Self-Rewarding Sequential Monte Carlo for Masked Diffusion Language Models

Ziwei Luo
Uppsala University
Sweden

Ziqi Jin
Nanyang Technological University
Singapore

Lei Wang*
Apodex
Singapore

Lidong Bing
Apodex
Singapore

Thomas B. Schön
Uppsala University
Sweden

Abstract

This work presents self-rewarding sequential Monte Carlo (SMC), an inference-time scaling algorithm enabling effective sampling of masked diffusion language models (MDLMs). Our algorithm stems from the observation that most existing MDLMs rely on a confidence-based sampling strategy, where only tokens with the highest prediction confidence are preserved at each step. This can restrict generation to a noise-sensitive, greedy decoding paradigm, often limiting trajectory diversity. We address this problem by launching multiple interacting diffusion processes in parallel, referred to as *particles*, for trajectory exploration. Importantly, we introduce the trajectory-level confidence as a self-rewarding signal for assigning particle importance weights. During sampling, particles are iteratively weighted and resampled to systematically steer generation towards globally confident, high-quality samples. Our self-rewarding SMC is verified on various masked diffusion language models and benchmarks, achieving significant improvement without extra training or reward guidance, while effectively converting parallel inference capacity into improved sampling quality. Our code is available at `https://github.com/Algolzw/self-rewarding-smc`.

1 Introduction

Generative modeling with diffusion models [46, 21] has led to remarkable advances across a wide range of applications, including image generation [12, 39, 36, 13], text-to-image generation [40, 53, 17], and video synthesis [22, 5]. More recently, diffusion models have further shown strong potential for discrete data generation, particularly for text [51, 41, 1, 34, 52], by modeling a forward masking process and iteratively predicting masked tokens at inference [30, 7]. Despite this progress, existing masked diffusion language models (MDLMs) adopt a greedy, confidence-based sampling strategy [41], in which only tokens with the highest prediction probability are preserved at each step, leading to myopic trajectory exploration and suboptimal generation performance.

Inference-time scaling methods have been proposed to improve MDLMs' sample diversity and quality without modifying pretrained models [8, 44, 35, 27]. Typically, they leverage human preference guidance, such as promoting fluency, enforcing structured format, or controlling toxicity [9, 37, 31], to tilt the generation trajectories towards high-reward target distributions [8, 47, 26]. While their results are impressive, such methods rely on external reward signals that are often task-specific and require hand-crafted tuning, which restricts their applicability to general MDLM-based generation.

*Corresponding author

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

In this work, we revisit the confidence-based sampling of masked diffusion and propose a *self-rewarding* sequential Monte Carlo (SMC) algorithm for inference-time scaling on general tasks. Specifically, we maintain a set of interacting diffusion processes, referred to as *particles*, to explore multiple trajectories in parallel. These particles still follow the low-confidence remasking strategy of MDLMs but will be resampled based on their trajectory-level confidence, which serves as an implicit reward signal to assign importance weights to each particle. Conceptually, our algorithm runs multiple generation processes, periodically duplicating promising candidates and discarding less confident ones based on their accumulated likelihood. The resulting self-rewarding SMC enables principled trajectory exploration and steers the sampling process towards stable, globally confident, and high-quality outputs, without extra reward models or task-specific design as guidance.

Our method is verified on various masked diffusion language models including MDLM [41] and BD3-LMs [1], and diffusion large language models (dLLMs) including LLaDA 1.5 [55] and Dream-7B [52]. The results show that the proposed self-rewarding SMC consistently improves the baseline models across multiple benchmarks.

In summary, we make the following contributions:

- We propose a general, self-rewarding sequential Monte Carlo algorithm for masked diffusion language models. The proposed method improves sampling without extra training or reward signals.
- We unify the sampling and remasking strategy of MDLMs from the probabilistic perspective, and theoretically show that the trajectory-level confidence is naturally a self-rewarding signal for SMC.
- Extensive experiments and analysis demonstrate our self-rewarding SMC improves sample quality on different pretrained models and benchmarks.

2 Background

Notation We consider $\mathbf{x} = \{x_1, \dots, x_L\}$ as a sequence of L tokens, where each token $x_\ell = \mathbf{x}(\ell)$ is a 'one-hot' column vector with V categories in the space $\mathcal{V} \triangleq \{x \in \{0,1\}^V : \sum_{i=1}^V x^{(i)} = 1\} \subset \Delta^V$ for the simplex Δ^V. We let the Vth category denote a special [MASK] token, where $\mathbf{m} \in \mathcal{V}$ is its one-hot vector. Moreover, we define $\mathrm{Cat}(\cdot; p)$ as a categorical distribution with probability $p \in \Delta^V$.

2.1 Masked Diffusion Models

Given prior distribution $\mathrm{Cat}(\cdot; \mathbf{m})$, masked diffusion models (MDMs) [2, 35, 43, 41] are characterized by parametric models p_θ trained to reverse a forward masking process for new data sampling from a full masked sequence. For a finite-time process, we let T be the number of diffusion steps and $t(i) = \frac{i}{T} \in [0, 1]$ be the continuous time. The marginal distribution of $\mathbf{x}_{t(i)}$ conditioned on the target data $\mathbf{x}_0$ is

$$p(\mathbf{x}_t \mid \mathbf{x}_0) = \mathrm{Cat}(\mathbf{x}_t;\, \alpha_t \mathbf{x}_0 + (1 - \alpha_t)\mathbf{m}), \tag{1}$$

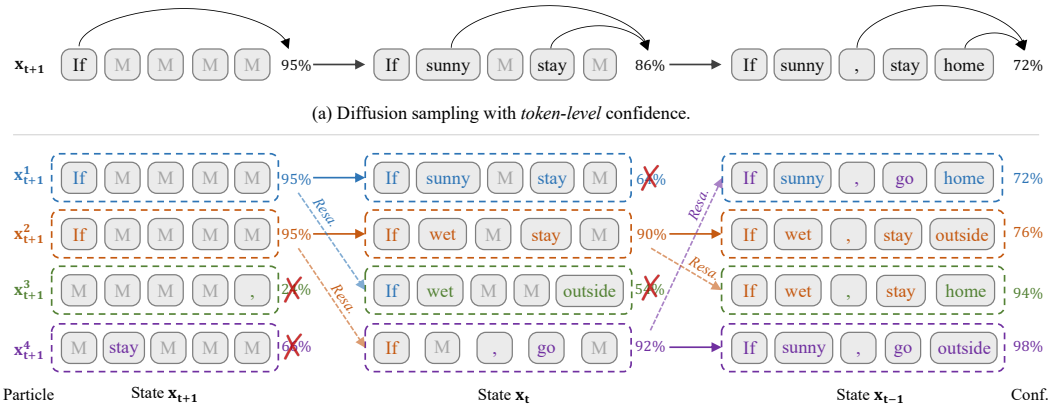
where α_t denotes a monotonically decreasing schedule satisfying $\alpha_0 \approx 1$ and $\alpha_1 \approx 0$. Let $s(i) = \frac{i-1}{T}$ be the time step directly preceding $t(i)$, the time-reversal conditional posterior for all [MASK] tokens i.e., $\mathbf{x}_t = \mathbf{m}$, can be obtained by

$$p(\mathbf{x}_s \mid \mathbf{x}_t, \mathbf{x}_0) = \mathrm{Cat}(\mathbf{x}_s;\, \frac{\alpha_s - \alpha_t}{1 - \alpha_t}\mathbf{x}_0 + \frac{1 - \alpha_s}{1 - \alpha_t}\mathbf{x}_t). \tag{2}$$

Notably, unmasked tokens i.e., $\mathbf{x}_t \neq \mathbf{m}$, remain unchanged in the reverse process. Since $\mathbf{x}_0$ is not available during inference, the reverse unmasking process is parametrized as $p_\theta(\mathbf{x}_s \mid \mathbf{x}_t, \hat{\mathbf{x}}_0)$, where $\hat{\mathbf{x}}_0 \sim \mathrm{Cat}(\mathbf{x}_0; p_\theta(\mathbf{x}_t))$ is sampled from the model predictive distribution. To simplify the notation, we only consider sampling from the diffusion reverse process and denote $t \coloneqq t(i)$ as the discrete time step throughout the work.

2.2 Importance Sampling and Sequential Monte Carlo

Assume we want to estimate expectations under trajectory target distribution $\pi(\mathbf{x}_{t:T})$, such as $\mathbb{E}_\pi[f(\mathbf{x}_{t:T})]$, where $f(\cdot)$ is a test function. Sampling from π is generally intractable.



(a) Diffusion sampling with *token-level* confidence.

(b) Our self-rewarding SMC with *trajectory-level* confidence for diffusion sampling.

Figure 1: Illustrative example of text generation using (a) masked diffusion models and (b) our self-rewarding SMC framework. Here, 'M' represents [MASK] tokens and '*Resa.*' denotes resampling. Our method maintains multiple diffusion processes, called *particles*, to explore trajectories in parallel.

Importance sampling (IS) [38] introduces a proposal distribution $q(\mathbf{x}_{t:T})$ that allows the expectation to be rewritten as

$$\mathbb{E}_\pi[f(\mathbf{x}_{t:T})] = \mathbb{E}_q[\frac{\tilde{\pi}(\mathbf{x}_{t:T})}{q(\mathbf{x}_{t:T})} f(\mathbf{x}_{t:T})] \approx \frac{1}{N}\sum_{i=1}^{N} w_t^i f(\mathbf{x}_{t:T}^i), \tag{3}$$

where $\mathbf{x}_{t:T}^i \sim q(\mathbf{x}_{t:T})$ and $\tilde{\pi}$ is the unnormalized density. Moreover, $w_t^i = \tilde{W}_t^i / \sum_{j=1}^N \tilde{W}_t^j$ is the normalized importance weight, where $\tilde{W}_t^i = \tilde{\pi}(\mathbf{x}_{t:T}^i)/q(\mathbf{x}_{t:T}^i)$ gauges the discrepancy between the target distribution and the proposal distribution. While conceptually simple, importance sampling often suffers from unfavorably high variance.

Sequential Monte Carlo (SMC) [11, 6] improves upon IS by introducing a sequence of intermediate unnormalized path measures $\tilde{\pi}_t(\mathbf{x}_{t:T})$, whose terminal distribution coincides with the desired trajectory target distribution. SMC incorporates resampling and sequential weighting techniques across the trajectory, thereby reducing variance in practice. We begin by defining the incremental importance weights as

$$\tilde{w}_{t-1}(\mathbf{x}_{t-1:T}) = \frac{\tilde{\pi}_{t-1}(\mathbf{x}_{t-1:T})}{\tilde{\pi}_t(\mathbf{x}_{t:T})\, q_{t-1}(\mathbf{x}_{t-1} \mid \mathbf{x}_t)}, \tag{4}$$

where $q_{t-1}(\mathbf{x}_{t-1} \mid \mathbf{x}_t)$ is a Markovian sequential proposal operating in reverse time (see Appendix A.1 for more details). During sampling, we initialize a set of N particles $\mathbf{x}_T^i \sim q_T(\mathbf{x}_T)$, each representing a trajectory distribution, with weights $w_T^i = \tilde{\pi}_T(\mathbf{x}_T^i)/q_T(\mathbf{x}_T^i)$. At each iteration e.g., from t to $t-1$, SMC takes the following three steps:

i) *Resample*: resample ancestor $\{\mathbf{x}_t^i\}_{i=1}^N$ according to the weights $\{w_t^i\}_{i=1}^N$;
ii) *Propagate*: sample new particles from proposal distribution $\mathbf{x}_{t-1}^i \sim q_{t-1}(\mathbf{x}_{t-1} \mid \mathbf{x}_t^i)$;
iii) *Re-weight*: compute and accumulate Eq. (4), and normalize weight $w_{t-1}^i = \frac{\tilde{w}_{t-1}^i}{\sum_{j=1}^N \tilde{w}_{t-1}^j}$.

The resulting collection of weighted particles provides an asymptotically consistent approximation of the trajectory target distribution.

3 Method

3.1 Reformulate the Sampling of MDMs

We consider sampling from a pretrained masked diffusion model $p_\theta(\mathbf{x}_t)$. Given a mask set $\mathcal{M}_t \triangleq \{\ell : \mathbf{x}_t(\ell) = [\texttt{MASK}]\}$ at time t, recall that the learned posterior $p_\theta(\mathbf{x}_{t-1} \mid \mathbf{x}_t, \hat{\mathbf{x}}_0)$ in Eq. (2) is only

applied for mask tokens i.e.,

$$\mathbf{x}_{t-1}(j) \sim p_\theta(\mathbf{x}_{t-1}(j) \mid \mathbf{x}_t, \hat{\mathbf{x}}_0), \quad j \in \mathcal{M}_t, \tag{5}$$

where $\hat{\mathbf{x}}_0 \sim \mathrm{Cat}(\mathbf{x}_0; p_\theta(\mathbf{x}_t))$ is sampled from the model predictive distribution. For each token $\mathbf{x}_{t-1}(j)$, we directly define its confidence as the model probability on j, as

$$\mathbf{c}_t(j) \coloneqq p_\theta\big(\hat{\mathbf{x}}_0(j) \mid \mathbf{x}_t\big), \quad j \in \mathcal{M}_t. \tag{6}$$

At each iteration, MDMs update a subset $\mathcal{S}_t \subseteq \mathcal{M}_t$ following a predefined policy, such as the *low-confidence remasking* strategy [34], to ensure an iterative unmasking process from $\mathbf{x}_T$ to $\mathbf{x}_0$.

Low-confidence remasking has been widely used in diffusion large language models as an efficient strategy for sequence generation. Typically, by defining a schedule ρ_t to specify the number of tokens to be unmasked at step t, we introduce the following policy:

$$\mathcal{S}_t^{\text{top-k}} = \{\, j \in \mathcal{M}_t : \ \text{Top-}\rho_t\{\mathbf{c}_t(j)\} \,\}, \tag{7}$$

which indicates only the highest probability tokens are preserved at each step. Note that the schedule ρ_t can be a scalar function of t, or be more flexible by explicitly defining ρ_t as a confidence threshold [50], such that

$$\mathcal{S}_t^{\text{thr}} = \{\, j \in \mathcal{M}_t : \ \mathbf{c}_t(j) \geq \rho_t \,\}, \tag{8}$$

which enables faster sampling while preserving performance when the model is confident in its predictions. In summary, the reverse transition distribution of each token $\mathbf{x}_t(j)$ can be formulated by

$$p_\theta(\mathbf{x}_{t-1}(j) \mid \mathbf{x}_t, \hat{\mathbf{x}}_0) = \begin{cases} p_\theta(\mathbf{x}_{t-1}(j) \mid \mathbf{x}_t, \hat{\mathbf{x}}_0), & j \in S_t, \\ \mathrm{Cat}(\mathbf{x}_{t-1}(j); \mathbf{m}), & j \in M_t \setminus S_t, \\ \mathrm{Cat}(\mathbf{x}_{t-1}(j); \mathbf{x}_t), & j \notin M_t, \end{cases} \tag{9}$$

where $\mathrm{Cat}(\mathbf{x}_{t-1}(j); \mathbf{x}_t)$ is a point-mass categorical distribution concentrated at $\mathbf{x}_t(j)$. Accordingly, the reverse transition kernel over the full sequence $\mathbf{x}_t$ is

$$K_t(\mathbf{x}_t, \mathbf{x}_{t-1}) = \prod_{j=1}^{L} p_\theta(\mathbf{x}_{t-1}(j) \mid \mathbf{x}_t, \hat{\mathbf{x}}_0). \tag{10}$$

This transition kernel deterministically preserves unmasked tokens, remasks low-confidence tokens, and samples newly accepted tokens according to the model prediction.

One problem of sampling from Eq. (9) is that only step-wise confidence i.e., $\mathbf{c}_t$ in Eq. (6), is utilized for remasking. This often biases generation towards locally optimal tokens, inducing noise-sensitive, myopic exploration of the sequence trajectory, as illustrated in Figure 1.

3.2 Self-Rewarding Sequential Monte Carlo

Assume that we have N diffusion sampling processes, called particles, to generate N sequences in parallel. To tilt sampling towards globally confident sequence generation, we define a Feynman–Kac model [10] with potential $G_{t-1}(\mathbf{x}_t, \mathbf{x}_{t-1}) = \prod_{j \in S_t} p_\theta(\mathbf{x}_{t-1}(j) \mid \mathbf{x}_t)$, which is the joint probability of accepted tokens within the set $\mathcal{S}_t$. Intuitively, the potential denotes how confident the model is in the tokens at step t, performing as a self-rewarding signal for SMC update. In addition, we define the intermediate unnormalized path measures $\tilde{\pi}_t(\mathbf{x}_{t:T})$ to satisfy the following recursion:

$$\tilde{\pi}_{t-1}(\mathbf{x}_{t-1:T}) = \tilde{\pi}_t(\mathbf{x}_{t:T})\, K_t(\mathbf{x}_t, \mathbf{x}_{t-1})\, G_{t-1}(\mathbf{x}_t, \mathbf{x}_{t-1}), \tag{11}$$

where $K_t(\mathbf{x}_t, \mathbf{x}_{t-1})$ is the reverse diffusion transition kernel in Eq. (10). Importantly, the potential G is used only for particle reweighting and resampling. Thus this recursion does not replace the diffusion sampler with a temperature-modified transition kernel. Recall that SMC defines an incremental importance weight for each particle (see Eq. (4)). By letting its proposal equal the transition kernel in Eq. (11), we obtain the following result:

Proposition 3.1. *Given a pretrained diffusion model p_θ, let $\{\tilde{\pi}_t(\mathbf{x}_{t:T})\}_{t=0}^{T}$ denote the unnormalized path measures defined by the recursion in Eq. (11). If the sequential proposal in SMC is chosen to be the diffusion transition kernel, i.e., $q_{t-1}(\mathbf{x}_{t-1} \mid \mathbf{x}_t) = K_t(\mathbf{x}_t, \mathbf{x}_{t-1})$, then the incremental importance weights at step $t-1$ are given by*

$$\tilde{w}_{t-1}(\mathbf{x}_{t-1:T}) = \prod_{j \in S_t} \mathbf{c}_t(j), \tag{12}$$

where $\mathbf{c}_t(j) \coloneqq p_\theta\big(\hat{\mathbf{x}}_0(j) \mid \mathbf{x}_t\big)$ is the token confidence assigned by the model predictive distribution and $\mathcal{S}_t$ denotes the selected mask subset to be updated at step t.

The proof is provided in Appendix A.2. Under our SMC framework, Proposition 3.1 shows that the incremental weight takes the form of a trajectory-level confidence score, accumulating the confidences of accepted tokens across sampling steps until the sequence is fully unmasked. Specifically, when the diffusion transition kernel is used as the proposal and the accepted-token likelihood is used as the potential, the resulting SMC weight analytically reduces to a simple product of token confidences.

3.3 Practical Sampling with Self-Rewarding SMC

We now describe how Eq. (12) is incorporated into the sampling procedure of masked diffusion models via sequential Monte Carlo. As shown in Figure 1 and Algorithm 1, we begin by initializing N particles that are fully masked sequences, with uniform weights. At each step, we perform *resample*, *propagate*, and *re-weight* as a standard SMC described in Sec. 2.2. Specifically, the diffusion sampling with local confidence-based remasking is performed during propagation, denoted by the transition kernel in Eq. (9) and Eq. (10). Then each particle is reweighted according to Eq. (12), which forms a trajectory-level confidence score for resampling. The final output is selected from the resulting particle set with the maximum weight. In addition, we use effective sample size (ESS) [14] and Gumbel-Max trick [23] to further improve the sample efficiency.

Algorithm 1 Self-Rewarding SMC (SR-SMC)

Require: Pretrained diffusion model p_θ, sampling steps T, number of particles N, remasking policy $\textsc{Select}(\cdot)$.
Ensure: Generated sequence $\hat{\mathbf{x}}_0$.
1: **Initialize** N particles i.e., sequences $\{\mathbf{x}_T^i\}_{i=1}^N$ with all tokens set to [MASK], and weights $w_T^i = 1/N$ for all i.
2: **for** $t = T, \dots, 1$ **do**
3: **Resample** $\{\mathbf{x}_t^i\}_{i=1}^N$ according to weights $\{w_t^i\}_{i=1}^N$.
4: **Propagate** with mask set $\mathcal{M}_t^i$ for all i:
5: i) sample $\hat{\mathbf{x}}_0^i \sim p_\theta(\mathbf{x}_t^i)$ and compute confidence $\mathbf{c}_t^i$.
6: ii) select update set $\mathcal{S}_t^i \leftarrow \textsc{Select}(\mathbf{c}_t^i, \mathcal{M}_t^i)$.
7: iii) sample $\mathbf{x}_{t-1}^i \sim K_t(\mathbf{x}_t^i, \mathbf{x}_{t-1}^i)$ using Eq. (9).
8: **Re-weight** by computing $\tilde{w}_{t-1}^i$ using Eq. (12).
9: **end for**
10: **return** $\hat{\mathbf{x}}_0 \leftarrow \mathbf{x}_0^{i^\star}$ where $i^\star = \arg\max_i \tilde{w}_0^i$.

Adaptive resampling. In practice, resampling at every diffusion step might be unnecessary in particular when the variance of weights w_t is low. We therefore adopt an adaptive resampling strategy based on the effective sample size, which is defined as $\mathrm{ESS} = 1/\sum_{i=1}^N (w_t^i)^2$. We follow a common practical setting [14] to let resample be triggered only when ESS falls below $N/2$, which indicates significant weight degeneracy.

Gumbel-Max sampling. Following Zheng et al. [54], we employ the Gumbel-Max trick to sample discrete tokens from a controlled categorical distribution for masked diffusion models. Particularly, given logits $\mathbf{z}_\ell$ over the vocabulary, token sampling is performed as

$$\mathbf{x} = \arg\max_\ell \big(\mathbf{z}_\ell/\tau + \mathbf{g}_\ell\big), \quad \mathbf{g}_\ell \sim \mathcal{G}(0,1), \tag{13}$$

where τ is a temperature and $\mathcal{G}$ denotes the Gumbel distribution. This Gumbel-Max trick approximates sampling from $\mathrm{Cat}(\cdot; \mathrm{softmax}(\mathbf{z}))$. Note that the temperature $\tau = 0$ represents deterministic `argmax` decoding and $\tau = 1$ recovers the standard categorical sampling.

4 Experiments

Our self-rewarding SMC is evaluated across multiple pretrained models and benchmarks to demonstrate its ability to improve sampling of diffusion language models.

4.1 Experimental Setup

Pretrained models. We investigate two kinds of pretrained models: 1) masked diffusion language models, including MDLM [41] and BD3-LMs [1] pretrained on the OpenWebText (OWT) dataset [15] for sample quality evaluation; and 2) diffusion large language models including LLaDA 1.5 [55] and Dream-7B [52]. All of them are pretrained and finetuned on >2T tokens for general task evaluations. Both settings adopt a semi-autoregressive (Semi-AR) generation structure for higher-quality sequence generation. In addition, all models except MDLM employ a block-wise generation policy for a more efficient sampling.

Implementation. For all experiments, SR-SMC and the corresponding baselines use the same hyperparameters, including temperature, top-p nucleus sampling threshold, sequence length, and

Table 1: Performance of Self-Rewarding SMC (SR-SMC) on Diffusion Large Language Models. Results compare baselines versus SR-SMC variants using block decoding with block size = 32 with KV cache and parallel decoding enabled [50].

Benchmark	Length	LLaDA-1.5	*w/ SR-SMC*	Dream-7B	*w/ SR-SMC*
GSM8K (5-shot)	256	79.8	**80.7** (+0.9)	76.2	**78.0** (+1.8)
	512	80.4	**82.0** (+1.6)	78.0	**78.0** (+0.0)
MATH (4-shot)	256	38.2	**41.8** (+3.6)	41.6	**45.2** (+3.6)
	512	41.4	**45.4** (+4.0)	44.6	**47.6** (+3.0)
HumanEval (0-shot)	256	38.7	**41.5** (+2.8)	47.9	**53.7** (+5.8)
	512	35.4	**41.5** (+6.1)	45.1	**53.7** (+8.6)
MBPP (3-shot)	256	40.4	**44.2** (+3.8)	42.0	**48.6** (+6.6)
	512	40.2	**43.2** (+3.0)	39.6	**46.4** (+6.8)
Average	256	49.3	**52.1** (+2.8)	51.9	**56.4** (+4.5)
	512	49.4	**53.0** (+3.6)	51.8	**56.4** (+4.6)

block size when applicable. We do not perform method-specific hyperparameter tuning for SR-SMC or any baseline. The default number of particles is set to 4. Moreover, we use temperature $\tau = 1$ and identical decoding settings for all comparisons. For dLLMs experiments, we follow Wu et al. [50] to enable KV cache and parallel decoding (i.e., with a threshold-based policy, see Eq. (8)) for inference acceleration. We test MDLM and BD3-LMs on a single NVIDIA H200 GPU and evaluate all dLLMs experiments on 8 NVIDIA A800 GPUs with a single batch size.

4.2 Sample Quality Evaluation

We first evaluate SR-SMC on pretrained masked diffusion language models, including MDLM [41] and BD3-LMs [1], for open-ended text generation. We also compare against autoregressive (AR), SEDD [30], and SSD-LM [18] baselines. As shown in Table 2 and Figure 6 (in Appendix), we generate sequences with lengths $L = 1024$ and $L = 2048$, and evaluate sample quality using generative perplexity under GPT2-Large. One can observe that SR-SMC consistently improves both MDLM and BD3-LMs. For MDLM, SR-SMC reduces Gen. PPL from 46.8 to 25.8 at $L = 1024$, and from 41.3 to 25.9 at $L = 2048$. For BD3-LMs, SR-SMC further improves all block-size variants, with the strongest setting $L' = 4$ achieving 16.1 and 15.1 Gen. PPL at $L = 1024$ and $L = 2048$, respectively, approaching the AR baseline. These results indicate that SR-SMC effectively converts additional inference-time compute into improved text sample quality. We further report entropy results in Table 9, showing that the improvement is not simply due to low-entropy collapse.

Table 2: Generative perplexity (Gen. PPL; ↓) comparison of 300 samples with sequence lengths $L = 1024, 2048$. We do not perform hyperparameter tuning.

Model	$L = 1024$	$L = 2048$
AR	14.1	13.2
Diffusion		
SEDD	52.0	–
MDLM	46.8	41.3
MDLM *w/ SR-SMC*	25.8	25.9
Block Diffusion (with block size L')		
SSD-LM		
$L' = 25$	37.2	35.3
BD3-LMs		
$L' = 16$	33.4	31.5
$L' = 8$	30.4	28.2
$L' = 4$	25.7	23.6
BD3-LMs *w/ SR-SMC*		
$L' = 16$	21.1	20.2
$L' = 8$	18.9	17.3
$L' = 4$	**16.1**	**15.1**

4.3 Results on Diffusion Large Language Models

To further evaluate the effectiveness and generalizability of our SR-SMC, we conduct experiments on two representative diffusion large language models: LLaDA-1.5 [55] and Dream-7B [52]. Following Wu et al. [50], we evaluate these models across four challenging benchmarks: GSM8K and MATH for mathematical reasoning, and HumanEval and MBPP for code generation. The performance is measured using two different generation lengths ($L \in \{256, 512\}$) with a block size of 32.

The results are summarized in Table 1. Overall, incorporating SR-SMC in sampling consistently improves the results across all benchmarks, model architectures, and generation lengths. Typically, our algorithm achieves average performance gains of **2.8+** and **4.5+** for LLaDA-1.5 and Dream-7B, respectively. This indicates a strong and consistent benefit of inference-time scaling through particle-based sampling. Moreover, the improvements on both mathematical and coding tasks suggest

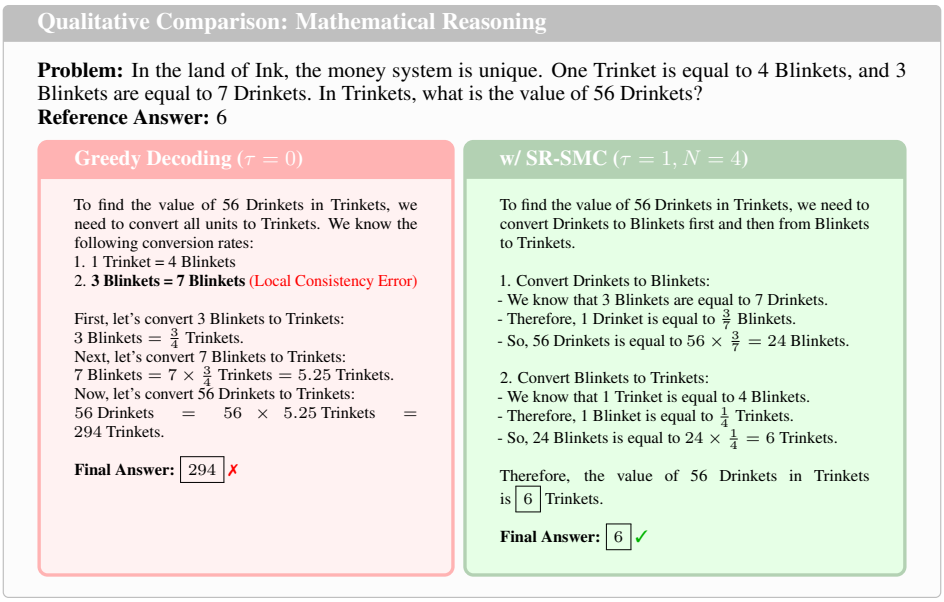


Figure 2: A qualitative comparison of reasoning trajectories. Greedy decoding leads to a hallucinated identity (3 Blinkets = 7 Blinkets) that persists through the chain. SR-SMC utilizes trajectory-level confidence to explore multiple trajectories and successfully recovers the correct multi-step conversion.

that SR-SMC generalizes well across different task domains. These gains are also sustained when generating much longer sequences, indicating that our trajectory-level resampling effectively mitigates error accumulation in diffusion-based generation. A qualitative comparison is shown in Figure 2. Collectively, these findings indicate that SR-SMC can serve as an effective and robust solution to computational scaling in dLLMs.

5 Discussion and Analysis

5.1 Scaling the Number of Particles

As the number of particles N increases, we observe a clear and consistent improvement in performance across both large models (LLaDA-1.5 and Dream-7B) on all four benchmarks, as shown in Table 3 and Figure 3. When $N = 1$, the models reduce to standard parallel decoding, which serves as a lower bound on performance across all benchmarks. Scaling the particles to $N = 2, 3, 4$ steadily improves performance (see Figure 3), with the strongest gains typically achieved at $N = 3$ or $N = 4$. On average, scaling particles to $N = 4$ improves LLaDA-1.5 from 49.3 to 52.1 and Dream-7B from 51.9 to 56.4. The same trend is also observed on the text generation task in Table 4. Since the text generation models are relatively small, we further scale the number of particles up to $N = 16$, which continues to improve generative perplexity. These results show that increasing the number of particles effectively expands the search space over generation trajectories, allowing the model to recover from locally suboptimal token choices and accumulate higher trajectory-level confidence. Interestingly, even a modest increase to $N = 2$ yields significant gains over the baseline, highlighting that our SR-SMC as a practical and scalable inference-time method to masked diffusion language models.

Table 3: Scaling the number of particles N for LLaDA-1.5 and Dream-7B (with $L = 256$) across benchmarks.

Model	Benchmark	$N = 1$	$N = 2$	$N = 3$	$N = 4$
LLaDA-1.5	GSM8K (5-shot)	79.8	**81.7**	80.7	80.7
	MATH (4-shot)	38.2	41.8	41.8	41.8
	HumanEval (0-shot)	38.7	39.0	39.0	**41.5**
	MBPP (3-shot)	40.4	44.2	44.2	44.2
	Average	49.3	51.7	51.4	**52.1**
Dream-7B	GSM8K (5-shot)	76.2	76.3	77.9	**78.0**
	MATH (4-shot)	41.6	43.2	**45.6**	45.2
	HumanEval (0-shot)	47.9	52.4	**54.6**	53.7
	MBPP (3-shot)	42.0	43.6	44.2	**48.6**
	Average	51.9	53.9	55.6	**56.4**

Table 4: Scaling the number of particles for MDLM w/ SR-SMC on text generation with $L = 1024$.

#Particles	$N = 1$	$N = 4$	$N = 8$	$N = 16$
Gen. PPL↓	46.8	25.8	22.7	20.5
Entropy ↑	5.33	5.15	5.08	5.00

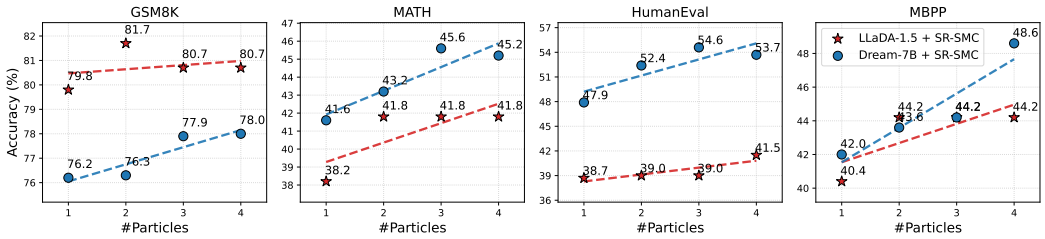


Figure 3: Comparison results of LLaDA 1.5 and Dream-7B using different numbers of particles on four tasks. Each marker denotes the empirical result while the dashed curves indicate first-order polynomial fits used solely to illustrate the overall trends as N increases.

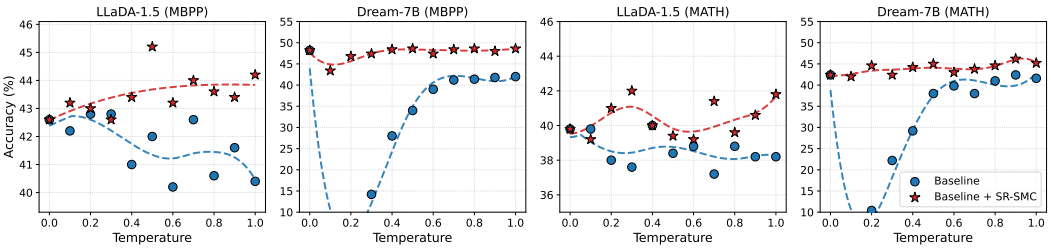


Figure 4: The accuracy of LLaDA-1.5 and Dream-7B on MBPP and MATH datasets as the temperature varies uniformly from 0.0 to 1.0. The blue circles represent the baseline with standard parallel decoding, while the red stars denote the results using our SR-SMC with $N = 4$ particles. Our method consistently demonstrates better robustness across benchmarks and the entire temperature range.

5.2 Analysis of Particle Overtake

To investigate the mechanical advantage of maintaining multiple interacting trajectories, we quantify the occurrence of *overtake* events during the block-wise generation process. We define an overtake event as a case where the particle with the highest probability at the end of a block was not the dominant particle (i.e., highest probability) at the start of that same block.

Table 5: Analysis of particle "Overtake".

Benchmark	# Overtake	Percentage
GSM8K	160 / 640	25.0%
MBPP	154 / 640	24.1%
MATH	197 / 640	30.8%
HumanEval	196 / 640	30.6%
Average	-	**27.6%**

Table 5 analyzes 640 blocks per benchmark using LLaDA-1.5. We observe that overtake events occur in approximately 24% to 31% of all blocks across tasks. This empirical evidence indicates that the particles in SR-SMC are not merely performing idle exploration around greedy trajectories. Instead, the resampling mechanism preserves initially non-dominant but high-potential trajectories, allowing them to eventually outperform the local greedy choices that standard decoding methods would otherwise lock into. Notably, such behavior cannot be captured by single-trajectory decoding or purely greedy confidence-based remasking, which irreversibly commits to the locally dominant particle at the beginning of each block.

5.3 Effect of Gumbel Noise

In this study, we examine the robustness of SR-SMC under different sampling stochasticity by varying the Gumbel noise temperature $\tau \in [0, 1.0]$. As shown in Figure 4, standard parallel decoding is sensitive to the temperature, especially for Dream-7B, where low-temperature sampling leads to severe performance degradation on both MBPP and MATH, likely due to repetitive deterministic decoding. In contrast, SR-SMC with $N = 4$ particles remains consistently more stable across the full temperature range. By maintaining multiple interacting trajectories and resampling them according to trajectory-level confidence, SR-SMC avoids being trapped by locally confident but suboptimal generations. For LLaDA-1.5, although the baseline is less unstable, SR-SMC still outperforms it under most temperature settings. These results demonstrate that SR-SMC improves not only inference-time scaling, but also the robustness of diffusion-based language generation to the choice of sampling temperature.

Table 6: Compute matched comparison (Best-of-N and ReMDM [48]) on text generation.

Method	Gen. PPL↓	Entropy↑
MDLM	46.8	5.33
MDLM w/ Best-of-N	38.6	5.31
MDLM w/ ReMDM	35.5	**5.35**
MDLM w/ SR-SMC (Ours)	**25.8**	5.15

Table 7: Ablation experiment of different SMC potentials on text generation.

Method	Gen. PPL↓	Entropy↑
MDLM (baseline)	46.8	5.33
SR-SMC (Ours)	**25.8**	5.15
SR-SMC w/ token entropy	39.0	**5.36**
SR-SMC w/ token prob. margin	38.2	5.32

5.4 Compute-Matched Comparison

To verify that the gains of SR-SMC do not simply come from using more inference-time computation, we compare it with other inference-time scaling baselines under the same sampling budget. Specifically, we evaluate Best-of-N and the stochastic remasking sampler ReMDM [48] with 4K NFEs, matching the computational cost of MDLM with SR-SMC.

As shown in Table 6, all inference-time scaling methods improve over the MDLM baseline, indicating that additional sampling computation can generally benefit masked diffusion language models. The entropy remains reasonably high for all methods, showing that the improvement in sample quality is not caused by a trivial collapse toward overly deterministic generations. In addition, SR-SMC achieves the best generative perplexity among all compute-matched methods. This demonstrates that SR-SMC does not merely rely on drawing more samples or performing additional diffusion steps, but uses the extra computation more effectively through trajectory-level reweighting and resampling.

5.5 Ablation on SMC Potentials

We further study whether the proposed trajectory-level confidence is essential for the effectiveness of SR-SMC. To this end, we replace our trajectory-level confidence with two alternative self-rewarding signals: token entropy [3] and probability margin [24]. The entropy-based potential encourages particles with lower predictive uncertainty, while the probability-margin potential favors particles whose selected tokens have a larger gap between the top prediction and competing alternatives.

The results are reported in Table 7. Both alternatives improve over the MDLM baseline, reducing Gen. PPL from 46.8 to 39.0 and 38.2, respectively. This confirms that the SMC framework can benefit from different model-internal confidence signals. Nevertheless, our trajectory-level probability potential achieves a substantially lower Gen. PPL of 25.8, outperforming both alternatives by a large margin. These results indicate that accumulating the probabilities of accepted tokens along the generation trajectory provides a more effective global confidence estimate than step-wise uncertainty or local probability separation, supporting the design choice derived in Proposition 3.1.

5.6 Trajectory Confidence Analysis

We examine whether trajectory confidence is informative for downstream correctness and whether maximum-weight selection is a reasonable final decoding rule. For each example, we rank the $N = 4$ final particles according to their accumulated trajectory confidence and evaluate each rank separately. We additionally report uniform random selection from the final particle set. As shown in Figure 5, the highest-confidence particle achieves the best performance on both GSM8K and HumanEval, and performance generally decreases with confidence rank. Maximum-confidence selection also outperforms random selection. These results indicate that trajectory confidence is positively associated with downstream correctness and provides a useful, although imperfect, criterion for final particle selection.

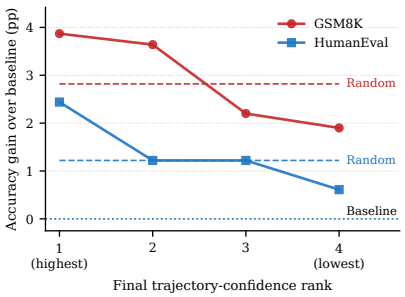


Figure 5: Confidence–correctness analysis on LLaDA-1.5 with $N = 4$. Rank 1 denotes the highest-confidence particle, and dashed lines denote random selection.

5.7 Zero-Shot Evaluation

To evaluate generalizability across prompting settings, we conduct zero-shot experiments on GSM8K and MATH. We note that the results for HumanEval presented in our main experiments (Table 1) are inherently zero-shot. For the MBPP benchmark, however, we observed that both base models failed to produce code in the requisite format for automated post-processing without few-shot examples. Unlike few-shot settings, zero-shot generation relies exclusively on the model's intrinsic reasoning capabilities and the effectiveness of the inference-time exploration. As illustrated in Table 8, SR-SMC consistently enhances the performance of both LLaDA-1.5 and Dream-7B in these zero-shot settings. These results demonstrate that trajectory-level confidence serves as a robust implicit reward signal, enabling the model to effectively navigate a wider exploration space and steer away from low-quality outputs even in the absence of prompting demonstrations.

Table 8: Zero-shot performance of LLaDA-1.5 and Dream-7B variants on GSM8K and MATH benchmarks with length $L = 256$.

Model	Benchmark	Baseline	w/ *SR-SMC*
LLaDA-1.5	GSM8K (0-shot)	75.4	**76.3**
	MATH (0-shot)	35.2	**36.6**
Dream-7B	GSM8K (0-shot)	77.9	**81.1**
	MATH (0-shot)	44.2	**49.2**

6 Related Work

As D3PM [2] introduced the state absorbing with mask token for discrete diffusion models, masked diffusion language models have attracted increasing attention as a promising alternative to autoregressive (AR) models [41, 30, 42, 43, 1]. Built upon MDLMs, diffusion large language models such as LLaDAs [34, 55, 4], Dream [52], and DiffuLLaMA [16] have demonstrated strong scalability and achieved competitive performance when compared to similarly sized AR models. Subsequently, dLLM-Cache [29] and Fast-dLLM [50] introduced caching and parallel decoding to MDLMs, further improving inference efficiency and their potential for real-world applications.

Despite these advances, existing MDLMs primarily improve performance through model scaling [33, 34], architectural modifications [49, 4], or training-time interventions [42, 19], while the role of inference-time scaling remains largely unexplored. Recent works [32, 51, 13, 6] proposed scaling test-time compute for diffusion models, substantially improving the performance beyond simply scaling diffusion sampling steps. Singhal et al. [44] further illustrated this idea on MDLMs using the sequential Monte Carlo framework. Dang et al. [8] extend it with particle Gibbs sampling that enables generation refinement via an external task-specific reward guidance. In parallel, ReMDM [48] introduces a principled remasking strategy to improve text quality with more sampling steps. In contrast, our proposed algorithm is self-rewarding and can be used for general tasks with arbitrary pretrained models and remasking strategies.

7 Conclusion

In this paper, we propose a novel inference-time scaling algorithm for masked diffusion language models (MDLMs). Our key algorithm is a sequential Monte Carlo (SMC) method where the trajectory confidence is used as importance weights. This results in a self-rewarding SMC framework that promotes globally confident generation trajectories, without requiring additional training or external reward guidance. Extensive experiments and ablation studies across multiple benchmarks and model families demonstrate that our self-rewarding SMC significantly improves pretrained MDLMs in terms of both sample quality and diversity, unlocking an effective and principled inference-time scaling dimension for diffusion-based language generation.

Acknowledgements

This research was partially supported by *Kjell och Märta Beijer Foundation* and by the project *Blending probabilistic and nonlinear representations* (contract number: 2025-04318), funded by the Swedish Research Council. Partial computational resources were provided by the National Academic Infrastructure for Supercomputing in Sweden (NAISS), funded by the Swedish Research Council.

References

[1] Marianne Arriola, Subham Sekhar Sahoo, Aaron Gokaslan, Zhihan Yang, Zhixuan Qi, Jiaqi Han, Justin T Chiu, and Volodymyr Kuleshov. Block diffusion: Interpolating between autoregressive and diffusion language models. In *The Thirteenth International Conference on Learning Representations*, 2025. URL `https://openreview.net/forum?id=tyEyYT267x`.

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

[3] 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.

[4] Tiwei Bie, Maosong Cao, Kun Chen, Lun Du, Mingliang Gong, Zhuochen Gong, Yanmei Gu, Jiaqi Hu, Zenan Huang, Zhenzhong Lan, et al. Llada2. 0: Scaling up diffusion language models to 100b. *arXiv preprint arXiv:2512.15745*, 2025.

[5] Andreas Blattmann, Tim Dockhorn, Sumith Kulal, Daniel Mendelevitch, Maciej Kilian, Dominik Lorenz, Yam Levi, Zion English, Vikram Voleti, Adam Letts, et al. Stable video diffusion: Scaling latent video diffusion models to large datasets. *arXiv preprint arXiv:2311.15127*, 2023.

[6] Gabriel Cardoso, Yazid Janati El Idrissi, Sylvain Le Corff, and Eric Moulines. Monte carlo guided diffusion for bayesian linear inverse problems. *arXiv preprint arXiv:2308.07983*, 2023.

[7] Huiwen Chang, Han Zhang, Lu Jiang, Ce Liu, and William T Freeman. Maskgit: Masked generative image transformer. In *Proceedings of the IEEE/CVF conference on computer vision and pattern recognition*, pages 11315–11325, 2022.

[8] Meihua Dang, Jiaqi Han, Minkai Xu, Kai Xu, Akash Srivastava, and Stefano Ermon. Inference-time scaling of diffusion language models with particle gibbs sampling. *arXiv preprint arXiv:2507.08390*, 2025.

[9] Sumanth Dathathri, Andrea Madotto, Janice Lan, Jane Hung, Eric Frank, Piero Molino, Jason Yosinski, and Rosanne Liu. Plug and play language models: A simple approach to controlled text generation. *arXiv preprint arXiv:1912.02164*, 2019.

[10] Pierre Del Moral. Feynman-kac formulae. In *Feynman-Kac Formulae: Genealogical and Interacting Particle Systems with Applications*, pages 47–93. Springer, 2004.

[11] Pierre Del Moral, Arnaud Doucet, and Ajay Jasra. Sequential monte carlo samplers. *Journal of the Royal Statistical Society Series B: Statistical Methodology*, 68(3):411–436, 2006.

[12] Prafulla Dhariwal and Alexander Nichol. Diffusion models beat gans on image synthesis. *Advances in neural information processing systems*, 34:8780–8794, 2021.

[13] Zehao Dou and Yang Song. Diffusion posterior sampling for linear inverse problem solving: A filtering perspective. In *The Twelfth International Conference on Learning Representations*, 2024.

[14] Arnaud Doucet, Nando De Freitas, and Neil Gordon. An introduction to sequential monte carlo methods. In *Sequential Monte Carlo methods in practice*, pages 3–14. Springer, 2001.

[15] Aaron Gokaslan and Vanya Cohen. Openwebtext corpus. `http://Skylion007.github.io/OpenWebTextCorpus`, 2019.

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

[17] Shuyang Gu, Dong Chen, Jianmin Bao, Fang Wen, Bo Zhang, Dongdong Chen, Lu Yuan, and Baining Guo. Vector quantized diffusion model for text-to-image synthesis. In *Proceedings of the IEEE/CVF conference on computer vision and pattern recognition*, pages 10696–10706, 2022.

[18] Xiaochuang Han, Sachin Kumar, and Yulia Tsvetkov. Ssd-lm: Semi-autoregressive simplex-based diffusion language model for text generation and modular control. In *Proceedings of the 61st Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, pages 11575–11596, 2023.

[19] Michael Hersche, Samuel Moor-Smith, Thomas Hofmann, and Abbas Rahimi. Soft-masked diffusion language models. *arXiv preprint arXiv:2510.17206*, 2025.

[20] Jonathan Ho and Tim Salimans. Classifier-free diffusion guidance. *arXiv preprint arXiv:2207.12598*, 2022.

[21] Jonathan Ho, Ajay Jain, and Pieter Abbeel. Denoising diffusion probabilistic models. *Advances in neural information processing systems*, 33:6840–6851, 2020.

[22] Jonathan Ho, Tim Salimans, Alexey Gritsenko, William Chan, Mohammad Norouzi, and David J Fleet. Video diffusion models. *Advances in neural information processing systems*, 35: 8633–8646, 2022.

[23] Iris AM Huijben, Wouter Kool, Max B Paulus, and Ruud JG Van Sloun. A review of the gumbel-max trick and its extensions for discrete stochasticity in machine learning. *IEEE transactions on pattern analysis and machine intelligence*, 45(2):1353–1371, 2022.

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

[25] Seo Hyun Kim, Sunwoo Hong, Hojung Jung, Youngrok Park, and Se-Young Yun. Klass: Kl-guided fast inference in masked diffusion models. *arXiv preprint arXiv:2511.05664*, 2025.

[26] Sunwoo Kim, Minkyu Kim, and Dongmin Park. Test-time alignment of diffusion models without reward over-optimization. *arXiv preprint arXiv:2501.05803*, 2025.

[27] Cheuk Kit Lee, Paul Jeha, Jes Frellsen, Pietro Lio, Michael Samuel Albergo, and Francisco Vargas. Debiasing guidance for discrete diffusion with sequential monte carlo. *arXiv preprint arXiv:2502.06079*, 2025.

[28] Runze Liu, Junqi Gao, Jian Zhao, Kaiyan Zhang, Xiu Li, Biqing Qi, Wanli Ouyang, and Bowen Zhou. Can 1b llm surpass 405b llm? rethinking compute-optimal test-time scaling. *arXiv preprint arXiv:2502.06703*, 2025.

[29] Zhiyuan Liu, Yicun Yang, Yaojie Zhang, Junjie Chen, Chang Zou, Qingyuan Wei, Shaobo Wang, and Linfeng Zhang. dllm-cache: Accelerating diffusion large language models with adaptive caching. *arXiv preprint arXiv:2506.06295*, 2025.

[30] Aaron Lou, Chenlin Meng, and Stefano Ermon. Discrete diffusion modeling by estimating the ratios of the data distribution. In *Forty-first International Conference on Machine Learning*, 2024. URL `https://openreview.net/forum?id=CNicRIVIPA`.

[31] João Loula, Benjamin LeBrun, Li Du, Ben Lipkin, Clemente Pasti, Gabriel Grand, Tianyu Liu, Yahya Emara, Marjorie Freedman, Jason Eisner, et al. Syntactic and semantic control of large language models via sequential monte carlo. *arXiv preprint arXiv:2504.13139*, 2025.

[32] Nanye Ma, Shangyuan Tong, Haolin Jia, Hexiang Hu, Yu-Chuan Su, Mingda Zhang, Xuan Yang, Yandong Li, Tommi Jaakkola, Xuhui Jia, et al. Inference-time scaling for diffusion models beyond scaling denoising steps. *arXiv preprint arXiv:2501.09732*, 2025.

[33] Shen Nie, Fengqi Zhu, Chao Du, Tianyu Pang, Qian Liu, Guangtao Zeng, Min Lin, and Chongxuan Li. Scaling up masked diffusion models on text. *arXiv preprint arXiv:2410.18514*, 2024.

[34] 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.

[35] Zijing Ou, Chinmay Pani, and Yingzhen Li. Inference-time scaling of discrete diffusion models via importance weighting and optimal proposal design. *arXiv preprint arXiv:2505.22524*, 2025.

[36] William Peebles and Saining Xie. Scalable diffusion models with transformers. In *Proceedings of the IEEE/CVF international conference on computer vision*, pages 4195–4205, 2023.

[37] Rafael Rafailov, Archit Sharma, Eric Mitchell, Christopher D Manning, Stefano Ermon, and Chelsea Finn. Direct preference optimization: Your language model is secretly a reward model. *Advances in neural information processing systems*, 36:53728–53741, 2023.

[38] Christian P Robert, George Casella, and George Casella. *Monte Carlo statistical methods*, volume 2. Springer, 1999.

[39] Robin Rombach, Andreas Blattmann, Dominik Lorenz, Patrick Esser, and Björn Ommer. High-resolution image synthesis with latent diffusion models. In *Proceedings of the IEEE/CVF conference on computer vision and pattern recognition*, pages 10684–10695, 2022.

[40] Chitwan Saharia, William Chan, Saurabh Saxena, Lala Li, Jay Whang, Emily L Denton, Kamyar Ghasemipour, Raphael Gontijo Lopes, Burcu Karagol Ayan, Tim Salimans, et al. Photorealistic text-to-image diffusion models with deep language understanding. *Advances in neural information processing systems*, 35:36479–36494, 2022.

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

[42] Yair Schiff, Subham Sekhar Sahoo, Hao Phung, Guanghan Wang, Sam Boshar, Hugo Dallatorre, Bernardo P de Almeida, Alexander Rush, Thomas Pierrot, and Volodymyr Kuleshov. Simple guidance mechanisms for discrete diffusion models. *arXiv preprint arXiv:2412.10193*, 2024.

[43] Jiaxin Shi, Kehang Han, Zhe Wang, Arnaud Doucet, and Michalis Titsias. Simplified and generalized masked diffusion for discrete data. *Advances in neural information processing systems*, 37:103131–103167, 2024.

[44] Raghav Singhal, Zachary Horvitz, Ryan Teehan, Mengye Ren, Zhou Yu, Kathleen McKeown, and Rajesh Ranganath. A general framework for inference-time scaling and steering of diffusion models. In *Forty-second International Conference on Machine Learning*, 2025. URL `https://openreview.net/forum?id=Jp988ELppQ`.

[45] Charlie Snell, Jaehoon Lee, Kelvin Xu, and Aviral Kumar. Scaling llm test-time compute optimally can be more effective than scaling model parameters. *arXiv preprint arXiv:2408.03314*, 2024.

[46] Jascha Sohl-Dickstein, Eric Weiss, Niru Maheswaranathan, and Surya Ganguli. Deep unsupervised learning using nonequilibrium thermodynamics. In *International conference on machine learning*, pages 2256–2265. pmlr, 2015.

[47] Masatoshi Uehara, Yulai Zhao, Chenyu Wang, Xiner Li, Aviv Regev, Sergey Levine, and Tommaso Biancalani. Inference-time alignment in diffusion models with reward-guided generation: Tutorial and review. *arXiv preprint arXiv:2501.09685*, 2025.

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

[49] Chengyue Wu, Hao Zhang, Shuchen Xue, Shizhe Diao, Yonggan Fu, Zhijian Liu, Pavlo Molchanov, Ping Luo, Song Han, and Enze Xie. Fast-dllm v2: Efficient block-diffusion llm. *arXiv preprint arXiv:2509.26328*, 2025.

[50] 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.

[51] Luhuan Wu, Brian Trippe, Christian Naesseth, David Blei, and John P Cunningham. Practical and asymptotically exact conditional sampling in diffusion models. *Advances in Neural Information Processing Systems*, 36:31372–31403, 2023.

[52] 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.

[53] Lvmin Zhang, Anyi Rao, and Maneesh Agrawala. Adding conditional control to text-to-image diffusion models. In *Proceedings of the IEEE/CVF international conference on computer vision*, pages 3836–3847, 2023.

[54] Kaiwen Zheng, Yongxin Chen, Hanzi Mao, Ming-Yu Liu, Jun Zhu, and Qinsheng Zhang. Masked diffusion models are secretly time-agnostic masked models and exploit inaccurate categorical sampling. *arXiv preprint arXiv:2409.02908*, 2024.

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

A Sequential Monte Carlo for Diffusion Reverse Process

A.1 Incremental Importance Weights

Consider a sequence of tokens $\mathbf{x}_T, \ldots, \mathbf{x}_0$ in the diffusion reverse path, with unnormalized intermediate target distribution $\tilde{\pi}_t(\mathbf{x}_{t:T})$. We define the following Markov sequential proposal

$$q_t(\mathbf{x}_{t:T}) = q_T(\mathbf{x}_T) \prod_{k=t}^{T-1} q_k(\mathbf{x}_k \mid \mathbf{x}_{t+1}), \tag{14}$$

so that it satisfies a standard factorization:

$$q_{t-1}(\mathbf{x}_{t-1:T}) = q_t(\mathbf{x}_{t:T})\, q_{t-1}(\mathbf{x}_{t-1} \mid \mathbf{x}_t). \tag{15}$$

The importance weight of trajectory $\mathbf{x}_{t:T}$ (from the full-mask tokens to a partial masked state) at time t is given by

$$\tilde{W}_t(\mathbf{x}_{t:T}) = \frac{\tilde{\pi}_t(\mathbf{x}_{t:T})}{q_t(\mathbf{x}_{t:T})}. \tag{16}$$

Likewise, at next step, we could write

$$\begin{aligned}
\tilde{W}_{t-1}(\mathbf{x}_{t-1:T}) &= \frac{\tilde{\pi}_{t-1}(\mathbf{x}_{t-1:T})}{q_{t-1}(\mathbf{x}_{t-1:T})} && (17)\\
&= \frac{\tilde{\pi}_{t-1}(\mathbf{x}_{t-1:T})}{q_t(\mathbf{x}_{t:T})\, q_{t-1}(\mathbf{x}_{t-1} \mid \mathbf{x}_t)} && (18)\\
&= \underbrace{\frac{\tilde{\pi}_{t-1}(\mathbf{x}_{t-1:T})}{\tilde{\pi}_t(\mathbf{x}_{t:T})\, q_{t-1}(\mathbf{x}_{t-1} \mid \mathbf{x}_t)}}_{=\tilde{w}_{t-1}(\mathbf{x}_{t-1:T})} \cdot \underbrace{\frac{\tilde{\pi}_t(\mathbf{x}_{t:T})}{q_t(\mathbf{x}_{t:T})}}_{=\tilde{W}_t(\mathbf{x}_{t:T})}, && (19)
\end{aligned}$$

where $\tilde{W}_t(\mathbf{x}_{t:T})$ is the previous weight as in Eq. (16), and $\tilde{w}_{t-1}(\mathbf{x}_{t-1:T})$ is the incremental importance weights in Eq. (4). Conceptually, $\tilde{w}_{t-1}(\mathbf{x}_{t-1:T})$ is the local ratio that updates the global path-wise importance weight when extending trajectory with more unmasked tokens i.e., from $\mathbf{x}_{t:T}$ to $\mathbf{x}_{t-1:T}$. In practice, SMC maintains particle weights recursively via Eq. (19) and performs resampling using normalized version of $\tilde{W}_{t-1}(\mathbf{x}_{t:T})$.

A.2 Proof for Confidence-based Sequential Monte Carlo

Proposition 3.1. *Given a pretrained diffusion model p_θ, let $\{\tilde{\pi}_t(\mathbf{x}_{t:T})\}_{t=0}^T$ denote the unnormalized path measures defined by the recursion in Eq. (11). If the sequential proposal in SMC is chosen to be the diffusion transition kernel, i.e., $q_{t-1}(\mathbf{x}_{t-1} \mid \mathbf{x}_t) = K_t(\mathbf{x}_t, \mathbf{x}_{t-1})$, then the incremental importance weights at step $t-1$ are given by*

$$\tilde{w}_{t-1}(\mathbf{x}_{t-1:T}) = \prod_{j \in S_t} \mathbf{c}_t(j), \tag{20}$$

where $\mathbf{c}_t(j) \coloneqq p_\theta\big(\hat{\mathbf{x}}_0(j) \mid \mathbf{x}_t\big)$ is the token confidence and $\mathcal{S}_t$ denotes the selected mask subset to be updated at step t.

Proof. Recall that in sequential Monte Carlo, the incremental importance weight at step $t-1$ is defined as

$$\tilde{w}_{t-1}(\mathbf{x}_{t-1:T}) = \frac{\tilde{\pi}_{t-1}(\mathbf{x}_{t-1:T})}{\tilde{\pi}_t(\mathbf{x}_{t:T})\, q_{t-1}(\mathbf{x}_{t-1} \mid \mathbf{x}_t)}, \tag{21}$$

Substituting the path recursion in Eq. (11) into the numerator yields

$$\tilde{w}_{t-1}(\mathbf{x}_{t-1:T}) = \frac{\tilde{\pi}_t(\mathbf{x}_{t:T})\, K_t(\mathbf{x}_t, \mathbf{x}_{t-1})\, G_{t-1}(\mathbf{x}_t, \mathbf{x}_{t-1})}{\tilde{\pi}_t(\mathbf{x}_{t:T})\, q_{t-1}(\mathbf{x}_{t-1} \mid \mathbf{x}_t)}. \tag{22}$$

Removing $\tilde{\pi}_t(\mathbf{x}_{t:T})$ in both numerator and denominator, we obtain

$$\tilde{w}_{t-1}(\mathbf{x}_{t-1:T}) = \frac{K_t(\mathbf{x}_t, \mathbf{x}_{t-1})\, G_{t-1}(\mathbf{x}_t, \mathbf{x}_{t-1})}{q_{t-1}(\mathbf{x}_{t-1} \mid \mathbf{x}_t)}. \tag{23}$$

When the proposal distribution $q_{t-1}(\mathbf{x}_{t-1} \mid \mathbf{x}_t)$ is selected to be the same as the diffusion transition kernel, i.e, $q_{t-1}(\mathbf{x}_{t-1} \mid \mathbf{x}_t) = K_t(\mathbf{x}_t, \mathbf{x}_{t-1})$, the transition kernel disappears, and

$$\tilde{w}_{t-1}(\mathbf{x}_{t-1:T}) = G_{t-1}(\mathbf{x}_t, \mathbf{x}_{t-1}). \tag{24}$$

And recall that we define the potential as the joint probability of accepted tokens within set $\mathcal{S}_t$, i.e., $G_{t-1}(\mathbf{x}_t, \mathbf{x}_{t-1}) = \prod_{j \in S_t} p_\theta(\mathbf{x}_{t-1}(j) \mid \mathbf{x}_t)$, and that $\mathbf{c}_t(j) \coloneqq p_\theta\big(\hat{\mathbf{x}}_0(j) \mid \mathbf{x}_t\big)$, we finally have the following

$$\tilde{w}_{t-1}(\mathbf{x}_{t-1:T}) = \prod_{j \in S_t} p_\theta(\mathbf{x}_{t-1}(j) \mid \mathbf{x}_t) \tag{25}$$

$$= \prod_{j \in S_t} \mathbf{c}_t(j), \tag{26}$$

which completes the proof. □

B Limitation and Future Work

In general, the practical motivation of SR-SMC is consistent with prior work in LLMs: scaling test-time compute optimally can be more effective than simply scaling model parameters [45, 28]. Within the dLLM setting, this is particularly relevant in quality-critical tasks such as code generation, offline generation, and benchmark evaluation, where users may prefer a slower but more reliable answer. Beyond direct inference, SR-SMC can also be used to generate higher-quality synthetic outputs or pseudo-targets from a pretrained dLLM, which may further benefit post-training, distillation, or preference optimization.

While the proposed self-rewarding SMC provides a principled framework for trajectory-level confidence-guided sampling, there are several worth-noting limitations. First, our method increases inference-time computation by running multiple diffusion processes in parallel. This trade-off is inherent to inference-time scaling methods and can be controllable by adjusting the number of particles. Second, the proposed trajectory confidence relies solely on model likelihood. While this choice is generic and task-agnostic, it does not explicitly optimize for downstream objectives such as reasoning correctness or human preference. In our future work, we plan to explore more informed proposals, such as look-ahead or twisted diffusion transitions, to further improve sampling efficiency and quality.

C Additional Experiments

C.1 Additional Results on Text Generation

In Table 2 and Figure 6, we primarily use generative perplexity to evaluate the sample quality of open-ended text generation. Here, we additionally report the entropy of generated samples in Table 9 to

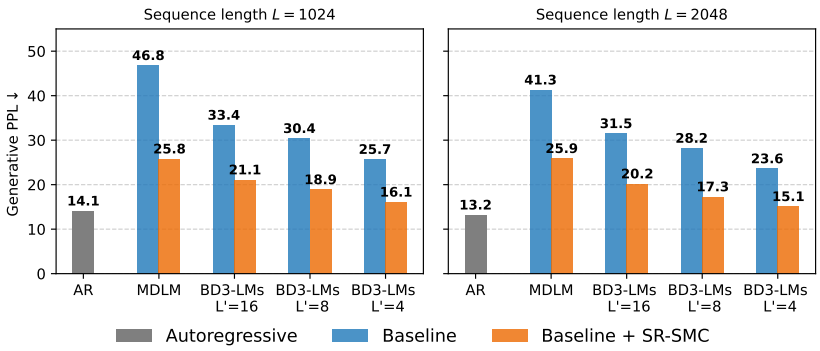


Figure 6: Generative perplexity (↓) results on 300 samples of lengths $L = 1024, 2048$ using our self-rewarding SMC and the corresponding baselines.

examine whether SR-SMC improves Gen. PPL by collapsing to low-diversity outputs. For reference, the Gen. PPL and entropy of the original data are 14.8 and 5.44, respectively, as reported in Wang et al. [48].

Overall, SR-SMC substantially reduces Gen. PPL while keeping the entropy at a comparable level to the corresponding baselines and the original data. For example, on MDLM with $L = 1024$, SR-SMC reduces Gen. PPL from 46.8 to 25.8, while the entropy only changes from 5.33 to 5.15. For BD3-LMs, SR-SMC consistently improves Gen. PPL across different block sizes, with entropy values remaining around 5.2 for $L = 1024$ and 5.5 for $L = 2048$. These values are close to the entropy of the original data and the AR baseline, suggesting that SR-SMC improves sample quality without causing mode collapse or overly deterministic generation.

Table 9: Generative perplexity (Gen. PPL; $\downarrow$) and entropy results of 300 samples of lengths $L = 1024, 2048$. All models are trained on OWT dataset. For reference, the Gen. PPL and entropy of the original data are 14.8 and 5.44, respectively, as reported in Wang et al. [48].

		$L = 1024$		$L = 2048$	
Model		Gen. PPL	Entropy	Gen. PPL	Entropy
AR		14.1	5.29	13.2	5.54
MDLM		46.8	5.33	41.3	5.64
MDLM *w/ Best-of-N*		38.6	5.31	40.3	5.61
MDLM *w/ ReMDM*		35.5	5.35	45.1	5.27
MDLM *w/ SR-SMC*		25.8	5.15	25.9	5.41
BD3-LMs	$L' = 16$	33.4	5.34	31.5	5.59
	$L' = 8$	30.4	5.33	28.2	5.63
	$L' = 4$	25.7	5.33	23.6	5.62
BD3-LMs *w/ SR-SMC*	$L' = 16$	21.1	5.19	20.2	5.46
	$L' = 8$	18.9	5.18	17.3	5.45
	$L' = 4$	16.1	5.20	15.1	5.49

C.2 Detailed Results of Inference with Gumbel Noise

We provide the detailed results of diffusion sampling with different Gumbel noise temperatures τ ranging from 0.0 to 1.0 in Table 10. We also note that the Fast-dLLM implementation may introduce small numerical perturbations under multi-GPU parallel decoding. Therefore, for the deterministic setting $\tau = 0$, we treat SR-SMC and the corresponding baseline as using the same deterministic decoding result by default. This ensures that the comparison focuses on the effect of SR-SMC under stochastic sampling rather than implementation-level perturbations. Overall, our self-rewarding SMC (SR-SMC) consistently outperforms the baseline LLaDA-1.5 [55] and Dream-7B [52] over a wide range of temperatures on both MBPP and MATH benchmarks. Notably, the Dream-7B baseline is highly sensitive to the noise temperature, with performance degrading severely when τ is slightly increased to 0.1 or 0.2. In contrast, SR-SMC substantially improves robustness across different temperature settings, suggesting that particle-based exploration and trajectory-level resampling can mitigate the brittleness of standard diffusion sampling.

Table 10: Effect of Gumbel noise temperature τ on MBPP and MATH.

Benchmark	Method	$\tau = 0.0$	$\tau = 0.1$	$\tau = 0.2$	$\tau = 0.3$	$\tau = 0.4$	$\tau = 0.5$	$\tau = 0.6$	$\tau = 0.7$	$\tau = 0.8$	$\tau = 0.9$	$\tau = 1.0$
MBPP	LLaDA-1.5	42.6	42.2	42.8	42.8	41.0	42.0	40.2	42.6	40.6	41.6	40.4
	LLaDA-1.5 *w/ SR-SMC*	-	**43.2**	**43.0**	42.6	**43.4**	**45.2**	**43.2**	**44.0**	**43.6**	**43.4**	**44.2**
	Dream-7B	48.2	1.80	5.60	14.2	28.0	34.0	39.0	41.2	41.4	41.8	42.0
	Dream-7B *w/ SR-SMC*	-	**43.4**	**46.8**	**47.4**	**48.4**	**48.6**	**47.4**	**48.4**	**48.6**	**48.0**	**48.6**
MATH	LLaDA-1.5	39.8	39.8	38.0	37.6	40.0	38.4	38.8	37.2	38.8	38.2	38.2
	LLaDA-1.5 *w/ SR-SMC*	-	39.2	**41.0**	**42.0**	40.0	**39.4**	**39.2**	**41.4**	**39.6**	**40.6**	**41.8**
	Dream-7B	42.4	2.40	10.4	22.2	29.2	38.0	39.8	38.0	41.0	42.4	41.6
	Dream-7B *w/ SR-SMC*	-	**42.0**	**44.6**	**42.4**	**44.2**	**45.0**	**43.0**	**43.8**	**44.6**	**46.2**	**45.2**

C.3 Compute-Matched Comparison on Diffusion Large Language Models

We compare SR-SMC with compute-matched inference-time scaling baselines on Dream-7B, including Best-of-N, MCMC, and majority voting. More specifically, the Best-of-N algorithm generates

N independent particles in parallel using the same diffusion denoising process as SR-SMC, but without resampling. The particle with the highest cumulative log-probability is selected as the final output. MCMC generates a complete sequence using the standard diffusion process. Then, for each refinement step, it randomly selects one block, re-masks it entirely, and regenerates it conditioned on the current content of all other blocks. Note that all these additional compute-matched experiments are re-evaluated on 8 NVIDIA H200 GPUs for a fair comparison. As shown in Table 11, Best-of-N, MCMC, and majority voting improve the baseline average score from 53.2 to 55.5, 53.9, and 54.2, respectively, while SR-SMC achieves the best average score of 57.5. In particular, SR-SMC provides large gains on HumanEval and MATH, suggesting that trajectory-level reweighting and resampling are more effective than simply selecting from independent samples or aggregating final answers. We also compare SR-SMC with classifier-free guidance (CFG) [20] under a smaller compute budget using two particles. As shown in Table 12, SR-SMC with two particles achieves an average score of 53.8, outperforming CFG with guidance scale 1.5, which obtains 52.3.

Table 11: Compute-matched comparison on diffusion large language models Dream-7B.

Method	GSM8K	MATH	HumanEval	MBPP	Average
Dream-7B (baseline)	77.3	42.2	50.3	43.2	53.2
Best-of-N	77.4	44.6	54.3	45.8	55.5
MCMC	76.9	44.2	54.3	40.2	53.9
Majority vote	81.9	37.6	50.0	47.4	54.2
SR-SMC	78.9	45.4	60.7	45.2	**57.5**

Table 12: Classifier-free guidance on Dream-7B.

Method	GSM8K	MATH	HumanEval	MBPP	Average
CFG (1.5)	75.4	40.4	50.3	43.0	52.3
SR-SMC (2 particles)	76.3	43.6	54.3	41.0	53.8

C.4 Practical Efficiency

We further evaluate the practical inference cost of SR-SMC on LLaDA-1.5. We compare standard decoding, Best-of-N, and SR-SMC under identical hardware and decoding settings. Best-of-N and SR-SMC both use four particles, allowing us to isolate the additional overhead introduced by trajectory-level reweighting and resampling. We report the average wall-clock latency per example, generation throughput, and the corresponding task accuracy.

As shown in Table 13, running four particles naturally increases the inference cost relative to single-particle decoding. However, compared with the compute-matched Best-of-N baseline, SR-SMC introduces only a modest additional wall-clock overhead while consistently achieving higher accuracy. At generation length 256, SR-SMC increases latency from 6.18 to 6.31 seconds on GSM8K and from 5.32 to 5.45 seconds on HumanEval, while improving accuracy from 81.8 to 82.2 and from 40.2 to 41.7, respectively. Similar trends are observed at generation length 512. These results indicate that most of the inference cost arises from evaluating multiple particles, whereas the additional SMC operations provide measurable performance gains with relatively small wall-clock overhead.

Table 13: Quality-efficiency comparison of LLaDA-1.5. Best-of-N and SR-SMC use the same four-particle budget. Latency is measured under identical hardware and decoding settings.

		GSM8K			HumanEval		
Method	Length	Latency (s)↓	Tok/s↑	Acc.↑	Latency (s)↓	Tok/s↑	Acc.↑
Baseline	256	1.96	107.81	79.5	2.81	31.33	37.7
Best-of-N	256	6.18	34.46	81.8	5.32	15.52	40.2
SR-SMC	256	6.31	32.84	**82.2**	5.45	14.46	**41.7**
Baseline	512	2.97	69.31	80.6	4.42	20.32	33.8
Best-of-N	512	10.50	18.01	80.7	13.08	10.83	39.0
SR-SMC	512	11.36	15.85	**82.5**	13.79	7.18	**40.7**

C.5 Additional Zero-Shot Results

In the main text, we report zero-shot results with generation length $L = 256$. Here, we provide the complete zero-shot evaluation with both $L = 256$ and $L = 512$ in Table 14. Overall, SR-SMC consistently improves both LLaDA-1.5 and Dream-7B across GSM8K and MATH under different generation lengths. For $L = 512$, SR-SMC improves the average score of LLaDA-1.5 from 59.7 to 63.4, and Dream-7B from 63.4 to 64.8. These results further confirm that the benefit of trajectory-level confidence remains stable under longer generation lengths and zero-shot prompting.

Table 14: Zero-shot performance of LLaDA-1.5 and Dream-7B variants on GSM8K and MATH benchmarks across different generation lengths.

Benchmark	Length	LLaDA-1.5	*w/ SR-SMC*	Dream-7B	*w/ SR-SMC*
GSM8K	256	75.4	**76.3**	77.9	**81.1**
(0-shot)	512	80.4	**82.0**	78.0	78.0
MATH	256	35.2	**36.6**	44.2	**49.2**
(0-shot)	512	39.0	**44.8**	48.8	**51.6**

C.6 Exploring Beyond Confidence-Based Framework

All dLLM experiments in the main text are built on top of the Fast-dLLM framework, as described in Section 4. Thus, our reported results already show that SR-SMC can be combined with a stronger decoding framework using KV cache and parallel decoding, rather than only with naive confidence-based sampling. To further examine its compatibility with more advanced dLLM samplers, we additionally apply SR-SMC on top of KL-Guided fast inference (KLASS) [25] using LLaDA-1.5. As shown in Table 15, KLASS + SR-SMC improves the average score from 49.84 to 51.01, with gains on GSM8K, MATH, and HumanEval, and no degradation on MBPP. These results suggest that SR-SMC is a general particle-based inference-time scaling method that can be integrated with stronger dLLM decoding frameworks.

Table 15: Applying SR-SMC to KLASS framework.

Method	GSM8K	MATH	HumanEval	MBPP	Average
KLASS	77.48	34.2	40.24	47.47	49.85
KLASS + SR-SMC	78.92	35.0	42.68	47.47	51.02

C.7 ESS Analysis

We further analyze the effective sample size (ESS) of SR-SMC to examine whether particle degeneracy occurs during sampling. In our implementation, adaptive resampling is checked once per block rather than at every decoding step. This allows each particle to follow its own trajectory over a longer horizon, making the accumulated trajectory-level confidence more informative for resampling than purely step-wise token confidence. Table 16 reports the final ESS values using $N = 4$ particles across different tasks. The ESS values range from 2.65 to 3.52, remaining well above 1, which suggests that the final particle population does not collapse to a single dominant particle. These results indicate that SR-SMC can improve generation quality while still maintaining a reasonable level of particle diversity.

Table 16: The final ESS values using SR-SMC (LLaDA-1.5 with 4 particles) across tasks.

Task	GSM8K	MATH	HumanEval	MBPP
ESS value	2.79	3.52	2.65	3.05

C.8 Additional Examples

We provide additional qualitative examples comparing greedy decoding with SR-SMC in Figure 7. These examples illustrate that greedy decoding can commit to locally confident but incorrect intermediate steps, leading to persistent reasoning or calculation errors. In contrast, SR-SMC maintains

multiple candidate trajectories and uses trajectory-level confidence for resampling, which helps recover more globally coherent solutions.

Qualitative Comparison: Arithmetic Reasoning

Problem: Simplify $\frac{1}{5} \cdot \frac{8}{7} \div \frac{12}{20}$.
Reference Answer: $\frac{8}{21}$

Greedy Decoding ($\tau = 0$)

First, simplify the multiplication of the fractions:

$$\frac{1}{5} \cdot \frac{8}{7} = \frac{8}{7}$$

(Calculation Hallucination: ignored $\frac{1}{5}$)

Next, simplify the division of the fractions:

$$\frac{8}{7} \div \frac{12}{20} = \frac{8}{7} \cdot \frac{20}{12}$$

Now, multiply the numerators and denominators:

$$\frac{8 \cdot 20}{7 \cdot 12} = \frac{160}{84}$$

Simplify the fraction:

$$\frac{160 \div 40}{84 \div 40} = \frac{4}{7}$$

Final Answer: $\boxed{\frac{4}{7}}$ ✗

w/ SR-SMC ($\tau = 1, N = 4$)

To simplify $\frac{1}{5} \cdot \frac{8}{7} \div \frac{12}{20}$:
1. **Multiply the fractions first:**

$$\frac{1}{5} \cdot \frac{8}{7} = \frac{1 \cdot 8}{5 \cdot 7} = \frac{8}{35}$$

2. **Divide the fractions:**

$$\frac{8}{35} \div \frac{12}{20} = \frac{8}{35} \cdot \frac{20}{12} = \frac{160}{420}$$

3. **Simplify the fraction:**

$$\frac{160}{420} = \frac{8}{21}$$

Final Answer: $\boxed{\frac{8}{21}}$ ✓

Qualitative Comparison: Distance Calculation

Problem: At 50 miles per hour, how far would a car travel in $2\frac{3}{4}$ hours? Express your answer as a mixed number.
Reference Answer: $137\frac{1}{2}$

Greedy Decoding ($\tau = 0$)

To determine how far a car would travel in $2\frac{3}{4}$ hours at 50 miles per hour, use:

$$\text{Distance} = \text{Speed} \times \text{Time}$$

First, convert the mixed number:

$$2\frac{3}{4} = \frac{1}{4}$$

(Severe Conversion Error)
Next, multiply speed by time:

$$\text{Distance} = 50 \times \frac{1}{4} = 12.5$$

Finally, convert to a mixed number:

$$12.5 = 12\frac{1}{2}$$

Final Answer: $\boxed{12\frac{1}{2}}$ ✗

w/ SR-SMC ($\tau = 1, N = 4$)

To determine how far a car would travel in $2\frac{3}{4}$ hours at 50 miles per hour, use:

$$\text{Distance} = \text{Speed} \times \text{Time}$$

Convert the mixed number:

$$2\frac{3}{4} = \frac{8}{4} + \frac{3}{4} = \frac{11}{4}$$

Multiply speed by time:

$$\text{Distance} = 50 \times \frac{11}{4} = \frac{550}{4} = 137.5$$

Convert to a mixed number:

$$137.5 = 137\frac{1}{2}$$

Final Answer: $\boxed{137\frac{1}{2}}$ ✓

Figure 7: Additional qualitative comparisons of reasoning trajectories. Greedy decoding can commit to locally confident but incorrect intermediate steps, while SR-SMC maintains multiple candidate trajectories and recovers globally coherent solutions.

NeurIPS Paper Checklist

1. **Claims**

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

   Answer: [Yes]

   Justification: See the abstract and introduction sections.

   Guidelines:

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

2. **Limitations**

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

   Answer: [Yes]

   Justification: See appendix B.

   Guidelines:

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

3. **Theory assumptions and proofs**

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

   Answer: [Yes]

Justification: See the method section and appendix.

Guidelines:

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

4. **Experimental result reproducibility**

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

   Answer: [Yes]

   Justification: See experiments and the code in supplementary materials.

   Guidelines:

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

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

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

Answer: [Yes]

Justification: The code has been uploaded in the supplementary materials and will be released publicly.

Guidelines:

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

6. **Experimental setting/details**

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

Answer: [Yes]

Justification: See experiments and appendix. Our method consistently demonstrates performance improvements compared to baselines.

Guidelines:

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

7. **Experiment statistical significance**

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

Answer: [No]

Justification: All our experiments are done once.

Guidelines:

- The answer [N/A] means that the paper does not include experiments.
- The authors should answer [Yes] if the results are accompanied by error bars, confidence intervals, or statistical significance tests, at least for the experiments that support the main claims of the paper.
- The factors of variability that the error bars are capturing should be clearly stated (for example, train/test split, initialization, random drawing of some parameter, or overall run with given experimental conditions).
- The method for calculating the error bars should be explained (closed form formula, call to a library function, bootstrap, etc.)
- The assumptions made should be given (e.g., Normally distributed errors).

- It should be clear whether the error bar is the standard deviation or the standard error of the mean.
- It is OK to report 1-sigma error bars, but one should state it. The authors should preferably report a 2-sigma error bar than state that they have a 96% CI, if the hypothesis of Normality of errors is not verified.
- For asymmetric distributions, the authors should be careful not to show in tables or figures symmetric error bars that would yield results that are out of range (e.g., negative error rates).
- If error bars are reported in tables or plots, the authors should explain in the text how they were calculated and reference the corresponding figures or tables in the text.

8. **Experiments compute resources**

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

   Answer: [Yes]

   Justification: See the experiment section.

   Guidelines:

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

9. **Code of ethics**

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

   Answer: [Yes]

   Justification: We have reviewed the NeurIPS Code of Ethics.

   Guidelines:

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

10. **Broader impacts**

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

    Answer: [N/A]

    Justification: The proposed method only changes the sampling procedure of existing models.

    Guidelines:

    - The answer [N/A] means that there is no societal impact of the work performed.
    - If the authors answer [N/A] or [No], they should explain why their work has no societal impact or why the paper does not address societal impact.
    - Examples of negative societal impacts include potential malicious or unintended uses (e.g., disinformation, generating fake profiles, surveillance), fairness considerations (e.g., deployment of technologies that could make decisions that unfairly impact specific groups), privacy considerations, and security considerations.

- The conference expects that many papers will be foundational research and not tied to particular applications, let alone deployments. However, if there is a direct path to any negative applications, the authors should point it out. For example, it is legitimate to point out that an improvement in the quality of generative models could be used to generate Deepfakes for disinformation. On the other hand, it is not needed to point out that a generic algorithm for optimizing neural networks could enable people to train models that generate Deepfakes faster.
- The authors should consider possible harms that could arise when the technology is being used as intended and functioning correctly, harms that could arise when the technology is being used as intended but gives incorrect results, and harms following from (intentional or unintentional) misuse of the technology.
- If there are negative societal impacts, the authors could also discuss possible mitigation strategies (e.g., gated release of models, providing defenses in addition to attacks, mechanisms for monitoring misuse, mechanisms to monitor how a system learns from feedback over time, improving the efficiency and accessibility of ML).

11. **Safeguards**

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

    Answer: [No]

    Justification: No such risk.

    Guidelines:

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

12. **Licenses for existing assets**

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

    Answer: [Yes]

    Justification: Data and models are properly cited in our paper.

    Guidelines:

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

- If this information is not available online, the authors are encouraged to reach out to the asset's creators.

13. **New assets**

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

    Answer: [Yes]

    Justification: [N/A]

    Guidelines:

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

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

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

    Answer: [N/A]

    Justification: not involved in this paper.

    Guidelines:

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

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

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

    Answer: [N/A]

    Justification: [N/A]

    Guidelines:

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

16. **Declaration of LLM usage**

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

Answer: [N/A]

Justification: We didn't use LLMs for any core component.

Guidelines:

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