ACCELERATING DIFFUSION LARGE LANGUAGE MODELS WITH SLOWFAST SAMPLING: THE THREE GOLDEN PRINCIPLES

Anonymous authors
Paper under double-blind review

ABSTRACT

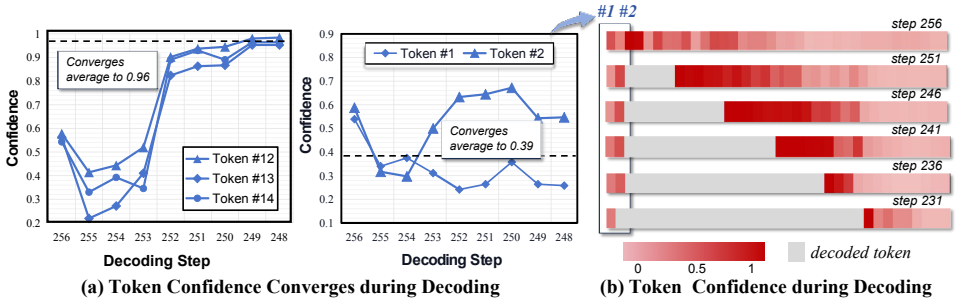
Diffusion-based language models (dLLMs) have emerged as a promising alternative to traditional autoregressive LLMs by enabling parallel token generation and significantly reducing inference latency. However, existing sampling strategies for dLLMs, such as confidence-based or semi-autoregressive decoding, often suffer from static behavior, leading to suboptimal efficiency and limited flexibility. In this paper, we propose **SlowFast Sampling**, a novel dynamic sampling strategy that adaptively alternates between exploratory and accelerated decoding stages. Our method is guided by three golden principles: *certainty principle*, *convergence principle*, and *positional principle*, which govern when and where tokens can be confidently and efficiently decoded. We further integrate our strategy with dLLM-Cache to reduce redundant computation. Extensive experiments across various benchmarks demonstrate the efficiency of our method. Specifically, on the GPQA benchmark, SlowFast Sampling achieves up to **15.63**× speedup on LLaDA with minimal accuracy drop, and up to **34.22**× when combined with caching. Notably, our approach outperforms strong autoregressive baselines like LLaMA3 8B in throughput, demonstrating that well-designed sampling can unlock the full potential of dLLMs for fast and high-quality generation. *Our codes are available in the supplementary materials and will be released on Github.*

1 INTRODUCTION

Large Language Models (LLMs) (Zhao et al., 2025) have rapidly become cornerstone technologies in artificial intelligence, demonstrating remarkable capabilities across a diverse range of natural language understanding and generation tasks. However, the prevalent autoregressive nature of most LLMs, where tokens are generated sequentially one after another, introduces significant inference latency, particularly for long sequences. To address this inherent bottleneck, diffusion-based LLMs (dLLMs) (Ye et al., 2025; Nie et al., 2025b) have emerged as a promising alternative paradigm. These models are capable of generating multiple tokens in parallel, departing from the strict token-by-token process. This parallel decoding capability offers the distinct advantage of potentially accelerating text generation significantly, positioning dLLMs as a compelling and forward-looking direction for efficient language model inference.

However, current ways of sampling with dLLMs often don't perform as well as they could. Common methods include confidence-based selection (Chang et al., 2022) like Fast-dLLM (Wu et al., 2025), where tokens exceeding a confidence threshold are selected for decoding. Another popular method, semi-autoregressive decoding (Arriola et al., 2025), divides the sequence into fixed blocks and decodes within them. Unfortunately, these methods frequently yield unsatisfactory results (*i.e.*, significant accuracy drop when decoding many tokens in parallel), and are characterized by a static, constant sampling speed throughout the generation process. This lack of flexibility highlights the need for a more dynamic sampling approach: one that can smartly decide how many tokens to sample at each step and where these tokens should be located in the sequence.

Motivated by these limitations, we introduce a novel dynamic sampling approach designed to accelerate dLLMs, aiming to unlock the real potential of dLLMs under high-parallel decoding. As



(a) Token Confidence Converges during Decoding (b) Token Confidence during Decoding

Figure 1: **The Three Golden Principles for sampling in diffusion LLMs.** (a) *Convergence Principle*: As decoding proceeds, the confidence values of tokens largely converge to high values, while a few tokens converge to lower values. (b) The confidence map over 256 diffusion steps: High-confidence tokens (in deep red) emerge progressively and are preferentially decoded (*the Certainty Principle*), while selection tends to cluster in contiguous regions (*the Positional Principle*), enabling cache reuse and acceleration.

illustrated in Figure 1, our method is guided by three core observations, which we formulate as the **Three Golden Principles** for effective acceleration:

- **The Certainty Principle:** Tokens exhibiting higher confidence are inherently more determined. Consequently, they are more likely to be decoded correctly early in the process and require less adjustment in subsequent diffusion steps.
- **The Convergence Principle:** As the diffusion process unfolds and tokens are progressively refined, the semantic meaning of many tokens stabilizes, and their associated confidence scores converge towards a steady value. This convergence indicates that these tokens have largely settled into their final form and require minimal further refinement.
- **The Positional Principle:** We observe that even without explicit constraints, the model's sampling preferences often gravitate towards tokens in specific, frequently neighboring or clustered, positions. This inherent positional bias can be strategically exploited. For instance, parts of the sequence can be effectively cached, leading to significant acceleration gains.

Integrating these principles, we propose SlowFast Sampling with two phases: an Exploratory Stage and an Accelerated Decoding Stage. In the exploratory stage, the model loosely decodes to locate spans with emerging certainty and convergence. The accelerated stage then parallelly decodes these high-certainty tokens, reducing effort on already determined parts. This division yields significant speedups, reaching $15.63\times$ on LLaDA and up to $34.22\times$ when combined with **dLLM-Cache** (Liu et al., 2025) on the GPQA benchmark, with minimal accuracy loss. Our contributions are threefold:

1. We propose three golden principles based on token certainty, convergence, and positional influence, which critically govern effective and efficient sampling in dLLMs.

2. Building on these principles, we introduce SlowFast Sampling, a novel two-stage dynamic strategy specifically designed to leverage these principles for optimal acceleration of dLLM.

3. Through experiments on various benchmarks, we demonstrate that SlowFast Sampling achieves significant inference acceleration (*e.g.*, up to ***15.63***$\times$ on LLaDA with SlowFast Sampling alone, and up to ***34.22***$\times$ when combined with dLLM-Cache on the GPQA dataset) without compromising response quality, thereby offering a superior speed-quality trade-off compared to baseline and simpler sampling methods.

2 Related Work

2.1 Diffusion Models for Language

Diffusion Models (DMs) (Sohl-Dickstein et al., 2015; Ho et al., 2020; Song et al., 2021) have revolutionized generative modeling, particularly in continuous domains like images (Rombach et al., 2022; Peebles & Xie, 2023). However, adapting these models to discrete data such as text presents unique challenges due to its discrete nature. A promising approach in discrete diffusion models

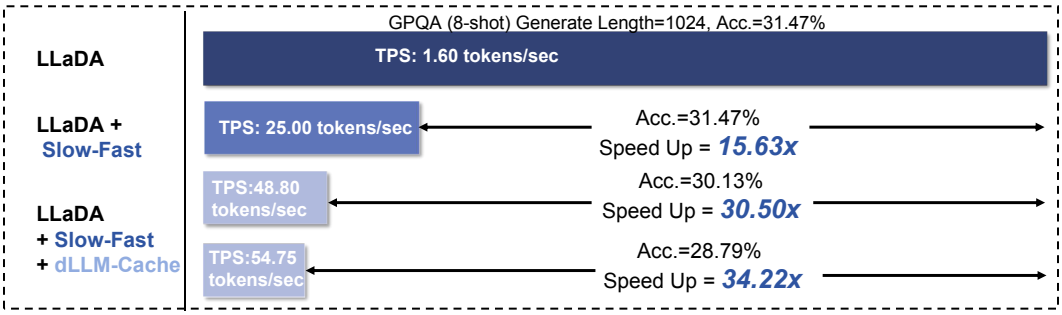


Figure 2: **Throughput and accuracy comparison on GPQA (8-shot, Length=1024) on LLaDA with our method,** including (1) vanilla decoding, (2) SlowFast Sampling, and (3) SlowFast Sampling further enhanced by dLLM-Cache. Compared to the vanilla setting, SlowFast Sampling alone achieves a **15.63**× speedup while maintaining comparable accuracy. With dLLM-Cache, throughput improves further to **54.75 tokens/sec** (up to **34.22**× speedup), with only minor drops in accuracy.

involves Masked Diffusion Models (MDMs) (Austin et al., 2021; Lou et al., 2023; Shi et al., 2024; Nie et al., 2025a;b; Hoogeboom et al., 2021; Campbell et al., 2022), which iteratively predict masked tokens based on their context. These advancements have transformed text generation, offering a compelling alternative to autoregressive paradigms in large language models (LLMs). Notable examples include LLaDA (Nie et al., 2025b), an 8B MDM trained from scratch with a bidirectional Transformer, and Dream (Ye et al., 2025), which initializes from pre-trained ARM weights. Both models demonstrate performance comparable to similarly-sized ARMs like LLaMA3 8B (Dubey et al., 2024). Their bidirectional architecture may overcome ARM limitations such as the reversal curse (Berglund et al., 2023), making diffusion a competitive alternative for foundational LLMs.

2.2 Acceleration Methods for Diffusion-based LLMs

The high inference latency of dLLMs, primarily due to their iterative denoising process (Nie et al., 2025b; Ye et al., 2025), has spurred research into acceleration techniques. Various strategies have been developed, mainly including caching mechanisms and advanced sampling techniques.

Caching Mechanisms. Feature caching reduces redundant computations by reusing intermediate features. dLLM-Cache (Liu et al., 2025) combines long-interval prompt and short-interval response caching with a V-verify mechanism for faster inference. Sparse-dLLM (Song et al., 2025) applies dynamic cache eviction with sparse attention, retaining only salient tokens to cut memory and boost speed. dKV-Cache (Ma et al., 2025) adopts delayed KV caching to reuse decoded tokens' representations, achieving 2–10× acceleration.

Advanced Sampling Techniques. Optimizing the sampling process itself is another major direction for accelerating dLLMs. Low-confidence remasking (Chang et al., 2022; Nie et al., 2025b) prioritizes high-confidence tokens to speed up convergence; semi-autoregressive (Nie et al., 2025b; Arriola et al., 2025) remasking divides sequences into blocks, applying random and low-confidence strategies. Additionally, exact simulation methods for MDMs like the first-hitting sampler (Zheng et al., 2024) have made progress in reducing sampling steps or enhancing per-step efficiency.

3 Methodology

3.1 Preliminary

Inference Process of Diffusion Large Language Models. Diffusion Large Language Models (dLLMs) generate text $\mathbf{y} = (y_1, \ldots, y_L)$ from a prompt $\mathbf{c} = (c_1, \ldots, c_M)$ through an iterative denoising process. The model refines an intermediate state $\mathbf{y}^{(k)} \in \mathcal{T}^L$ over N discrete steps, from $k = N$ to $k = 0$, where $\mathcal{T}$ is the token vocabulary. The process begins with a fully masked sequence:

$$\mathbf{y}^{(N)} = (\underbrace{\texttt{[MASK]}, \ldots, \texttt{[MASK]}}_{L \text{ times}}) \tag{1}$$

where $\texttt{[MASK]} \in \mathcal{T}$ is the special mask token.

At each step $k \in \{N, N-1, \ldots, 1\}$, a mask predictor p_θ estimates the original clean sequence $\mathbf{r}_0 = (r_{0,1}, \ldots, r_{0,L})$ from the noisy state $\mathbf{y}^{(k)}$ and prompt $\mathbf{c}$:

$$P_\theta(\mathbf{r}_0|\mathbf{c}, \mathbf{y}^{(k)}) \tag{2}$$

An estimate of the clean sequence at step k, $\hat{\mathbf{r}}_0^{(k)}$, is obtained through greedy decoding:

$$\hat{r}_{0,i}^{(k)} = \arg\max_{v \in \mathcal{T}} P_\theta(r_{0,i} = v|\mathbf{c}, \mathbf{y}^{(k)}) \quad \forall i \in \{1, \ldots, L\} \tag{3}$$

Although the predictor p_θ can decode all masked tokens `[MASK]` in one step, to ensure high-quality generation, dLLM adopts a multi-step decoding process. At each step, the remask strategy refines the tokens. The transition to the next state $\mathbf{y}^{(k-1)}$ is governed by a sampling strategy S:

$$\mathbf{y}^{(k-1)} = S(\hat{\mathbf{r}}_0^{(k)}, \mathbf{y}^{(k)}, k) \tag{4}$$

This iterative denoising process is a sampling procedure. Our work aims to improve the sampling efficiency of dLLM inference, which can be computationally expensive due to the N sequential steps. Using LLaDA (Nie et al., 2025b) as an example, we describe its core sampling strategies. In LLaDA, time steps are defined as $t_k = k/N$ and $t_{k-1} = (k-1)/N$.

LLaDA explores various strategies for the transition function S in Equation 4, which differ mainly in how tokens from $\hat{\mathbf{r}}_0^{(k)}$ update $\mathbf{y}^{(k)}$ to form $\mathbf{y}^{(k-1)}$, especially for positions that were `[MASK]` in $\mathbf{y}^{(k)}$:

Random Remasking. In this strategy, for each position i:
If $y_i^{(k)} \neq$ `[MASK]`, then $y_i^{(k-1)} = y_i^{(k)}$ (known tokens are preserved).
If $y_i^{(k)} =$ `[MASK]`, then $y_i^{(k-1)}$ is set to $\hat{r}_{0,i}^{(k)}$ with probability $1 - \frac{k-1}{k}$, and remains `[MASK]` with probability $\frac{k-1}{k}$, ensuring the expected number of masked tokens aligns with the noise schedule.

Low-Confidence Remasking. This deterministic strategy aims to improve sample quality by selectively unmasking tokens. For each position i, if $y_i^{(k)} =$ `[MASK]`, the model predicts $\hat{r}_{0,i}^{(k)}$ and computes its confidence. Specifically, the confidence c_i is given by:

$$c_i = P_\theta(\hat{r}_{0,i}^{(k)}|\mathbf{c}, \mathbf{y}^{(k)}). \tag{5}$$

If $y_i^{(k)} \neq$ `[MASK]`, then $c_i = 1$.

The target number of unmasked tokens for state $\mathbf{y}^{(k-1)}$ is given by:

$$n_{un} = \lfloor L(1 - t_{k-1}) \rfloor = \lfloor L\left(1 - \frac{k-1}{N}\right) \rfloor. \tag{6}$$

The tokens corresponding to the n_{un} highest confidences are unmasked in $\mathbf{y}^{(k-1)}$, while the remaining positions are set to `[MASK]`.

Semi-Autoregressive Remasking. This strategy is used in LLaDA after Supervised Fine-Tuning. The sequence is divided into blocks, and generation proceeds block by block from left to right. Within each block, the random remasking or the low-confidence remasking strategy is applied iteratively to denoise the block before moving to the next.

The choice of sampling strategy S, along with the number of steps N, significantly affects both the generation quality and latency.

3.2 Three Golden Principles of dLLMs

Building upon the iterative denoising process outlined in Section 3.1, our empirical analysis of dLLM behavior, particularly with models like LLaDA, has revealed consistent patterns in token generation. These patterns, which we term the **Three Golden Principles**, form the bedrock of our proposed acceleration strategy. They describe how token certainty, convergence, and positional effects interact during the diffusion process, offering key insights into optimizing sampling.

The Certainty Principle: High Confidence Indicates Determination. We observe that tokens predicted with high confidence by the mask predictor p_θ are significantly more likely to be part of the

final, correct sequence and tend to remain unchanged in subsequent denoising steps. The confidence for a predicted token $\hat{r}_{0,i}^{(k)}$ at position i and step k is given by $P_\theta(\hat{r}_{0,i}^{(k)}|\mathbf{c}, \mathbf{y}^{(k)})$. As illustrated in Figure 1 (b), at any given step k, a subset of tokens typically exhibits substantially higher confidence scores compared to others. These high-confidence tokens are prime candidates for early "acceptance" or less frequent resampling.

Implication for Acceleration: By prioritizing tokens that quickly reach a high confidence threshold, we can reduce redundant computations on already determined parts of the sequence.

The Convergence Principle: Tokens Stabilize Over Iterations. During the iterative refinement process, individual tokens (both their predicted identity $\hat{r}_{0,i}^{(k)}$ and their confidence $P_\theta(\hat{r}_{0,i}^{(k)}|\mathbf{c}, \mathbf{y}^{(k)})$) undergo a period of fluctuation before settling. As shown in Figure 1 (a), a representative token might initially change its predicted identity and confidence across several early diffusion steps. However, as k decreases, the token's confidence converges are often to a stable value. This convergence signals that the model has formed a consistent belief about the token's identity within its current context.

Implication for Acceleration: Tokens that have demonstrated convergence (*i.e.*, stable identity and confidence over a window of recent steps) are less likely to change. Aggressively decoding can prevent unnecessary re-evaluation, thereby speeding up the process.

The Positional Principle: Decoding Exhibits Regional Preferences. Beyond individual token behaviors, we find that the generation process often exhibits spatial patterns. High-confidence and early-converging tokens do not appear randomly scattered throughout the sequence. Instead, they frequently emerge in contiguous blocks or localized regions, as suggested by Figure 1 (b). This phenomenon might be due to local semantic dependencies or the influence of strongly contextualized parts of the prompt $\mathbf{c}$. For example, after a few initial steps, a particular span of tokens might collectively achieve high confidence and stability, while other regions remain largely masked or uncertain. The model appears to focus its decoding efforts on specific segments at different stages.

Implication for Acceleration: Recognizing these regions allows for targeted decoding. Instead of uniformly processing all tokens, computational resources can be concentrated on regions that are currently most amenable to decoding.

These three principles collectively indicate that a one-size-fits-all, static sampling strategy is inherently inefficient. They motivate a dynamic approach where the number of tokens sampled, their selection criteria, and their locations are adapted based on the evolving state of the generated sequence. Our SlowFast Sampling methodology, detailed in Section 3.3, is designed to explicitly leverage these observations to achieve significant inference speedups.

3.3 SLOWFAST SAMPLING

1. Exploratory Stage (Slow Phase): Identifying the Next Stable Region. As illustrated in Figure 3, the primary goal of this stage is to cautiously advance decoding while identifying a promising, stable region for subsequent rapid processing. Starting from s_{cycle} and extending to the end of the full sequence L, this stage operates as follows for a limited number of dLLM steps:

- **Cautious Decoding:** At each dLLM step k within this stage, we perform a conservative decoding update. We select the top-k_{slow} tokens within the current exploratory window $[s_{cycle}, L]$ that exhibit the highest confidence $P_\theta(\hat{r}_{0,i}^{(k)}|\mathbf{c}, \mathbf{y}^{(k)})$, and these are unmasked to form part of $\hat{\mathbf{r}}_0^{(k)}$ (as per Equation 3 for these tokens, followed by Equation 4).
- **End Point of Convergence Prediction:** Concurrently, the model predicts a end point of candidate convergence, $e_{cand}^{(k)}$. This is defined as the furthest token position $i \in [s_{cycle}, L]$ for which the predicted confidence $P_\theta(\hat{r}_{0,i}^{(k)}|\mathbf{c}, \mathbf{y}^{(k)})$ exceeds a minimum confidence threshold τ_{min_conf}. Mathematically:

$$e_{cand}^{(k)} = \max\{i \mid i \in [s_{cycle}, L] \wedge P_\theta(\hat{r}_{0,i}^{(k)}|\mathbf{c}, \mathbf{y}^{(k)}) > \tau_{min\_conf}\} \tag{7}$$

  This $e_{cand}^{(k)}$ represents an estimate of how far into the sequence the model can confidently decode at the current step k.
- **Stability Check:** Throughout the Exploratory Stage, candidate convergence horizons $e_{cand}^{(j)}$ predicted at each dLLM step j are tracked. Let $H_W(k_s) = \{e_{cand}^{(l)} \mid l \in [\max(1, k_s - W_{hist} + 1), k_s]\}$

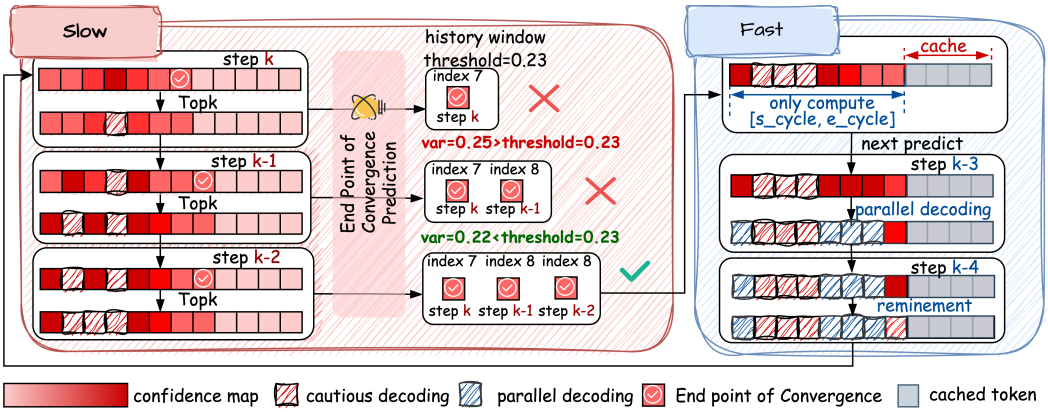


Figure 3: **Overview of the SlowFast Sampling Pipeline.** The method alternates between a **Slow (Exploratory) stage** and a **Fast (Accelerated) stage** for efficient token generation. In the Slow phase (left), the model conducts *cautious decoding* by selecting top-k high-confidence tokens per step while continuously predicting the *End Point of Convergence* and calculating confidence variance across a history window. Once variance drops below threshold (*e.g.*, $0.22 < 0.23$), the corresponding region $[s_{cycle}, e_{cycle}]$ is considered stable. In the Fast phase (right), this stable span is decoded in parallel with aggressive unmasking of high-confidence tokens, while tokens beyond the span are temporarily skipped and their results *cached* for reuse. This alternating structure reduces redundant computation and accelerates decoding while maintaining output quality.

be the set of candidate horizons in the sliding window of the latest W_{hist} predictions ending at exploratory step k_s. Stability is achieved, concluding this stage, when $\mathrm{Var}(H_W(k_s)) < \sigma^2_{stable}$ for a window where $k_s \geq W_{hist}$. The exploratory stage ends at step k_{final}, defined as:

$$k_{final} = \min\left(\{k_s \mid W_{hist} \leq k_s \leq K_{max} \wedge \mathrm{Var}(H_W(k_s)) < \sigma^2_{stable}\} \cup \{K_{max}\}\right) \tag{8}$$

If the stability criterion is not met by K_{max} (the maximum allotted exploratory steps), then $k_{final} = K_{max}$. The cycle's convergence horizon e_{cycle}, is then set to the mean of the candidate horizons in the final window $H_W(k_{final})$:

$$e_{cycle} = \mathrm{Mean}(H_W(k_{final})) = \frac{1}{|H_W(k_{final})|} \sum_{e \in H_W(k_{final})} e \tag{9}$$

Once this stability criterion is met, the Exploratory Stage concludes. The endpoint for the subsequent Accelerated Decoding Stage, e_{cycle}, is set to the last recorded candidate horizon, $e^{(k)}_{cand}$. If stability is not achieved within a maximum number of exploratory steps, e_{cycle} can be set to a conservatively determined position or the process might default to a full-sequence cautious decode for that cycle.

2. Accelerated Decoding Stage (Fast Phase): Rapid Parallel Refinement. As shown in the right half of Figure 3, once a stable region $[s_{cycle}, e_{cycle}]$ is identified, this stage aims to rapidly denoise tokens within this span, while efficiently handling tokens outside it:

- **Out-of-Span Caching:** For tokens outside the identified span (*i.e.*, positions $i > e_{cycle}$), if their current predicted confidence is low (*e.g.*, below τ_{min_conf}), their predicted values $\hat{r}^{(k)}_{0,i}$ from one dLLM step within this stage are computed and then cached. These cached values can be reused in subsequent dLLM steps for these positions, provided they remain outside an active decoding span, thus saving redundant computations.
- **In-Span Parallel Decoding:** Within the span $[s_{cycle}, e_{cycle}]$, an aggressive parallel decoding is attempted. All tokens $y^{(k)}_i = \texttt{[MASK]}$ for $i \in [s_{cycle}, e_{cycle}]$ for which the predicted confidence $P_\theta(\hat{r}^{(k)}_{0,i} | \mathbf{c}, \mathbf{y}^{(k)})$ exceeds the high certainty threshold τ_{high_conf}, *i.e.*,

$$P_\theta(\hat{r}^{(k)}_{0,i} | \mathbf{c}, \mathbf{y}^{(k)}) > \tau_{high\_conf} \tag{10}$$

are unmasked by setting their value to the corresponding prediction $\hat{r}^{(k)}_{0,i}$. This update is performed simultaneously for all such qualifying tokens in a single conceptual step to form $\hat{\mathbf{r}}^{(k)}_0$.

Table 1: **Performance of LLaDA 8B and Dream 7B with SlowFast Sampling** on 8 benchmarks.

Task	Method	Inference Efficiency		Performance	Method	Inference Efficiency		Performance
		TPS↑	Speed↑	Score↑		TPS↑	Speed↑	Score↑
Mathematics & Science								
GSM8K	LLaDA	4.55	1.00×	69.83	Dream	8.16	1.00×	77.02
	+ Fast-dLLM (Parallel)	7.45$_{+2.90}$	1.64×$_{+0.64}$	69.60$_{-0.23}$	+ Fast-dLLM (Parallel)	12.72$_{+4.56}$	1.55×$_{+0.55}$	73.09$_{-3.93}$
	+ SlowFast	14.57$_{+10.02}$	3.20×$_{+2.20}$	69.59$_{-0.27}$	**+ SlowFast**	17.15$_{+8.99}$	2.10×$_{+1.10}$	76.50$_{-0.52}$
GPQA	LLaDA	3.31	1.00×	31.47	Dream	5.43	1.00×	35.93
	+ Fast-dLLM (Parallel)	11.72$_{+8.41}$	3.54×$_{+2.54}$	32.13$_{+0.66}$	+ Fast-dLLM (Parallel)	15.88$_{+10.45}$	2.92×$_{+1.92}$	31.01$_{-4.92}$
	+ SlowFast	16.36$_{+13.05}$	4.94×$_{+3.94}$	31.91$_{+0.44}$	**+ SlowFast**	16.56$_{+11.13}$	3.05×$_{+2.05}$	35.94$_{+0.01}$
Math	LLaDA	5.14	1.00×	30.16	Dream	8.48	1.00×	38.68
	+ Fast-dLLM (Parallel)	8.94$_{+3.80}$	1.74×$_{+0.74}$	30.52$_{+0.36}$	+ Fast-dLLM (Parallel)	22.51$_{+14.03}$	2.65×$_{+1.65}$	34.14$_{-4.54}$
	+ SlowFast	11.27$_{+6.13}$	2.19×$_{+1.19}$	29.64$_{-0.52}$	**+ SlowFast**	23.00$_{+14.52}$	2.71×$_{+1.71}$	38.24$_{-0.44}$
General Tasks								
MMLU-pro	LLaDA	9.16	1.00×	23.30	Dream	14.97	1.00×	24.14
	+ Fast-dLLM (Parallel)	15.62$_{+6.46}$	1.71×$_{+0.71}$	23.50$_{+0.20}$	+ Fast-dLLM (Parallel)	25.50$_{+10.53}$	1.70×$_{+0.70}$	19.53$_{-4.61}$
	+ SlowFast	23.14$_{+13.98}$	2.53×$_{+1.53}$	23.85$_{+0.55}$	**+ SlowFast**	22.80$_{+7.83}$	1.52×$_{+0.52}$	22.91$_{-1.23}$
MMLU	LLaDA	5.02	1.00×	62.11	Dream	8.46	1.00×	72.61
	+ Fast-dLLM (Parallel)	12.94$_{+7.92}$	2.58×$_{+1.58}$	61.67$_{-0.44}$	+ Fast-dLLM (Parallel)	17.39$_{+8.93}$	2.05×$_{+1.05}$	64.59$_{-8.02}$
	+ SlowFast	16.81$_{+11.79}$	3.35×$_{+2.35}$	66.56$_{+4.45}$	**+ SlowFast**	18.43$_{+9.97}$	2.18×$_{+1.18}$	75.13$_{+2.52}$
BBH	LLaDA	4.04	1.00×	44.97	Dream	6.93	1.00×	51.83
	+ Fast-dLLM (Parallel)	10.73$_{+6.69}$	2.66×$_{+1.66}$	44.13$_{-0.84}$	+ Fast-dLLM (Parallel)	27.84$_{+20.91}$	4.01×$_{+3.01}$	52.28$_{+0.45}$
	+ SlowFast	21.19$_{+17.15}$	5.24×$_{+4.24}$	44.60$_{-0.37}$	**+ SlowFast**	28.14$_{+21.21}$	4.06×$_{+3.06}$	50.55$_{-1.28}$
Code								
MBPP	LLaDA	4.98	1.00×	40.80	Dream	8.92	1.00×	54.20
	+ Fast-dLLM (Parallel)	8.15$_{+3.17}$	1.64×$_{+0.64}$	40.80$_{+0.00}$	+ Fast-dLLM (Parallel)	22.87$_{+13.95}$	2.56×$_{+1.56}$	49.40$_{-4.80}$
	+ SlowFast	13.32$_{+8.34}$	2.67×$_{+1.67}$	41.00$_{+0.20}$	**+ SlowFast**	29.07$_{+20.15}$	3.26×$_{+2.26}$	54.60$_{+0.40}$
HumanEval	LLaDA	11.24	1.00×	31.71	Dream	11.49	1.00×	34.15
	+ Fast-dLLM (Parallel)	23.05$_{+11.81}$	2.05×$_{+1.05}$	32.32$_{+0.61}$	+ Fast-dLLM (Parallel)	24.33$_{+12.84}$	2.12×$_{+1.12}$	32.92$_{-1.23}$
	+ SlowFast	35.46$_{+24.22}$	3.15×$_{+2.15}$	33.54$_{+1.83}$	**+ SlowFast**	25.38$_{+13.89}$	2.21×$_{+1.21}$	35.36$_{+1.21}$

- **Fallback Top-k Refinement:** If the number of tokens meeting the τ_{high_conf} criterion within the span is insufficient to make substantial progress (*e.g.*, less than one), we revert to a more conservative update for this dLLM step within the span. Specifically, we select the top-k_{fast} tokens within $[s_{cycle}, e_{cycle}]$ based on confidence and unmask them. This ensures steady progress even when widespread high certainty is not yet achieved.

After the Accelerated Decoding Stage completes, the starting position for the next cycle's Exploratory Stage is updated to $s_{cycle} \leftarrow e_{cycle}$. This cyclical process of exploration and accelerated decoding continues until the entire sequence is generated. This SlowFast approach dynamically adapts the decoding focus and intensity, leveraging the Certainty and Positional principles to identify promising regions and the Convergence principle to stabilize them efficiently.

4 Experiment

4.1 Experiment Settings

Implementation Details To evaluate the effectiveness of our proposed dynamic sampling approach SlowFast Sampling, we conducted experiments on representative dLLMs: LLaDA 8B (Nie et al., 2025b) and Dream 7B (Ye et al., 2025), focusing on measuring the inference acceleration across various benchmarks. All experiments were conducted on NVIDIA RTX 4090 GPUs.

Evaluation Metrics We evaluated sampling acceleration and generation quality of SlowFast Sampling using quantitative metrics. Inference speed is measured in Tokens Per Second (TPS), indicating the average number of tokens generated per second. Generation quality is assessed using task-specific metrics, *e.g.*, accuracy on GSM8K, reflecting the model's performance under inference acceleration.

4.2 Main Results

Performance and efficiency gains across models. Table 1 reports throughput and model performance for both LLaDA 8B and Dream 7B, with and without SlowFast Sampling. These results demonstrate that our method brings significant improvements in inference efficiency and achieves lossless acceleration in most cases. In our experiments, the key hyperparameters of SlowFast Sampling, τ_{min_conf} and τ_{high_conf}, were set to 0.1 and 0.85, respectively. The remaining hyperparameters in our method were set as follows: maximum exploratory steps $K_{max} = 8$, sliding window size

Table 2: **Performance of LLaDA 8B and Dream 7B with SlowFast Sampling and dLLM-Cache.**

Task	Method	Inference Efficiency		Performance	Method	Inference Efficiency		Performance
		TPS↑	Speed(TPS)↑	Score↑		TPS↑	Speed(TPS)↑	Score↑
Mathematics & Science								
GSM8K	LLaDA	4.55	1.00×	69.83	Dream	8.16	1.00×	77.02
	+ Fast-dLLM (Parallel+Cache)	15.50$_{+10.95}$	3.41×$_{+2.41}$	68.77$_{-1.06}$	+ Fast-dLLM (Parallel+Cache)	31.07$_{+22.91}$	3.81×$_{+2.81}$	69.45$_{-7.57}$
	+ SlowFast + Cache	26.99$_{+22.44}$	5.93×$_{+4.93}$	69.60$_{-0.23}$	**+ SlowFast + Cache**	46.17$_{+38.01}$	5.66×$_{+4.66}$	72.10$_{+4.92}$
GPQA	LLaDA	3.31	1.00×	31.47	Dream	5.43	1.00×	35.93
	+ Fast-dLLM (Parallel+Cache)	30.21$_{+26.90}$	9.13×$_{+8.13}$	33.03$_{+1.56}$	+ Fast-dLLM (Parallel+Cache)	40.98$_{+35.55}$	7.54×$_{+6.54}$	32.37$_{-3.56}$
	+ SlowFast + Cache	29.06$_{+25.75}$	8.78×$_{+7.78}$	33.48$_{+2.01}$	**+ SlowFast + Cache**	39.63$_{+34.20}$	7.30×$_{+6.30}$	34.82$_{-1.11}$
Math	LLaDA	5.14	1.00×	30.16	Dream	8.48	1.00×	38.68
	+ Fast-dLLM (Parallel+Cache)	21.50$_{+16.36}$	4.18×$_{+3.18}$	28.34$_{-1.82}$	+ Fast-dLLM (Parallel+Cache)	46.80$_{+38.32}$	5.52×$_{+4.52}$	33.24$_{-5.44}$
	+ SlowFast + Cache	26.50$_{+21.36}$	5.16×$_{+4.16}$	29.42$_{-0.74}$	**+ SlowFast + Cache**	56.44$_{+47.96}$	6.66×$_{+5.66}$	37.10$_{-1.58}$
General Tasks								
MMLU-pro	LLaDA	9.16	1.00×	23.30	Dream	14.97	1.00×	24.14
	+ Fast-dLLM (Parallel+Cache)	27.85$_{+18.69}$	3.04×$_{+2.04}$	27.66$_{+4.36}$	+ Fast-dLLM (Parallel+Cache)	42.73$_{+27.76}$	2.85×$_{+1.85}$	22.57$_{-1.57}$
	+ SlowFast + Cache	33.38$_{+24.22}$	3.64×$_{+2.64}$	25.53$_{+2.23}$	**+ SlowFast + Cache**	43.50$_{+28.53}$	2.91×$_{+1.91}$	21.81$_{-2.33}$
MMLU	LLaDA	5.02	1.00×	62.11	Dream	8.46	1.00×	72.61
	+ Fast-dLLM (Parallel+Cache)	32.36$_{+27.34}$	6.45×$_{+5.45}$	61.45$_{-0.66}$	+ Fast-dLLM (Parallel+Cache)	44.73$_{+36.27}$	5.28×$_{+4.28}$	64.46$_{-8.15}$
	+ SlowFast + Cache	38.42$_{+33.40}$	7.65×$_{+6.65}$	61.20$_{-0.91}$	**+ SlowFast + Cache**	44.18$_{+35.72}$	5.22×$_{+4.22}$	71.57$_{-1.04}$
BBH	LLaDA	4.04	1.00×	44.97	Dream	6.93	1.00×	51.83
	+ Fast-dLLM (Parallel+Cache)	22.56$_{+18.52}$	5.58×$_{+4.58}$	45.35$_{+0.38}$	+ Fast-dLLM (Parallel+Cache)	57.02$_{+50.09}$	8.22×$_{+7.22}$	44.95$_{-6.88}$
	+ SlowFast + Cache	36.04$_{+32.00}$	8.92×$_{+7.92}$	44.81$_{-0.16}$	**+ SlowFast + Cache**	70.20$_{+63.27}$	10.13×$_{+9.13}$	48.24$_{-3.59}$
Code								
MBPP	LLaDA	4.98	1.00×	40.80	Dream	8.92	1.00×	54.20
	+ Fast-dLLM (Parallel+Cache)	22.18$_{+17.20}$	4.45×$_{+3.45}$	38.20$_{-2.60}$	+ Fast-dLLM (Parallel+Cache)	50.67$_{+41.75}$	5.68×$_{+4.68}$	50.40$_{-3.80}$
	+ SlowFast + Cache	27.26$_{+22.28}$	5.47×$_{+3.87}$	39.00$_{-1.80}$	**+ SlowFast + Cache**	69.48$_{+60.56}$	7.79×$_{+6.79}$	51.00$_{-3.20}$
HumanEval	LLaDA	11.24	1.00×	31.71	Dream	11.49	1.00×	34.15
	+ Fast-dLLM (Parallel+Cache)	26.25$_{+15.01}$	2.34×$_{+1.34}$	29.88$_{-1.83}$	+ Fast-dLLM (Parallel+Cache)	49.47$_{+37.98}$	4.31×$_{+3.31}$	29.27$_{-4.88}$
	+ SlowFast + Cache	41.14$_{+29.90}$	3.66×$_{+2.66}$	31.10$_{-0.61}$	**+ SlowFast + Cache**	47.86$_{+36.37}$	4.17×$_{+3.17}$	35.36$_{+1.21}$

Sampling Strategy	TPS↑	Score↑
Autoregressive (AR)	5.25	60.80
Diffusion Sampling	4.55	69.83
Semi-Autoregressive	5.44	66.41
SlowFast Sampling	9.87	69.59

Table 3: **Comparison of Sampling Strategies** on inference efficiency and generation performance.

Method	TPS↑	Speed(TPS)↑	Accuracy↑
LLaMA3 8B (Dubey et al., 2024)	33.79	21.12×	31.92
LLaDA	1.60$_{-32.19}$	1.00×	31.47$_{-0.45}$
+ SlowFast	25.00$_{-8.79}$	15.63×	31.47$_{-0.45}$
+ SlowFast + Cache ($K_p = 100, K_r = 5$)	48.80$_{+15.01}$	30.50×	30.13$_{-1.79}$
+ SlowFast + Cache ($K_p = 500, K_r = 30$)	54.75$_{+20.96}$	34.22×	28.79$_{-3.13}$

Table 4: **Comparison of LLaDA 8B Base with other representative LLMs**. Compared to LLaMA3 8B, LLaDA with SlowFast Sampling and dLLM-Cache achieves significantly higher throughput (up to +20.96 TPS) while maintaining comparable accuracy.

$W_{hist} = 2$, and stable variance threshold $\sigma^2_{stable} = 1.0$, consistent with the default configuration used in Section 4.2 and ablation studies.

Compatibility with dLLM-Cache. Recently, several studies have explored leveraging feature cache to reduce the computational cost of dLLM inference. Our SlowFast Sampling is highly compatible with existing caching mechanisms and the integration can lead to higher acceleration compared to using either alone. Table 2 compares the performance and inference speed of LLaDA with and without applying our method in combination with dLLM-Cache. The results show that this integration can deliver higher throughput while maintaining comparable model performance.

Comparison with Other Sampling Strategies. We compared our method SlowFast Sampling with four alternative sampling strategies: diffusion sampling, diffusion semi-autoregressive sampling, autoregressive (AR) sampling and Fast-dLLM (Wu et al., 2025). Diffusion sampling adopts a remasking strategy to iteratively select token to decode in parallel, while AR sampling generates tokens strictly from left to right. The semi-autoregressive approach generates blocks left-to-right and applies the remasking strategy within each block. Fast-dLLM, in contrast, performs parallel decoding only based on token confidence, and we report results for both its parallel and cache-augmented variants. As shown in Table 1, 2 and 3, our method SlowFast Sampling achieves higher inference efficiency while maintaining competitive generation quality.

4.3 Ablation Study

Case Study of SlowFast Sampling in Sentence Generation. As shown in Figure 4, this case study shows how SlowFast Sampling generates text by alternating slow and fast phases. In the slow phases, the model outputs a few reliable tokens such as subjects, verbs, or punctuation to anchor the sentence. The fast phases then produce longer spans of high-confidence tokens in one step, e.g., "*she has 9 - 4 = 5 yuan left*", ensuring both efficiency and correctness. This balance of cautious slow decoding

and confident fast decoding maintains accuracy while accelerating generation, making the approach intuitive and effective.

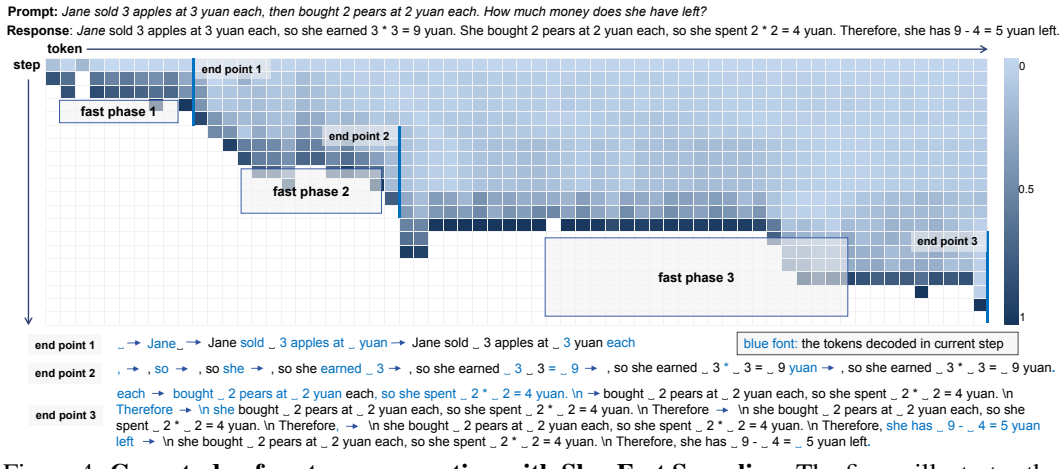


Figure 4: **Case study of sentence generation with SlowFast Sampling.** The figure illustrates the dynamic evolution of the confidence map across three phases, where exploratory and accelerated decoding are alternated. Blue tokens denote those generated in the current step, highlighting how the method gradually builds on high-confidence spans to produce a coherent final response.

Effect of Hyperparameters in the Stability Check. The stability check, which transitions the model from the slow to the fast phase, is governed by three hyperparameters whose effects are shown in Figure 5. The maximum number of exploratory steps, K_{max}, is for allowing the convergence horizon to stabilize; we find $K_{max} = 8$ provides sufficient exploration for high-quality generation without incurring excessive overhead. This is complemented by the sliding window size, W_{hist}. Since prediction changes and convergence occur rapidly, we find a smaller window of $W_{hist} = 2$ is effective, striking a strong balance between maintaining quality and maximizing inference speed. Finally, a strict stable variance threshold of $\sigma^2_{stable} = 1.0$ ensures that the accelerated phase is only triggered for genuinely stable regions, solidifying the reliability of our method.

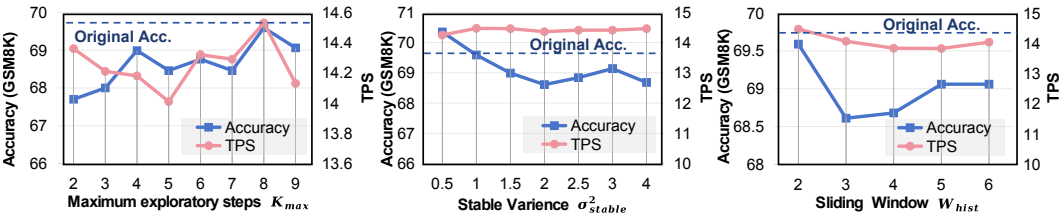


Figure 5: **The sensitivity study on hyper-parameters in the stability-Check.** Accuracy and TPS vary with K_{max}, σ^2_{stable}, and W_{hist}. The chosen defaults ($K_{max} = 8$, $\sigma^2_{stable} = 1.0$, $W_{hist} = 2$) offer strong speed-quality trade-offs.

Outperforming Autoregressive LLMs in Inference Speed. As shown in Table 4, when equipped with our SlowFast Sampling and dLLM-Cache, LLaDA Base not only accelerates significantly over its default setting but also **outperforms the autoregressive LLaMA3 8B** (Dubey et al., 2024) in inference speed (54.75 vs. 33.79 TPS), while maintaining comparable accuracy. This demonstrates that dLLMs, with proper sampling and optimization, can surpass traditional autoregressive models in both efficiency and practicality.

5 Conclusion

In this work, we present **SlowFast Sampling**, a dynamic and principled approach to accelerate diffusion-based large language models (dLLMs). By leveraging three key observations: token certainty, convergence, and positional bias, we design a two-stage decoding pipeline that adaptively balances exploration and efficient parallel decoding. Extensive experiments across benchmarks demonstrate that our method not only significantly improves inference speed (up to **15.63**× and up to **34.22**× combined with dLLM-Cache on the GPQA benchmark), but also maintains strong generation quality, outperforming even autoregressive LLMs in throughput. We believe this work marks an important step toward making dLLMs practical and competitive in real-world deployment.

Reproducibility Statement

We have made every effort to ensure the reproducibility of our results. The main text (Sections 3–4) provides detailed descriptions of the proposed SlowFast Sampling method, its integration with dLLM-Cache, and the corresponding experimental setups. Hyperparameter configurations and additional implementation details are presented in Appendix A.3. Comprehensive performance comparisons across benchmarks are reported in Tables 1–4. Ablation studies and a case study are provided in Section 4.3 (e.g., Figure 4) to illustrate the robustness of our method. To facilitate replication, we include the source code and scripts for running experiments in the supplementary materials.

Ethics Statement

This work focuses on developing efficient sampling strategies for diffusion-based large language models (dLLMs). Our research does not involve human subjects, personal or sensitive data, or applications directly impacting individuals. All experiments were conducted on publicly available benchmarks, ensuring fairness, reproducibility, and transparency. We release code to promote open research and to enable verification of our results. We are not aware of any potential misuse or harmful applications of the proposed methods beyond standard concerns inherent to language models, such as bias or misuse in downstream tasks, and we encourage responsible use in line with the ICLR Code of Ethics.

References

Marianne Arriola, Aaron Gokaslan, Justin T Chiu, Zhihan Yang, Zhixuan Qi, Jiaqi Han, Subham Sekhar Sahoo, and Volodymyr Kuleshov. Block diffusion: Interpolating between autoregressive and diffusion language models. *arXiv preprint arXiv:2503.09573*, 2025.

Jacob Austin, Daniel D Johnson, Jonathan Ho, Daniel Tarlow, and Rianne Van Den Berg. Structured denoising diffusion models in discrete state-spaces. *Advances in Neural Information Processing Systems*, 34:17981–17993, 2021.

Lukas Berglund, Meg Tong, Max Kaufmann, Mikita Balesni, Asa Cooper Stickland, Tomasz Korbak, and Owain Evans. The reversal curse: Llms trained on" a is b" fail to learn" b is a". *arXiv preprint arXiv:2309.12288*, 2023.

Andrew Campbell, Joe Benton, Valentin De Bortoli, Thomas Rainforth, George Deligiannidis, and Arnaud Doucet. A continuous time framework for discrete denoising models. *Advances in Neural Information Processing Systems*, 35:28266–28279, 2022.

Huiwen Chang, Han Zhang, Lu Jiang, Ce Liu, and William T Freeman. Maskgit: Masked generative image transformer. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pp. 11315–11325, 2022.

Abhimanyu Dubey, Abhinav Jauhri, Abhinav Pandey, Abhishek Kadian, Ahmad Al-Dahle, Aiesha Letman, Akhil Mathur, Alan Schelten, Amy Yang, Angela Fan, et al. The llama 3 herd of models. *arXiv preprint arXiv:2407.21783*, 2024.

Jonathan Ho, Ajay Jain, and Pieter Abbeel. Denoising diffusion probabilistic models. *Advances in neural information processing systems*, 33:6840–6851, 2020.

Emiel Hoogeboom, Didrik Nielsen, Priyank Jaini, Patrick Forré, and Max Welling. Argmax flows and multinomial diffusion: Learning categorical distributions. *Advances in Neural Information Processing Systems*, 34:12454–12465, 2021.

Zhiyuan Liu, Yicun Yang, Yaojie Zhang, Junjie Chen, Chang Zou, Qingyan Wei, Shaobo Wang, and Linfeng Zhang. dllm-cache: Accelerating diffusion large language models with adaptive caching, 2025. URL `https://github.com/maomaocun/dLLM-cache`.

Aaron Lou, Chenlin Meng, and Stefano Ermon. Discrete diffusion language modeling by estimating the ratios of the data distribution. *arXiv preprint arXiv:2310.16834*, 2023.

Xinyin Ma, Runpeng Yu, Gongfan Fang, and Xinchao Wang. dkv-cache: The cache for diffusion language models, 2025. URL https://arxiv.org/abs/2505.15781.

Shen Nie, Fengqi Zhu, Chao Du, Tianyu Pang, Qian Liu, Guangtao Zeng, Min Lin, and Chongxuan Li. Scaling up masked diffusion models on text, 2025a. URL https://arxiv.org/abs/2410.18514.

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, 2025b. URL https://arxiv.org/abs/2502.09992.

William Peebles and Saining Xie. Scalable diffusion models with transformers. In *Proceedings of the IEEE/CVF International Conference on Computer Vision*, pp. 4195–4205, 2023.

Robin Rombach, Andreas Blattmann, Dominik Lorenz, Patrick Esser, and Björn Ommer. High-resolution image synthesis with latent diffusion models. In *Proceedings of the IEEE/CVF conference on computer vision and pattern recognition*, pp. 10684–10695, 2022.

Jiaxin Shi, Kehang Han, Zhe Wang, Arnaud Doucet, and Michalis K Titsias. Simplified and generalized masked diffusion for discrete data. *arXiv preprint arXiv:2406.04329*, 2024.

Jascha Sohl-Dickstein, Eric Weiss, Niru Maheswaranathan, and Surya Ganguli. Deep unsupervised learning using nonequilibrium thermodynamics. In *International conference on machine learning*, pp. 2256–2265. PMLR, 2015.

Jiaming Song, Chenlin Meng, and Stefano Ermon. Denoising diffusion implicit models. In *International Conference on Learning Representations*, 2021.

Yuerong Song, Xiaoran Liu, Ruixiao Li, Zhigeng Liu, Zengfeng Huang, Qipeng Guo, Ziwei He, and Xipeng Qiu. Sparse-dllm: Accelerating diffusion llms with dynamic cache eviction, 2025. URL https://arxiv.org/abs/2508.02558.

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, 2025. URL https://arxiv.org/abs/2505.22618.

Jiacheng Ye, Zhihui Xie, Lin Zheng, Jiahui Gao, Zirui Wu, Xin Jiang, Zhenguo Li, and Lingpeng Kong. Dream 7b, 2025. URL https://hkunlp.github.io/blog/2025/dream.

Wayne Xin Zhao, Kun Zhou, Junyi Li, Tianyi Tang, Xiaolei Wang, Yupeng Hou, Yingqian Min, Beichen Zhang, Junjie Zhang, Zican Dong, Yifan Du, Chen Yang, Yushuo Chen, Zhipeng Chen, Jinhao Jiang, Ruiyang Ren, Yifan Li, Xinyu Tang, Zikang Liu, Peiyu Liu, Jian-Yun Nie, and Ji-Rong Wen. A survey of large language models, 2025. URL https://arxiv.org/abs/2303.18223.

Kaiwen Zheng, Yongxin Chen, Hanzi Mao, Ming-Yu Liu, Jun Zhu, and Qinsheng Zhang. Masked diffusion models are secretly time-agnostic masked models and exploit inaccurate categorical sampling. *arXiv preprint arXiv:2409.02908*, 2024.

A Appendix

A.1 LLM Usage Statement.

We used a Large Language Model (LLM) solely to aid in polishing the writing and improving the clarity of language. The LLM was not involved in research ideation, experimental design, data analysis, or generation of scientific content. All research ideas, methods, analyses, and conclusions presented in this paper are entirely the work of the authors. The authors take full responsibility for the content of this paper.

A.2 Compatibility with Advanced Sampling Methods.

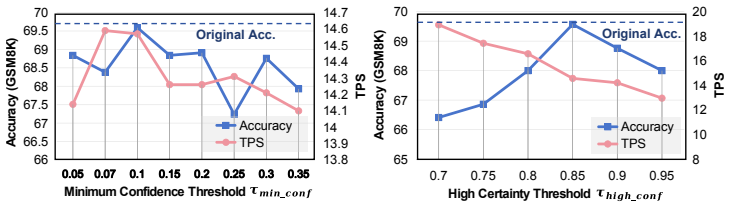


Figure 6: **Effect of Confidence Thresholds on GSM8K.** τ_{min_conf} controls the exploratory range, and τ_{high_conf} balances accuracy and speed during fast decoding. Other hyperparameters follow the default settings in Section 4.2.

Sensitivity to Confidence Thresholds. The core confidence thresholds, τ_{min_conf} and τ_{high_conf}, dictate the behavior of our SlowFast pipeline. As shown in Figure 6, the minimum confidence threshold τ_{min_conf} defines the candidate region in the exploratory stage. A moderate value of $\tau_{min\_conf} = 0.1$ is optimal, as lower values lead to unstable regions and higher values create overly conservative, inefficient regions. The high certainty threshold τ_{high_conf} directly manages the speed-quality trade-off during the accelerated stage. A higher value leads to more cautious and accurate generation at the cost of speed (TPS). We select $\tau_{high\_conf} = 0.85$, which achieves a near-peak GSM8K score without a drastic reduction in inference speed, striking an effective balance.

A.3 Experimental Details

To ensure fair and reproducible comparisons, we standardize the setting of inference generation across benchmarks and run each model with its officially recommended settings. The benchmark-specific parameters considered include the number of inference steps, the block length, the total generation length, and the number of few-shot examples. For clarity, we summarize the settings for LLaDA and Dream in Tables 5 and 6, respectively.

Table 5: Experimental settings for LLaDA across benchmarks.

Task	Steps	Block Length	Generation Length	Few-shot
MMLU	3	3	3	5
MMLU-pro	256	256	256	0
GSM8K	256	256	256	4
Math	256	256	256	0
GPQA	256	256	256	5
HumanEval	256	256	256	0
MBPP	256	256	256	3
BBH	256	256	256	3

Table 6: Experimental settings for Dream across benchmarks.

Task	Steps	Generation Length	Few-shot
MMLU	3	3	5
MMLU-pro	256	256	0
GSM8K	256	256	8
Math	256	256	4
GPQA	256	256	5
HumanEval	256	256	0
MBPP	256	256	3
BBH	256	256	3

A.4 Ablation Study on Hyperparameters and Robustness.

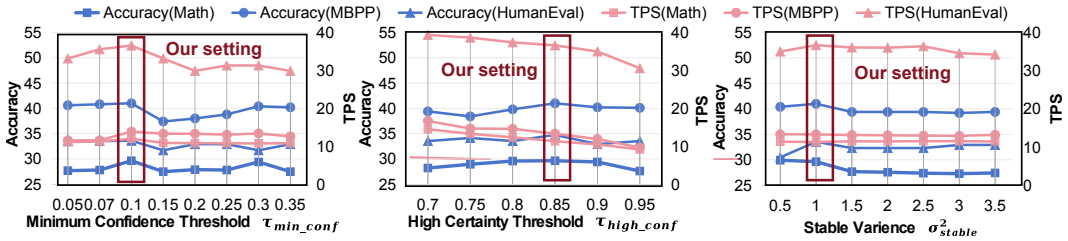


Figure 7: **Parameter Robustness and Sensitivity Across Multiple Benchmarks.** Figure shows sensitivity across MATH, MBPP, and HumanEval. Our setting is shown to be highly robust across all tasks.

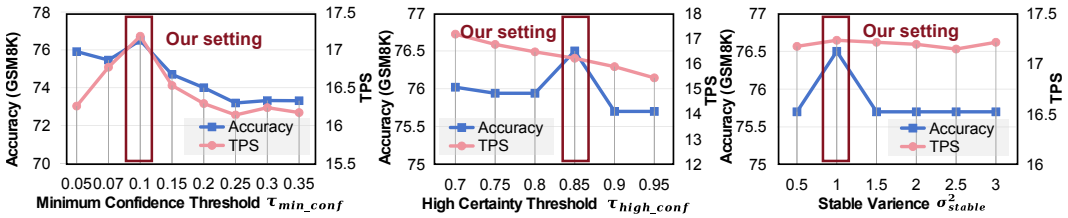


Figure 8: **Parameter Stability Across Model Variants on Dream 7B.** Sensitivity analysis for τ_{min_conf}, τ_{high_conf}, and σ^2_{stable} was conducted on Dream 7B using the GSM8K benchmark. The findings confirm that the unified set of hyperparameters successfully transfers to this demanding model variant, validating overall parameter stability.

Hyperparameter Robustness and Generalization Analysis. The overall hyperparameter analysis confirms the robustness and strong generalization ability of the SlowFast approach. As shown in Figure 7, a single, unified parameter set maintains near-optimal performance across diverse benchmarks (MATH, MBPP, HumanEval). This stability is further validated by Figure 8, where the same unified settings successfully transfer to Dream 7B on the GSM8K benchmark. These findings confirm that the SlowFast approach is highly robust and does not require extensive tuning for new tasks or model variants.

A.5 Performance-Efficiency Trade-off Across Strategies

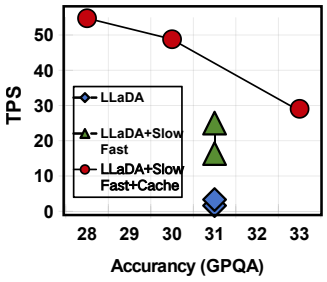


Figure 9: **Performance Efficiency Trade-off on GPQA.**

Figure 9 evaluates the Performance-Efficiency for our method against various baselines under two distinct evaluation setups. The results confirm that the integration of our method consistently shifts the trade-off curve decisively towards the top-right corner (higher TPS and higher Accuracy) compared to the baseline. This superiority holds true across both the standard configuration (5-shot, 256 tokens) and the more demanding scenario (8-shot, 1024 tokens), validating the robustness and scalability of our adaptive sampling strategy.

A.6 Case Studies on Positional Principle and Non-Contiguous Decoding.

Robustness to Minor Positional Uncertainty. Figure 10 confirms SlowFast's ability to cover multiple stable blocks even when separated by unstable regions. Crucially, the distance between the end of one stable block and the start of the next is generally small, allowing our adaptive mechanism to efficiently jump over the intermediate low-confidence span. This ensures that the overall acceleration is maintained without sacrificing accuracy, even when faced with minor positional uncertainties.

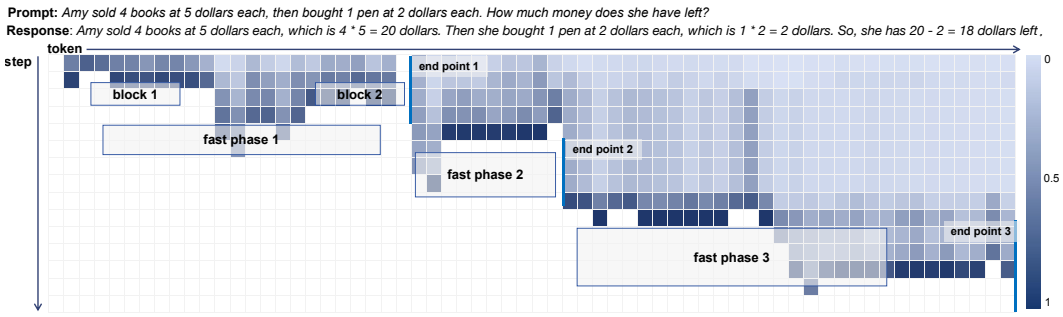


Figure 10: **Case Study I: Dynamic Span Coverage During Positional Jumps.** This case study illustrates a scenario where the model exhibits a positional jump between decoded tokens, confirming that the dynamic search window of SlowFast successfully identifies and covers the subsequent stable region.

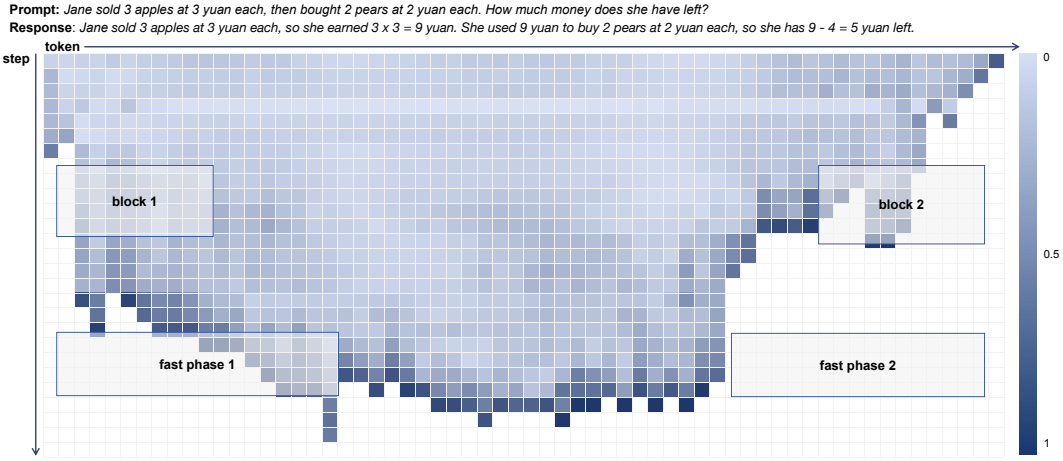


Figure 11: **Case Study II: SlowFast's Adaptive Response to Induced Non-Contiguous Blocks.** This case study demonstrates the adaptive capability of the SlowFast framework. By manually inducing a positional jump, the figure shows how the dynamic adjustment mechanism immediately recognizes and proceeds to the subsequent high-confidence region.

Adaptive Handling of Non-Contiguous Blocks. Figure 11 rigorously tests SlowFast's adaptive limits by manually constructing an extreme scenario featuring two widely separated high-confidence blocks. The visualization confirms that upon encountering the unstable region between the blocks, the dynamic adjustment mechanism successfully identifies the subsequent high-confidence region (Block 2) and resumes acceleration (Fast Phase 2). This successful navigation demonstrates the robustness of SlowFast's dynamic scheduling, confirming its ability to automatically identify and accelerate multiple non-contiguous stable blocks even under extreme conditions.

A.7 Algorithm: SlowFast Sampling for dLLMs

Algorithm 1 outlines the SlowFast Sampling Strategy, a dynamic, two-phase framework for accelerating dLLMs. It starts with the Slow, Exploratory Stage, which proceeds step-by-step (Lines 10-12) to determine a safe decoding span boundary. This boundary is validated by a Variance-based Stability Check (Lines 16-22), confirming stability when the variance of endpoint predictions falls below σ^2_{stable}. Once stability is confirmed, the algorithm enters the Fast, Accelerated Decoding Stage. Here, acceleration is achieved via In-Span Parallel Decoding (Lines 40-47), where high-confidence tokens within the span are unmasked simultaneously. Additionally, Out-of-Span Caching (Lines 34-39) prepares for the subsequent cycle. This adaptive, two-stage mechanism ensures maximum efficiency is achieved only in reliably stable regions.

Algorithm 1 SlowFast Sampling Strategy

Require: Masked sequence $y^{(N)}$, Prompt c, Max steps N, Exploratory limit K_{max}, History window W_{hist}, Stability threshold σ^2_{stable}, Confidence thresholds $\tau_{min\_conf}, \tau_{high\_conf}$.
Ensure: Generated sequence $y^{(0)}$.

$k \leftarrow N$ ▷ Initialize diffusion step
$s_{cycle} \leftarrow 1$ ▷ Start of the current decoding cycle
$H_W \leftarrow \emptyset$ ▷ History of convergence points
while $k > 0$ **and** $s_{cycle} < L$ **do**
  // Phase 1: Exploratory Stage (Slow)
  $step\_count \leftarrow 0$
  $is\_stable \leftarrow$ False
  while $step\_count < K_{max}$ **and** $k > 0$ **do**
    Predict $\hat{r}_0^{(k)}$ and confidence $P_\theta(\cdot|c, y^{(k)})$
    Cautious Decoding:
    Unmask Top-k_{slow} tokens in range $[s_{cycle}, L]$
    End Point Prediction:
    $e^{(k)}_{cand} \leftarrow \max\{i \mid i \in [s_{cycle}, L] \wedge P_\theta(\hat{r}^{(k)}_{0,i}) > \tau_{min\_conf}\}$
    Update history window H_W with $e^{(k)}_{cand}$
    Stability Check:
    $Var_{curr} \leftarrow$ Variance(H_W)
    if $|H_W| \geq W_{hist}$ **and** $Var_{curr} < \sigma^2_{stable}$ **then**
      $is\_stable \leftarrow$ True
      $e_{cycle} \leftarrow$ Mean(H_W)
      $k \leftarrow k - 1$
      break ▷ Exit slow phase early
    end if
    Update state $y^{(k-1)}$ using standard scheduler
    $k \leftarrow k - 1$
    $step\_count \leftarrow step\_count + 1$
  end while
  if not is_stable **then**
    $e_{cycle} \leftarrow e^{(k)}_{cand}$ ▷ Fallback if no convergence
  end if
  // Phase 2: Accelerated Decoding Stage (Fast)
  if $k > 0$ **then**
    Predict $\hat{r}_0^{(k)}$ and confidence $P_\theta(\cdot|c, y^{(k)})$
    Out-of-Span Caching:
    for $i \in (e_{cycle}, L]$ **do**
      if $P_\theta(\hat{r}^{(k)}_{0,i}) < \tau_{min\_conf}$ **then**
        Cache features for position i
      end if
    end for
    In-Span Parallel Decoding:
    $S_{high} \leftarrow \{i \mid i \in [s_{cycle}, e_{cycle}] \wedge P_\theta(\hat{r}^{(k)}_{0,i}) > \tau_{high\_conf}\}$
    if $S_{high} \neq \emptyset$ **then**
      $y_i^{(k-1)} \leftarrow \hat{r}^{(k)}_{0,i}$ for all $i \in S_{high}$ ▷ Aggressive unmasking
    else
      Unmask Top-k_{fast} tokens in $[s_{cycle}, e_{cycle}]$ ▷ Fallback refinement
    end if
    $s_{cycle} \leftarrow e_{cycle}$ ▷ Advance the stable region start
    $k \leftarrow k - 1$
  end if
end while
return $y^{(0)}$