# Accelerating Diffusion Language Models via Structured Suffix Modeling

**Zifeng Cheng**[1*] **Keda Li**[1*] **Zhiwei Jiang**[1†] **Cong Wang**[1] **Fei Shen**[2] **Qing Gu**[1]
[1] State Key Laboratory of Novel Software Technology, Nanjing University
[2] National University of Singapore
chengzf@nju.edu.cn, 522026320069@smail.nju.edu.cn, jzw@nju.edu.cn,
wang.c@nju.edu.cn, shenfei29@nus.edu.sg, guq@nju.edu.cn

## Abstract

Diffusion Language Models (DLMs) exhibit strong parallel decoding capabilities by denoising multiple tokens in a single generation step. However, this parallelism comes with substantial computational overhead, as each step requires interactions with all suffix tokens. Existing methods typically reduce this cost by retaining only a local suffix window as a substitute for the full suffix. Despite their effectiveness, these methods overlook the structural heterogeneity across suffix regions and re-initialize suffix tokens with identical representations at each timestep. To this end, we propose a structured suffix modeling method for efficient DLM inference. Specifically, we divide the suffix into three regions, i.e., the local, middle, and tail regions, and retain different numbers of suffix tokens in each region according to their structural roles. Moreover, we incorporate the decoding results from the previous step into the suffix token representations at the current step, allowing them to carry evolving denoising information across generation steps. Notably, our method is training-free and orthogonal to several existing acceleration techniques, such as parallel decoding strategies and KV cache. Empirical results across multiple benchmarks on three DLMs demonstrate that our method can further accelerate DLM inference and improve performance in most cases. In particular, in long-sequence inference, our method achieves up to a $72.81\times$ speedup when combined with other acceleration techniques. Our code is available at `https://github.com/zifengcheng/SSM`.

## 1 Introduction

Diffusion Language Models (DLMs) (Li et al., 2022; He et al., 2023; Shi et al., 2024; Nie et al., 2025; Ye et al., 2025) offer a new paradigm for parallel decoding over entire sequences or blocks by leveraging bidirectional context. Compared to autoregressive decoding (Yang et al., 2024; Dubey et al., 2024), this paradigm can predict multiple tokens within a single generation step and effectively reduce latency. However, in block-wise decoding (Nie et al., 2025; Ye et al., 2025), each step not only denoises the current block but also computes attention scores and hidden states for a large number of masked suffix tokens. Although these suffix tokens provide future contextual signals, their associated computation accounts for a substantial portion of the overall cost, limiting the decoding speed. As the generation length increases, this overhead becomes increasingly pronounced, making suffix computation a critical bottleneck for efficient DLM inference.

Recently, several studies (Chen et al., 2026; Xiao et al., 2026) have explored accelerating DLM inference by directly excluding redundant suffix tokens from the model input. Unlike conventional

*Equal contribution.
†Corresponding author.

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

attention-score pruning methods (Wang et al., 2021; Song et al., 2026), which require computing attention scores before pruning tokens based on their magnitudes, suffix dropout predefines the sparsity pattern before the input is fed into the model, without accessing internal model states. DPad (Chen et al., 2026) presents the first systematic investigation into suffix tokens, revealing that they primarily function as a non-semantic information reservoir. Motivated by this observation, DPad retains tokens only within a local suffix window and applies distance-aware dropout to determine whether each token in the window should be preserved. Streaming-dLLM (Xiao et al., 2026) preserves neighboring suffix tokens and the final token to maintain essential contextual and boundary cues, while pruning redundant suffix tokens.

Despite their effectiveness, existing methods still largely formulate suffix dropout as a locality-oriented pruning problem. First, these methods fail to explicitly model the functional roles of different suffix blocks during block-wise denoising. Tokens located in different suffix blocks may provide distinct contextual, structural, and boundary cues, and thus should be assigned different token budgets. Second, suffix tokens are re-initialized at each timestep without incorporating information from previous timesteps, thereby discarding useful denoising signals accumulated during iterative refinement. As a result, existing methods may fail to preserve informative suffix cues and exploit the evolving information propagated across generation steps.

To address these limitations, we propose SSM, a structured suffix modeling framework for efficient DLM inference. Our key insight is that suffix tokens in different regions serve different roles during block-wise decoding. Specifically, the attention scores received by suffix tokens first drop rapidly as the token index increases, then remain stable, and finally rise again. Thus, we first divide the suffix into three regions: the local region, the middle region, and the tail region. For the local region, we retain all neighboring suffix tokens to preserve essential contextual information. For the middle region, we keep only the start token of each suffix block as a lightweight structural anchor, maintaining the coarse suffix skeleton with minimal computation. For the tail region, we additionally retain the final token to explicitly preserve terminal boundary information. Furthermore, we introduce a soft suffix embedding mechanism. At each timestep, we incorporate the decoding results from the previous step into the suffix token representations at the current step. In this way, suffix tokens can carry evolving denoising information across generation steps.

Our method is training-free and can be directly applied to existing DLMs at inference time. Moreover, it is orthogonal to existing acceleration techniques such as parallel decoding and KV cache, enabling further efficiency gains when combined with them. We conduct extensive experiments on mathematical reasoning and code generation benchmarks using LLaDA-Instruct, LLaDA-1.5, and Dream-Base. Empirical results show that SSM consistently reduces inference latency while maintaining competitive accuracy. Notably, in long-sequence inference, our method achieves up to a $72.81\times$ speedup when combined with other acceleration techniques.

Our contributions are summarized as follows:

- We reformulate suffix dropout in DLMs as a role-aware structured suffix modeling problem, revealing that suffix tokens in different regions provide distinct contextual, structural, and boundary cues rather than serving as a homogeneous redundancy reservoir.
- We introduce a soft suffix embedding mechanism that incorporates previous-step decoding results into current suffix token representations, enabling suffix tokens to carry evolving denoising information across generation steps.
- We develop a training-free inference acceleration method that is orthogonal to existing acceleration techniques, achieving efficient DLM inference with competitive or improved task performance.

## 2 Related work

**Accelerating Diffusion Language Models** Unlike autoregressive models that generate one token at a time, dLLMs leverage bidirectional attention to denoise multiple tokens within a single timestep, thereby enabling parallel decoding. Recently, many studies have been devoted to accelerating the inference process of dLLMs to fully exploit their parallel decoding capabilities. Among them, decoding strategies (Wu et al., 2026; Kim et al., 2025b; Huang et al., 2025; Ben-Hamu et al., 2025; Kim et al., 2025a) and KV cache (Ma et al., 2025; Liu et al., 2025; Hu et al., 2026; Song et al., 2026)

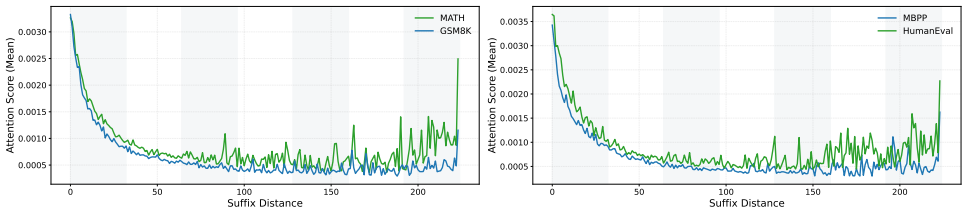


Figure 1: Average attention scores assigned to suffix tokens across all attention heads on four datasets using LLaDA-Instruct, with the generation length set to 256 and the block size set to 32.

are two of the most representative categories of methods. Decoding methods focus on defining new decoding criteria to accelerate generation, with Fast-dLLM (Wu et al., 2026) as a representative approach that performs decoding based on confidence thresholds. In addition, some studies (Zhu et al., 2025b; Zhong et al., 2026) have explored latent refinement decoding. Unlike these methods, our approach focuses on scenarios with only a small number of suffix tokens and does not require iterative refinement. KV cache methods accelerate inference by caching the representations of a subset of tokens to avoid redundant computation. Notably, these methods are orthogonal to our approach and can be combined with it to further improve inference efficiency.

**Suffix Dropout** Recently, some studies (Chen et al., 2026; Xiao et al., 2026) have explored accelerating DLM inference by directly excluding a subset of suffix tokens from the model input. Unlike conventional sparsification strategies (Song et al., 2026; Wang et al., 2026) that require access to the internal states of DLMs, such as attention weights, suffix dropout can discard suffix tokens in advance without accessing internal model states.

DPad (Chen et al., 2026) systematically investigates suffix tokens and reveals that they primarily function as a non-semantic information reservoir that aggregates signals propagated from prefix tokens across multiple Transformer layers. Motivated by this observation, DPad defines a local window and applies distance-aware token dropout within the window at the beginning of decoding each block. Streaming-dLLM (Xiao et al., 2026) preserves neighboring suffix tokens and the final token to maintain essential contextual and boundary cues while pruning redundant suffix tokens, and further introduces an orthogonal dynamic confidence-aware parallel decoding strategy. This parallel decoding mechanism adaptively adjusts decoding thresholds based on both the diffusion stage and the intra-block confidence distribution. However, these methods overlook the heterogeneity across suffix regions and keep suffix token representations unchanged throughout inference.

# 3 Method

## 3.1 Motivation

To investigate whether all suffix tokens contribute equally to block-wise denoising, we visualize the average attention scores received by suffix tokens across all attention heads in Figure 1. The results reveal a clear region-dependent pattern: attention scores initially decrease, then plateau, and finally rise again near the tail region. Nearby suffix tokens receive substantially higher attention scores, suggesting that they provide fine-grained local context for the current decoding block. As the suffix distance increases, the attention scores drop rapidly, indicating that dense interactions with distant suffix tokens are largely redundant. Nevertheless, the attention scores increase sharply in the final block, suggesting that terminal boundary information remains important for decoding the current block.

These observations indicate that suffix tokens should not be handled by a uniform dropout rule. Instead, different suffix regions should be assigned different token budgets according to their functional roles. Specifically, regions close to the current decoding block are the most important and should retain more tokens to preserve local contextual information. The final block is also relatively important for maintaining terminal boundary cues. In contrast, middle suffix blocks are relatively less important and can be represented with a much smaller token budget.

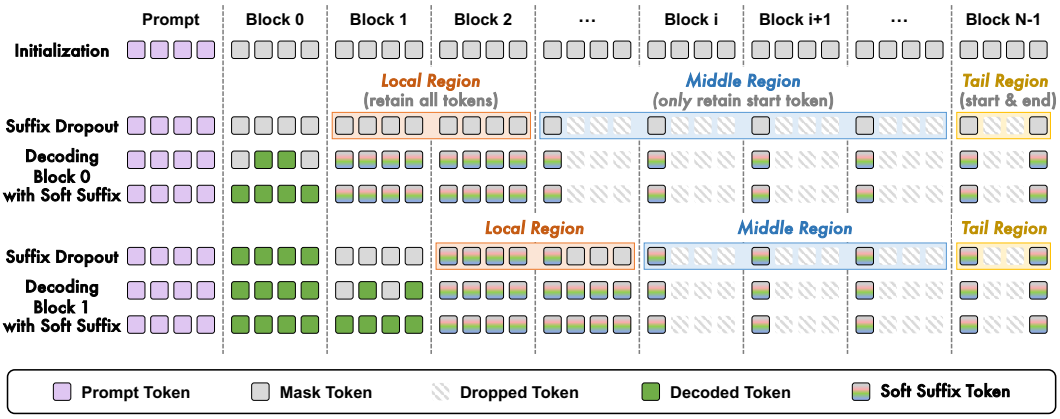


Figure 2: Overview of the proposed structured suffix modeling framework. The method divides the suffix into three regions and retains suffix tokens separately for each region.

### 3.2 Structured Suffix Modeling

Motivated by the above observations, we propose a training-free Structured Suffix Modeling (SSM) framework for efficient DLM inference, as shown in Figure 2. The core idea of SSM is to model the suffix as a structured information field rather than a homogeneous set of redundant masked tokens.

Specifically, SSM consists of two key components. First, we decompose the suffix into three functional regions: a local region that carries fine-grained contextual information, a middle region that provides lightweight structural anchors, and a tail region that preserves terminal boundary information. Different token budgets are then assigned to these regions according to their functional roles, reducing redundant suffix computation while preserving the key information required for block-wise denoising. Second, we introduce a soft suffix embedding mixing mechanism, which incorporates previous-step decoding results into current suffix token representations. This enables retained suffix tokens to carry evolving denoising signals across generation steps, instead of being repeatedly initialized as identical `[MASK]` embeddings. Finally, the resulting sparsified input is fed into the DLM for decoding.

**Suffix Region Partition** Since suffix tokens in different regions have varying importance, we first divide the suffix into three regions, namely the local region, the middle region, and the tail region, and assign different retention ratios to different regions.

We first partition the suffix blocks $S_t$ at timestep $t$ into three regions:

$$S_t = S_t^{\text{local}} \cup S_t^{\text{middle}} \cup S_t^{\text{tail}}. \tag{1}$$

Specifically, the first $w$ suffix blocks are defined as the local region, the last suffix block is defined as the tail region, and the remaining blocks between them are defined as the middle region, where $w$ controls the size of the local region. In particular, when the number of suffix blocks is less than $w$, all suffix blocks are assigned to the local region.

**Suffix Token Retention** After suffix region partitioning, we apply different token retention strategies to the three regions. For the local region, since it is close to the current decoding block and receives higher attention scores, we retain all tokens in the local region to preserve short-range contextual dependencies:

$$\mathcal{R}_t^{\text{local}} = S_t^{\text{local}} \tag{2}$$

where $\mathcal{R}_t^{\text{local}}$ denotes the tokens retained in the local region.

For the middle region, since it receives lower attention scores, we retain only the start token of each block to preserve the skeletal structure of the middle region.

$$\mathcal{R}_t^{\text{middle}} = \{\text{start}(B_j) \mid B_j \in S_t^{\text{middle}}\}. \tag{3}$$

where $B_j$ denotes $j$-th block and start$(\cdot)$ denotes the start-token selection function. Notably, we empirically find that the start token in each middle block usually receives higher attention scores than

other suffix tokens, possibly because it is closer to the current decoding block. Therefore, we choose to retain the start token as a simple and effective strategy.

For the tail region, in addition to the start token, we further retain the final token:

$$\mathcal{R}_t^{\text{tail}} = \{\text{start}(B_{\text{tail}}), \text{end}(B_{\text{tail}})\}. \tag{4}$$

The final token provides important boundary information, allowing the DLM to plan the future generation length from the beginning of decoding.

Finally, we merge all tokens retained from the three regions to form the suffix set retained at step $t$:

$$\mathcal{R}_t = \mathcal{R}_t^{\text{local}} \cup \mathcal{R}_t^{\text{middle}} \cup \mathcal{R}_t^{\text{tail}}. \tag{5}$$

**Soft Suffix Embedding Mixing** To enable suffix tokens to retain prediction information from previous timesteps, we introduce a soft suffix embedding mixing mechanism. Specifically, we mix the embeddings of previously decoded tokens into the `[MASK]` representations, allowing suffix tokens to carry evolving denoising information across generation steps.

For each retained suffix position $i \in \mathcal{R}_t$, let $p_{t+1}^i(v)$ denote the predicted probability of token $v$ at the previous timestep $t+1$. We select the top-$k$ candidate tokens according to $p_{t+1}^i$, denoted as $\mathcal{T}_{t+1}^i$, and re-normalize their probabilities within this set:

$$\tilde{p}_{t+1}^i(v) = \frac{p_{t+1}^i(v)}{\sum_{u \in \mathcal{T}_{t+1}^i} p_{t+1}^i(u)}, \quad v \in \mathcal{T}_{t+1}^i. \tag{6}$$

Given the token embedding $\mathbf{e}_v$ for token $v$, we construct the soft decoded embedding as a weighted combination of the top-$k$ token embeddings:

$$\tilde{\mathbf{e}}_{t+1}^i = \sum_{v \in \mathcal{T}_{t+1}^i} \tilde{p}_{t+1}^i(v) \cdot \mathbf{e}_v. \tag{7}$$

Then, the current input embedding of suffix token $i$ is obtained by mixing the `[MASK]` embedding with the previous-step soft decoded embedding:

$$\tilde{\mathbf{e}}_t^i = (1-\alpha) \cdot \mathbf{e}_{\text{mask}} + \alpha \cdot \tilde{\mathbf{e}}_{t+1}^i, \tag{8}$$

where $\alpha \in [0, 1]$ controls the contribution of the previous-step decoding result.

**Sparse Input Construction** After the above process, we can construct the sparse input for the DLM. Given the prompt $P$, the current block $B_t$, and the full suffix $S_t$, the standard block-wise decoding input can be written as

$$X_t = [P; B_t; S_t].$$

Instead of feeding the full suffix into the model, SSM constructs a sparse input:

$$\tilde{X}_t = [P; B_t; \mathcal{R}_t].$$

The retained suffix tokens keep their original positional encodings (Su et al., 2024), allowing the model to still perceive their relative positions in the original sequence. This is important because the retained middle-region start tokens and the final token are used as structural anchors rather than merely local context tokens.

**Early Termination** To further enhance efficiency, we also use the early termination technique (Chen et al., 2026; Bao et al., 2026) that halts generation upon detecting an end-of-sequence (<eos>) token. This technique performs verification after each decoding block, thereby addressing the computational redundancy caused by fixed-length generation in DLMs. Its utility is most pronounced in long-sequence settings where the maximum length significantly exceeds the actual generated content.

Table 1: Performance comparison of LLaDA-Instruct and Dream-Base on four benchmarks.

| Benchmark | Method | LLaDA-Instruct Efficiency | | | | | LLaDA-Instruct Accuracy (%) | | Dream-Base Efficiency | | | | | Dream-Base Accuracy (%) | |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| | | Latency(s)↓ | | TPS↑ | | $\bar{\ell}/\ell_{\max}$ | Flexible↑ | Strict↑ | Latency(s)↓ | | TPS↑ | | $\bar{\ell}/\ell_{\max}$ | Flexible↑ | Strict↑ |
| GSM8K *4-shot* | Vanilla | 28.52 | 1.00× | 8.13 | 1.00× | 232 / 256 | 78.39 | 37.38 | 22.28 | 1.00× | 11.44 | 1.00× | 255 / 256 | **76.27** | **75.74** |
| | Par. | 8.65 | 3.30× | 26.81 | 3.30× | 232 / 256 | 78.54 | 38.67 | 14.91 | 1.49× | 17.10 | 1.49× | 255 / 256 | 75.51 | 75.51 |
| | DPad+Par. | 6.68 | 4.27× | 24.07 | 2.96× | 160 / 256 | **79.76** | 64.97 | 5.21 | 4.28× | 24.29 | 2.12× | 127 / 256 | 72.78 | 74.75 |
| | Streaming+Par. | 5.25 | 5.43× | 21.86 | 2.69× | 114 / 256 | 78.09 | 74.00 | **5.12** | **4.35**× | 24.31 | 2.13× | 124 / 256 | 75.28 | 75.36 |
| | SSM+Par. | **4.87** | **5.86**× | 22.63 | 2.78× | 110 / 256 | 79.38 | **76.95** | 5.13 | 4.34× | 24.41 | 2.13× | 125 / 256 | 75.44 | 75.44 |
| MATH *4-shot* | Vanilla | 26.25 | 1.00× | 9.47 | 1.00× | 249 / 256 | **33.66** | 8.42 | 20.84 | 1.00× | 12.27 | 1.00× | 255 / 256 | 34.32 | 37.76 |
| | Par. | 10.00 | 2.63× | 24.84 | 2.62× | 249 / 256 | 33.48 | 8.76 | 8.76 | 2.38× | 29.19 | 2.38× | 255 / 256 | **35.36** | **38.62** |
| | DPad+Par. | 9.38 | 2.80× | 22.48 | 2.37× | 211 / 256 | 33.54 | 27.98 | 7.53 | 2.77× | 33.86 | 2.76× | 255 / 256 | 34.44 | 38.32 |
| | Streaming+Par. | 7.35 | 3.57× | 21.41 | 2.26× | 157 / 256 | 31.38 | 30.68 | 7.75 | 2.69× | 32.89 | 2.68× | 255 / 256 | 34.78 | 38.18 |
| | SSM+Par. | **6.11** | **4.30**× | 21.64 | 2.29× | 132 / 256 | 32.04 | **31.96** | **7.37** | **2.83**× | 34.59 | 2.82× | 255 / 256 | 34.48 | 38.00 |
| HumanEval *0-shot* | Vanilla | 35.09 | 1.00× | 13.47 | 1.00× | 473 / 512 | 43.90 | – | 28.43 | 1.00× | 17.96 | 1.00× | 510 / 512 | 51.21 | – |
| | Par. | 11.59 | 3.03× | 40.99 | 3.04× | 475 / 512 | 43.29 | – | 14.57 | 1.95× | 35.06 | 1.95× | 510 / 512 | 53.04 | – |
| | DPad+Par. | 9.23 | 3.80× | 47.36 | 3.52× | 437 / 512 | 46.34 | – | 3.78 | 7.52× | 56.54 | 3.15× | 214 / 512 | 52.43 | – |
| | Streaming+Par. | 5.41 | 6.49× | 54.32 | 4.03× | 293 / 512 | 46.34 | – | 3.83 | 7.42× | 54.99 | 3.06× | 210 / 512 | 52.43 | – |
| | SSM+Par. | **3.35** | **10.47**× | 65.76 | 4.88× | 220 / 512 | **48.17** | – | **3.40** | **8.36**× | 55.72 | 3.10× | 189 / 512 | **53.04** | – |
| MBPP *3-shot* | Vanilla | 62.82 | 1.00× | 4.76 | 1.00× | 299 / 512 | 15.00 | – | 49.07 | 1.00× | 10.43 | 1.00× | 512 / 512 | 52.40 | – |
| | Par. | 14.43 | 4.35× | 20.74 | 4.36× | 299 / 512 | 15.00 | – | 12.35 | 3.97× | 41.45 | 3.97× | 512 / 512 | **56.20** | – |
| | DPad+Par. | 6.03 | 10.42× | 18.22 | 3.83× | 110 / 512 | 39.40 | – | 9.85 | 4.98× | 51.96 | 4.98× | 512 / 512 | 54.80 | – |
| | Streaming+Par. | 6.17 | 10.18× | 16.34 | 3.43× | 100 / 512 | 42.40 | – | 9.66 | 5.08× | 52.94 | 5.08× | 512 / 512 | 54.00 | – |
| | SSM+Par. | **2.68** | **23.44**× | 19.44 | 4.08× | 52 / 512 | **45.20** | – | **8.99** | **5.46**× | 56.89 | 5.45× | 512 / 512 | 55.60 | – |

# 4 Experiments

## 4.1 Experimental Settings

**Datasets and Evaluation Metrics** We evaluate our method on two reasoning datasets (GSM8K (Cobbe et al., 2021) and MATH (Hendrycks et al., 2021)) and two code generation datasets (HumanEval (Chen et al., 2021) and MBPP (Austin et al., 2021)).

We use efficiency and accuracy to evaluate our method. **Accuracy** is evaluated using task-specific metrics: mathematical reasoning datasets are measured by flexible-extract and strict-match, while code generation tasks are measured by pass@1. To assess **efficiency**, we report the average latency, defined as the end-to-end generation time, as well as Tokens Per Second (TPS) and the generation length ratio ($\bar{\ell}/\ell_{\max}$), which compares the actual generated length to the benchmark setting.

**Implementation Details** All experiments were conducted on NVIDIA A800 40GB GPU. We evaluate the performance of SSM on three representative dLLMs: LLaDA-8B-Instruct (Nie et al., 2025), LLaDA-1.5 (Zhu et al., 2025a), and Dream-7B-Base (Ye et al., 2025). Following Chen et al. (2026), we use a block size of 32 and a confidence threshold of 0.9 for parallel decoding. We search for the top-$k$ in the range of [3, 5, 7] and the mixing ratio $\alpha$ is [0.2, 0.3, 0.4]. For the local region size, the search range is [1, 3] with a step size of 0.5 on the GSM8K, MATH, and MBPP datasets, and [3, 5] with a step size of 0.5 on the HumanEval dataset.

## 4.2 Baselines

We use two decoding methods as baselines for comparison. **Vanilla** denotes the original top-1 decoding strategy, where only the token with the highest probability is decoded at each step. **Parallel** decoding (**Par.**) (Wu et al., 2026) denotes threshold-based decoding, where all tokens with probabilities above a predefined threshold are decoded.

Since parallel decoding provides substantial speedups over Vanilla decoding, we combine our suffix dropping method with parallel decoding to demonstrate its effectiveness. **DPad** (Chen et al., 2026) defines a local window and applies distance-aware token dropout within the window at the beginning of decoding each block. **Streaming** (Xiao et al., 2026) preserves all neighboring suffix tokens and the final token to maintain essential contextual and boundary cues, while pruning redundant suffix tokens. In particular, both **DPad** and **Streaming** employ an early termination strategy for fair comparison. Since our paper focuses exclusively on suffix modeling, the **Streaming** baseline does not employ its proposed dynamic confidence-aware parallel decoding strategy.

## 4.3 Main Results

Table 1 compares Vanilla decoding, parallel decoding, DPad+Par., Streaming+Par., and our method combined with Par. on four benchmarks using LLaDA-Instruct and Dream-Base. We also report

Table 2: Ablation study of LLaDA-Instruct on two datasets.

| Method | GSM8K Efficiency | | | | | GSM8K Accuracy (%) | | HumanEval Efficiency | | | | | HumanEval Accuracy (%) |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| | Latency(s)↓ | | TPS↑ | | $\bar{\ell}/\ell_{\max}$ | Flexible↑ | Strict↑ | Latency(s)↓ | | TPS↑ | | $\bar{\ell}/\ell_{\max}$ | pass@1↑ |
| **Vanilla** | 28.52 | 1.00× | 8.13 | 1.00× | 232/256 | 78.39 | 37.38 | 35.09 | 1.00× | 13.47 | 1.00× | 473/512 | 43.90 |
| **SSM+Par. (*Ours*)** | **4.87** | **5.86×** | **22.63** | **2.78×** | **110/256** | **79.38** | **76.95** | **3.35** | **10.47×** | **65.76** | **4.88×** | **220/512** | **48.17** |
| *Local Region:* | | | | | | | | | | | | | |
| **w/o Local Region** | 1.96 | 14.55× | 14.36 | 1.77× | 28/256 | 33.51 | 25.32 | 0.51 | 68.80× | 13.26 | 1.02× | 6 / 512 | 15.24 |
| *Middle Region:* | | | | | | | | | | | | | |
| **w/o Start Token** | 4.95 | 5.76× | 22.82 | 2.81× | 113/256 | 78.32 | 75.59 | 3.59 | 9.77× | 65.25 | 4.84× | 234/512 | 43.29 |
| **Replace the start token with the end token** | 4.79 | 5.95× | 22.71 | 2.79× | 109/256 | 78.62 | 75.82 | 3.30 | 10.63× | 65.14 | 4.84× | 214/512 | 46.34 |
| **Start+End Tokens** | 4.83 | 5.90× | 22.57 | 2.78× | 109/256 | 79.98 | 77.33 | 3.20 | 10.97× | 67.03 | 4.98× | 214/512 | 43.29 |
| *Tail Region:* | | | | | | | | | | | | | |
| **Start Token Only** | 4.95 | 5.76× | 22.62 | 2.78× | 112/256 | 78.85 | 76.19 | 3.30 | 10.63× | 65.05 | 4.83× | 215/512 | 43.29 |
| **End Token Only** | 4.87 | 5.86× | 22.65 | 2.79× | 110/256 | 79.61 | 76.80 | 3.29 | 10.67× | 65.58 | 4.87× | 216/512 | 43.29 |
| **Start+Mid+End Tokens** | 4.83 | 5.90× | 22.83 | 2.81× | 110/256 | 78.62 | 76.12 | 3.42 | 10.26× | 65.56 | 4.87× | 224/512 | 45.73 |
| *Soft Embedding:* | | | | | | | | | | | | | |
| **w/o Soft Embedding** | 5.06 | 5.64× | 22.09 | 2.72× | 111/256 | 77.41 | 73.54 | 4.26 | 8.24× | 59.54 | 4.42× | 253/512 | 48.17 |
| *Early Termination:* | | | | | | | | | | | | | |
| **w/o Early Termination** | 5.34 | 5.34× | 20.61 | 2.54× | 110/256 | 79.38 | 76.95 | 3.96 | 8.86× | 55.94 | 4.15× | 220/512 | 48.17 |

the results using LLaDA-1.5 in Table 5. Table 1 shows that SSM outperforms all baseline methods in most cases, achieving the lowest latency. Notably, compared with the strongest baseline method, Streaming+Par., our method achieves a 2.3× speedup on the MBPP dataset using LLaDA-Instruct. Compared with Par. alone, our method further reduces latency by 43.7%, 38.9%, 71.1%, and 81.4% on LLaDA-Instruct. This also demonstrates that our method can be further combined with decoding strategies for acceleration.

**Efficiency.** The efficiency gains come from two sources. First, SSM reduces per-step suffix computation by assigning different token budgets to different suffix regions. Second, SSM often produces more concise generations, as indicated by smaller $\bar{\ell}/\ell_{\max}$ values. This reduction is not attributed to early termination; instead, it arises because SSM exposes each block to fewer suffix tokens, reducing redundant contextual information and leading to more compact continuations. The reduction in output length is particularly pronounced in code generation tasks: on MBPP with LLaDA-Instruct, $\bar{\ell}/\ell_{\max}$ decreases from 299/512 to 52/512. As a result, SSM achieves substantially larger latency reductions on HumanEval and MBPP than on GSM8K and MATH, since longer generations involve more suffix computation and thus provide greater opportunities for suffix pruning. In contrast, multi-shot mathematical reasoning benchmarks typically feature longer input prompts but shorter outputs, limiting the achievable speedup from our suffix pruning strategy. Beyond this, the improvement of our method on Dream is less pronounced than that on LLaDA-Instruct. We attribute this to differences in their training strategies: LLaDA is trained from scratch, whereas Dream is initialized from an autoregressive model. As a result, LLaDA's native DLM architecture may be more sensitive to suffix tokens, making it more compatible with our method.

We find that latency and TPS are not always aligned. This reflects an inherent limitation of TPS as an efficiency metric (Qian et al., 2026; Chen et al., 2026): when the generation length is substantially shortened, reduced GPU utilization can mechanically lower TPS, even though the end-to-end latency is improved. This motivates the community to seek more appropriate throughput metrics for evaluating performance in future work.

**Accuracy.** In terms of accuracy, SSM maintains competitive performance and often improves over strong baselines. Our improvements are more significant on code tasks and on the strict-match metric for mathematical tasks. Notably, strict-match requires not only generating the correct answer but also following the specific reasoning format demonstrated in the examples, thus reflecting the model's in-context learning ability to some extent. By retaining a small number of suffix tokens, our method encourages the model to produce concise and well-formatted outputs as early as possible, thereby avoiding erroneous intermediate steps. These gains also suggest that suffix dropout does not necessarily trade accuracy for speed.

### 4.4 Ablation Study

Table 2 studies the contribution of each component. All of the proposed components contributed to the performance improvement.

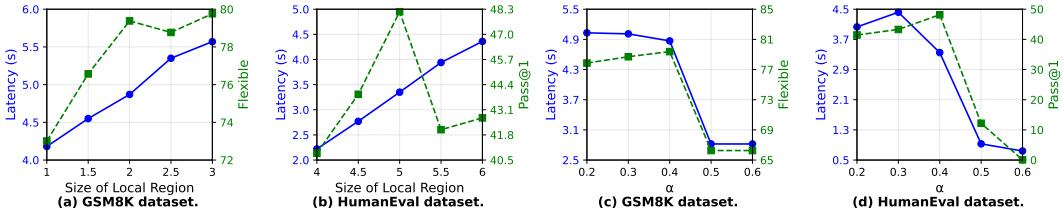


Figure 3: Effects of hyper-parameters on two datasets.

Removing the *local region* leads to severe performance degradation, and the average generation length also becomes extremely short. This suggests that neighboring suffix tokens provide essential fine-grained context and prevent premature or incomplete generation. Thus, the local region should be densely retained. For the *middle region*, removing start tokens not only hurts accuracy on both datasets but also increases latency. This indicates that retaining start tokens in the middle region is necessary for preserving the textual skeleton. Replacing start tokens with end tokens slightly reduces latency but degrades accuracy. This may be because end tokens are better at locating the end boundary rather than preserving block-level structural cues. Notably, additionally retaining end tokens in the middle region does not lead to consistent performance improvements. For the *tail region*, retaining both the start and end tokens achieves more stable performance on the two datasets. Retaining only the start token or only the end token leads to comparable performance, with the end token yielding better results on GSM8K. In addition, introducing extra middle tokens does not bring further performance gains. This suggests that retaining the start and end tokens in the final block is sufficient. Notably, since the tail region contains only one block, removing or adding a single token does not lead to a noticeable change in latency.

*Early termination* fills the remaining blocks with EOS tokens once an EOS token is detected; therefore, it only eliminates the time required to generate subsequent blocks without changing the generation length. *Soft suffix embedding mixing* reduces latency and improves performance on GSM8K, while substantially shortening the generation length and reducing latency on HumanEval. This shows that incorporating previous-step decoding results into current suffix representations helps preserve evolving denoising information and encourages more efficient convergence.

### 4.5 Effects of Hyper-Parameters

**Effects of the size of local region.** We explore the effects of local region size in Figures 3(a) and 3(b). Increasing the local region size leads to higher latency on both datasets, since more retained suffix tokens introduce additional computation. On GSM8K, performance generally improves with a larger local region. This indicates that, for the GSM8K dataset, the information provided by the local region is highly beneficial to the overall performance. However, HumanEval is more sensitive to changes in the hyperparameter. It achieves the highest accuracy when the local region size is set to 5. This result indicates that different tasks require different local region sizes to balance performance and computational cost.

**Effects of $\alpha$.** We then explore the effects of $\alpha$ in Figures 3(c) and 3(d). When $\alpha$ is set to 0.2, 0.3, or 0.4, both latency and accuracy remain relatively stable on the two datasets. The best accuracy on both datasets is achieved at $\alpha = 0.4$. However, when $\alpha$ is further increased to 0.5 or 0.6, accuracy drops sharply. This is likely because predictions from early timesteps are still unreliable; assigning them overly large weights may corrupt suffix token representations and lead to representation collapse. Overall, $\alpha = 0.4$ provides the best accuracy while maintaining low latency on both datasets. This further indicates that the setting of $\alpha$ generalizes well across different tasks.

### 4.6 Comparison with Other Parallel Decoding Methods

We further compare our method with a margin-based parallel decoding strategy (Kim et al., 2025a) to verify its effectiveness. In this strategy, the decision to decode a token depends on whether the probability gap between the top two predictions exceeds a predefined threshold.

Table 3 shows that our method achieves the best accuracy and latency compared with the three baselines. Notably, our method further accelerates the margin-based decoding strategy, achieving a 3.33× speedup on HumanEval. These results further demonstrate the effectiveness and generalizability of our method. Meanwhile, they show that suffix dropping methods are orthogonal to parallel decoding strategies and can be combined with them to further improve both speed and performance.

Table 3: Performance of margin-based parallel decoding methods on two datasets using LLaDA-Instruct.

| Benchmark | Method | Acc. (Flex. / Str.) | Lat.(s) / TPS | Speedup |
|---|---|---|---|---|
| GSM8K *4-shot* | Margin | 78.54 / 38.67 | 8.52 / **27.20** | 1.00× |
| | DPad+Margin | 79.00 / 65.88 | 7.65 / 20.78 | 1.11× |
| | Streaming+Margin | 77.86 / 74.07 | 6.07 / 19.00 | 1.40× |
| | SSM+Margin | **79.61 / 76.65** | **5.57** / 19.83 | **1.53**× |
| HumanEval *0-shot* | Margin | 43.29 / – | 11.54 / 41.16 | 1.00× |
| | DPad+Margin | 42.68 / – | 10.65 / 40.89 | 1.08× |
| | Streaming+Margin | 42.68 / – | 5.74 / 49.41 | 2.01× |
| | SSM+Margin | **43.90** / – | **3.47 / 60.73** | **3.33**× |

## 4.7 Effects of Generation Length

We further compare our method with DPad under different generation lengths on two datasets.

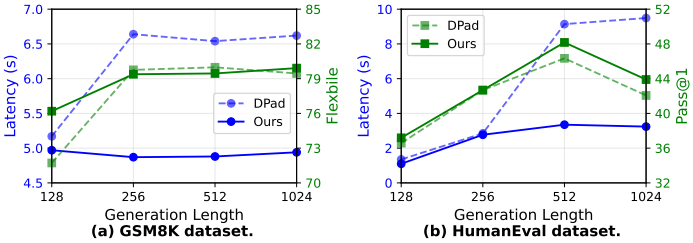


Figure 4: Effects of generation length on two datasets.

Our method achieves lower latency across all generation lengths, with the advantage becoming more pronounced as the generation length increases in Figure 4. For example, on HumanEval with a generation length of 1024, our method is nearly 3× faster than DPad. This indicates that our method can accelerate inference across various generation lengths. The overall trend remains that shorter generation lengths lead to lower latency. Our early termination strategy can reduce the overhead caused by overly long generations to some extent. In terms of accuracy, our method achieves performance comparable to DPad on GSM8K and better performance on HumanEval. Moreover, a longer generation length does not necessarily lead to better performance.

## 4.8 Performance on Long-sequence Generation

To further evaluate the efficiency of SSM in long-sequence generation, we follow Chen et al. (2026) and report the results on GSM8K using LLaDA-1.5 with a maximum generation length of 1024 tokens. As shown in Table 4, we combine SSM with parallel decoding (+Par.) (Wu et al., 2026) and prefix caching (+PC.) (Wu et al., 2026) to examine its compatibility with decoding-based and cache-based acceleration techniques. We compare SSM with DPad under the same acceleration settings.

SSM consistently achieves lower latency than DPad and obtains better accuracy in most cases across different combinations of acceleration techniques, further confirming the effectiveness of our method. It is worth noting that long-sequence generation scenarios involve longer suffixes, where suffix dropout methods can achieve more substantial improvements.

Table 4: Performance on GSM8K with LLaDA-1.5 (1024 tokens, 1-shot).

| **Method** | **Acc. (Flex. / Str.)** | **Lat.(s) / TPS** | **Speedup** |
|---|---|---|---|
| Par. | 78.77 / 49.43 | 11.98 / 16.43 | 1.00× |
| Par.+DPad | **80.89** / 70.36 | 2.74 / 52.61 | 4.37× |
| Par.+SSM | 79.91 / **74.37** | **2.18** / 46.94 | **5.49**× |
| Par.+PC. | 79.76 / 51.63 | 10.85 / 18.09 | 1.00× |
| Par.+PC.+DPad | 80.67 / 72.10 | 2.14 / **65.97** | 5.07× |
| Par.+PC.+SSM | **80.74 / 75.13** | **1.76** / 58.02 | **6.16**× |
| Vanilla | 78.17 / 48.98 | 128.16 / 1.54 | 1.00× |
| **Overall** (Par.+PC.+SSM vs. Vanilla) | | | **72.81**× |

In addition, SSM can be combined with KV cache and parallel decoding for further speedup, confirming that it is orthogonal to these techniques. In addition, both DPad and our method lead to accuracy improvements. We attribute this to the fact that suffix dropping often shortens the generation length, which helps reduce the likelihood of generating erroneous continuations. Overall, compared with Vanilla top-1 decoding, our method combined with Par. and PC. achieves a 72.81× speedup.

## 5 Conclusion

In this paper, we propose SSM, a training-free framework for accelerating DLM inference through structured suffix modeling. Our method exploits the structural heterogeneity of suffix tokens by assigning different token budgets to local, middle, and tail regions, thereby preserving essential contextual, structural, and boundary cues while reducing redundant suffix computation. We further introduce soft suffix embeddings to incorporate previous step decoding information into current suffix representations, preserving evolving denoising signals across generation steps. Experiments on mathematical reasoning and code generation benchmarks show that SSM effectively reduces inference latency while maintaining competitive accuracy. Moreover, SSM is orthogonal to existing acceleration techniques such as parallel decoding and KV caching, enabling further efficiency gains when combined with them. Notably, in long-sequence inference, our method achieves up to a $72.81\times$ speedup when combined with other acceleration techniques.

## Acknowledgments and Disclosure of Funding

We would like to thank the anonymous reviewers for their insightful comments. This work is supported by the JiangSu Natural Science Foundation under Grant No. BK20251989; the National Natural Science Foundation of China under Grants Nos. 62606226, 62441225; the Jiangsu Funding Program for Excellent Postdoctoral Talent under Grants Nos. 2026ZB185, 2026ZB245; the Fundamental and Interdisciplinary Disciplines Breakthrough Plan of the Ministry of Education of China (No. JYB2025XDXM118); the "111 Center" (No. B26023). This work is partially supported by Collaborative Innovation Center of Novel Software Technology and Industrialization.

## References

Austin, J., Odena, A., Nye, M., Bosma, M., Michalewski, H., Dohan, D., Jiang, E., Cai, C., Terry, M., Le, Q., et al. (2021). Program synthesis with large language models. *arXiv preprint arXiv:2108.07732*.

Bao, W., Chen, Z., Xu, D., and Shang, Y. (2026). Learning to parallel: Accelerating diffusion large language models via learnable parallel decoding. In *The Fourteenth International Conference on Learning Representations*.

Ben-Hamu, H., Gat, I., Severo, D., Nolte, N., and Karrer, B. (2025). Accelerated sampling from masked diffusion models via entropy bounded unmasking. In *The Thirty-ninth Annual Conference on Neural Information Processing Systems*.

Chen, M., Tworek, J., Jun, H., Yuan, Q., Pinto, H. P. D. O., Kaplan, J., Edwards, H., Burda, Y., Joseph, N., Brockman, G., et al. (2021). Evaluating large language models trained on code. *arXiv preprint arXiv:2107.03374*.

Chen, X., Huang, S., Guo, C., Wei, C., He, Y., Zhang, J., Li, H. H., and Chen, Y. (2026). DPad: Efficient diffusion language models with suffix dropout. In *The Fourteenth International Conference on Learning Representations*.

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. (2021). Training verifiers to solve math word problems. *arXiv preprint arXiv:2110.14168*.

Dubey, A., Jauhri, A., Pandey, A., Kadian, A., Al-Dahle, A., Letman, A., Mathur, A., Schelten, A., Yang, A., Fan, A., Goyal, A., Hartshorn, A., Yang, A., Mitra, A., Sravankumar, A., Korenev, A., Hinsvark, A., Rao, A., Zhang, A., Rodriguez, A., Gregerson, A., Spataru, A., Rozière, B., Biron, B., Tang, B., Chern, B., Caucheteux, C., Nayak, C., Bi, C., Marra, C., McConnell, C., Keller, C., Touret, C., Wu, C., Wong, C., Ferrer, C. C., Nikolaidis, C., Allonsius, D., Song, D., Pintz, D., Livshits, D., Esiobu, D., Choudhary, D., Mahajan, D., Garcia-Olano, D., Perino, D., Hupkes, D., Lakomkin, E., AlBadawy, E., Lobanova, E., Dinan, E., Smith, E. M., Radenovic, F., Zhang, F., Synnaeve, G., Lee, G., Anderson, G. L., Nail, G., Mialon, G., Pang, G., Cucurell, G., Nguyen, H., Korevaar, H., Xu, H., Touvron, H., Zarov, I., Ibarra, I. A., Kloumann, I. M., Misra, I., Evtimov, I., Copet, J., Lee, J., Geffert, J., Vranes, J., Park, J., Mahadeokar, J., Shah, J., van der Linde, J., Billock, J., Hong, J., Lee, J., Fu, J., Chi, J., Huang, J., Liu, J., Wang, J., Yu, J., Bitton, J., Spisak, J., Park, J., Rocca, J., Johnstun, J., Saxe, J., Jia, J., Alwala, K. V., Upasani, K., Plawiak, K., Li, K., Heafield, K., Stone, K., and et al. (2024). The llama 3 herd of models. *CoRR*, abs/2407.21783.

He, Z., Sun, T., Tang, Q., Wang, K., Huang, X., and Qiu, X. (2023). Diffusionbert: Improving generative masked language models with diffusion models. In *Proceedings of the 61st Annual Meeting of the Association*

*for Computational Linguistics (Volume 1: Long Papers), ACL 2023*, pages 4521–4534. Association for Computational Linguistics.

Hendrycks, D., Burns, C., Kadavath, S., Arora, A., Basart, S., Tang, E., Song, D., and Steinhardt, J. (2021). Measuring mathematical problem solving with the math dataset. *Annual Conference on Neural Information Processing Systems 2021, NeurIPS 2021*.

Hu, Z., Meng, J., Akhauri, Y., Abdelfattah, M. S., sun Seo, J., Zhang, Z., and Gupta, U. (2026). FlashDLM: Accelerating diffusion language model inference via efficient KV caching and guided diffusion. In *The Fourteenth International Conference on Learning Representations*.

Huang, P., Liu, S., Liu, Z., Yan, Y., Wang, S., Chen, Z., and Xiao, T. (2025). Pc-sampler: Position-aware calibration of decoding bias in masked diffusion models. *arXiv preprint arXiv:2508.13021*.

Kim, J., Shah, K., Kontonis, V., Kakade, S. M., and Chen, S. (2025a). Train for the worst, plan for the best: Understanding token ordering in masked diffusions. In *Forty-second International Conference on Machine Learning*.

Kim, S. H., Hong, S., Jung, H., Park, Y., and Yun, S.-Y. (2025b). KLASS: KL-guided fast inference in masked diffusion models. In *The Thirty-ninth Annual Conference on Neural Information Processing Systems*.

Li, X. L., Thickstun, J., Gulrajani, I., Liang, P., and Hashimoto, T. B. (2022). Diffusion-lm improves controllable text generation. In *Annual Conference on Neural Information Processing Systems 2022, NeurIPS 2022*.

Liu, Z., Yang, Y., Zhang, Y., Chen, J., Zou, C., Wei, Q., Wang, S., and Zhang, L. (2025). dllm-cache: Accelerating diffusion large language models with adaptive caching. *arXiv preprint arXiv:2506.06295*.

Ma, X., Yu, R., Fang, G., and Wang, X. (2025). dkv-cache: The cache for diffusion language models. In *The Thirty-ninth Annual Conference on Neural Information Processing Systems*.

Nie, S., Zhu, F., You, Z., Zhang, X., Ou, J., Hu, J., Zhou, J., Lin, Y., Wen, J., and Li, C. (2025). Large language diffusion models. *CoRR*, abs/2502.09992.

Qian, Y.-Y., Su, J., Hu, L., Zhang, P., Deng, Z., Zhao, P., and Zhang, H. (2026). d3llm: Ultra-fast diffusion llm using pseudo-trajectory distillation. *arXiv preprint arXiv:2601.07568*.

Shi, J., Han, K., Wang, Z., Doucet, A., and Titsias, M. K. (2024). Simplified and generalized masked diffusion for discrete data. In *Annual Conference on Neural Information Processing Systems 2024, NeurIPS 2024*.

Song, Y., Liu, X., Li, R., Liu, Z., Huang, Z., Guo, Q., He, Z., and Qiu, X. (2026). Sparse-dllm: Accelerating diffusion llms with dynamic cache eviction. In *Fortieth AAAI Conference on Artificial Intelligence, AAAI 2026*, pages 33038–33046.

Su, J., Ahmed, M. H. M., Lu, Y., Pan, S., Bo, W., and Liu, Y. (2024). Roformer: Enhanced transformer with rotary position embedding. *Neurocomputing*, 568:127063.

Wang, H., Zhang, Z., and Han, S. (2021). Spatten: Efficient sparse attention architecture with cascade token and head pruning. In *IEEE International Symposium on High-Performance Computer Architecture, HPCA 2021*, pages 97–110. IEEE.

Wang, Z., Fang, G., Ma, X., Yang, X., and Wang, X. (2026). Sparsed: Sparse attention for diffusion language models. In *The Fourteenth International Conference on Learning Representations*.

Wu, C., Zhang, H., Xue, S., Liu, Z., Diao, S., Zhu, L., Luo, P., Han, S., and Xie, E. (2026). Fast-dLLM: Training-free acceleration of diffusion LLM by enabling KV cache and parallel decoding. In *The Fourteenth International Conference on Learning Representations*.

Xiao, Z., Hao, Z., Guo, J., Luo, Y., Liu, J., Xu, J., and Hu, H. (2026). Streaming-dllm: Accelerating diffusion llms via suffix pruning and dynamic decoding. *CoRR*, abs/2601.17917.

Yang, A., Yang, B., Zhang, B., Hui, B., Zheng, B., Yu, B., Li, C., Liu, D., Huang, F., Wei, H., Lin, H., Yang, J., Tu, J., Zhang, J., Yang, J., Yang, J., Zhou, J., Lin, J., Dang, K., Lu, K., Bao, K., Yang, K., Yu, L., Li, M., Xue, M., Zhang, P., Zhu, Q., Men, R., Lin, R., Li, T., Xia, T., Ren, X., Ren, X., Fan, Y., Su, Y., Zhang, Y., Wan, Y., Liu, Y., Cui, Z., Zhang, Z., and Qiu, Z. (2024). Qwen2.5 technical report. *CoRR*, abs/2412.15115.

Ye, J., Xie, Z., Zheng, L., Gao, J., Wu, Z., Jiang, X., Li, Z., and Kong, L. (2025). Dream 7b: Diffusion large language models. *CoRR*, abs/2508.15487.

Zhong, L., Wu, L., Fang, B., Feng, T., Jing, C., Wang, W., Zhang, J., Chen, H., and Shen, C. (2026). Beyond hard masks: Progressive token evolution for diffusion language models. *CoRR*, abs/2601.07351.

Zhu, F., Wang, R., Nie, S., Zhang, X., Wu, C., Hu, J., Zhou, J., Chen, J., Lin, Y., Wen, J., and Li, C. (2025a). Llada 1.5: Variance-reduced preference optimization for large language diffusion models. *CoRR*, abs/2505.19223.

Zhu, Q., Yao, Y., Zhao, R., Xiang, Y., Saseendran, A., Jin, C., Teare, P., Liang, B., He, Y., and Gui, L. (2025b). Latent refinement decoding: Enhancing diffusion-based language models by refining belief states. *CoRR*, abs/2510.11052.

# A Limitations

First, the efficiency gain of our method depends on the suffix length. When the suffix constitutes only a small proportion of the sequence, e.g., with a long prefix or a short generation length, the efficiency improvement is relatively limited. Second, our method may cause performance degradation on some datasets, making it less suitable for scenarios where accuracy is highly critical. Finally, we have not evaluated our method on models larger than 10B parameters.

# B Results on LLaDA-1.5

Table 5 reports the performance of LLaDA-1.5 across four benchmarks, further validating the effectiveness of the SSM strategy on advanced diffusion language models.

**Efficiency.** In terms of inference efficiency, SSM+Par. demonstrates superior and stable acceleration across all benchmarks. Compared to DPad and Streaming, SSM consistently maintains the lowest latency. This efficiency is further evidenced by the reduction in average generated sequence length ($\bar{\ell}/\ell_{\max}$), with values reaching as low as 73/512 on HumanEval and 49/512 on MBPP. These results indicate that the SSM strategy induces the model to converge more rapidly to valid outputs, thereby reducing computational redundancy by minimizing the generation of unnecessary long sequences.

**Accuracy.** SSM+Par. can reliably maintain task performance and sometimes further improve it. On GSM8K, it achieves the highest Flexible and Strict accuracy scores. Notably, compared to Vanilla, our method improves the Strict accuracy from 61.87% to 80.14%, effectively bridging the performance gap typically caused by non-standard output formats. Compared to the results on LLaDA-8B-Instruct, the performance gains on LLaDA-1.5 are even more pronounced, suggesting that the effectiveness of SSM can be further amplified by stronger base models.

# C Comparison under Vanilla Top-1 Decoding

We further compare our method with the baselines under vanilla Top-1 decoding in Table 6. Results show that SSM achieves the best latency and accuracy in most cases. These results demonstrate that our method remains effective under Top-1 decoding, indicating its compatibility with different decoding strategies.

Table 5: Performance of LLaDA-1.5 on four benchmarks.

| Benchmark | Method | LLaDA-1.5 Efficiency | | | | | Accuracy (%) | |
|---|---|---|---|---|---|---|---|---|
| | | Latency(s)↓ | Speedup↑ | TPS↑ | Speedup↑ | $\bar{\ell}/\ell_{\max}$ | Flexible↑ | Strict↑ |
| GSM8K *4-shot* | Vanilla | 27.70 | 1.00× | 7.74 | 1.00× | 214 / 256 | 80.59 | 61.87 |
| | Par. | 8.28 | 3.35× | 25.91 | 3.35× | 214 / 256 | 80.82 | 62.62 |
| | DPad+Par. | 6.37 | 4.35× | 24.67 | 3.19× | 157 / 256 | 80.89 | 78.92 |
| | Streaming+Par. | 5.26 | 5.27× | 23.07 | 2.98× | 121 / 256 | 79.15 | 78.09 |
| | SSM+Par. | **4.96** | **5.58**× | 23.96 | 3.10× | 110 / 256 | **81.50** | **80.14** |
| MATH *4-shot* | Vanilla | 25.70 | 1.00× | 8.47 | 1.00× | 217 / 256 | 33.62 | 32.72 |
| | Par. | 9.89 | 2.60× | 21.99 | 2.60× | 217 / 256 | **33.72** | 32.92 |
| | DPad+Par. | 8.74 | 2.94× | 22.32 | 2.64× | 195 / 256 | 33.08 | **35.96** |
| | Streaming+Par. | 7.13 | 3.60× | 21.82 | 2.58× | 155 / 256 | 32.54 | 35.74 |
| | SSM+Par. | **5.79** | **4.44**× | 22.34 | 2.64× | 129 / 256 | 32.24 | 35.86 |
| HumanEval *0-shot* | Vanilla | 34.81 | 1.00× | 3.15 | 1.00× | 109 / 512 | 40.85 | – |
| | Par. | 11.54 | 3.02× | 9.47 | 3.01× | 109 / 512 | 39.63 | – |
| | DPad+Par. | 5.19 | 6.71× | 16.19 | 5.14× | 84 / 512 | 39.63 | – |
| | Streaming+Par. | 8.27 | 4.21× | 9.91 | 3.15× | 141 / 512 | 39.63 | – |
| | SSM+Par. | **4.77** | **7.30**× | 15.38 | 4.88× | 73 / 512 | **45.73** | – |
| MBPP *3-shot* | Vanilla | 62.65 | 1.00× | 1.01 | 1.00× | 63 / 512 | 38.20 | – |
| | Par. | 5.66 | 11.07× | 11.22 | 11.11× | 63 / 512 | 38.60 | – |
| | DPad+Par. | 4.48 | 13.98× | 14.57 | 14.43× | 65 / 512 | 41.60 | – |
| | Streaming+Par. | 3.79 | 16.53× | 21.54 | 21.33× | 81 / 512 | 41.40 | – |
| | SSM+Par. | **2.66** | **23.55**× | 18.50 | 18.32× | 49 / 512 | **42.40** | – |

Table 6: Performance comparison of LLaDA-Instruct using the Top-1 decoding strategy on three benchmarks.

| Benchmark | Method | LLaDA-Instruct Efficiency | | | | | Accuracy (%) | |
|---|---|---|---|---|---|---|---|---|
| | | Latency(s)↓ | | TPS↑ | | $\bar{\ell}/\ell_{\max}$ | Flexible↑ | Strict↑ |
| GSM8K *4-shot* | Vanilla | 28.52 | 1.00× | 8.13 | 1.00× | 232 / 256 | 78.39 | 37.38 |
| | DPad+Van. | 18.80 | 1.52× | 8.54 | 1.05× | 160 / 256 | 78.54 | 63.84 |
| | Streaming+Van. | 13.73 | 2.08× | 8.35 | 1.03× | 114 / 256 | 78.32 | 73.69 |
| | SSM+Van. | **13.51** | **2.11**× | 8.14 | 1.00× | 110 / 256 | **79.68** | **76.65** |
| HumanEval *0-shot* | Vanilla | 34.67 | 1.00× | 13.64 | 1.00× | 473 / 512 | 43.90 | – |
| | DPad+Van. | 27.13 | 1.52× | 16.11 | 1.05× | 437 / 512 | **47.56** | – |
| | Streaming+Van. | 16.78 | 1.28× | 17.05 | 1.18× | 286 / 512 | 45.73 | – |
| | SSM+Van. | **12.64** | **2.74**× | 16.67 | 1.22× | 210 / 512 | 46.34 | – |
| MBPP *3-shot* | Vanilla | 62.11 | 1.00× | 4.82 | 1.00× | 299 / 512 | 15.00 | – |
| | DPad+Van. | 16.25 | 3.82× | 6.70 | 1.39× | 108 / 512 | 40.40 | – |
| | Streaming+Van. | 12.42 | 5.00× | 7.17 | 1.49× | 89 / 512 | 41.60 | – |
| | SSM+Van. | **7.40** | **8.39**× | 7.00 | 1.45× | 51 / 512 | **43.00** | – |

# 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: We have described our claims and contributions clearly in the abstract and introduction.

   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 provide a Limitaion section in Appendix.

   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: We prove our claims with extensive empirical results, instead of theoretical proofs.

   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 provided the hyperparameters to reproduce the main experimental results in the Experimental Settings.

   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: We provide data and code in Github.

   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 have provided these details in the Experimental Settings.

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: [No]

   Justification: The proposed method is training-free and evaluated under a fixed inference-only setting, so there is no variance from random initialization or training runs. Therefore, we do not report error bars or statistical significance tests.

   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: All experiments are conducted on 4 NVIDIA A800 GPUs, as described in the Experimental Settings.

   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: Our research conforms to the NeurIPS Code of Ethics.

   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: [No]

    Justification: Our method focuses on accelerating diffusion language models through suffix-based optimization. Beyond this technical contribution, we do not foresee any direct societal impact.

    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: Our paper does not have such risks.

    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: All of the creators or original owners of assets used in the paper are cited properly.

    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 the data and code, including documentation, after the paper is accepted.

    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: The paper does not involve crowdsourcing nor research with 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: The paper does not involve crowdsourcing nor research with 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: The core method development in this research does not involve LLMs as any important, original, or non-standard components.

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.