REVISE WHAT MATTERS: ATTENTION-PRIORITIZED VERIFICATION FOR DIFFUSION LANGUAGE MODELS

Anonymous authors
Paper under double-blind review

ABSTRACT

Diffusion language models enable parallel token generation, but existing revocable decoders often treat uncertainty as sufficient evidence for revision, causing many suspicious predictions to be remasked even when revision provides little benefit. We argue that effective revision should consider not only whether a token is unreliable, but also whether correcting it is likely to matter to the evolving decoding context. We propose CAPER (Context-Aware Probation for Efficient Revision), a training-free revocable decoding framework that introduces a one-step probation state between token commitment and remasking. CAPER combines assignment uncertainty with incoming attention from already decoded and newly drafted positions to prioritize contextually consequential candidates for verification. During probation, priority-weighted drafting limits the influence of unresolved predictions on concurrent generation, while verification determines whether their current assignments should be retained or revoked. Drafting and verification are integrated into the same decoding step, allowing CAPER to concentrate revision on actionable predictions without sacrificing parallel generation. Experiments on LLaDA-8B-Base and Dream-Base-7B across code generation and mathematical reasoning demonstrate a consistently stronger quality–efficiency trade-off than existing irreversible and revocable decoding methods.

1 INTRODUCTION

Autoregressive (AR) language models generate text by predicting each token conditioned on the preceding sequence Vaswani et al. (2017); Brown et al. (2020). Under standard left-to-right decoding, generation is sequential and previously emitted tokens remain fixed, even when the continuation exposes a problematic earlier decision. Masked diffusion language models offer an alternative: they predict missing tokens from a partially observed sequence and progressively remask positions through iterative denoising Sahoo et al. (2024). Recent models such as LLaDA Nie et al. (2025) and Dream Ye et al. (2025) have extended this approach to large-scale language modeling. Their ability to predict multiple positions in one forward pass creates an opportunity to reduce sequential decoding steps.

However, diffusion language models still face a trade-off between decoding speed and generation quality. When multiple tokens are decoded in the same step, they cannot use one another's predicted values as context, which can reduce generation quality Wu et al. (2025). Decoding fewer tokens per step helps preserve quality but requires more decoding steps, slowing down generation Nie et al. (2025); Wu et al. (2025).

Remasking allows the decoder to revisit earlier predictions during generation. ReMDM introduces a probabilistic remasking process with configurable schedules, allowing previously generated tokens to be resampled as decoding continues Wang et al. (2025). WINO combines parallel drafting with bidirectional verification, remasking earlier tokens that become unreliable under the updated context Hong et al. (2025). Saber further adapts the number of tokens generated at each step and remasks earlier tokens whose confidence decreases as new context becomes available Dong et al. (2025). While these methods improve error correction, they primarily identify tokens that are likely to be incorrect. However, a token being unreliable does not necessarily mean that revising it is useful, especially when revision opportunities are limited.

Reviewers: please read the Reviewer Guidelines (iclr.cc/Conferences/2027/ReviewerGuidelines) and the AI Policy for Reviewers (iclr.cc/Conferences/2027/AIPolicyForReviewers). If you used AI to expand, edit, or polish your review, please provide the input text to the LLM. Better still, consider skipping the LLM and submitting your original text: we, and the authors, are much more interested in your unedited thoughts than in what an LLM has to say. AI-assisted or not, you are putting your name and reputation behind your review: LLM-generated falsehoods, hallucinations or misrepresentations are subject to disciplinary action, which may include desk-rejecting all papers you have authored.

We observe two important patterns. First, most remasked tokens return to their previous values within only a few decoding steps, often immediately at the next step. Such revisions introduce extra decoding work without changing the prediction. Second, tokens that do change after remasking tend to receive more attention from positions about to be unmasked and positions already unmasked, suggesting that their correctness has greater influence on subsequent predictions. These observations indicate that effective revision should consider not only whether a token is unreliable, but also whether revising it is likely to produce a meaningful and consequential correction.

To this end, we propose CAPER (*Context-Aware Probation for Efficient Revision*), a training-free revocable decoding framework that introduces an intermediate *probation* state between token commitment and remasking. Rather than immediately revoking a suspicious token, CAPER moves a generated token into probation based on its confidence and the incoming attention it receives from already decoded and newly drafted positions. During probation, the token retains its current assignment and cached context so that parallel drafting can continue, while its value is temporarily masked for contextual verification. CAPER further uses the probation token as an priority-weighted negative guidance direction to limit the influence of a potentially incorrect token on new drafts. Probation ends by either restoring the original token when it is consistently re-predicted with sufficient confidence, or remasking the position for later reconsideration. By decoupling suspicion from immediate revocation, CAPER avoids wasteful mask-and-restore cycles while concentrating revision on tokens whose correction is more likely to matter for subsequent generation.

2 Related Work

Diffusion language models. Diffusion language models generate text through iterative denoising rather than a fixed left-to-right order. D3PM Austin et al. (2021a) and SEDD Lou et al. (2024) develop diffusion models for discrete data, while MDLM Sahoo et al. (2024) simplifies the training of masked diffusion language models. LLaDA Nie et al. (2025) and Dream Ye et al. (2025) extend masked denoising to large-scale language modeling. These models can predict multiple masked positions in a single forward pass, offering an alternative to sequential autoregressive decoding. However, this parallelism does not by itself ensure fast inference. Standard decoding repeatedly processes the evolving sequence, and bidirectional attention prevents the direct reuse of conventional autoregressive key-value (KV) caches Ma et al. (2025); Wu et al. (2025). Reducing the number of denoising steps can also degrade generation quality Wu et al. (2025). Practical acceleration therefore needs to consider both the cost of each forward pass and the number of passes required.

Efficient diffusion decoding. Training-free acceleration for diffusion language models mainly reduces repeated computation or decoding steps. Fast-dLLM and dKV-Cache reuse intermediate KV states to avoid redundant computation Wu et al. (2025); Ma et al. (2025), while EB-Sampler and Adaptive Parallel Decoding adapt parallel token generation based on uncertainty or token dependencies Ben-Hamu et al. (2025); Israel et al. (2025). These methods primarily optimize the generation of new tokens rather than the revision of previously decoded ones. Revision-based methods instead reconsider earlier predictions as context evolves: ReMDM uses probabilistic or confidence-aware remasking Wang et al. (2025), while WINO and Saber combine parallel decoding with confidence-based verification or backtracking Hong et al. (2025); Dong et al. (2025). However, these methods do not explicitly prioritize revision according to how strongly a token influences the evolving context. CAPER addresses this by combining assignment uncertainty with incoming attention to select a small set of tokens for one-step verification, while continuing parallel drafting in the same forward pass.

3 Preliminaries

3.1 Remasking in Revocable Diffusion Decoding

For a fixed prompt X, let $Y^{(t)} = (y_1^{(t)}, \ldots, y_L^{(t)})$ denote the partially generated response after t decoding steps. Generation starts from $Y^{(0)} = ([\mathrm{MASK}], \ldots, [\mathrm{MASK}])$, where $[\mathrm{MASK}]$ is the mask token and L is the allocated generation length Nie et al. (2025). We denote the masked positions by $\mathcal{M}_t = \{i : y_i^{(t)} = [\mathrm{MASK}]\}$ and the already unmasked positions by $\mathcal{U}_t = \{1, \ldots, L\} \setminus \mathcal{M}_t$. Let

V be the vocabulary of non-mask tokens and $p_{\theta,i}(\cdot \mid X, Y^{(t)})$ be the model's predictive distribution at position i, parameterized by θ. Under greedy decoding, the proposed token and its drafting confidence for $i \in \mathcal{M}_t$ are

$$\hat{y}_i^{(t)} = \arg\max_{v \in V} p_{\theta,i}(v \mid X, Y^{(t)}),$$
$$c_i^{(t)} = p_{\theta,i}(\hat{y}_i^{(t)} \mid X, Y^{(t)}). \tag{1}$$

The decoder selects a set $\mathcal{D}_t \subseteq \mathcal{M}_t$ to unmask. Confidence-based schedules select either a prescribed number of the most confident predictions or those above a threshold τ_{d}, with the highest-confidence prediction selected if none passes the threshold Nie et al. (2025); Wu et al. (2025). These predictions share the same input $Y^{(t)}$ and cannot condition on one another's newly proposed values Wu et al. (2025). Standard decoding leaves previously unmasked tokens unchanged. Revocable decoding additionally selects a set $\mathcal{R}_t \subseteq \mathcal{U}_t$ for remasking, allowing earlier predictions to be reconsidered using the current context Hong et al. (2025). The resulting sequence update is

$$y_i^{(t+1)} = \begin{cases} \hat{y}_i^{(t)}, & i \in \mathcal{D}_t, \\ [\mathrm{MASK}], & i \in \mathcal{R}_t, \\ y_i^{(t)}, & \text{otherwise.} \end{cases} \tag{2}$$

The sets $\mathcal{D}_t$ and $\mathcal{R}_t$ are disjoint, so the number of unmasked positions changes by $|\mathcal{D}_t| - |\mathcal{R}_t|$. Setting $\mathcal{R}_t = \varnothing$ recovers standard irreversible decoding. Here, remasking means returning an already committed token to [MASK]; proposals that are not selected from $\mathcal{M}_t$ simply remain masked. A remasked position can later recover either its previous token or a different one.

In semi-autoregressive decoding, the response is partitioned into blocks of length L_b and generated block by block from left to right Nie et al. (2025); Hong et al. (2025). The prompt and completed blocks remain fixed, while future blocks stay masked. Within each active block, the same notation and update rule apply, with $\mathcal{M}_t$ and $\mathcal{U}_t$ restricted to that block. Taking $L_b = L$ recovers full-sequence diffusion decoding.

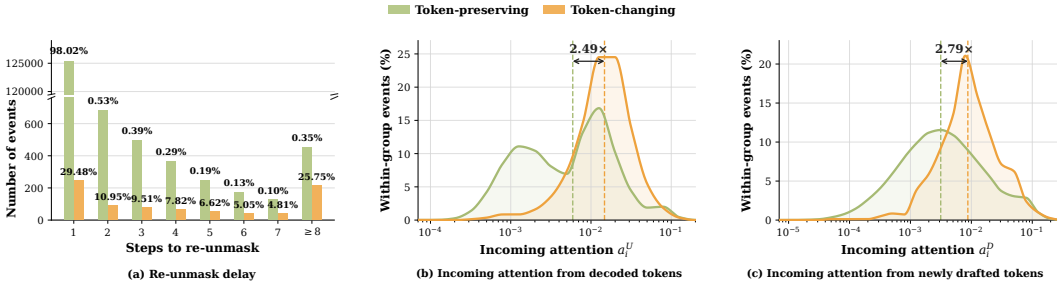

Figure 1: Characterization of remasking behavior under lowest-confidence remasking, aggregated over LLaDA-8B-Base and Dream-Base-7B on GSM8K, MATH-500, MBPP, and HumanEval. (a) Distribution of re-unmask delay for token-preserving and token-changing events. (b) Distribution of incoming attention from already decoded tokens. (c) Distribution of incoming attention from newly drafted tokens. Dashed lines in (b) and (c) indicate the corresponding medians.

3.2 Key Limitation of Immediate Remasking

Existing revocable decoders typically use token uncertainty to decide which committed predictions should be reconsidered. However, uncertainty can fluctuate as the decoding context evolves, and a temporary drop in confidence does not necessarily imply that the token itself needs to change (Figure 1(a)). Consequently, most remasks revoke a token only for the same prediction to be reconstructed shortly afterwards. This suggests that suspicion should trigger further verification rather than immediate revocation.

Most token-preserving remasks are resolved almost immediately: 98.02% recover the original token at the next decoding step (Figure 1(a)). More importantly, tokens that eventually change after remasking exhibit a distinct dependency pattern. Their median incoming attention from already decoded(a_i^U) and newly drafted(a_i^D) tokens is 2.49× and 2.79× higher, respectively, than that of

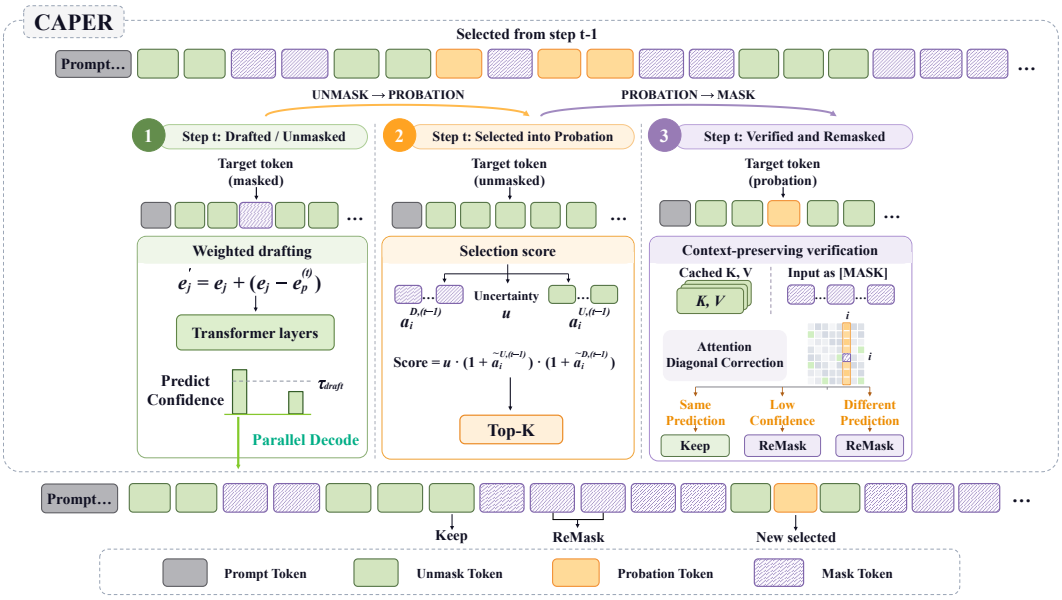

Figure 2: Overview of CAPER. At each decoding step, the model jointly drafts new masked positions and verifies the probation tokens selected in the previous step. The current forward pass further evaluates decoded tokens using assignment uncertainty and incoming attention from already decoded and newly drafted positions, from which the top-K candidates enter probation for the next step when decoding continues. Each probation token is verified for exactly one decoding step and is then either kept or remasked.

token-preserving events (Figure 1(b,c)). Thus, token-changing events tend to occur at positions that are more strongly coupled to the evolving decoding context. Incoming attention therefore provides a complementary signal to token uncertainty, helping distinguish consequential uncertainty from transient confidence fluctuations.

These observations expose a key limitation of immediate confidence-based remasking. Revoking every suspicious token wastes decoding progress on predictions that are likely to be restored, whereas leaving a highly influential suspicious token unchecked allows its potentially unreliable representation to affect subsequent predictions.

4 Method

CAPER introduces an intermediate *probation* state between token commitment and remasking. Rather than immediately revoking a suspicious decoded token, CAPER first places a small set of candidates under one-step verification. At decoding step t, the model jointly drafts masked positions and verifies the probation set $\mathcal{P}_t$ selected at the previous step. The same forward pass also provides the uncertainty and attention statistics used to construct the next probation set $\mathcal{P}_{t+1}$ when decoding continues. Each selected token therefore remains in probation for exactly one decoding step before being either retained or remasked. Figure 2 illustrates the overall procedure.

4.1 Probation Selection

Let $\mathcal{M}_t$ and $\mathcal{U}_t$ denote the masked and decoded positions within the active block at step t, respectively. Tokens currently under probation are excluded from the next selection, yielding the candidate set

$$\mathcal{W}_t = \mathcal{U}_t \setminus \mathcal{P}_t. \tag{3}$$

For masked positions, we follow confidence-based parallel drafting. Predictions whose confidence exceeds τ_{d} are eligible for drafting, subject to a per-step budget B, producing the draft set $\mathcal{D}_t$. If no masked position exceeds the threshold, the most confident prediction is selected.

Probation priority. Token uncertainty alone indicates whether a current assignment is unreliable, but not whether revising it is likely to affect the evolving decoding context. Motivated by the observations in Section 3.2, we combine assignment uncertainty with incoming attention from both already decoded and newly drafted positions.

For each candidate $i \in \mathcal{W}_t$, we measure the uncertainty of its current assignment as

$$u_i^{(t)} = -\log\left(\bar{p}_i^{(t)}(y_i^{(t)}) + \epsilon\right), \tag{4}$$

where $\bar{p}_i^{(t)}$ is the predictive distribution from the current forward pass and ϵ is a small constant for numerical stability. A larger $u_i^{(t)}$ indicates weaker support for the current assignment.

Let $A_{ji}^{(t)}$ denote the last-layer attention from query position j to key position i, averaged across attention heads. We compute the incoming attention from already decoded and newly drafted positions as

$$a_i^{U,(t)} = \frac{\sum_{j \in \mathcal{U}_t \setminus \{i\}} A_{ji}^{(t)}}{|\mathcal{U}_t \setminus \{i\}|}, \qquad a_i^{D,(t)} = \frac{\sum_{j \in \mathcal{D}_t} A_{ji}^{(t)}}{|\mathcal{D}_t|}. \tag{5}$$

An empty source set contributes zero. The first term captures how strongly the existing decoded context depends on token i, while the second measures its interaction with the newly introduced context.

Because the two attention signals can vary in scale across decoding steps, we normalize them independently over the current candidate set:

$$\widetilde{a}_i^{G,(t)} = \frac{a_i^{G,(t)} - \min_{r \in \mathcal{W}_t} a_r^{G,(t)}}{\max_{r \in \mathcal{W}_t} a_r^{G,(t)} - \min_{r \in \mathcal{W}_t} a_r^{G,(t)} + \epsilon}, \qquad G \in \{U, D\}. \tag{6}$$

We leave uncertainty unnormalized and define the probation priority as

$$S_i^{(t)} = u_i^{(t)} \left(1 + \widetilde{a}_i^{U,(t)}\right) \left(1 + \widetilde{a}_i^{D,(t)}\right). \tag{7}$$

Uncertainty provides the base revision risk, while the two attention terms increase the priority of tokens that are more strongly coupled to the evolving context. The additive constants prevent either attention source from suppressing an otherwise uncertain candidate.

At the end of step t, if masked positions remain after the current update, the top-K candidates with the largest $S_i^{(t)}$ form $\mathcal{P}_{t+1}$; otherwise, $\mathcal{P}_{t+1} = \varnothing$. At most $\min(K, |\mathcal{W}_t|)$ candidates are selected. Only statistics from the current forward pass are used; no temporal averaging or multi-step stability history is required. For each selected token, we retain its current assignment, probation score, embedding, and layer-wise key/value states for use in the next step.

4.2 Joint Drafting and Probation Verification

At step t, the probation set $\mathcal{P}_t$ selected in the previous step is verified while the remaining masked positions are drafted in parallel. This requires addressing two issues: unresolved probation assignments should not exert unrestricted influence on new drafts, while masking them for verification should not remove their contextual information from the rest of the sequence.

Priority-weighted drafting. Probation tokens can differ substantially in revision priority. We therefore use their stored scores to determine their relative contribution during concurrent drafting:

$$w_i^{(t)} = \frac{S_i^{(t-1)}}{\sum_{r \in \mathcal{P}_t} S_r^{(t-1)} + \epsilon}, \qquad e_{\mathcal{P}}^{(t)} = \sum_{i \in \mathcal{P}_t} w_i^{(t)} e_i, \tag{8}$$

where e_i is the stored embedding of probation token i.

For each masked position j, we then adjust its input representation as

$$e'_j = e_j + \left(e_j - e_{\mathcal{P}}^{(t)}\right), \qquad j \in \mathcal{M}_t. \tag{9}$$

This moves the drafting representation away from the aggregate representation of unresolved probation tokens, reducing their influence on concurrent predictions before their validity has been established. When $\mathcal{P}_t = \varnothing$, the original representations are used unchanged.

Context-preserving verification. The probation set $\mathcal{P}_t$ is selected at the end of step $t-1$ and verified during step t. For each $i \in \mathcal{P}_t$, we keep its current assignment $y_i^{(t)}$ as the value to be verified, while temporarily replacing its input at step t with [MASK]. When the token enters probation at step $t-1$, its layer-wise key/value states are cached for use in the next decoding step.

For one layer and attention head, let $(\mathbf{k}_j^{(t)}, \mathbf{v}_j^{(t)})$ denote the key and value computed in the current step after the probation positions are masked. Let $(\mathbf{k}_{j,\mathrm{c}}^{(t-1)}, \mathbf{v}_{j,\mathrm{c}}^{(t-1)})$ denote the cached pair stored when $\mathcal{P}_t$ was formed at the end of step $t-1$. Omitting layer and head indices, we construct the shared attention memory at step t as

$$(\overline{\mathbf{k}}_j^{(t)}, \overline{\mathbf{v}}_j^{(t)}) = \begin{cases} (\mathbf{k}_{j,\mathrm{c}}^{(t-1)}, \mathbf{v}_{j,\mathrm{c}}^{(t-1)}), & j \in \mathcal{P}_t, \\ (\mathbf{k}_j^{(t)}, \mathbf{v}_j^{(t)}), & \text{otherwise.} \end{cases} \tag{10}$$

Thus, although probation positions are masked in the current input, non-verification queries still attend to their cached representations from step $t-1$, preserving the context available before verification.

For a verification query $i \in \mathcal{P}_t$, however, retaining its own cached entry would expose the assignment $y_i^{(t)}$ being checked. We therefore replace only the self-entry at $j = i$ with the mask-derived pair $(\mathbf{k}_i^{(t)}, \mathbf{v}_i^{(t)})$, while leaving all other entries in the shared memory unchanged. Since changing the diagonal attention score also changes the softmax normalization of the entire row, we correct the row normalization and replace the cached self-value contribution with the mask-derived contribution. This correction is applied independently to every probation query at each layer and head. Consequently, each probation token is verified without direct self-conditioning, while non-verification queries retain the original cached context. Drafting and all probation checks therefore share a single model forward pass.

4.3 Probation Resolution

After the step-t forward pass, every token in $\mathcal{P}_t$ leaves probation. Let $\hat{y}_{i,\mathrm{v}}^{(t)}$ denote the top-1 prediction produced under the verification view for position i, and let $c_{i,\mathrm{v}}^{(t)}$ denote its corresponding confidence. We compare this prediction with $y_i^{(t)}$, the assignment carried into step t, and resolve the token as

$$\mathrm{Decision}_t(i) = \begin{cases} \mathrm{KEEP}, & \hat{y}_{i,\mathrm{v}}^{(t)} = y_i^{(t)} \text{ and } c_{i,\mathrm{v}}^{(t)} > \tau_{\mathrm{d}}, \\ \mathrm{REMASK}, & \text{otherwise.} \end{cases} \tag{11}$$

A KEEP decision preserves $y_i^{(t)}$ and returns position i to the ordinary decoded state. A REMASK decision resets the position to [MASK], allowing it to be reconsidered under future context. A different verification prediction is not committed directly: disagreement indicates that the current assignment is unreliable, but does not by itself establish that the alternative prediction is sufficiently reliable for immediate replacement.

Positions assigned REMASK are reset to [MASK], while newly drafted tokens in $\mathcal{D}_t$ are committed to the sequence. Meanwhile, the uncertainty and attention statistics produced by the same step-t forward pass are used to construct $\mathcal{P}_{t+1}$. If masked positions remain, $\mathcal{P}_{t+1}$ is verified at step $t+1$; otherwise, no new probation set is created and the active block terminates.

5 Experiments

5.1 Experimental Setup

Implementation details. We evaluate CAPER on LLaDA-8B-Base Nie et al. (2025) and Dream-Base-7B Ye et al. (2025), with all model weights frozen. We use greedy decoding with generation lengths of 256 and 512 and a block length of 64. The drafting and verification threshold is $\tau_{\mathrm{d}} = 0.8$, with per-step budgets of $B = 15$ for drafting and $K = 2$ for probation. All experiments run on four NVIDIA A100 80GB GPUs with bfloat16 precision, a per-device batch size of one, and a random seed of 42.

Table 1: Generation quality and decoding efficiency across four benchmarks and two generation lengths. Best results in each setting are highlighted in **orange**.

Model	Len	Method	HumanEval			MBPP			GSM8K			MATH-500		
			Score↑	NFE↓	Speedup↑	Score↑	NFE↓	Speedup↑	Acc.↑	NFE↓	Speedup↑	Acc.↑	NFE↓	Speedup↑
LLaDA-8B-Base	256	Baseline	37.20	256	1.00×	39.20	256	1.00×	**69.67**	256	1.00×	26.80	256	1.00×
		Saber	35.37	172	1.37×	39.80	125	1.94×	**69.67**	152	1.63×	24.40	149	1.62×
		WINO	**39.63**	127	1.22×	40.20	81	1.80×	67.40	95	1.43×	27.60	99	1.52×
		CAPER	**39.63**	**75**	**2.24×**	**41.20**	**79**	**2.27×**	**69.67**	**71**	**2.52×**	**30.60**	**93**	**2.07×**
	512	Baseline	37.20	512	1.00×	40.20	512	1.00×	69.83	512	1.00×	30.60	512	1.00×
		Saber	39.02	289	2.15×	37.60	204	2.98×	69.45	251	2.38×	30.40	220	**3.58×**
		WINO	39.02	244	1.23×	40.20	125	2.07×	68.99	168	1.49×	31.00	**140**	2.18×
		CAPER	**39.63**	**76**	**4.81×**	**41.20**	**122**	**3.83×**	**70.66**	**71**	**5.69×**	**31.20**	151	3.20×
Dream-7B-Base	256	Baseline	53.05	256	1.00×	54.40	256	1.00×	74.91	256	1.00×	36.80	256	1.00×
		Saber	54.27	179	1.38×	52.60	125	1.96×	73.62	180	1.38×	33.80	130	1.91×
		WINO	53.05	142	1.64×	52.40	82	2.59×	68.69	134	1.71×	33.40	120	1.85×
		CAPER	**57.32**	**70**	**2.97×**	**55.80**	**81**	**2.63×**	**76.80**	**66**	**3.51×**	**38.40**	**82**	**2.71×**
	512	Baseline	53.66	512	1.00×	54.20	512	1.00×	75.82	512	1.00×	38.00	512	1.00×
		Saber	**54.88**	271	2.08×	54.80	175	2.85×	**76.80**	283	2.23×	38.20	188	2.59×
		WINO	**54.88**	201	2.18×	52.40	116	4.17×	71.87	206	2.94×	37.40	125	3.89×
		CAPER	**54.88**	**54**	**7.22×**	**56.80**	**108**	**4.21×**	**76.80**	**48**	**12.47×**	**38.40**	**115**	**4.46×**

Datasets. We use HumanEval Chen et al. (2021) and MBPP Austin et al. (2021b) for code generation, and GSM8K Cobbe et al. (2021) and MATH-500 Hendrycks et al. (2021); Lightman et al. (2023) for mathematical reasoning. We evaluate the test splits using zero-shot prompting for HumanEval, 3-shot for MBPP, 5-shot for GSM8K, and 4-shot for MATH-500.
Metrics. We report pass@1 for HumanEval and MBPP and accuracy for GSM8K and MATH-500. Decoding efficiency is measured by the mean number of model forward passes per example (NFE). Speedup is computed from end-to-end decoding wall-clock time relative to the corresponding irreversible baseline. To characterize revision behavior, we additionally report the total number of remasks (#Remask) and effective remask rate (Eff. Rate). A remask is effective if the first subsequent unmasking changes the original token, and Eff. Rate is the fraction of such events among all remasks.
Baselines. We compare with irreversible highest-confidence decoding, WINO Hong et al. (2025), and Saber Dong et al. (2025), using the same prompts and evaluation protocols. The irreversible baseline uses as many denoising steps as the generation length. For Dream, we port the released WINO and Saber algorithms to its decoding framework.

5.2 Main Results

Generation quality. CAPER achieves the highest or tied-highest score in all 16 settings (Table 1). On Dream at length 256, HumanEval pass@1 improves from 53.05% under irreversible decoding to 57.32%, while on LLaDA at length 512, GSM8K accuracy increases from 69.83% to 70.66%. The consistent results across code generation and mathematical reasoning show that selective revision preserves, and in several cases improves, generation quality.
Decoding efficiency. CAPER achieves speedups of up to 5.69× on LLaDA and 12.47× on Dream, and is the fastest method in 15 of the 16 settings. On Dream's GSM8K benchmark at length 512, it matches Saber's 76.80% accuracy while reducing NFE from 283 to 48. Its advantage is particularly pronounced at length 512, where larger speedups are observed across all four benchmarks on both models. The only exception among the revocable baselines is LLaDA's MATH-500 setting at length 512, where Saber is faster but CAPER achieves higher accuracy. Overall, CAPER provides a stronger quality–efficiency trade-off across models, tasks, and generation lengths.

5.3 Remask Effectiveness

Selective revision. CAPER concentrates revision on a substantially smaller set of candidates with much higher revision utility. Its effective remask rate reaches 26.37% on LLaDA and 39.90% on Dream, compared with 8.61% and 3.19% for WINO and 1.05% and 0.77% for Saber, respectively (Figure 3(a)). Meanwhile, CAPER performs only 484 and 336 remasks on the two models, versus 19.16K and 33.00K for WINO and 164K and 166K for Saber (Figure 3(b)). Together with the

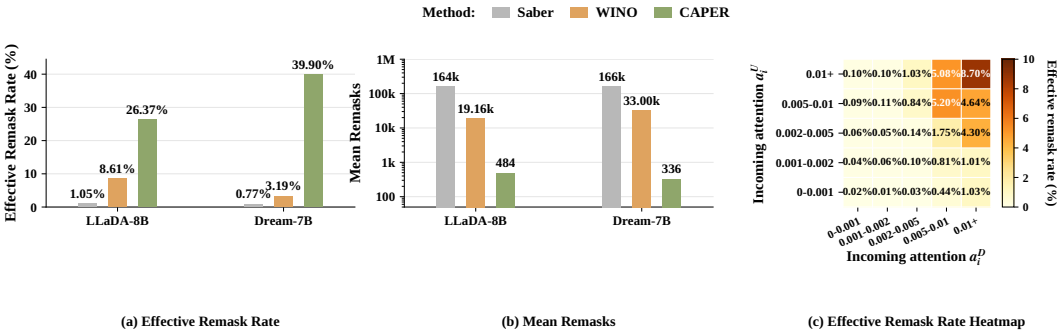

Figure 3: Revision behavior and attention-conditioned remask utility on LLaDA-8B-Base and Dream-Base-7B at sequence length 256, evaluated across GSM8K, MATH-500, MBPP, and HumanEval. (a) Mean effective remask rate of Saber, WINO, and CAPER across the four benchmarks. (b) Mean number of remasks per benchmark on a logarithmic scale. (c) Mean effective remask rate conditioned on incoming attention from a_i^U and a_i^D, averaged over the two models and four benchmarks.

generation results in Table 1, this indicates that strong generation quality does not require frequent revision; selectively targeting candidates with higher revision utility is substantially more effective.

Attention and revision utility. Incoming attention provides a strong signal for identifying these actionable candidates (Figure 3(c)). Effective remasks are rare when both a_i^U and a_i^D are weak, but become substantially more frequent as the two signals increase. Importantly, large a_i^D alone does not consistently correspond to high revision utility when a_i^U remains small, indicating that attention from newly drafted tokens and already decoded context provides complementary evidence. This relationship supports jointly incorporating the two signals when determining which uncertain tokens enter probation.

Table 2: Component ablations of CAPER on LLaDA-8B-Base at sequence length 256. Normalized performance is computed from GPU-time and normalized to full CAPER (1.0×).

Dataset	Variant	Score (%) ↑	Norm. Perf. ↑	#Remask ↓	Eff. Rate (%) ↑
HumanEval	**CAPER**	39.63	1.000×	75	30.67
	w/o Guided Drafting	36.59	0.831×	86	11.63
	w/o Context Selection	37.80	0.532×	31	16.13
	w/o Verification	36.59	0.373×	4,083	3.13
MBPP	**CAPER**	41.20	1.000×	227	15.86
	w/o Guided Drafting	39.20	0.895×	225	11.56
	w/o Context Selection	39.40	0.590×	77	7.79
	w/o Verification	40.00	0.383×	15,375	4.36
MATH-500	**CAPER**	30.60	1.000×	519	30.25
	w/o Guided Drafting	30.40	0.812×	318	13.21
	w/o Context Selection	27.80	0.523×	126	8.73
	w/o Verification	29.40	0.333×	17,145	2.96
GSM8K	**CAPER**	69.67	1.000×	1,115	28.70
	w/o Guided Drafting	68.76	0.842×	706	23.37
	w/o Context Selection	69.60	0.644×	250	18.40
	w/o Verification	69.29	0.363×	31,960	2.26

5.4 Ablation Study

Ablation protocol. Each variant removes only the indicated component while keeping all other decoding settings identical to full CAPER. *w/o Guided Drafting* removes the input-representation adjustment in Eq. 9, so masked tokens are drafted from their original input representations. *w/o Context Selection* removes the incoming-attention terms from the probation priority and selects the same number K of candidates using uncertainty $u_i^{(t)}$ alone. *w/o Verification* immediately remasks the selected candidates instead of placing them under one-step context-preserving verification.

Guided drafting. Removing guided drafting lowers accuracy on all four benchmarks, with the largest drop on HumanEval, from 39.63% to 36.59% (Table 2). Because guided drafting modifies token predictions rather than directly determining probation or verification, its removal incurs the smallest efficiency loss among the three ablations, retaining 0.812–$0.895\times$ normalized performance. Nevertheless, the average effective remask rate decreases to 14.94%, indicating that shaping the drafting context also improves the quality of subsequent revision candidates.

Context selection and verification. The two components that directly control revision play complementary roles. Removing context selection lowers the average effective remask rate from 26.37% to 12.76% and decreases accuracy on all four benchmarks despite producing fewer remasks. Removing verification has an even larger efficiency cost: the effective rate falls to 3.18%, remask counts increase by 28.7–67.7×, and normalized performance drops to 0.333–$0.383\times$, without improving accuracy. These results show that efficient revocable decoding requires both identifying promising revision candidates and suppressing unnecessary remasks.

5.5 Sensitivity to the Probation Budget

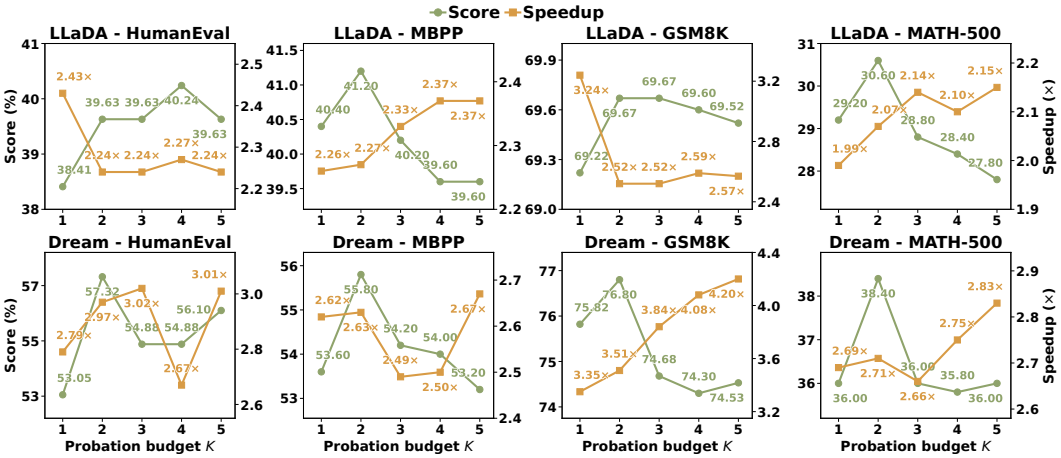

Figure 4: Sensitivity to the probation budget $K \in \{1, \ldots, 5\}$ on LLaDA-8B-Base and Dream-7B-Base at sequence length 256 across four benchmarks.

Accuracy. Generation quality generally improves as K initially increases and then declines at larger budgets (Figure 4). A small budget can exclude candidates that would benefit from revision, whereas an overly large budget increasingly admits lower-priority tokens whose remasking provides little utility. For example, Dream's GSM8K accuracy rises from 75.82% at $K = 1$ to 76.80% at $K = 2$, but decreases to 74.53% at $K = 5$. The default $K = 2$ achieves the highest or tied-highest score in seven of the eight settings, with LLaDA's HumanEval as the only case that peaks at $K = 4$.

Efficiency. Efficiency is comparatively insensitive to K because probation tokens are verified together with drafting in the same forward pass rather than through separate verification passes. Consequently, increasing the probation budget does not increase NFE proportionally to the number of selected tokens. For example, LLaDA's HumanEval speedup remains between $2.24\times$ and $2.27\times$ for $K = 2$–5, although changes in the resulting decoding trajectory can still affect total runtime. Taken together, the quality and efficiency trends support $K = 2$ as the default operating point.

6 Conclusion

In this work, we introduced CAPER, a training-free framework for efficient revocable diffusion decoding. By placing selected tokens into a one-step probation state and prioritizing verification using assignment uncertainty and incoming attention, CAPER avoids unnecessary remasking while retaining the corrective ability of revocable decoding. Experiments on two diffusion language models across code generation and mathematical reasoning demonstrate consistent reductions in decoding steps and inference time while preserving or improving task performance, resulting a better quality–efficiency trade-off.

AI USE STATEMENT

(This section is **required** and does not count toward the page limit.)
In this work, we used generative AI tools for [figure generation]. We have not used generative AI tools for [data generation], and [data analysis] is not applicable to this work. Additionally, we used generative AI tools for [code generation]. We have reviewed all AI-assisted work. [LLM-generated code was verified and tested for correctness by 2 authors]. We take responsibility for the final content of this work, including text, claims or artifacts produced with the aid of generative AI.
See the ICLR 2027 AI Policy for Authors for more details. This statement should not be more than 1 page.

ETHICS STATEMENT

(This section is **recommended** and does not count toward the page limit.)
If authors feel that their paper submission raises questions regarding the Code of Ethics, they are encouraged to include a paragraph of Ethics Statement (at the end of the main text before references) to address potential concerns where appropriate. Topics include, but are not limited to, studies that involve human subjects, practices to data set releases, potentially harmful insights, methodologies and applications, potential conflicts of interest and sponsorship, discrimination/bias/fairness concerns, privacy and security issues, legal compliance, and research integrity issues (e.g., IRB, documentation, research ethics). This statement should not be more than 1 page.

REPRODUCIBILITY STATEMENT

(This section is **recommended** and does not count toward the page limit.)
It is important that the work published in ICLR is reproducible. Authors are strongly encouraged to include a paragraph-long Reproducibility Statement at the end of the main text (before references) to discuss the efforts that have been made to ensure reproducibility. This paragraph should not itself describe details needed for reproducing the results, but rather reference the parts of the main paper, appendix, and supplemental materials that will help with reproducibility. For example, for novel models or algorithms, a link to an anonymous downloadable source code can be submitted as supplementary materials; for theoretical results, clear explanations of any assumptions and a complete proof of the claims can be included in the appendix; for any datasets used in the experiments, a complete description of the data processing steps can be provided in the supplementary materials. Each of the above are examples of things that can be referenced in the reproducibility statement.

REFERENCES

Jacob Austin, Daniel D. Johnson, Jonathan Ho, Daniel Tarlow, and Rianne van den Berg. Structured denoising diffusion models in discrete state-spaces. *arXiv preprint arXiv:2107.03006*, 2021a. URL https://arxiv.org/abs/2107.03006.

Jacob Austin, Augustus Odena, Maxwell Nye, Maarten Bosma, Henryk Michalewski, David Dohan, Ellen Jiang, Carrie Cai, Michael Terry, Quoc Le, and Charles Sutton. Program synthesis with large language models. *arXiv preprint arXiv:2108.07732*, 2021b. URL https://arxiv.org/abs/2108.07732.

Heli Ben-Hamu, Itai Gat, Daniel Severo, Niklas Nolte, and Brian Karrer. Accelerated sampling from masked diffusion models via entropy bounded unmasking. *arXiv preprint arXiv:2505.24857*, 2025. URL https://arxiv.org/abs/2505.24857.

Tom B. Brown, Benjamin Mann, Nick Ryder, Melanie Subbiah, Jared Kaplan, Prafulla Dhariwal, Arvind Neelakantan, Pranav Shyam, Girish Sastry, Amanda Askell, Sandhini Agarwal, Ariel Herbert-Voss, Gretchen Krueger, Tom Henighan, Rewon Child, Aditya Ramesh, Daniel M. Ziegler, Jeffrey Wu, Clemens Winter, Christopher Hesse, Mark Chen, Eric Sigler, Mateusz Litwin, Scott Gray, Benjamin Chess, Jack Clark, Christopher Berner, Sam McCandlish, Alec Radford, Ilya Sutskever, and Dario Amodei. Language models are few-shot learners. *arXiv preprint arXiv:2005.14165*, 2020. URL https://arxiv.org/abs/2005.14165.

Mark Chen, Jerry Tworek, Heewoo Jun, Qiming Yuan, Henrique Ponde de Oliveira Pinto, Jared Kaplan, Harri Edwards, Yuri Burda, Nicholas Joseph, Greg Brockman, Alex Ray, Raul Puri, Gretchen Krueger, Michael Petrov, Heidy Khlaaf, Girish Sastry, Pamela Mishkin, Brooke Chan, Scott Gray, Nick Ryder, Mikhail Pavlov, Alethea Power, Lukasz Kaiser, Mohammad Bavarian,

Clemens Winter, Philippe Tillet, Felipe Petroski Such, Dave Cummings, Matthias Plappert, Fotios Chantzis, Elizabeth Barnes, Ariel Herbert-Voss, William Hebgen Guss, Alex Nichol, Alex Paino, Nikolas Tezak, Jie Tang, Igor Babuschkin, Suchir Balaji, Shantanu Jain, William Saunders, Christopher Hesse, Andrew N. Carr, Jan Leike, Josh Achiam, Vedant Misra, Evan Morikawa, Alec Radford, Matthew Knight, Miles Brundage, Mira Murati, Katie Mayer, Peter Welinder, Bob McGrew, Dario Amodei, Sam McCandlish, Ilya Sutskever, and Wojciech Zaremba. Evaluating large language models trained on code. *arXiv preprint arXiv:2107.03374*, 2021. URL https://arxiv.org/abs/2107.03374.

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

Yihong Dong, Zhaoyu Ma, Xue Jiang, Zhiyuan Fan, Jiaru Qian, Yongmin Li, Jianha Xiao, Zhi Jin, Rongyu Cao, Binhua Li, Fei Huang, Yongbin Li, and Ge Li. Saber: An efficient sampling with adaptive acceleration and backtracking enhanced remasking for diffusion language model. *arXiv preprint arXiv:2510.18165*, 2025. URL https://arxiv.org/abs/2510.18165.

Dan Hendrycks, Collin Burns, Saurav Kadavath, Akul Arora, Steven Basart, Eric Tang, Dawn Song, and Jacob Steinhardt. Measuring mathematical problem solving with the MATH dataset. *arXiv preprint arXiv:2103.03874*, 2021. URL https://arxiv.org/abs/2103.03874.

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. *arXiv preprint arXiv:2507.18578*, 2025. URL https://arxiv.org/abs/2507.18578.

Daniel Israel, Guy Van den Broeck, and Aditya Grover. Accelerating diffusion LLMs via adaptive parallel decoding. *arXiv preprint arXiv:2506.00413*, 2025. URL https://arxiv.org/abs/2506.00413.

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. *arXiv preprint arXiv:2305.20050*, 2023. URL https://arxiv.org/abs/2305.20050.

Aaron Lou, Chenlin Meng, and Stefano Ermon. Discrete diffusion modeling by estimating the ratios of the data distribution. *Proceedings of Machine Learning Research*, 235:32819–32848, 2024. URL https://proceedings.mlr.press/v235/lou24a.html.

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

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. *arXiv preprint arXiv:2502.09992*, 2025. URL https://arxiv.org/abs/2502.09992.

Subham Sekhar Sahoo, Marianne Arriola, Yair Schiff, Aaron Gokaslan, Edgar Marroquin, Justin T. Chiu, Alexander Rush, and Volodymyr Kuleshov. Simple and effective masked diffusion language models. *arXiv preprint arXiv:2406.07524*, 2024. URL https://arxiv.org/abs/2406.07524.

Ashish Vaswani, Noam Shazeer, Niki Parmar, Jakob Uszkoreit, Llion Jones, Aidan N. Gomez, Lukasz Kaiser, and Illia Polosukhin. Attention is all you need. *arXiv preprint arXiv:1706.03762*, 2017. URL https://arxiv.org/abs/1706.03762.

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

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. *arXiv preprint arXiv:2505.22618*, 2025. URL https://arxiv.org/abs/2505.22618.

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