

Breaking Block Boundaries: Anchor-based History-stable Decoding for Diffusion Large Language Models

Shun Zou^{1*}, Yong Wang^{2†‡}, Zehui Chen¹, Lin Chen¹,
Chongyang Tao, Feng Zhao^{1‡}, Xiangxiang Chu²

¹University of Science and Technology of China

²AMAP, Alibaba Group

<https://github.com/zs1314/AHD>

Abstract

Diffusion Large Language Models (dLLMs) have recently become a promising alternative to autoregressive large language models (ARMs). Semi-autoregressive (Semi-AR) decoding is widely employed in base dLLMs and advanced decoding strategies due to its superior performance. However, our observations reveal that Semi-AR decoding suffers from inherent block constraints, which cause the decoding of many cross-block stable tokens to be unnecessarily delayed. To address this challenge, we systematically investigate the identification of stable tokens and present three key findings: (1) naive lookahead decoding is unreliable, (2) token stability closely correlates with convergence trend, and (3) historical information is isolated. Building on these insights, we propose **Anchor-based History-stable Decoding (AHD)**, a training-free, plug-and-play dynamic decoding strategy. Specifically, AHD monitors the stability trend of tokens in real time through dynamic anchors. Once a token reaches stability, it initiates early cross-block decoding to enhance efficiency and performance. Extensive experiments across language, vision-language, and audio-language domains demonstrate that AHD simultaneously improves both performance and inference efficiency. Notably, AHD effectively reverses the performance degradation typically observed in existing advanced decoding acceleration strategies. For instance, on the BBH benchmark, our approach reduces decoding steps by **80%** while improving performance by **3.67%**.

1 Introduction

Recently, diffusion large language models (dLLMs) have attracted extensive attention from the community and become a strong alternative to autoregressive large language models (ARMs) (Nie et al., 2025; Ye et al., 2025). In contrast to the next-token prediction mechanism employed by

ARMs, dLLMs utilize bidirectional attention and iterative decoding, facilitating efficient parallel decoding, flexible generation order, and controllable output (Berglund et al., 2024). In recent work, semi-autoregressive (Semi-AR) decoding has been widely adopted in base dLLMs (Yang et al., 2025; Nie et al., 2025; Zhu et al., 2025; Bie et al., 2025), and numerous advanced decoding strategies have been developed based on this framework (Wei et al., 2026; Wang et al., 2025a; Dong et al., 2025; Wu et al., 2025; Ma et al., 2025b; Wang et al., 2025b).

However, Semi-AR decoding requires the completion of the current block prior to initiating the subsequent one, thereby limiting the potential for parallel decoding. To investigate this constraint, we conducted a systematic study and obtained the following key observations: *(i) Block-Boundary Delay is Prevalent.* Specifically, Figure 1 (a) and (b) reveal that many tokens already match the final output before their corresponding blocks are decoded and maintain stability until they are eventually unmasked. We define such tokens as cross-block stable tokens. However, the sequential nature of block decoding introduces unnecessary delays for these stable tokens, wasting many decoding steps. *(ii) Cross-block Stable Tokens Exhibit Radiative Effects.* Figure 1 (c) illustrates that the initial stable state of a token often emerges in a clustered pattern (i.e., the first stable step). This indicates that once a token reaches stability, it exerts an influence on neighboring tokens, accelerating their transition into stable states. Conversely, delaying the decoding of a stable token suppresses the entire local region, leading to performance degradation (Lu et al., 2026; Wang et al., 2025b).

Therefore, we urgently need to free these stable tokens from the sequential block constraint, which is crucial for both inference efficiency and performance. This raises a key question: *how can we accurately identify cross-block stable tokens?* Previous studies have often relied on metrics such as

* Work done during the internship at AMAP, Alibaba Group. † Project lead. ‡ Corresponding author.

confidence and entropy to determine whether a token should be decoded (Nie et al., 2025; Wu et al., 2025; Ben-Hamu et al., 2025). Therefore, we systematically examine the evolution of the confidence metric for cross-block stable tokens during decoding, and identify two critical insights: **(I) Naive Lookahead Decoding is unreliable.** As shown in Figure 2 (a), even these tokens that have already reached stability may still experience local fluctuations in subsequent decoding steps. These temporary variations can potentially result in erroneous judgments. **(II) Token stability closely correlates with convergence trend.** As shown in Figure 2 (b), after the first stable step, the confidence trend of these stable tokens changes sharply, exhibiting a rapid growth pattern that we define as the **absolute stability trend**. This motivates us to redefine token stability in terms of the trajectory across decoding steps. Furthermore, it is more critical to capture the emergence of the absolute stability trend at an early stage than solely monitoring peak values. If tokens are unlocked only when their confidence exceeds a high threshold, the decoding process is forced to wait for many unnecessary steps and could exhibit the phenomenon outlined in *insight I*.

Based on the above insights, we identify two essential conditions for the precise detection of cross-block token stability: (1) the formulation of a global stability metric that transcends local behavior; and (2) the capability to dynamically monitor trajectory information. To meet these conditions effectively, the integration of historical information is indispensable. However, by examining the decoding characteristics of dLLMs, we find that: **(III) Historical information is isolated in standard decoding.** In conventional iterative decoding within dLLMs, each prediction only depends on the outcome of the immediately preceding step. The final output is determined exclusively by the last decoding step, with no consideration of earlier iterations. Therefore, we argue that breaking this historical isolation by integrating trajectory information is essential for identifying these stable tokens.

In response to these insights, we propose **Anchor-based History-stable Decoding (AHD)**, a training-free, plug-and-play dynamic decoding approach for monitoring cross-block stability. Specifically, at each decoding step, we take the current step as a dynamic anchor and retrospectively examine prior steps in the historical buffer. Through this process, we establish a stability criterion grounded in historical consistency. This allows us to cap-

ture the absolute stability trend at an early stage, avoiding additional decoding steps, and improving robustness to local fluctuations. Moreover, the early unlocking of these cross-block stable tokens preserves the coherence of the decoding, thereby enhancing generation quality. Our contributions are summarized as follows:

- We systematically analyze the key limitations of Semi-AR decoding and derive three critical insights, which serves as the motivation and key evidence for determining cross-block stability.
- Based on the analysis, we propose AHD, a novel cross-block stability decoding method. It uses dynamic anchors to trace back historical trajectories and capture absolute stability trends, enabling early unlocking of cross-block stable tokens. We also provide theoretical proof of the method.
- Extensive experiments on language, vision-language, and audio-language domains show that AHD achieves consistent improvements in both performance and inference efficiency, significantly surpassing previous decoding strategies.

2 Preliminary

In this section, we present an overview of the inference process in dLLMs. Due to space constraints, additional related work is provided in Appendix A.

Inference Process of dLLMs. During inference, diffusion large language models (dLLMs) generate a sequence by iteratively denoising from an initial all-mask state (Nie et al., 2025). Given a target sequence length L , the model initializes

$$\mathbf{x}^T = ([\text{MASK}], \dots, [\text{MASK}]) \in \mathcal{V}^L, \quad (1)$$

and performs T reverse diffusion steps to reconstruct the final output \mathbf{x}^0 . At each timestep t , the model performs a forward pass on the current state \mathbf{x}^t , yielding token distributions \mathbf{p}_j^t over the vocabulary for each position $j = 1, \dots, L$. Subsequently, a confidence-aware remasking mechanism selects a subset of positions \mathcal{M}_{t-1} such that:

$$|\mathcal{M}_{t-1}| \approx \alpha_{t-1}L, \quad (2)$$

where $\alpha_{t-1} \in [0, 1]$ is the target masking ratio at step $t - 1$. For each position j , a confidence score is computed by $s_j^t = \max_{v \in \mathcal{V}} \mathbf{p}_j^t(v)$.

Positions with the lowest confidence scores are iteratively added to the masked token set \mathcal{M}_{t-1} until the desired number of masks is reached. The

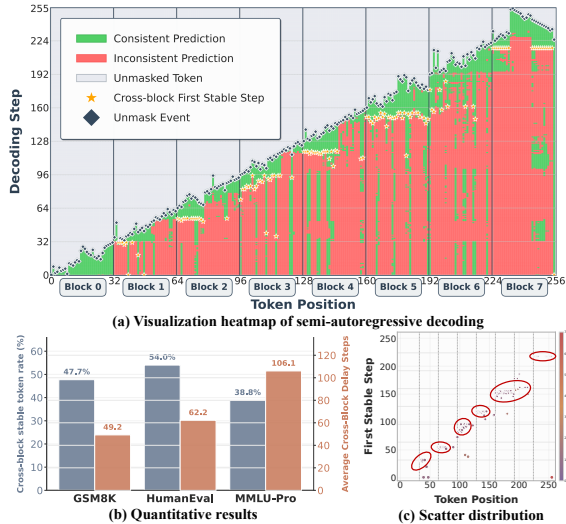


Figure 1: Systematic analysis of Semi-AR decoding on dLLMs. (a) Heatmap visualization on a sample. Stars indicate the earliest step at which cross-block tokens reach stability, while black squares denote the finally unmasked step. For instance, position 65 is expected to be decoded no earlier than step 64; however, it stabilizes as early as step 53. (b) Quantitative analysis of cross-block delayed decoding. (c) Distribution of the first stable step.

remaining positions are updated to their predicted tokens (e.g., via greedy decoding). This process is repeated for each step until $t = 0$, at which point the final sequence \mathbf{x}^0 is obtained.

Semi-autoregressive Decoding. To further enhance quality and controllability, the Semi-AR decoding has been widely adopted in base dLLMs (Nie et al., 2025; Yang et al., 2025; Zhu et al., 2025; Wang et al., 2025c; Xin et al., 2025). Recent advances in decoding algorithms often build upon this paradigm (Wu et al., 2025; Dong et al., 2025; Wang et al., 2025a; Wei et al., 2026; Kong et al., 2025; Kim et al., 2026; Li et al., 2026; Ma et al., 2025b). Its core principle is to partition the output sequence into multiple blocks and decode them sequentially from left to right. Within each block, the aforementioned diffusion decoding strategy is applied. Although this method constrains the generation order, it consistently achieves superior performance across various tasks.

3 Method

3.1 Key Limitation of Semi-AR Decoding

Semi-AR decoding has been widely adopted across various base dLLMs and advanced decoding strategies. However, its architecture supports parallel generation only within individual blocks, while requiring sequential processing across blocks. This

structural constraint inherently limits the parallel decoding potential of dLLMs. To address this, we first conducted a systematic analysis of Semi-AR decoding and identified several key observations, which serve as the motivation and foundation of our proposed method.

Core observation 1: Block-boundary Delay is Prevalent.

First, we conducted experiments using LLaDA-8B-Instruct (Nie et al., 2025) on a representative sample from GSM8K (Cobbe et al., 2021). As shown in Figure 1 (a), we observe that many tokens already match their final predicted values before their corresponding blocks are decoded, and these tokens remain unchanged until eventually being unmasked. We refer to such tokens as *cross-block stable tokens*. To further quantify this phenomenon, we further evaluated three widely-used benchmarks (Chen et al., 2021; Wang et al., 2024b; Cobbe et al., 2021). As shown in Figure 1 (b), all three benchmarks contain many tokens whose decoding is delayed by block boundaries, resulting in significant waste of decoding steps.

Core observation 2: Cross-block Stable Tokens Exhibit Radiative Effects.

As shown in Figure 1 (c), we further visualize the temporal and spatial distribution of the *first stable step* (marked as stars in Figure 1 (a)). It is defined as the earliest decoding step at which a token reaches stability. We observe that the onset of stable tokens is not randomly dispersed but instead forms noticeable clusters. Specifically, once a token achieves stability, adjacent tokens tend to stabilize in subsequent steps. We interpret these findings as follows: during iterative decoding, stable tokens function as semantic anchors, exerting radiative and attractive influences on surrounding tokens. The delayed decoding of these stable tokens suggests that block boundaries disrupt local inference within the block.

3.2 Prerequisites for Cross-Block Decoding

Based on above analysis, it is evident that Semi-AR decoding incurs significant block-boundary delays. To address this, a natural objective is to accelerate the release of stable tokens, which are delayed during decoding, thereby reducing overall decoding steps and improving inference coherence. This raises a key question: *How can cross-block stable tokens be effectively and accurately identified?*

Core insight 1: Naive Lookahead Decoding is Unreliable. Previous studies have frequently em-

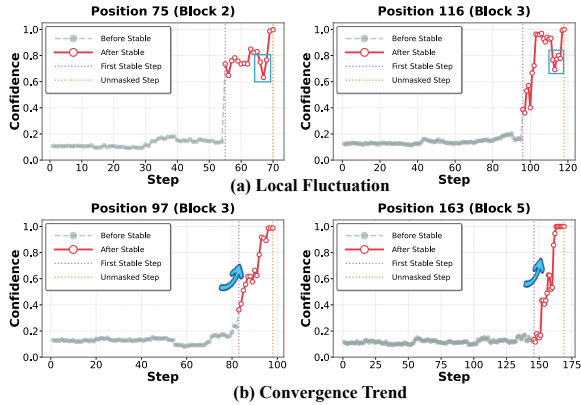


Figure 2: Dynamics of confidence during the decoding. (a) Local fluctuations persist even after the token has become stable. (b) Prior to stabilization, the confidence remains flat or oscillatory; however, after crossing the first stable step, it exhibits a marked upward trend.

ployed confidence and entropy as decoding indicators (Wu et al., 2025; Ben-Hamu et al., 2025). Here, we utilize the confidence metric as stability measures for cross-block unlocking, and track the evolution of it. As shown in Figure 2 (a), our observation reveals that stable tokens can still experience brief, large fluctuations in confidence metric even after their first stable step. These low-frequency fluctuations are critical, as they can induce misjudgments, and propagate erroneous signals to subsequent iterations, ultimately affecting the quality of the final output. Moreover, both confidence and entropy metrics are computed based on a single decoding step, rendering them particularly sensitive to local fluctuations.

Core insight 2: Token Stability Closely Correlates with Convergence Trend. In addition to local fluctuations, we also identify another common and critical phenomenon. As illustrated in Figure 2 (b), the confidence curves for cross-block stable tokens typically remain persistently low or oscillatory prior to the first stable step, indicating that the position is still engaged in iterative inference. However, after exceeding this step, the curves enter a distinct “*convergence phase*”, characterized by a sharp and monotonic increase in confidence. We term this phenomenon the *absolute stability trend*. If a token remains unlocked until its confidence exceeds an extremely high threshold, this leads to a significant increase in decoding steps and could exhibit the phenomenon outlined in **insight 1**. To address this, we redefine token stability based on its confidence trajectory throughout the decoding process. Specifically, our approach is to identify and capture the trajectory at the earliest stage when

it exhibits an absolutely stable trend.

Core insight 3: Historical Information is Isolated in Standard Decoding. Building upon above insights, we conclude that two essential conditions must be satisfied to effectively and efficiently detect token stability: (1) the adoption of a global stability metric that transcends single-step evaluation, and (2) the ability to dynamically monitor information and capture the absolute stability trend. To satisfy these conditions, a crucial consideration is the incorporation of historical information. Through analysis of the standard decoding in dLLMs, we observe that each inference step operates independently, disregarding previous predictions. Consequently, the final output is completely determined by the prediction $p_\theta(x^0 | x^1)$ at the last step, while all intermediate predictive distributions during decoding are ignored. Therefore, it is essential to break this inherent characteristic and leverage historical trajectory information.

3.3 Anchor-based History-stable Decoding

Building on the above analysis, we propose **Anchor-based History-stable Decoding (AHD)**, a training-free, plug-and-play decoding strategy. The core idea is to use dynamic anchors to compute a historical consistency stability criterion and capture the *absolute stability trend* at an early stage. This safely unlocks the potential of cross-block parallel decoding and improves the inference coherence.

As shown in Figure 3, at each decoding step t , we first divide the sequence X^t into the current block B_{current}^t and the future block B_{future}^t . In traditional Semi-AR decoding, token generation at step t is restricted to B_{current}^t . Our approach adopts the confidence-aware parallel decoding strategy to define a subset G_c^t within B_{current}^t . Simultaneously, within B_{future}^t , we employ same confidence-aware parallel decoding strategy and AHD to construct a cross-block lookahead decoding subset G_f^t . To capture the dynamic evolution of token distributions across decoding steps, we utilize KL divergence as a metric for distributional change. Specifically, for position j at decoding step t , we define the one-step KL divergence $D_j^t(\text{one-step})$, as follows:

$$\begin{aligned}
 D_j^t(\text{one-step}) &\triangleq D_{KL}(P_j^t || P_j^{t-1}) \\
 &= \sum_{v \in V} P_j^t(v) \log \frac{P_j^t(v)}{P_j^{t-1}(v)}, \quad (3)
 \end{aligned}$$

where $P_j^t(v)$ and $P_j^{t-1}(v)$ represent the probabil-

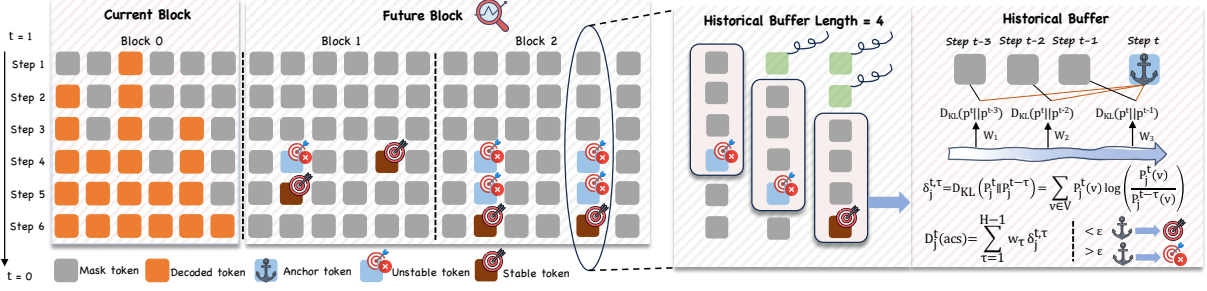


Figure 3: Illustration of Anchor-based History-stable Decoding (AHD). AHD retrospectively tracks historical trajectories from anchor points to dynamically monitor the absolute stability trend. Cross-block stable tokens are unlocked early in the absolute stability trend, reducing decoding steps while further unleashing the potential of parallel diffusion decoding and achieving significant performance gains.

ities of generating token $v \in V$ at position j for decoding steps t and $t - 1$, where V is the vocabulary. However, as demonstrated by our **core insight 1**, $D(\text{one-step})$ is highly sensitive to local fluctuations, which can lead to misjudgments.

Based on **observation 2** and **insights 2 and 3**, we leverage the historical consistency stability criterion to retrospectively monitor the **absolute stability trend** in real time. Specifically, for each position $j \in B_{\text{future}}^t$ within the future block, we maintain a historical buffer of length H , denoted as $\mathcal{H}_j^t = \{P_j^{t-H+1}, \dots, P_j^t\}$. Simultaneously, for each historical buffer, we dynamically designate an anchor point for the current decoding step, denoted as $P_{j,\text{anchor}}^t$, which is formally equivalent to P_j^t . The core idea of the strategy is to employ the dynamic anchor as the evaluation baseline, and retrospectively examine all steps within the historical buffer $\{P_j^{t-H+1}, \dots, P_j^t\}$, excluding the anchor, to evaluate the absolute stability trend within that historical buffer. We formally define the anchored KL divergence as follows:

$$\delta_j^{t,\tau} \triangleq D_{KL}(P_{j,\text{anchor}}^t || P_j^{t-\tau}), \quad (4)$$

where $\tau \in [1, H - 1]$ represents the backtracking step size. Accordingly, we obtain a historical consistency sequence $\{\delta_j^{t,1}, \dots, \delta_j^{t,H-1}\}$. To aggregate evidence across the entire historical window, we compute a weighted sum of the $\delta_j^{t,\tau}$ within the sequence, resulting in the final anchored consistency score $D_j^t(\text{acs})$. Formally, this can be expressed as:

$$D_j^t(\text{acs}) = \sum_{\tau=1}^{H-1} w_\tau \delta_j^{t,\tau}. \quad (5)$$

Here, we assign the exponentially decaying weights w_τ to historical gap τ , thereby assigning greater importance to more recent history. Specifically, the

weight is defined as:

$$w_\tau = \frac{e^{-\lambda\tau}}{Z}, \quad Z = \sum_{k=1}^{H-1} e^{-\lambda k}, \quad (6)$$

where $\lambda > 0$ controls the rate of decay. When $D_j^t(\text{acs})$ falls below the historical consistency threshold ϵ , we consider that, within this historical time window, $P_{j,\text{anchor}}^t$ has already reached an absolutely stable trend and will be incorporated into the G_f^t . Formally, this subset is defined as:

$$G_f^t = \{j | j \in B_{\text{future}} \wedge D_j^t(\text{acs}) < \epsilon\}. \quad (7)$$

Finally, we update the sequence X^t at the corresponding decoding positions from G_{unmasked}^t , which G_{unmasked}^t is the union of G_c^t and G_f^t .

4 Experiments

4.1 Experimental Setup

Implementation Details. In the language domain, we evaluate the proposed method on two widely used dLLMs: LLaDA-8B-Instruct (Nie et al., 2025) and LLaDA-1.5 (Zhu et al., 2025), with a generation length of 256 and a block length of 32. In the vision-language and audio-language domains, we use MMaDA-8B-MixCoT (Yang et al., 2025) and DIFFA (Zhou et al., 2025), respectively. Additionally, the historical consistency threshold ϵ and the historical buffer length H are set to 0.01 and 6 by default. More implementation details are provided in Appendix B.

Datasets and Tasks. Our evaluation in the language domain encompasses three key capability areas across seven benchmarks. For code generation, we utilize HumanEval (Chen et al., 2021) and MBPP (Austin et al., 2021b). For general tasks, we employ BBH (Suzgun et al., 2023), MMLU-Pro (Wang et al., 2024b), and TruthfulQA (Lin et al., 2022). For mathematical reasoning, we

Task	Metric	Vanilla	PC-sampler	Fast-dLLM	KLASS	Saber	Ours
HumanEval	Score \uparrow	40.85	40.24 (-0.61)	41.46 (+0.61)	40.85 (0.00)	36.59 (-4.26)	43.29 (+2.44)
	Step \downarrow	256	256	78.36 (-69%)	131.48 (-49%)	82.83 (-68%)	77.24 (-70%)
MBPP	Score \uparrow	29.20	31.20 (+2.00)	29.40 (+0.20)	29.20 (0.00)	26.00 (-3.20)	31.20 (+2.00)
	Step \downarrow	256	256	70.05 (-73%)	113.26 (-56%)	71.79 (-72%)	66.03 (-74%)
BBH	Score \uparrow	53.11	48.55 (-4.56)	53.17 (+0.06)	53.03 (-0.08)	52.88 (-0.23)	56.78 (+3.67)
	Step \downarrow	256	256	55.85 (-78%)	98.04 (-62%)	87.31 (-66%)	51.48 (-80%)
MMLU-Pro	Score \uparrow	35.57	33.90 (-1.67)	35.70 (+0.13)	35.67 (+0.10)	36.10 (+0.53)	37.42 (+1.85)
	Step \downarrow	256	256	134.76 (-47%)	187.17 (-27%)	123.83 (-52%)	133.06 (-48%)
TruthfulQA	Score \uparrow	40.39	40.39 (0.00)	40.39 (0.00)	40.39 (0.00)	40.76 (+0.37)	41.49 (+1.10)
	Step \downarrow	256	256	67.22 (-74%)	85.56 (-67%)	53.25 (-79%)	52.91 (-79%)
Math	Score \uparrow	33.50	33.62 (+0.12)	33.26 (-0.24)	33.38 (-0.12)	33.48 (-0.02)	33.62 (+0.12)
	Step \downarrow	256	256	96.49 (-62%)	156.72 (-39%)	96.41 (-62%)	88.50 (-65%)
Asdiv	Score \uparrow	75.57	74.19 (-1.38)	74.79 (-0.78)	75.40 (-0.17)	75.18 (-0.39)	77.09 (+1.52)
	Step \downarrow	256	256	65.57 (-74%)	115.35 (-55%)	73.37 (-71%)	61.58 (-76%)

Table 1: Evaluation results of LLaDA-8B-Instruct on seven benchmarks. Values in parentheses denote the performance change relative to the baseline naive decoding and the reduction in decoding steps. The best performance and lowest decoding steps are highlighted in **bold**.

adopt Math (Hendrycks et al., 2021) and Asdiv (Miao et al., 2020). In the vision-language domain, our assessment covers five multimodal benchmarks: MATH-Vision (Wang et al., 2024a), MathVista (Lu et al., 2024), ScienceQA (Lu et al., 2022), GQA (Hudson and Manning, 2019) and MME (Fu et al., 2025). In the audio-language domain, we conduct evaluations on five tasks from VoiceBench (Chen et al., 2024).

Evaluation Metrics. For each benchmark, we employ the standard performance evaluation metrics specified for that dataset. Additionally, to evaluate inference efficiency, we report both the decoding steps and latency (speed-up) as measures of computational speed. A reduction in decoding steps directly indicates the mitigation of redundant computations inherent in traditional dLLMs, thereby corresponding to decreased latency (Wang et al., 2025a; Chen et al., 2025a; Lu et al., 2026).

4.2 Main Results

Language Domain. We present the evaluation results for LLaDA-8B-Instruct and LLaDA-1.5 across seven benchmarks in Tables 1 and 2, comparing baseline approaches, recent advanced decoding strategies, and our proposed AHD. Overall, our method consistently surpasses both naive decoding and existing advanced strategies in terms of both performance and inference efficiency. Notably, most advanced decoding methods achieve inference acceleration at the expense of model performance. Specifically, in rare cases where perfor-

mance does improve, the gains are marginal (typically within 1%), falling within the expected variation and not reflecting a significant enhancement. In contrast, our method achieves consistent performance improvements across all seven benchmarks while simultaneously accelerating inference. For instance, on the BBH benchmark, LLaDA achieves a performance gain of +3.67 points while reducing decoding steps by 80%.

Multimodal Domain. In addition to evaluating text-only models, we further assess multimodal understanding and reasoning. As shown in Table 3, we present both performance metrics and inference efficiency (latency/speed-up) for MMaDA’s naive decoding and AHD across five benchmarks, where the latency (speed-up) of naive decoding is normalized to 1. Overall, AHD demonstrates consistent and stable improvements in both performance and inference efficiency across all five benchmarks. Notably, in multimodal mathematical reasoning, AHD achieves a +3.1 performance gain and 2.37 \times speedup on MathVista, underscoring its applicability to a wide range of multimodal tasks.

Audio Domain. We utilize DIFFA to evaluate semantic audio understanding. As shown in Table 4, our method outperforms DIFFA’s naive decoding on all five tasks in VoiceBench. For example, on OpenBookQA, AHD achieves a +2.0 performance gain while reducing decoding steps by 78%. Although there is a slight performance decrease on Wildvoice, it is negligible compared with the improvement in inference efficiency.

Task	Metric	Vanilla	PC-sampler	Fast-dLLM	KLASS	Saber	Ours
HumanEval	Score \uparrow	42.68	31.88 (-10.80)	43.29 (+0.61)	43.29 (+0.61)	42.07 (-0.61)	43.90 (+1.22)
	Step \downarrow	256	256	75.88 (-70%)	131.01 (-49%)	82.07 (-68%)	75.85 (-70%)
MBPP	Score \uparrow	27.20	25.60 (-1.60)	28.00 (+0.80)	27.20 (0.00)	25.40 (-1.80)	28.60 (+1.40)
	Step \downarrow	256	256	73.01 (-71%)	119.34 (-53%)	77.70 (-70%)	72.88 (-72%)
BBH	Score \uparrow	50.35	47.03 (-3.32)	49.85 (-0.50)	50.28 (-0.07)	50.02 (-0.33)	51.90 (+1.55)
	Step \downarrow	256	256	57.89 (-77%)	101.11 (-61%)	88.90 (-65%)	56.00 (-78%)
MMLU-Pro	Score \uparrow	37.15	35.01 (-2.14)	37.21 (+0.06)	37.26 (+0.11)	36.78 (-0.37)	38.48 (+1.33)
	Step \downarrow	256	256	134.72 (-47%)	188.14 (-27%)	125.71 (-51%)	133.02 (-48%)
TruthfulQA	Score \uparrow	41.27	41.78 (+0.51)	41.49 (+0.22)	41.37 (+0.10)	40.88 (-0.39)	41.98 (+0.71)
	Step \downarrow	256	256	83.93 (-67%)	104.26 (-59%)	68.50 (-73%)	72.21 (-72%)
Math	Score \uparrow	34.32	34.22 (-0.10)	34.21 (-0.11)	34.30 (-0.02)	34.26 (-0.06)	34.34 (+0.02)
	Step \downarrow	256	256	93.36 (-64%)	154.26 (-40%)	95.56 (-63%)	92.72 (-64%)
Asdiv	Score \uparrow	77.18	74.97 (-2.21)	76.36 (-0.82)	76.79 (-0.39)	75.84 (-1.34)	78.79 (+1.61)
	Step \downarrow	256	256	63.83 (-75%)	114.52 (-55%)	73.19 (-71%)	60.21 (-76%)

Table 2: Evaluation results of LLaDA-1.5 on seven benchmarks.

Task	Method	Score \uparrow	Speed-up \uparrow
MathVista-mini	MMaDA	32.90	1.00 \times
	Ours	36.00 (+3.10)	2.37 \times
MATH-Vision	MMaDA	9.44	1.00 \times
	Ours	10.00 (+0.56)	2.16 \times
ScienceQA	MMaDA	48.88	1.00 \times
	Ours	49.93 (+1.15)	2.70 \times
GQA	MMaDA	50.52	1.00 \times
	Ours	50.58 (+0.06)	16.40 \times
MME	MMaDA	1074.97	1.00 \times
	Ours	1075.72 (+0.75)	15.98 \times

Table 3: Comparison of performance and inference speedup across different multi-modal benchmarks.

Scalability. Previous studies have primarily focused on short-sequence generation, typically restricting output length to 256 or 512 tokens and largely overlooking long-sequence scenarios. To evaluate the robustness of our method under extended lengths, we conduct experiments with outputs up to 1024 tokens. As presented in Table 5, our approach consistently demonstrates superior performance even with increased generation lengths.

Intuitively, reducing decoding steps is typically perceived as inevitably leading to performance degradation. However, AHD primarily eliminates redundant iterations that have shown minimal variation or even introduced noise, while maintaining sufficient iterative capacity for tokens still undergoing meaningful evolutionary updates. Therefore, our method significantly reduces decoding steps while further improving the inference coherence and generation quality.

Computational Complexity. When computing the anchored consistency score, there is some ad-

Task	Method	Score \uparrow	Step \downarrow
AlpacaEval	DIFFA	3.75	128
	Ours	3.78 (+0.03)	80.19 (-37%)
OpenBookQA	DIFFA	36.50	128
	Ours	38.50 (+2.00)	28.67 (-78%)
CommonEval	DIFFA	2.90	128
	Ours	2.94 (+0.04)	72.45 (-44%)
BBH	DIFFA	53.00	128
	Ours	54.30 (+1.30)	56.49 (-56%)
Wildvoice	DIFFA	2.80	128
	Ours	2.76 (-0.04)	80.52 (-37%)

Table 4: Evaluation results of DIFFA on VoiceBench.

ditional GPU memory usage and inference time. However, as shown in Figure 4 (a), this overhead is negligible compared with the total inference cost of dLLM, since the logits for all tokens have already been computed (see Appendix A).

4.3 Ablation Study

In this section, we conduct comprehensive ablation experiments using LLaDA-8B-Instruct. More detailed experimental settings and results are provided in Appendix B and C.

Effect of Different Generation Length. To investigate the effect of generation length, we fix the block length at 32 and evaluate model performance with generation lengths of 128, 256, and 512. As summarized in Table 6, AHD consistently outperforms vanilla decoding in terms of both performance and inference efficiency across all generation length configurations.

Effect of Different Block Length. To assess the influence of block length, the generation length is

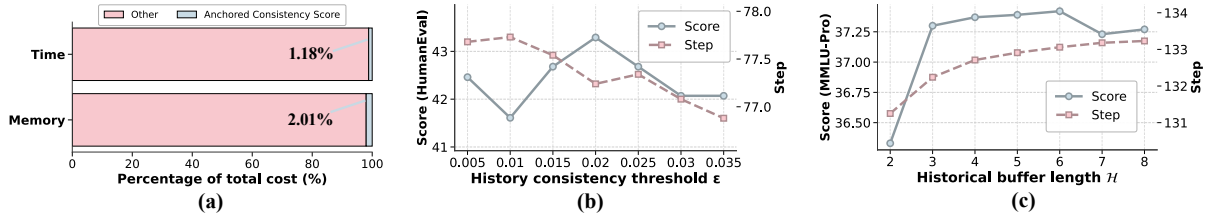


Figure 4: (a) Computational complexity of related operations; (b) Ablation study on ϵ ; (c) Ablation study on \mathcal{H} .

Method	BBH		HumanEval	
	Score \uparrow	Step \downarrow	Score \uparrow	Step \downarrow
LLaDA	53.28	1024	43.90	1024
PC-sampler	54.03	1024	42.07	1024
Fast-dLLM	53.68	65.54	42.68	184.79
KLASS	53.42	122.66	43.90	309.97
Saber	52.80	213.02	43.90	210.43
Ours	59.82 (+6.54)	58.25 (-94%)	48.78 (+4.88)	146.48 (-86%)

Table 5: Long-sequence results of LLaDA-8B-Instruct on two benchmarks (generation length 1024).

Task	Method	128		256		512	
		Score \uparrow	Step \downarrow	Score \uparrow	Step \downarrow	Score \uparrow	Step \downarrow
MBPP	LLaDA	39.00	128	29.20	256	16.20	512
	Ours	40.40	47.65	31.20	66.03	16.40	116.75
MMLU-Pro	LLaDA	37.42	128	35.57	256	37.46	512
	Ours	38.12	65.51	37.42	133.06	39.00	197.20
BBH	LLaDA	45.71	128	53.11	256	53.14	512
	Ours	50.07	33.11	55.69	52.62	59.05	60.68

Table 6: Performance and inference speed on three benchmarks at different generation lengths.

fixed at 256, and AHD is applied with four block configurations: 16, 32, 64, and 128. As reported in Table 7, AHD maintains superior task performance and inference efficiency over vanilla decoding for all block lengths, indicating strong robustness. Notably, this robustness extends to multimodal configurations, as evidenced in Table 8.

Effect of Historical Consistency Threshold ϵ . We conducted ablation experiments to analyze the impact of the ϵ on performance and decoding steps. As illustrated in Figure 4 (b), small values of ϵ prevent the early release of cross-block stable tokens, resulting in increased fluctuations and degraded performance (Wang et al., 2025b). Conversely, excessively large values of ϵ result in the premature release of numerous unstable tokens across blocks, similarly impairing performance. In addition, the decoding steps decrease monotonically with increasing ϵ . Notably, when $\epsilon = 0.02$, we achieve a balance between inference efficiency and performance, representing an optimal trade-off.

Effect of Historical Buffer Length \mathcal{H} . Our proposed method can achieve a trade-off between performance and inference efficiency by adjusting the historical buffer length. As shown in Figure 4 (c), a significantly small \mathcal{H} , such as 2, leads to a substan-

Block length	Method	MBPP		MMLU-Pro		BBH	
		Score \uparrow	Step \downarrow	Score \uparrow	Step \downarrow	Score \uparrow	Step \downarrow
16	LLaDA	28.60	256	33.74	256	52.51	256
	Ours	30.80	72.73	37.36	137.02	56.12	59.88
32	LLaDA	29.20	256	35.57	256	53.11	256
	Ours	31.20	66.03	37.42	133.06	55.69	52.62
64	LLaDA	27.40	256	36.67	256	54.32	256
	Ours	28.80	63.50	38.02	131.54	55.57	48.93
128	LLaDA	26.40	256	36.67	256	53.46	256
	Ours	26.40	59.47	37.84	132.14	53.49	48.08

Table 7: Performance and inference speed on three benchmarks at different block lengths.

Method	16		32		64		128	
	Score \uparrow	Speed Up \uparrow	Score \uparrow	Speed Up \uparrow	Score \uparrow	Speed Up \uparrow	Score \uparrow	Speed Up \uparrow
MMaDA	31.3	1.00 \times	32.1	1.00 \times	32.9	1.00 \times	32.5	1.00 \times
Ours	33.9	2.01\times	34.0	2.08\times	36.0	2.37\times	32.9	2.04\times

Table 8: Performance and inference speed on Mathvista-mini benchmark at different block lengths.

Method	temperature=0		temperature=0.1		temperature=0.4		temperature=0.7		temperature=1	
	Score \uparrow	Step \downarrow	Score \uparrow	Step \downarrow	Score \uparrow	Step \downarrow	Score \uparrow	Step \downarrow	Score \uparrow	Step \downarrow
LLaDA	35.57	256	35.86	256	35.80	256	35.73	256	35.20	256
Ours	37.42	133.06	37.99	132.77	37.46	132.08	37.22	131.06	36.93	131.62

Table 9: Performance and decoding steps on MMLU-Pro under different temperature coefficients.

tial performance degradation. This decline occurs because a short monitoring history fails to capture the true *absolute stability trend*, as noted in **core insights 1 and 2**. Conversely, we observe optimal performance when $\mathcal{H} = 6$, indicating an effective compromise between capturing historical trends and maintaining computational efficiency.

Robustness to Temperature Coefficients. In the main experiments, we use greedy decoding to ensure reproducibility. Additionally, we investigate the impact of temperature on AHD. Table 9 shows AHD exhibits strong robustness across varying temperature settings, while consistently delivering notable gains in accuracy and decoding efficiency.

5 Conclusion

In this work, we propose Anchor-based History-stable Decoding (AHD). Its core idea is to use dynamic anchors to monitor the historical trajectory in real time, capture the absolute stability trend at an early stage, and thereby unlock stable tokens across blocks in advance. AHD substantially re-

duces the decoding steps while consistently improving performance across multiple benchmarks in the language, multimodal, and audio domains. This offers a novel perspective for the development of future advanced decoding strategies.

Limitations

In this section, we discuss several limitations of our work. (1) Although AHD is a training-free, plug-and-play method, its parameters vary slightly across tasks, which increases the adaptation cost for new scenarios. (2) Our experiments mainly focus on models of around 8B parameters, and have not yet validated effectiveness on larger models such as 72B or 256B. With more computational resources, we plan to explore longer sequences and larger-scale models. (3) Although the additional computation introduced by AHD is almost negligible, we further optimize the related operations, such as vectorization, to further improve inference efficiency.

Acknowledgments

This work was supported by the Anhui Provincial Natural Science Foundation under Grant 2108085UD12, and the National Natural Science Foundation of China under Grant 62572034. We acknowledge the support of GPU cluster built by MCC Lab of Information Science and Technology Institution, USTC. The AI-driven experiments, simulations and model training were performed on the robotic AI-Scientist platform of Chinese Academy of Sciences.

References

- Jacob Austin, Daniel D Johnson, Jonathan Ho, Daniel Tarlow, and Rianne Van Den Berg. 2021a. [Structured denoising diffusion models in discrete state-spaces](#). *Advances in Neural Information Processing Systems*, 34:17981–17993.
- Jacob Austin, Augustus Odena, Maxwell Nye, Maarten Bosma, Henryk Michalewski, David Dohan, Ellen Jiang, Carrie Cai, Michael Terry, Quoc Le, and Charles Sutton. 2021b. [Program synthesis with large language models](#). *arXiv preprint arXiv:2108.07732*.
- Heli Ben-Hamu, Itai Gat, Daniel Severo, Niklas Nolte, and Brian Karrer. 2025. [Accelerated sampling from masked diffusion models via entropy bounded unmasking](#). *arXiv preprint arXiv:2505.24857*.
- Lukas Berglund, Meg Tong, Max Kaufmann, Mikita Balesni, Asa Cooper Stickland, Tomasz Korbak, and Owain Evans. 2024. [The reversal curse: LLMs trained on "a is b" fail to learn "b is a"](#). *arXiv preprint arXiv:2309.12288*.
- Tiwei Bie, Maosong Cao, Kun Chen, Lun Du, Mingliang Gong, Zhuochen Gong, Yanmei Gu, Jiaqi Hu, Zenan Huang, Zhenzhong Lan, Chengxi Li, Chongxuan Li, Jianguo Li, Zehuan Li, Huabin Liu, Lin Liu, Guoshan Lu, Xiaocheng Lu, Yuxin Ma, and 12 others. 2025. [Llada2.0: Scaling up diffusion language models to 100b](#). *arXiv preprint arXiv:2512.15745*.
- Andrew Campbell, Joe Benton, Valentin De Bortoli, Thomas Rainforth, George Deligiannidis, and Arnaud Doucet. 2022. [A continuous time framework for discrete denoising models](#). *Advances in Neural Information Processing Systems*, 35:28266–28279.
- Jiaao Chen, Aston Zhang, Mu Li, Alex Smola, and Diyi Yang. 2023. [A cheaper and better diffusion language model with soft-masked noise](#). In *Proceedings of the Conference on Empirical Methods in Natural Language Processing*, pages 4765–4775.
- Kecheng Chen, Ziru Liu, Xijia Tao, Hui Liu, Xinyu Fu, Suiyun Zhang, Dandan Tu, Lingpeng Kong, Rui Liu, and Haoliang Li. 2025a. [Beyond confidence: Adaptive and coherent decoding for diffusion language models](#). *arXiv preprint arXiv:2512.02044*.
- Mark Chen, Jerry Tworek, Heewoo Jun, Qiming Yuan, Henrique Ponde de Oliveira Pinto, Jared Kaplan, Harri Edwards, Yuri Burda, Nicholas Joseph, Greg Brockman, Alex Ray, Raul Puri, Gretchen Krueger, Michael Petrov, Heidy Khlaaf, Girish Sastry, Pamela Mishkin, Brooke Chan, Scott Gray, and 39 others. 2021. [Evaluating large language models trained on code](#). *arXiv preprint arXiv:2107.03374*.
- Xinhua Chen, Sitao Huang, Cong Guo, Chiyue Wei, Yintao He, Jianyi Zhang, Hai "Helen" Li, and Yiran Chen. 2025b. [Dpad: Efficient diffusion language models with suffix dropout](#). *arXiv preprint arXiv:2508.14148*.
- Yiming Chen, Xianghu Yue, Chen Zhang, Xiaoxue Gao, Robby T. Tan, and Haizhou Li. 2024. [Voicebench: Benchmarking llm-based voice assistants](#). *arXiv preprint arXiv:2410.17196*.
- Karl Cobbe, Vineet Kosaraju, Mohammad Bavarian, Mark Chen, Heewoo Jun, Lukasz Kaiser, Matthias Plappert, Jerry Tworek, Jacob Hilton, Reiichiro Nakano, Christopher Hesse, and John Schulman. 2021. [Training verifiers to solve math word problems](#). *arXiv preprint arXiv:2110.14168*.
- Yihong Dong, Zhaoyu Ma, Xue Jiang, Zhiyuan Fan, Jiaru Qian, Yongmin Li, Jianha Xiao, Zhi Jin, Rongyu Cao, Binhua Li, Fei Huang, Yongbin Li, and Ge Li. 2025. [Saber: An efficient sampling with adaptive acceleration and backtracking enhanced remasking for diffusion language model](#). *arXiv preprint arXiv:2510.18165*.

- Chaoyou Fu, Peixian Chen, Yunhang Shen, Yulei Qin, Mengdan Zhang, Xu Lin, Jinrui Yang, Xiawu Zheng, Ke Li, Xing Sun, Yunsheng Wu, Rongrong Ji, Caifeng Shan, and Ran He. 2025. **Mme: A comprehensive evaluation benchmark for multimodal large language models.** *arXiv preprint arXiv:2306.13394*.
- Shansan Gong, Shivam Agarwal, Yizhe Zhang, Jiacheng Ye, Lin Zheng, Mukai Li, Chenxin An, Peilin Zhao, Wei Bi, Jiawei Han, Hao Peng, and Lingpeng Kong. 2025. **Scaling diffusion language models via adaptation from autoregressive models.** *arXiv preprint arXiv:2410.17891*.
- Dan Hendrycks, Collin Burns, Saurav Kadavath, Akul Arora, Steven Basart, Eric Tang, Dawn Song, and Jacob Steinhardt. 2021. **Measuring mathematical problem solving with the math dataset.** *arXiv preprint arXiv:2103.03874*.
- Jonathan Ho, Ajay Jain, and Pieter Abbeel. 2020. **Denoising diffusion probabilistic models.** *Advances in Neural Information Processing Systems*, 33:6840–6851.
- Feng Hong, Geng Yu, Yushi Ye, Haicheng Huang, Huangjie Zheng, Ya Zhang, Yanfeng Wang, and Jiangchao Yao. 2025. **Wide-in, narrow-out: Revokable decoding for efficient and effective dllms.** *arXiv preprint arXiv:2507.18578*.
- Jianuo Huang, Yaojie Zhang, Yicun Yang, Benhao Huang, Biqing Qi, Dongrui Liu, and Linfeng Zhang. 2025a. **Mask tokens as prophet: Fine-grained cache eviction for efficient dllm inference.** *arXiv preprint arXiv:2510.09309*.
- Pengcheng Huang, Shuhao Liu, Zhenghao Liu, Yukun Yan, Shuo Wang, Zulong Chen, and Tong Xiao. 2025b. **Pc-sampler: Position-aware calibration of decoding bias in masked diffusion models.** *arXiv preprint arXiv:2508.13021*.
- Drew A Hudson and Christopher D Manning. 2019. **Gqa: A new dataset for real-world visual reasoning and compositional question answering.** In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pages 6700–6709.
- Seo Hyun Kim, Sunwoo Hong, Hojung Jung, Youngrok Park, and Se-Young Yun. 2026. **Klass: Kl-guided fast inference in masked diffusion models.** *arXiv preprint arXiv:2511.05664*.
- Fanheng Kong, Jingyuan Zhang, Yahui Liu, Zirui Wu, Yu Tian, Victoria W., and Guorui Zhou. 2025. **Accelerating diffusion llm inference via local determinism propagation.** *arXiv preprint arXiv:2510.07081*.
- Pengxiang Li, Yefan Zhou, Dilxat Muhtar, Lu Yin, Shilin Yan, Li Shen, Soroush Vosoughi, and Shiwei Liu. 2026. **Diffusion language models know the answer before decoding.** *arXiv preprint arXiv:2508.19982*.
- Stephanie Lin, Jacob Hilton, and Owain Evans. 2022. **Truthfulqa: Measuring how models mimic human falsehoods.** In *Proceedings of the 60th Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, pages 3214–3252.
- Zhiyuan Liu, Yicun Yang, Yaojie Zhang, Junjie Chen, Chang Zou, Qingyuan Wei, Shaobo Wang, and Linfeng Zhang. 2025. **dllm-cache: Accelerating diffusion large language models with adaptive caching.** *arXiv preprint arXiv:2506.06295*.
- Justin Lovelace, Varsha Kishore, Chao Wan, Eliot Shekhtman, and Kilian Q Weinberger. 2023. **Latent diffusion for language generation.** *Advances in Neural Information Processing Systems*, 36:56998–57025.
- Guanxi Lu, Hao Mark Chen, Yuto Karashima, Zhi-can Wang, Daichi Fujiki, and Hongxiang Fan. 2026. **Adablock-dllm: Semantic-aware diffusion llm inference via adaptive block size.** *arXiv preprint arXiv:2509.26432*.
- Pan Lu, Hritik Bansal, Tony Xia, Jiacheng Liu, Chunyuan Li, Hannaneh Hajishirzi, Hao Cheng, Kai-Wei Chang, Michel Galley, and Jianfeng Gao. 2024. **Mathvista: Evaluating mathematical reasoning of foundation models in visual contexts.** *arXiv preprint arXiv:2310.02255*.
- Pan Lu, Swaroop Mishra, Tony Xia, Liang Qiu, Kai-Wei Chang, Song-Chun Zhu, Oyvind Tafjord, Peter Clark, and Ashwin Kalyan. 2022. **Learn to explain: Multimodal reasoning via thought chains for science question answering.** *Advances in Neural Information Processing Systems*, 35:2507–2521.
- Xinyin Ma, Runpeng Yu, Gongfan Fang, and Xinchao Wang. 2025a. **dkv-cache: The cache for diffusion language models.** *arXiv preprint arXiv:2505.15781*.
- Yuxin Ma, Lun Du, Lanning Wei, Kun Chen, Qian Xu, Kangyu Wang, Guofeng Feng, Guoshan Lu, Lin Liu, Xiaojing Qi, Xinyuan Zhang, Zhen Tao, Haibo Feng, Ziyun Jiang, Ying Xu, Zenan Huang, Yihong Zhuang, Haokai Xu, Jiaqi Hu, and 4 others. 2025b. **dinfer: An efficient inference framework for diffusion language models.** *arXiv preprint arXiv:2510.08666*.
- Andrew Melnik, Michal Ljubljanc, Cong Lu, Qi Yan, Weiming Ren, and Helge Ritter. 2024. **Video diffusion models: A survey.** *arXiv preprint arXiv:2405.03150*.
- Shen-Yun Miao, Chao-Chun Liang, and Keh-Yih Su. 2020. **A diverse corpus for evaluating and developing english math word problem solvers.** In *Proceedings of the 58th Annual Meeting of the Association for Computational Linguistics*, pages 975–984.
- Shen Nie, Fengqi Zhu, Zebin You, Xiaolu Zhang, Jingyang Ou, Jun Hu, Jun Zhou, Yankai Lin, Ji-Rong Wen, and Chongxuan Li. 2025. **Large language diffusion models.** *arXiv preprint arXiv:2502.09992*.

- Jingyang Ou, Shen Nie, Kaiwen Xue, Fengqi Zhu, Jiacheng Sun, Zhenguo Li, and Chongxuan Li. 2026. [Your absorbing discrete diffusion secretly models the conditional distributions of clean data](#). *arXiv preprint arXiv:2406.03736*.
- Robin Rombach, Andreas Blattmann, Dominik Lorenz, Patrick Esser, and Björn Ommer. 2022. [High-resolution image synthesis with latent diffusion models](#). In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pages 10684–10695.
- Subham Sekhar Sahoo, Marianne Arriola, Yair Schiff, Aaron Gokaslan, Edgar Marroquin, Justin T Chiu, Alexander Rush, and Volodymyr Kuleshov. 2024. [Simple and effective masked diffusion language models](#). *Advances in Neural Information Processing Systems*, 37:130136–130184.
- Jiaxin Shi, Kehang Han, Zhe Wang, Arnaud Doucet, and Michalis K. Titsias. 2024. [Simplified and generalized masked diffusion for discrete data](#). *Advances in Neural Information Processing Systems*, 37:103131–103167.
- Yuerong Song, Xiaoran Liu, Ruixiao Li, Zhigeng Liu, Zengfeng Huang, Qipeng Guo, Ziwei He, and Xipeng Qiu. 2025. [Sparse-dllm: Accelerating diffusion llms with dynamic cache eviction](#). *arXiv preprint arXiv:2508.02558*.
- Mirac Suzgun, Nathan Scales, Nathanael Schärli, Sebastian Gehrmann, Yi Tay, Hyung Won Chung, Aakanksha Chowdhery, Quoc V. Le, Ed H. Chi, Denny Zhou, and Jason Wei. 2023. [Challenging BIG-bench tasks and whether chain-of-thought can solve them](#). In *Findings of the Association for Computational Linguistics: ACL 2023*, pages 13003–13051.
- Kangyu Wang, Zhiyun Jiang, Haibo Feng, Weijia Zhao, Lin Liu, Jianguo Li, Zhenzhong Lan, and Weiyao Lin. 2025a. [Creditdecoding: Accelerating parallel decoding in diffusion large language models with trace credits](#). *arXiv preprint arXiv:2510.06133*.
- Ke Wang, Junting Pan, Weikang Shi, Zimu Lu, Houxing Ren, Aojun Zhou, Mingjie Zhan, and Hongsheng Li. 2024a. [Measuring multimodal mathematical reasoning with math-vision dataset](#). *Advances in Neural Information Processing Systems*, 37:95095–95169.
- Wen Wang, Bozhen Fang, Chenchen Jing, Yongliang Shen, Yangyi Shen, Qiuyu Wang, Hao Ouyang, Hao Chen, and Chunhua Shen. 2025b. [Time is a feature: Exploiting temporal dynamics in diffusion language models](#). *arXiv preprint arXiv:2508.09138*.
- Xu Wang, Chenkai Xu, Yijie Jin, Jiachun Jin, Hao Zhang, and Zhijie Deng. 2025c. [Diffusion llms can do faster-than-ar inference via discrete diffusion forcing](#). *arXiv preprint arXiv:2508.09192*.
- Yubo Wang, Xueguang Ma, Ge Zhang, Yuansheng Ni, Abhranil Chandra, Shiguang Guo, Weiming Ren, Aaran Arulraj, Xuan He, Ziyang Jiang, Tianle Li, Max Ku, Kai Wang, Alex Zhuang, Rongqi Fan, Xiang Yue, and Wenhua Chen. 2024b. [Mmlu-pro: A more robust and challenging multi-task language understanding benchmark](#). *Advances in Neural Information Processing Systems*, 37:95266–95290.
- Qingyan Wei, Yaojie Zhang, Zhiyuan Liu, Puyu Zeng, Yuxuan Wang, Biqing Qi, Dongrui Liu, and Linfeng Zhang. 2026. [Accelerating diffusion large language models with slowfast sampling: The three golden principles](#). *arXiv preprint arXiv:2506.10848*.
- Chengyue Wu, Hao Zhang, Shuchen Xue, Zhijian Liu, Shizhe Diao, Ligeng Zhu, Ping Luo, Song Han, and Enze Xie. 2025. [Fast-dllm: Training-free acceleration of diffusion llm by enabling kv cache and parallel decoding](#). *arXiv preprint arXiv:2505.22618*.
- Yi Xin, Qi Qin, Siqi Luo, Kaiwen Zhu, Juncheng Yan, Yan Tai, Jiayi Lei, Yuwen Cao, Keqi Wang, Yibin Wang, Jinbin Bai, Qian Yu, Dengyang Jiang, Yuan-dong Pu, Haoxing Chen, Le Zhuo, Junjun He, Gen Luo, Tianbin Li, and 13 others. 2025. [Lumina-dimoo: An omni diffusion large language model for multimodal generation and understanding](#). *arXiv preprint arXiv:2510.06308*.
- Ling Yang, Ye Tian, Bowen Li, Xinchen Zhang, Ke Shen, Yunhai Tong, and Mengdi Wang. 2025. [Mmada: Multimodal large diffusion language models](#). *arXiv preprint arXiv:2505.15809*.
- Ling Yang, Zhilong Zhang, Yang Song, Shenda Hong, Runsheng Xu, Yue Zhao, Wentao Zhang, Bin Cui, and Ming-Hsuan Yang. 2023. [Diffusion models: A comprehensive survey of methods and applications](#). *ACM Computing Surveys*, 56(4):1–39.
- Jiacheng Ye, Zhihui Xie, Lin Zheng, Jiahui Gao, Zirui Wu, Xin Jiang, Zhenguo Li, and Lingpeng Kong. 2025. [Dream 7b: Diffusion large language models](#). *arXiv preprint arXiv:2508.15487*.
- Runpeng Yu, Qi Li, and Xinchao Wang. 2025a. [Discrete diffusion in large language and multimodal models: A survey](#). *arXiv preprint arXiv:2506.13759*.
- Runpeng Yu, Xinyin Ma, and Xinchao Wang. 2025b. [Dimple: Discrete diffusion multimodal large language model with parallel decoding](#). *arXiv preprint arXiv:2505.16990*.
- Kaichen Zhang, Bo Li, Peiyuan Zhang, Fanyi Pu, Joshua Adrian Cahyono, Kairui Hu, Shuai Liu, Yuanhan Zhang, Jingkang Yang, Chunyuan Li, and Ziwei Liu. 2025. [LMMs-eval: Reality check on the evaluation of large multimodal models](#). In *Findings of the Association for Computational Linguistics: NAACL 2025*, pages 881–916.
- Tianyi Zhang, Zheng Wang, Jing Huang, Mohiuddin Muhammad Tasnim, and Wei Shi. 2023. [A survey of diffusion based image generation models: Issues and their solutions](#). *arXiv preprint arXiv:2308.13142*.

Jiaming Zhou, Hongjie Chen, Shiwan Zhao, Jian Kang, Jie Li, Enzhi Wang, Yujie Guo, Haoqin Sun, Hui Wang, Aobo Kong, Yong Qin, and Xuelong Li. 2025. [Diffa: Large language diffusion models can listen and understand](#). *arXiv preprint arXiv:2507.18452*.

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

A Related Work

Diffusion Large Language Model. Diffusion models have achieved widespread success in image and video generation (Ho et al., 2020; Rombach et al., 2022; Melnik et al., 2024; Zhang et al., 2023; Yang et al., 2023), inspiring growing interest in applying them to discrete text domains. Migrating diffusion models from continuous to discrete domains presents unique challenges, and researchers have explored several technical approaches (Campbell et al., 2022; Lovelace et al., 2023; Yu et al., 2025a). Among these, Masked Diffusion Models (MDMs) have become the most mainstream branch (Austin et al., 2021a; Sahoo et al., 2024; Chen et al., 2023; Gong et al., 2025). This paradigm models text generation as an iterative “mask–denoise” process. In the forward process, noise is introduced by gradually replacing tokens in the text with a special [MASK] absorption state. In the reverse process, the model learns to reconstruct the original text from a fully masked sequence (Ou et al., 2026; Sahoo et al., 2024; Shi et al., 2024; Nie et al., 2025). Inspired by this framework, researchers have successfully extended MDMs to diffusion large language models (dLLMs), with LLaDA (Nie et al., 2025) and Dream (Ye et al., 2025) attracting wide attention in the community. Their performance is competitive with autoregressive models with similar scale (Yu et al., 2025a).

Parallel Decoding. In dLLMs, logits for all token positions are computed during each decoding iteration. Fast-dLLM identifies the root cause of degraded generation quality in parallel decoding as the violation of token dependencies under the conditional independence assumption. To address this, Fast-dLLM and Dimple propose confidence-aware parallel decoding (Wu et al., 2025; Yu et al., 2025b), which dynamically adjusts the number of tokens to unmask in each iteration based on a confidence threshold. This allows generating multiple tokens simultaneously while mitigating dependency violations and preserving generation quality, thereby achieving a trade-off between quality and inference efficiency.

Advanced Decoding Strategies for dLLM. Due to the iterative denoising nature of dLLMs, their inference latency is relatively high (Nie et al., 2025). Furthermore, as the attention in dLLMs is bidirectional, traditional KV caching is not directly applicable (Wu et al., 2025). This has led to the

development of various acceleration strategies, including specialized KV cache mechanisms (Chen et al., 2025b; Liu et al., 2025; Ma et al., 2025a; Wu et al., 2025; Song et al., 2025; Huang et al., 2025a) and advanced decoding methods (Wu et al., 2025; Dong et al., 2025; Wang et al., 2025a; Wei et al., 2026; Kong et al., 2025; Kim et al., 2026; Li et al., 2026; Lu et al., 2026). Parallel decoding strategies typically sample multiple tokens at each step, thereby reducing the total number of iterative steps without retraining and achieving faster generation. Early work such as Fast-dLLM (Wu et al., 2025) proposed a parallel decoding scheme based on a fixed confidence threshold. Saber (Dong et al., 2025) introduced adaptive acceleration by dynamically adjusting the parallel size and employed a backtracking re-masking mechanism to correct prediction errors, improving both speed and quality. WINO (Hong et al., 2025) implements revocable decoding in dLLMs, enabling the refinement of suspicious tokens. However, these decoding strategies are all based on semi-autoregressive decoding, where advanced decoding typically operates only within a single block per step, without leveraging already stable tokens in future blocks.

B More Implementation Details

Main Experiment. In the language domain, we employ the lm-eval library¹ to ensure evaluation consistency, stability, and reproducibility. Table 10 reports the in-context learning (ICL) configuration used for each benchmark. In the vision-language domain, we adopt the lmms-eval (Zhang et al., 2025) library to unify the evaluation of performance across multimodal understanding and reasoning benchmarks. In the audio-language domain, we follow the experimental setup of DIFFA. The generation and block lengths are set to 128 and 16, respectively. We evaluate our method on five tasks using the VoiceBench benchmark. For AlpacaEval, CommonEval, and WildVoice, we employ GPT-4o-mini as the evaluator. We set *max_tokens* to 1024, *temperature* to 0.5, and *top_p* to 0.95.

Additionally, we present detailed inference efficiency metrics in our experiments. Among these metrics, TPF and Step focus exclusively on the decoding process of the dLLM itself. They are unaffected by hardware quantity, hardware type, GPU communication protocols, or inference frameworks,

¹<https://github.com/EleutherAI/lm-evaluation-harness>

and thus directly reflect intrinsic improvements to dLLMs. All experiments are conducted on eight NVIDIA H20 GPUs.

We provide the detailed hyperparameter settings for each benchmark in Table 11. In general, we set the historical-consistency threshold $\varepsilon=0.01$ and the historical buffer length $\mathcal{H}=6$. Although the optimal hyperparameters vary slightly across benchmarks, their performance remains relatively consistent.

In the main experiments on LLaDA-8B-Instruct and LLaDA-1.5, we compared not only with naive decoding but also with several recent advanced decoding strategies, including PC-sample (Huang et al., 2025b), Fast-dLLM (Wu et al., 2025), KLASS (Kim et al., 2026), and Saber (Dong et al., 2025). We kept their parameters consistent with the original papers to ensure fair comparison.

Task	ICL
HumanEval	0-shot
MBPP	3-shot
BBH	3-shot
MMLU-Pro	0-shot
TruthfulQA	0-shot
Math	3-shot
Asdiv	0-shot

Table 10: In-context learning settings for different tasks.

Ablation Experiment. In all ablation experiments, we use LLaDA-8B-Instruct as the base model. For the ablations on generation length and block length, the hyperparameters of historical consistency threshold ε and historical buffer length \mathcal{H} are set to 0.01 and 6, respectively. For the ablations on ε and \mathcal{H} , we fix the generation length at 256 and the block length at 32.

C More Experimental Results

C.1 Scalability.

Previous studies have mostly neglected the case of long-sequence generation, typically limiting generation length to 256 or 512. To verify whether our method remains effective for long sequences, we conducted experiments with extended generation lengths. Specifically, the block length was set to 64 for a generation length of 1024 and 128 for a generation length of 2048, while all other parameters remained consistent with those of the main experiments. As shown in Table 12, on the HumanEval benchmark, our method continues to be

Task	LLaDA-8B-Instruct		LLaDA-1.5	
	ε	\mathcal{H}	ε	\mathcal{H}
HumanEval	0.02	6	0.01	6
MBPP	0.01	6	0.01	4
BBH	0.05	6	0.01	6
MMLU-Pro	0.01	6	0.01	6
TruthfulQA	0.01	6	0.01	6
Math	0.01	6	0.001	7
Asdiv	0.01	6	0.01	6

Table 11: Hyperparameters ε and \mathcal{H} for different tasks.

effective for generation lengths of 1024 and 2048, maintaining a stable and consistent improvement trend.

C.2 Further Results on Additional Benchmarks.

In this section, we present the results of LLaDA-8B-Instruct on additional benchmarks. As shown in Table 13, AHD maintains superior performance and inference efficiency on GSM8K-COT (Cobbe et al., 2021) compared with naive decoding and other advanced decoding strategies, demonstrating a consistent improvement trend with the main experiments described in the text.

C.3 Further Results from the Ablation Study.

In this section, we provide detailed ablation studies on generation length and block length. As shown in Tables 14 and 15, we include comparisons with other advanced decoding strategies on the MMLU-Pro and BBH general task benchmarks. The comprehensive results demonstrate that AHD is robust to both generation length and block length. Notably, as the generation length increases, the reduction in decoding steps for AHD compared to vanilla decoding becomes increasingly pronounced. This phenomenon can be attributed to the delayed decoding of stable tokens caused by block boundaries, which becomes more significant with longer generation lengths.

D Comparison with Other Works

In this section, we focus on the differences between AHD and other decoding methods. Although KLASS (Kim et al., 2026) also introduces the concept of KL divergence, it is important to emphasize that AHD is fundamentally different from the step-wise KL adopted in KLASS, which only compares

Method	1024			2048		
	Score \uparrow	TPF \uparrow	Step \downarrow	Score \uparrow	TPF \uparrow	Step \downarrow
LLaDA	43.90	1.00	1024	44.51	1.00	2048
PC-sampler	42.07	1.00	1024	41.46	1.00	2048
Fast-dLLM	42.68	5.54	184.79	44.51	10.50	195.13
KLASS	43.90	3.30	309.97	44.51	6.39	320.63
Saber	43.90	4.87	210.43	48.17	9.34	219.25
Ours	48.78	6.99	146.48	49.39	11.92	171.84

Table 12: Long-sequence results of LLaDA-8B-Instruct on HumanEval benchmark.

Method	Score \uparrow	TPF \uparrow	Step \downarrow
LLaDA	79.53	1.00	256
PC-sampler	79.98 (+0.45)	1.00	256
Fast_dLLM	80.29 (+0.76)	3.41 ($\times 3.41$)	75.10 (-71%)
KLASS	80.06 (+0.53)	1.99 ($\times 1.99$)	128.64 (-50%)
Saber	79.61 (+0.08)	3.35 ($\times 3.35$)	76.49 (-70%)
Ours	80.51 (+0.98)	3.41 ($\times 3.41$)	75.06 (-71%)

Table 13: Results of LLaDA-8B-Instruct on the GSM8K-CoT benchmark.

adjacent distributions (i.e., p^t and p^{t+1}). In contrast, AHD fixes the current distribution P_j^t as an anchor and retrospectively compares it against the predictions along the entire historical trajectory.

On the one hand, such an anchor-based tracking mechanism provides a more direct characterization of whether the current prediction remains consistently stable over a period of time, rather than merely reflecting local fluctuations between two neighboring steps, thereby avoiding misleading judgments caused by local fluctuations. On the other hand, by performing anchored real-time backtracking, AHD can more sensitively capture the emergence of the absolute stability trend and thus avoid unnecessary iterative decoding steps.

Moreover, KLASS is mainly applied to decoding within the current block, whereas AHD is designed for decoding in future blocks. Therefore, our method is substantially different from the KLASS strategy that computes KL divergence only between adjacent decoding steps.

Although Prophet (Li et al., 2026) also mentions the phenomenon of delayed decoding, the phenomenon it describes is fundamentally different from the one discussed in our paper. Our focus is on the delayed decoding of cross-block tokens caused by block boundaries under the semi-autoregressive decoding framework. In contrast, Prophet considers the situation where certain tokens can already

be confidently determined at an early stage of decoding, and then uses dynamic monitoring of confidence gaps to terminate the iterative decoding process early. Therefore, both the phenomenon we investigate and the motivation of our method are fundamentally different from those of Prophet.

E Theoretical Analysis

E.1 Assumption and Notation

Token-level Prediction Trajectory. Let \mathcal{V} be the vocabulary with $|\mathcal{V}|$ tokens, and $\Delta^{|\mathcal{V}|-1}$ denote the probability simplex. For a fixed position j (we sometimes omit j when clear from context), dLLM produces a token-level prediction distribution at reverse step $t = 0, \dots, T$:

$$P_j^t \in \Delta^{|\mathcal{V}|-1}. \quad (8)$$

Historical buffer and exponential weights. In AHD, we maintain a *historical buffer* of length H containing recent predictions for each position j :

$$\mathcal{H}_j^t = \{P_j^{t-H+1}, \dots, P_j^{t-1}, P_j^t\}, \quad (9)$$

where we only keep the most recent H steps and assume $t \geq H$ in practice.

For a given $\lambda > 0$, time weights are defined

Task	Method	128		256		512	
		Score \uparrow	Step \downarrow	Score \uparrow	Step \downarrow	Score \uparrow	Step \downarrow
MMLU-Pro	LLaDA	37.42	128	35.57	256	37.46	512
	PC-sampler	35.59 (-1.83)	128	33.90 (-1.67)	256	35.31 (-2.15)	512
	Fast_dllm	37.36 (-0.06)	67.72 (-47%)	35.70 (+0.13)	134.76 (-47%)	37.32 (-0.14)	199.31 (-61%)
	KLASS	37.43 (+0.01)	88.96 (-31%)	35.67 (+0.10)	187.17 (-27%)	37.43 (-0.03)	293.96 (-43%)
	Saber	36.48 (-0.94)	55.65 (-57%)	36.10 (+0.53)	128.83 (-50%)	37.16 (-0.30)	195.01 (-62%)
	Ours	38.12 (+0.70)	65.51 (-49%)	37.42 (+1.85)	133.06 (-48%)	39.00 (+1.54)	197.20 (-61%)
BBH	LLaDA	45.71	128	53.11	256	53.14	512
	PC-sampler	42.05 (-3.66)	128	48.55 (-4.56)	256	49.62 (-3.52)	512
	Fast_dllm	45.37 (-0.34)	33.13 (-74%)	53.17 (+0.06)	55.85 (-78%)	52.81 (-0.33)	68.66 (-87%)
	KLASS	45.69 (-0.02)	59.90 (-53%)	53.03 (-0.08)	98.04 (-62%)	53.17 (+0.03)	125.37 (-76%)
	Saber	45.62 (-0.09)	47.10 (-63%)	52.88 (-0.23)	87.31 (-66%)	51.84 (-1.30)	120.27 (-77%)
	Ours	50.07 (+4.36)	33.11 (-74%)	56.78 (+3.67)	51.48 (-80%)	59.05 (+5.91)	60.68 (-88%)

Table 14: Performance and steps on the MMLU-Pro and BBH benchmarks at various generation lengths.

using an exponential decay function as follows:

$$w_\tau = \frac{\exp(-\lambda\tau)}{\sum_{k=1}^{H-1} \exp(-\lambda k)}, \quad \tau = 1, \dots, H-1, \quad (10)$$

where $w_\tau > 0$ and $\sum_{\tau=1}^{H-1} w_\tau = 1$. Moreover, due to exponential decay, we have:

$$w_1 = \max_{\tau \in \{1, \dots, H-1\}} w_\tau. \quad (11)$$

This property will be used to relate the trajectory-level consistency score $D_j^t(\text{acs})$ to the one-step KL divergence.

Token Embeddings and Semantic Representations. Let $E(v) \in \mathbb{R}^d$ denote the embedding of token $v \in \mathcal{V}$. **Assumption 1 (Bounded embeddings).** There exists $R > 0$ such that for all $v \in \mathcal{V}$:

$$\|E(v)\|_2 \leq R. \quad (12)$$

In practice, the predictive distribution at each position is computed from high-dimensional hidden states in a Transformer network. For theoretical analysis, we abstract these hidden states by the following *semantic representation*:

$$\mathbf{e}_j^t = \sum_{v \in \mathcal{V}} P_j^t(v) E(v) \in \mathbb{R}^d. \quad (13)$$

This representation can be viewed as a proxy for the position-wise hidden state in a lower-dimensional embedding space. Our results rely only on the boundedness of such representations (Assumption 1), not on the exact form of the network.

One-step KL and Anchor-based Historical Consistency. In Section 3.3 of the main text, we define the one-step KL divergence for position j at step t as:

$$\begin{aligned} D_j^t(\text{one-step}) &= D_{\text{KL}}\left(P_j^t \parallel P_j^{t-1}\right) \\ &= \sum_{v \in \mathcal{V}} P_j^t(v) \log \frac{P_j^t(v)}{P_j^{t-1}(v)}. \end{aligned} \quad (14)$$

To extend beyond single-step behavior, AHD introduces an *anchor-based historical consistency* criterion. Specifically, at step t , the current prediction P_t^j for position j is designated as the *anchor*:

$$P_{j, \text{anchor}}^t = P_j^t. \quad (15)$$

For each backtracking offset $\tau \in \{1, \dots, H-1\}$, the anchored KL divergence is:

$$\delta_j^{t, \tau} = D_{\text{KL}}\left(P_{j, \text{anchor}}^t \parallel P_j^{t-\tau}\right), \quad (16)$$

and the corresponding *historical consistency sequence* is:

$$\{\delta_j^{t, \tau}, \dots, \delta_j^{t, H-1}\}. \quad (17)$$

Definition 1 (Anchor-based historical consistency score). For position j at step t , we define the *anchor-based historical consistency score* as an exponentially weighted sum:

$$\begin{aligned} D_j^t(\text{acs}) &= \sum_{\tau=1}^{H-1} w_\tau \delta_j^{t, \tau} \\ &= \sum_{\tau=1}^{H-1} w_\tau D_{\text{KL}}\left(P_{j, \text{anchor}}^t \parallel P_j^{t-\tau}\right). \end{aligned} \quad (18)$$

Block length	Method	MMLU-Pro		BBH	
		Score \uparrow	Step \downarrow	Score \uparrow	Step \downarrow
16	LLaDA	33.74	256	52.51	256
	PC-sampler	33.98 (+0.24)	256	48.44 (-4.07)	256
	Fast_dllm	33.65 (-0.09)	138.76 (-46%)	52.47 (-0.04)	63.37 (-75%)
	KLASS	33.73 (-0.01)	192.61 (-25%)	52.57 (+0.06)	112.90 (-56%)
	Saber	34.55 (+0.81)	125.00 (-51%)	51.64 (-0.87)	91.47 (-64%)
	Ours	37.36 (+3.62)	137.02 (-46%)	56.12 (+3.61)	59.88 (-77%)
32	LLaDA	35.57	256	53.11	256
	PC-sampler	33.90 (-1.67)	256	48.55 (-4.56)	256
	Fast_dllm	35.70 (+0.13)	134.76 (-47%)	53.17 (+0.06)	55.85 (-78%)
	KLASS	35.67 (+0.10)	187.17 (-27%)	53.03 (-0.08)	98.04 (-62%)
	Saber	36.10 (+0.53)	128.83 (-50%)	52.88 (-0.23)	87.31 (-66%)
	Ours	37.42 (+1.85)	133.06 (-48%)	56.78 (+3.67)	51.48 (-80%)
64	LLaDA	36.67	256	54.32	256
	PC-sampler	34.13 (-2.54)	256	48.49 (-5.83)	256
	Fast_dllm	36.79 (+0.12)	133.33 (-48%)	53.40 (-0.92)	51.05 (-80%)
	KLASS	36.82 (+0.15)	185.64 (-27%)	53.54 (-0.78)	89.47 (-65%)
	Saber	37.28 (+0.61)	124.10 (-52%)	54.32 (-0.0)	86.45 (-66%)
	Ours	38.02 (+1.35)	131.54 (-49%)	55.57 (+1.25)	48.93 (-81%)
128	LLaDA	36.67	256	53.46	256
	PC-sampler	34.06 (-2.61)	256	48.55 (-4.91)	256
	Fast_dllm	36.74 (+0.07)	134.67 (-47%)	52.48 (-0.98)	48.92 (-81%)
	KLASS	36.71 (+0.04)	187.96 (-27%)	52.66 (-0.80)	85.47 (-67%)
	Saber	37.14 (+0.47)	126.63 (-51%)	53.11 (-0.35)	89.07 (-65%)
	Ours	37.84 (+1.17)	132.14 (-48%)	53.49 (+0.03)	48.08 (-81%)

Table 15: Performance and inference speed on the MMLU-Pro and BBH benchmarks at different block lengths.

Intuitively, $D_j^t(acs)$ measures how consistent the current ‘‘anchor’’ distribution P_t^j is with the entire recent prediction trajectory stored in the historical buffer, rather than with a single previous step.

E.2 Anchor-based Historical Consistency Controls Distributional and Semantic Stability

We first demonstrate that when the anchor-based historical consistency score $D_j^t(acs)$ is small, the current prediction P_t^j and its semantic representation e_j^t are both close to the exponentially weighted historical mean. This justifies $D_j^t(acs)$ as a *trajectory-level* stability metric. For notational simplicity, we drop the position index j in this subsection.

Historical Mean Distribution and Embedding.

First of all, we define the historical exponential

average distribution as:

$$\bar{P}_t = \sum_{\tau=1}^{H-1} w_\tau P_{t-\tau}, \quad (19)$$

and the corresponding semantic average as:

$$\bar{e}_t = \sum_{\tau=1}^{H-1} w_\tau e_{t-\tau}. \quad (20)$$

Lemma 1 (Anchor-based historical consistency controls deviation from the historical mean distribution). For $t \geq H$, we have:

$$\|P_t - \bar{P}_t\|_1 \leq \sqrt{2D_t(acs)}, \quad (21)$$

where $D_t(acs)$ is defined as in Equation (18).

Proof. We use the standard Pinsker inequality, which states that for any distributions P, Q :

$$\|P - Q\|_1 \leq \sqrt{2D_{\text{KL}}(P\|Q)}. \quad (22)$$

Applying this with $P = P_t$ and $Q = P_{t-\tau}$, we obtain:

$$\|P_t - P_{t-\tau}\|_1 \leq \sqrt{2D_{\text{KL}}(P_t \| P_{t-\tau})}, \quad (23)$$

$$\tau = 1, \dots, H-1.$$

On the other hand, we have:

$$\begin{aligned} \|P_t - \bar{P}_t\|_1 &= \left\| P_t - \sum_{\tau=1}^{H-1} w_\tau P_{t-\tau} \right\|_1 \\ &= \left\| \sum_{\tau=1}^{H-1} w_\tau (P_t - P_{t-\tau}) \right\|_1 \\ &\leq \sum_{\tau=1}^{H-1} w_\tau \|P_t - P_{t-\tau}\|_1 \quad (24) \\ &\quad (\text{triangle inequality}) \\ &\leq \sum_{\tau=1}^{H-1} w_\tau \sqrt{2D_{\text{KL}}(P_t \| P_{t-\tau})} \\ &\quad (\text{by equation(23)}). \end{aligned}$$

Since $f(x) = \sqrt{x}$ is concave on \mathbb{R}_+ , using Jensen's inequality yields:

$$\begin{aligned} &\sum_{\tau=1}^{H-1} w_\tau \sqrt{2D_{\text{KL}}(P_t \| P_{t-\tau})} \\ &\leq \sqrt{2 \sum_{\tau=1}^{H-1} w_\tau D_{\text{KL}}(P_t \| P_{t-\tau})} \quad (25) \\ &= \sqrt{2D_t(acs)}. \end{aligned}$$

Combining Equations (24) and (25) gives Equation (21). \square

Lemma 2 (Anchor-based historical consistency controls deviation in embedding space). Under Assumption 1, for $t \geq H$, we have:

$$\|\mathbf{e}_t - \bar{\mathbf{e}}_t\|_2 \leq R\sqrt{2D_t(acs)}. \quad (26)$$

Proof. By the definitions (13) and (20),

$$\mathbf{e}_t = \sum_v P_t(v)E(v), \quad \bar{\mathbf{e}}_t = \sum_v \bar{P}_t(v)E(v). \quad (27)$$

Hence,

$$\mathbf{e}_t - \bar{\mathbf{e}}_t = \sum_v (P_t(v) - \bar{P}_t(v))E(v). \quad (28)$$

Taking the ℓ_2 norm and using the triangle inequality

and Assumption 1,

$$\begin{aligned} \|\mathbf{e}_t - \bar{\mathbf{e}}_t\|_2 &\leq \sum_v |P_t(v) - \bar{P}_t(v)| \cdot \|E(v)\|_2 \\ &\leq R \sum_v |P_t(v) - \bar{P}_t(v)| \\ &= R\|P_t - \bar{P}_t\|_1. \end{aligned} \quad (29)$$

By **Lemma 1**, $\|P_t - \bar{P}_t\|_1 \leq \sqrt{2D_t(acs)}$, thus (26) follows. \square

Combining Lemmas 1 and 2 show that small anchor-based historical consistency $D_t(acs)$ implies that the current prediction is close to an exponentially weighted average over its recent trajectory, both in distribution and embedding space. This is one aspect of the *trajectory-level* stability captured by AHD.

E.3 Anchor-base Historical Consistency is Sensitive to Long-range Oscillations

We demonstrate that the anchor-based score $D_t(acs)$ is sensitive to long-range oscillations, whereas the one-step KL $D_t(one-step)$ can remain small under such behavior.

Lemma 3. Suppose there exists some $\tau^* \in \{1, \dots, H-1\}$ such that:

$$D_{\text{KL}}(P_t \| P_{t-\tau^*}) \geq \Delta > 0. \quad (30)$$

Then,

$$D_t^{(acs)} \geq w_{\tau^*} \Delta. \quad (31)$$

Proof. By definition (18),

$$\begin{aligned} D_t(acs) &= \sum_{\tau=1}^{H-1} w_\tau D_{\text{KL}}(P_t \| P_{t-\tau}) \\ &\geq w_{\tau^*} D_{\text{KL}}(P_t \| P_{t-\tau^*}) \\ &\geq w_{\tau^*} \Delta. \end{aligned} \quad (32)$$

Therefore, as long as the current distribution P_t differs significantly from at least one historical point in the buffer, $D_t(acs)$ cannot be arbitrarily small. In contrast, the one-step KL $D_t(one-step) = D_{\text{KL}}(P_t \| P_{t-1})$ may remain near zero under long-period oscillations (e.g., with period greater than 1), failing to capture instability on a longer time scale. \square

F Case Study

In this section, we present several detailed cases. By unlocking stable tokens across blocks in advance, AHD significantly reduces the number of

decoding steps while maintaining high accuracy. In contrast, the naive decoding strategy of LLaDA performs redundant inference and even produces incorrect answers.

G LLM Usage

This section clarifies the role of large language models (LLMs) in this study. LLMs are used only for language refinement, including grammar and readability. They do not participate in any aspect of research design.

Question: Given the following problem, reason and give a final answer to the problem. Problem: If Tom plans to fix 158 watches at the rate of 12 watches per day, how many watches will he fix on the last day? Your response should end with "The final answer is [answer]" where [answer] is the response to the problem.

Ground-truth answer: 2

LLaDA-8B-Instruct (decoding step: 256) To determine how many watches Tom will fix on the last day, we need to follow these steps:

1. Calculate the total number of days Tom will be fixing watches.
2. Determine the total number of watches Tom will fix in the full days.
3. Subtract the number of watches fixed in the full days from the total number of watches to find out how many watches he will fix on the last day.

First, let's calculate the total number of days Tom will be fixing watches. Since he plans to fix 158 watches at a rate of 12 watches per day, we divide the total number of watches by the rate:

$$\frac{158}{12} = 13$$

This means Tom will be fixing watches for 13 days.

Next, we calculate the total number of watches Tom will fix in the full days:

$$13 \times 12 = 156$$

Finally, we find out how many watches Tom will fix on the last day:

$$158 - 156 = 12$$

The final answer is 12.

AHD (decoding step: 100)

To determine how many watches Tom will fix on the last day, we need to follow these steps:

1. Calculate the total number of watches Tom plans to fix.
2. Determine how many days it will take him to fix the watches at the rate of 12 watches per day.
3. Find out how many watches he will fix on the last day.

First, we know the total number of watches Tom plans to fix is 158.

Next, we need to calculate how many days it will take to fix 158 watches at the rate of 12 watches per day. We do this by dividing the total number of watches by the rate:

$$158 \div 12 = 13 \text{ R } 2$$

This means Tom will need 13 days to fix 158 watches, and he will have 2 watches left to fix on the last day.

Therefore, the number of watches Tom will fix on the last day is 2.

The final answer is 2.

Question:

```
def triples_sum_to_zero(l: list):
    """
    triples_sum_to_zero takes a list of integers as an input.
    it returns True if there are three distinct elements in the list that
    sum to zero, and False otherwise.

    >>> triples_sum_to_zero([1, 3, 5, 0])
    False
    >>> triples_sum_to_zero([1, 3, -2, 1])
    True
    >>> triples_sum_to_zero([1, 2, 3, 7])
    False
    >>> triples_sum_to_zero([2, 4, -5, 3, 9, 7])
    True
    >>> triples_sum_to_zero([1])
    False
    """
```

LLaDA-8B-Instruct (decoding step: 256)

```
def triples_sum_to_zero(l: list):
    """
    triples_sum_to_zero takes a list of integers as an input.
    it returns True if there are three distinct elements in the list that
    sum to zero, and False otherwise.

    >>> triples_sum_to_zero([1, 3, 5, 0])
    False
    >>> triples_sum_to_zero([1, 3, -2, 1])
    True
    >>> triples_sum_to_zero([1, 2, 3, 7])
    False
    >>> triples_sum_to_zero([2, 4, -5, 3, 9, 7])
    True
    >>> triples_sum_to_zero([1])
    False
    """
    l.sort()
    for i in range(len(l) - 2):
        for j in range(i + 1, len(l)):
            if l[i] + l[j] + l[j + 1] == 0:
                return True
    return False

# Test cases
if __name__ == "__main__":
    import doctest

    doctest.testmod()
```

AHD (decoding step: 29)

```
def triples_sum_to_zero(l: list):
    """
    triples_sum_to_zero takes a list of integers as an input.
    it returns True if there are three distinct elements in the list that
    sum to zero, and False otherwise.

    >>> triples_sum_to_zero([1, 3, 5, 0])
    False
    >>> triples_sum_to_zero([1, 3, -2, 1])
    True
    >>> triples_sum_to_zero([1, 2, 3, 7])
    False
    >>> triples_sum_to_zero([2, 4, -5, 3, 9, 7])
    True
    >>> triples_sum_to_zero([1])
    False
    """
    n = len(l)
    for i in range(n):
        for j in range(i + 1, n):
            for k in range(j + 1, n):
                if l[i] + l[j] + l[k] == 0:
                    return True
    return False

import doctest
if __name__ == "__main__":
    doctest.testmod()
```

Case E.3: MMLU-Pro, Generation Length 256, Block Length 32

Question: The following are multiple choice questions (with answers) about math. Think step by step and then finish your answer with "the answer is (X)" where X is the correct letter choice.

Question:

A group of scientists wanted to estimate the proportion of geese returning to the same site for the next breeding season. Suppose they decided to increase the sample size from 200 to 2,000. How will this affect the distribution of the sample proportion?

Options:

- A. The shape of the distribution of the sample proportion will change significantly.
- B. There will be no noticeable change in the distribution of the sample proportion.
- C. The distribution of the sample proportion will become more uniform.
- D. The distribution of the sample proportion will become skewed.
- E. The distribution of the sample proportion will be more spread out.
- F. The distribution of the sample proportion will more closely resemble the binomial distribution.
- G. The distribution of the sample proportion will be less spread out.
- H. The spread of the distribution of the sample proportion will remain unaffected.

Answer: Let's think step by step.

A: Let's think step by step.

Ground-truth answer: G

LLaDA-8B-Instruct (decoding step: 256)

To answer this question, let's consider the effect of increasing the sample size on the distribution of the sample proportion.

1. Understanding the Distribution of the Sample Proportion:

- The distribution of the sample proportion is described by the sampling distribution of the binomial proportion.
- As the sample size increases, the sampling distribution of the binomial proportion converges to a normal distribution. This is known as the Central Limit Theorem.

2. Effect of Increasing Sample Size:

- Increasing the sample size from 200 to 2,000 will make the sampling distribution of the binomial proportion more closely resemble a normal distribution.
- This means that the distribution of the sample proportion will become more spread out (i.e., it will have a wider range of values).

3. Conclusion:

- The correct answer is E. The distribution of the sample proportion will be more spread out.
- The answer is (E).

AHD (decoding step: 157)

To answer this question, let's consider the effect of increasing the sample size on the distribution of the sample proportion.

1. Understanding the Distribution of Sample Proportion:

- The distribution of the sample proportion is influenced by the sample size.
- As the sample size increases, the distribution of the sample proportion tends to become more closely related to the population proportion.

2. Effect of Increasing Sample Size:

- When the sample size is increased from 200 to 2,000, the distribution of the sample proportion will become more concentrated around the population proportion.
- This is because a larger sample size provides more information about the population, leading to a more accurate estimate of the population proportion.

3. Conclusion:

- As the sample size increases, the distribution of the sample proportion will become less spread out.
- This is because the sample proportion will be closer to the population proportion, resulting in a narrower range of possible sample proportions.

Therefore, the correct description of how the sample size affects the distribution of the sample proportion is:

- G. The distribution of the sample proportion will be less spread out.

The answer is (G).