Renoise Consistency: Unlocking Efficient Self-Correction for Diffusion Large Language Models

Jihoon Lee[1] **Yeongbin Seo**[2,3] **Jaehyung Kim**[1]

[1]Yonsei University [2]University of Toronto [3]Vector Institute

{jihoonlee98,jaehyungk}@yonsei.ac.kr

Abstract

Diffusion Large Language Models (DLMs) are being actively explored as promising alternatives to autoregressive models due to their fast and flexible generation capabilities. To optimize inference in DLMs, recent studies have proposed a variety of decoding algorithms. However, the tendency of these methods to mimic autoregressive generation limits the speed and flexibility of DLMs, while also preventing the methods from fully realizing their potential. Moreover, this rigid approach leads to error propagation, while existing sampling-based correction methods incur prohibitive computational costs. To overcome this, we propose Renoise Consistency, a novel post-hoc approach that enables effective self-correction by maximizing train–inference consistency and can be applied to any decoding algorithm with a flexible computational budget. Furthermore, we introduce Adaptive Block Drafting, which leverages consistent internal representations of DLMs to significantly reduce the overall computational cost. When combined, our proposed method achieves up to a 9.00% performance improvement over the semi-autoregressive baseline, along with a 25.85% reduction in forward steps. Code is available at https://github.com/JiHoonLee9898/ReCon.

1 Introduction

Diffusion Large Language Models (DLMs) offer a distinct advantage over conventional autoregressive LLMs (ARMs) by flexibly determining both the order and the number of tokens to generate at each decoding step [1, 22]. Recent studies have shown that the choice of the decoding rule is critical for inference quality [14, 15, 25], motivating a variety of strategies to optimize performance. For example, semi-autoregressive (semi-AR) decoding has emerged as a standard, operating in a block-wise left-to-right manner to achieve performance competitive with ARMs [22]. However, methods in this line of work face a fundamental limitation: they typically enforce position priors (*e.g.*, fixed left-to-right ordering and high-confidence positions first) to boost performance. This constrains the model to a single biased generation path and suppresses the bidirectional nature of DLMs. Consequently, DLMs become susceptible to error propagation similar to ARMs, where mistakes made at early decoding steps cascade into and degrade all subsequent predictions.

To mitigate this limitation, prompt-based refinement methods [21, 27, 36] have been proposed for ARMs. While attractive for their post-hoc operation on already-generated outputs and agnosticism to the underlying decoding procedure, these methods incur substantial computational overhead by repeatedly regenerating full sequences. Since DLMs should process a fixed-length sequence longer than the intended output at every forward pass, this burden becomes prohibitive for DLMs. As a DLM-specific alternative, recent works inject revocability directly into the inference procedure [6, 12, 29, 35], enabling mid-decoding correction. However, these approaches provide limited control over when correction is triggered and often add non-trivial generation cost. More crucially, because

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

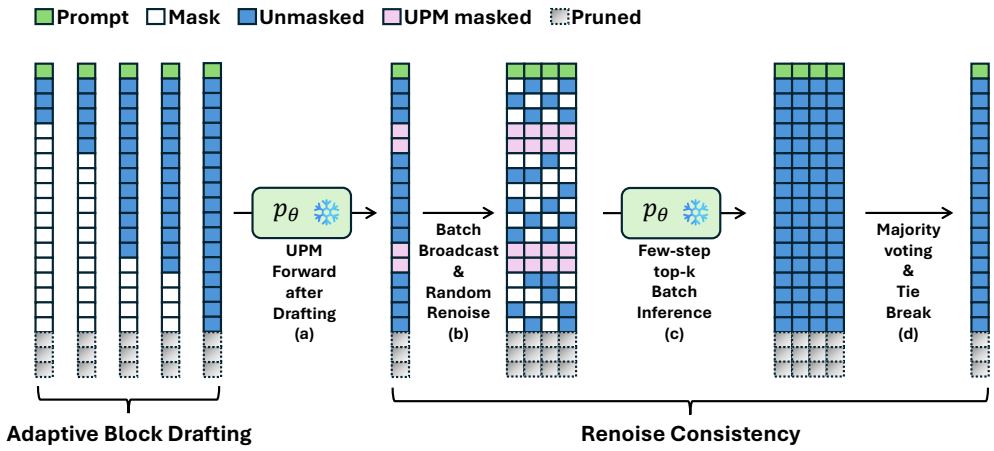


Figure 1: **Overview.** Adaptive Block Drafting (ABD) generates the initial draft and can be replaced by any other decoding method; Renoise Consistency (ReCon) then refines this draft post-hoc. (a) *Uncertainty-aware pre-masking (UPM)*: a single forward pass on the draft identifies low-confidence positions to be re-masked. (b) *Batch-level renoising*: we broadcast the draft into a batch and inject diverse random masking patterns, retaining only $\sim$10% scattered anchor tokens. (c) *Few-step parallel reconstruction*: unlike standard multi-step decoding, we reconstruct all candidates in just a few steps via full-space diffusion without local block constraints, exploiting the DLM's training objective (Section 4.1). (d) *Self-consistency voting*: we aggregate the candidates by majority voting, breaking ties with model confidence. Algorithmic details are provided in Appendix A.

correction is entangled with the decoding procedure itself, these methods cannot be applied in a post-hoc manner and hence have limited compatibility with other decoding methods.

Contribution. In this work, we propose leveraging the full-space diffusion process of DLMs as an *Intrinsic Correction Mechanism*. We empirically find that DLMs can rapidly regenerate high-quality outputs from sparse context anchors derived from an initial draft (see Section 4.1).

Building on this observation, we propose **Renoise Consistency** (ReCon), a novel post-hoc self-correction framework tailored to DLMs. The key idea of ReCon is to explore diverse alternatives of a given draft within only a few steps, and then aggregate them to yield an improved output. Specifically, given a fully unmasked draft, ReCon first identifies low-confidence positions via a single forward pass (*uncertainty-aware pre-masking*) and then constructs multiple re-masked variants by additionally applying random masking, retaining only a few scattered anchor tokens. These re-masked variants are reconstructed in parallel through batch-level broadcasting, exploiting the rapid recovery capability provided by scattered anchors with only a few unmasking steps. Finally, ReCon derives the final response through self-consistency voting [31].

Next, to further reduce end-to-end latency, we propose **Adaptive Block Drafting** (ABD), which accelerates the initial draft generation without sacrificing output quality. To achieve this, we identify a distinct directional pattern in the model's forwarded confidence distribution, which we term *bidirectional inertia*: the tendency of the model's confidence at each forward pass to decay with distance from the known context. We leverage it to dynamically adjust block sizes and prune unnecessary tail positions during decoding. This adaptive scheme allows the model to flexibly determine its preferred output length and block granularity, reducing computational redundancy. By combining ABD with ReCon, our full pipeline achieves superior performance while significantly reducing the total computational budget compared to conventional decoding methods.

We demonstrate the effectiveness of the proposed framework using two state-of-the-art DLMs [22, 37] on four reasoning benchmarks [3, 4, 18, 19]. For example, ReCon is post-hoc compatible with existing decoding methods and improves accuracy by up to 7.17% with only 4.7% additional forward steps. Moreover, when combined with ABD, our full framework delivers superior performance, achieving up to 9.00% accuracy improvement while reducing the overall forward steps by 25.85% compared to semi-AR decoding. Beyond performance gains and step reduction, our framework remains effective even under lightweight configurations, offering a flexible cost-accuracy trade-off: it can be up to 4.64$\times$ faster than Self-Refine [21], while still achieving 11.25% higher accuracy.

2 Related Works

Diffusion large language models. Diffusion models have achieved state-of-the-art performance in image generation [10, 28], and recent advances have extended their impact to discrete domains, most notably natural language generation. Unlike ARMs that generate tokens sequentially, DLMs generate text through an iterative denoising process, predicting the original tokens from a corrupted (*e.g.*, masked or noisy) sequence. Early studies primarily focused on continuous diffusion–based text generation [2, 5, 17], but these approaches exhibited limitations in scalability. Subsequently, masked diffusion emerged as a more promising alternative, as they start to exhibit performance comparable to ARMs of similar scale [22]. Moreover, recent works [8, 33, 37] propose effective methods for converting existing ARMs into DLMs, which indicate that DLM research can substantially benefit from the architectural and representational strengths of ARMs.

Decoding algorithms for DLMs. A defining characteristic of DLMs is their ability to flexibly determine both the number and positions of tokens generated at each decoding (unmasking) step. This flexibility allows decoding for DLMs to adopt the approaches originally studied in ARMs [31, 32] while also enabling a diverse range of new unmasking rules specific to DLMs. Representative approaches include methods that leverage extensive diversity in unmasking positions at high computational cost [15], utilize early decoded outputs across steps [30], exploit inter-step stability differences [14], and re-adjust stride and confidence distributions [25]. However, these methods suffer from notable limitations: some incur excessive inference cost [15], others require joint optimization with training [25, 30], and some demand dataset-specific hyperparameter tuning [14]. Moreover, studies that inject revocability during the decoding process [6, 29] tend to incur additional computations and still suffer from the fundamental limitation of being built on top of AR-mimicking schemes.

3 Preliminary

We adopt the probabilistic formulation of DLMs commonly used in recent studies [1, 22, 25, 26]. For a continuous time step $t \in [0, 1]$, the forward process corrupts a data sequence x_0 into x_t by independently replacing each token with a special mask token M with probability t. Conversely, the reverse process aims to reconstruct the original tokens from the masked sequence, progressing from a fully masked state ($t = 1$) to the clean data ($t = 0$).

Training. Under this formulation, DLMs are trained to predict the original tokens for all masked positions in x_t given the corrupted sequence. Formally, the parametric model $p_\theta(\cdot \mid x_t)$ is trained to replace all masked tokens in x_t with the original tokens in x_0 in a single forward pass. Accordingly, the training objective minimizes the cross-entropy loss over masked positions, normalized by the masking rate:

$$\mathcal{L}(\theta) \triangleq - \mathop{\mathbb{E}}_{t, x_0, x_t} \left[\frac{1}{t} \sum_{i=1}^{L} \mathbb{1}\left[x_t^i = \mathrm{M}\right] \log p_\theta\left(x_0^i \mid x_t\right) \right], \tag{1}$$

where x_0 is a training sequence of length L, $t \sim \mathcal{U}[0, 1]$, and x_t is sampled from the forward process. The term $\mathbb{1}[\cdot]$ denotes the indicator function, ensuring the loss is computed only on masked positions.

Inference and Semi-AR decoding. During inference, generation begins with a sequence consisting of prompt tokens and L mask tokens, where L is a preset target length. While the model can theoretically unmask all tokens in a single forward pass, prior works have shown that such one-shot generation often yields suboptimal quality [15, 22, 25]. To address this, iterative decoding strategies have been proposed to gradually unmask tokens over multiple steps; for instance, top-k strategy [13] only unmasks tokens with the highest-k confidence at each step.

Among them, semi-AR decoding [22] has emerged as a standard approach. It imposes a left-to-right prior by partitioning the target sequence into N disjoint blocks of fixed size ℓ (*i.e.*, $N = L/\ell$). The n-th block is defined as indices $\mathcal{B}_n = \{i \mid (n-1)\ell < i \leq n\ell\}$ and decoding proceeds block-by-block; at the s-th decoding step processing the n-th block, the input x_s consists of: (1) fully unmasked past blocks ($\mathcal{B}_{1:n-1}$), (2) the currently active block ($\mathcal{B}_n$) which is partially masked, and (3) fully masked future blocks ($\mathcal{B}_{n+1:N}$). Then, given x_s, the model predicts the probability distribution over the vocabulary V for the current block; for each position $i \in \mathcal{B}_n$, the model estimates the token $\hat{x}_0^i$

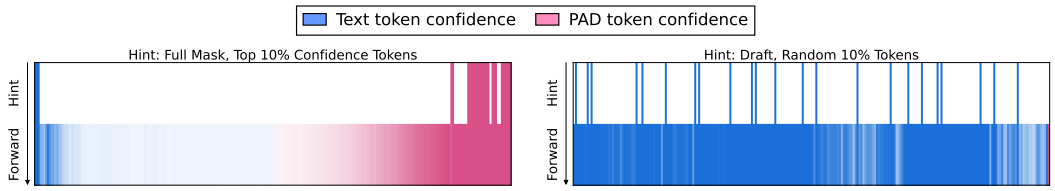


Figure 2: **Effect of context initialization and evenly scattered hints on parallel decoding.** The key to parallel decoding is context initialization. **Left**: Forwarded confidence distribution with high-confidence hints from the early context. Since most of the center of the output window has low confidence, parallel decoding fails. **Right**: Forwarded confidence distribution with draft hints. With evenly distributed hints across the entire output space, the overall forwarded confidence becomes high, allowing parallel decoding to succeed in only a few steps.

and its associated confidence conf_i^s:

$$\hat{x}_0^i = \arg\max_{v \in V} p_\theta(x_0^i = v \mid x_s), \quad i \in \mathcal{B}_n, \tag{2}$$

where $\mathrm{conf}_i^s := p_\theta(x_0^i = \hat{x}_0^i \mid x_s)$. Based on these confidence scores, a subset of tokens in $\mathcal{B}_n$ (*e.g.*, top-k highest confidence tokens [13]) is unmasked to update the state to x_{s+1}. This process repeats until $\mathcal{B}_n$ is fully generated, after which decoding moves to the next block.

While semi-AR decoding is widely adopted [6, 14, 15, 22, 34], this approach incurs heavy computation by processing the full length-L sequence at every step and lacks flexibility due to fixed block sizes. Although these limitations have been prevalent in prior work, we show in Table 5a that they can be alleviated by leveraging the hidden mechanism of DLMs identified in Section 4.3.

4 Method: Efficient Self-Correction and Drafting in DLMs

4.1 Motivation: Few-step Reconstruction in DLMs

The training objective of DLMs (Eq. 1) implies that if the unmasked tokens (*i.e.*, anchor tokens) are uniformly scattered and drawn from the training distribution, then reconstructing the remaining masked tokens in a single forward pass is precisely what the model has been trained to do. However, current left-to-right inference paradigms (e.g., semi-AR) construct contexts concentrated on one side, deviating from this training distribution and thus requiring heavy refinement steps (e.g., 128) to compensate. Accordingly, by constructing scattered anchor contexts at inference time, generation can be brought closer to the training setting, reducing reliance on costly multi-step refinement.

To empirically validate this implication, we conduct a controlled experiment using `LLaDA 8B Instruct` [22]. Given a fully decoded sequence, we re-mask 90% of its tokens at uniformly random positions and test whether the model can reconstruct the original prediction in a few forward passes. To ensure that the source sequences are close to the model's training distribution while preserving sample quality, we use only outputs that the model originally generated correctly. Surprisingly, the model successfully recovers the original prediction at a high rate even under this aggressive re-masking ratio (Table 1). To investigate whether this recovery property is specific to uniformly scattered contexts, we also apply right-side re-masking (masking the rightmost 90% of tokens) to the same examples. In contrast to uniform re-masking, recovery largely fails under this setup. This contrast confirms that efficient few-step self-correction is feasible when the pattern of anchor tokens resembles the context patterns observed during training, and that resolving train-inference mismatch is therefore key.

Table 1: **Recovery success rate.** Success rate of restoring GSM8K outputs that were initially correct, after aggressively remasking 90% of tokens either uniformly at random or from the right side, followed by denoising with only a few forward steps.

Forward steps	**Uniform Random**	Right Side
1	**64.57%**	26.50%
5	**86.56%**	31.40%
10	**89.21%**	48.28%

A natural follow-up question is whether anchor tokens must come from a fully generated draft, or whether intermediate decoding states could suffice. To answer this, we evaluate several anchor construction strategies based on early decoding states. As shown in Figure 2 and Table 6, all such strategies fail to reconstruct reliably, indicating that self-correction via intermediate DLM states is inherently suboptimal from a robustness perspective.

4.2 Renoise Consistency for Efficient Self-Correction in DLMs

Motivated by the observation in Section 4.1, we propose **Renoise Consistency** (ReCon), a post-hoc self-correction framework that refines an initial draft via two principles: *renoising* the draft to obtain diverse but recoverable candidates, and *aggregating* them through self-consistency voting. Let $\hat{x} = (\hat{x}^1, \ldots, \hat{x}^L)$ denote an initial draft sequence generated by any decoding method (*e.g.*, semi-AR, Eq. 2). ReCon refines $\hat{x}$ through the following three steps, illustrated in Figure 1.

Step 1: Uncertainty-aware pre-masking (UPM). First, we identify and mask potentially incorrect tokens in the draft $\hat{x}$. We leverage the observation that DLMs assign low confidence to erroneous tokens when conditioning on the full context (Appendix D.5). Specifically, we perform a single forward pass on the fully unmasked $\hat{x}$ to compute the probability $p_\theta(x^i = \hat{x}^i \mid \hat{x})$ for each token position $i \in [1, L]$. We then mask the tokens with the lowest confidence scores (bottom-u tokens):

$$\hat{x}^i_{\text{masked}} = \begin{cases} \text{M}, & \text{if } i \in \mathcal{I}_{\text{uncertain}}, \\ \hat{x}^i, & \text{otherwise}, \end{cases} \tag{3}$$

where $\mathcal{I}_{\text{uncertain}}$ is the set of indices corresponding to the bottom-u confidence scores. This step ensures that likely errors are targeted for regeneration.

Step 2: Batch-level stochastic re-noising. To induce diversity, we broadcast the pre-masked draft $\hat{x}_{\text{masked}}$ into a batch of size N. For each sequence in the batch, we apply additional uniformly random masking to cover $P\%$ of the tokens (we fix $P = 90$ in all experiments; for the ablation study on P, see Table 15), ensuring that the indices in $\mathcal{I}_{\text{uncertain}}$ remain masked. Formally, for the n-th sequence in the batch ($n = 1, \ldots, N$), we generate a binary mask $m_n \in \{0, 1\}^L$ such that $\sum_i m_n^i \approx \lfloor (P/100)L \rfloor$ and $m_n^i = 1$ for all $i \in \mathcal{I}_{\text{uncertain}}$. The re-noised state is then set to M where $m_n^i = 1$. This heavy re-masking forces the model to explore diverse reconstruction paths, while the remaining anchor tokens keep candidates grounded to the original draft's semantic content.

Step 3: Few-step reconstruction and selection. We reconstruct the N distinct re-noised sequences in parallel using a highly accelerated diffusion schedule. Unlike standard generation, we employ a Top-k unmasking schedule [13] over a small number of steps (*i.e.*, unmasking a large number of tokens at each step). This aggressive reconstruction is justified by our finding in Section 4.1; when DLMs are conditioned on uniformly scattered anchor tokens, only a few steps are sufficient to reconstruct because the input distribution closely matches the training distribution. Specifically, at each step, we unmask k tokens with the highest confidence, where k is determined by evenly distributing the $P\%$ masked tokens across the entire decoding steps. Finally, we select the optimal response using a consistency-based majority voting rule. We group identical final predictions and choose the representative from the largest group that has the highest model confidence.

Rationale. ReCon is grounded in a principle of *train–inference consistency*. Conventional AR-style decoding methods generate from contexts concentrated on the left, which fundamentally diverges from the uniformly scattered contexts that DLMs are trained on (Eq. 1). To compensate, conventional inference relies on hundreds of generation steps for sufficient output quality (*i.e.*, few unmasked tokens per forward pass). ReCon instead constructs candidates whose context distribution directly matches training: by uniformly re-masking an already-generated draft, ReCon makes each candidate input approximate the scattered-context distribution seen during DLM training, enabling high-quality reconstruction in only a few steps. Table 6 and Appendix D.2 quantitatively and qualitatively demonstrate the effectiveness and robustness of this post-hoc re-masking, respectively; Table 15 shows that a high renoising rate simultaneously maximizes diversity and robustness of generated candidates.

4.3 Adaptive Block Drafting for Efficient Drafting in DLMs

Next, we propose **Adaptive Block Drafting** (ABD), an efficient drafting method that further reduces inference cost. ABD is built on another observation we make in DLMs, which we term *bidirectional inertia*. At any forward pass, the model's confidence decays away from known context — left-to-right for generated tokens, and right-to-left for PAD tokens (Figure 3). Exploiting this pattern, ABD adaptively determines truncation points (*Tail Pruning*) first and block sizes (*Flexible Block Sizing*) at each step, replacing the fixed input length and block size of standard semi-AR decoding.

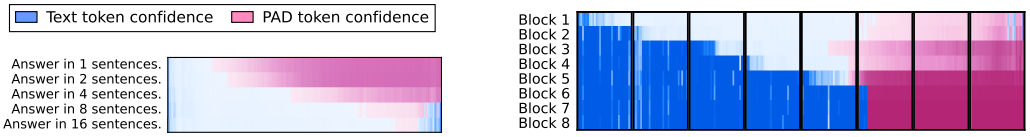


Figure 3: **Hidden sequence-level information in confidence distributions. Left**: Confidence distribution at the *very first forward* step when "Answer in N sentences." is appended to the prompt of a sampled instance from GSM8K. The model's preferred output length is already implied at the first forward step. **Right**: Confidence distribution at the first forward step for each block in semi-AR decoding. The model's preferred block size is implied at each forward step, which differs substantially from using a fixed size (black bold line). Interestingly, the PAD token confidence pattern in the first block closely aligns with that of the final block.

Tail pruning. Given the fully masked input x_1 (*i.e.*, $x_1^i = \mathrm{M}$ for every token position $i \in \{1, \ldots, L\}$), we extract the prediction $\hat{x}_0$ from the model p_θ: $\hat{x}_0^i = \arg\max_{v \in V} p_\theta(x_0^i = v \mid x_1)$. We define the corresponding confidence as $\mathrm{conf}_i^1 := p_\theta(x_0^i = \hat{x}_0^i \mid x_1)$. We then compute the cumulative average confidence over the suffix starting at position L: $\overline{\mathrm{conf}}_i^{\mathrm{tail}} = \frac{1}{L-i+1} \sum_{k=i}^{L} \mathrm{conf}_k^1$. We determine the earliest position in the reverse order (starting from L) at which the cumulative average confidence becomes lower than the threshold α: $i^\star = \max\{\, i \in \{1, \ldots, L\} \mid \overline{\mathrm{conf}}_i^{\mathrm{tail}} < \alpha \,\}$. Finally, we obtain the pruned input sequence $x_1^\star$ by truncating x_1: $x_1^\star = (x_1^1, \ldots, x_1^{i^\star - 1})$. Intuitively, we drop the meaningless tail region where the model is likely to enumerate PAD tokens, and hence reduce the computational cost by shortening input sequence length below the preset length L.

Flexible block sizing. Let $\mathcal{S}_{\mathrm{block}} = \{s_1 < \cdots < s_M\}$ denote the block-selection steps among the total S forward steps. At each $s_m \in \mathcal{S}_{\mathrm{block}}$, given x_{s_m}, we compute $\hat{x}_0^i = \arg\max_{v \in V} p_\theta(x_0^i = v \mid x_{s_m})$ and $\mathrm{conf}_i^{s_m} := p_\theta(x_0^i = \hat{x}_0^i \mid x_{s_m})$. Let b_m be the right boundary of the m-th block, with $b_0 \triangleq 0$. For $i > b_{m-1}$, we compute $\overline{\mathrm{conf}}_i^{\mathrm{block}} = \frac{1}{i - b_{m-1}} \sum_{k=b_{m-1}+1}^{i} \mathrm{conf}_k^{s_m}$, and choose the first position where this average falls below β: $j^\star = \min\{\, i \in \{b_{m-1}+1, \ldots, L\} \mid \overline{\mathrm{conf}}_i^{\mathrm{block}} < \beta \,\}$. We set $b_m \triangleq j^\star$ and $\mathcal{B}_m = \{\, j \mid b_{m-1} + 1 \le j \le b_m \,\}$. After $\mathcal{B}_m$ is filled, the next block is selected at s_{m+1}. In contrast to a conventional use of fixed block size ℓ, this lets the model dynamically allocate longer blocks where it is confident and shorter blocks where uncertainty remains, aligning the decoding granularity with the model's certainty (Figure 3, right).

Block filling. Within each selected block $\mathcal{B}_m$, we adopt the unmasking rule of [14] to unmask multiple confident positions in parallel. At each step s, we obtain the prediction $\hat{x}_0^i = \arg\max_{v \in V} p_\theta(x_0^i = v \mid x_s)$ and its confidence $\mathrm{conf}_i^s := p_\theta(x_0^i = \hat{x}_0^i \mid x_s)$. We then select confident masked positions within the block: $\mathcal{S}_s = \{\, i \in \mathcal{B}_m \mid x_s^i = \mathrm{M},\ \mathrm{conf}_i^s > \tau \,\}$. If $\mathcal{S}_s$ is non-empty, all positions in $\mathcal{S}_s$ are unmasked simultaneously; otherwise, Top-k masked positions in $\mathcal{B}_m$ are unmasked according to conf_i^s. This process repeats until $\mathcal{B}_m$ is fully filled, after which the next block $\mathcal{B}_{m+1}$ is selected.

In summary, ABD aligns the decoding schedule with the model's own predictive distribution rather than predetermined hyperparameters, yielding both efficiency and quality. Table 5a demonstrates that each component contributes substantially, and their combination outperforms semi-AR by a significant margin while using fewer forward passes.

5 Experiments

5.1 Setups

Models and baselines. We adopt widely used instruction-tuned DLMs, `LLaDA 8B Instruct` [22] and `Dream 7B Instruct` [37], and employ various training-free decoding methods as baselines, for fair comparison. We set the output length to 256. For all baseline methods using a fixed block size, we follow the common choice of 32 as the block size. For single-sample generation, we use settings similar to those of KLASS [14]: the generation temperature is set to 0 for LLaDA and 0.2 for Dream, with nucleus (top-p) filtering at $p = 0.95$ for Dream. Full configuration details for all experiments are provided in Appendix B. All experiments were conducted using a single NVIDIA H100 GPU.

Table 2: **Main results: post-hoc compatibility.** Comparison of performance and sampling steps on reasoning benchmarks across various training-free decoding methods. 15 voting candidates for ReCon. See Appendix D.1 for a detailed description of the additional sampling steps.

Method	GSM8K		ARC-C		TruthfulQA		MATH	
	Acc↑	Steps↓	Acc↑	Steps↓	Acc↑	Steps↓	Acc↑	Steps↓
LLaDA								
semi-AR	75.21	128.0	78.41	128.0	51.61	128.0	33.00	128.0
semi-AR + Self-Refine	77.63	384.0	**82.00**	384.0	51.32 (-0.29)	384.0	29.80 (-3.20)	384.0
semi-AR + ReCon (Ours)	**78.32** (+3.11)	134.0	81.31 (+2.90)	134.0	**54.24** (+2.63)	134.0	**33.20** (+0.20)	134.0
confidence fallback	78.54	69.7	75.51	51.8	48.98	52.4	31.60	79.8
confidence fallback + Self-Refine	79.61	157.1	77.13	190.2	47.95 (-1.03)	190.8	29.80 (-1.80)	184.1
confidence fallback + ReCon (Ours)	**80.67** (+2.13)	75.7	**80.12** (+4.61)	57.8	**52.78** (+3.80)	58.4	**33.60** (+2.00)	85.8
KLASS	79.38	115.9	77.99	95.8	49.12	98.3	32.00	137.5
KLASS + Self-Refine	80.74	258.3	79.78	303.9	50.15	324.1	30.40 (-1.60)	300.3
KLASS + ReCon (Ours)	**81.12** (+1.74)	121.9	**82.94** (+4.95)	101.8	**52.78** (+3.66)	104.3	**33.00** (+1.00)	143.5
ABD (Ours)	75.36	68.6	77.56	44.1	47.81	41.6	31.20	82.6
ABD (Ours) + Self-Refine	77.94	168.1	77.05 (-0.51)	166.4	51.17	159.3	28.80 (-2.40)	186.8
ABD (Ours) + ReCon (Ours)	**81.05** (+5.69)	74.6	**81.06** (+3.50)	50.1	**53.36** (+5.55)	47.6	**35.60** (+4.40)	88.6
DREAM								
semi-AR	69.45	128.0	81.31	128.0	51.02	128.0	29.00	128.0
semi-AR + Self-Refine	**77.94**	384.0	80.20 (-1.11)	384.0	52.34	384.0	34.60	384.0
semi-AR + ReCon (Ours)	76.12 (+6.67)	134.0	**87.20** (+5.89)	134.0	**58.19** (+7.17)	134.0	**34.80** (+5.80)	134.0
confidence fallback	70.96	70.4	79.61	48.0	52.05	42.1	32.00	82.5
confidence fallback + Self-Refine	77.03	109.6	80.89	91.0	50.58 (-1.47)	90.1	37.00	122.6
confidence fallback + ReCon (Ours)	**77.79** (+6.83)	76.4	**86.35** (+6.74)	54.0	**58.63** (+6.58)	48.1	**38.20** (+6.20)	88.5
KLASS	76.80	150.0	85.24	83.8	55.26	78.6	35.80	153.3
KLASS + Self-Refine	**79.68**	224.6	85.15 (-0.09)	159.2	56.14	161.5	**38.00**	243.4
KLASS + ReCon (Ours)	77.03 (+0.23)	156.0	**86.60** (+1.36)	89.8	**58.04** (+2.78)	84.6	37.20 (+1.40)	159.3
ABD (Ours)	76.50	76.6	84.22	65.2	59.21	61.6	32.80	88.9
ABD (Ours) + Self-Refine	77.26	104.6	77.73 (-6.49)	90.4	52.63 (-6.58)	88.4	34.40	120.9
ABD (Ours) + ReCon (Ours)	**78.39** (+1.89)	82.6	**87.88** (+3.66)	71.2	**59.50** (+0.29)	67.6	**38.00** (+5.20)	94.9

Datasets and metrics. We evaluate on multiple widely used reasoning benchmarks. GSM8K [4] is a benchmark dataset for evaluating language models' mathematical reasoning and problem-solving abilities, while MATH [18] is a more challenging mathematical reasoning benchmark that includes competition-level problems. ARC-C [3] is the Challenge Set of AI2's ARC dataset, which tests scientific knowledge that is difficult to solve via simple keyword matching or retrieval. TruthfulQA [19] evaluates a language model's tendency to generate false information. For evaluation on comprehensive open-ended generation tasks, refer to Appendix F.2.

5.2 Main Results

As shown in Table 2, ReCon substantially improves the performance of outputs from various decoding baselines with only a minimal additional computational overhead (**only 6 batched steps**, *i.e.*, 4.7% of total steps for semi-AR). ReCon demonstrates consistent and robust gains across a wide range of challenging reasoning tasks, including mathematical, scientific, and factual reasoning. In particular, it consistently improves a widely used semi-AR baseline [22], a general parallel generation method (confidence fallback) [14], and a state-of-the-art parallel generation method (KLASS) [14], with gains of up to 7.17%, 6.83%, and 4.95%, respectively. Moreover, when combined with ABD, our full pipeline (ABD+ReCon) not only outperforms the state-of-the-art parallel decoding method KLASS [14] on TruthfulQA [19] by 4.24%, but also reduces the number of forward steps by 51.64%. Overall, these results indicate the strong post-hoc compatibility and broad applicability of ReCon across diverse decoding frameworks, while emphasizing the efficiency and practical advantages of ABD. Meanwhile, Self-Refine [21], a prompt-based refinement method proposed as a post-hoc refinement approach for ARMs, performs poorly when applied to DLMs. In particular, despite imposing a substantial cost burden by requiring up to 3× as many forward steps as the baseline, it fails to yield consistent performance gains and even degrades performance by up to 6.58%. These results highlight that ReCon is a robust post-hoc refinement method tailored to DLMs.

Table 3: **Main results: comparison with lightweight methods.** Comparison of accuracy (%), sampling steps, latency (sec), and peak VRAM usage (GB) on reasoning benchmarks for lightweight and revocable decoding frameworks using `LLaDA 8B Instruct`. ReCon uses the lightweight 6 candidate configuration. Refer to Appendix B for full configuration. We bold the best and underline the second-best results for each metric among lightweight and revocable decoding frameworks.

Method	GSM8K				ARC-C				TruthfulQA				MATH			
	Acc↑	Steps↓	Lat↓	Vram↓	Acc↑	Steps↓	Lat↓	Vram↓	Acc↑	Steps↓	Lat↓	Vram↓	Acc↑	Steps↓	Lat↓	Vram↓
semi-AR	75.2	128.0	4.8	16.5	78.4	128.0	4.9	20.0	51.6	128.0	4.8	20.2	33.0	128.0	4.9	25.9
KLASS	<u>79.4</u>	115.9	3.1	31.1	<u>78.1</u> (-0.3)	95.8	2.6	36.6	<u>49.1</u> (-2.5)	98.3	2.7	27.0	32.0 (-1.0)	137.5	3.7	41.4
WINO	76.4	**55.3**	**2.0**	<u>28.3</u>	58.0 (-20.4)	<u>59.1</u>	2.7	39.6	46.1 (-5.5)	**41.3**	**1.5**	<u>21.8</u>	<u>33.6</u>	**73.7**	**2.6**	34.6
Saber	73.3 (-1.9)	91.3	<u>2.3</u>	**17.5**	63.4 (-15.0)	92.9	**2.4**	**18.7**	41.2 (-10.4)	92.0	2.4	**16.5**	33.2	105.4	<u>2.8</u>	**19.4**
ABD + ReCon	**80.1** (+4.9)	<u>74.6</u>	3.2	28.9	**81.5** (+3.1)	**50.1**	<u>2.5</u>	<u>26.8</u>	**52.1** (+0.5)	<u>47.6</u>	<u>2.3</u>	24.3	**35.2** (+2.2)	<u>88.6</u>	3.8	<u>31.4</u>

Scaling analysis. Table 4 examines how the proposed framework (Ours: ABD + ReCon) scales with the number of voting candidates N, and contrasts it with Self-Refine as a representative prompt-based post-hoc baseline on `Dream 7B Instruct`. Our method scales gracefully: across $N \in \{3, 6, 9\}$, accuracy improves monotonically while latency remains below the baseline. The only resource that scales meaningfully with N is peak VRAM, which we treat as a controllable trade-off rather than a hard cost: $N = 3$ nearly matches baseline VRAM while still improving accuracy more than +3.0%, and larger N can be selected when memory budget allows. Self-Refine, in contrast, fails on DLMs; its sequential text-level refinement requires 3× forward steps and 3–4× latency of the baseline, yet yields marginal or negative accuracy changes, suggesting that prompt-based refinement for ARMs may have limitations when applied to DLMs.

Table 4: **Effect of candidate count.** Impact of the number of voting candidates N on performance and computing cost.

Method	TruthfulQA				MATH			
	Acc↑	Lat↓	Steps↓	Vram↓	Acc↑	Lat↓	Steps↓	Vram↓
semi-AR	51.0	2.7	128.0	17.4	29.0	2.8	128.0	20.8
+ Self-Refine	49.7 (-1.3)	9.3	384.0	19.2	33.2	10.5	384.0	23.7
Ours ($N = 3$)	54.1 (+3.1)	1.6	68.4	18.6	32.6 (+3.6)	2.3	96.2	26.6
Ours ($N = 6$)	59.4 (+8.4)	1.8	68.4	24.8	33.8 (+4.8)	2.5	96.2	34.8
Ours ($N = 9$)	**61.0** (+10.0)	2.0	68.4	29.5	**37.2** (+8.2)	2.8	96.2	43.5

Next, to further demonstrate the advantage of ABD + ReCon as decoding frameworks for DLMs, we additionally compare it with recent state-of-the-art lightweight and revocable decoding frameworks, WINO [12] and Saber [6], both of which are based on a draft-and-verify mechanism during intermediate decoding steps. As shown in Table 3, it substantially outperforms these methods, while incurring comparable computational cost, showing that its gains do not simply come from additional computation. Even with only 6 voting candidates, ReCon consistently outperforms WINO across all datasets, with gains of up to 23.5% on ARC-C, while using less VRAM than WINO on ARC-C, and MATH. It also substantially improves over Saber, especially on ARC-C (+18.1%) and TruthfulQA (+10.9%), with far fewer decoding steps (*e.g.*, -42.8 and -44.4 steps, respectively).

5.3 Ablation Study

Submodules of ABD and ReCon. Here, we conduct experiments to ablate the submodules that constitute ABD and ReCon. First, Table 5a presents the effectiveness of the two key components of ABD. Flexible Block Sizing and Tail Pruning, when applied individually, significantly improve the performance of the confidence fallback [14] baseline. Moreover, when both components are applied together, they yield substantial gains (+5.04% on ARC-C and +9.79% on TruthfulQA). This demonstrates that ABD can overcompensate for the performance trade-off made by the confidence fallback method to accelerate decoding, by introducing only a minimal number of additional optimization steps. Notably, ABD achieves substantially higher performance than the semi-AR baseline (+2.91% on ARC-C and +8.19% on TruthfulQA), while using about 2× fewer forward passes. Second, Table 5b illustrates the impact of UPM in ReCon. With only a single additional forward pass on a fully unmasked sample, DLMs tend to assign extremely low confidence to suspicious generations (see Appendix D.5). UPM leverages this property by excluding such positions from the hint context, yielding up to a 2.80% gain over random hint construction on MATH.

Table 5: **Ablation studies of ReCon.** (a) Effectiveness of ABD. F and T indicate whether *Flexible Block Sizing* and *Tail Pruning* are enabled. (b) Effectiveness of UPM renoising. UPM stabilizes ReCon's renoising patterns using only a single forward pass. Refer to Appendices C–D for further analysis.

(a) Effectiveness of ABD

Method	Cfg		ARC-C		TruthfulQA	
	F	T	Acc ↑	Steps ↓	Acc ↑	Steps ↓
semi-AR	–	–	81.31	128.0	51.02	128.0
confidence fallback	–	–	79.18	47.1	49.42	40.6
ABD (F)	✓	–	82.42	68.7	53.95	69.8
ABD (T)	–	✓	83.70	51.9	54.97	48.5
ABD (F,T)	✓	✓	**84.22**	65.2	**59.21**	61.6

(b) Effectiveness of UPM

Method	vote	GSM8K	ARC-C	TruthfulQA	MATH
LLaDA					
ABD	–	75.36	77.56	47.81	31.20
ABD + Random Renoise	✓	80.36	80.38	52.78	34.20
ABD + UPM Renoise	✓	**81.05**	**81.06**	**53.36**	**35.60**
DREAM					
ABD	–	76.50	84.22	59.21	32.80
ABD + Random Renoise	✓	**78.39**	**88.05**	58.92	35.20
ABD + UPM Renoise	✓	**78.39**	87.88	**59.50**	**38.00**

Table 6: **Effectiveness of ReCon's batch anchor initialization strategy.** (a) ReCon enables voting over high-quality samples by injecting diverse post-hoc uniform renoising patterns into a draft. For other methods, the top 10% is selected based on confidence scores. (b) majority voting consistently improves accuracy over the draft output, while cases where a correct draft becomes wrong are rare. Evaluated on `LLaDA 8B Instruct`.

(a) Effectiveness of anchor initialization strategy

Method	vote	GSM8K	ARC-C	TruthfulQA	MATH
1-step → top 10% hint	–	16.91	3.84	1.46	12.20
1-step → rand. 10% hint	✓	46.47	66.72	41.81	22.60
ABD → top 10% hint	–	58.00	76.11	48.54	23.60
ABD → ReCon	✓	**80.74**	**80.63**	**51.61**	**35.80**

(b) Analysis of voting candidates robustness

Metric	GSM8K	ARC-C	TruthfulQA	MATH
Correct among N	86.56	80.51	74.42	80.34
Correct → Wrong	0.38	1.54	2.78	1.00
Wrong → Correct	3.18	10.41	8.78	5.00
Draft Accuracy	77.94	71.76	45.61	31.80
Vote Accuracy	**80.74**	**80.63**	**51.61**	**35.80**

Robustness of anchor context initialization in ReCon. Next, we conduct experiments to validate ReCon by contrasting it with alternative approaches. Table 6a shows that *1-step → top 10% hint*, which anchors the most confident tokens from a single forward pass on fully masked tokens, produces hints clustered around the left or right context rather than being uniformly scattered, and thus causes generation collapse under severe train–inference distribution mismatch (Section 4.1). Meanwhile, randomly anchoring tokens from a single forward pass (*1-step → rand. 10% hint*) yields scattered anchors, but they remain semantically unreliable because they are derived from zero-context one-step predictions; intuitive qualitative examples are provided in Appendix D.2. These results show that effective anchors must be both scattered and semantically reliable, while *ABD → top 10% hint* further shows that anchor diversity is also important, since fixed highest-confidence anchors cannot diversify the candidates, reducing consistency voting to a meaningless procedure. Unlike these failure cases, ReCon's post-hoc anchor initialization avoids all of these collapse modes while achieving high accuracy (*ABD → ReCon*), by satisfying all of these conditions for robust self-correction.

Table 6b further explains how ReCon achieves reliable gains under majority voting. The high (*Correct among* N) rate represents the dataset-level average fraction of ReCon-generated candidates that produce the correct prediction for each question. Its consistently high value indicates that most candidate sets are already biased toward the correct answer, making majority voting robust. This is further supported by the voting outcomes: the (*Wrong → Correct*) rate, which measures successful self-correction from the wrong draft, is substantially higher than the (*Correct → Wrong*) rate (*i.e.*, decreasing the quality compared to draft), which captures cases where voting overturns an initially correct draft. Consequently, the resulting accuracy gain reaches up to 8.87% on ARC-C, highlighting ReCon's ability to correct erroneous drafts robustly.

Further analysis of anchor context. Given a draft candidate, ReCon randomly re-noises and reconstructs most of the draft (*e.g.*, 90%). A natural question arises: Which type of tokens in the anchor context particularly lead to correct or wrong reconstruction? To answer this question, we analyze the relationship between reconstruction success and token types in anchor contexts. Interestingly, *induced tokens* (intermediate computation results essential to the final answer) are particularly associated with reconstruction success or failure, whereas the other token types are not. Specifically, we categorize retained anchor tokens by type (Appendix D.7). For each type, we compute the average number of anchor tokens in correct and wrong ReCon reconstruction candidates before voting, and then subtract the latter average from the former, separately for initially wrong and correct drafts. Positive values indicate an association with correct reconstruction, while negative values indicate an association with wrong reconstruction.

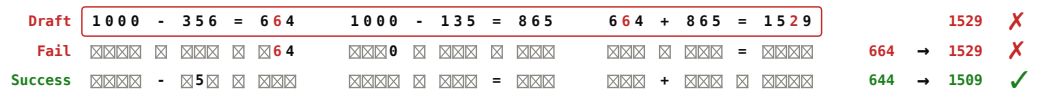


Figure 4: **An illustrative ReCon example from GSM8K test datapoint #620.** Errors in the draft primarily arise from *induced tokens*. Eventual hallucination is introduced by these intermediate mistakes (*e.g.*, 664). If these errors remain in the 10% anchor context, the reconstruction becomes conditioned on them and reproduces the hallucinations. In contrast, when these errors are probabilistically masked by 90% heavy random masking, the erroneous information is no longer available to be conditioned, creating an opportunity for correction.

We collect the outputs of ReCon on GSM8K with `LLaDA` and `Dream`, and select those in which the prediction flipped after voting, *i.e.*, cases where the anchor affected the flip. We then use `Gemini 3.5 Flash` [9] to classify every token in the drafts from this population into five token types (Appendix D.7). As shown in Table 7, retaining more *induced tokens* as anchors can help maintain a correct draft, while it can also maintain a wrong one. Conversely, removing *induced tokens* may compromise a correct draft, but it gives direct opportunities to revise the incorrect draft. Here, removing most of the *induced tokens* by ReCon's aggressive 90% remasking yields asymmetric gains. Since wrong reconstruction can vary widely, corruption of correct drafts tends to scatter votes across different answers. In contrast, successful reconstruction from wrong drafts converge toward a single correct answer, concentrating votes on that answer. This asymmetry allows majority voting to mitigate the losses from draft corruption while leveraging gains from correction (Table 6b). In other words, ReCon can be interpreted as a simple and efficient statistical filter to detect *induced tokens* and correct them: for initially incorrect drafts, heavy random remasking probabilistically masks most of these tokens and enables their correction.

Table 7: **Average anchor token counts gap.** Difference in retained anchor token counts, averaged per prompt.

Token type	Initially Wrong	Initially Correct
Induced	$\mathbf{-0.342}$	$\mathbf{+0.491}$
Formatting	$+0.169$	$+0.189$
Reasoning-structure	-0.007	-0.210
Input numerical	$+0.110$	-0.065
Local lexical	$+0.018$	$+0.121$

Figure 4 illustrates the self-correction of ReCon on mathematical reasoning. GSM8K test datapoint #620 requires the model to perform step-by-step reasoning: separately computing $1000 - 356$ and $1000 - 135$, and then aggregating the intermediate results. However, as illustrated in the draft, if the model mistakenly computes $1000 - 356$ as 664, this error propagates through subsequent reasoning steps, ultimately leading to an incorrect final answer, 1529. Successful self-correction therefore requires the model to revisit and correct the erroneous *induced tokens*, 664. If this value remains fixed in the anchor context, the model fails by attempting correction conditioned on the error. Thus, the key to self-correction is to remask potentially erroneous *induced tokens* so that they no longer serve as conditioning context during reconstruction. In practice, identifying and remasking such tokens across diverse domains remains challenging. ReCon offers a simple solution to this challenge: applying heavy random remasking to the draft in a post-hoc manner, thereby constructing a pool of samples in which successful reconstructions form a majority. As demonstrated by comprehensive experiments, ReCon exhibits broad compatibility and effectiveness across a wide range of reasoning tasks.

6 Conclusion

We propose Renoise Consistency (ReCon) and Adaptive Block Drafting (ABD), training-free and post-hoc decoding methods for Diffusion Large Language Models (DLMs). We reveal the conditions for successful full-space diffusion, which can be achieved by resolving the train-inference mismatch in DLMs during the self-correction stage. We also uncover bidirectional inertia in DLMs, a phenomenon that can be leveraged for efficient and robust drafting. Based on these findings, ReCon reduces both the cost of achieving sample diversity and the cost of reconstruction, thereby unlocking efficient self-correction for DLMs. In parallel, ABD aligns drafting with model-preferred lengths and chunking to improve efficiency and performance. When combined, our methods significantly reduce latency and forward steps while robustly improving reasoning performance across comprehensive model architectures and datasets. Finally, we discuss the potential limitations of our work in Appendix I.

Acknowledgements

This research was supported in part by Institute for Information & communications Technology Planning & Evaluation (IITP) grant funded by the Korea government (MSIT) (No. RS-2020-II201361, Artificial Intelligence Graduate School Program (Yonsei University); No. RS-2026-25522672, Development of Unified Reasoning Technology Mimicking Human Cognition for Hierarchical Understanding and Unbounded Problem Solving; No. RS-2022-00143911, AI Excellence Global Innovative Leader Education Program)

References

[1] Marianne Arriola, Aaron Gokaslan, Justin T. Chiu, Zhihan Yang, Zhixuan Qi, Jiaqi Han, Subham Sekhar Sahoo, and Volodymyr Kuleshov. Block diffusion: Interpolating between autoregressive and diffusion language models, 2025. URL https://arxiv.org/abs/2503.09573.

[2] Jacob Austin, Daniel D. Johnson, Jonathan Ho, Daniel Tarlow, and Rianne van den Berg. Structured denoising diffusion models in discrete state-spaces, 2023. URL https://arxiv.org/abs/2107.03006.

[3] Peter Clark, Isaac Cowhey, Oren Etzioni, Tushar Khot, Ashish Sabharwal, Carissa Schoenick, and Oyvind Tafjord. Think you have solved question answering? try arc, the ai2 reasoning challenge, 2018. URL https://arxiv.org/abs/1803.05457.

[4] Karl Cobbe, Vineet Kosaraju, Mohammad Bavarian, Mark Chen, Heewoo Jun, Lukasz Kaiser, Matthias Plappert, Jerry Tworek, Jacob Hilton, Reiichiro Nakano, Christopher Hesse, and John Schulman. Training verifiers to solve math word problems, 2021. URL https://arxiv.org/abs/2110.14168.

[5] Sander Dieleman, Laurent Sartran, Arman Roshannai, Nikolay Savinov, Yaroslav Ganin, Pierre H. Richemond, Arnaud Doucet, Robin Strudel, Chris Dyer, Conor Durkan, Curtis Hawthorne, Rémi Leblond, Will Grathwohl, and Jonas Adler. Continuous diffusion for categorical data, 2022. URL https://arxiv.org/abs/2211.15089.

[6] 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. Saber: An efficient sampling with adaptive acceleration and backtracking enhanced remasking for diffusion language model, 2025. URL https://arxiv.org/abs/2510.18165.

[7] Yann Dubois, Balázs Galambosi, Percy Liang, and Tatsunori B. Hashimoto. Length-controlled alpacaeval: A simple way to debias automatic evaluators, 2025. URL https://arxiv.org/abs/2404.04475.

[8] 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. Scaling diffusion language models via adaptation from autoregressive models, 2025. URL https://arxiv.org/abs/2410.17891.

[9] Google DeepMind. Gemini 3.5 Flash, 2026. URL https://deepmind.google/models/model-cards/gemini-3-5-flash/.

[10] Jonathan Ho, Ajay Jain, and Pieter Abbeel. Denoising diffusion probabilistic models, 2020. URL https://arxiv.org/abs/2006.11239.

[11] Ari Holtzman, Jan Buys, Li Du, Maxwell Forbes, and Yejin Choi. The curious case of neural text degeneration, 2020. URL https://arxiv.org/abs/1904.09751.

[12] Feng Hong, Geng Yu, Yushi Ye, Haicheng Huang, Huangjie Zheng, Ya Zhang, Yanfeng Wang, and Jiangchao Yao. Wide-in, narrow-out: Revokable decoding for efficient and effective dllms, 2025. URL https://arxiv.org/abs/2507.18578.

[13] Jaeyeon Kim, Kulin Shah, Vasilis Kontonis, Sham Kakade, and Sitan Chen. Train for the worst, plan for the best: Understanding token ordering in masked diffusions, 2025. URL https://arxiv.org/abs/2502.06768.

[14] Seo Hyun Kim, Sunwoo Hong, Hojung Jung, Youngrok Park, and Se-Young Yun. Klass: Kl-guided fast inference in masked diffusion models, 2025. URL https://arxiv.org/abs/2511.05664.

[15] Jihoon Lee, Hoyeon Moon, Kevin Zhai, Arun Kumar Chithanar, Anit Kumar Sahu, Soummya Kar, Chul Lee, Souradip Chakraborty, and Amrit Singh Bedi. Test-time scaling in diffusion llms via hidden semi-autoregressive experts, 2025. URL https://arxiv.org/abs/2510.05040.

[16] Tianle Li, Wei-Lin Chiang, Evan Frick, Lisa Dunlap, Tianhao Wu, Banghua Zhu, Joseph E. Gonzalez, and Ion Stoica. From crowdsourced data to high-quality benchmarks: Arena-hard and benchbuilder pipeline, 2024. URL https://arxiv.org/abs/2406.11939.

[17] Xiang Lisa Li, John Thickstun, Ishaan Gulrajani, Percy Liang, and Tatsunori B. Hashimoto. Diffusion-lm improves controllable text generation, 2022. URL https://arxiv.org/abs/2205.14217.

[18] Hunter Lightman, Vineet Kosaraju, Yura Burda, Harri Edwards, Bowen Baker, Teddy Lee, Jan Leike, John Schulman, Ilya Sutskever, and Karl Cobbe. Let's verify step by step, 2023. URL https://arxiv.org/abs/2305.20050.

[19] Stephanie Lin, Jacob Hilton, and Owain Evans. Truthfulqa: Measuring how models mimic human falsehoods, 2022. URL https://arxiv.org/abs/2109.07958.

[20] Yang Liu, Dan Iter, Yichong Xu, Shuohang Wang, Ruochen Xu, and Chenguang Zhu. G-eval: Nlg evaluation using gpt-4 with better human alignment, 2023. URL https://arxiv.org/abs/2303.16634.

[21] Aman Madaan, Niket Tandon, Prakhar Gupta, Skyler Hallinan, Luyu Gao, Sarah Wiegreffe, Uri Alon, Nouha Dziri, Shrimai Prabhumoye, Yiming Yang, Shashank Gupta, Bodhisattwa Prasad Majumder, Katherine Hermann, Sean Welleck, Amir Yazdanbakhsh, and Peter Clark. Self-refine: Iterative refinement with self-feedback, 2023. URL https://arxiv.org/abs/2303.17651.

[22] Shen Nie, Fengqi Zhu, Zebin You, Xiaolu Zhang, Jingyang Ou, Jun Hu, Jun Zhou, Yankai Lin, Ji-Rong Wen, and Chongxuan Li. Large language diffusion models, 2025. URL https://arxiv.org/abs/2502.09992.

[23] OpenAI, Josh Achiam, Steven Adler, Sandhini Agarwal, Lama Ahmad, Ilge Akkaya, Florencia Leoni Aleman, Diogo Almeida, Janko Altenschmidt, Sam Altman, Shyamal Anadkat, Red Avila, Igor Babuschkin, Suchir Balaji, Valerie Balcom, Paul Baltescu, Haiming Bao, Mohammad Bavarian, Jeff Belgum, Irwan Bello, Jake Berdine, Gabriel Bernadett-Shapiro, Christopher Berner, Lenny Bogdonoff, Oleg Boiko, Madelaine Boyd, Anna-Luisa Brakman, Greg Brockman, Tim Brooks, Miles Brundage, Kevin Button, Trevor Cai, Rosie Campbell, Andrew Cann, Brittany Carey, Chelsea Carlson, Rory Carmichael, Brooke Chan, Che Chang, Fotis Chantzis, Derek Chen, Sully Chen, Ruby Chen, Jason Chen, Mark Chen, Ben Chess, Chester Cho, Casey Chu, Hyung Won Chung, Dave Cummings, Jeremiah Currier, Yunxing Dai, Cory Decareaux, Thomas Degry, Noah Deutsch, Damien Deville, Arka Dhar, David Dohan, Steve Dowling, Sheila Dunning, Adrien Ecoffet, Atty Eleti, Tyna Eloundou, David Farhi, Liam Fedus, Niko Felix, Simón Posada Fishman, Juston Forte, Isabella Fulford, Leo Gao, Elie Georges, Christian Gibson, Vik Goel, Tarun Gogineni, Gabriel Goh, Rapha Gontijo-Lopes, Jonathan Gordon, Morgan Grafstein, Scott Gray, Ryan Greene, Joshua Gross, Shixiang Shane Gu, Yufei Guo, Chris Hallacy, Jesse Han, Jeff Harris, Yuchen He, Mike Heaton, Johannes Heidecke, Chris Hesse, Alan Hickey, Wade Hickey, Peter Hoeschele, Brandon Houghton, Kenny Hsu, Shengli Hu, Xin Hu, Joost Huizinga, Shantanu Jain, Shawn Jain, Joanne Jang, Angela Jiang, Roger Jiang, Haozhun Jin, Denny Jin, Shino Jomoto, Billie Jonn, Heewoo Jun, Tomer Kaftan, Łukasz Kaiser, Ali Kamali, Ingmar Kanitscheider, Nitish Shirish Keskar, Tabarak Khan, Logan Kilpatrick, Jong Wook Kim, Christina Kim, Yongjik Kim, Jan Hendrik Kirchner, Jamie Kiros, Matt Knight, Daniel Kokotajlo, Łukasz Kondraciuk, Andrew Kondrich, Aris Konstantinidis, Kyle Kosic, Gretchen Krueger, Vishal Kuo, Michael Lampe, Ikai Lan, Teddy Lee, Jan Leike,

Jade Leung, Daniel Levy, Chak Ming Li, Rachel Lim, Molly Lin, Stephanie Lin, Mateusz Litwin, Theresa Lopez, Ryan Lowe, Patricia Lue, Anna Makanju, Kim Malfacini, Sam Manning, Todor Markov, Yaniv Markovski, Bianca Martin, Katie Mayer, Andrew Mayne, Bob McGrew, Scott Mayer McKinney, Christine McLeavey, Paul McMillan, Jake McNeil, David Medina, Aalok Mehta, Jacob Menick, Luke Metz, Andrey Mishchenko, Pamela Mishkin, Vinnie Monaco, Evan Morikawa, Daniel Mossing, Tong Mu, Mira Murati, Oleg Murk, David Mély, Ashvin Nair, Reiichiro Nakano, Rajeev Nayak, Arvind Neelakantan, Richard Ngo, Hyeonwoo Noh, Long Ouyang, Cullen O'Keefe, Jakub Pachocki, Alex Paino, Joe Palermo, Ashley Pantuliano, Giambattista Parascandolo, Joel Parish, Emy Parparita, Alex Passos, Mikhail Pavlov, Andrew Peng, Adam Perelman, Filipe de Avila Belbute Peres, Michael Petrov, Henrique Ponde de Oliveira Pinto, Michael, Pokorny, Michelle Pokrass, Vitchyr H. Pong, Tolly Powell, Alethea Power, Boris Power, Elizabeth Proehl, Raul Puri, Alec Radford, Jack Rae, Aditya Ramesh, Cameron Raymond, Francis Real, Kendra Rimbach, Carl Ross, Bob Rotsted, Henri Roussez, Nick Ryder, Mario Saltarelli, Ted Sanders, Shibani Santurkar, Girish Sastry, Heather Schmidt, David Schnurr, John Schulman, Daniel Selsam, Kyla Sheppard, Toki Sherbakov, Jessica Shieh, Sarah Shoker, Pranav Shyam, Szymon Sidor, Eric Sigler, Maddie Simens, Jordan Sitkin, Katarina Slama, Ian Sohl, Benjamin Sokolowsky, Yang Song, Natalie Staudacher, Felipe Petroski Such, Natalie Summers, Ilya Sutskever, Jie Tang, Nikolas Tezak, Madeleine B. Thompson, Phil Tillet, Amin Tootoonchian, Elizabeth Tseng, Preston Tuggle, Nick Turley, Jerry Tworek, Juan Felipe Cerón Uribe, Andrea Vallone, Arun Vijayvergiya, Chelsea Voss, Carroll Wainwright, Justin Jay Wang, Alvin Wang, Ben Wang, Jonathan Ward, Jason Wei, CJ Weinmann, Akila Welihinda, Peter Welinder, Jiayi Weng, Lilian Weng, Matt Wiethoff, Dave Willner, Clemens Winter, Samuel Wolrich, Hannah Wong, Lauren Workman, Sherwin Wu, Jeff Wu, Michael Wu, Kai Xiao, Tao Xu, Sarah Yoo, Kevin Yu, Qiming Yuan, Wojciech Zaremba, Rowan Zellers, Chong Zhang, Marvin Zhang, Shengjia Zhao, Tianhao Zheng, Juntang Zhuang, William Zhuk, and Barret Zoph. Gpt-4 technical report, 2024. URL https://arxiv.org/abs/2303.08774.

[24] Priyanka Sen, Alham Fikri Aji, and Amir Saffari. Mintaka: A complex, natural, and multilingual dataset for end-to-end question answering, 2022. URL https://arxiv.org/abs/2210.01613.

[25] Yeongbin Seo, Dongha Lee, Jaehyung Kim, and Jinyoung Yeo. Fast and fluent diffusion language models via convolutional decoding and rejective fine-tuning, 2025. URL https://arxiv.org/abs/2509.15188.

[26] Jiaxin Shi, Kehang Han, Zhe Wang, Arnaud Doucet, and Michalis K. Titsias. Simplified and generalized masked diffusion for discrete data, 2025. URL https://arxiv.org/abs/2406.04329.

[27] Noah Shinn, Federico Cassano, Edward Berman, Ashwin Gopinath, Karthik Narasimhan, and Shunyu Yao. Reflexion: Language agents with verbal reinforcement learning, 2023. URL https://arxiv.org/abs/2303.11366.

[28] Jiaming Song, Chenlin Meng, and Stefano Ermon. Denoising diffusion implicit models, 2022. URL https://arxiv.org/abs/2010.02502.

[29] Guanghan Wang, Yair Schiff, Subham Sekhar Sahoo, and Volodymyr Kuleshov. Remasking discrete diffusion models with inference-time scaling, 2025. URL https://arxiv.org/abs/2503.00307.

[30] Wen Wang, Bozhen Fang, Chenchen Jing, Yongliang Shen, Yangyi Shen, Qiuyu Wang, Hao Ouyang, Hao Chen, and Chunhua Shen. Time is a feature: Exploiting temporal dynamics in diffusion language models, 2025. URL https://arxiv.org/abs/2508.09138.

[31] Xuezhi Wang, Jason Wei, Dale Schuurmans, Quoc Le, Ed Chi, Sharan Narang, Aakanksha Chowdhery, and Denny Zhou. Self-consistency improves chain of thought reasoning in language models, 2023. URL https://arxiv.org/abs/2203.11171.

[32] Jason Wei, Xuezhi Wang, Dale Schuurmans, Maarten Bosma, Brian Ichter, Fei Xia, Ed Chi, Quoc Le, and Denny Zhou. Chain-of-thought prompting elicits reasoning in large language models, 2023. URL https://arxiv.org/abs/2201.11903.

[33] Chengyue Wu, Hao Zhang, Shuchen Xue, Shizhe Diao, Yonggan Fu, Zhijian Liu, Pavlo Molchanov, Ping Luo, Song Han, and Enze Xie. Fast-dllm v2: Efficient block-diffusion llm, 2025. URL https://arxiv.org/abs/2509.26328.

[34] Chengyue Wu, Hao Zhang, Shuchen Xue, Zhijian Liu, Shizhe Diao, Ligeng Zhu, Ping Luo, Song Han, and Enze Xie. Fast-dllm: Training-free acceleration of diffusion llm by enabling kv cache and parallel decoding, 2025. URL https://arxiv.org/abs/2505.22618.

[35] Yanzheng Xiang, Lan Wei, Yizhen Yao, Qinglin Zhu, Hanqi Yan, Chen Jin, Philip Alexander Teare, Dandan Zhang, Lin Gui, Amrutha Saseendran, and Yulan He. Stop the flip-flop: Context-preserving verification for fast revocable diffusion decoding, 2026. URL https://arxiv.org/abs/2602.06161.

[36] Shunyu Yao, Dian Yu, Jeffrey Zhao, Izhak Shafran, Thomas L. Griffiths, Yuan Cao, and Karthik Narasimhan. Tree of thoughts: Deliberate problem solving with large language models, 2023. URL https://arxiv.org/abs/2305.10601.

[37] Jiacheng Ye, Zhihui Xie, Lin Zheng, Jiahui Gao, Zirui Wu, Xin Jiang, Zhenguo Li, and Lingpeng Kong. Dream 7b: Diffusion large language models, 2025. URL https://arxiv.org/abs/2508.15487.

[38] Kaiwen Zheng, Yongxin Chen, Hanzi Mao, Ming-Yu Liu, Jun Zhu, and Qinsheng Zhang. Masked diffusion models are secretly time-agnostic masked models and exploit inaccurate categorical sampling, 2025. URL https://arxiv.org/abs/2409.02908.

[39] Lianmin Zheng, Wei-Lin Chiang, Ying Sheng, Siyuan Zhuang, Zhanghao Wu, Yonghao Zhuang, Zi Lin, Zhuohan Li, Dacheng Li, Eric P. Xing, Hao Zhang, Joseph E. Gonzalez, and Ion Stoica. Judging llm-as-a-judge with mt-bench and chatbot arena, 2023. URL https://arxiv.org/abs/2306.05685.

[40] Fengqi Zhu, Rongzhen Wang, Shen Nie, Xiaolu Zhang, Chunwei Wu, Jun Hu, Jun Zhou, Jianfei Chen, Yankai Lin, Ji-Rong Wen, and Chongxuan Li. Llada 1.5: Variance-reduced preference optimization for large language diffusion models, 2025. URL https://arxiv.org/abs/2505.19223.

Appendix

A Algorithms **16**

B Hyperparameter Settings **17**

C Additional Ablation Study on Adaptive Block Drafting **17**

C.1 Additional Analysis of Tail Pruning 17

D Additional Ablation Study on Renoise Consistency **18**

D.1 Clarification on Additional Steps Computation 18

D.2 Analysis of Hint Context Initialization 19

D.3 Performance with Varying Numbers of Candidates and Steps 23

D.4 Effectiveness of Anchor Initialization 24

D.5 Uncertainty-Aware Pre-Masking 25

D.6 Performance with Varying Numbers of bottom-u Tokens of UPM 26

D.7 Examples of Anchor Token Types 26

E Additional Analyses **27**

F Evaluation on Additional Models and Datasets **28**

F.1 Additional Performance Comparison on LLaDA 1.5 28

F.2 Evaluation of Open-Ended Generation 28

G Further Wall-Clock Latency Measurement **29**

H Difficulty of Stochastic Sampling in DLMs **30**

I Limitations **30**

A Algorithms

Algorithm 1 Adaptive Block Drafting

Require: Model p_θ, masked prompt[1], x_1, forward steps T
Ensure: Draft $\hat{X}$

$x_1 \leftarrow \textsc{TailPrune}(x_1)$ ▷ Sec. (4.3)
$m \leftarrow 0$, current block $\mathcal{B}_m \leftarrow$ `None`
for $t = 1$ to T **do**
  $(\hat{x}_0,\ \text{conf}) \sim p_\theta(x_0 \mid x_t)$
  if $\mathcal{B}_m$ is `None` **then**
    $m \leftarrow m + 1$
    $\mathcal{B}_m \leftarrow \textsc{BlockSize}(\hat{x}_0,\ \text{conf})$ ▷ Sec. (4.3)
  end if
  $x_{t+1} \leftarrow \textsc{BlockFill}(\hat{x}_0,\ \text{conf}, \mathcal{B}_m)$ ▷ Sec. (4.3)
  if $\textsc{BlockFilled}(B_m)$ **then**
    $\mathcal{B}_m \leftarrow$ `None`
  end if
end for
$\hat{X} \leftarrow \textsc{Decode}(x_{T+1})$
return $\hat{X}$

Algorithm 1: Adaptive Block Drafting. TailPrune and BlockSize refer to tail pruning and flexible block sizing in Section 4.3, while BlockFill corresponds to block filling in the same section. BlockFilled indicates the transition to the next block selection step.

Algorithm 2 Renoise Consistency

Require: Model p_θ, draft $\hat{X}$, batch B, forward steps F
Ensure: Final prediction $y^\star$

$\hat{X} \leftarrow \text{UPM}(\hat{X})$ ▷ Eq. (3)
$X \leftarrow \textsc{BroadcastCopy}(\hat{X}, B)$ ▷ $X \in \mathcal{V}^{B \times |\hat{X}|}$
$X \leftarrow \textsc{Renoise}(X,\ 0.9)$ ▷ renoise 90%, including UPM Masks
$k \leftarrow \lfloor 0.9|\hat{X}|/F \rfloor$
for $t = 1$ to F **do**
  $(\hat{X}_0,\ \text{conf}) \sim p_\theta(X_0 \mid X)$ ▷ batch forward
  $X \leftarrow \textsc{Unmask}(\hat{X}_0,\ \text{conf},\ k)$ ▷ unmask top-k confidence positions
end for
Parse $\{X^{(b,:)}\}_{b=1}^{B}$ into $\{(y_b, s_b)\}_{b=1}^{B}$ ▷ $\{(\text{pred}_b/\text{conf}_b)\}_{b=1}^{B}$
$y^\star \leftarrow \textsc{MajorityVote}(\{y_b\}_{b=1}^{B};\ \text{tie} \Rightarrow \arg\max s_b)$
return $y^\star$

Algorithm 2: Renoise Consistency. In Section 4.2, UPM corresponds to Step 1, BroadcastCopy and Renoise correspond to Step 2, and Unmask and MajorityVote correspond to Step 3.

[1]For simplicity, we denote the fully masked prompt by X_1 which omits the query tokens from the actual input sequence ($\big[\text{query};\ \texttt{[MASK]}^{|L|}\big]$).

B Hyperparameter Settings

The proposed Adaptive Block Drafting (ABD) and Renoise Consistency (ReCon) can be further optimized through hyperparameter tuning. We transparently report the representative settings used in our main experiments in Table 8, and also provide the hyperparameter search results for ABD in Table 9. Additional engineering ablation studies are discussed in Section E. Following the general parallel decoding approach, confidence fallback [14], ABD applies top-k = 2 in fallback scenarios.

Table 8: **Hyperparameter settings.** Default settings for different decoding methods.

Method	**LLaDA**	**DREAM**
semi-AR	block size = 32, top-k = 2	
conf. fallback	block size = 32, τ = 0.9, fallback top-k = 2	
KLASS	τ = 0.6, kl = 0.015	τ = 0.9, kl = 0.001
WINO	drafting threshold = 0.7, verification threshold = 0.9	
Saber	n = 2, μ = 8	
ABD	τ = 0.9, α = 0.7, β = 0.6	τ = 0.9, α = 0.9, β = 0.5
ReCon	N = 15, steps = 5, UPM bottom-u = 3	

Table 9: **Ablation study on β and α.** β (flexible block sizing) and α (tail pruning) for MATH accuracy (%).

LLaDA			**DREAM**		
β	α	Acc	β	α	Acc
0.9	0.7	32.6	0.9	0.9	**38.6**
0.8	0.7	30.0	0.8	0.9	37.6
0.7	0.7	<u>34.4</u>	0.7	0.9	32.6
0.6	0.7	**35.6**	0.6	0.9	37.2
0.5	0.7	32.2	0.5	0.9	<u>38.0</u>

LLaDA			**DREAM**		
β	α	Acc	β	α	Acc
0.6	0.9	<u>33.8</u>	0.5	0.9	<u>38.0</u>
0.6	0.8	32.6	0.5	0.8	**38.8**
0.6	0.7	**35.6**	0.5	0.7	37.8
0.6	0.6	29.6	0.5	0.6	35.8
0.6	0.5	29.6	0.5	0.5	32.6

C Additional Ablation Study on Adaptive Block Drafting

C.1 Additional Analysis of Tail Pruning

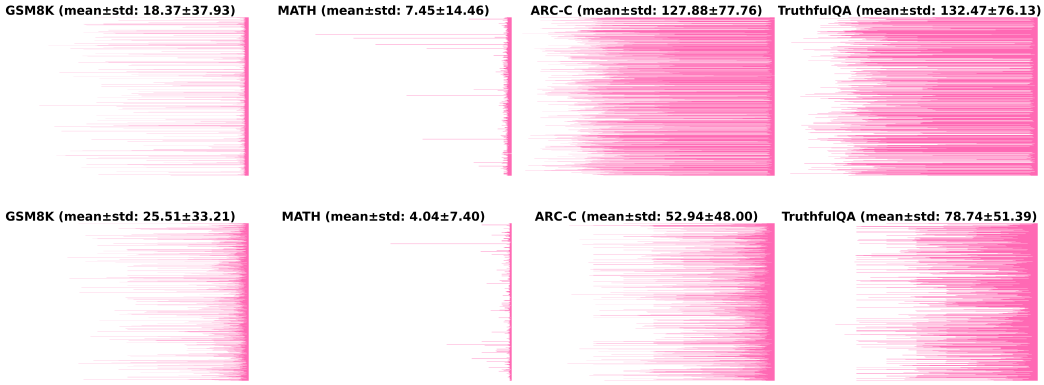


Figure 5: **Pruned tail length (pink) visualization.** Top: `LLaDA 8B Instruct`. Bottom: `Dream 7B Instruct`.

Tail Pruning is designed to reduce FLOPs wasted on generating unnecessary PAD tokens by predicting the output length desired by the model. As shown in Figure 5, for the mathematically intensive

reasoning dataset MATH, the model desires the longest reasoning trajectory, and thus pruning is minimal. In contrast, for multi-choice tasks such as ARC-C and TruthfulQA, where the input questions contain many clues that allow reasoning to be shortened, the model's desired output length is relatively shorter, leading to a larger amount of pruning. We verify through the re-paraphrase experiment in Table 10 that this difference is **not** caused by the model memorizing the dataset formats. In other words, the observed variation in tail length arises from a characteristic where the model implicitly encodes the desired output length during the first forward pass.

Table 10: **Effect of stylistic paraphrasing on tail length.** The tail length is not determined solely by the stylistic patterns of the input prompt that the model has memorized. Even when we use `GPT-4o-mini` to re-paraphrase the questions in GSM8K and ARC-C while preserving their semantic content but substantially altering the question style, we observe similar tail lengths. This suggests that the model implicitly encodes the final output length from the very first forward pass conditioned on the input prompt, as we argued in Section 4.3.

Prompt	Method	GSM8K			ARC-C		
		Acc. ↑	Length	ROUGE-1	Acc. ↑	Length	ROUGE-1
Original	semiAR	69.45%	256.00	F1 0.75±0.11 P 0.75 / R 0.75	81.31%	256.00	F1 0.74±0.07 P 0.70 / R 0.80
	Ours	**78.39%**	**230.49**		**87.20%**	**203.06**	
Paraphrased	semiAR	72.23%	256.00		83.41%	256.00	
	Ours	**78.28%**	**233.19**		**88.64%**	**214.00**	

The ROUGE-1 scores reported in Table 10 confirm that re-paraphrasing introduces sufficient variation in question styles. We enforced the system prompt to preserve the content of the questions while allowing stylistic changes, and additionally applied a verification procedure using `GPT-4o-mini` [23], to filter out potentially hallucinated instances. After verification, 48 out of 1,319 instances in GSM8K and 63 out of 1,172 instances in ARC-C were identified as hallucinated and excluded from evaluation.

The results demonstrate that the length of the pruned tails in the re-paraphrased dataset is similar to that in the original dataset. For example, the average generation lengths after pruning in GSM8K are 233.19 and 230.49, respectively, indicating strong consistency. This highlights the stability and robustness of our first-step tail pruning, suggesting that the observed behavior is not a pattern overfitted to a specific dataset format during training, but rather reflects DLMs' implicit ability to predict output length at the first step. We explicitly extract this capability in Adaptive Block Drafting, thereby reducing token waste.

D Additional Ablation Study on Renoise Consistency

D.1 Clarification on Additional Steps Computation

Across all experiments in Table 2, we additionally generate all samples using only **6 forward steps**. Given a prepared output $\hat{X}$, prompt length P, output length L, and the number of self-correction candidates being N, the 6 steps are computed as follows:

- **1 step** consists of a single Uncertainty-aware pre-masking (UPM) forward pass over the sequence of shape $[1, P + L]$.
- The remaining **5 steps** are applied in parallel over a broadcasted batch of shape $[N, P + L]$, where $\hat{X}$ is replicated N times.

Renoise-consistency. ReCon is an efficient post-hoc self-correction method applied to an already generated output $\hat{X}$. It rapidly produces N diverse candidates and selects the best one. To maximize self-consistency, we perform *majority voting* and choose the most frequently occurring value. In the event of a tie (*i.e.*, two or more values appearing with the same highest frequency), we select the final output as the sample with the highest mean confidence.

Noise injection and fixed-step unmasking. To generate diversity, we inject 90% randomized re-mask noise patterns into the batch of shape $[N, P : P + L]$, while also including fixed UPM masks. This allows us to allocate the top-k unmasking budget across the 90% masked tokens over five steps, enabling the use of a fixed number of steps.

For example, when the output length of $\hat{X}$ is 256, 90% corresponds to 230 tokens. To unmask all 230 tokens within 5 steps, we compute:

$$230 \,//\, 5 = 46.$$

Thus, at each step, we unmask the **top-46 confidence tokens in parallel**.

Prior work [25] reports that such massive simultaneous unmasking (*e.g.*, 46 tokens at once) can cause collapse; however, in Section 4, we present a method that fundamentally resolves this issue.

When using conventional sampling methods [14, 22, 37, 38], generating N sequences of similar quality requires $N \times 128$ decoding steps for an input of length L, based on semi-autoregressive decoding with top-2 confidence unmasking. In contrast, our method requires only 1×1 step for UPM and an additional $N \times 5$ steps. Therefore, comparing $128N$ with $5N + 1$, our approach achieves approximately $\times 25$ reduction in computing cost for generating N candidates.

Furthermore, conventional sampling methods for DLMs have clear limitations not only in terms of cost, but also from the perspective of robustly generating diverse candidates. For this discussion, please refer to Appendix H.

D.2 Analysis of Hint Context Initialization

In the ReCon framework, we demonstrate that constructing a scattered hint context enables the recovery of the original output quality within only a few steps. One might question whether it is possible to entirely eliminate the process of constructing such hints from samples generated by other multi-step decoding methods. However, as quantitatively shown in Table 13, the hint context derived from the early steps of naive parallel generation fails to achieve either sufficient diversity or high generation quality. In this section, we provide intuitive insights into the source of this discrepancy through concrete qualitative examples.

For instance, Figure 6 initializes the hint by retaining the top 10% most confident tokens selected via a conventional top-k strategy. Because these tokens tend to be clustered rather than dispersed, this initialization maximizes the mismatch between the uniform masking objective used during training and the unmasking process at inference time, leading to collapsed samples. This collapse resembles the "After-EoT collapse" phenomenon reported in prior work [15], where padding tokens accumulate in reverse order. Furthermore, since the top-10% selection is deterministic, it inherently limits sample diversity, thereby severely constraining the model's capacity for self-correction.

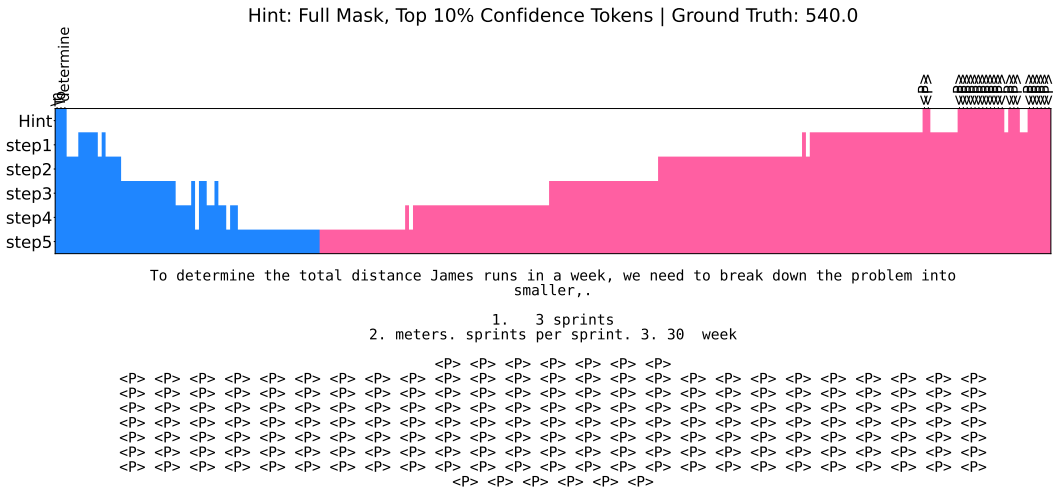


Figure 6: **Failure of confident hint initialization.** Hint initialization from naive forward generation by retaining the top 10% most confident tokens.

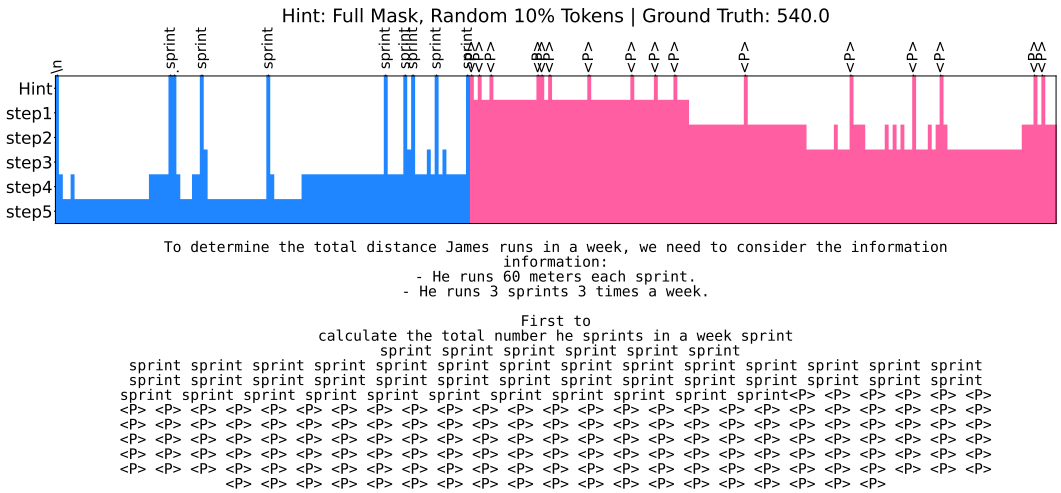


Figure 7: **Failure of random hint initialization.** Hint initialization from naive forward generation by retaining a uniform random 10% of tokens.

Figure 7 attempts to mitigate this issue by uniformly retaining 10% of tokens sampled from the forward probability distribution at an early step. However, due to the instability of the forward distribution in the early stages, repeated collapse occurs within the hint itself, and the subsequent sampling process is further degraded through error propagation, ultimately ruining the generated samples.

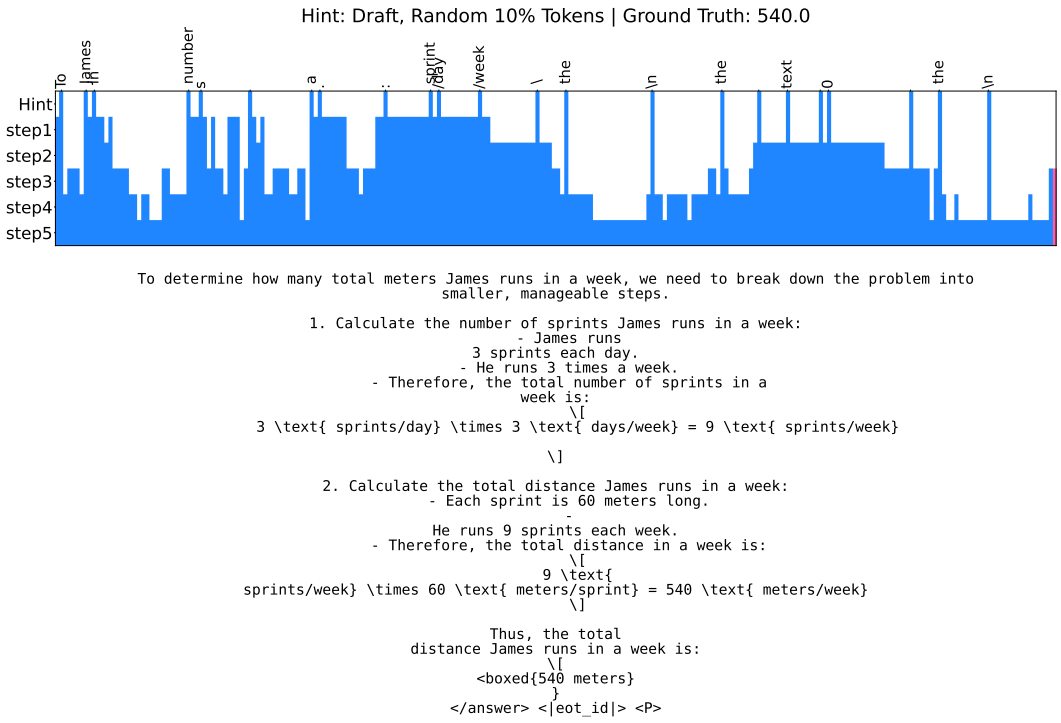
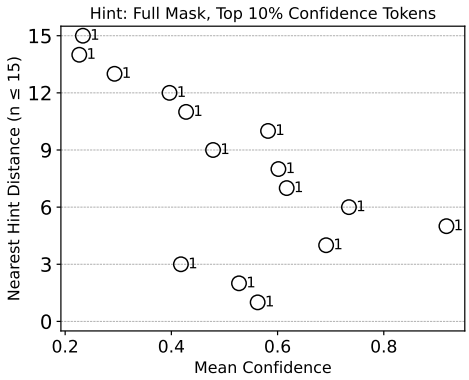
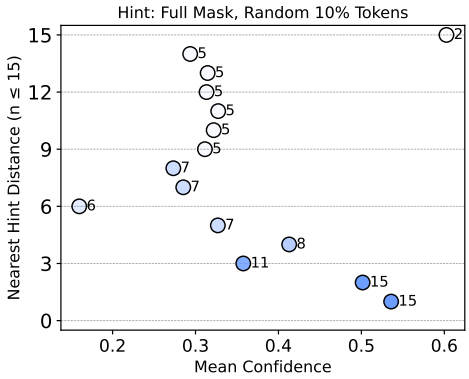


Figure 8: **Effective initialization with Renoise Consistency.** Hint initialization using Renoise Consistency by retaining a uniform random 10% of tokens from a previously generated sample.

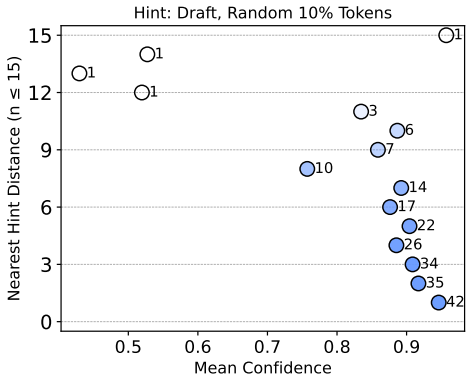
In contrast, Figure 8 illustrates our proposed hint initialization strategy. Uniformly retaining a random 10% of tokens from a readily generated sample is compatible with the outputs of any decoding method. This approach not only restores the original quality within just five steps but also enables self-correction by increasing the diversity of the hint context. We ensure this diversity efficiently through the method illustrated in Figure 1b.



(a) Top-10%



(b) Random 10%



(c) Ours

Figure 9: **Token confidence under different hint initializations.** Renoise Consistency achieves consistently higher confidence for tokens to be unmasked compared to alternative hint initialization strategies.

Furthermore, Figure 9 shows that the majority of tokens to be unmasked in (a) and (b) exhibit significantly lower confidence compared to those in (c). This indicates that simple scattering in hint construction alone is insufficient to achieve the goal of few-step full-diffusion reconstruction, thereby highlighting the effectiveness of our ReCon framework.

D.3 Performance with Varying Numbers of Candidates and Steps

Table 11: **Effect of the number of candidates in ReCon.**

Method	GSM8K		ARC-C		TruthfulQA		MATH	
	Acc ↑	Steps ↓	Acc ↑	Steps ↓	Acc ↑	Steps ↓	Acc ↑	Steps ↓
LLaDA								
draft + ReCon (N = 3, 5 steps)	78.85	74.63	79.35	50.06	52.05	47.56	32.60	88.64
draft + ReCon (N = 6, 5 steps)	80.14	74.63	81.48	50.06	52.05	47.56	35.20	88.64
draft + ReCon (N = 9, 5 steps)	80.52	74.63	81.66	50.06	52.63	47.56	34.60	88.64
draft + ReCon (N = 12, 5 steps)	80.29	74.63	81.31	50.06	51.90	47.56	35.40	88.64
draft + ReCon (N = 15, 5 steps)	81.05	74.63	81.06	50.06	53.36	47.56	35.60	88.64
DREAM								
draft + ReCon (N = 3, 5 steps)	70.51	82.61	79.69	71.23	54.09	67.62	32.60	94.91
draft + ReCon (N = 6, 5 steps)	77.56	82.61	86.52	71.23	59.36	67.62	33.80	94.91
draft + ReCon (N = 9, 5 steps)	77.94	82.61	88.23	71.23	60.96	67.62	37.20	94.91
draft + ReCon (N = 12, 5 steps)	79.00	82.61	87.97	71.23	59.36	67.62	37.00	94.91
draft + ReCon (N = 15, 5 steps)	78.39	82.61	87.88	71.23	59.50	67.62	38.00	94.91

Table 12: **Effect of the number of steps in ReCon.**

Method	GSM8K		ARC-C		TruthfulQA		MATH	
	Acc ↑	Steps ↓	Acc ↑	Steps ↓	Acc ↑	Steps ↓	Acc ↑	Steps ↓
LLaDA								
draft + ReCon (N = 15, 1 step)	73.31	70.63	80.46	46.06	51.32	43.56	30.00	84.64
draft + ReCon (N = 15, 5 steps)	81.05	74.63	81.06	50.06	53.36	47.56	35.60	88.64
draft + ReCon (N = 15, 10 steps)	80.82	79.63	81.83	55.06	53.51	52.56	36.00	93.64
draft + ReCon (N = 15, 15 steps)	81.05	84.63	83.45	60.06	53.95	57.56	36.20	98.64
draft + ReCon (N = 15, 20 steps)	81.35	89.63	84.56	65.06	54.09	62.56	35.80	103.64
DREAM								
draft + ReCon (N = 15, 1 step)	48.67	78.61	86.43	67.23	57.75	63.62	25.20	90.91
draft + ReCon (N = 15, 5 steps)	78.39	82.61	87.88	71.23	59.50	67.62	38.00	94.91
draft + ReCon (N = 15, 10 steps)	79.30	87.61	88.05	76.23	58.92	72.62	39.80	99.91
draft + ReCon (N = 15, 15 steps)	79.83	92.61	88.31	81.23	60.38	77.62	40.00	104.91
draft + ReCon (N = 15, 20 steps)	79.83	97.61	88.05	86.23	60.23	82.62	40.60	109.91

Tables 11 and 12 illustrate the impact of ReCon under different numbers of candidate samples and full-space diffusion reconstruction steps, as analyzed in Section E. The numerical results indicate performance gains with additional computational cost, and we select a cost-efficient configuration while retaining strong performance as our representative setting (N = 15, 5 steps).

D.4 Effectiveness of Anchor Initialization

Table 13: **Effectiveness of ReCon's hint initialization strategy.** ReCon enables voting over high-quality samples by injecting diverse post-hoc renoising patterns into generated outputs. The top 10% is selected based on confidence scores.

Method (source → strategy)	vote	GSM8K	ARC-C	TruthfulQA	MATH
LLaDA					
(a) 1 step → top 10% hint	–	16.91	3.84	1.46	12.20
(b) 1 step → random 10% hint	✓	46.47	66.72	41.81	22.60
(c) ABD → top 10% hint	–	58.00	76.11	48.54	23.60
(d) ABD → ReCon	✓	**80.74**	**80.63**	**51.61**	**35.80**
DREAM					
(a) 1 step → top 10% hint	–	2.35	0.00	0.00	0.80
(b) 1 step → random 10% hint	✓	21.91	31.40	11.84	12.60
(c) ABD → top 10% hint	–	33.43	63.48	43.57	17.00
(d) ABD → ReCon	✓	**78.39**	**87.88**	**59.50**	**38.00**

Table 13 presents an ablation study on hint context construction. (a) and (b) use one-step-forwarded full-mask samples, while (c) and (d) use a draft generated by ABD as hint source. Concentrated high-confident hints generated by using the prediction from the full mask in (a) fail to control both quality and diversity, leading to a complete performance collapse to 0%. Random selection in (b) slightly improves performance but still fails to preserve generation quality, due to the low quality of anchor tokens. By leveraging the draft as hint source, the performance is significantly increased, but the deterministic hints in (c) lack diversity. In contrast, ReCon's post-hoc renoising pattern injection (d) enables diversity with preserved quality, thereby unlocking effective self-correction. In particular, the qualitative examples in Appendix D.2 provide a stark contrast between the collapse and success of reconstruction processes under different hint contexts.

D.5 Uncertainty-Aware Pre-Masking

When performing a single forward pass on maskless (*i.e.*, fully unmasked) samples generated by various decoding strategies, the resulting probability distributions vary significantly depending on the location and amount of erroneous tokens. As illustrated in Figure 10, DLMs are able to identify positions where reasoning errors occur by leveraging their bidirectional contextual understanding. Meanwhile, we empirically observe that both `LLaDA 8B Instruct` and `Dream 7B Instruct` consistently assign low confidence to the "." (period) token, which we attribute to issues in the datasets used during training. However, this token is not related to contextual information and does not substantially contribute to error propagation detection, and is therefore not considered in UPM masking.

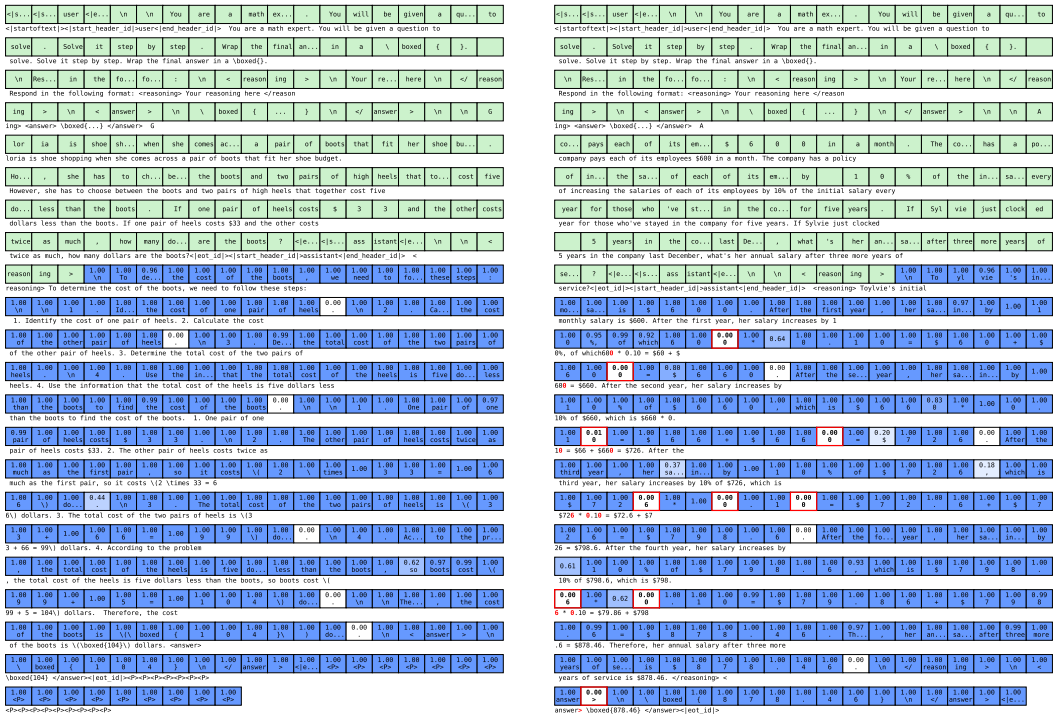


Figure 10: **Error propagation detection capability of UPM.** By performing a single forward pass on a fully unmasked sample (*i.e.*, without any masking), DLMs can identify positions where reasoning errors occur based on their bidirectional contextual understanding. For instance, in scenarios where error propagation happens (right), the confidence of error-related key contextual information tokens drops dramatically (below 0.1). By excluding these positions from the hint context, the robustness of Renoise Consistency can be strengthened. In contrast, when correct reasoning and answers are generated (left), the confidence of the key contextual information tokens remains nearly unchanged.

D.6 Performance with Varying Numbers of bottom-u Tokens of UPM

Table 14: **Sensitivity analysis of UPM.** Effect of the choice of bottom-u tokens on performance.

Model	TruthfulQA			MATH		
	Acc ↑	Lat. ↓	Steps ↓	Acc ↑	Lat. ↓	Steps ↓
LLaDA						
$u = 0$	50.58	2.05 (×1.00)	55.57 (×1.00)	33.60	7.30 (×1.00)	92.53 (×1.00)
$u = 3$	51.61	2.06 (×1.00)	56.57 (×1.02)	34.20	7.33 (×1.00)	93.53 (×1.01)
$u = 6$	52.34	2.06 (×1.00)	56.57 (×1.02)	33.80	7.33 (×1.00)	93.53 (×1.01)
$u = 9$	52.78	2.07 (×1.01)	56.57 (×1.02)	35.40	7.33 (×1.00)	93.53 (×1.01)
$u = 12$	53.07	2.07 (×1.01)	56.57 (×1.02)	34.60	7.32 (×1.00)	93.53 (×1.01)
$u = 15$	52.92	2.06 (×1.00)	56.57 (×1.02)	35.00	7.31 (×1.00)	93.53 (×1.01)

Table 14 shows that a broad range of u values (3–15) consistently improves performance over $u = 0$. This result validates the proposed UPM approach and demonstrates that the performance gain is not due to a specific hyperparameter choice.

D.7 Examples of Anchor Token Types

Induced numerical. Digits of intermediate values the draft computed itself.

```
Half of them were Trekking bikes, so he sold 300/2 = 150 Trekking bikes.
The difference between the weekly electric bill before and after is $147 -
$126 = $21.  So, it costs $65,990 more to repave Lewis' street than Monica's
street. Remaining play count for the first DVD = 1000 - 356 = 644 times.
```

Formatting. Punctuation, delimiters, and markup that carry format information.

```
Each quarter is worth $0.25, and there are 32 quarters.
Total cost of purchased fruits = $1.50 + $2.00 + $3.00 + $3.75 = $9.25
So, 50,000 - (8,000 + 4,000 + 24,000) = 50,000 - 36,000 = 14,000 bars.
Igor gathered half as many test tubes as requested:  so he gathered 16 / 2
= 8 test tubes.
```

Reasoning-structure. Operators and relation words that fix how quantities are combined.

```
60 milliliters from ice cubes + 5 milliliters from cream = 65 milliliters.
The daily cost after adding the new device is 14 kilowatts * $1.50 per
kilowatt-hour = $21 per day.
Profit = Total revenue for 10 jars - Total cost for 10 jars = $500 - $440 =
$60.
Since he buys 2 pairs of shoes a month, he buys 12 * 2 = 24 pairs of shoes
in a year.
```

Input numerical. Numbers restated from the problem rather than computed.

```
Total cost for Lewis' street = 490 meters * $194/meter = $95,060.  Gissela's
truck can haul 4,000 pounds of gravel.
Next, John decides to pay off his debts using 60% of the loan amount.
The total cost of the house and the lot is $120,000, so we can write the
equation L + 3L = 120,000.
```

Local lexical. Words that constrain the wording but not the logic.

```
The company sold 4000 gallons of milk at $3.5 per gallon.
Gissela's truck can haul 4,000 pounds of gravel.
Andrea has 44 + 52 = 96 apples and 48 / 2 = 24 bananas.
Saanvi combines 14 mL of the first medicine with 3 times that amount of the
second medicine.
```

E Additional Analyses

Table 15: **Effect of renoise rate (R) in ReCon.**

R	GSM8K	ARC-C	TruthfulQA	MATH
LLaDA				
0.9	**81.05**	**81.06**	**53.36**	**35.60**
0.7	79.08	80.89	52.78	33.60
0.5	79.15	80.55	52.05	33.40
0.3	79.00	80.46	52.78	32.80
0.1	79.08	80.12	52.05	32.80
DREAM				
0.9	**78.39**	**87.88**	59.50	**38.00**
0.7	76.65	86.26	**59.65**	36.80
0.5	76.19	85.92	**59.65**	35.40
0.3	75.97	85.92	59.21	35.80
0.1	76.04	85.84	58.92	35.60

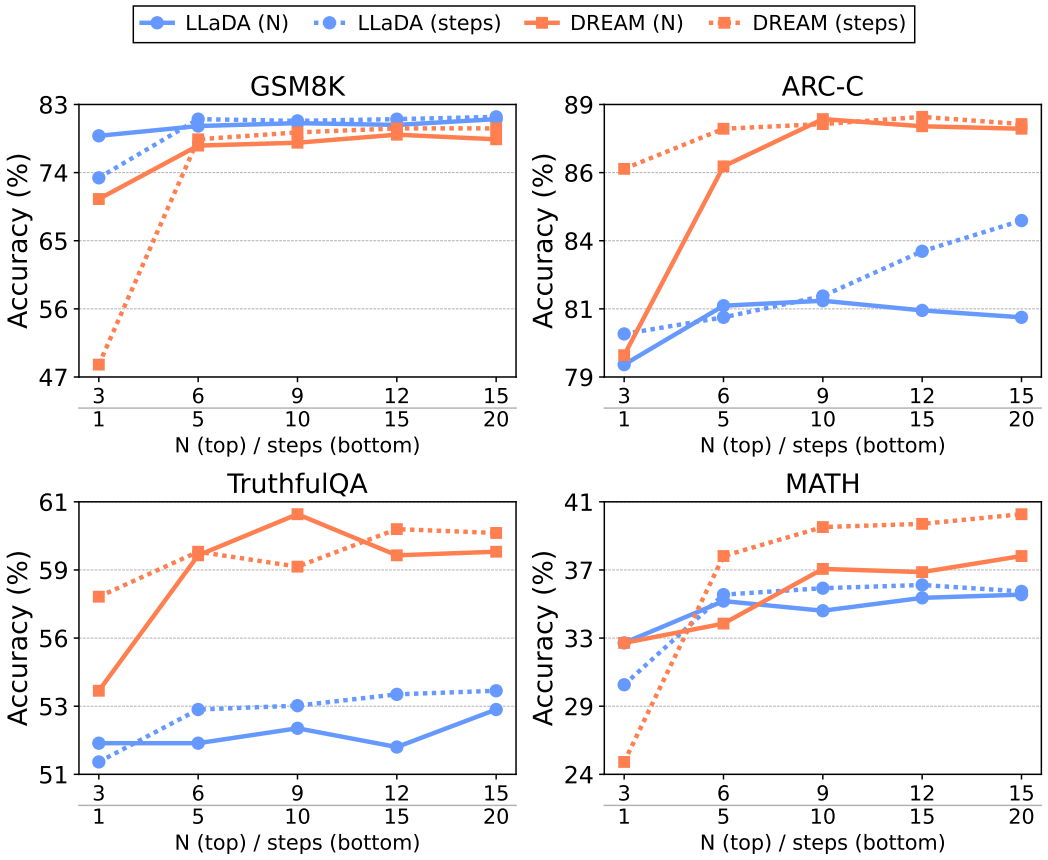


Figure 11: **Accuracy variation with respect to the number of candidates (N) and the number of full-space diffusion steps (steps) in ReCon.** The top x-ticks correspond to N with steps = 5, while the bottom x-ticks corresponds to the number of steps with N = 15. See Appendix D.3 for full information.

Table 15 studies renoising effects, and Figure 11 shows sample and step effects. As shown in Table 15, increasing the renoise rate (*i.e.*, minimizing the hint context) enables DLMs to explore more diverse correction possibilities, thereby improving the quality of self-correction. Figure 11 illustrates the performance gains achieved by explicitly increasing the number of correction candidates

or by increasing the number of full-space diffusion reconstruction steps. Notably, the few-step reconstruction strategy yields drastic quality improvements with only five steps, leading us to adopt 5 steps as our default baseline. Consequently, ReCon consistently achieves stable performance improvements by incurring only an additional cost of 6 steps (1 step UPM + 5 steps), regardless of the decoding strategy (Table 2).

F Evaluation on Additional Models and Datasets

F.1 Additional Performance Comparison on LLaDA 1.5

Table 16: **Evaluation on LLaDA 1.5.** Values denote accuracy (%) and number of steps.

Method	GSM8K		MATH		ARC-C		TruthfulQA	
	Acc↑	Steps↓	Acc↑	Steps↓	Acc↑	Steps↓	Acc↑	Steps↓
Semi-AR	80.14	128	32.80	128	79.95	128	51.02	128
Self-Refine	80.52(+0.38)	384(×3.00)	31.80(-1.00)	384(×3.00)	81.57(+1.62)	384(×3.00)	50.44(-0.58)	384(×3.00)
Ours (ABD + ReCon)	**82.49**(+2.35)	**90.44**(×0.71)	**36.00**(+3.20)	**106.26**(×0.83)	**84.30**(+4.35)	**78.05**(×0.61)	**53.80**(+2.78)	**76.73**(×0.60)

LLaDA 1.5 [40] improves performance by applying Variance-Reduced Preference Optimization (VRPO) to LLaDA-8B-Instruct [22]. We show that ReCon not only significantly outperforms Self-Refine [21] on LLaDA 1.5 as well, but also requires up to 5× fewer forward steps.

F.2 Evaluation of Open-Ended Generation

We evaluate not only the reasoning capability of ABD + ReCon, but also its performance on instruction-based open-ended text generation. The evaluation results are shown in Table 17. Unlike reasoning tasks, where predictions can be parsed and aggregated through majority voting, parsing is difficult in open-ended text generation. Therefore, we leverage a key advantage of UPM (section 4.2) — namely, its ability to assign very low confidence to uncertain positions.

Specifically, we perform an additional single forward pass on the batch sequences generated by ABD + ReCon, and select the final output sequence with the highest average confidence. This method achieves strong performance in terms of win rate and length-controlled (LC) win rate [7] compared to the semi-AR baseline.

Table 17: **Evaluation on open-ended generation tasks.** W and LC refer to Win rate and Length-Controlled Win rate [7], respectively.

Method	Arena Hard Auto			MT Bench			Wiki			Alpaca Eval		
	W	LC	steps	W	LC	steps	W	LC	steps	W	LC	steps
LLaDA												
semi-AR	50.00	50.00	128	50.00	50.00	128	50.00	50.00	128	50.00	50.00	128
ABD + ReCon (20 steps)	63.38	63.60	144.91	56.48	57.36	132.66	56.79	55.36	148.50	61.68	62.10	142.70
ABD + ReCon (30 steps)	**69.11**	**69.15**	154.91	**60.13**	**62.46**	142.66	**65.59**	**64.54**	158.50	**68.70**	**69.68**	152.70
DREAM												
semi-AR	50.00	50.00	128	50.00	50.00	128	50.00	50.00	128	50.00	50.00	128
ABD + ReCon (20 steps)	52.68	55.08	142.04	54.67	54.36	124.40	55.31	54.72	147.05	53.97	53.14	135.93
ABD + ReCon (30 steps)	**56.87**	**58.87**	152.04	**59.37**	**60.46**	134.40	**60.48**	**60.19**	157.05	**61.68**	**62.10**	145.93

Following prior work [25], we center our evaluation on AlpacaEval [7] with G-eval [20], which is known to have the highest alignment with human judgments, and assess open-ended generation performance across four benchmarks: AlpacaEval, MT-Bench [39], Wiki, and Arena-Hard-Auto [16]. The Wiki dataset consists of 500 Wikipedia entities randomly sampled from the test set of Mintaka [24], with each entity prompted by an instruction requesting an explanation. Responses are evaluated using the same metric as AlpacaEval, where the output under evaluation is compared against a reference output to measure the win rate. The reference outputs are generated using the semi-AR baseline. We report G-eval results using `GPT-4o-mini` as the evaluator.

As shown in Table 17, ABD + ReCon achieves a high win rate over the standard semi-AR baseline even on open-ended generation tasks, and this performance improves substantially as the number

of full-space diffusion steps increases. This demonstrates that our method is effective not only for reasoning tasks but also for more flexible generation tasks.

G Further Wall-Clock Latency Measurement

Table 18: **Wall-clock time comparison.** Per-sample decoding time for semi-AR (top-2 unmasking at each step) and our method (ABD + ReCon).

Model	Dataset	Accuracy		Time (s)		Speedup
		Top-2	ABD + ReCon	Top-2	ABD + ReCon	
LLaDA	GSM8K	75.21	81.05 (+5.84)	4.82	3.76 (-1.06)	1.28×
	ARC-C	78.41	81.06 (+2.65)	4.87	2.92 (-1.95)	1.67×
	TruthfulQA	51.61	53.36 (+1.75)	4.83	2.73 (-2.10)	1.77×
	MATH	33.00	35.60 (+2.60)	4.85	4.42 (-0.43)	1.10×
Dream	GSM8K	69.45	78.39 (+8.94)	3.71	3.21 (-0.50)	1.16×
	ARC-C	81.31	87.88 (+6.57)	3.70	2.93 (-0.77)	1.26×
	TruthfulQA	51.02	59.50 (+8.48)	3.71	2.72 (-0.99)	1.36×
	MATH	29.00	38.00 (+9.00)	3.71	3.60 (-0.11)	1.03×

Table 19: **Wall-clock time comparison.** Per-sample decoding time for KLASS and our method (ABD + ReCon).

Model	Dataset	Accuracy		Time (s)		Speedup
		KLASS	ABD + ReCon	KLASS	ABD + ReCon	
LLaDA	GSM8K	79.38	81.05 (+1.67)	4.66	3.76 (-0.90)	1.24×
	ARC-C	77.99	81.06 (+3.07)	3.88	2.92 (-0.96)	1.33×
	TruthfulQA	49.12	53.36 (+4.24)	4.04	2.73 (-1.31)	1.48×
	MATH	32.00	35.60 (+3.60)	5.68	4.42 (-1.26)	1.29×
Dream	GSM8K	76.80	78.39 (+1.59)	5.18	3.21 (-1.97)	1.61×
	ARC-C	85.24	87.88 (+2.64)	2.86	2.93 (+0.07)	0.98×
	TruthfulQA	55.26	59.50 (+4.24)	2.69	2.72 (+0.03)	0.99×
	MATH	35.80	38.00 (+2.20)	5.34	3.60 (-1.74)	1.48×

Table 18 and Table 19 present the wall-clock latency of our full ABD + ReCon pipeline. Our approach achieves up to a 9.00% performance improvement over the semi-AR baseline while being faster in latency (1.03× speedup), and consistently outperforms the state-of-the-art parallel decoding method KLASS [14] across models and benchmarks, offering latency benefits in most cases and only minor overhead (1–2%) in a few cases while still improving performance by up to 4.24%.

H Difficulty of Stochastic Sampling in DLMs

For stochastic sampling in DLMs, prior work [14, 22, 38] adds Gumbel noise to the logits for diversity. Let $\pi = (\pi_1, \ldots, \pi_V)$ denote the original class probabilities over V categories ($\pi_i \geq 0$, $\sum_{i=1}^{V} \pi_i = 1$). They draw Gumbel noise $g_i \sim \mathrm{Gumbel}(0, 1)$ independently for each class and sample

$$n = \arg \max_{i \in \{1,\ldots,V\}} \left(\log \pi_i + g_i\right), \tag{4}$$

where $n \in \{1, \ldots, V\}$ is the sampled category (vocabulary) index. Because g_i changes the token index and positions to be unmasked at each forward step, it enables generating diverse samples.

Meanwhile, Ye et al. [37] simply promotes diversity using traditional sampling methods in ARMs, such as top-p sampling [11]. However, these sampling strategies can cause collapse in DLMs; we observe that these strategies significantly degrade output quality even with one of the most recent state-of-the-art parallel decoding methods [14] in our experiments (Figure 12). This issue arises from the "curse of parallel decoding" [34]: simultaneously unmasked multiple tokens are in fact sampled *independently*, which can produce undesirable combinations. Prior sampling strategies inevitably accelerate this curse of parallel decoding, because they disrupt the model's predictions *independently* at each token position, thereby breaking fluent token combinations. In the end, any method that performs parallel decoding carries the risk of this kind of sampling collapse, meaning that with traditional sampling strategies it is difficult to obtain diversity in a stable way.

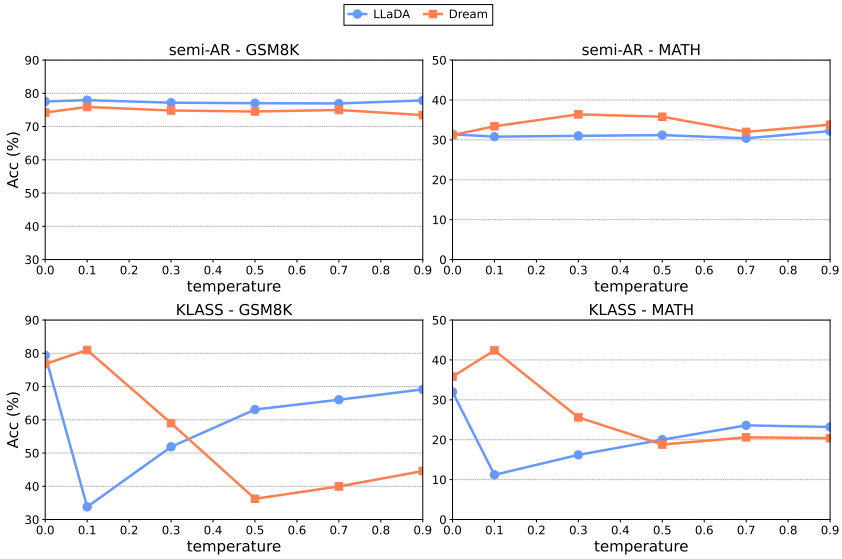


Figure 12: **Stochastic sampling collapse in SOTA parallel decoding.** Any stochastic sampling strategies [22, 37] lead to drastic performance degradation in state-of-the-art parallel decoding methods [14], while non-parallel semi-AR decoding remains stable.

I Limitations

Due to limitations in computing power, we evaluated our method on widely used diffusion LLMs at the 7–8B scale, but evaluations at substantially larger scales are still lacking. In addition, our method introduces multiple modules and therefore involves a number of hyperparameters. Although we showed that our method operates robustly across a wide range of sweeps, there may still exist potentially better configurations.

Regarding the societal impact of this research, our method renoises the inference outputs of DLMs and generates robust and diverse candidates at low cost. If misused, this could lead to harm by enabling the mass production of fake content that is difficult to distinguish from authentic content. Therefore, to mitigate such risks, the development of sampling methods should be accompanied by ethical guidelines.

NeurIPS Paper Checklist

1. **Claims**

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

   Answer: [Yes]

   Justification: The abstract and introduction of this paper clearly specify its contributions and scope: it presents a new decoding method for Diffusion Large Language Models (DLMs).

   Guidelines:

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

2. **Limitations**

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

   Answer: [Yes]

   Justification: Section I is an independent section that discusses the limitations of this work.

   Guidelines:

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

3. **Theory assumptions and proofs**

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

   Answer: [N/A]

Justification: The motivation of this work is the mismatch between the theoretical learning objective in Eq. 1 and the current inference paradigm of Diffusion LLMs, and it addresses this problem primarily through empirical findings and comprehensive ablations.

Guidelines:

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

4. **Experimental result reproducibility**

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

   Answer: [Yes]

   Justification: The main experimental results can be reproduced by referring to the specific details in Section 4, Section 5, Appendix A, and Appendix B.

   Guidelines:

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

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

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

Answer: [Yes]

Justification: We include access to the full code to reproduce the main experimental results.

Guidelines:

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

6. **Experimental setting/details**

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

Answer: [Yes]

Justification: The methods proposed in this work are training-free, and all details required for inference can be found in Section 4, Section 5, Appendix A, and Appendix B.

Guidelines:

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

7. **Experiment statistical significance**

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

Answer: [No]

Justification: Our evaluation focuses primarily on reasoning datasets and open-ended generation datasets. For the reasoning datasets, we use the full test sets with a fixed seed across all experiments, so the results are deterministic, reproducible, and unaffected by subsampling variance. For the open-ended generation experiments, we follow the evaluation protocol of prior work [25], as described in Appendix F.2. As a result, statistical significance testing is not central to our evaluation setup.

Guidelines:

- The answer [N/A] means that the paper does not include experiments.
- The authors should answer [Yes] if the results are accompanied by error bars, confidence intervals, or statistical significance tests, at least for the experiments that support the main claims of the paper.

- The factors of variability that the error bars are capturing should be clearly stated (for example, train/test split, initialization, random drawing of some parameter, or overall run with given experimental conditions).
- The method for calculating the error bars should be explained (closed form formula, call to a library function, bootstrap, etc.)
- The assumptions made should be given (e.g., Normally distributed errors).
- It should be clear whether the error bar is the standard deviation or the standard error of the mean.
- It is OK to report 1-sigma error bars, but one should state it. The authors should preferably report a 2-sigma error bar than state that they have a 96% CI, if the hypothesis of Normality of errors is not verified.
- For asymmetric distributions, the authors should be careful not to show in tables or figures symmetric error bars that would yield results that are out of range (e.g., negative error rates).
- If error bars are reported in tables or plots, the authors should explain in the text how they were calculated and reference the corresponding figures or tables in the text.

8. **Experiments compute resources**

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

   Answer: [Yes]

   Justification: In Section 5, we report the type of GPU used, latency, and peak memory usage.

   Guidelines:

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

9. **Code of ethics**

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

   Answer: [Yes]

   Justification: The research conducted in this paper conforms to the NeurIPS Code of Ethics in all respects. The work does not involve human subjects, sensitive personal data, privacy-invasive data collection, or applications that could reasonably be expected to cause direct harm.

   Guidelines:

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

10. **Broader impacts**

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

    Answer: [Yes]

    Justification: We discuss this issue in Section I.

Guidelines:

- The answer [N/A] means that there is no societal impact of the work performed.
- If the authors answer [N/A] or [No], they should explain why their work has no societal impact or why the paper does not address societal impact.
- Examples of negative societal impacts include potential malicious or unintended uses (e.g., disinformation, generating fake profiles, surveillance), fairness considerations (e.g., deployment of technologies that could make decisions that unfairly impact specific groups), privacy considerations, and security considerations.
- The conference expects that many papers will be foundational research and not tied to particular applications, let alone deployments. However, if there is a direct path to any negative applications, the authors should point it out. For example, it is legitimate to point out that an improvement in the quality of generative models could be used to generate Deepfakes for disinformation. On the other hand, it is not needed to point out that a generic algorithm for optimizing neural networks could enable people to train models that generate Deepfakes faster.
- The authors should consider possible harms that could arise when the technology is being used as intended and functioning correctly, harms that could arise when the technology is being used as intended but gives incorrect results, and harms following from (intentional or unintentional) misuse of the technology.
- If there are negative societal impacts, the authors could also discuss possible mitigation strategies (e.g., gated release of models, providing defenses in addition to attacks, mechanisms for monitoring misuse, mechanisms to monitor how a system learns from feedback over time, improving the efficiency and accessibility of ML).

11. **Safeguards**

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

    Answer: [N/A]

    Justification: The paper poses no such risks.

    Guidelines:

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

12. **Licenses for existing assets**

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

    Answer: [Yes]

    Justification: All datasets used in this work are publicly available and available for research purposes. We use these benchmarks under their respective licenses or terms of use: GSM8K (MIT), MATH-500 (MIT), TruthfulQA (Apache-2.0), ARC-Challenge (CC-BY-SA-4.0), Arena-Hard-Auto (Apache-2.0), MT-Bench (Apache-2.0), Wiki from Mintaka (CC-BY-4.0), and AlpacaEval (CC-BY-NC-4.0 for the evaluation data; Apache-2.0 for the code). We also cite the corresponding papers.

    Guidelines:

    - The answer [N/A] means that the paper does not use existing assets.

- The authors should cite the original paper that produced the code package or dataset.
- The authors should state which version of the asset is used and, if possible, include a URL.
- The name of the license (e.g., CC-BY 4.0) should be included for each asset.
- For scraped data from a particular source (e.g., website), the copyright and terms of service of that source should be provided.
- If assets are released, the license, copyright information, and terms of use in the package should be provided. For popular datasets, `paperswithcode.com/datasets` has curated licenses for some datasets. Their licensing guide can help determine the license of a dataset.
- For existing datasets that are re-packaged, both the original license and the license of the derived asset (if it has changed) should be provided.
- If this information is not available online, the authors are encouraged to reach out to the asset's creators.

13. **New assets**

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

    Answer: [N/A]

    Justification: The paper does not release new assets.

    Guidelines:

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

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

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

    Answer: [N/A]

    Justification: The paper does not involve crowdsourcing nor research with human subjects.

    Guidelines:

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

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

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

    Answer: [N/A]

    Justification: The paper does not involve crowdsourcing nor research with human subjects.

Guidelines:

- The answer [N/A] means that the paper does not involve crowdsourcing nor research with human subjects.
- Depending on the country in which research is conducted, IRB approval (or equivalent) may be required for any human subjects research. If you obtained IRB approval, you should clearly state this in the paper.
- We recognize that the procedures for this may vary significantly between institutions and locations, and we expect authors to adhere to the NeurIPS Code of Ethics and the guidelines for their institution.
- For initial submissions, do not include any information that would break anonymity (if applicable), such as the institution conducting the review.

16. **Declaration of LLM usage**

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

    Answer: [N/A]

    Justification: The core method development in this research does not involve LLMs as any important, original, or non-standard components.

    Guidelines:

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