Follow the Latent Roadmap: Navigating Revocable Decoding for Diffusion LLMs with Anchor Tokens

Anonymous ACL submission

Abstract

Diffusion Large Language Models (dLLMs) offer a promising avenue for parallel generation but face a trade-off between decoding speed and quality. While revocable decoding strategies attempt to mitigate errors by verifying and remasking tokens, they typically operate within a mixed-quality context. This leads to two critical failures: *Error Propagation*, where new tokens absorb toxic information from erroneous context, and *Local Error Reinforcement*, where errors mutually reinforce each other to evade detection. To alleviate these challenges, we propose ASRD (Anchor Supervised Revocable Decoding), a training-free framework that operates within the embedding space. ASRD explicitly decouples the decoding context into trusted *Anchor Tokens*, which are identified via temporal consistency, and uncertain candidates. Leveraging a dynamic Anchor Tokens Cache, we introduce two complementary mechanisms: (1) Anchor-Guided Generation, which injects entropy-weighted anchor signals into masked positions to implicitly rectify attention toward the reliable global skeleton; and (2) Anchor-Perturbed Verification, which applies orthogonal perturbations to uncertain candidate tokens, destabilizing and remasking errors driven by fragile local consensus. Extensive experiments on math and coding benchmarks demonstrate that ASRD outperforms recent remasking baselines, achieving accuracy improvements of up to 6.4% while accelerating inference throughput by up to 7.2×.

1 Introduction

Diffusion Large Language Models (dLLMs) (Nie et al., 2025; Ye et al., 2025; Song et al., 2025b) decode a sequence by progressively unmasking tokens from a fully-masked initialization, enabling parallel generation of multiple tokens per step rather than the sequential decoding of autoregressive models (Achiam et al., 2023; Grattafiori et al., 2024; Kang et al., 2025; Li et al., 2025b). However,

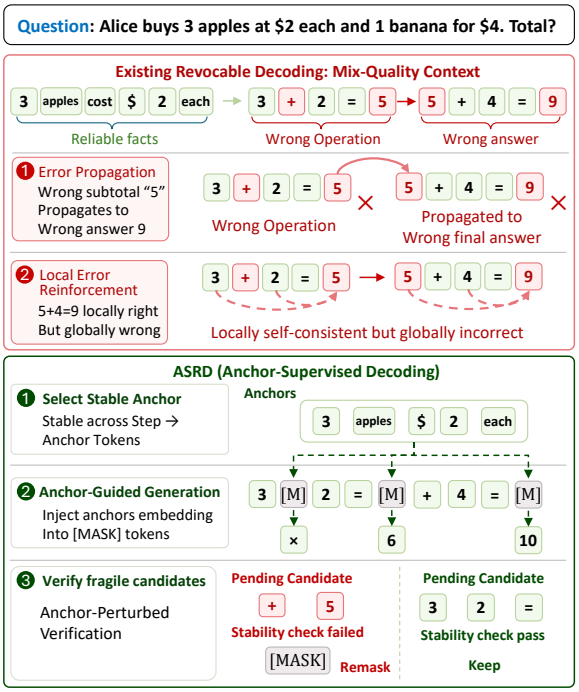


Figure 1: Overview of ASRD and its motivation. Existing revocable decoding verifies and generates tokens under a mixed-quality context, causing error propagation and local error reinforcement. ASRD instead selects temporally stable tokens as anchors, injects anchor guidance into masked positions, and verifies pending candidates with anchor-based perturbations.

current threshold-based unmasking strategies (Wu et al., 2025; Chen et al., 2025; Hong et al., 2025) face a trade-off between speed and quality: a lower threshold commits more tokens per step but admits low-confidence predictions whose errors propagate through later steps (Azangulov et al., 2025). Revocable decoding (Hong et al., 2025; Dong et al., 2025; Wang et al., 2025a) mitigates this by verifying committed tokens within the evolving sequence and *remasking* those judged unreliable. To preserve parallelism, verification and generation share a single forward pass (Hong et al., 2025), which forces every step to operate on a *mixed-quality context* that

interleaves reliable and tentative commitments.

This mixed-quality context compromises both the generation and verification phases of revocable decoding, leading to two critical challenges, as shown in Figure 1: (1) **Error Propagation** (Azangulov et al., 2025; Bie et al., 2025): `[MASK]` tokens are allowed to freely attend to the pending tokens during generation. This, however, hampers the new tokens' quality, since the generating tokens indiscriminately treat all pending tokens as reliable context, absorbing the potential errors from the pending tokens. Consequently, errors are propagated through the entire generation process, leading to incorrect final outputs. (2) **Local Error Reinforcement**: Pending tokens are also verified while surrounded by the potential errors from other pending tokens. These erroneous tokens reinforce each other into a locally consistent but globally incorrect cluster.

Addressing the above two failure modes requires two steps: first identifying which committed tokens are trustworthy, then using them to steer the remaining unconfirmed positions. We propose **ASRD** (Anchor Supervised Re-vocable Decoding), which carries out both steps in the embedding space, preserving compatibility with FlashAttention (Dao et al., 2022). The core of ASRD is to promote temporally consistent tokens into a dynamic **Anchor Tokens Cache (ATC)**, and to use the resulting trusted skeleton to steer every unconfirmed position. Based on ATC, we propose two complementary updates: (1) **Anchor-Guided Generation** injects an entropy-weighted anchor signal into mask embeddings to mitigate Error Propagation; (2) **Anchor-Perturbed Verification** probes committed tokens with an orthogonal anchor-derived signal, flipping only those sustained by Local Error Reinforcement so that they can be remasked for regeneration.

Our contributions are threefold: (1) We propose ASRD, a training-free revocable decoding paradigm whose intervention is confined to the embedding space, supplying anchor-derived guidance to every unconfirmed position. (2) We introduce two embedding-level strategies to handle mixed-quality contexts. (3) Serving as a *plug-and-play enhancement* for existing dLLMs, ASRD is a training-free strategy boosts performance on math and coding.

2 Related Work

Diffusion Large Language Models (dLLMs). To overcome the sequential generation bottleneck of autoregressive LLMs, diffusion LLMs (dLLMs) have emerged as a promising alternative with potential parallel generation. Early attempts span both continuous (Mahabadi et al., 2024; Strudel et al., 2022; Gong et al., 2022; Li et al., 2022) and discrete formulations (He et al., 2023; Sahoo et al., 2024; Austin et al., 2021a; Ou et al., 2024; Lou et al., 2023). Among them, masked diffusion models (MDMs) have shown the most convincing scalability (Gong et al., 2024): they operate directly in token space and iteratively denoise by predicting masked tokens for a fixed number of steps. Recent progress includes strong open-source models such as LLaDA (Nie et al., 2025) and Dream (Ye et al., 2025), as well as extensions to coding (Gong et al., 2025; Xie et al., 2025), and reasoning (Zhao et al., 2025; Wang et al., 2025c; Zhu et al., 2025b). Despite this progress, dLLMs still face a gap between theoretical parallelism and practical performance, due to both expensive bidirectional attention and error accumulation under aggressive parallel unmasking.

Efficient Inference for dLLMs. Existing efforts to accelerate dLLMs mainly fall into two directions: (i) reducing attention cost via approximate/reusable KV computation (Wu et al., 2025; Liu et al., 2025; Song et al., 2025a; Ma et al., 2025), and (ii) improving parallel decoding via dynamic thresholding (Israel et al., 2025; Wang et al., 2025b) or error correction (Wang et al., 2025a; Hong et al., 2025; Dong et al., 2025). Dynamic thresholding adaptively adjusts how many tokens to unmask per step based on confidence, while error correction verifies newly unmasked tokens and remasks low-quality ones, which is generally more robust for complex tasks. For instance, WINO (Hong et al., 2025) employs a draft-and-verify mechanism with dual thresholds, and Saber (Dong et al., 2025) combines confidence-aware acceleration with backtracking and remasking. However, existing error-correction methods often verify tokens under a *mixed-quality* context that may already contain error tokens.

Instead, our work addresses this issue by introducing an *anchor-based context* that explicitly decouples the context into trusted *anchors* and uncertain *candidates*, breaking local reinforcement and enabling more reliable generation and verification.

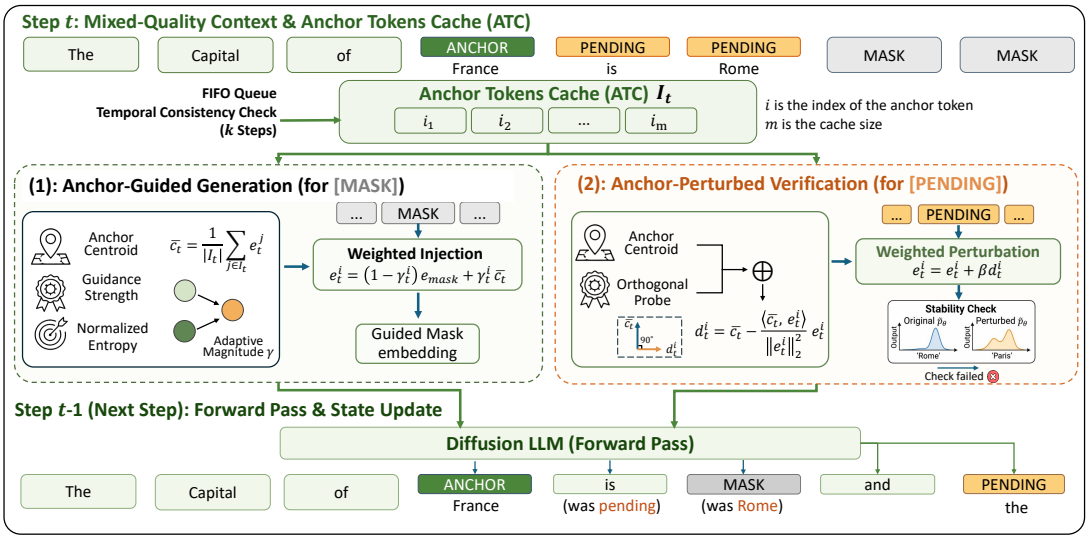


Figure 2: Overview of ASRD. At each decoding step, threshold-passed tokens are classified by temporal consistency: those stable across k consecutive steps become *anchors* (stored in the ATC), while the rest are *pending tokens* awaiting verification. The anchor centroid $\bar{c}_t$ then supervises every unconfirmed position through two embedding-space updates. **Anchor-Guided Generation** (§4.2) replaces each mask embedding with an entropy-weighted convex blend toward $\bar{c}_t$, injecting a trusted prior before generation and mitigating *Error Propagation*. **Anchor-Perturbed Verification** (§4.3) adds a fixed-coefficient probe orthogonal to the committed embedding, stress-testing pending tokens and flipping those sustained by *Local Error Reinforcement* so they can be remasked.

3 Preliminaries: Revocable Parallel Decoding

For dLLMs (Austin et al., 2021a; Ou et al., 2024) and a targeting discrete decoding sequences: $\mathbf{x}_0 \in \{1, \ldots, V\}^L$, where V is the vocabulary size and L is the sequence length, the forward process progressively corrupts tokens to a special [MASK] state: $q(x_t^i = \texttt{[MASK]} \mid x_{t-1}^i) = \omega_t$, where each token at i-th position, $i \in \{0, 1, ..., L-1\}$, is independently masked with probability ω_t at step t, and $t \in \{1, 2, ..., T\}$. Accordingly, the reverse process $p_\theta(\mathbf{x}_t|\mathbf{x}_{t+1})$ with parameter θ aims to iteratively unmask sequence from $\mathbf{x}_T$ (fully masked) to $\mathbf{x}_0$ (clean text).

$$\underbrace{x_0 \xrightarrow{q} x_1 \xrightarrow{q} \cdots \xrightarrow{q}}_{\text{forward masking (noising)}} x_T \underbrace{\xrightarrow{p_\theta} x_{T-1} \xrightarrow{p_\theta} \cdots \xrightarrow{p_\theta} x_0}_{\text{reverse unmasking (denoising)}}$$

In threshold-based parallel decoding, at step t, based on the given state in previous step $t+1$, the model predicts $p_\theta(x_t^i \mid \mathbf{x}_{t+1})$ for all masked positions i simultaneously. A token is unmasked and finalized if its confidence exceeds a pre-defined threshold τ.

Revocable parallel decoding adds a verification-and-revision mechanism atop threshold decoding. At each decoding step t, the procedure consists of three phases: drafting, verification, and revision. Let $\mathcal{M}_t = \{i : x_t^i = \texttt{[MASK]}\}$ denote the set of all mask positions at step t.

Drafting. Select positions whose confidence exceeds τ: $\mathcal{T}_t = \{i \in \mathcal{M}_{t+1} : \max_v p_\theta(x_0^i{=}v \mid \mathbf{x}_{t+1}) > \tau\}$ and assigns each $i \in \mathcal{T}_t$ a candidate token $\breve{x}_t^i = \arg\max_v p_\theta(x_0^i{=}v \mid \mathbf{x}_{t+1})$.

Verification. For each $i \in \mathcal{T}_t$, a masked context $\mathbf{x}_t^{\setminus i}$ is built by re-masking position i while keeping the other drafts in place; a forward pass yields the re-evaluated distribution $p_\theta(x_0^i \mid \mathbf{x}_t^{\setminus i})$, and a criterion $\mathcal{F}$ decides whether the draft survives.

Revision. After verification, the set of failed positions is denoted as $\mathcal{R}_t = \{i \in \mathcal{T}_t : \mathcal{F}(i) = \text{fail}\}$. Common revision rules include: *Top-1 Stability* (position i fails if $\arg\max_v p_\theta(x_{t-1}^i{=}v \mid \mathbf{x}_t^{\setminus i}) \neq \breve{x}_t^i$); *Confidence threshold* (position i fails if $\max_v p_\theta(x_{t-1}^i{=}v \mid \mathbf{x}_t^{\setminus i}) < \tau'$ for some threshold τ'), and *Margin criterion* (position i fails if the probability margin decreases significantly). Failed positions are revoked by resetting them to the masked state.

4 Methodology

To address both failure modes in Section 1, we propose ASRD: rather than letting generation and verification share an unfiltered mixed-quality context, we maintain a set of anchors that supervises both. As shown in Figure 2, at each step we partition newly generated tokens by temporal consistency into anchor tokens ($\hat{\cdot}$) and pending tokens

(˘). Anchors are admitted into the Anchor Tokens Cache (ATC) that forms the trusted backbone of the sequence, and in the next step they supervise the generation of [MASK] tokens via an entropy-weighted injection that suppresses error propagation, as well as the verification of pending tokens via a direction-specific perturbation that exposes local error reinforcement.

4.1 Anchor Tokens Cache (ATC)

Not every threshold-passed token deserves to anchor downstream decisions: some merely exhibit transient confidence that fluctuates in later steps (Chen et al., 2025; Li et al., 2025a). We therefore promote a token to an *anchor* only when its top-1 prediction stays consistent across k consecutive steps. Concretely, let $\hat{a}_s^i = \arg\max_v p_\theta(x_0^i{=}v \mid \mathbf{x}_{s+1})$ denote the top-1 prediction at step s. We define the per-step anchor candidates at step t as

$$\mathcal{A}_t = \left\{ i \in \mathcal{T}_t \;:\; \hat{a}_{t-k+1}^i = \hat{a}_{t-k+2}^i = \cdots = \hat{a}_t^i \right\}$$

We show that the temporal consistency exponentially suppresses false anchors in idealized assumptions. Moreover, observation in experiments (Table 3) empirically shows rapid drop in error rate.

Proposition 1 (Exponential Suppression of False Anchors)**.** *Assume that (i) there exists a positive semantic margin $\Delta_{\text{sem}} > 0$ between the ground-truth token and any incorrect token w in the ideal distribution, and (ii) the model prediction can be decomposed as $p_\theta = p^* + \epsilon$, and for any two candidates the differential noise $\eta_s = \epsilon_{s,x_0^i} - \epsilon_{s,w}$ is zero-mean, symmetric around 0, and i.i.d. across decoding steps. Then, for any position i and any incorrect token $w \neq x_0^i$, the probability that w is the top-1 prediction at position i for k consecutive steps decays exponentially in k:*

$$\mathbb{P}\big(\hat{a}_{t-k+1}^i = \cdots = \hat{a}_t^i = w\big) \leq \exp(-\lambda k)$$

for some $\lambda > 0$ determined by the semantic margin and the noise distribution.[1]

ATC Management. To track the evolving sequence backbone, we maintain a fixed-size Anchor Tokens Cache (ATC) $\mathcal{I}_t$ of capacity m that accumulates per-step candidates from $\mathcal{A}_t$ under a first-in-first-out (FIFO) policy:

$$\mathcal{I}_t = \text{FIFO}_m(\mathcal{I}_{t-1} \cup \{i : i \in \mathcal{A}_t\})$$

[1]Proof see in Appendix C.1.

where each entry records a position index i. This policy aligns with the approximate left-to-right confirmation order of dLLM unmasking (Horvitz et al., 2025), so a FIFO cache retains the most recently stabilized region of the backbone.

4.2 Anchor-Guided Generation: Combating Error Propagation

A masked position carries no semantic content; its embedding is the generic $\mathbf{e}_{\text{mask}}$, leaving the subsequent forward pass vulnerable to pollution from erroneous pending neighbors (Azangulov et al., 2025; Bie et al., 2025). We counter this by injecting an anchor-derived prior into mask embedding before the forward pass. To this end, we summarize the trusted backbone by the anchor centroid at step t,

$$\bar{c}_t = \frac{1}{|\mathcal{I}_t|} \sum_{j \in \mathcal{I}_t} \mathbf{e}_t^j$$

where each $\mathbf{e}_t^j$ is the input-layer word embedding of the anchor token committed at position j, so the centroid pools the input embeddings of all positions currently held in the ATC and serves as the source of anchor supervision for both the mask and pending updates.

Given this centroid, for each mask position $i \in \mathcal{M}_t$ we replace the uninformative mask embedding by a convex blend toward $\bar{c}_t$:

$$\mathbf{e}_t^i = (1 - \gamma_t^i)\, \mathbf{e}_{\text{mask}} + \gamma_t^i\, \bar{c}_t$$

The mask representation is therefore steered into the trusted subspace before any attention is computed.

To avoid over-correcting reliable predictions, we scale the update magnitude by the normalized prediction entropy of each mask position, taken from the previous step's logits at the same position so that no extra forward pass is required:

$$\gamma_t^i = \alpha\, \bar{E}_t^i$$

where $\alpha \in [0, 1]$ controls the guidance strength and $\bar{E}_t^i \in [0, 1]$ is the min–max normalized entropy of position i across all mask positions:

$$\bar{E}_t^i = \begin{cases} \dfrac{E_t^i - E_{\min}}{E_{\max} - E_{\min}}, & \text{if } E_{\max} > E_{\min} \\ 0, & \text{otherwise} \end{cases}$$

Consequently, high-entropy masks receive stronger anchor supervision, while low-entropy masks remain nearly unchanged.

Implicit attention rectification. Although the above embedding update modifies embeddings rather than attention scores, the induced query shift biases subsequent attention toward anchor-aligned context. We provide an intuition for this in Appendix C.2.

4.3 Anchor-Perturbed Verification: Breaking Local Error Reinforcement

Let *pending set* be the threshold-passed positions that have not been anchored: $\mathcal{P}_t = \mathcal{T}_t \setminus \mathcal{A}_t$. A pending token at position $i \in \mathcal{P}_t$ already carries a committed embedding $\mathbf{e}_t^i$ whose top-1 prediction we wish to verify. However, this semantic direction may be sustained by *local error reinforcement* with neighboring errors rather than by the global trajectory. Because direct overwriting, as in the mask update, would be destructive, we instead apply a non-destructive *probe* that reuses the anchor centroid $\bar{c}_t$ defined above.

To keep the probe orthogonal to the committed direction, we extract the component of $\bar{c}_t$ orthogonal to $\mathbf{e}_t^i$:

$$\mathbf{d}_t^i = \bar{c}_t - \frac{\langle \bar{c}_t, \mathbf{e}_t^i \rangle}{\|\mathbf{e}_t^i\|_2^2}\, \mathbf{e}_t^i$$

Since $\mathbf{d}_t^i$ is the orthogonal residual of $\bar{c}_t$ with respect to $\mathbf{e}_t^i$, its norm is bounded by $\|\bar{c}_t\|_2$. Thus, the probe remains on the same scale regardless of how pending tokens change.

We then add this probe to the committed embedding with a constant coefficient β, yielding the perturbed embedding

$$\mathbf{e}_t^i = \mathbf{e}_t^i + \beta\, \mathbf{d}_t^i$$

Anchor-aligned tokens induce small residuals and are only weakly perturbed, whereas consensus-sustained errors produce larger orthogonal residuals and receive stronger probes. The perturbed embeddings then enter the next forward pass, and we adopt the Top-1 Stability criterion from Section 3 as our verification rule: position i fails and is remasked whenever its post-perturbation top-1 prediction differs from the committed draft $\breve{x}_t^i$.

5 Experiments

5.1 Experimental Setup

Implementation Details. We evaluate on two representative dLLMs, LLaDA-8B (Nie et al., 2025; Zhu et al., 2025a) and Dream-7B (Ye et al., 2025). The main results in Section 5.2 use their instruct variants (LLaDA-8B-Ins and Dream-7B-Ins), while results on the base and RL-tuned variants are reported in Appendix B. We adopt the Semi-AR sampling strategy from LLaDA with block size 32 throughout. Hyperparameter sensitivity is examined in Section 5.4, and additional configuration details are provided in Appendix A.

Datasets. We use four benchmarks covering two task families. GSM8K (Cobbe et al., 2021) and MATH500 (Lightman et al., 2024) target mathematical reasoning, while MBPP (Austin et al., 2021b) and HumanEval (Chen et al., 2021) target code generation. Detailed dataset descriptions are deferred to Appendix A.

Metrics. We report three metrics. For effectiveness, *Accuracy* corresponds to final-answer accuracy on the math benchmarks and pass@1 on the code benchmarks. For efficiency, *Steps* is the number of forward passes required to generate a fixed-length sequence, which provides a model-agnostic measure of decoding cost, and *Speedup* is the wall-clock time reduction relative to the standard decoder, which we use as the primary efficiency metric.

Baselines. We compare ASRD against the standard decoder and two state-of-the-art revocable-decoding methods, WINO (Hong et al., 2025) and Saber (Dong et al., 2025). To ensure a fair comparison, both baselines are re-implemented from their official repositories within the LM-Eval (Gao et al., 2024) framework, and all methods share the same decoding configuration as ASRD.

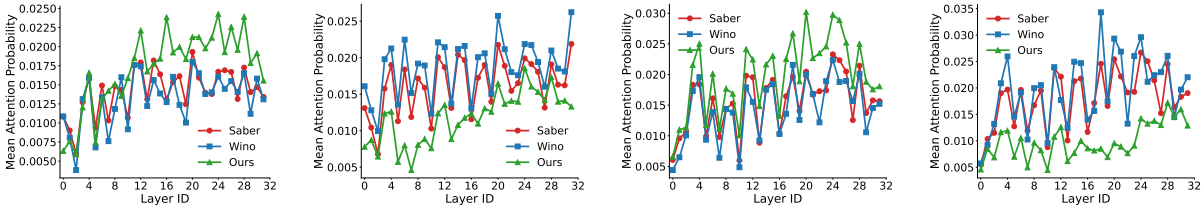
5.2 Main Results

Accuracy Performance. Table 1 summarizes accuracy across all configurations. ASRD achieves the best average accuracy on every benchmark and backbone, with representative gains of $+4.9\%$ on LLaDA-Ins-8B for HumanEval at length 512 and $+6.4\%$ on Dream-Ins-7B for MATH500 at length 512. Existing baselines occasionally fall below the standard decoder, as in WINO on Dream-Ins-7B HumanEval, which reflects the cost of verifying against a context of mixed quality. The advantage of ASRD also amplifies with sequence length, growing from $+3.8\%$ to $+6.4\%$ on Dream-Ins-7B MATH500 as the length doubles, consistent with the ATC providing a stronger global skeleton over longer horizons.

Efficiency and Decoding Speed. ASRD achieves wall-clock speedups of $2.5\times$ to $7.2\times$ across all configurations. Anchor-Guided Gener-

Model	Len	Method	HumanEval			MBPP			GSM8K			MATH500		
			Acc ↑	Steps ↓	Speed ↑	Acc ↑	Steps ↓	Speed ↑	Acc ↑	Steps ↓	Speed ↑	Acc ↑	Steps ↓	Speed ↑
LLaDA-Ins-8B	256	baseline	38.7	256.0	1.0×	36.8	256.0	1.0×	77.4	256.0	1.0×	33.8	256.0	1.0×
		WINO	37.5	75.8	2.3×	36.2	91.1	1.9×	77.3	**53.7**	2.9×	34.2	78.6	2.0×
		Saber	39.4	101.2	2.4×	37.8	115.4	2.1×	77.8	120.5	1.9×	34.8	135.2	1.8×
		Ours	**41.5**$_{+2.8}$	**75.4**	**3.1×**	**38.6**$_{+1.8}$	**88.3**	**2.5×**	**79.5**$_{+2.1}$	56.9	**3.2×**	**35.2**$_{+1.4}$	**74.8**	**2.9×**
	512	baseline	43.9	512.0	1.0×	38.2	512.0	1.0×	80.3	512.0	1.0×	37.8	512.0	1.0×
		WINO	43.9	133.4	2.8×	37.4	133.3	2.4×	78.7	**83.1**	3.6×	35.6	120.6	2.6×
		Saber	44.5	198.3	2.3×	37.8	188.9	2.6×	79.7	141.7	3.2×	37.4	192.5	2.5×
		Ours	**48.8**$_{+4.9}$	**127.8**	**3.6×**	**39.4**$_{+1.2}$	**110.4**	**2.9×**	**81.1**$_{+0.8}$	88.5	**4.3×**	**38.8**$_{+1.0}$	**119.3**	**3.8×**
Dream-Ins-7B	256	baseline	54.9	256.0	1.0×	57.4	256.0	1.0×	80.4	256.0	1.0×	37.8	256.0	1.0×
		WINO	51.3	83.8	2.4×	56.8	64.7	3.6×	78.6	72.7	3.4×	38.4	126.6	1.6×
		Saber	52.5	165.1	1.5×	57.6	191.0	1.3×	79.9	185.6	1.3×	38.8	179.4	1.4×
		Ours	**56.1**$_{+1.2}$	**82.6**	**3.1×**	**58.8**$_{+1.4}$	**62.4**	**4.1×**	**80.7**$_{+0.3}$	**65.6**	**3.9×**	**41.6**$_{+3.8}$	**78.1**	**3.5×**
	512	baseline	56.1	512.0	1.0×	56.2	512.0	1.0×	80.2	512.0	1.0×	38.6	512.0	1.0×
		WINO	51.3	102.5	3.9×	55.4	86.1	5.2×	79.5	92.7	5.5×	39.8	139.9	2.8×
		Saber	53.1	264.6	1.9×	55.8	388.2	1.3×	79.9	314.5	1.6×	41.4	250.7	2.0×
		Ours	**56.7**$_{+0.6}$	**85.3**	**5.5×**	**57.2**$_{+1.0}$	**74.5**	**7.2×**	**81.1**$_{+0.9}$	**78.2**	**6.8×**	**45.0**$_{+6.4}$	**94.8**	**5.4×**

Table 1: Performance of different instruct (ins) models and methods across benchmarks. Speed denotes relative runtime (baseline = 1.0×). Baseline results are shown in grey, and ASRD improvements in green.



(a) Mask-to-anchor attention: Anchor-Guided Generation increases attention from mask to anchor tokens.

(b) Mask-to-pending attention: Anchor-Guided Generation suppresses attention to uncertain pending tokens.

(c) Pending-to-anchor attention: Anchor-Perturbed Verification strengthens alignment with the backbone.

(d) Pending-to-pending attention: Anchor-Perturbed Verification disrupts mutual reinforcement.

Figure 3: *Layer-wise attention redistribution induced by ASRD.* (a)–(b): **Anchor-Guided Generation** implicitly rectifies attention; mask tokens reallocate weight from pending neighbors to anchor tokens. (c)–(d): **Anchor-Perturbed Verification** induces a complementary effect; orthogonal probing increases the attention from pending tokens to anchors while dampening mutual reinforcement among pending tokens.

ation produces higher-quality drafts that require fewer subsequent corrections, which directly translates accuracy gains into a smaller number of decoding steps. The largest speedup appears on Dream-Ins-7B for MBPP at length 512, where ASRD reaches 7.2×, compared to 5.2× for WINO and 1.3× for Saber. The efficiency advantage also scales with sequence length, growing from 4.1× to 7.2× on the same task.

5.3 Empirical Evidence: Attention Redistribution

To validate the implicit attention rectification claimed in Section 4.2, the per-layer attention matrices are recorded and the mean attention probability between tokens of different roles is reported. Under Anchor-Guided Generation, mask tokens consistently allocate higher attention to anchors (Figure 3a) and lower attention to pending tokens (Figure 3b), with the effect most pronounced in deeper layers where semantic composition is believed to occur. An embedding-space injection alone is therefore enough to achieve attention rectification, without modifying any attention score and while preserving FlashAttention. A similar redistribution emerges under Anchor-Perturbed Verification (Figures 3c–3d): the orthogonal perturbation raises the attention from pending tokens to anchors while lowering the attention among pending tokens themselves, which complements the stability-probing rule of Section 4.3 by weakening local error reinforcement.

5.4 Ablation Study

Module ablation. Table 2 reports a module-level ablation. Removing Anchor-Guided Generation causes the largest accuracy loss on code generation, with HumanEval dropping by 2.5% at length 256 and 1.3% at length 512, while decoding steps grow correspondingly. Without an anchor-derived prior, the forward pass absorbs error from pending neighbors, which yields lower-quality drafts. Re-

Length	Method	HumanEval			MBPP			GSM8K			MATH500		
		Acc ↑	Steps ↓	Speed ↑	Acc ↑	Steps ↓	Speed ↑	Acc ↑	Steps ↓	Speed ↑	Acc ↑	Steps ↓	Speed ↑
256	baseline	38.7	256.0	1.0×	36.8	256.0	1.0×	77.4	256.0	1.0×	33.8	256.0	1.0×
	Ours	$41.5_{+2.8}$	75.4	3.1×	$38.6_{+1.8}$	88.3	2.5×	$79.5_{+2.1}$	56.9	3.2×	$35.2_{+1.4}$	74.8	2.9×
	w/o Anchor-Guided Generation	$39.0_{-2.5}$	86.2	2.8×	$37.6_{-1.0}$	102.5	2.3×	$78.4_{-1.1}$	78.4	3.0×	$34.0_{-1.2}$	89.6	2.7×
	w/o Anchor-Perturbed Verification	$40.3_{-1.2}$	68.9	3.5×	$38.0_{-0.6}$	81.3	2.9×	$79.1_{-0.4}$	67.1	3.5×	$33.6_{-1.6}$	75.6	3.2×
512	baseline	43.9	512.0	1.0×	38.2	512.0	1.0×	80.3	512.0	1.0×	37.8	512.0	1.0×
	Ours	$48.8_{+4.9}$	127.8	3.6×	$39.4_{+1.2}$	110.4	2.9×	$81.1_{+0.8}$	88.5	4.3×	$38.8_{+1.0}$	119.3	3.8×
	w/o Anchor-Guided Generation	$47.5_{-1.3}$	148.2	3.3×	$38.4_{-1.0}$	144.1	2.7×	$80.8_{-0.3}$	104.5	4.1×	$38.6_{-0.2}$	134.1	3.6×
	w/o Anchor-Perturbed Verification	$46.9_{-1.9}$	122.3	4.0×	$38.6_{-0.8}$	121.7	3.2×	$80.4_{-0.7}$	93.2	4.6×	$37.0_{-1.8}$	117.8	4.1×

Table 2: Module-level ablation on LLaDA-Instruct-8B. "w/o Anchor-Guided Generation" disables the mask-side update ($\gamma_t^i = 0$ on the mask role); "w/o Anchor-Perturbed Verification" disables the pending-side update ($\beta = 0$ on the pending role). Red and green subscripts show the accuracy change relative to full ASRD. Design-level ablations (magnitude form on mask, direction form on pending) are reported in Table 5.

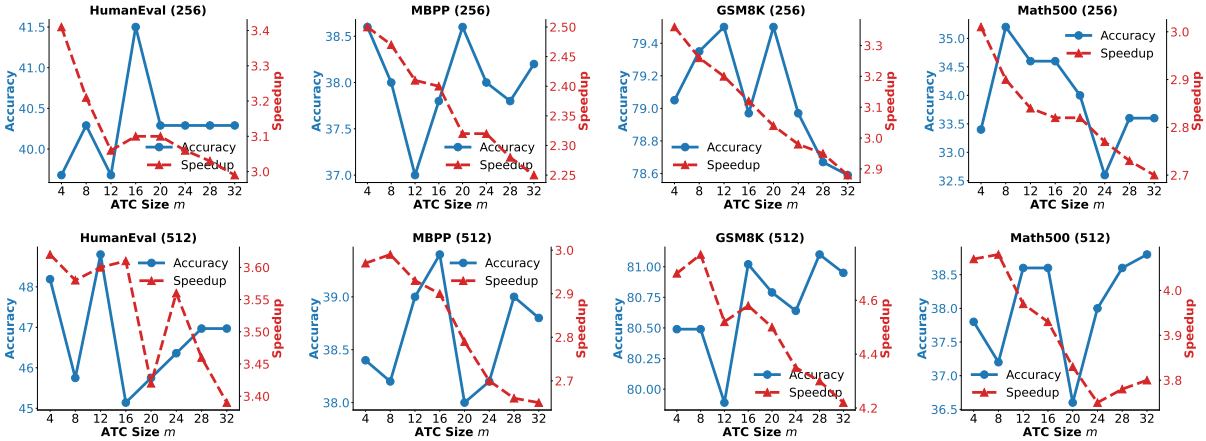


Figure 4: Ablation on the ATC size $m \in [4, 32]$ for LLaDA-Instruct-8B across sequence lengths and benchmarks; block size 32 throughout.

moving Anchor-Perturbed Verification consistently hurts MATH500, with drops of 1.6% at length 256 and 1.8% at length 512. The speedup correspondingly increases in this variant (e.g., from 3.6× to 4.0× on HumanEval at length 512), reflecting that verification introduces additional refinement steps.

Effect of ATC size m. Figure 4 presents the sensitivity analysis of ATC capacity $m \in [4, 32]$. Accuracy follows an inverted-U pattern: a cache that is too small cannot represent the sequence backbone, while one that is too large retains stale anchors that drift away from the current decoding frontier. The location of the peak is mildly task-dependent, but m in $[12, 20]$ works well across the four benchmarks. For speedup, larger m values lead to lower throughput due to the increased computational overhead of anchor-derived operations.

Effect of consistency window k. Table 3 sweeps $k \in \{1, 2, 3, 4\}$ and additionally reports the fraction of anchor tokens that fall inside Claude-Opus-4.6-annotated error spans (*Contam.*). As k grows, contamination decays sharply on both benchmarks, which is the empirical signature of

Dataset	k	Acc (%) ↑	Anchors/sample	Contam. (%) ↓
GSM8K	1	79.7	307.6	6.68
	2	**81.1**	101.5	2.56
	3	81.0	87.1	1.79
	4	78.5	35.3	**1.74**
HumanEval	1	43.9	338.4	43.53
	2	**48.8**	134.4	11.28
	3	48.5	107.9	9.45
	4	45.1	50.9	**8.87**

Table 3: Effect of the consistency window k for ASRD on LLaDA-Instruct-8B (sequence length 512). *Anchors/sample* is the average number of tokens promoted to the ATC per sample; *Contam. (%)* is the fraction of anchor tokens that fall inside error spans of the generated output, annotated token-level by Claude-Opus-4.6 against the reference solution (lower is better).

the q^k bound in Proposition 1: each additional consistency step multiplies the false-anchor probability by $q \ll 1$. The number of admitted anchors shrinks even faster, since the same filter also rejects briefly fluctuating tokens. The two effects compose into a non-monotonic accuracy curve, and k=2 sits at the balance point between purity and density, which

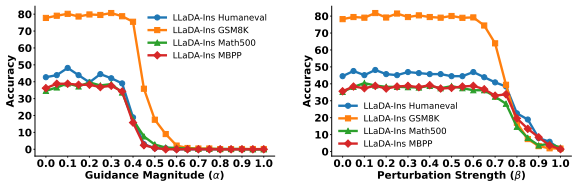
motivates it as the default.

Effect of block size. Table 4 reports the effect of the Semi-AR block size. A block size of 32 yields the best accuracy across all configurations. Doubling to 64 causes a mild degradation, while 128 leads to pronounced drops, with GSM8K falling to 72.4% at length 512, since anchor tokens become too sparsely distributed to steer the current block. Speedup follows the same trend on GSM8K, whereas HumanEval throughput stays stable.

Length	Method	HumanEval		GSM8K	
		Acc ↑	Speed ↑	Acc ↑	Speed ↑
256	baseline	38.7	1.0×	77.4	1.0×
	Ours (BS = 32)	**41.5**	**3.1×**	**79.5**	3.2×
	Ours (BS = 64)	40.9	3.0×	78.3	**3.3×**
	Ours (BS = 128)	38.7	3.0×	70.8	2.5×
512	baseline	43.9	1.0×	80.3	1.0×
	Ours (BS = 32)	**48.8**	3.6×	**81.1**	**4.3×**
	Ours (BS = 64)	46.3	3.5×	80.0	4.2×
	Ours (BS = 128)	45.7	**3.7×**	72.4	2.3×

Table 4: Effect of the Semi-AR block size on LLaDA-Instruct-8B. Block size 32 attains the best accuracy across configurations.

Sensitivity of α and β. Figure 5 sweeps the two knobs of the embedding-space updates: the mask-side guidance strength α and the pending-side perturbation strength β defined in Section 4.2 and Section 4.3. Both follow an inverted-U pattern with peaks in $\alpha \in [0.05, 0.3]$ and $\beta \in [0.1, 0.6]$. The two differ in their failure mode at large values: extreme α catastrophically overwrites the mask representation, whereas extreme β degrades only gradually, since committed embeddings of pending tokens partially resist the orthogonal perturbation.



(a) Guidance Strength α (b) Perturbation Strength β

Figure 5: Sensitivity to the two embedding-space-update knobs α (mask side) and β (pending side) on LLaDA-Instruct-8B at sequence length 512.

Design choices in the embedding-space updates. Table 5 dissects the two design degrees of freedom in Anchor-Guided Generation and Anchor-Perturbed Verification of Section 4.2 and Section 4.3: the *magnitude form* on the mask side and the *direction form* on the pending side. For the magnitude form, replacing the entropy-modulated $\gamma_t^i = \alpha \bar{E}_t^i$ with a constant $\gamma_t^i = \alpha$ produces a consistent but moderate drop, which confirms that per-token entropy modulation contributes beyond the global α. For the direction form, removing the probe loses up to 4.3%, while Gaussian noise and random embeddings lose up to 7.4% and 6.6%, since they corrupt valid tokens by discarding the anchor signal. Even the parallel probe, which preserves the anchor signal but pushes along $\mathbf{e}_t^i$, fails to expose fragile predictions because it merely reinforces the token's existing direction. Orthogonality is therefore the load-bearing design choice.

Length	Variant	HumanEval		GSM8K	
		Acc ↑	Speed ↑	Acc ↑	Speed ↑
256	**Ours (full)**	**41.5**	**3.1×**	**79.5**	**3.2×**
	Anchor-Guided Generation: magnitude form				
	Constant α	$40.3_{-1.2}$	3.0×	$78.9_{-0.6}$	3.1×
	Anchor-Perturbed Verification: direction form				
	Parallel probe	$36.0_{-5.5}$	2.8×	$77.1_{-2.4}$	2.7×
	Gaussian noise	$38.4_{-3.1}$	3.0×	$72.6_{-6.9}$	2.6×
	Random embedding	$34.9_{-6.6}$	2.7×	$77.2_{-2.3}$	2.8×
	No perturbation	$37.2_{-4.3}$	2.9×	$77.9_{-1.6}$	3.0×
512	**Ours (full)**	**48.8**	**3.6×**	**81.1**	**4.3×**
	Anchor-Guided Generation: magnitude form				
	Constant α	$46.4_{-2.4}$	3.4×	$80.3_{-0.8}$	4.3×
	Anchor-Perturbed Verification: direction form				
	Parallel probe	$42.6_{-6.2}$	3.2×	$75.9_{-5.2}$	3.9×
	Gaussian noise	$41.4_{-7.4}$	3.0×	$75.1_{-6.0}$	3.8×
	Random embedding	$44.5_{-4.3}$	3.5×	$77.9_{-3.2}$	4.1×
	No perturbation	$45.7_{-3.1}$	3.3×	$78.1_{-3.0}$	4.1×

Table 5: Design-level ablation on the embedding-space updates with LLaDA-Instruct-8B. *Top block (Anchor-Guided Generation: magnitude form)*: Constant α replaces $\gamma_t^i = \alpha \bar{E}_t^i$ with $\gamma_t^i = \alpha$. *Bottom block (Anchor-Perturbed Verification: direction form)*: Parallel probe projects $\bar{c}_t$ onto $\mathbf{e}_t^i$ rather than its orthogonal complement; Gaussian noise is isotropic noise; Random embedding is a random token embedding; No perturbation disables the perturbation.

6 Conclusion

We present ASRD, a training-free revocable decoding framework for dLLMs that operates in the embedding space. By promoting temporally consistent tokens into a dynamic Anchor Tokens Cache, ASRD addresses error propagation and local error reinforcement through two complementary updates: Anchor-Guided Generation injects an entropy-weighted anchor signal into mask embeddings, and Anchor-Perturbed Verification probes committed tokens with an orthogonal anchor-derived signal to remask those sustained by spurious local consensus. Experiments on math and coding benchmarks show consistent accuracy gains across Dream and LLaDA. As a decoding-time procedure that does not alter the underlying model, ASRD inherits the risks of the host dLLM without amplifying them.

Limitations

Our evaluation of ASRD spans instruction-tuned, base, and RL-tuned variants of two diffusion LLM families (Dream and LLaDA) at the 7B–8B scale, covering closed-form math and code benchmarks as well as open-ended question answering and summarization. Several gaps nevertheless remain. The parameter scale we study is capped at 8B, so whether the same anchor dynamics carry over to substantially larger dLLMs, or to multi-turn dialogue and chat-style instruction-following settings, is left to future work. The decoding configuration is also not free of brittleness; accuracy degrades sharply at very large Semi-AR block sizes (GSM8K falls to 72.4% at BS$=128$ on LLaDA-Instruct-8B at sequence length 512), where anchors become too sparsely distributed within a block to supervise it. The per-step overhead is modest (7.6 ms, a 9.8% increase on LLaDA-Instruct-8B) and is amortized by the reduced step count, but the relative wall-clock benefit shrinks on very short sequences where only a handful of decoding steps are needed. On the theoretical side, Proposition 1 relies on idealized assumptions (i.i.d. noise across decoding steps and a positive semantic margin between the ground-truth token and its competitors) that need not hold in practice; in complex contexts an incorrect token may stay top-1 for several steps and slip into the ATC. We therefore treat the exponential bound as a qualitative insight rather than a guarantee on real dLLMs, and rely on the empirical false-anchor decay in Table 3 as the actual evidence that the consistency filter suppresses incorrect anchors as k grows.

References

Josh Achiam, Steven Adler, Sandhini Agarwal, Lama Ahmad, Ilge Akkaya, Florencia Leoni Aleman, Diogo Almeida, Janko Altenschmidt, Sam Altman, Shyamal Anadkat, and 1 others. 2023. Gpt-4 technical report. *arXiv preprint arXiv:2303.08774*.

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. *Preprint*, arXiv:2108.07732.

Iskander Azangulov, Teodora Pandeva, Niranjani Prasad, Javier Zazo, and Sushrut Karmalkar. 2025. Parallel sampling from masked diffusion models via conditional independence testing. *Preprint*, arXiv:2510.21961.

Tiwei Bie, Maosong Cao, Kun Chen, Lun Du, Mingliang Gong, Zhuochen Gong, Yanmei Gu, Jiaqi Hu, Zenan Huang, Zhenzhong Lan, and 1 others. 2025. LLaDA2.0: Scaling up diffusion language models to 100B. *arXiv preprint arXiv:2512.15745*.

Kecheng Chen, Ziru Liu, Xijia Tao, Hui Liu, Xinyu Fu, Suiyun Zhang, Dandan Tu, Lingpeng Kong, Rui Liu, and Haoliang Li. 2025. Beyond confidence: Adaptive and coherent decoding for diffusion language models. *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. *Preprint*, arXiv:2107.03374.

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. *Preprint*, arXiv:2110.14168.

Tri Dao, Dan Fu, Stefano Ermon, Atri Rudra, and Christopher Ré. 2022. Flashattention: Fast and memory-efficient exact attention with io-awareness. *Advances in neural information processing systems*, 35:16344–16359.

Yihong Dong, Zhaoyu Ma, Xue Jiang, Zhiyuan Fan, Jiaru Qian, Yongmin Li, Jianha Xiao, Zhi Jin, Rongyu Cao, Binhua Li, and 1 others. 2025. Saber: An efficient sampling with adaptive acceleration and backtracking enhanced remasking for diffusion language model. *arXiv preprint arXiv:2510.18165*.

Leo Gao, Jonathan Tow, Baber Abbasi, Stella Biderman, Sid Black, Anthony DiPofi, Charles Foster, Laurence Golding, Jeffrey Hsu, Alain Le Noac'h, Haonan Li, Kyle McDonell, Niklas Muennighoff, Chris Ociepa, Jason Phang, Laria Reynolds, Hailey Schoelkopf, Aviya Skowron, Lintang Sutawika, and 5 others. 2024. The language model evaluation harness.

Shansan Gong, Shivam Agarwal, Yizhe Zhang, Jiacheng Ye, Lin Zheng, Mukai Li, Chenxin An, Peilin Zhao, Wei Bi, Jiawei Han, and 1 others. 2024. Scaling diffusion language models via adaptation from autoregressive models. *arXiv preprint arXiv:2410.17891*.

Shansan Gong, Mukai Li, Jiangtao Feng, Zhiyong Wu, and LingPeng Kong. 2022. Diffuseq: Sequence to sequence text generation with diffusion models. *arXiv preprint arXiv:2210.08933*.

Shansan Gong, Ruixiang Zhang, Huangjie Zheng, Jiatao Gu, Navdeep Jaitly, Lingpeng Kong, and Yizhe Zhang. 2025. Diffucoder: Understanding and improving masked diffusion models for code generation. *arXiv preprint arXiv:2506.20639*.

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

Zhengfu He, Tianxiang Sun, Qiong Tang, Kuanning Wang, Xuan-Jing Huang, and Xipeng Qiu. 2023. Diffusionbert: Improving generative masked language models with diffusion models. In *Proceedings of the 61st annual meeting of the association for computational linguistics (volume 1: Long papers)*, pages 4521–4534.

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*.

Zachary Horvitz, Raghav Singhal, Hao Zou, Carles Domingo-Enrich, Zhou Yu, Rajesh Ranganath, and Kathleen McKeown. 2025. No compute left behind: Rethinking reasoning and sampling with masked diffusion models. *Preprint*, arXiv:2510.19990.

Daniel Israel, Guy Van den Broeck, and Aditya Grover. 2025. Accelerating diffusion llms via adaptive parallel decoding. *arXiv preprint arXiv:2506.00413*.

Wonjun Kang, Kevin Galim, Seunghyuk Oh, Minjae Lee, Yuchen Zeng, Shuibai Zhang, Coleman Hooper, Yuezhou Hu, Hyung Il Koo, Nam Ik Cho, and 1 others. 2025. Parallelbench: Understanding the tradeoffs of parallel decoding in diffusion llms. *arXiv preprint arXiv:2510.04767*.

Tom Kwiatkowski, Jennimaria Palomaki, Olivia Redfield, Michael Collins, Ankur Parikh, Chris Alberti, Danielle Epstein, Illia Polosukhin, Jacob Devlin, Kenton Lee, Kristina Toutanova, Llion Jones, Matthew Kelcey, Ming-Wei Chang, Andrew M. Dai, Jakob Uszkoreit, Quoc Le, and Slav Petrov. 2019. Natural questions: A benchmark for question answering research. *Transactions of the Association for Computational Linguistics*, 7:452–466.

Pengxiang Li, Yefan Zhou, Dilxat Muhtar, Lu Yin, Shilin Yan, Li Shen, Yi Liang, Soroush Vosoughi, and Shiwei Liu. 2025a. Diffusion language models know the answer before decoding. *arXiv preprint arXiv:2508.19982*.

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

Xiang Li, John Thickstun, Ishaan Gulrajani, Percy S Liang, and Tatsunori B Hashimoto. 2022. Diffusion-lm improves controllable text generation. *Advances in neural information processing systems*, 35:4328–4343.

Hunter Lightman, Vineet Kosaraju, Yuri Burda, Harrison Edwards, Bowen Baker, Teddy Lee, Jan Leike, John Schulman, Ilya Sutskever, and Karl Cobbe. 2024. Let's verify step by step. In *The Twelfth International Conference on Learning Representations*.

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. *Preprint*, arXiv:2506.06295.

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

Xinyin Ma, Runpeng Yu, Gongfan Fang, and Xinchao Wang. 2025. dkv-cache: The cache for diffusion language models. *arXiv preprint arXiv:2505.15781*.

Rabeeh Karimi Mahabadi, Hamish Ivison, Jaesung Tae, James Henderson, Iz Beltagy, Matthew E Peters, and Arman Cohan. 2024. Tess: Text-to-text self-conditioned simplex diffusion. In *Proceedings of the 18th Conference of the European Chapter of the Association for Computational Linguistics (Volume 1: Long Papers)*, pages 2347–2361.

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. *Preprint*, arXiv:2502.09992.

Jingyang Ou, Shen Nie, Kaiwen Xue, Fengqi Zhu, Jiacheng Sun, Zhenguo Li, and Chongxuan Li. 2024. Your absorbing discrete diffusion secretly models the conditional distributions of clean data. *arXiv preprint arXiv:2406.03736*.

Subham Sahoo, Marianne Arriola, Yair Schiff, Aaron Gokaslan, Edgar Marroquin, Justin Chiu, Alexander Rush, and Volodymyr Kuleshov. 2024. Simple and effective masked diffusion language models. *Advances in Neural Information Processing Systems*, 37:130136–130184.

Abigail See, Peter J. Liu, and Christopher D. Manning. 2017. Get to the point: Summarization with pointer-generator networks. In *Proceedings of the 55th Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, pages 1073–1083, Vancouver, Canada. Association for Computational Linguistics.

Yuerong Song, Xiaoran Liu, Ruixiao Li, Zhigeng Liu, Zengfeng Huang, Qipeng Guo, Ziwei He, and Xipeng Qiu. 2025a. Sparse-dllm: Accelerating diffusion llms with dynamic cache eviction. *arXiv preprint arXiv:2508.02558*.

Yuxuan Song, Zheng Zhang, Cheng Luo, Pengyang Gao, Fan Xia, Hao Luo, Zheng Li, Yuehang Yang, Hongli Yu, Xingwei Qu, and 1 others. 2025b. Seed diffusion: A large-scale diffusion language model with high-speed inference. *arXiv preprint arXiv:2508.02193*.

Robin Strudel, Corentin Tallec, Florent Altché, Yilun Du, Yaroslav Ganin, Arthur Mensch, Will Grathwohl, Nikolay Savinov, Sander Dieleman, Laurent Sifre, and 1 others. 2022. Self-conditioned embedding diffusion for text generation. *arXiv preprint arXiv:2211.04236*.

Guanghan Wang, Yair Schiff, Subham Sekhar Sahoo, and Volodymyr Kuleshov. 2025a. Remasking discrete diffusion models with inference-time scaling. *arXiv preprint arXiv:2503.00307*.

Xu Wang, Chenkai Xu, Yijie Jin, Jiachun Jin, Hao Zhang, and Zhijie Deng. 2025b. Diffusion llms can do faster-than-ar inference via discrete diffusion forcing. *Preprint*, arXiv:2508.09192.

Yinjie Wang, Ling Yang, Bowen Li, Ye Tian, Ke Shen, and Mengdi Wang. 2025c. Revolutionizing reinforcement learning framework for diffusion large language models. *arXiv preprint arXiv:2509.06949*.

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. *Preprint*, arXiv:2505.22618.

Zhihui Xie, Jiacheng Ye, Lin Zheng, Jiahui Gao, Jingwei Dong, Zirui Wu, Xueliang Zhao, Shansan Gong, Xin Jiang, Zhenguo Li, and 1 others. 2025. Dreamcoder 7b: An open diffusion language model for code. *arXiv preprint arXiv:2509.01142*.

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*.

Siyan Zhao, Devaansh Gupta, Qinqing Zheng, and Aditya Grover. 2025. d1: Scaling reasoning in diffusion large language models via reinforcement learning. *arXiv preprint arXiv:2504.12216*.

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

Qinglin Zhu, Yizhen Yao, Runcong Zhao, Yanzheng Xiang, Amrutha Saseendran, Chen Jin, Philip Teare, Bin Liang, Yulan He, and Lin Gui. 2025b. Latent refinement decoding: Enhancing diffusion-based language models by refining belief states. *Preprint*, arXiv:2510.11052.

A Detailed Experimental Setup

A.1 Hyperparameters and Implementation

For base models (Dream-Base-7B and LLaDA-Base-8B), we use the standard full-sequence diffusion sampling strategy and follow few-shot evaluation settings for each benchmark: zero-shot for HumanEval, 3-shot for MBPP, 4-shot for MATH500, and 8-shot for GSM8K. Note that the RL model in our experiments refers to LLaDA-1.5-8B, which is post-trained with reinforcement learning on top of LLaDA-Instruct. We report additional results on these three models (Dream-Base-7B, LLaDA-Base-8B, and LLaDA-1.5-8B) in Appendix B. We set the temperature to 0 for all experiments to ensure reproducibility. The unmasking threshold τ is selected from $\{0.6, 0.7, 0.8\}$ depending on the model and dataset. Since instruct models are trained with supervised fine-tuning (SFT), they exhibit more concentrated token distributions; accordingly, we use a slightly higher base threshold for instruct models to avoid premature decoding collapse. All experiments are conducted on 8 NVIDIA A100 80GB GPUs.

Hyperparameter Details for ASRD. For our method, the ATC size m is selected from $\{12, 16, 20\}$ depending on the sequence length and task complexity. The guidance strength α ranges over $[0.05, 0.2]$, with higher values providing stronger guidance at the cost of reduced exploration. The perturbation strength β is selected from $[0.1, 0.4]$ for deployment, the sub-interval of the swept range $[0.1, 0.6]$ (Section 5.4) used in our main results, controlling the probe intensity for pending token verification. During the first $k - 1$ decoding steps where the consistency window is incomplete, we seed the ATC with the single lowest-entropy threshold-passed token at each step to avoid an empty cache.

Reporting protocol. All reported numbers are from a single deterministic run per configuration with temperature 0, so each cell is a point estimate rather than a mean over seeds.

A.2 Datasets

We evaluate on four benchmarks covering mathematical reasoning and code generation:

- **GSM8K** (Cobbe et al., 2021): A dataset of grade-school-level multi-step math word problems requiring arithmetic reasoning.
- **MATH500** (Lightman et al., 2024): A collection of 500 challenging competition-level mathematics problems spanning algebra, geometry, and calculus.
- **MBPP** (Austin et al., 2021b): A benchmark of basic Python function synthesis tasks designed to test fundamental programming capabilities.

Model	Len	Method	HumanEval			MBPP			GSM8K			MATH500		
			Acc	Steps	Speed	Acc	Steps	Speed	Acc	Steps	Speed	Acc	Steps	Speed
Dream-Base-7B	256	baseline	50.6	238.4	1.0×	54.6	239.1	1.0×	75.2	237.5	1.0×	36.9	240.2	1.0×
		Ours	$54.3_{+3.7}$	111.4	2.0×	$57.2_{+2.6}$	73.8	3.1×	$76.3_{+1.1}$	116.2	1.9×	$39.8_{+2.9}$	68.1	3.4×
	512	baseline	54.3	494.2	1.0×	55.8	495.7	1.0×	75.5	493.3	1.0×	37.4	496.1	1.0×
		Ours	$54.9_{+0.6}$	195.1	2.4×	$56.8_{+1.0}$	113.6	3.9×	$77.8_{+2.3}$	138.4	3.3×	$40.2_{+2.8}$	93.9	4.6×
LLaDA-Base-8B	256	baseline	32.3	240.5	1.0×	39.6	238.8	1.0×	69.1	237.2	1.0×	30.0	236.9	1.0×
		Ours	$34.8_{+2.5}$	124.7	1.8×	$41.4_{+1.8}$	115.8	1.9×	$71.3_{+2.2}$	125.3	1.8×	$31.2_{+1.2}$	105.1	2.1×
	512	baseline	32.9	495.1	1.0×	39.8	493.7	1.0×	70.8	496.5	1.0×	30.8	492.8	1.0×
		Ours	$34.8_{+1.9}$	179.2	2.6×	$42.0_{+2.2}$	180.4	2.6×	$71.8_{+1.0}$	158.7	2.9×	$32.2_{+1.4}$	194.5	2.4×
LLaDA-1.5-8B	256	baseline	38.4	239.7	1.0×	38.6	237.9	1.0×	79.2	236.4	1.0×	33.4	238.1	1.0×
		Ours	$42.1_{+4.3}$	72.3	3.2×	$39.8_{+1.2}$	63.5	3.6×	$81.1_{+2.3}$	44.1	5.1×	$33.6_{+0.2}$	46.2	4.8×
	512	baseline	45.1	493.5	1.0×	37.6	496.8	1.0×	82.9	494.2	1.0×	38.6	495.3	1.0×
		Ours	$47.6_{+2.5}$	107.4	4.1×	$38.4_{+0.8}$	106.8	4.1×	$83.9_{+1.0}$	71.6	6.4×	$40.2_{+1.6}$	89.1	5.2×

Table 6: Performance on a wider range of models. Dream-Base-7B and LLaDA-Base-8B serve as representative base models, while LLaDA-1.5-8B represents RL-tuned models. Green subscripts denote accuracy improvements over the baseline.

Model	Method	NQ-Open		CNN/DailyMail			
		EM	Speed	BERTScore	R-1	R-L	Speed
Dream-7B	Baseline	15.8	1.0×	85.5	22.6	15.6	1.0×
	WINO	16.1	1.7×	85.8	23.2	15.8	1.4×
	Saber	16.4	1.4×	85.6	22.9	16.0	1.9×
	ASRD	**17.5**$_{+1.7}$	**2.1**×	**86.0**	**23.6**	**16.5**	**2.3**×
LLaDA-8B	Baseline	12.3	1.0×	82.2	19.8	13.7	1.0×
	WINO	13.4	2.0×	82.3	20.3	13.8	2.2×
	Saber	13.2	1.6×	82.5	19.9	14.0	1.8×
	ASRD	**14.6**$_{+2.3}$	**2.3**×	**82.8**	**20.5**	**14.3**	**2.5**×

Table 7: Open-ended generation results at sequence length 512. NQ-Open is scored by exact match (EM); CNN/DailyMail is scored by BERTScore, ROUGE-1, and ROUGE-L. Speed is wall-clock speedup over the standard decoder for each task.

- **HumanEval** (Chen et al., 2021): A set of 164 manually curated programming problems with unit-test-based evaluation.

B Additional Experimental Results

B.1 Generalization Across Base and RL-Tuned dLLMs

To verify that ASRD captures the intrinsic generative dynamics of diffusion models rather than relying on the specific alignment patterns of instruction-tuned weights, we extend our evaluation to Base models and newer models with RL post-training (Table 6). Remarkably, ASRD delivers consistent performance gains on raw Base models (e.g., +3.7% on Dream-Base-7B), which typically exhibit higher entropy and are more challenging for parallel decoding. Furthermore, on the stronger LLaDA-1.5, our method achieves its most significant improvement (+4.3%), suggesting that ASRD scales effectively with model capability: better underlying representations yield more accurate anchor centroids, further enhancing the efficiency of our Anchor mechanism.

B.2 Open-Ended Generation

The four benchmarks in Section 5.2 all emit short, closed-form answers: numeric outputs for the math benchmarks and single-function code completions for the coding benchmarks. To test whether ASRD transfers to free-form generation, we evaluate two open-ended tasks at sequence length 512: NQ-Open (Kwiatkowski et al., 2019) for open-domain question answering, scored by exact match (EM), and CNN/DailyMail (See et al., 2017) for abstractive summarization, scored by BERTScore, ROUGE-1 (R-1), and ROUGE-L (R-L).

ASRD improves both quality and speed on both tasks for both backbones (Table 7). The largest accuracy gain is on Dream-7B for NQ-Open (15.8 $\rightarrow$ 17.5 EM with a 2.1× speedup), and BERTScore improvements are consistent across both models. These results indicate that the anchor mechanism transfers beyond closed-form benchmarks to settings where multiple valid outputs exist.

B.3 Perturbation Diagnostics and Case Study

Method	HumanEval		GSM8K	
	Error (%) ↓	R (%) ↓	Error (%) ↓	R (%) ↓
Baseline	35.52	59.73	17.15	48.35
WINO	32.22	50.66	15.46	39.25
ASRD	**28.17**	**16.28**	**11.25**	**11.83**

Table 8: Perturbation diagnostics on LLaDA-Instruct-8B (sequence length 512, block size 32). Errors are annotated token-level by Claude-Opus-4.6 against the reference solution, following the same protocol as Table 3. The error rate is the fraction of generated tokens marked as errors; the long-span ratio R is the fraction of those error tokens that belong to a contiguous run of at least 15 error tokens.

We report two diagnostic metrics on the generated outputs of LLaDA-Instruct-8B at sequence length 512 with block size 32, comparing the standard decoder (Baseline), WINO, and ASRD. For each generated sequence, we ask Claude-Opus-4.6 to align it against the reference solution and emit a token-level error mask, following the same judge protocol used for the consistency-window ablation (Table 3).

ASRD reduces both quantities on both benchmarks (Table 8), and the reduction in R is consistently sharper than the reduction in error rate. On HumanEval, R falls from 59.73 to 16.28 while the error rate falls from 35.52 to 28.17; on GSM8K, R falls from 48.35 to 11.83 while the error rate falls from 17.15 to 11.25. This pattern is the empirical signature of Anchor-Perturbed Verification (Section 4.3): by destabilizing mutually reinforcing token clusters, the probe converts long error runs into short scattered errors that are easier for downstream verification to catch.

Case study. In HumanEval problem 92, the WINO decoder commits to `x==sum_xy or y==sum_xy`, where the identifier `sum_xy` is never defined anywhere in the generated function. The expression is internally self-consistent, since both clauses share the same undefined symbol, and therefore evades local verification: any check that compares the two clauses against each other finds them mutually compatible. Under ASRD, the orthogonal probe destabilizes this cluster, the affected tokens are remasked, and the regenerated expression `(x==y+z) or (y==x+z)` closes the gap with no undefined identifiers.

Component	Time (ms)
Baseline: Forward	74.3
Baseline: Post-process	3.0
Baseline: Total	**77.3**
ASRD: Forward	74.1
ASRD: ATC management	1.3
ASRD: Guided injection	3.4
ASRD: Perturbed verification	3.1
ASRD: Post-process	3.0
ASRD: Total	**84.9**

(a) Per-step component breakdown.

Metric	Baseline	ASRD
Average steps	256	75.4
Wall-clock (s)	19.8	6.4
Speedup	1.0×	**3.0**×

(b) End-to-end cost and speedup.

Table 9: Runtime breakdown on LLaDA-Instruct-8B (sequence length 256, A100 80GB). Speedup is wall-clock over the standard decoder. *Guided injection* and *Perturbed verification* refer to Anchor-Guided Generation (Section 4.2) and Anchor-Perturbed Verification (Section 4.3), respectively.

B.4 Per-Step Runtime Breakdown

We profile the per-step cost of ASRD on LLaDA-Instruct-8B with sequence length 256 on a single NVIDIA A100 80GB GPU, averaging over the generated samples. Table 9a breaks the cost down by component, and Table 9b reports the end-to-end consequence in steps and wall-clock time.

ASRD adds 7.6 ms per step on average, a 9.8% increase over the baseline's 77.3 ms. The overhead is concentrated in the embedding-space update: 3.4 ms for guided injection and 3.1 ms for perturbed verification, both linear in the cache size m. Despite this per-step overhead, the average step count drops from 256 to 75.4, so wall-clock time falls from 19.8 s to 6.4 s, giving a 3.0× end-to-end speedup.

B.5 Sensitivity to Block Size and Unmasking Schedule

Section 5.2 reports the main results at the default block size BS=32 and unmasking threshold τ=0.7. To assess whether the gains of ASRD carry over to other decoding configurations, we sweep both axes on MATH500 and MBPP with LLaDA-Instruct-8B.

Block size sweep. Table 10 reports accuracy and wall-clock speedup for BS $\in \{16, 32, 64, 128\}$. On MATH500, ASRD reaches 35.2 accuracy at BS=32 with a 2.9× speedup, exceeding WINO (34.2, 2.0×) and Saber (34.8, 1.8×); the ranking

Dataset	Method	BS=16		BS=32		BS=64		BS=128	
		Acc	Speed	Acc	Speed	Acc	Speed	Acc	Speed
MATH500	Baseline	32.4	1.0×	33.8	1.0×	34.2	1.0×	31.6	1.0×
	WINO	33.6	1.9×	34.2	2.0×	34.0	2.0×	32.2	1.8×
	Saber	34.0	1.7×	34.8	1.8×	34.4	1.8×	32.4	1.7×
	ASRD	**34.6**	**2.8×**	**35.2**	**2.9×**	**34.6**	**2.6×**	**32.6**	**2.8×**
MBPP	Baseline	37.8	1.0×	36.4	1.0×	36.8	1.0×	34.2	1.0×
	WINO	36.6	1.8×	36.2	1.9×	37.4	2.0×	36.4	1.8×
	Saber	37.4	2.0×	37.8	2.1×	37.8	2.1×	36.6	1.9×
	ASRD	**38.2**	**2.5×**	**38.6**	**2.5×**	**38.0**	**2.3×**	**36.8**	**2.6×**

Table 10: Block-size sweep on LLaDA-Instruct-8B. Accuracy in percent; speed is wall-clock speedup over the standard decoder.

holds at every block size we tested. On MBPP, ASRD likewise dominates both baselines across all block sizes. Accuracy degrades for all methods at BS=128, consistent with the observation in Section 5.2 that overly wide blocks weaken anchor supervision.

Schedule sweep. Table 11 sweeps the unmasking schedule. WINO and ASRD are controlled by the confidence threshold τ; Saber instead uses a top-K schedule and is not directly comparable across rows, so we report it in a separate panel. ASRD dominates WINO across the entire τ range on both datasets (for example, on MBPP at τ=0.8, ASRD reaches 38.6 versus WINO's 36.8). On the Saber panel, ASRD at τ=0.7 (35.2 on MATH500, 37.8 on MBPP) is competitive with Saber's best top-K setting. The relative ordering of methods is stable across configurations, indicating that the ATC and orthogonal probe are not tied to a particular schedule.

C Theoretical Analysis

C.1 Exponential Suppression of False Anchors

In this section, we provide a formal justification for the effectiveness of the Anchor Tokens Cache (ATC). We model the decoding process as a signal-plus-noise system to demonstrate that the temporal consistency requirement (Eq. (4.1)) exponentially suppresses the selection of incorrect tokens.

We consider a specific masked position i with ground-truth token x_0^i. Let $w \in V$ be an incorrect candidate token ($w \neq x_0^i$). We posit the following assumptions regarding the model's behavior and the decoding environment:

1. **Semantic Capability:** We assume the model is semantically capable, meaning that in a noise-free (ideal) context, the intrinsic confidence for the ground truth x_0^i is strictly higher than that for the incorrect candidate w. That is, there exists a positive *semantic margin* Δ_{sem}:

$$p^*(x_0^i) - p^*(w) = \Delta_{\text{sem}} > 0, \qquad (1)$$

where $p^*(\cdot)$ denotes the intrinsic, denoised probability of a token.

2. **i.i.d. Noise:** The observed probability $p_\theta(x_0^i = v|x_s)$ at any step s is the sum of the intrinsic prior $p^*(v)$ and a stochastic noise term $\epsilon_{s,v}$. We further assume that the differential noise $\eta_s \triangleq \epsilon_{s,x_0^i} - \epsilon_{s,w}$ is symmetric around 0 and i.i.d. across decoding steps s (this holds, e.g., when $\epsilon_{s,x_0^i} \perp \epsilon_{s,w}$ with each marginally symmetric, or when the noise vector is jointly sign-symmetric).

We now prove Proposition 1.

Proof. **Step 1: Condition for Single-Step Error.** For the incorrect token w to be identified as the prediction $\hat{a}_s^i$ at step s, it must achieve the highest probability among all vocabulary items. A necessary condition for this is that w must specifically surpass the ground truth x_0^i:

$$\hat{a}_s^i = w \implies p_\theta(x_0^i = w|x_s) > p_\theta(x_0^i = x_0^i|x_s). \qquad (2)$$

Using the noise decomposition from Assumption 2 ($p_\theta = p^* + \epsilon$), this inequality becomes:

$$p^*(w) + \epsilon_{s,w} > p^*(x_0^i) + \epsilon_{s,x_0^i}. \qquad (3)$$

We define the *differential noise* at step s as $\eta_s \triangleq \epsilon_{s,x_0^i} - \epsilon_{s,w}$. Rearranging the inequality yields: $\eta_s < p^*(w) - p^*(x_0^i) = -\Delta_{\text{sem}}$. This means that the error occurs only when the differential noise η_s is sufficiently negative to overcome the

Threshold schedule τ (WINO, ASRD)									
Dataset	**Method**	τ=0.6		τ=0.7		τ=0.8		τ=0.9	
		Acc	Speed	Acc	Speed	Acc	Speed	Acc	Speed
MATH500	WINO	33.0	2.9×	33.6	2.3×	34.2	2.0×	34.8	1.5×
	ASRD	**33.2**	**3.3**×	**35.2**	**2.9**×	**35.6**	**2.2**×	**36.0**	**1.7**×
MBPP	WINO	36.2	2.9×	36.2	1.9×	36.8	1.6×	37.4	1.3×
	ASRD	**36.2**	**3.4**×	**37.8**	**3.1**×	**38.6**	**2.5**×	**38.6**	**2.0**×
Top-K schedule (Saber)									
Dataset		K=4		K=3		K=2		K=1	
		Acc	Speed	Acc	Speed	Acc	Speed	Acc	Speed
MATH500	Saber	32.4	2.5×	34.2	1.9×	34.8	1.8×	35.0	1.4×
MBPP	Saber	35.4	2.4×	37.2	2.7×	37.8	2.1×	37.8	1.8×

Table 11: Schedule sweep on LLaDA-Instruct-8B. Top panel: WINO and ASRD as a function of the unmasking threshold τ. Bottom panel: Saber as a function of its top-K schedule; the two panels use different control variables and are not row-aligned.

positive semantic margin Δ_{sem}. Then, substituting the semantic margin Δ_{sem} from Assumption 1, the condition for a ranking reversal in a single step is $-\eta_s > \Delta_{\text{sem}}$.

Step 2: Probability of Ranking Reversal. Let q be the probability that the noise fluctuation is sufficient to overcome the semantic margin $q \triangleq \mathbb{P}(\eta_s > \Delta_{\text{sem}})$. Since the noise distribution is assumed to be symmetric, the probability of the differential noise falling into the left tail is equal to that of the right tail: $q \triangleq \mathbb{P}(\eta_s < -\Delta_{\text{sem}}) = \mathbb{P}(\eta_s > \Delta_{\text{sem}})$. Consequently, $q < 0.5$ (and typically $q \ll 0.5$ for strong models).

Step 3: Temporal Consistency and Exponential Decay. According to Eq. (4.1), the event that position i is identified as an anchor with the incorrect token w requires that $\hat{a}_s^i = w$ for all $s \in \{t-k+1, \dots, t\}$. By Step 1, each event $\{\hat{a}_s^i = w\}$ is contained in $\{\eta_s < -\Delta_{\text{sem}}\}$, so we may bound the joint probability by the joint probability of the η-events, which factorizes under the i.i.d. assumption:

$$\begin{aligned}
&\mathbb{P}\big(\hat{a}_{t-k+1}^i = \cdots = \hat{a}_t^i = w\big) \\
&\quad \le \mathbb{P}(\eta_{t-k+1} < -\Delta_{\text{sem}}, \dots, \eta_t < -\Delta_{\text{sem}}) \\
&\quad = \prod_{j=0}^{k-1} \mathbb{P}(\eta_{t-j} < -\Delta_{\text{sem}}) \\
&\quad = q^k.
\end{aligned} \tag{4}$$

If we rewrite q^k in exponential form $q^k = \exp(-k\ln(1/q))$, let $\lambda = \ln(1/q)$, and note that $q < 1 \implies \lambda > 0$, we then obtain:

$$\mathbb{P}\big(\hat{a}_{t-k+1}^i = \cdots = \hat{a}_t^i = w\big) \le \exp(-\lambda k). \tag{5}$$

This completes the proof, demonstrating that increasing the window size k exponentially suppresses the likelihood of false anchors driven by stochastic noise. □

Remark 1. While this mechanism suppresses false positives, a correct token x_0^i with $p^*(x_0^i) > \tau$ may be excluded from the ATC if a single noise spike ϵ_{s,x_0^i} pushes its top-1 prediction off x_0^i, or its confidence below τ so that it leaves $\mathcal{T}_t$, at any of the k consecutive steps. ASRD prioritizes minimizing this "False Positive" (accepting a wrong anchor) over a "False Negative" (missing a true anchor) due to the asymmetric cost of errors in diffusion decoding: Accepting an incorrect anchor introduces "toxic context", which actively steers the generation of [MASK] tokens toward invalid trajectories via the attention mechanism. This leads to irreversible error propagation. In comparison, a missed valid anchor merely results in a sparser guidance signal. In this case, the model defaults to its standard decoding behavior (baseline), ensuring that ASRD never performs worse than the unguided baseline.

Remark 2. The i.i.d. assumption serves to derive a tractable upper bound that captures the qualitative behavior of the consistency mechanism. In practice, even with temporally correlated noise, the probability of an incorrect token persisting across k steps diminishes significantly, as supported by our empirical results.

C.2 Anchor-Induced Attention Bias

In this section, we provide a calculation-based discussion to illustrate how anchor-guided embedding injection influences attention behavior and down-

stream predictions. We emphasize that the following analysis is intended as a mechanistic intuition, rather than a formal guarantee under the full Transformer dynamics.

Intuition: Anchor-Induced Attention Bias. Consider a self-attention layer, where queries, keys, and values are computed as linear projections of token representations between a mask token at position i and context token at position u: $q_i = W_Q \mathbf{e}_t^i$, $k_u = W_K \mathbf{e}_t^u$, and $v_u = W_V \mathbf{e}_t^u$. Here, for a masked position i, where $\mathbf{e}_t^i = \mathbf{e}_{\text{mask}}$ initially, the anchor-guided generation replaces the uninformative [MASK] embedding $\mathbf{e}_{\text{mask}}$, with the anchor centroid $\bar{c}_t = \frac{1}{|\mathcal{I}_t|} \sum_{j \in \mathcal{I}_t} \mathbf{e}_t^j$ defined in Section 4.2,

$$\mathbf{e}_t'^i = (1 - \gamma_t^i)\, \mathbf{e}_t^i + \gamma_t^i\, \bar{c}_t, \tag{6}$$

where $\mathcal{I}_t$ denotes the Anchor Tokens Cache (ATC) at decoding step t. For notational clarity, we treat the per-position magnitude γ_t^i (defined in Section 4.2) as a fixed scalar throughout this derivation; the entropy-adaptive form only modulates the magnitude of the induced bias per position, not its geometry. Due to the linearity of the query projection, this replacement induces a corresponding shift in the query representation. We have $q_i' = W_Q \mathbf{e}_t'^i = (1 - \gamma_t^i)\, q_i + \gamma_t^i\, q_{\mathcal{I}_t}$, where $q_{\mathcal{I}_t} = W_Q \bar{c}_t$. Thus, the attention logit between the masked position i and an arbitrary position u becomes

$$\begin{aligned} s_{iu}' &= \frac{(q_i')^\top k_u}{\sqrt{d}} \\ &= (1 - \gamma_t^i) \frac{(q_i)^\top k_u}{\sqrt{d}} + \gamma_t^i \frac{(q_{\mathcal{I}_t})^\top k_u}{\sqrt{d}} \\ &= (1 - \gamma_t^i)\, s_{iu} + \gamma_t^i\, s_{\mathcal{I}_t u}, \end{aligned} \tag{7}$$

where s_{iu} and $s_{\mathcal{I}_t u}$ are the attention if we simply replace the mixed token by original mask token or purely aggregate from anchor tokens. Therefore, although we do not assume any specific structure on W_Q or W_K, the injected query q_i' is biased toward the subspace spanned by anchor embeddings. When anchor tokens provide a more reliable semantic signal than unverified tokens, their keys tend to exhibit higher alignment with this shifted query. As a result, the additive change in attention logits is typically larger for anchor tokens than for noisy pending tokens.

More generally, the effect of anchor-guided injection extends beyond anchor tokens themselves to arbitrary context tokens. Consider two context tokens u_1 and u_2 that are indistinguishable under the original mask query, i.e., $\langle k_{u_1}, q_i \rangle \approx \langle k_{u_2}, q_i \rangle$, but the keys' inner product with query vector on anchor tokens' embedding satisfy

$$\langle k_{u_1}, q_{\mathcal{I}_t} \rangle > \langle k_{u_2}, q_{\mathcal{I}_t} \rangle, \tag{8}$$

which implies that k_{u_1} is closer to the collection of anchor tokens. Substituting into the interpolation formula yields

$$s_{iu_1}' > s_{iu_2}'. \tag{9}$$

To isolate the effect of this alignment, consider the case where the remaining context is otherwise comparable, i.e., the original logits s_{iu_1} and s_{iu_2} are of similar magnitude prior to anchor injection. Under this condition, the difference in attention logits after injection is dominated by the difference in $\gamma_t^i\, s_{\mathcal{I}_t u}$, leading to a relative increase in the softmax attention weight for token u_1 compared to u_2.

This comparison illustrates that anchor-guided injection does not merely increase attention to anchor tokens themselves, but more generally amplifies the influence of context tokens whose representations are closer to the anchor-induced direction, while suppressing contributions from less aligned or noisy tokens. In this sense, the mechanism selectively reshapes the effective context seen by the masked position in a direction consistent with anchor semantics.