DecoCal: Decoding with Calibration in Diffusion Large Language Models

Fan Xu and Huixuan Zhang and Xiaojun Wan
Wangxuan Institute of Computer Technology, Peking University
{xufan2000,wanxiaojun}@pku.edu.cn, zhanghuixuan@stu.pku.edu.cn

Abstract

Diffusion Large Language Models (DLLMs) generate text via iterative masked-token denoising, supporting parallel prediction and bidirectional context modeling. Despite these advantages, decoding remains challenging: many tokens appear predictable early, yet single-step predictions are often unstable, exhibiting temporal oscillations or overconfidence, making it difficult to determine which tokens can be safely committed. To address these challenges, we propose **DecoCal**[1], a Decoding framework that explicitly performs Calibration of token-level confidence across diffusion steps and leverages the calibrated results to guide decoding decisions. Specifically, DecoCal aggregates historical predictions to maintain calibrated confidence, triggering unmasking only when a token is sufficiently stable, while a remasking mechanism allows revision of premature commitments. This calibration-based design enables early decoding of reliably converged tokens while deferring or correcting unstable ones, balancing reliability and speed. Experiments on multiple DLLMs and benchmarks show that DecoCal improves generation accuracy compared to existing strategies. Our results highlight the importance of temporal calibration in unlocking the full potential of diffusion-based language generation.

1 Introduction

Diffusion Large Language Models (DLLMs) have recently emerged as a promising text generation paradigm (Yang et al., 2023; Li et al., 2025b), with representative models such as LLaDA (Nie et al., 2025; Zhu et al., 2025a) and Dream (Ye et al., 2025) achieving strong performance on diverse natural language generation tasks. Unlike mainstream autoregressive language models, DLLMs frame text generation as an iterative token denoising process, which confers three key advantages: high parallelism, native bidirectional context modeling for joint left-right reasoning, and fine-grained generation control.

Despite these advantages, DLLMs are still at an early stage of development, and their decoding procedures remain a major bottleneck (Li et al., 2025b). In contrast to autoregressive decoding, where tokens are committed sequentially (Sutskever et al., 2014), DLLMs repeatedly revise the sequence over multiple diffusion steps. This iterative process introduces a fundamental challenge: *decoding is no longer a pure sampling problem, but a decision-making problem*. Specifically, the decoding method must determine *when* a token prediction is sufficiently reliable to be unmasked and *whether* previously made decisions should be revised in light of new evidence.

Default decoding strategies for DLLMs rely on instantaneous model predictions at each diffusion step. A common approach is to sample or select tokens based on the current probability distribution and unmask a fixed number of positions according to a predefined schedule (Nie et al., 2025; Ye et al., 2025). While simple, such methods are highly sensitive to transient noise and overconfidence in early diffusion steps, often leading to premature or irreversible errors.

Recent efforts to improve DLLM decoding have primarily focused on mitigating the irreversibility of early token commitments and enhancing the robustness of iterative refinement. A representative line of work introduces explicit mechanisms for remasking or revision, allowing previously unmasked tokens to be corrected as more contextual information becomes available (He et al., 2025; Wang et al., 2025a). These approaches improve flexibility over fixed unmasking schedules, but typically rely on step-wise confidence estimates or heuristic criteria, which remain sensitive to transient noise and unstable predictions during early diffusion stages.

[1] https://github.com/pku0xff/DecoCal

Proceedings of the 64th Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers), pages 11866–11880

Despite these advances, diffusion decoding still faces intrinsic challenges. Many tokens in a sequence become sufficiently predictable early in the diffusion process (Li et al., 2025a), suggesting that they could be decoded quickly to improve efficiency. However, single-step predictions are often unstable (Wang et al., 2025b), so a high probability at a given step does not necessarily indicate that a token can be safely decoded. Tokens may appear confident due to temporary overconfidence or oscillatory behavior, making it difficult to identify which positions are reliably predictable. Early overconfident predictions can propagate errors through contextual feedback, leading to local optima. Together, these observations highlight the importance of explicitly modeling the uncertainty of token predictions over time, so that reliably converged tokens can be decoded early while unstable ones are deferred or corrected in later steps.

To address these challenges, we propose **DecoCal**, a decoding framework that explicitly models the uncertainty of token predictions throughout the diffusion process. Rather than relying solely on instantaneous probabilities, DecoCal aggregates historical evidence across diffusion steps to maintain a calibrated confidence. Unmasking decisions are then triggered only when a token's confidence surpasses a predefined threshold, allowing reliably converged tokens to be decoded early, while unstable tokens are deferred. At the same time, a remasking mechanism enables the correction of premature or overconfident commitments as additional evidence becomes available. By explicitly integrating temporal calibration and corrective revision into the decoding process, DecoCal provides a principled framework that improves reliability in diffusion-based text generation.

Our contributions are summarized as follows:

- We propose **DecoCal**, a decoding framework that explicitly aggregates historical evidence across diffusion steps to maintain calibrated, token-level confidence, allowing reliably converged tokens to be decoded early while deferring unstable ones.
- DecoCal integrates confidence-driven unmasking and remasking, enabling principled decisions on when to commit tokens and when to revise previous commitments as new evidence emerges.
- Extensive experiments on multiple models and benchmarks demonstrate that DecoCal improves generation accuracy in most cases, validating the benefits of confidence-aware decoding.

2 Preliminaries

2.1 Diffusion-Based Decoding for Language Models

Current DLLMs commonly generate text through an iterative denoising process over discrete tokens. Given a prompt, the generation starts from a fully masked sequence of length L

$$\mathbf{x}_0 = [\texttt{[MASK]}, \ldots, \texttt{[MASK]}],$$

and proceeds for T diffusion steps. At each step $t \in \{1, \ldots, T\}$, the model $\mathcal{M}$ predicts a categorical distribution over the vocabulary $\mathcal{V}$:

$$\mathbf{p}_t = \mathrm{softmax}(\mathcal{M}(\text{prompt}, \mathbf{x}_{t-1})).$$

Unlike autoregressive models, DLLMs predict token distributions at all positions in parallel. Decoding is performed by progressively replacing `[MASK]` tokens with sampled or selected tokens across decoding iterations, producing a sequence of intermediate states $\{\mathbf{x}_t\}_{t=1}^{T}$. Once a position is unmasked, it may remain fixed or be remasked in subsequent steps, depending on the decoding strategy.

The decoding process terminates when all positions are unmasked or when a predefined stopping criterion is met. The final generated output is defined as the terminal sequence

$$\mathbf{x}^* = \mathbf{x}_{t^\dagger},$$

where $t^\dagger \leq T$ denotes the stopping step of the decoding procedure.

2.2 Decoding Policies and Unmasking Decisions

The decoding process in DLLMs is not fully specified by the model itself, but by an external policy that determines how predictions $\mathbf{p}_t$ are converted into token assignments. Formally, at each step t, a decoding policy selects a subset of positions $\mathcal{U}_t \subseteq \{1, \ldots, L\}$ to unmask and assigns tokens to these positions based on $\mathbf{p}_t$. This yields the updated sequence:

$$x_t^{(i)} = \begin{cases} \hat{x}_t^{(i)}, & i \in \mathcal{U}_t, \\ x_{t-1}^{(i)}, & \text{otherwise}, \end{cases}$$

where $\hat{x}_t^{(i)}$ is obtained by selecting from $\mathbf{p}_t^{(i)}$.

Common decoding strategies rely on predefined schedules or instantaneous confidence measures,

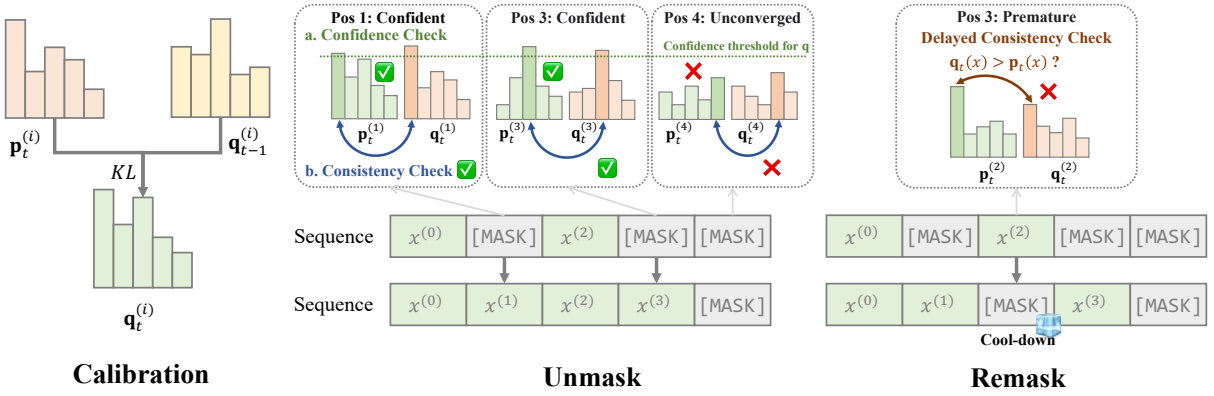

Figure 1: Overview of DecoCal: a calibration-guided decoding framework for DLLMs. It comprises three core stages: (1) Calibration, where token-level confidence is refined via KL-weighted aggregation across steps; (2) Unmask, where tokens are committed only if they pass both confidence and consistency checks; and (3) Remask, which remasks premature or inconsistent predictions with a cool-down mechanism.

such as maximum probability (Nie et al., 2025) or entropy (Ye et al., 2025), to determine $\mathcal{U}_t$. These strategies are typically short-sighted, as decisions are made solely based on the current step's predictions.

3 Method

3.1 Overview of DecoCal

Decoding in diffusion language models remains challenging due to unstable single-step predictions and premature overconfident commitments, leading to suboptimal results. Our goal is to stabilize token-level confidence over time, decode reliably converged tokens early, and allow corrective revision when predictions are inconsistent.

Therefore, we propose **DecoCal**, a Decoding framework that performs Calibration-guided unmasking and remasking for DLLMs. At a high level, DecoCal maintains a calibrated confidence distribution for each position by aggregating model predictions across diffusion steps. This calibrated confidence distribution provides the basis for subsequent decoding decisions, allowing tokens to be committed only when their reliability is sufficiently supported by accumulated evidence, while earlier commitments remain revisable if later observations indicate inconsistency.

Figure 1 presents the core workflow of DecoCal. At each step, it predicts current token probabilities, calibrates them using a KL-weighted moving average, selectively remasks inconsistent tokens and unmasks confident stable ones. Moreover, a fallback mechanism is designed to ensure decoding progress.

DecoCal exhibits several desirable properties:

- **Stability:** Historical confidence aggregation mitigates overconfidence and transient noise in single-step predictions.
- **Correctability:** Remasking enables revision of inconsistent token commitments.
- **Model-Agnosticism:** DecoCal can be applied to any masked diffusion language model.

Together, these properties make DecoCal an effective and robust decoding strategy for diffusion-based text generation.We present the pseudocode in Algorithm 1 in the Appendix. Below we describe its key components in detail.

3.2 Calibration via Historical Evidence Aggregation

DecoCal maintains a calibrated confidence distribution $\mathbf{q}$ that aggregates historical evidence across denoising steps. At each diffusion step t, given the current partially masked sequence $\mathbf{x}_{t-1}$, the model $\mathcal{M}$ first produces a step-wise prediction:

$$\mathbf{p}_t = \mathrm{softmax}(\mathcal{M}(\mathrm{prompt}, \mathbf{x}_{t-1})). \quad (1)$$

DecoCal calibrates the current prediction against historical evidence by adaptively updating $\mathbf{q}$. Specifically, if $\mathbf{q}$ is uninitialized, it is set to the first prediction: $\mathbf{q}_1 = \mathbf{p}_1$. Otherwise, we compute a calibration weight $\mathbf{w}_t \in (0,1)^L$ based on the divergence between the current prediction $\mathbf{p}_t$ and the maintained confidence distribution $\mathbf{q}$:

$$\mathbf{w}_t = \sigma(\gamma \cdot \mathrm{KL}(\mathbf{p}_t \,\|\, \mathbf{q}_{t-1}))\,, \quad (2)$$

where $\sigma(\cdot)$ is the sigmoid function and $\gamma > 0$ controls the sensitivity to distributional change.

The KL-adaptive weighting is motivated by balancing two scenarios of substantial distributional change: (a) The model is refining its prediction toward a more accurate and complete estimate; (b) The model exhibits overconfidence, potentially overriding previously correct information. Rather than fully trusting either the current step or the historical distribution, the mechanism performs a calibrated aggregation that remains sensitive to instability while avoiding abrupt dominance of noisy updates.

The calibrated confidence is then updated via a weighted geometric aggregation:

$$\mathbf{q}_t = \mathrm{softmax}\Big((1 - \mathbf{w}_t) \odot \log \mathbf{q}_{t-1} + \mathbf{w}_t \odot \log \mathbf{p}_t\Big) \tag{3}$$

where $\odot$ denotes element-wise multiplication with automatic broadcasting along the vocabulary dimension. As a result, $\mathbf{q}_t$ serves as a temporally smoothed and better-calibrated confidence estimate, which is subsequently used to guide remasking and unmasking decisions.

3.3 Calibration-Guided Unmasking

DecoCal determines whether a current-preferred token should be committed based on a calibration-guided criterion that explicitly distinguishes *instantaneous model confidence* from the *calibrated confidence*. At the start of diffusion step t, the sequence is $\mathbf{x}_t := \mathbf{x}_{t-1}$. Given the current prediction $\mathbf{p}_t$, we first identify the most likely token at each position i and define the corresponding instantaneous confidence:

$$v_t^{(i)} = \arg\max_{v \in \mathcal{V}} \mathbf{p}_t^{(i)}(v), \quad c_t^{(i)} = \mathbf{p}_t^{(i)}\left(v_t^{(i)}\right). \tag{4}$$

DecoCal does not commit tokens solely based on $c_t^{(i)}$. Instead, unmasking decisions are guided by the maintained calibrated confidence distribution $\mathbf{q}$. For each position i, we define the most likely token under the calibrated confidence and the corresponding confidence score:

$$v_t^{\star(i)} = \arg\max_{v \in \mathcal{V}} \mathbf{q}_t^{(i)}(v), \quad \tilde{c}_t^{(i)} = \mathbf{q}_t^{(i)}\left(v_t^{\star(i)}\right). \tag{5}$$

A masked position i is eligible for unmasking only if it satisfies the following conditions:

- **Confidence threshold:** the calibrated confidence exceeds a predefined threshold,

$$\tilde{c}_t^{(i)} \geq A; \tag{6}$$

- **Prediction consistency:** the instantaneous prediction agrees with the calibrated confidence,

$$v_t^{(i)} = v_t^{\star(i)}. \tag{7}$$

In addition, DecoCal enforces a **cool-down** constraint to prevent rapid oscillations: a position can only be unmasked if it has remained masked for at least C diffusion steps since its last remasking.

When all conditions are satisfied, the decoding algorithm commits to the token by setting

$$x_t^{(i)} := v_t^{(i)}. \tag{8}$$

3.4 Calibration-Guided Remasking

Although a token may be committed at an early diffusion step, subsequent denoising iterations can reveal that this early commitment is unreliable. As more contextual information is incorporated, the model's assessment may evolve, and tokens that initially appeared confident may no longer be supported by later evidence. Such delayed inconsistency motivates the need for a corrective revision mechanism during decoding.

DecoCal addresses this issue through calibration-guided remasking. For an already unmasked position i with committed token $x_t^{(i)} \neq \texttt{[MASK]}$, remasking is triggered when the instantaneous prediction $\mathbf{p}_t^{(i)}$ deviates from the temporally aggregated evidence $\mathbf{q}_t^{(i)}$:

$$\mathbf{q}_t^{(i)}\left(x_t^{(i)}\right) > \mathbf{p}_t^{(i)}\left(x_t^{(i)}\right). \tag{9}$$

This condition indicates that the current diffusion step assigns *lower support* to the committed token than the calibrated confidence, signaling a potential breakdown of local consistency. In such cases, remasking allows the model to reconsider the token in later steps.

To prevent excessive oscillation, DecoCal imposes a per-position **remasking budget**: each position can be remasked at most K times throughout the decoding process. This constraint, together with the cool-down constraint mentioned in Section 3.3, stabilize decoding dynamics by limiting revision frequency while still allowing important or uncertain positions to benefit from delayed refinement. Remasking is applied before unmasking at each diffusion step to prevent immediate reversal of newly committed tokens.

3.5 Fallback Unmasking for Decoding Progress

While calibration-guided unmasking and remasking improve reliability, they may temporarily stall decoding when no position satisfies the commitment criteria. To guarantee steady progress, DecoCal employs a fallback unmasking mechanism. If no position is committed at diffusion step t, we select the masked position with the highest instantaneous confidence and unmask it:

$$j = \underset{i:\, x_t^{(i)}=\texttt{[MASK]}}{\arg\max}\; c_t^{(i)}, \quad x_t^{(j)} \leftarrow v_t^{(j)}. \tag{10}$$

As a result, the decoding process is guaranteed to make progress at every diffusion step.

4 Experiments

4.1 Setup

Models. We conduct experiments on two representative DLLMs: **LLaDA-1.5** [2] (Zhu et al., 2025a), and **Dream**[3] (Ye et al., 2025). These models represent different architectural scales and decoding behaviors, enabling us to evaluate the robustness of decoding strategies across diverse model families.

Decoding methods. We compare the proposed decoding framework against several baseline methods, including **vanilla diffusion decoding**, **ReMDM** (Wang et al., 2025a), **RCR** (He et al., 2025), and **Tolerator** (Tian et al., 2025). The vanilla decoding methods are **maskgit** (Chang et al., 2022) for LLaDA-1.5 and **entropy** (Ye et al., 2025) for Dream as suggested in the model reports. Additionally, we apply an End-of-Sequence (EOS) penalty to Dream, following a similar approach to that used in Tolerator (Tian et al., 2025). Further details are provided in Appendix B.2.

Evaluation tasks. We evaluate decoding performance on a diverse set of reasoning and generation benchmarks, including:

- Question answering: **TriviaQA** (Joshi et al., 2017);
- Mathematical reasoning: **GSM8K** (Cobbe et al., 2021) and **MATH** (Hendrycks et al., 2021);
- Code generation: **HumanEval** (Chen, 2021) and **MBPP** (Austin et al., 2021b).

All experiments use zero-shot prompting to ensure a fair comparison. We report accuracy and steps used on each benchmark, averaged over the full evaluation set.

Parameter settings. For all models and datasets, we set confidence threshold $A = 0.9$, sensitivity $\gamma = 2.0$, and a maximum of $K = 1$ remasking step. The maximum generation length L is set to 64 for TriviaQA, and 256 for all other tasks. We set the block sizes of the LLaDA-1.5 model to be the same as those of L. Experiments of different length settings are presented in Appendix C.1. Accordingly, the cool-down duration is set to $C = 2$ for TriviaQA and $C = 8$ for the rest, scaling with the generation length L. Additionally, we set the decoding temperature to 0 to ensure deterministic, reproducible outputs.

4.2 Main Results

Table 1 presents the accuracy and step counts of DecoCal and baseline decoding methods across five diverse benchmarks, evaluated on LLaDA-1.5 and Dream. DecoCal consistently achieves state-of-the-art or highly competitive performance, demonstrating its effectiveness in enhancing decoding quality.

On LLaDA-1.5, DecoCal achieves the top accuracy on TriviaQA, HumanEval and MBPP (three out of five tasks). Specifically, DecoCal reaches 46.80% on TriviaQA and 33.80% on MBPP, outperforming all baselines. It matches Tolerator to achieve the highest accuracy of 42.68% on HumanEval. On GSM8K, DecoCal obtains a strong accuracy of 63.00%.

The most striking improvements are observed on Dream, where DecoCal sets new records across all five benchmarks. It boosts GSM8K accuracy to 60.12% and achieves 32.43% on MATH. On HumanEval and MBPP, it also delivers double-digit improvements. DecoCal's success on Dream demonstrates that with a more suitable decoding mechanism, DLLMs can achieve much better performance.

In summary, DecoCal demonstrates robust and often superior decoding accuracy across models and tasks. While it is not uniformly the top performer on every single metric and every model, its overall accuracy profile is consistently strong.

[2] https://huggingface.co/GSAI-ML/LLaDA-1.5

[3] https://huggingface.co/Dream-org/Dream-v0-Instruct-7B

Model	Method	TriviaQA		GSM8K		MATH		HumanEval		MBPP	
		Acc	Steps	Acc	Steps	Acc	Steps	Acc	Steps	Acc	Steps
LLaDA-1.5	Vanilla	45.90	64	58.98	256	21.14	256	41.46	256	33.00	256
	ReMDM	41.20	64	58.68	256	22.43	256	38.41	256	29.20	256
	RCR	46.40	64	60.80	256	21.00	256	39.02	256	31.60	256
	Tolerator	45.30	64	**73.09**	256	**23.14**	256	**42.68**	256	31.60	256
	DecoCal	**46.80**	48.71	63.00	174.01	21.86	237.11	**42.68**	255.65	**33.80**	204.90
Dream	Vanilla	52.20	64	36.01	256	14.71	256	35.98	256	30.20	256
	ReMDM	53.40	64	37.15	256	14.14	256	34.76	256	35.00	256
	RCR	52.40	64	32.75	256	14.00	256	29.88	256	29.00	256
	Tolerator	53.10	256	49.28	256	20.00	256	9.10	256	30.00	256
	DecoCal	**58.60**	100.98	**60.12**	233.43	**32.43**	280.31	**54.27**	247.95	**48.60**	272.70

Table 1: Main results on LLaDA-1.5 and Dream. Reported values are Accuracy (%) and average decoding steps. Bold text denotes the optimal result, and underlined text denotes the suboptimal result.

Method	GSM8K	HumanEval
DecoCal	**63.00**	**42.68**
fixed w	61.79	42.07
w/o cool-down	60.65	40.24
w/o remask	58.15	39.63
w/o calibration	57.47	40.24

Table 2: Ablation studies for LLaDA-1.5 on GSM8K and HumanEval.

4.3 Ablation Study

To gain deeper insight into the design choices in DecoCal, we conduct several ablation studies on GSM8K and HumanEval with LLaDA-1.5, which is the most recent among the models we evaluated, by systematically simplifying the following key components:

- *fixed* w replaces the KL-adaptive weighting in calibration with a fixed value $w = 0.5$, which is an equivalent implementation of exponential moving average (EMA);
- *w/o cool-down* disables the cool-down mechamism during unmasking;
- *w/o remask* removes the whole remasking mechanism;
- *w/o calibration* eliminates the calibration module entirely, reducing the method to a basic threshold-based unmasking strategy ($\mathbf{p} > A$).

Results are shown in Table 2. The full DecoCal model achieves the best performance on both GSM8K and HumanEval. Ablating any component degrades performance, with the largest drops observed when removing remasking (-4.85 on GSM8K, -3.05 on HumanEval) or calibration (-5.53 on GSM8K, -2.46 on HumanEval). Notably, calibration is especially crucial for GSM8K, while remasking benefits both reasoning and code generation, indicating its general utility. The cool-down mechanism and adaptive weighting also contribute consistently across tasks.

4.4 Calibration Results

To directly and quantitatively evaluate the calibration performance, we sample 200 sequences from the GSM8K benchmark and conduct inference with both vanilla decoding and the proposed DecoCal method. For each token upon unmasking, we record its output probability: the raw predictive probability for vanilla decoding, and the calibrated probability for DecoCal. We compute the Expected Calibration Error (ECE) with 20 equal-width bins for evaluation.

ECE	Min	Mean	Max	Last
Vanilla	0.1647	0.5480	0.3400	0.3818
DecoCal	0.2515	0.3283	0.3050	0.2158

Table 3: Calibration quality comparison (ECE) between Vanilla and DecoCal on 200 GSM8K sequences. Metrics are computed over all tokens in a sequence (minimum, mean, maximum) and on the last token.

As presented in Table 3, DecoCal yields consistently lower ECE values in terms of the mean, maximum, and last-token metrics. These results verify that the probabilities calibrated by DecoCal exhibit a stronger alignment with the ground-truth correctness of generated tokens, demonstrating its effectiveness beyond naive temporal smoothing.

4.5 Quality-Efficiency Trade-off

Figure 2 presents a comprehensive analysis of the trade-off between inference efficiency (measured in diffusion steps) and task accuracy for LLaDA-1.5 on GSM8K under varying parameter configurations. Each point in the plot corresponds to a

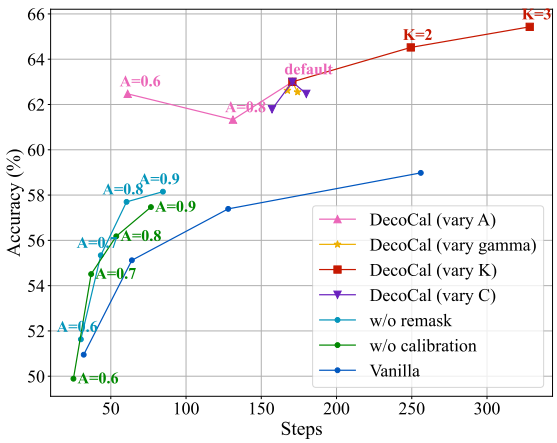

Figure 2: Results under different parameter settings for LLaDA-1.5 on GSM8K. Points closer to the top-left corner indicate better performance, achieving higher accuracy with fewer diffusion steps.

specific setting of parameters including unmasking confidence threshold A, maximum remask iterations K, cool-down steps C, and sensitivity γ.

Overall, our proposed DecoCal method consistently outperforms all ablated variants and the vanilla method. Reducing A accelerates decoding by allowing earlier commitment of tokens, often at the cost of marginal accuracy degradation. Notably, even at aggressive efficiency settings, such as $A = 0.6$, DecoCal achieves an accuracy of 62.47% using only 61.22 steps. In contrast, increasing K enables more opportunities for error correction. For instance, with $K = 3$, it reaches 65.43% accuracy using 328.38 steps, approaching the best accuracy. The parameter γ and cool-down steps C exhibit relatively minor impact on the final outcome, as evidenced by the tightly clustered points along the magenta and orange-red curves. As a result, DecoCal offers flexible control over the speed-accuracy trade-off via its core parameters, enabling practitioners to tailor inference behavior according to resource constraints.

	Runtime (s)	Avg GPU Memory Util. (%)
Vanilla	4179.71	33.47
DecoCal	2916.96	37.02

Table 4: Runtime and GPU utilization for 200 GSM8K samples on LLaDA-1.5. Runtime excludes file I/O and initialization overhead.

Beyond the speed-accuracy trade-off, we further evaluate the practical inference efficiency of DecoCal by measuring runtime and GPU resource consumption. We conduct experiments on 200 GSM8K samples with LLaDA-1.5, using a single NVIDIA A40 (48GB) GPU and a batch size of 1. The results in Table 4 demonstrate that our DecoCal method effectively reduces total inference runtime, while only incurring a moderate and acceptable rise in GPU memory utilization.

4.6 Decoding Behavior

To understand the dynamic strategy of DecoCal, we evaluate LLaDA-1.5 on 200 GSM8K samples and record which positions are subjected to *unmasking* or *remasking* at each step. The aggregated statistics are shown in Figure 3. Further phenomena are discussed in Appendix C.3.

Our analysis reveals several patterns:

1. Generation usually starts from sequence boundaries (start/end), with high activity near position 0 and the maximum index.

2. Early inference has higher uncertainty. Reduced calibration-guided unmasking and frequent remasking indicate a preference for iterative correction over direct confident generation.

3. Calibration-guided unmasking shows strong spatial coherence (contiguous bands), while remasking is sporadic, consistent with its corrective role.

4.7 Case Study

To gain a deeper understanding of how DecoCal improves decoding reliability, we analyze a representative GSM8K example in Table 5.

Without calibration, the decoding algorithm uses single-step probability thresholding and unmasks the answer digits “9” and “6” as early as steps 42–43 (89 steps in total), before valid reasoning is formed. This forms an incorrect computation $(20 + 28 + 56 = 96)$ and it becomes an irreversible decision, causing a wrong final answer.

In contrast, DecoCal defers unmasking answer tokens until late diffusion steps, i.e., step 203/218, ensuring that critical answer tokens are only committed after a complete and correct reasoning chain is almost established. This calibration-guided scheduling enables DecoCal to generate the correct answer 104 by delaying commitment to reduce premature errors. Another case study of the remasking mechanism can be found in Appendix C.4.

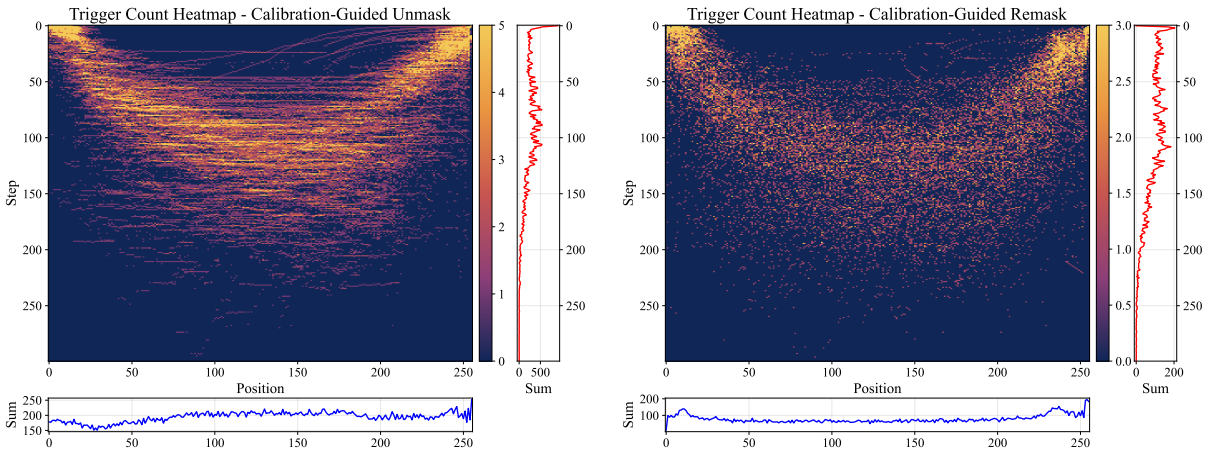

Figure 3: Heatmaps of calibration-guided decoding behavior, depicting operation frequency across steps (y-axis) and positions (x-axis), with marginal plots indicating per-step and per-position totals. **Left**: unmask frequency. **right**: remask frequency. Color intensity reflects operation density. **Red curve** (right margin): total operations per step; **blue curve** (bottom margin): total operations per position.

5 Related Work

5.1 Diffusion Models for Language Generation

Diffusion models were originally developed for continuous domains such as image generation (Yang et al., 2023; Croitoru et al., 2023), and have recently been extended to discrete sequence modeling. Early attempts to apply diffusion to language generation include Diffusion-LM (Li et al., 2022) and LD4LG (Lovelace et al., 2023), which formulate text generation as a denoising process in continuous or latent embedding spaces.

More recently, discrete diffusion language models have gained increasing attention due to their scalability and simplicity. Models such as the LLaDA (Nie et al., 2025; Zhu et al., 2025a) and Dream series (Ye et al., 2025) employ masking as a discrete noise mechanism, enabling generation through gradual unmasking and reverse prediction. Compared to autoregressive models, diffusion-based language models support parallel token prediction, allowing bidirectional context modeling and flexible control over the generation process.

5.2 Decoding Strategies for Diffusion Language Models

Decoding critically influences the quality and efficiency of diffusion-based text generation. Early work on masked and non-autoregressive models (Devlin et al., 2019; Ghazvininejad et al., 2019; Kasai et al., 2020) established iterative refinement via masking as a key paradigm.

Many methods adopt fixed unmasking schedules or rely on instantaneous signals such as token confidence (Chang et al., 2022), entropy (Ye et al., 2025), probability margins (Kim et al., 2025), or random remasking (Austin et al., 2021a). While efficient, these strategies lack mechanisms to correct early errors and are prone to noise and overconfidence.

To enable dynamic refinement, recent approaches incorporate richer signals: Running Confidence Remasking (RCR) (He et al., 2025) remasks tokens with persistently low confidence; ReMDM (Wang et al., 2025a) uses a time-dependent probabilistic remasking policy; Tolerator (Tian et al., 2025) separates generation and refinement via cross-validation; Latent Refinement Decoding (LDR) (Zhu et al., 2025b) refines in embedding space with KL-based stopping; Coherent Contextual Decoding (CCD) (Chen et al., 2025) leverages conditional mutual information for consistency and adaptive budgeting; and MEDAL (Huang et al., 2025) formulates inference as combinatorial search using Monte Carlo Tree Search during initialization. These methods enhance decoding through historical, latent, or contextual information.

5.3 Uncertainty Estimation and Calibration in Language Models

Modern neural networks are often overconfident and poorly calibrated, despite high accuracy (Guo et al., 2017), and excessive regularization to reduce overconfidence can even hurt post-hoc calibratability (Wang et al., 2021). In language gen-

Prompt: Meredith is a freelance blogger who writes about health topics and submits to clients each day as her permanent job. A blog article takes an average of 4 hours to research and write about. Last week, she wrote 5 articles on Monday and 2/5 times more articles on Tuesday than on Monday. On Wednesday, she wrote twice the number of articles she wrote on Tuesday. Calculate the total number of hours she spent writing articles in the three days.

w/o Calibration

Step 42/89: ... [MASK]: 9[MASK][EOS]
Step 43/89: ... [MASK]: 96[EOS]
Final Output:
... To find the total number of hours she spent writing articles in the three days, we add up the hours spent each day: 20 + 28 + 56 = 96 hours.
96 The answer is: 96

DecoCal

Step 203/218:
On Monday, Meredith wrote 5 articles.
On Tuesday, she wrote 5 + (2/5)*5 = 7 articles.
On Wednesday, she wrote 2*7 = 14 articles.
In total, she wrote 5 + 7 + 14 = 26 articles.
[MASK]...[MASK] 104[EOS]
Final Output
On Monday, Meredith wrote 5 articles.
On Tuesday, she wrote 5 + (2/5)*5 = 7 articles.
On Wednesday, she wrote 2*7 = 14 articles.
In total, she wrote 5 + 7 + 14 = 26 articles.
Therefore, she spent 26 * 4 = 104 hours writing articles.
Conclusively: 104

Table 5: A GSM8K sample tested with LLaDA-1.5. The correct answer is 104. Red/blue colored tokens denote the newly unmasked tokens at each step.

eration, high uncertainty correlates with hallucination (Xiao and Wang, 2021), and semantic equivalence further complicates confidence estimation (Kuhn et al., 2023). Token-level uncertainty has also been explored in structured prediction (Austin et al., 2021a), while consistency across reasoning paths, e.g., via self-consistency, serves as an implicit reliability signal (Wang et al., 2022).

Inspired by these ideas, we adapt the principle of calibration, not as a formal post-hoc method, but as a decoding heuristic in DLLMs. By aggregating predictions over time, DecoCal uses calibrated confidence to guide when to commit or revise tokens, bridging uncertainty-aware decision-making with iterative diffusion-based decoding.

6 Conclusion

We propose **DecoCal**, a decoding framework for diffusion large language models that calibrates token-level confidence by aggregating predictions across diffusion steps. Rather than unmasking tokens based on instantaneous probabilities, DecoCal uses calibrated confidence to decide when a token is stable enough to commit and when prior decisions should be revised. This approach mitigates premature errors caused by transient overconfidence and enables adaptive, accurate decoding.

Experiments across multiple models and tasks show that DecoCal consistently improves generation quality. By integrating calibration into the iterative refinement process, our method advances DLLM decoding toward more reliable and principled decision making.

Limitations

DecoCal introduces four hyperparameters: the confidence threshold A, sensitivity γ, maximum remask count K, and cool-down period C. While these parameters offer flexibility, they also require careful tuning for different tasks and datasets to achieve optimal performance. Additionally, DecoCal does not provide explicit per-sample control over the number of decoding steps. The total number of steps is bounded above by $T \cdot (K + 1)$, but the actual number of effective updates per token varies dynamically based on calibration signals and remasking behavior, making fine-grained latency or computation control challenging.

Acknowledgments

This work was supported by Beijing Natural Science Foundation (L253001), Key Laboratory of Science, Technology and Standard in Press Industry (Key Laboratory of Intelligent Press Media Technology) and National Engineering Research Center of New Electronic Publishing Technologies. We appreciate the anonymous reviewers for their helpful comments. Xiaojun Wan is the contact author.

References

Jacob Austin, Daniel D Johnson, Jonathan Ho, Daniel Tarlow, and Rianne Van Den Berg. 2021a. Structured denoising diffusion models in discrete state-spaces. *Advances in neural information processing systems*, 34:17981–17993.

Jacob Austin, Augustus Odena, Maxwell I. Nye, Maarten Bosma, Henryk Michalewski, David Dohan, Ellen Jiang, Carrie J. Cai, Michael Terry, Quoc V. Le, and Charles Sutton. 2021b. Program synthesis with large language models. *CoRR*, abs/2108.07732.

Huiwen Chang, Han Zhang, Lu Jiang, Ce Liu, and William T Freeman. 2022. Maskgit: Masked generative image transformer. In *Proceedings of the*

IEEE/CVF conference on computer vision and pattern recognition, pages 11315–11325.

Kecheng Chen, Ziru Liu, Xijia Tao, Hui Liu, Xinyu Fu, Suiyun Zhang, Dandan Tu, Lingpeng Kong, Rui Liu, and Haoliang Li. 2025. Beyond confidence: Adaptive and coherent decoding for diffusion language models. *arXiv preprint arXiv:2512.02044*.

Mark Chen. 2021. Evaluating large language models trained on code. *arXiv preprint arXiv:2107.03374*.

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

Florinel-Alin Croitoru, Vlad Hondru, Radu Tudor Ionescu, and Mubarak Shah. 2023. Diffusion models in vision: A survey. *IEEE transactions on pattern analysis and machine intelligence*, 45(9):10850–10869.

Jacob Devlin, Ming-Wei Chang, Kenton Lee, and Kristina Toutanova. 2019. BERT: Pre-training of deep bidirectional transformers for language understanding. In *Proceedings of the 2019 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies, Volume 1 (Long and Short Papers)*, pages 4171–4186, Minneapolis, Minnesota. Association for Computational Linguistics.

Marjan Ghazvininejad, Omer Levy, Yinhan Liu, and Luke Zettlemoyer. 2019. Mask-predict: Parallel decoding of conditional masked language models. In *Proceedings of the 2019 Conference on Empirical Methods in Natural Language Processing and the 9th International Joint Conference on Natural Language Processing (EMNLP-IJCNLP)*, pages 6112–6121, Hong Kong, China. Association for Computational Linguistics.

Chuan Guo, Geoff Pleiss, Yu Sun, and Kilian Q Weinberger. 2017. On calibration of modern neural networks. In *International conference on machine learning*, pages 1321–1330. PMLR.

Haoyu He, Katrin Renz, Yong Cao, and Andreas Geiger. 2025. Mdpo: Overcoming the training-inference divide of masked diffusion language models. *arXiv preprint arXiv:2508.13148*.

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

Zheng Huang, Kiran Ramnath, Yueyan Chen, Aosong Feng, Sangmin Woo, Balasubramaniam Srinivasan, Zhichao Xu, Kang Zhou, Shuai Wang, Haibo Ding, and Lin Lee Cheong. 2025. Diffusion language model inference with monte carlo tree search. *Preprint*, arXiv:2512.12168.

Mandar Joshi, Eunsol Choi, Daniel Weld, and Luke Zettlemoyer. 2017. TriviaQA: A large scale distantly supervised challenge dataset for reading comprehension. In *Proceedings of the 55th Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, pages 1601–1611, Vancouver, Canada. Association for Computational Linguistics.

Jungo Kasai, James Cross, Marjan Ghazvininejad, and Jiatao Gu. 2020. Non-autoregressive machine translation with disentangled context transformer. In *International conference on machine learning*, pages 5144–5155. PMLR.

Jaeyeon Kim, Kulin Shah, Vasilis Kontonis, Sham Kakade, and Sitan Chen. 2025. Train for the worst, plan for the best: Understanding token ordering in masked diffusions. *arXiv preprint arXiv:2502.06768*.

Lorenz Kuhn, Yarin Gal, and Sebastian Farquhar. 2023. Semantic uncertainty: Linguistic invariances for uncertainty estimation in natural language generation. *arXiv preprint arXiv:2302.09664*.

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

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

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

Justin Lovelace, Varsha Kishore, Chao Wan, Eliot Shekhtman, and Kilian Q Weinberger. 2023. Latent diffusion for language generation. *Advances in Neural Information Processing Systems*, 36:56998–57025.

Shen Nie, Fengqi Zhu, Zebin You, Xiaolu Zhang, Jingyang Ou, Jun Hu, Jun Zhou, Yankai Lin, Ji-Rong Wen, and Chongxuan Li. 2025. Large language diffusion models. *arXiv preprint arXiv:2502.09992*.

Ilya Sutskever, Oriol Vinyals, and Quoc V Le. 2014. Sequence to sequence learning with neural networks. *Advances in neural information processing systems*, 27.

Runchu Tian, Junxia Cui, Xueqiang Xu, Feng Yao, and Jingbo Shang. 2025. Finish first, perfect later: Test-time token-level cross-validation for diffusion large language models. *arXiv preprint arXiv:2510.05090*.

Deng-Bao Wang, Lei Feng, and Min-Ling Zhang. 2021. Rethinking calibration of deep neural networks: Do not be afraid of overconfidence. *Advances in Neural Information Processing Systems*, 34:11809–11820.

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

Wen Wang, Bozhen Fang, Chenchen Jing, Yongliang Shen, Yangyi Shen, Qiuyu Wang, Hao Ouyang, Hao Chen, and Chunhua Shen. 2025b. Time is a feature: Exploiting temporal dynamics in diffusion language models. *arXiv preprint arXiv:2508.09138*.

Xuezhi Wang, Jason Wei, Dale Schuurmans, Quoc Le, Ed Chi, Sharan Narang, Aakanksha Chowdhery, and Denny Zhou. 2022. Self-consistency improves chain of thought reasoning in language models. *arXiv preprint arXiv:2203.11171*.

Yijun Xiao and William Yang Wang. 2021. On hallucination and predictive uncertainty in conditional language generation. In *Proceedings of the 16th Conference of the European Chapter of the Association for Computational Linguistics: Main Volume*, pages 2734–2744, Online. Association for Computational Linguistics.

Ling Yang, Zhilong Zhang, Yang Song, Shenda Hong, Runsheng Xu, Yue Zhao, Wentao Zhang, Bin Cui, and Ming-Hsuan Yang. 2023. Diffusion models: A comprehensive survey of methods and applications. *ACM computing surveys*, 56(4):1–39.

Jiacheng Ye, Zhihui Xie, Lin Zheng, Jiahui Gao, Zirui Wu, Xin Jiang, Zhenguo Li, and Lingpeng Kong. 2025. Dream 7b: Diffusion large language models. *arXiv preprint arXiv:2508.15487*.

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

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

A Algorithm

We present the pseudocode of DecoCal in Algorithm 1. Starting from a fully masked sequence $\mathbf{x}_0$, the algorithm iteratively refines token predictions over T steps. At each step t, the model produces a probability distribution $\mathbf{p}_t$ over the vocabulary given the current sequence $\mathbf{x}_{t-1}$. To mitigate overconfidence and distributional drift, we maintain a calibrated confidence distribution $\mathbf{q}_t$, initialized as $\mathbf{q}_1 = \mathbf{p}_1$ and subsequently updated via a KL-divergence–guided interpolation.

Tokens are conditionally unmasked only when (i) their calibrated confidence $\tilde{c}_t^{(i)}$ exceeds a threshold A, (ii) the argmax under $\mathbf{p}_t$ and $\mathbf{q}_t$ agrees, and (iii) a cool-down period C has elapsed since their last remasking. Crucially, if a token has already been unmasked but the calibrated model assigns it higher probability than the uncalibrated one and its remask count r_i is below K, it is reverted to `[MASK]` to allow reconsideration. This remasking mechanism enables the algorithm to correct premature commitments.

To guarantee progress, a fallback step unconditionally unmasks the highest confidence masked token if no other token meets the unmasking criteria in a given iteration. The final output $\mathbf{x}_{t^\dagger}$ is returned after $t^\dagger$ refinement steps when there is no `[MASK]` in the sequence.

B Implementation Details

B.1 Further Details about Benchmarks

For TriviaQA (Joshi et al., 2017), we randomly sample 1,000 examples from the `wikipedia-dev` split to form our test set. For MATH (Hendrycks et al., 2021), we draw 100 samples uniformly at random from each of its seven subject-specific subsets, resulting in a test set of 700 examples. For all other datasets, we evaluate the entire official test split. The number of samples used for evaluation on each benchmark is summarized in Table 6.

Dataset	N samples
TriviaQA	1000
GSM8K	1319
MATH	700
HumanEval	164
MBPP	500

Table 6: Number of test samples used for evaluation on each benchmark dataset.

B.2 EOS Penalty for Dream Model

Tolerator (Tian et al., 2025) algorithm incorporates an end-of-sequence (EOS) penalty during decoding. The experiments reveal that merely increasing the EOS penalty can substantially improve model performance. In a similar vein, during our preliminary studies on the Dream (Ye et al., 2025) model, we observed a strong propensity to prematurely generate EOS tokens, which hinders the model's reasoning process and prevents it from completing tasks. Specifically, we compared the performance

Algorithm 1 DecoCal

Require: DLLM $\mathcal{M}$, prompt, maximum steps T, confidence threshold A, sensitivity γ, max remasks K, cool-down C

$\mathbf{x}_0 = [\texttt{[MASK]}]^L$, $\mathbf{q_0} = \texttt{undefined}$
$r_i = 0,\ \tau_i = -\infty, \forall i$ ▷ remask count & last step
for $t = 1$ to T **do**
  $\mathbf{p}_t = \text{softmax}(\mathcal{M}(\text{prompt}, \mathbf{x}_{t-1}))$ ▷ predict
  $\mathbf{x}_t := \mathbf{x}_{t-1}$
  $v_t^{(i)} = \arg\max_{v \in \mathcal{V}} \mathbf{p}_t^{(i)}(v),\ c_t^{(i)} = \mathbf{p}_t^{(i)}(v_t^{(i)}), \quad \forall i$
  ▷ Calibrate (Sec 3.2)
  if q is undefined **then**
    $\mathbf{q}_t = \mathbf{p}_t$ ▷ $\mathbf{q}_1 = \mathbf{p}_1$
  else
    $\mathbf{w}_t = \sigma(\gamma \cdot \text{KL}(\mathbf{p}_t \,\|\, \mathbf{q}_{t-1}))$
    $\mathbf{q}_t = \mathbf{q}_{t-1}^{1-\mathbf{w}_t} \odot \mathbf{p}_t^{\mathbf{w}_t}$
  end if
  $v_t^{\star(i)} = \arg\max_{v \in \mathcal{V}} \mathbf{q}_t^{(i)}(v),\ \tilde{c}_t^{(i)} = \mathbf{q}_t^{(i)}(v_t^{(i)}), \forall i$
  ▷ Remask (Sec 3.4)
  for all i with $x_t^{(i)} \neq \texttt{[MASK]}$ **do**
    if $\mathbf{q}_t^{(i)}(x_t^{(i)}) > \mathbf{p}_t^{(i)}(x_t^{(i)})$ **and** $r_i < K$ **then**
      $x_t^{(i)} := \texttt{[MASK]}$
      $r_i = r_i + 1, \quad \tau_i = t$
    end if
  end for
  ▷ Unmask (Sec 3.3)
  $\texttt{progress} = \texttt{false}$
  for all i with $x_t^{(i)} = \texttt{[MASK]}$ **and** $t - \tau_i \geq C$ **do**
    if $\tilde{c}_t^{(i)} \geq A$ **and** $v_t^{(i)} = v_t^{\star(i)}$ **then**
      $x_t^{(i)} := v_t^{(i)}$
      $\texttt{progress} = \texttt{true}$
    end if
  end for
  ▷ Fallback (Sec 3.5)
  if not $\texttt{progress}$ **then**
    $j = \arg\max_{i:\, x_t^{(i)} = \texttt{[MASK]}} c_t^{(i)}$
    $x_t^{(j)} := v_t^{(j)}$
  end if
  $t^\dagger := t$
  if $x_t^{(i)} \neq \texttt{[MASK]}, \quad \forall i$ **then**:
    break
  end if
end for
return $\mathbf{x}_{t^\dagger}$

Method	TriviaQA	GSM8K	MATH	HumanEval	MBPP
w penalty	52.20	36.01	14.71	35.98	30.20
wo penalty	49.70	30.63	15.14	32.32	30.40

Table 7: Effect of End-of-Sequence (EOS) penalty on dream model performance.

of the model under a vanilla decoding method (entropy sampling) with and without the application of an EOS penalty. Our penalty mechanism dynamically adjusts based on the proportion of EOS tokens already generated in the sequence, formulated as:

$$\alpha_t = 1 + \frac{\sum_{i=1}^{t-1} \mathbb{I}(x_i = \text{[EOS]})}{L}$$
$$\hat{logits}_t(\text{[EOS]}) = logits_t(\text{[EOS]})/\alpha_t$$

where L is the sequence length. As presented in Table 7, this simple penalty leads to markedly improved performance across multiple benchmarks when using standard entropy-based decoding.

C Supplementary Results

C.1 Results under Different Length Settings

In Section 4.1, we set fixed generation length L for both models and fixed block size B for LLaDA-1.5 (Zhu et al., 2025a). Block is a mechanism to control the degree of auto-regressive decoding. A smaller block size means a higher degree of auto-regressive generation. When $B = 1$, the diffusion generation process fallbacks to a total auto-regressive generation process.

Here we vary the generation length L for both LLaDA-1.5 (Zhu et al., 2025a) and Dream (Ye et al., 2025), and vary the block size B for LLaDA-1.5 only. We evaluate the models on TriviaQA (Joshi et al., 2017) and 200 GSM8K (Cobbe et al., 2021) samples.

As shown in Table 12, DecoCal outperforms other methods in most cases in accuracy. On LLaDA-1.5 with $L = 256$ and $B = 64$, DecoCal attains 81.50% accuracy on GSM8K, surpassing Vanilla (79.50%) and RCR (79.00%), using only 141.15 steps, a 45% reduction compared to the full 256-step baseline.

For the Dream model, which lacks block-wise control, DecoCal also demonstrates strong gains. At $L = 128$, it improves TriviaQA accuracy from 49.6% (Vanilla) to 56.5%, indicating that DecoCal effectively enhances correctness at a small cost of additional steps.

Parameter	Acc	Avg Steps
Default	63.00	170.66
Varying A		
$A = 0.8$	61.33	131.19
$A = 0.6$	62.47	61.22
Varying gamma		
$\gamma = 1.0$	62.55	174.13
$\gamma = 4.0$	62.62	167.39
Varying K		
$K = 2$	64.52	249.35
$K = 3$	65.43	328.38
Varying C		
$C = 4$	61.79	157.14
$C = 16$	62.47	179.91

Table 8: Performance of DecoCal under different parameter settings (varying A, γ, K, C) evaluated on GSM8K with LLaDA-1.5. Unspecified parameters follow the default settings given in Section 4.1.

Parameter		Acc	Avg Steps
$A = 0.8$	default	61.33	131.19
	$\gamma = 1.0$	62.02	135.49
	$K = 3$	64.75	255.37
	$C = 4$	61.41	131.19
$A = 0.6$	default	62.47	61.22
	$\gamma = 1.0$	63.15	64.20
	$K = 3$	66.79	110.9
	$C = 4$	61.79	51.08

Table 9: Additional results of DecoCal when $A = 0.8$ and $A = 0.6$, evaluated on GSM8K with LLaDA-1.5. Unspecified parameters follow the default settings given in Section 4.1.

C.2 Details on Quality-Efficiency Trade-off

Table 8 presents the detailed statistics of Figure 3 in Section 4.6. Among all configurations, the highest accuracy (Acc = 65.43) is achieved when $K = 3$, while the fewest average steps (61.22) are obtained with $A = 0.6$.

Table 9 presents additional results of DecoCal (with $A = 0.8$ and $A = 0.6$) on GSM8K using LLaDA-1.5. The results show that $A = 0.6$ outperforms $A = 0.8$ in both accuracy (62.47% vs. 61.33%) and efficiency (61.22 vs. 131.19 average steps). For parameter sensitivity, increasing K to 3 yields the most significant accuracy improvement (64.75% for $A = 0.8$, 66.79% for $A = 0.6$) but increases reasoning steps, while adjusting γ has modest effects and $C = 4$ either has little impact. Overall, K is the most impactful parameter for accuracy, and $A = 0.6$ balances performance and efficiency well on GSM8K.

A	Steps (avg $\pm$ std)	Unmask tr (%)	Unmask N_tok	Remask tr (%)	Remask N_tok
0.9	167.65 ± 45.29	34.66	4.23	44.34	4.73
0.8	133.23 ± 43.30	44.34	4.73	21.94	1.86
0.6	60.21 ± 31.76	70.82	7.66	14.97	2.73

Table 10: Statistics of the diffusion decoding process on 200 GSM8K samples with LLaDA-1.5. For the unmask and remask editing mechanisms, "tr (%)" is the trigger rate (fraction of steps where the mechanism activates), and "N_tok" is the average number of edited tokens per triggered step.

C.3 Additional Analysis on Decoding Behavior

Table 10 provides a breakdown of the decoding dynamics of DecoCal on GSM8K using LLaDA-1.5, across different confidence thresholds A. Specifically, we compute the trigger rate of unmasking and remasking operations across all steps, as well as the average number of tokens modified per operation. As A decreases, the average number of diffusion steps drops significantly, reflecting more aggressive early unmasking enabled by a lower confidence requirement.

This shift is accompanied by a marked increase in unmask activity: the unmask trigger rate rises from 34.66% to 70.82%, and the average number of tokens unmasked per activation increases from 4.23 to 7.66. Conversely, remasking becomes less frequent, indicating that DecoCal continues to revise low-confidence predictions despite faster convergence. Notably, at the default $A = 0.9$, remasking is the dominant corrective mechanism (44.34% trigger rate), while lowering A shifts the balance toward proactive unmasking. These results illustrate how DecoCal adaptively modulates the interplay between commitment and correction along the diffusion trajectory.

We also present the fallback frequency across steps and positions in Figure 4. This observation further corroborates that generation tends to initiate from sequence boundaries, as stated in Section 4.6. Unlike calibration-guided unmasking, the distribution here exhibits a scattered pattern that extends from the top-left to the bottom-right and from the top-right to the bottom-left, rather than taking the form of horizontal stripes. This discrepancy arises because the fallback mechanism is designed to decode only one token at a time.

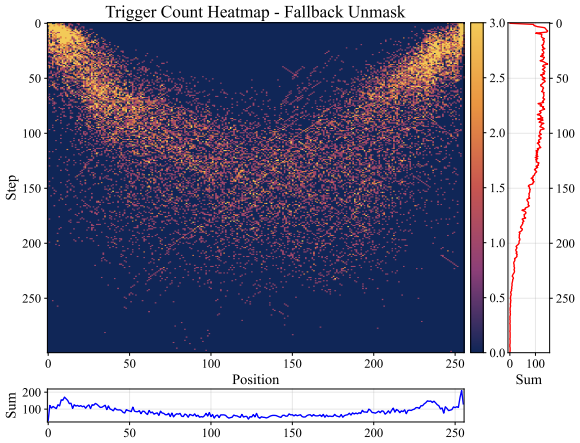

Figure 4: Heatmaps of fallback frequency across steps (y-axis) and positions (x-axis), with marginal plots indicating per-step and per-position totals. **Red curve** (right margin): total operations per step; **blue curve** (bottom margin): total operations per position.

C.4 Case Study of the Remasking Mechanism

To qualitatively illustrate the corrective role of remasking, we provide an example from TriviaQA using LLaDA-1.5 under DecoCal in Table 11. In the first step, the output exhibits structural inconsistency and redundancy. Through remasking, unstable positions are reconsidered and progressively corrected, resulting in a coherent and fully correct final answer.

Question: In what city would you find Yale University? **Answer:** New Haven, Connecticut.
Step 1: Yale University is located in [MASK] Haven [MASK] [MASK] Haven, Connecticut.[EOS] **Step 2:** Y[MASK] University is located in New Haven [MASK] [MASK] [MASK], Connecticut.[EOS] **Step 3:** Yale University is located in New Haven [MASK] [MASK] [MASK] [MASK] [MASK].[EOS] **Step 4:** Yale University is located in New Haven, Connecticut, United States.[EOS]

Table 11: Qualitative example of the remasking mechanism in DecoCal on LLaDA-1.5 model, TriviaQA dataset. At each step, blue tokens indicate newly unmasked tokens and red tokens indicate remasked tokens.

D AI Usage Disclosure

In this work, generative artificial intelligence tools were employed to support data processing workflows and refine the manuscript draft. All content generated via these tools underwent rigorous review and revision to ensure the accuracy, reliability, and academic integrity of the final work.

Model	Method	TriviaQA		GSM8K	
		Acc (%)	Steps	Acc (%)	Steps
LLaDA-1.5		*L=B=128*		*L=B=512*	
	Vanilla	43.4	128	41.5	512
	ReMDM	42.5	128	62.5	512
	RCR	43.0	128	39.0	512
	DecoCal	43.7	53.39	39.5	323.84
		L=B=32		*L=B=128*	
	Vanilla	44.0	32	61.5	128
	ReMDM	37.4	32	57.5	128
	RCR	44.3	32	68.5	128
	DecoCal	44.5	32	63.0	104.02
		L=64, B=32		*L=256, B=64*	
	Vanilla	48.2	64	79.5	256
	ReMDM	41.6	64	74.5	256
	RCR	47.8	64	79.0	256
	DecoCal	48.4	72.3	81.5	141.15
Dream		*L=128*		*L=512*	
	Vanilla	54.7	128	42.5	512
	ReMDM	55.5	128	45.0	512
	RCR	54.3	128	38.0	512
	DecoCal	60.9	146.22	55.0	312.35
		L=32		*L=128*	
	Vanilla	49.6	32	43.0	128
	ReMDM	50.5	32	42.0	128
	RCR	49.4	32	39.0	128
	DecoCal	56.5	47.61	59.5	140.37

Table 12: Accuracy (%) and average decoding steps on TriviaQA and GSM8K under varying generation lengths L and block sizes B.