasks show that PSD achieves favorable trade-offs between inference efficiency and generation quality, reaching up to $5.5\times$ tokens per forward pass with accuracy comparable to greedy decoding. Code is available at `https://github.com/sunshy-1/PSD`.

1 Introduction

Large language models (LLMs) have demonstrated remarkable capabilities in reasoning and generation [OpenAI et al., 2024, Grattafiori et al., 2024], yet the autoregressive paradigm suffers from sequential token-by-token decoding, leading to high latency and poor hardware utilization [Ivanov et al., 2021, Sun et al., 2025a, 2026a]. Diffusion language models (dLLMs) [Nie et al., 2025, Ye et al., 2025, Lin et al., 2026] have recently emerged as a promising alternative, formulating text generation as iterative denoising over discrete masked token sequences. Starting from a fully masked sequence, a dLLM progressively reveals tokens over multiple denoising steps, and at each step the model produces predictions for all masked positions simultaneously. This property makes dLLMs naturally amenable to parallel token generation, opening a path toward potentially more efficient inference.

Despite this potential, dLLM inference still requires many denoising iterations, each involving a model forward pass, and deployment efficiency remains a practical bottleneck [Fu et al., 2025a, Peng et al., 2025]. Two families of acceleration strategies have been proposed for dLLMs. Parallel

*Corresponding author.

40th Conference on Neural Information Processing Systems (NeurIPS 2026).

decoding methods [Kong et al., 2025, Xu et al., 2025, Wu et al., 2026a,b] compress the spatial dimension by unmasking multiple tokens per step, but quality degrades as the parallelism grows more aggressive, since low-confidence tokens are forced to commit prematurely and errors propagate through subsequent steps. Speculative decoding methods [Chen et al., 2023, Agrawal et al., 2025] instead compress the temporal dimension by anticipating future denoising iterations and verifying them in a single batched forward pass; however, existing approaches still unmask only one token per denoising step, leaving the spatial dimension unexploited and thereby capping the speedup.

Each strategy, however, faces an inherent ceiling when applied in isolation. For parallel decoding, increasing the number of tokens unmasked per step reduces the total number of iterations but inevitably degrades generation quality, yielding a quality–speed Pareto frontier that no transfer policy can escape. For speculative decoding, the speedup is bounded by the acceptance rate of speculated tokens; since each step still reveals only a single token, even perfect speculation yields limited gains. Both families therefore saturate at moderate acceleration levels, leaving considerable room for improvement. We observe, however, that these two axes of acceleration are largely complementary: the spatial axis governs how many tokens are revealed within a single denoising step, while the temporal axis governs how many denoising steps are collapsed into one verification call, and we find empirically that their speedups combine to deliver substantially larger gains than either axis alone.

In this work, we propose *Parallel Speculative Decoding (PSD)*, a framework that exploits this complementarity by unifying spatial and temporal acceleration for dLLMs (as shown in Figure 1). Rather than prescribing a specific parallel decoding policy, PSD is a general framework that can be built on arbitrary parallel decoding algorithms. PSD builds on a key observation: the set of high-confidence masked positions tends to stabilize as denoising progresses, so the positions currently identified as high-confidence will largely remain high-confidence after a few more tokens are revealed.

More specifically, PSD operates in three stages (Figure 1): (1) spatial parallel unmasking applies a configurable transfer policy to reveal multiple tokens per step; (2) temporal speculative drafting constructs candidate future denoising outcomes at multiple depths by filling remaining positions in descending confidence order, all without additional model calls; and (3) batched verification with hierarchical acceptance evaluates all candidates in a single forward pass and retains the deepest branch whose speculated tokens agree with the verifier's updated predictions, preventing error propagation from aggressive parallel unmasking. These designs allow PSD to combine the gains of both acceleration families within a single decoding pass.

We evaluate PSD on three open-source dLLMs that span different training paradigms across multiple benchmarks. Empirically, PSD achieves a favorable efficiency–quality trade-off across all tested configurations: it delivers additional speedup over temporal-only methods through spatial parallelism, yielding substantially more tokens per forward pass with negligible quality loss, while parallel-only baselines at comparable parallelism levels suffer notable accuracy degradation. Across all model$\times$benchmark settings, PSD consistently matches or improves upon the Pareto frontier established by either family of baselines. Our contributions are summarized as follows:

- We identify a complementary relationship between spatial and temporal acceleration in dLLMs: the two axes compress different dimensions of the decoding process, and we empirically show that combining them yields substantially larger speedups than either axis alone.
- We propose PSD, a training-free, policy-agnostic framework that couples any parallel unmasking policy with speculative verification in a single decoding pass, using a hierarchical acceptance mechanism to prevent error propagation from aggressive parallel unmasking.
- Experiments on three dLLMs across mathematical reasoning and code generation benchmarks show that PSD achieves up to $5.5\times$ tokens per forward pass with negligible accuracy loss, consistently matching or improving the Pareto frontier of prior methods.

2 Background

Our framework builds on two complementary lines of work: dLLMs, which formulate text generation as iterative denoising over masked sequences, and speculative decoding, which accelerates inference via a draft-then-verify paradigm. The former progressively generates text through successive denoising steps, while the latter improves decoding efficiency by proposing candidate tokens and verifying them with a target model. We review each below to motivate the design of PSD.

2.1 Diffusion Language Models (dLLMs)

As an emerging paradigm, dLLMs [Nie et al., 2025, Ye et al., 2025, Zhu et al., 2025a] formulate text generation as a learned reversal of a token-level masking process. During training, a clean sequence is corrupted by independently replacing tokens with a mask token `[M]`, and the model is trained to recover the original tokens from the corrupted input [Li et al., 2025a, Sun et al., 2025b]. At inference, generation starts from a fully masked sequence and proceeds iteratively: at each denoising step, the model predicts tokens for all masked positions and then unmasks a subset of them, progressively refining the sequence until all positions are filled [Bie et al., 2025, Ma et al., 2025, Zhu et al., 2025b].

The inference cost is governed by k, the number of tokens unmasked per step. When $k=1$, producing N tokens demands N sequential forward passes, placing dLLMs at a latency regime comparable to autoregressive models of similar size. Setting $k > 1$ reduces the total iteration count to $\lceil N/k \rceil$ but forces low-confidence positions to commit prematurely, leading to notable quality degradation [Wu et al., 2026a, Kong et al., 2025]. This fundamental tension between throughput and accuracy motivates acceleration methods that can reduce the forward-pass count without sacrificing generation fidelity.

2.2 Speculative Decoding

Speculative decoding [Leviathan et al., 2023, Chen et al., 2023, Sun et al., 2026b] accelerates autoregressive LLM inference through a draft-then-verify paradigm. Instead of invoking the large model for every decoding step, a lightweight drafter first proposes several future tokens, and the large target model then evaluates these proposals in a single forward pass [Xia et al., 2024, Sun et al., 2026c]. Tokens whose draft predictions agree with the target model are accepted, so one expensive target-model call can advance the decoding process by multiple steps while still preserving the target distribution [He et al., 2024, Luo et al., 2025, Oliaro et al., 2024, Abramovich et al., 2026, Li et al., 2026a].

Adapting this paradigm to dLLMs, however, is non-trivial [Agrawal et al., 2025, Gao et al., 2025]. In autoregressive generation, tokens are produced in a fixed left-to-right order, so the next prediction target is always clear. In dLLMs, by contrast, the unmasking order is determined dynamically by model confidence and is not known in advance [Li et al., 2026b]; the unit of speculation therefore shifts from a single next token to an entire set of positions to be revealed at each step.

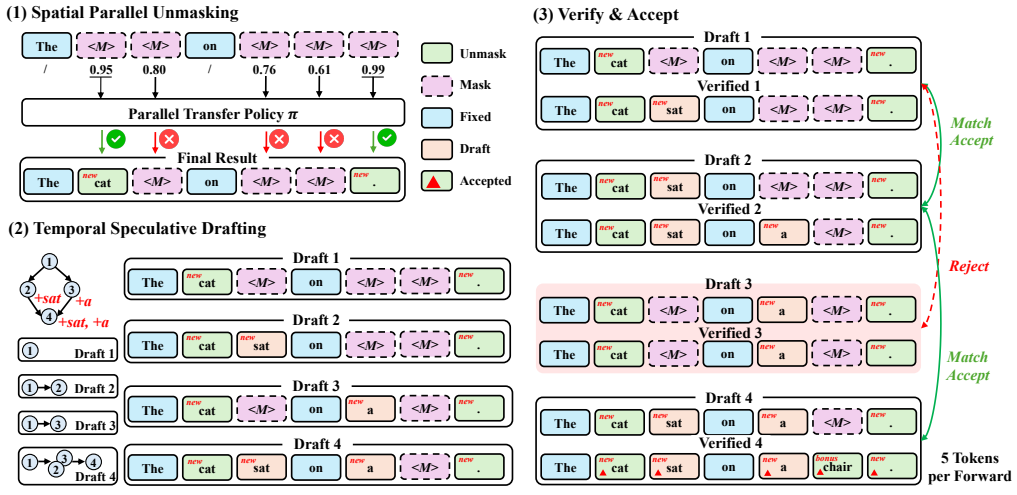

Figure 1: Overview of the proposed PSD. As a general framework compatible with any parallel decoding strategy, PSD consists of three stages: *(1) Spatial parallel unmasking* reveals multiple tokens per denoising step; *(2) Temporal speculative drafting* builds a directed acyclic graph of confidence-ranked drafts for future steps; *(3) Verify & Accept* evaluates all drafts in one batched forward pass and accepts the deepest consistent branch. By combining spatial and temporal acceleration, PSD achieves compounded speedups beyond either axis alone.

3 Methodology

In this section, we first formalize the dLLM decoding process and then present the three stages of PSD (Figure 1): spatial parallel unmasking reveals multiple tokens per step via an arbitrary transfer policy, temporal speculative drafting constructs candidate future outcomes, and batched verification with hierarchical acceptance retains the deepest consistent branch in a single forward pass.

3.1 Preliminaries

Denote the sequence state at denoising step t by $\mathbf{x}^{(t)} \in (\mathcal{V} \cup \{\texttt{[M]}\})^N$, where $\mathcal{V}$ is the vocabulary and `[M]` is a distinguished mask token. At each step t and each masked position i, the model assigns a probability $p_\theta(x_i{=}v \mid \mathbf{x}^{(t)})$ to every token $v \in \mathcal{V}$. We denote the most likely token by $\hat{x}_i^{(t)} = \arg\max_{v \in \mathcal{V}} p_\theta(x_i{=}v \mid \mathbf{x}^{(t)})$ and its confidence by $c_i^{(t)} = \max_{v \in \mathcal{V}} p_\theta(x_i{=}v \mid \mathbf{x}^{(t)})$. Let $\mathcal{M}(\mathbf{x}^{(t)}) = \{i : x_i^{(t)} = \texttt{[M]}\}$ be the set of masked positions at step t. A transfer policy π then selects a subset $\mathcal{T}^{(t)} \subseteq \mathcal{M}(\mathbf{x}^{(t)})$ of positions to unmask at that step, and $\bar{r}_\pi = \mathbb{E}[|\mathcal{T}^{(t)}|]$ denotes the average number of positions revealed per step.

3.2 Parallel Speculative Decoding

As illustrated in Figure 1, PSD integrates spatial and temporal acceleration into a unified three-stage pipeline. *Stage (1) Spatial parallel unmasking* reveals multiple tokens per denoising step via an *arbitrary* transfer policy, PSD is agnostic to this choice. *Stage (2) Temporal speculative drafting* reuses the confidence scores from the current step to construct a directed acyclic graph of candidate future denoising outcomes without additional model calls. *Stage (3) Batched verification with hierarchical acceptance* evaluates all draft candidates in a single forward pass and accepts the deepest branch that remains consistent with the verifier's updated predictions. We detail each stage below.

Spatial parallel unmasking. (Stage (1) in Figure 1.) Although transfer policies such as confidence thresholding can in principle operate over the entire sequence, state-of-the-art dLLMs organize generation into contiguous blocks of length L [Wu et al., 2026a], a design shared by recent methods including LocalLeap and LoPA [Yu et al., 2025, Kong et al., 2025, Xu et al., 2025, Zhong et al., 2026]. Within each block, all L positions are initially set to `[M]`, and the model then enters an iterative denoising loop indexed by $t = 0, 1, \ldots, T$. At iteration t, a single forward pass produces the predicted token $\hat{x}_i^{(t)}$ and confidence score $c_i^{(t)}$ for every masked position $i \in \mathcal{M}(\mathbf{x}^{(t)})$. The transfer policy then chooses a set $\mathcal{T}^{(t)}$ of positions to unmask, and the sequence state advances as

$$x_i^{(t+1)} = \begin{cases} \hat{x}_i^{(t)}, & i \in \mathcal{T}^{(t)}, \\ x_i^{(t)}, & \text{otherwise.} \end{cases} \tag{1}$$

The loop terminates at step T when $\mathcal{M}(\mathbf{x}^{(T)}) = \varnothing$, requiring $T \approx \lceil L/\bar{r}_\pi \rceil$ forward passes. Spatial acceleration alone faces a fundamental ceiling: increasing $\bar{r}_\pi$ reduces T but degrades generation quality, yielding a quality / speed Pareto frontier that no transfer policy can escape. Importantly, PSD imposes no constraints on the choice of π: any existing or future parallel decoding policy can serve as the spatial backbone, and the subsequent speculative stages remain unchanged.

Temporal speculative drafting. (Stage (2) in Figure 1.) To push beyond this ceiling, PSD couples spatial unmasking with temporal speculation. The key observation is that, after step t, the confidence ranking over still-masked positions tends to remain stable across the next few denoising iterations. This ranking can therefore be used to construct draft candidates that approximate future reveal sets without additional model calls. After applying the transfer policy π at iteration t to produce $\mathbf{x}^{(t+1)}$, let $m = |\mathcal{M}(\mathbf{x}^{(t+1)})|$ denote the number of positions that remain masked within the block. We reuse the confidence scores from iteration t for these remaining positions and sort them in descending order to obtain the speculative ordering $\boldsymbol{\sigma} = (\sigma_1, \ldots, \sigma_m)$, where each $\sigma_j \in \mathcal{M}(\mathbf{x}^{(t+1)})$ and $c_{\sigma_1}^{(t)} \geq \cdots \geq c_{\sigma_m}^{(t)}$. Following the directed draft graph framework of Agrawal et al. [2025], we organize K speculative drafts as the nodes of a directed acyclic graph $\mathcal{G} = (\mathcal{N}, \mathcal{E})$. Each node $k \in \mathcal{N}$ is associated with a subset $\mathcal{S}_k \subseteq \{\sigma_1, \ldots, \sigma_m\}$ of positions to speculatively unmask. The draft

sequence $\tilde{\mathbf{x}}_k = (\tilde{x}_{k,1}, \ldots, \tilde{x}_{k,N})$ is defined entry-wise as

$$\tilde{x}_{k,i} = \begin{cases} \hat{x}_i^{(t)}, & i \in \mathcal{S}_k, \\ x_i^{(t+1)}, & \text{otherwise}, \end{cases} \tag{2}$$

which replaces the speculatively filled positions with the cached step-t predictions while leaving already-committed tokens unchanged. The root node r has $\mathcal{S}_r = \varnothing$ (hence $\tilde{\mathbf{x}}_r = \mathbf{x}^{(t+1)}$), representing the current state with no speculation. An edge $(p, k) \in \mathcal{E}$ exists whenever $\mathcal{S}_p \subset \mathcal{S}_k$, indicating that node k extends node p by filling $|\mathcal{S}_k \setminus \mathcal{S}_p|$ additional positions. Because the bidirectional attention of dLLMs makes the unmasking order immaterial, a node may have multiple parents, in contrast to the tree structures used in autoregressive speculative decoding, which increases the number of pathways through which a deeper draft can be accepted.

This construction relies on two approximations: (i) the confidence ranking from step t remains a good proxy for the ranking at future steps, and (ii) the predicted tokens $\hat{x}_i^{(t)}$ remain accurate despite context changes from intervening unmaskings. The topology of $\mathcal{G}$ is calibrated offline following Agrawal et al. [2025] to maximize empirical throughput. At inference time, $\mathcal{G}$ is fixed, and all K drafts are constructed by simple index-level operations on cached predictions, requiring no extra model calls.

Batched verification and hierarchical acceptance. (Stage (3) in Figure 1.) All K draft sequences are concatenated into a single batch and evaluated in a single model forward pass. For each node k and each masked position $i \in \mathcal{M}(\tilde{\mathbf{x}}_k)$, the verifier returns an updated prediction $\hat{x}_{k,i} = \arg\max_{v \in \mathcal{V}} p_\theta(x_i{=}v \mid \tilde{\mathbf{x}}_k)$ and its associated confidence $c_{k,i} = \max_{v \in \mathcal{V}} p_\theta(x_i{=}v \mid \tilde{\mathbf{x}}_k)$. Because all drafts share the same prompt prefix and differ only in the speculatively filled positions, KV-cache sharing can be exploited to reduce redundant computation.

Once all drafts have been verified, the acceptance step traverses $\mathcal{G}$ in topological order to find the deepest draft whose speculated tokens all agree with the verifier's greedy predictions. The root r is unconditionally accepted since it contains no speculation. For each child node k with parent p, we check only the newly speculated positions in $\mathcal{S}_k \setminus \mathcal{S}_p$: a speculated token $\tilde{x}_{k,i}$ is accepted if the verifier's prediction at position i satisfies the same acceptance criterion that the transfer policy π uses to commit tokens (e.g., the verifier assigns $\tilde{x}_{k,i}$ sufficient confidence). Because $\mathcal{G}$ is a DAG, a child may have multiple parents and is accepted whenever it is consistent with at least one of them. The denoising state then advances to the accepted draft that reveals the most positions, amortizing multiple future unmasking steps into a single verification call. More details are shown in the Appendix A.

4 Experiments

In this section, we evaluate PSD on three open-source dLLMs across mathematical reasoning and code generation benchmarks, against seven representative spatial and temporal decoding baselines. Beyond the accuracy-vs-speedup comparison (Figures 2–4), we further analyze the predictability of future decoding positions (Figure 5) and the stage-wise contributions of parallel and speculative decoding within a block (Figure 6), which together explain how the two axes jointly contribute to the observed speedups.

4.1 Experimental Setup

Models and benchmarks. We evaluate three representative open-source dLLMs that cover distinct training paradigms, including AR-initialized fine-tuning, from-scratch diffusion pretraining with preference alignment, and continual diffusion pretraining. Specifically, our model suite consists of Dream-v0-Base-7B [Ye et al., 2025], LLaDA-1.5 [Zhu et al., 2025a], and openPangu-7B-Diffusion-Base[2]. These models have comparable parameter scales, while differing in their initialization and pretraining strategies, making them suitable for examining whether the observed behaviors are consistent across different dLLM training recipes rather than being tied to a single model family. We conduct experiments on three downstream benchmarks that cover mathematical reasoning and code generation. For mathematical reasoning, we use GSM8K [Cobbe et al., 2021], a benchmark of grade-school math word problems requiring multi-step arithmetic reasoning, and report accuracy

[2] `https://ai.gitcode.com/ascend-tribe/openPangu-7B-Diffusion-Base`

under chain-of-thought prompting. For code generation, we evaluate on HumanEval [Chen et al., 2021] and MBPP [Austin et al., 2021], which contain Python programming problems with executable test cases. Further details of the evaluated models and benchmarks are provided in Appendix B.1.

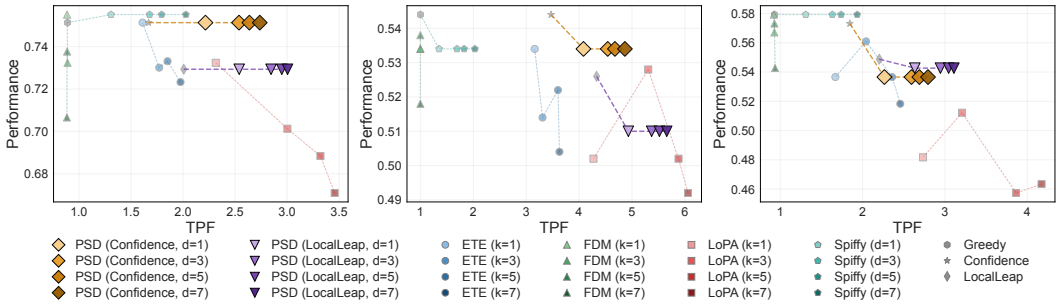

Figure 2: Accuracy vs. speedup on Dream-v0-Base-7B across 27 parameter configurations of different methods. Left: GSM8K; Center: MBPP; Right: HumanEval.

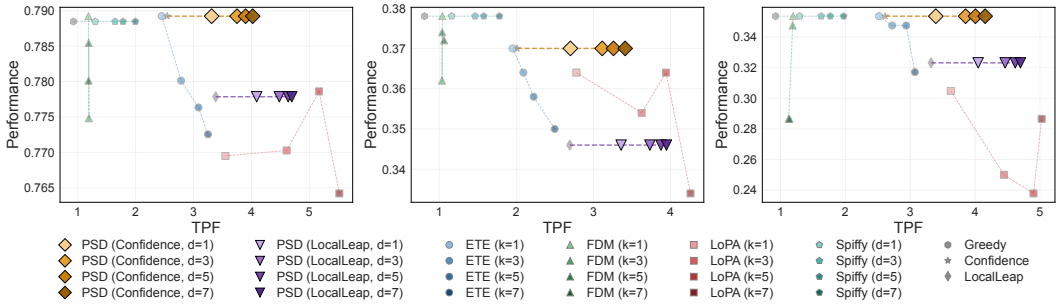

Figure 3: Accuracy vs. speedup on LLaDA 1.5 across 27 parameter configurations of different methods. Left: GSM8K; Center: MBPP; Right: HumanEval.

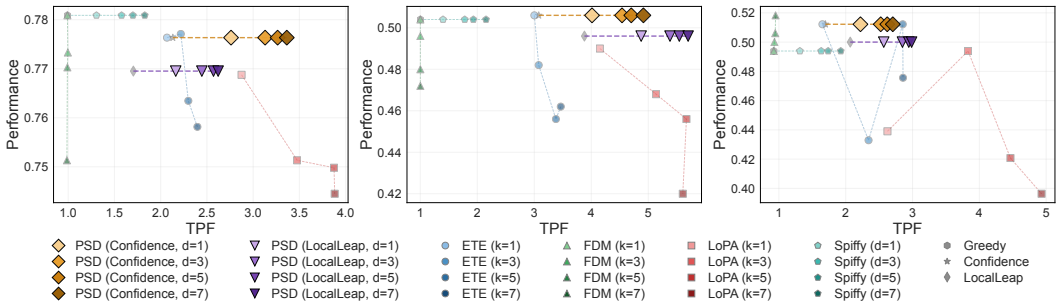

Figure 4: Accuracy vs. speedup on openPangu-7B-Diffusion-Base across 27 parameter configurations of different methods. Left: GSM8K; Center: MBPP; Right: HumanEval.

Baseline settings and metrics. We compare PSD against seven representative decoding baselines, including six *spatial* parallel decoding methods and one *temporal* acceleration method; detailed descriptions of all baselines are deferred to the Appendix C.2. Specifically, the spatial baselines include Greedy Decoding, Confidence Parallel Decoding with threshold τ=0.9 [Wu et al., 2026a, Yu et al., 2025], LocalLeap [Kong et al., 2025], LoPA [Xu et al., 2025], ETE [Fu et al., 2025b], and FDM [Mo et al., 2025], where LoPA, ETE, and FDM each consider a tunable branching or

lookahead parameter $k \in \{1, 3, 5, 7\}$. The temporal baseline is Spiffy [Agrawal et al., 2025], an auto-speculative decoding method with speculative draft depth $d \in \{1, 3, 5, 7\}$. PSD is policy-agnostic and can be combined with arbitrary spatial parallel decoding policies by adding temporal speculation on top. To demonstrate this generality, we instantiate PSD with two representative spatial backbones: PSD (Confidence), which uses Confidence Parallel Decoding with threshold τ=0.9, and PSD (LocalLeap), which uses LocalLeap with its default parameters. For both PSD variants, we sweep the speculative draft depth $d \in \{1, 3, 5, 7\}$. For task performance, we report accuracy (%) on GSM8K and pass@1 (%) on HumanEval and MBPP. For inference efficiency, following Agrawal et al. [2025], we report tokens per forward pass (TPF), defined as the average number of tokens decoded per model invocation.

4.2 Main Results

Figures 2–4 plot each method as a point in the accuracy vs. speedup plane: rightward indicates higher effective parallelism and upward indicates better task quality. Across all three dLLMs and every benchmark, PSD consistently achieves a superior trade-off between quality and parallelism, occupying the upper-right region of the plane. Furthermore, we also provide two additional analyses in Figures 5 and 6: the predictability of future decoding positions, and the stage-wise complementarity of parallel and speculative decoding within a block.

Spatial and temporal acceleration combine to yield compounded speedups. The defining advantage of PSD is that it compounds two independent sources of parallelism, spatial and temporal, into a single decoding pass, achieving speedups that neither axis can deliver alone. The temporal-only baseline, Spiffy, already improves the greedy baseline's $1.0\times$ parallelism to roughly 1.5–$2.0\times$ by collapsing multiple denoising iterations via batched verification. PSD further scales this temporal gain by an additional spatial factor: first unmasking multiple tokens per step along the spatial axis and then speculating over future parallel-unmasking steps along the temporal axis. Concretely, on the GSM8K benchmark with LLaDA 1.5 (Figure 3, left), PSD (Confidence) doubles Spiffy's $2.0\times$ to $4.0\times$ tokens per step, contributing an additional $2.0\times$ gain from the spatial axis alone, while preserving 78.9% accuracy that is virtually identical to greedy decoding (78.8%). On the MBPP benchmark with Dream (Figure 2, center), PSD (Confidence) reaches $4.9\times$ at 53.4% pass@1, compared with Spiffy's $2.0\times$ at the same accuracy level, yielding a $2.5\times$ additional factor. Similarly, on MBPP with openPangu (Figure 4, center), PSD (Confidence) attains $4.9\times$ while maintaining greedy-level accuracy at 50.6%. These results suggest that PSD's two-axis design can offer speedups superior to those achievable by any single-axis method alone.

Speculative verification preserves quality under aggressive parallelism. Spatial-only baselines face a sharp quality cliff: pushing parallelism higher inevitably forces low-confidence tokens to commit prematurely, causing cascading errors. PSD's hierarchical speculative verification fundamentally mitigates this failure mode. For example, on the HumanEval benchmark with LLaDA 1.5 (Figure 3, right), LoPA k=5 in its aggressive mode pushes parallelism to $4.9\times$ but collapses pass@1 to 23.8%, a catastrophic 11.6-point drop from the greedy baseline (35.4%). In stark contrast, PSD (Confidence) achieves a comparable parallelism of $4.2\times$ while retaining 35.4% pass@1 with *zero* degradation. On the GSM8K benchmark with Dream (Figure 2, left), LoPA k=7 in aggressive mode reaches $3.5\times$ but drops accuracy to 67.1%, losing 8.0 points from the greedy baseline (75.1%), whereas PSD (Confidence) at $2.7\times$ maintains the full 75.1% accuracy without quality loss. The same pattern holds on the MBPP benchmark with openPangu (Figure 4, center), where non-speculative methods sacrifice substantial accuracy for marginal additional speedup gains while PSD maintains generation quality at comparable or even higher throughput. Across these settings, PSD generally advances the Pareto frontier of the quality–speedup plane, suggesting that speculative verification is an effective approach for achieving higher parallelism while largely preserving output fidelity.

PSD generalizes across model families and spatial policies. Notably, the advantage of PSD appears largely architecture-agnostic. Across Dream (AR-initialized, 7B), LLaDA 1.5 (from-scratch base, aligned, 8B), and openPangu (continual-pretrained, 7B), PSD generally achieves a favorable quality–speedup trade-off that is competitive with or better than the strongest baselines. Moreover, PSD's modular design exposes a useful speed–quality knob through the choice of spatial policy: PSD (LocalLeap) tends to achieve higher tokens per step than PSD (Confidence) on most benchmarks, while PSD (Confidence) consistently exhibits better accuracy retention (e.g., 75.1% vs. 72.9%

on GSM8K with Dream; 35.4% vs. 32.3% on HumanEval with LLaDA 1.5). This plug-and-play flexibility allows practitioners to select a spatial policy suited to their latency or quality constraints, while the temporal speculation layer tends to provide additional gains on top of the chosen policy.

Code generation is more sensitive to spatial parallelism than mathematical reasoning. A cross-benchmark comparison reveals that mathematical reasoning (GSM8K) is considerably more tolerant of aggressive parallel unmasking than code generation (HumanEval, MBPP). On the GSM8K benchmark with LLaDA 1.5 (Figure 3, left), PSD (Confidence) at d=7 reaches $4.0\times$ TPF while maintaining the full greedy-level accuracy of 78.9%, whereas on the MBPP benchmark with the same model (Figure 3, center) the same configuration achieves $3.4\times$ but incurs a 0.8-point drop. This gap widens dramatically for spatial-only baselines: LoPA k=5 in aggressive mode loses only 2.0 points on GSM8K with LLaDA 1.5 yet collapses by 11.6 points on HumanEval with the same model. The disparity likely stems from tighter inter-token syntactic constraints in code, where a single misplaced delimiter or incorrect identifier propagates fatally through unit-test evaluation, whereas arithmetic reasoning chains can absorb minor token-level perturbations that preserve semantic correctness. Notably, PSD's speculative verification acts as a built-in safety net that helps narrow this sensitivity gap: even on code tasks, PSD (Confidence) tends to keep the accuracy drop small while still offering substantial speedup. For practitioners, this suggests a practical guideline: code generation workloads may benefit from PSD (Confidence) with moderate speculative depth to better preserve quality, while math-heavy workloads can more comfortably push toward higher parallelism.

Speculative depth saturates at d=3 without degrading quality. Examining the marginal gains across increasing speculative depths $d \in \{1, 3, 5, 7\}$ reveals a clear diminishing-returns pattern. On the MBPP benchmark with Dream (Figure 2, center), the non-speculative confidence baseline achieves $3.5\times$ TPF. Adding temporal speculation at d=1 raises this to $4.1\times$, and d=3 further extends it to $4.6\times$. Beyond d=3, however, the curve flattens noticeably, with d=5 and d=7 each contributing only an additional 0.1–$0.2\times$. The three points $d \in \{3, 5, 7\}$ cluster tightly on the TPF axis. This saturation is expected because the probability of all drafted steps passing hierarchical verification decays geometrically as the speculative horizon lengthens, causing the expected accepted length to converge. Importantly, increasing d does not lead to noticeable accuracy degradation: the PSD points remain largely horizontal in the quality dimension, suggesting that the hierarchical acceptance mechanism is effective at filtering incorrect speculative branches across different depths. This property makes PSD reasonably robust to over-speculation; practitioners may consider d=3 as a sensible default that captures much of the achievable speedup with modest overhead.

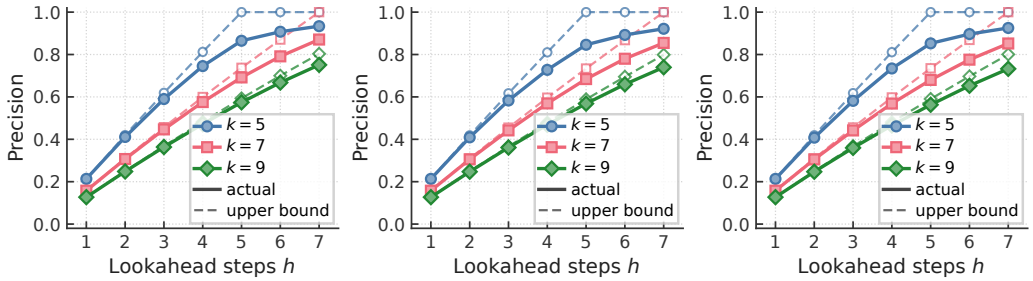

Figure 5: Precision@K of the undecoded candidate positions selected at step t for predicting which positions will be decoded within the next h steps on GSM8K, MBPP, and HumanEval. Colors denote different candidate set sizes ($K \in \{5, 7, 9\}$). Solid lines show empirical precision, and dashed lines show the corresponding oracle upper bound. Notably, this metric demonstrates considerable stability across various experimental settings.

Predictability of Future Decoding Positions. Figure 5 shows that the undecoded candidate positions selected at the current decoding step have substantial predictive value for near-future decoding events across all three benchmarks. At small lookahead horizons, Precision@K is consistently high, indicating that a large fraction of the positions selected at time t are indeed decoded within the next few steps. As the lookahead horizon increases, Precision@K gradually decreases, which is expected because the same fixed candidate set must account for a broader future decoding window. Importantly,

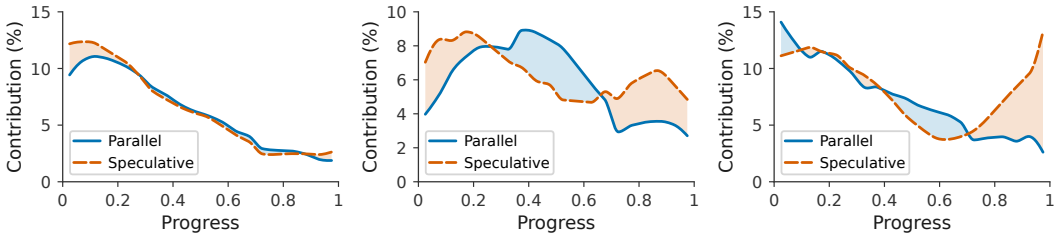

Figure 6: Contribution profiles of parallel decoding and speculative decoding over normalized block progress. The x-axis denotes normalized within-block decoding progress, and the y-axis denotes the percentage of tokens contributed by each mechanism at different progress stages. The three panels correspond to representative blocks from the early, middle, and late phases of generation.

the precision remains non-trivial over multiple steps, suggesting that the current candidate set captures more than purely instantaneous local information. The gap between the empirical curves and the oracle upper bounds further indicates that the selected candidate set already provides a strong signal about near-future decoding, while still leaving room for improved candidate selection strategies. Overall, these results support the hypothesis that the positions highlighted at the current step can be exploited as effective early indicators of where decoding progress is likely to occur next.

Complementary operating regimes across block phases. Across 500 generation trajectories on the GSM8K dataset using LLaDA-1.5, the three panels in Figure 6 reveal that parallel and speculative decoding act as complementary axes whose relative dominance shifts with the block's position in generation. In the early-phase block, the two curves decay in tandem, indicating that when context is still being established neither mechanism is preferentially triggered. The middle-phase block exhibits a three-stage structure: speculative leads at the start, parallel peaks in the mid-progress region where accumulated context yields batches of high-confidence tokens, and speculative resurges near the end as remaining masked positions shrink. The late-phase block accentuates this pattern, with parallel dominating the mid-section but speculative rising sharply at completion, as the confidence ordering stabilizes and deeper draft paths are almost always accepted. This stage-wise complementarity is what enables the spatial and temporal axes to interlock rather than compete under PSD.

Table 1: Open-ended generation on Multi-News. ROUGE-1 and ROUGE-2 measure unigram and bigram overlap between generated and reference summaries, respectively, while ROUGE-L measures overlap based on the longest common subsequence. TPF speedup is measured relative to Greedy per model. Higher is better for all metrics. PSD improves decoding efficiency while largely preserving generation quality, demonstrating its effectiveness beyond reasoning and code generation tasks.

Model	Method	ROUGE-1	ROUGE-2	ROUGE-L	TPF Speedup
Dream	Greedy	31.0	13.1	15.6	1.00×
	Spiffy	31.3	13.2	15.9	2.37×
	PSD (Confidence)	31.3	13.2	15.9	2.53×
	PSD (LocalLeap)	32.9	13.5	16.3	2.61×
openPangu	Greedy	41.0	15.4	18.3	1.00×
	Spiffy	39.2	15.4	17.8	2.05×
	PSD (Confidence)	38.9	15.5	17.8	2.50×
	PSD (LocalLeap)	39.1	16.1	17.7	2.73×

Open-Ended Generation. To examine a less constrained, longer-form generation setting beyond mathematical reasoning and code generation, we additionally evaluate Dream and openPangu on the Multi-News multi-document news-summarization benchmark [Fabbri et al., 2019]. We compare Greedy, Spiffy, PSD (Confidence) with τ=0.9, and PSD (LocalLeap), using speculative depth d=7 for Spiffy and both PSD variants. We report ROUGE-1, ROUGE-2, and ROUGE-L to assess summarization quality, measuring unigram, bigram, and longest common subsequence overlap with

reference summaries, respectively. For efficiency, we report TPF speedup relative to Greedy. Table 1 shows that PSD provides additional acceleration while largely preserving summarization quality. On Dream, PSD (Confidence) matches Spiffy's ROUGE scores at a higher TPF speedup of $2.53\times$. PSD (LocalLeap) further reaches $2.61\times$ and improves all three ROUGE scores over both Greedy and Spiffy. On openPangu, PSD (Confidence) and PSD (LocalLeap) achieve $2.50\times$ and $2.73\times$ TPF speedups, respectively, while maintaining ROUGE scores close to Spiffy's. PSD (LocalLeap) also improves ROUGE-2 from 15.4 to 16.1. Overall, PSD achieves up to $2.73\times$ TPF speedup over Greedy and improves TPF over Spiffy by up to 10.1% on Dream and 33.2% on openPangu. The comparable ROUGE scores indicate that this additional acceleration has little impact on open-ended generation quality, supporting PSD's effectiveness beyond reasoning and code tasks.

5 Related Work

Diffusion Large Language Models (dLLMs). The sequential generation bottleneck of autoregressive LLMs [OpenAI et al., 2024, DeepSeek-AI et al., 2025, Grattafiori et al., 2024, Yang et al., 2025] has driven growing interest in non-autoregressive alternatives capable of decoding multiple tokens simultaneously during the generation process. Within this broader line of research, dLLMs [Nie et al., 2025, Ye et al., 2025, Gong et al., 2026] address this bottleneck by casting text generation as a discrete denoising process. Starting from a fully masked sequence, the model progressively reveals tokens over multiple refinement steps, gradually filling in the sequence as denoising proceeds. Throughout this process, bidirectional attention allows each prediction to condition on the entire sequence context. In terms of model construction, scaling this paradigm to billions of parameters has proven effective across different training settings, including both autoregressive-initialized pipelines [Ye et al., 2025] and from-scratch training pipelines with preference optimization [Zhu et al., 2025a]. The resulting models have been shown to match autoregressive baselines on benchmarks covering both reasoning and code generation [Ma et al., 2025, Wu et al., 2026a].

Inference Acceleration for dLLMs. Existing work on training-free acceleration of dLLM inference has been pursued along two largely independent directions, each approaching decoding efficiency from a different perspective. The first direction focuses on reducing the total number of denoising iterations by unmasking multiple tokens within each decoding step. Representative methods in this direction include confidence-based thresholding [Wu et al., 2026a, Yu et al., 2025], LocalLeap [Kong et al., 2025], and LoPA [Xu et al., 2025]. These methods each improve per-step parallelism, but they face an inherent quality ceiling as the unmasking policy becomes more aggressive. The second direction seeks to reduce per-iteration cost through the use of speculative decoding. This paradigm was originally developed for autoregressive LLMs [Chen et al., 2023, Leviathan et al., 2023, Sun et al., 2026b, Xia et al., 2024, Liu et al., 2026], with subsequent extensions to tree-structured verification [Miao et al., 2023], multi-head prediction [Cai et al., 2024], and feature-level drafting [Li et al., 2024a,b, 2025b]. In the context of diffusion language modeling, Spiffy [Agrawal et al., 2025] adapted this paradigm to dLLMs, though each individual decoding step remains restricted to a single token. Beyond language generation, speculative sampling has also been explored in the setting of continuous image diffusion [Bortoli et al., 2025, Hu et al., 2025]. PSD operates at the intersection of these two directions, bringing spatial and temporal acceleration together within a single framework to jointly accelerate decoding along both axes.

6 Conclusion

We presented PSD, a policy-agnostic framework that composes spatial parallel unmasking with temporal speculative verification to accelerate dLLM inference along both axes simultaneously. The key insight is that confidence rankings over masked positions stabilize across denoising steps, enabling reliable multi-depth draft construction without additional model calls. Because PSD treats the spatial unmasking policy as a modular component, it is compatible with any existing or future parallel decoding method. Experiments across diverse configurations confirm that PSD achieves 3–$5.5\times$ tokens per forward pass while keeping accuracy within 1 point of greedy decoding on most settings. We believe PSD establishes a general recipe for dLLM acceleration: practitioners can plug in any spatial policy and benefit from temporal speculation on top, opening the door to further gains as stronger parallel decoding methods emerge.

Acknowledgement

We thank all our collaborators and the anonymous reviewers for their constructive feedback and valuable suggestions. We also thank our company for providing resources to support this research. The authors declare no conflicts of interest.

References

OpenAI, Aaron Jaech, Adam Kalai, Adam Lerer, Adam Richardson, Ahmed El-Kishky, et al. Openai o1 system card. In *arXiv:2412.16720*, 2024.

Aaron Grattafiori, Abhimanyu Dubey, Abhinav Jauhri, Abhinav Pandey, Abhishek Kadian, et al. The llama 3 herd of models. In *arXiv:2407.21783*, 2024.

Andrei Ivanov, Nikoli Dryden, Tal Ben-Nun, Shigang Li, and Torsten Hoefler. Data movement is all you need: A case study on optimizing transformers. In *Proceedings of Machine Learning and Systems*, 2021.

Weigao Sun, Jiaxi Hu, Yucheng Zhou, Jusen Du, Disen Lan, Kexin Wang, Tong Zhu, Xiaoye Qu, Yu Zhang, Xiaoyu Mo, Daizong Liu, Yuxuan Liang, Wenliang Chen, Guoqi Li, and Yu Cheng. Speed Always Wins: a survey on efficient architectures for large language models. In *arXiv:2508.09834*, 2025a.

Shengyin Sun, Yiming Li, Yingzhao Lian, Xing Li, Xingzhi Zhou, Anxin Tian, Zhili Wang, Haoyang Li, Ziqiang Cui, and Chen Ma. Salvation Lies Within: Eliciting inherent style transfer in step-distilled diffusion models. In *arXiv:2610.05066*, 2026a.

Shen Nie, Fengqi Zhu, Zebin You, Xiaolu Zhang, Jingyang Ou, Jun Hu, Jun Zhou, Yankai Lin, Ji-Rong Wen, and Chongxuan Li. Large language diffusion models. In *arXiv:2502.09992*, 2025.

Jiacheng Ye, Zhihui Xie, Lin Zheng, Jiahui Gao, Zirui Wu, Xin Jiang, Zhenguo Li, and Lingpeng Kong. Dream 7b: Diffusion large language models. In *arXiv:2508.15487*, 2025.

Haokun Lin, Xinle Jia, Shaozhen Liu, Shujun Xia, Weitao Huang, Haobo Xu, Junyang Li, Yicheng Xiao, Xingrun Xing, Ziyu Guo, Renrui Zhang, Qi Li, Yichen Wu, Renzhen Wang, Xiaojuan Qi, Caifeng Shan, Hongsheng Li, and Zhenan Sun. Efficient Diffusion Language Models: A comprehensive survey. In *Authorea:au.176918713.36402137*, 2026.

Yonggan Fu, Lexington Whalen, Zhifan Ye, Xin Dong, Shizhe Diao, Jingyu Liu, Chengyue Wu, Hao Zhang, Enze Xie, Song Han, Maksim Khadkevich, Jan Kautz, Yingyan Celine Lin, and Pavlo Molchanov. Efficient-DLM: from autoregressive to diffusion language models, and beyond in speed. In *arXiv:2512.14067*, 2025a.

Han Peng, Peiyu Liu, Zican Dong, Daixuan Cheng, Junyi Li, Yiru Tang, Shuo Wang, and Wayne Xin Zhao. How efficient are diffusion language models? a critical examination of efficiency evaluation practices. In *arXiv:2510.18480*, 2025.

Fanheng Kong, Jingyuan Zhang, Yahui Liu, Zirui Wu, Yu Tian, Victoria W., and Guorui Zhou. Accelerating diffusion llm inference via local determinism propagation. In *arXiv:2510.07081*, 2025.

Chenkai Xu, Yijie Jin, Jiajun Li, Yi Tu, Guoping Long, Dandan Tu, Mingcong Song, Hongjie Si, Tianqi Hou, Junchi Yan, and Zhijie Deng. LoPA: scaling dllm inference via lookahead parallel decoding. In *arXiv:2512.16229*, 2025.

Chengyue Wu, Hao Zhang, Shuchen Xue, Zhijian Liu, Shizhe Diao, Ligeng Zhu, Ping Luo, Song Han, and Enze Xie. Fast-dLLM: training-free acceleration of diffusion llm by enabling kv cache and parallel decoding. In *Proc. Int. Conf. Learning Representations*, 2026a.

Chengyue Wu, Hao Zhang, Shuchen Xue, Shizhe Diao, Yonggan Fu, Zhijian Liu, Pavlo Molchanov, Ping Luo, Song Han, and Enze Xie. Fast-dLLM v2: Efficient block-diffusion llm. In *Proc. Int. Conf. Learning Representations*, 2026b.

Charlie Chen, Sebastian Borgeaud, Geoffrey Irving, Jean-Baptiste Lespiau, Laurent Sifre, and John Jumper. Accelerating large language model decoding with speculative sampling. In *arXiv:2302.01318*, 2023.

Sudhanshu Agrawal, Risheek Garrepalli, Raghavv Goel, Mingu Lee, Christopher Lott, and Fatih Porikli. Spiffy: multiplying diffusion llm acceleration via lossless speculative decoding. In *arXiv:2509.18085*, 2025.

Fengqi Zhu, Rongzhen Wang, Shen Nie, Xiaolu Zhang, Chunwei Wu, Jun Hu, Jun Zhou, Jianfei Chen, Yankai Lin, Ji-Rong Wen, and Chongxuan Li. Llada 1.5: Variance-reduced preference optimization for large language diffusion models. In *arXiv:2505.19223*, 2025a.

Tianyi Li, Mingda Chen, Bowei Guo, and Zhiqiang Shen. A survey on diffusion language models. In *arXiv:2508.10875*, 2025a.

Shengyin Sun, Wenhao Yu, Yuxiang Ren, Weitao Du, Liwei Liu, Xuecang Zhang, Ying Hu, and Chen Ma. GDiffRetro: Retrosynthesis prediction with dual graph enhanced molecular representation and diffusion generation. In *arXiv:2501.08001*, 2025b.

Tiwei Bie, Maosong Cao, Kun Chen, Lun Du, Mingliang Gong, Zhuochen Gong, Yanmei Gu, Jiaqi Hu, Zenan Huang, Zhenzhong Lan, Chengxi Li, Chongxuan Li, Jianguo Li, Zehuan Li, Huabin Liu, Lin Liu, Guoshan Lu, Xiaocheng Lu, Yuxin Ma, Jianfeng Tan, Lanning Wei, Ji-Rong Wen, Yipeng Xing, Xiaolu Zhang, Junbo Zhao, Da Zheng, Jun Zhou, Junlin Zhou, Zhanchao Zhou, Liwang Zhu, and Yihong Zhuang. Llada2.0: Scaling up diffusion language models to 100b. In *arXiv:2512.15745*, 2025.

Yuxin Ma, Lun Du, Lanning Wei, Kun Chen, Qian Xu, Kangyu Wang, Guofeng Feng, Guoshan Lu, Lin Liu, Xiaojing Qi, Xinyuan Zhang, Zhen Tao, Haibo Feng, Ziyun Jiang, Ying Xu, Zenan Huang, Yihong Zhuang, Haokai Xu, Jiaqi Hu, Zhenzhong Lan, Junbo Zhao, Jianguo Li, and Da Zheng. dInfer: an efficient inference framework for diffusion language models. In *arXiv:2510.08666*, 2025.

Fengqi Zhu, Zebin You, Yipeng Xing, Zenan Huang, Lin Liu, Yihong Zhuang, Guoshan Lu, Kangyu Wang, Xudong Wang, Lanning Wei, Hongrui Guo, Jiaqi Hu, Wentao Ye, Tieyuan Chen, Chenchen Li, Chengfu Tang, Haibo Feng, Jun Hu, Jun Zhou, Xiaolu Zhang, Zhenzhong Lan, Junbo Zhao, Da Zheng, Chongxuan Li, Jianguo Li, and Ji-Rong Wen. LLaDA-MoE: a sparse moe diffusion language model. In *arXiv:2509.24389*, 2025b.

Yaniv Leviathan, Matan Kalman, and Yossi Matias. Fast inference from transformers via speculative decoding. In *Proc. Int. Conf. Machine Learning*, 2023.

Shengyin Sun, Yiming Li, Xing Li, Yingzhao Lian, Weizhe Lin, Hui-Ling Zhen, Zhiyuan Yang, Chen Chen, Xianzhi Yu, Mingxuan Yuan, and Chen Ma. Scaling up, speeding up: A benchmark of speculative decoding for efficient LLM test-time scaling. In *Proc. Int. Conf. Learning Representations*, 2026b.

Heming Xia, Zhe Yang, Qingxiu Dong, Peiyi Wang, Yongqi Li, Tao Ge, Tianyu Liu, Wenjie Li, and Zhifang Sui. Unlocking efficiency in large language model inference: A comprehensive survey of speculative decoding. In *ACL*, 2024.

Shengyin Sun, Yiming Li, Renxi Liu, Weizhe Lin, Hui-Ling Zhen, Xianzhi Yu, Mingxuan Yuan, and Chen Ma. JuDi: Revisiting judge decoding from first principles via training-free distributional divergence. In *arXiv:2601.04766*, 2026c.

Zhenyu He, Zexuan Zhong, Tianle Cai, Jason Lee, and Di He. REST: retrieval-based speculative decoding. In *NAACL*, 2024.

Xianzhen Luo, Yixuan Wang, Qingfu Zhu, Zhiming Zhang, Xuanyu Zhang, Qing Yang, and Dongliang Xu. Turning trash into treasure: Accelerating inference of large language models with token recycling. In *ACL*, 2025.

Gabriele Oliaro, Zhihao Jia, Daniel Campos, and Aurick Qiao. Suffixdecoding: Extreme speculative decoding for emerging ai applications. In *arXiv:2411.04975*, 2024.

Talor Abramovich, Maor Ashkenazi, Carl Putterman, Benjamin Chislett, Tiyasa Mitra, Bita Darvish Rouhani, Ran Zilberstein, and Yonatan Geifman. SPEED-Bench: A unified and diverse benchmark for speculative decoding. In *arXiv:2604.09557*, 2026.

Shenggui Li, Chao Wang, Yikai Zhu, Yubo Wang, Fan Yin, Shuai Shi, Yefei Chen, Xiaomin Dong, Qiaoling Chen, Jin Pan, Ji Li, Laixin Xie, Yineng Zhang, Lei Yu, Yonggang Wen, Ivor Tsang, and Tianwei Zhang. SpecForge: A flexible and efficient open-source training framework for speculative decoding. In *arXiv:2603.18567*, 2026a.

Yifeng Gao, Ziang Ji, Yuxuan Wang, Biqing Qi, Hanlin Xu, and Linfeng Zhang. Self speculative decoding for diffusion large language models. In *arXiv:2510.04147*, 2025.

Pengxiang Li, Dilxat Muhtar, Tianlong Chen, Lu Yin, and Shiwei Liu. Why diffusion language models struggle with truly parallel (non-autoregressive) decoding? In *arXiv:2602.23225*, 2026b.

Runpeng Yu, Xinyin Ma, and Xinchao Wang. Dimple: Discrete diffusion multimodal large language model with parallel decoding. In *arXiv:2505.16990*, 2025.

Yangyang Zhong, Yanmei Gu, Zhengqing Zang, Xiaomeng Li, Yuqi Ding, Xibei Jia, Yuting Shen, Zhenzhong Lan, Liwang Zhu, Weiping Liu, Junlin Zhou, Haisheng Liu, Zhong Xin Yu, Pengxin Luo, Donglian Qi, Yunfeng Yan, and Junbo Zhao. Parallelism and generation order in masked diffusion language models: Limits today, potential tomorrow. In *arXiv:2601.15593*, 2026.

Karl Cobbe, Vineet Kosaraju, Mohammad Bavarian, Mark Chen, Heewoo Jun, Lukasz Kaiser, Matthias Plappert, Jerry Tworek, Jacob Hilton, Reiichiro Nakano, Christopher Hesse, and John Schulman. Training verifiers to solve math word problems. In *arXiv:2110.14168*, 2021.

Mark Chen, Jerry Tworek, Heewoo Jun, Qiming Yuan, Henrique Ponde de Oliveira Pinto, Jared Kaplan, Harri Edwards, Yuri Burda, Nicholas Joseph, Greg Brockman, Alex Ray, Raul Puri, Gretchen Krueger, Michael Petrov, Heidy Khlaaf, Girish Sastry, Pamela Mishkin, Brooke Chan, Scott Gray, Nick Ryder, Mikhail Pavlov, Alethea Power, Lukasz Kaiser, Mohammad Bavarian, Clemens Winter, Philippe Tillet, Felipe Petroski Such, Dave Cummings, Matthias Plappert, Fotios Chantzis, Elizabeth Barnes, Ariel Herbert-Voss, William Hebgen Guss, Alex Nichol, Alex Paino, Nikolas Tezak, Jie Tang, Igor Babuschkin, Suchir Balaji, Shantanu Jain, William Saunders, Christopher Hesse, Andrew N. Carr, Jan Leike, Josh Achiam, Vedant Misra, Evan Morikawa, Alec Radford, Matthew Knight, Miles Brundage, Mira Murati, Katie Mayer, Peter Welinder, Bob McGrew, Dario Amodei, Sam McCandlish, Ilya Sutskever, and Wojciech Zaremba. Evaluating large language models trained on code. In *arXiv:2107.03374*, 2021.

Jacob Austin, Augustus Odena, Maxwell Nye, Maarten Bosma, Henryk Michalewski, David Dohan, Ellen Jiang, Carrie Cai, Michael Terry, Quoc Le, and Charles Sutton. Program synthesis with large language models. In *arXiv:2108.07732*, 2021.

Hengyu Fu, Baihe Huang, Virginia Adams, Charles Wang, Venkat Srinivasan, and Jiantao Jiao. From bits to rounds: Parallel decoding with exploration for diffusion language models. In *arXiv:2511.21103*, 2025b.

Yichuan Mo, Quan Chen, Mingjie Li, Zeming Wei, and Yisen Wang. Decoding large language diffusion models with foreseeing movement. *CoRR*, abs/2512.04135, 2025.

Alexander R. Fabbri, Irene Li, Tianwei She, Suyi Li, and Dragomir R. Radev. Multi-News: A large-scale multi-document summarization dataset and abstractive hierarchical model. In *ACL*, 2019.

DeepSeek-AI, Daya Guo, Dejian Yang, Haowei Zhang, Junxiao Song, Ruoyu Zhang, et al. Deepseek-r1: Incentivizing reasoning capability in LLMs via reinforcement learning. In *arXiv:2501.12948*, 2025.

An Yang, Anfeng Li, Baosong Yang, Beichen Zhang, Binyuan Hui, et al. Qwen3 technical report. In *arXiv:2505.09388*, 2025.

Shansan Gong, Ruixiang Zhang, Huangjie Zheng, Jiatao Gu, Navdeep Jaitly, Lingpeng Kong, and Yizhe Zhang. DiffuCoder: understanding and improving masked diffusion models for code generation. In *Proc. Int. Conf. Learning Representations*, 2026.

Renxi Liu, Yiming Li, Shengyin Sun, Chen Chen, Xian Wang, Shuai Yang, chen xi, Wenbo GAO, Xiaoguang Li, Hui-Ling Zhen, Xianzhi Yu, Weizhe Lin, Xiaosong Li, and Yaoyuan Wang. PV-Decoding: Progressive verification decoding with decode gating for diffusion language models. In *EMNLP*, 2026.

Xupeng Miao, Gabriele Oliaro, Zhihao Zhang, Xinhao Cheng, Zeyu Wang, Zhengxin Zhang, Rae Ying Yee Wong, Alan Zhu, Lijie Yang, Xiaoxiang Shi, Chunan Shi, Zhuoming Chen, Daiyaan Arfeen, Reyna Abhyankar, and Zhihao Jia. Specinfer: Accelerating generative large language model serving with tree-based speculative inference and verification. In *arXiv:2305.09781*, 2023.

Tianle Cai, Yuhong Li, Zhengyang Geng, Hongwu Peng, Jason D. Lee, Deming Chen, and Tri Dao. Medusa: Simple LLM inference acceleration framework with multiple decoding heads. In *arXiv:2401.10774*, 2024.

Yuhui Li, Fangyun Wei, Chao Zhang, and Hongyang Zhang. Eagle: Speculative sampling requires rethinking feature uncertainty. In *Proc. Int. Conf. Machine Learning*, 2024a.

Yuhui Li, Fangyun Wei, Chao Zhang, and Hongyang Zhang. Eagle-2: Faster inference of language models with dynamic draft trees. In *Proc. Conf. Empirical Methods in Natural Language Processing*, 2024b.

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

Valentin De Bortoli, Alexandre Galashov, Arthur Gretton, and Arnaud Doucet. Accelerated diffusion models via speculative sampling. In *Proc. Int. Conf. Machine Learning*, 2025.

Hengyuan Hu, Aniket Das, Dorsa Sadigh, and Nima Anari. Diffusion models are secretly exchangeable: Parallelizing ddpms via auto speculation. In *Proc. Int. Conf. Machine Learning*, 2025.

An Yang, Baosong Yang, Beichen Zhang, Binyuan Hui, Bo Zheng, Bowen Yu, et al. Qwen2.5 technical report. In *arXiv:2412.15115*, 2024.

A Pseudocode of Algorithm

Algorithm 1 Parallel Speculative Decoding (PSD)

Require: dLLM p_θ; transfer policy π with acceptance criterion $\mathcal{A}_\pi$; draft DAG $\mathcal{G} = (\mathcal{N}, \mathcal{E})$ with root r and node subsets $\{\mathcal{S}_k\}_{k\in\mathcal{N}}$; block length L
Ensure: Fully denoised block $\mathbf{x}^\star$

Initialize $\mathbf{x}^{(0)} \leftarrow [\texttt{[M]}]^L$; $t \leftarrow 0$
while $\mathcal{M}(\mathbf{x}^{(t)}) \neq \varnothing$ **do**
  // Stage 1: Spatial parallel unmasking
  $\{(\hat{x}_i^{(t)}, c_i^{(t)})\}_{i\in\mathcal{M}(\mathbf{x}^{(t)})} \leftarrow p_\theta(\mathbf{x}^{(t)})$ ▷ *single forward pass*
  $\mathcal{T}^{(t)} \leftarrow \pi\big(\mathbf{x}^{(t)}, \{c_i^{(t)}\}\big)$ ▷ *positions to commit*
  $x_i^{(t+1)} \leftarrow \hat{x}_i^{(t)}$ for $i \in \mathcal{T}^{(t)}$; $x_i^{(t+1)} \leftarrow x_i^{(t)}$ otherwise
  if $\mathcal{M}(\mathbf{x}^{(t+1)}) = \varnothing$ **then**
    $t \leftarrow t+1$; **break**
  end if
  // Stage 2: Temporal speculative drafting (no extra model call)
  Sort residual masked positions by confidence into $\boldsymbol{\sigma} = (\sigma_1, \ldots, \sigma_m)$, $c_{\sigma_1}^{(t)} \geq \cdots \geq c_{\sigma_m}^{(t)}$.
  for all $k \in \mathcal{N}$ **do**
    $\tilde{x}_{k,i} \leftarrow \hat{x}_i^{(t)}$ for $i \in \mathcal{S}_k$; $\tilde{x}_{k,i} \leftarrow x_i^{(t+1)}$ otherwise ▷ *index-level assembly*
  end for
  // Stage 3: Batched verification with hierarchical acceptance
  $\{(\hat{x}_{k,i}, c_{k,i})\} \leftarrow p_\theta\big(\{\tilde{\mathbf{x}}_k\}_{k\in\mathcal{N}}\big)$ ▷ *one batched forward pass*
  $\mathcal{A} \leftarrow \{r\}$ ▷ *root carries no speculation and is always accepted*
  for all $k \in \mathcal{N} \setminus \{r\}$ in topological order **do**
    if $\exists\, p \in \mathcal{A}$ with $(p,k) \in \mathcal{E}$ s.t. $\mathcal{A}_\pi\big(\tilde{x}_{k,i}, \hat{x}_{p,i}, c_{p,i}\big) = \texttt{true}\ \forall i \in \mathcal{S}_k \setminus \mathcal{S}_p$ **then**
      $\mathcal{A} \leftarrow \mathcal{A} \cup \{k\}$ ▷ *consistent with at least one accepted parent*
    end if
  end for
  $k^\star \leftarrow \arg\max_{k\in\mathcal{A}} |\mathcal{S}_k|$ ▷ *deepest accepted draft*
  $\mathbf{x}^{(t+1)} \leftarrow \tilde{\mathbf{x}}_{k^\star}$; additionally commit verifier predictions $\hat{x}_{k^\star,i}$ at positions passing $\mathcal{A}_\pi$
  $t \leftarrow t+1$
end while
return $\mathbf{x}^\star \leftarrow \mathbf{x}^{(t)}$

B Additional Experimental Setup

B.1 Additional Details on Models and Benchmarks

Models. We evaluate three open-source dLLMs that are comparable in scale but differ substantially in how diffusion modeling is introduced during training. This selection allows us to examine whether the empirical observations hold across different dLLM construction pipelines, rather than being specific to a single model family or training recipe:

- **Dream-v0-Base-7B**[3] [Ye et al., 2025]: a 7B masked diffusion model initialized from Qwen2.5-7B [Yang et al., 2024] and further pretrained on approximately 580B tokens. We include it as a representative of the *AR-initialized fine-tuning* paradigm, where a pretrained autoregressive backbone is adapted into a diffusion-style language model.
- **LLaDA-1.5**[4] [Zhu et al., 2025a]: an 8B diffusion model pretrained from scratch on 2.3T tokens and further aligned via Variance-Reduced Preference Optimization. We include it as a representative of *from-scratch diffusion pretraining with preference alignment*, where the model is trained natively as a diffusion language model rather than converted from an autoregressive checkpoint.

[3] `https://huggingface.co/Dream-org/Dream-v0-Base-7B`
[4] `https://huggingface.co/GSAI-ML/LLaDA-1.5`

- **openPangu-7B-Diffusion-Base**[5]: a 7B context-causal block diffusion model initialized from openPangu-Embedded-7B and continually pretrained on 700B tokens. We include it as a representative of *continual diffusion pretraining*. The model uses context-causal block diffusion, where tokens inside the current block are decoded with full attention while previous context is handled with causal attention.

Together, these models cover three practically important routes to building dLLMs: adapting an autoregressive model, training a diffusion model from scratch, and continually pretraining an existing model with a block-diffusion objective. This diversity is important for our evaluation because differences in decoding behavior may arise from either the diffusion formulation itself or from the underlying training procedure.

Benchmarks. We evaluate on three downstream benchmarks that test two capabilities where generation quality can be judged with relatively objective criteria: mathematical reasoning and code generation. For mathematical reasoning, we use GSM8K; for code generation, we use HumanEval and MBPP:

- **GSM8K** [Cobbe et al., 2021]: a benchmark of 1,319 grade-school math word problems requiring multi-step arithmetic reasoning. We use chain-of-thought prompting and report accuracy. GSM8K is included to evaluate whether dLLMs can maintain coherent intermediate reasoning steps before producing the final answer.
- **HumanEval** [Chen et al., 2021]: a set of 164 hand-crafted Python programming problems, where generated solutions are evaluated by executing hidden unit tests. We report pass@1. HumanEval is included because it requires models to synthesize complete functions that satisfy precise functional specifications.
- **MBPP** [Austin et al., 2021]: a benchmark of 500 crowd-sourced Python programming tasks with natural language descriptions and test cases. We report pass@1. Compared with HumanEval, MBPP provides a complementary code-generation setting with shorter and more diverse programming tasks.

These benchmarks are suitable for our study because they provide objective and automatically verifiable evaluation signals across reasoning and code-generation tasks. Following the setting in Fast-dLLM [Wu et al., 2026a], we use 5-shot prompting for GSM8K, 0-shot prompting for HumanEval, and 3-shot prompting for MBPP. For evaluation, we report flexible-match accuracy on GSM8K to account for minor formatting variations in numerical answers, and pass@1 on HumanEval and MBPP based on unit-test execution.

C Additional Implementation Details and Baselines

C.1 Implementation Details

All experiments are conducted on a single node equipped with $8\times$ GPUs. We implement our method on top of PyTorch and the HuggingFace Transformers library, and run all baselines within the same codebase and evaluation harness to eliminate discrepancies introduced by differing implementations. Unless otherwise specified, we use a semi-autoregressive block size of 32 and a maximum generation length of 512 tokens for all methods. Within each block, tokens are iteratively unmasked according to the policy of the corresponding method, and the next block is only initialized after the current block is fully resolved. All methods share the same tokenizer, prompt templates, and random seeds for a given task so that differences in measured throughput and accuracy are attributable solely to the decoding algorithm. To ensure a fair efficiency comparison, we further enable EOS-based early stopping for every baseline as well as for our method: decoding terminates as soon as an `<eos>` token is produced. Because parallel decoding methods may unmask tokens *after* the EOS position within the same forward pass, naively counting all emitted tokens would artificially inflate throughput. We therefore compute both the speedup and the tokens-per-forward-pass (TPF) statistics *only over tokens generated up to and including the EOS position*, discarding any tokens unmasked beyond it; the same truncation rule is applied consistently to all methods.

[5] `https://ai.gitcode.com/ascend-tribe/openPangu-7B-Diffusion-Base`

C.2 Detailed Baseline Descriptions

We provide the full descriptions of the seven baselines used in the main paper. All baselines are reimplemented or adapted to use the same backbone model, tokenizer, and block-wise decoding skeleton, so that they differ only in the token-selection rule (spatial axis) or the iteration-compression rule (temporal axis). The first six baselines accelerate decoding along the *spatial* axis by unmasking multiple positions within a single forward pass:

- **Greedy Decoding**: at each forward pass, unmasks exactly one position—the token with the highest predicted confidence across all currently masked positions in the active block. This method does not exploit any parallelism and therefore serves as the *quality reference*: any deviation of other methods from Greedy's outputs reflects the quality cost of parallel decoding.
- **Confidence Parallel Decoding** (τ=0.9) [Wu et al., 2026a, Yu et al., 2025]: at each forward pass, unmasks *all* positions whose predicted top-1 confidence exceeds a global threshold τ. A higher τ yields more conservative (quality-preserving) decoding, while a lower τ unmasks more tokens per step. Following prior work, we fix τ=0.9, which provides a strong quality–efficiency operating point.
- **LocalLeap** [Kong et al., 2025]: exploits *local determinism propagation*, the empirical observation that tokens neighboring a high-confidence anchor tend to become highly deterministic as well. LocalLeap identifies such anchors and unmasks a local window around each anchor jointly, producing localized parallel decoding without relying on a global confidence threshold. We use the default hyperparameters from the original paper.
- **LoPA** [Xu et al., 2025]: explores k candidate *Token Filling Orders* at each step and selects the order whose resulting intermediate state has the highest estimated future parallelism potential, thereby decoupling the next unmasking decision from greedy per-token confidence. Here k controls the number of candidate filling orders that are scored at each step; larger k trades extra lookahead computation for better long-horizon parallelism. We sweep $k \in \{1, 3, 5, 7\}$.
- **ETE** [Fu et al., 2025b]: *strategically explores* high-uncertainty tokens by leveraging *cross-block parallelism*, allowing tokens from adjacent blocks to be unmasked together when their predictions are mutually informative. Here k controls the cross-block exploration width, i.e., how many blocks are jointly considered when selecting positions to unmask in a single forward pass; larger k increases potential parallelism at the cost of greater interference between block predictions. We sweep $k \in \{1, 3, 5, 7\}$.
- **FDM** [Mo et al., 2025]: simultaneously predicts the current and future token distributions within a single forward pass, so that tokens at different denoising horizons can be unmasked in parallel. Here k is the number of future steps whose distributions are jointly predicted per forward pass; larger k increases the potential per-step yield but also amplifies prediction drift for distant future tokens. We sweep $k \in \{1, 3, 5, 7\}$.

The seventh baseline instead accelerates decoding along the *temporal* axis by compressing the iteration count rather than increasing per-iteration throughput:

- **Spiffy** [Agrawal et al., 2025]: a *lossless auto-speculative* method that drafts multiple denoising iterations ahead and then verifies them against the target model in a single pass, preserving the exact output distribution of the underlying diffusion decoder. Here d denotes the *speculative draft depth*, i.e., the number of consecutive denoising iterations that are speculatively drafted before each verification step; larger d yields greater potential speedup when drafts are accepted but increases the penalty of rejection. We sweep $d \in \{1, 3, 5, 7\}$.

C.3 Our Method Configurations

PSD is intentionally designed to be *policy-agnostic*: it treats the spatial parallel decoding rule as a plug-in module and layers temporal speculation on top, so that advances along either axis can be incorporated into PSD without modification. To demonstrate this generality, we instantiate two variants with complementary spatial backbones:

- **PSD (Confidence)**: uses Confidence Parallel Decoding with threshold τ=0.9 as its spatial backbone. This variant represents a *global, confidence-driven* spatial policy that aggressively unmasks any position the model is already certain about.

- **PSD (LocalLeap)**: uses LocalLeap with its default parameters as its spatial backbone. This variant represents a *local, structure-driven* spatial policy that exploits determinism propagation around anchors, and does not rely on a fixed global confidence threshold.

Both variants share the same temporal speculation module and sweep the speculative draft depth $d \in \{1, 3, 5, 7\}$, where d carries the same semantics as in Spiffy (the number of speculatively drafted iterations per verification step). All other hyperparameters (block size, maximum generation length, tokenizer, and random seeds) are identical to those used for the baselines, so that the reported improvements are attributable purely to PSD's combination of spatial and temporal acceleration.

D Limitations

While PSD demonstrates strong empirical performance and establishes a favorable quality-efficiency Pareto frontier, our study has certain boundaries that present natural avenues for future exploration. Our empirical analysis is heavily anchored in rigorous mathematical reasoning and code generation benchmarks, which were chosen deliberately for their objective and automatically verifiable correctness. Consequently, evaluating the impact of aggressive spatial-temporal parallel decoding on highly subjective, open-ended generation tasks such as creative writing or open-ended dialogue remains an area for future investigation. Furthermore, although we demonstrate that default configurations (e.g., a moderate speculative depth) generalize well and capture the majority of the achievable algorithmic speedup across our tested scenarios, deploying PSD in highly specialized or out-of-distribution domains might benefit from further domain-specific hyperparameter tuning to achieve the absolute optimal trade-off between parallel throughput and generation fidelity.

NeurIPS Paper Checklist

1. **Claims**

   Question: Do the main claims made in the abstract and introduction accurately reflect the paper's contributions and scope?

   Answer: [Yes]

   Justification: The claims made in the abstract and introduction are verified in Section 4.

   Guidelines:

   - The answer [N/A] means that the abstract and introduction do not include the claims made in the paper.
   - The abstract and/or introduction should clearly state the claims made, including the contributions made in the paper and important assumptions and limitations. A [No] or [N/A] answer to this question will not be perceived well by the reviewers.
   - The claims made should match theoretical and experimental results, and reflect how much the results can be expected to generalize to other settings.
   - It is fine to include aspirational goals as motivation as long as it is clear that these goals are not attained by the paper.

2. **Limitations**

   Question: Does the paper discuss the limitations of the work performed by the authors?

   Answer: [Yes]

   Justification: We mention our limitations in Section 4 and Appendix D.

   Guidelines:

   - The answer [N/A] means that the paper has no limitation while the answer [No] means that the paper has limitations, but those are not discussed in the paper.
   - The authors are encouraged to create a separate "Limitations" section in their paper.
   - The paper should point out any strong assumptions and how robust the results are to violations of these assumptions (e.g., independence assumptions, noiseless settings, model well-specification, asymptotic approximations only holding locally). The authors should reflect on how these assumptions might be violated in practice and what the implications would be.
   - The authors should reflect on the scope of the claims made, e.g., if the approach was only tested on a few datasets or with a few runs. In general, empirical results often depend on implicit assumptions, which should be articulated.
   - The authors should reflect on the factors that influence the performance of the approach. For example, a facial recognition algorithm may perform poorly when image resolution is low or images are taken in low lighting. Or a speech-to-text system might not be used reliably to provide closed captions for online lectures because it fails to handle technical jargon.
   - The authors should discuss the computational efficiency of the proposed algorithms and how they scale with dataset size.
   - If applicable, the authors should discuss possible limitations of their approach to address problems of privacy and fairness.
   - While the authors might fear that complete honesty about limitations might be used by reviewers as grounds for rejection, a worse outcome might be that reviewers discover limitations that aren't acknowledged in the paper. The authors should use their best judgment and recognize that individual actions in favor of transparency play an important role in developing norms that preserve the integrity of the community. Reviewers will be specifically instructed to not penalize honesty concerning limitations.

3. **Theory assumptions and proofs**

   Question: For each theoretical result, does the paper provide the full set of assumptions and a complete (and correct) proof?

   Answer: [N/A]

Justification: [N/A]

Guidelines:

- The answer [N/A] means that the paper does not include theoretical results.
- All the theorems, formulas, and proofs in the paper should be numbered and cross-referenced.
- All assumptions should be clearly stated or referenced in the statement of any theorems.
- The proofs can either appear in the main paper or the supplemental material, but if they appear in the supplemental material, the authors are encouraged to provide a short proof sketch to provide intuition.
- Inversely, any informal proof provided in the core of the paper should be complemented by formal proofs provided in appendix or supplemental material.
- Theorems and Lemmas that the proof relies upon should be properly referenced.

4. **Experimental result reproducibility**

   Question: Does the paper fully disclose all the information needed to reproduce the main experimental results of the paper to the extent that it affects the main claims and/or conclusions of the paper (regardless of whether the code and data are provided or not)?

   Answer: [Yes]

   Justification: We provide all information needed for reproduction in Section C.1.

   Guidelines:

   - The answer [N/A] means that the paper does not include experiments.
   - If the paper includes experiments, a [No] answer to this question will not be perceived well by the reviewers: Making the paper reproducible is important, regardless of whether the code and data are provided or not.
   - If the contribution is a dataset and/or model, the authors should describe the steps taken to make their results reproducible or verifiable.
   - Depending on the contribution, reproducibility can be accomplished in various ways. For example, if the contribution is a novel architecture, describing the architecture fully might suffice, or if the contribution is a specific model and empirical evaluation, it may be necessary to either make it possible for others to replicate the model with the same dataset, or provide access to the model. In general. releasing code and data is often one good way to accomplish this, but reproducibility can also be provided via detailed instructions for how to replicate the results, access to a hosted model (e.g., in the case of a large language model), releasing of a model checkpoint, or other means that are appropriate to the research performed.
   - While NeurIPS does not require releasing code, the conference does require all submissions to provide some reasonable avenue for reproducibility, which may depend on the nature of the contribution. For example
     (a) If the contribution is primarily a new algorithm, the paper should make it clear how to reproduce that algorithm.
     (b) If the contribution is primarily a new model architecture, the paper should describe the architecture clearly and fully.
     (c) If the contribution is a new model (e.g., a large language model), then there should either be a way to access this model for reproducing the results or a way to reproduce the model (e.g., with an open-source dataset or instructions for how to construct the dataset).
     (d) We recognize that reproducibility may be tricky in some cases, in which case authors are welcome to describe the particular way they provide for reproducibility. In the case of closed-source models, it may be that access to the model is limited in some way (e.g., to registered users), but it should be possible for other researchers to have some path to reproducing or verifying the results.

5. **Open access to data and code**

   Question: Does the paper provide open access to the data and code, with sufficient instructions to faithfully reproduce the main experimental results, as described in supplemental material?

Answer: [Yes]

Justification: The benchmarks used and the three dLLMs are all publicly available, with URLs provided in the paper. We provide all information needed for reproduction and the algorithmic details in Section C.1 and A. We provide data, code, and instructions for the final paper.

Guidelines:

- The answer [N/A] means that paper does not include experiments requiring code.
- Please see the NeurIPS code and data submission guidelines (`https://neurips.cc/public/guides/CodeSubmissionPolicy`) for more details.
- While we encourage the release of code and data, we understand that this might not be possible, so [No] is an acceptable answer. Papers cannot be rejected simply for not including code, unless this is central to the contribution (e.g., for a new open-source benchmark).
- The instructions should contain the exact command and environment needed to run to reproduce the results. See the NeurIPS code and data submission guidelines (`https://neurips.cc/public/guides/CodeSubmissionPolicy`) for more details.
- The authors should provide instructions on data access and preparation, including how to access the raw data, preprocessed data, intermediate data, and generated data, etc.
- The authors should provide scripts to reproduce all experimental results for the new proposed method and baselines. If only a subset of experiments are reproducible, they should state which ones are omitted from the script and why.
- At submission time, to preserve anonymity, the authors should release anonymized versions (if applicable).
- Providing as much information as possible in supplemental material (appended to the paper) is recommended, but including URLs to data and code is permitted.

6. **Experimental setting/details**

   Question: Does the paper specify all the training and test details (e.g., data splits, hyperparameters, how they were chosen, type of optimizer) necessary to understand the results?

   Answer: [Yes]

   Justification: We provide all information about the experimental setting in Section C.1.

   Guidelines:

   - The answer [N/A] means that the paper does not include experiments.
   - The experimental setting should be presented in the core of the paper to a level of detail that is necessary to appreciate the results and make sense of them.
   - The full details can be provided either with the code, in appendix, or as supplemental material.

7. **Experiment statistical significance**

   Question: Does the paper report error bars suitably and correctly defined or other appropriate information about the statistical significance of the experiments?

   Answer: [Yes]

   Justification: All reported TPF and accuracy / pass@1 values are averages over the full test set of each benchmark.

   Guidelines:

   - The answer [N/A] means that the paper does not include experiments.
   - The authors should answer [Yes] if the results are accompanied by error bars, confidence intervals, or statistical significance tests, at least for the experiments that support the main claims of the paper.
   - The factors of variability that the error bars are capturing should be clearly stated (for example, train/test split, initialization, random drawing of some parameter, or overall run with given experimental conditions).
   - The method for calculating the error bars should be explained (closed form formula, call to a library function, bootstrap, etc.)

- The assumptions made should be given (e.g., Normally distributed errors).
- It should be clear whether the error bar is the standard deviation or the standard error of the mean.
- It is OK to report 1-sigma error bars, but one should state it. The authors should preferably report a 2-sigma error bar than state that they have a 96% CI, if the hypothesis of Normality of errors is not verified.
- For asymmetric distributions, the authors should be careful not to show in tables or figures symmetric error bars that would yield results that are out of range (e.g., negative error rates).
- If error bars are reported in tables or plots, the authors should explain in the text how they were calculated and reference the corresponding figures or tables in the text.

8. **Experiments compute resources**

   Question: For each experiment, does the paper provide sufficient information on the computer resources (type of compute workers, memory, time of execution) needed to reproduce the experiments?

   Answer: [Yes]

   Justification: We provide all the information in Section C.1.

   Guidelines:

   - The answer [N/A] means that the paper does not include experiments.
   - The paper should indicate the type of compute workers CPU or GPU, internal cluster, or cloud provider, including relevant memory and storage.
   - The paper should provide the amount of compute required for each of the individual experimental runs as well as estimate the total compute.
   - The paper should disclose whether the full research project required more compute than the experiments reported in the paper (e.g., preliminary or failed experiments that didn't make it into the paper).

9. **Code of ethics**

   Question: Does the research conducted in the paper conform, in every respect, with the NeurIPS Code of Ethics `https://neurips.cc/public/EthicsGuidelines`?

   Answer: [Yes]

   Justification: We have read and understood the code of ethics and have made every effort to adhere to it.

   Guidelines:

   - The answer [N/A] means that the authors have not reviewed the NeurIPS Code of Ethics.
   - If the authors answer [No], they should explain the special circumstances that require a deviation from the Code of Ethics.
   - The authors should make sure to preserve anonymity (e.g., if there is a special consideration due to laws or regulations in their jurisdiction).

10. **Broader impacts**

    Question: Does the paper discuss both potential positive societal impacts and negative societal impacts of the work performed?

    Answer: [N/A]

    Justification: Our work contributes to better accelerate dLLMs. It does not impact society at large.

    Guidelines:

    - The answer [N/A] means that there is no societal impact of the work performed.
    - If the authors answer [N/A] or [No], they should explain why their work has no societal impact or why the paper does not address societal impact.

- Examples of negative societal impacts include potential malicious or unintended uses (e.g., disinformation, generating fake profiles, surveillance), fairness considerations (e.g., deployment of technologies that could make decisions that unfairly impact specific groups), privacy considerations, and security considerations.
- The conference expects that many papers will be foundational research and not tied to particular applications, let alone deployments. However, if there is a direct path to any negative applications, the authors should point it out. For example, it is legitimate to point out that an improvement in the quality of generative models could be used to generate Deepfakes for disinformation. On the other hand, it is not needed to point out that a generic algorithm for optimizing neural networks could enable people to train models that generate Deepfakes faster.
- The authors should consider possible harms that could arise when the technology is being used as intended and functioning correctly, harms that could arise when the technology is being used as intended but gives incorrect results, and harms following from (intentional or unintentional) misuse of the technology.
- If there are negative societal impacts, the authors could also discuss possible mitigation strategies (e.g., gated release of models, providing defenses in addition to attacks, mechanisms for monitoring misuse, mechanisms to monitor how a system learns from feedback over time, improving the efficiency and accessibility of ML).

11. **Safeguards**

    Question: Does the paper describe safeguards that have been put in place for responsible release of data or models that have a high risk for misuse (e.g., pre-trained language models, image generators, or scraped datasets)?

    Answer: [N/A]

    Justification: We do not release any new models or datasets.

    Guidelines:

    - The answer [N/A] means that the paper poses no such risks.
    - Released models that have a high risk for misuse or dual-use should be released with necessary safeguards to allow for controlled use of the model, for example by requiring that users adhere to usage guidelines or restrictions to access the model or implementing safety filters.
    - Datasets that have been scraped from the Internet could pose safety risks. The authors should describe how they avoided releasing unsafe images.
    - We recognize that providing effective safeguards is challenging, and many papers do not require this, but we encourage authors to take this into account and make a best faith effort.

12. **Licenses for existing assets**

    Question: Are the creators or original owners of assets (e.g., code, data, models), used in the paper, properly credited and are the license and terms of use explicitly mentioned and properly respected?

    Answer: [Yes]

    Justification: We use publicly available datasets and pretrained dLLMs, all of which are properly cited in the paper. We adhere to the licenses and terms of use as specified by their original creators.

    Guidelines:

    - The answer [N/A] means that the paper does not use existing assets.
    - The authors should cite the original paper that produced the code package or dataset.
    - The authors should state which version of the asset is used and, if possible, include a URL.
    - The name of the license (e.g., CC-BY 4.0) should be included for each asset.
    - For scraped data from a particular source (e.g., website), the copyright and terms of service of that source should be provided.

- If assets are released, the license, copyright information, and terms of use in the package should be provided. For popular datasets, `paperswithcode.com/datasets` has curated licenses for some datasets. Their licensing guide can help determine the license of a dataset.
- For existing datasets that are re-packaged, both the original license and the license of the derived asset (if it has changed) should be provided.
- If this information is not available online, the authors are encouraged to reach out to the asset's creators.

13. **New assets**

    Question: Are new assets introduced in the paper well documented and is the documentation provided alongside the assets?

    Answer: [Yes]

    Justification: We will release our complete codebase and thorough documentation to facilitate reproducibility.

    Guidelines:

    - The answer [N/A] means that the paper does not release new assets.
    - Researchers should communicate the details of the dataset/code/model as part of their submissions via structured templates. This includes details about training, license, limitations, etc.
    - The paper should discuss whether and how consent was obtained from people whose asset is used.
    - At submission time, remember to anonymize your assets (if applicable). You can either create an anonymized URL or include an anonymized zip file.

14. **Crowdsourcing and research with human subjects**

    Question: For crowdsourcing experiments and research with human subjects, does the paper include the full text of instructions given to participants and screenshots, if applicable, as well as details about compensation (if any)?

    Answer: [N/A]

    Justification: We do not use crowdsourcing or human subjects.

    Guidelines:

    - The answer [N/A] means that the paper does not involve crowdsourcing nor research with human subjects.
    - Including this information in the supplemental material is fine, but if the main contribution of the paper involves human subjects, then as much detail as possible should be included in the main paper.
    - According to the NeurIPS Code of Ethics, workers involved in data collection, curation, or other labor should be paid at least the minimum wage in the country of the data collector.

15. **Institutional review board (IRB) approvals or equivalent for research with human subjects**

    Question: Does the paper describe potential risks incurred by study participants, whether such risks were disclosed to the subjects, and whether Institutional Review Board (IRB) approvals (or an equivalent approval/review based on the requirements of your country or institution) were obtained?

    Answer: [N/A]

    Justification: We do not use crowdsourcing or human subjects.

    Guidelines:

    - The answer [N/A] means that the paper does not involve crowdsourcing nor research with human subjects.
    - Depending on the country in which research is conducted, IRB approval (or equivalent) may be required for any human subjects research. If you obtained IRB approval, you should clearly state this in the paper.

- We recognize that the procedures for this may vary significantly between institutions and locations, and we expect authors to adhere to the NeurIPS Code of Ethics and the guidelines for their institution.
- For initial submissions, do not include any information that would break anonymity (if applicable), such as the institution conducting the review.

16. **Declaration of LLM usage**

    Question: Does the paper describe the usage of LLMs if it is an important, original, or non-standard component of the core methods in this research? Note that if the LLM is used only for writing, editing, or formatting purposes and does *not* impact the core methodology, scientific rigor, or originality of the research, declaration is not required.

    Answer: [N/A]

    Justification: We used an LLM exclusively for language editing assistance.

    Guidelines:

    - The answer [N/A] means that the core method development in this research does not involve LLMs as any important, original, or non-standard components.
    - Please refer to our LLM policy in the NeurIPS handbook for what should or should not be described.