RETHINKING REASONING WITH MASKED DIFFUSION MODELS

Anonymous authors
Paper under double-blind review

ABSTRACT

Masked diffusion language models (MDLMs) are trained to infill positions in randomly masked sequences, in contrast to traditional next-token prediction (NTP) models. Discussions around MDLMs focus on two benefits: (1) multi-token decoding and 2) any-order decoding. However, we observe that for math and coding tasks, any-order algorithms often underperform or behave similarly to *left-to-right sampling*, and standard multi-token decoding significantly degrades performance. At inference time, MDLMs compute the conditional distribution of all masked positions. A natural question is: *How can we justify this additional compute when left-to-right one-token-at-a-time decoding is on par with any-order decoding algorithms?* These findings warrant rethinking how MDLMs are utilized. First, we propose multi-token entropy decoding (MED), a simple adaptive sampler that minimizes the error incurred by decoding positions in parallel based on the conditional entropies of those positions. MED preserves performance across benchmarks and leads to $3\times$ fewer steps. Second, we propose a *reasoning-as-infilling* framework. By using MDLMs to infill a reasoning template, we can structure outputs and distinguish between reasoning and answer tokens. In turn, this enables measuring answer uncertainty *during* reasoning. This enables early exits when the model converges on an answer. Combined with MED, this leads to a 69% speed-up on GSM8K with a minimal (0.1%) effect on accuracy. Finally, given an answer, our framework enables sampling from the posterior over reasoning traces *conditioned on the answer*, even when the model is incorrect. On GSM8K, this enables generating correct reasoning traces for 43% of problems originally solved incorrectly. Our work demonstrates that the training objective and compute used by MDLMs unlock many new possibilities for inference and post-training methods.

1 INTRODUCTION

The current dominant approach for language modeling is based on next-token prediction (NTP) training. NTP language models learn the conditional distribution of the *next token*, given the previous tokens in a sequence (Shannon, 1951; Radford et al., 2019). The resulting language model is sampled auto-regressively left-to-right, one token at a time. Recent work proposes MDLMs (Austin et al., 2021; Sahoo et al., 2024; Shi et al., 2024) as an alternative to NTP models. MDLMs are trained to in-fill sequences with randomly masked positions. The resulting model learns the distribution $p_\theta(x^i|\mathbf{x}_{\text{UN-MASKED}})$ at every masked position i.

While modeling all masked positions requires additional compute, MDLMs have several potential benefits, such as parallel token decoding (Sahoo et al., 2024; 2025), and flexible decoding orders (Kim et al., 2025) that lead to significant improvements on logic puzzles, such as Sudoku. Additionally, Bachmann & Nagarajan (2024); Prabhudesai et al. (2025) show that multi-token prediction objectives can achieve better likelihoods and accuracy on tasks, and access to the distribution and samples from masked positions supports controllable generation (Schiff et al., 2024; Singhal et al., 2025).

In our work, we first examine two purported benefits of MDLMs: multi-token and any-order decoding, on mathematical reasoning and coding benchmarks. Despite the flexibility enabled by MDLMs, we observe that decoding one token in a left-to-right order, identically to an NTP model, is a strong decoding choice for MDLM models. Even decoding just two tokens in parallel substantially reduces performance on popular benchmarks. These findings raise questions about the substantial extra compute MDLMs spend to model the distribution of all masked positions.

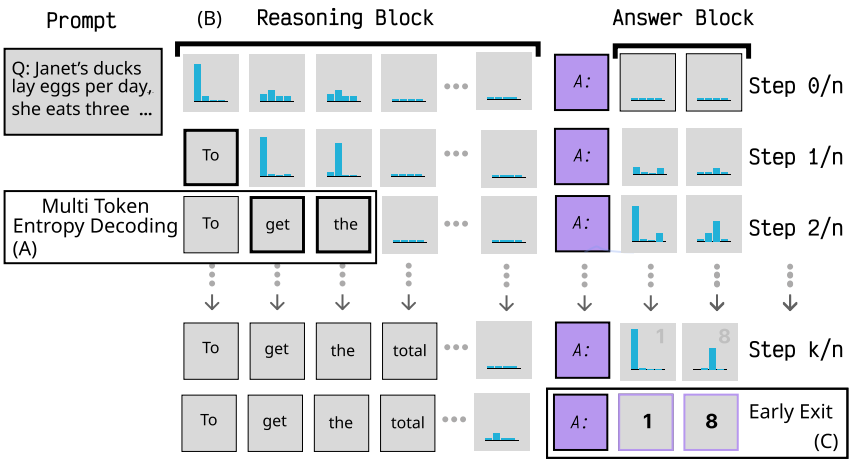

Figure 1: MDLMs learn the conditional distributions at each masked token position. A) We utilize the entropy of these distributions to adaptively set the number of tokens decoded at each step. B) We also reframe reasoning as infilling a prompted reasoning template, which enables directly modeling answer token probabilities *during* reasoning. This provides several benefits, like C) enabling early exits and also *post-hoc* reasoning given a pre-filled answer.

In this work, we show how this compute can be made *useful*. We demonstrate that (a) the access that MDLMs provide to the conditional distributions of all masked positions and (b) their ability to in-fill, unlocks new sampling and post-training capabilities that are not readily available for NTP models.

First, we revisit multi-token decoding. Decoding multiple positions in a single step results in samples that are not from the MDLM's learned distribution, as typically $p_\theta(x^i, x^j \mid \mathbf{x}_{\text{UN-MASKED}}) \neq p_\theta(x^i \mid \mathbf{x}_{\text{UN-MASKED}}) p_\theta(x^j \mid \mathbf{x}_{\text{UN-MASKED}})$. However, by making use of the entropy of the masked positions to inform decoding, we can control how much multi-token decoding deviates from single token sampling. We propose Multi-token Entropy Decoding (MED), an **adaptive multi-token decoder** that decodes multiple positions only if the conditional entropy of those positions falls below a specified threshold. We find that MED leads to 2-3$\times$ fewer function calls, with a minor or no drop in performance.

Next, we demonstrate that the ability of MDLM to in-fill opens up new model prompting paradigms. In this work, we propose prompting-as-infilling, where we add the user-specified context in multiple positions, not just the beginning of the sequence, unlike NTP models. Specifically, we consider *reasoning-as-infilling*. Here we pre-fill an explicit reasoning template, with specific reasoning and answer positions (see fig. 1). This enables explicit control over the length and format of the reasoning trace. We demonstrate that infilling this template provides many other significant advantages. By explicitly distinguishing token answer positions, we make use of the conditional distributions of the masked positions provided by MDLMs to measure the uncertainty of the answer *while reasoning*. In turn, this enables **early exits** once the model converges on an answer, further reducing inference costs.

Additionally, *reasoning-as-infilling* has consequences for analyzing model behavior and improving performance. Given access to an answer, we can sample from the MDLM's posterior distribution of reasoning traces conditioned on the answer, $p_\theta(\mathbf{r} \mid \mathbf{c}, \mathbf{a})$. This enables generating high-quality *post-hoc* reasoning traces for use in model fine-tuning.

Contributions.

- **Examining MDLM sampling benefits.** We evaluate MDLM models, such as Dream (Wu et al., 2025) and LLaDA (Nie et al., 2025), on several tasks. We find that the any-order sampling capability of MDLM provides limited benefits, and that standard multi-token decoding degrades performance.

- **Multi-token Entropy Decoding**: We propose MED, an adaptive sampler that provides a $2-3\times$ speed-up, without any loss in performance on math and coding benchmarks.
- **Reasoning-as-Infilling.** We propose *reasoning-as-infilling* for MDLMs, which leverages their infilling capabilities. We then show that distinguishing reasoning and answer tokens can provide several benefits, such as:
  - *Early exits*, which lead to a $3.3\times$ speed when combined with MED.
  - *Post-hoc reasoning with* MDLMs, where given question-answer pairs, we generate reasoning traces conditioned on the answer. On the GSM8K dataset, sampling from the posterior distribution finds correct reasoning traces for 43% of the questions the original model failed to solve. Given answers, we can also estimate the correctness of the MDLM reasoning process at intermediate steps without an external verifier.

2 Related Work

Multi-Token Prediction and Speculative Decoding. Gloeckle et al. (2024) show that models trained with the multi-token objective can enable parallel multi-token decoding, or *speculative decoding*, without making use of another model. However, unlike MDLMs, Gloeckle et al. (2024) limit to predicting the next $2, 4$ tokens. Several other works (Leviathan et al., 2023; Chen et al., 2023) show that using smaller draft models for generation and then rejection sampling can also enable parallel decoding with NTP models. However, MDLMs offer many possibilities beyond left-to-right parallel decoding, such as in-filling, and error correction through re-masking of unmasked tokens (Wang et al., 2025). Israel et al. (2025) propose an adaptive multi-token decoder which samples from the product of an NTP and MDLM model. Unlike MED, their approach relies on rejection sampling based on an external NTP model.

Post-hoc reasoning. Zelikman et al. (2022) proposes generating and fine-tuning on reasoning traces with language models that are generated given a correct answer. Phan et al. (2023); Ruan et al. (2025) propose fine-tuning a model on samples from approximations of the posterior $p_\theta(\mathbf{r} \mid \mathbf{c}, \mathbf{a})$. In contrast, MDLMs enable exact sampling from the posterior of the reasoning traces given the answer by simply in-filling the answer in the answer block provided in the *reasoning-as-infilling* framework.

Concurrent Work. Concurrent to our work, Ben-Hamu et al. (2025) propose entropy-bound (EB) sampler, an adaptive multi-token decoder, which similar to MED controls the error incurred by multi-token decoding. EB sampler adds multiple positions to unmask based on the difference between the sum of the positions added and the maximum entropy until the difference exceeds a specified threshold γ. MED unmasks positions based on the individual entropies rather than thresholding based on the sum. We observe that MED leads to fewer NFEs while getting higher accuracies, see tables 6 and 7 in Ben-Hamu et al. (2025) versus table 4 for a comparison.

3 Masked Diffusion Language Models

MDLMs (Sohl-Dickstein et al., 2015; Austin et al., 2021; Sahoo et al., 2024; Shi et al., 2024) are a class of generative models for modeling discrete data. These models are trained by taking data $\mathbf{x} \sim q_{\text{data}}$, where $\mathbf{x}$ is an arbitrary length sequence where each position takes values in a finite vocabulary $\mathcal{V}$. For training, the sequence $\mathbf{x} \sim q_{\text{data}}$ is masked randomly, and the model learns to predict the distributions of the masked positions for a fixed length sequence:

$$\max_{\theta} \frac{1}{|\text{MASKED-SET}|} \sum_{j \in \text{MASKED-SET}} \log p_\theta(x^j \mid \mathbf{x}_{\text{un-masked}}, \mathbf{c}) \tag{1}$$

where $\mathbf{c}$ is the context and MASKED-SET is the set of positions that have been masked. Sampling from an MDLM is performed by iteratively un-masking positions. Notably, an MDLM can also be viewed as any-order auto-regressive model (Uria et al., 2014; Ou et al., 2024), where given a decoding order $\mathbf{o} = [o_1, \ldots, o_\ell]$ with $o_j \in \{1, 2, \ldots, \ell\}$, the sampling model can be defined as:

$$p_\theta(\mathbf{x} \mid \mathbf{c}, \mathbf{o}) = \prod_{j=0}^{\ell-1} p_\theta(x^{o(j)} \mid \mathbf{x}_{o(<j)}, \mathbf{c}) \tag{2}$$

where o_j refers to the position decoded at step j and $\mathbf{x}_{o(<j)}$ refers to all positions decoded prior to step j. Additionally, block-sampling (Sahoo et al., 2024; Nie et al., 2025; Arriola et al., 2025) approaches define a left-to-right sequence of fixed length blocks and decode within each block in an arbitrary order.

3.1 Preliminary Observations

In our work, we first examine two purported benefits of MDLMs: *any-order* and *multi-token* decoding, on popular mathematical reasoning, GSM8K (Cobbe et al., 2021a) and MATH500 (Luo et al., 2024), and coding benchmarks, HumanEval (Chen et al., 2021) as well as Sudoku (Shahab, 2025):

1. **Does any-order decoding help for text?** Popular sampling approaches for MDLMs select positions to unmask based on confidence (e.g. token probability (Chang et al., 2022) or entropy (Kim et al., 2025; Ye et al., 2025b)). We find that these any-order decoding algorithms either sample a large portion of tokens in a left-to-right order or underperform left-to-right sampling. For example, on GSM8K, the best configuration of any-order entropy decoding samples $\sim 50\%$ of tokens left-to-right. Without block sizes (Arriola et al., 2025) that enforce a semi-auto-regressive (AR) left-to-right structure, any-order significantly affects performance, see table 1. A notable exception where any-order sampling provides a significant benefit is Sudoku. We include additional analysis in appendix A.
2. **Does parallel decoding work?** We observe that even decoding two tokens in parallel at a time can severely hurt model performance across all tasks, see table 2. The resulting distributions also have high KL with respect to a one-token sampling algorithm, see table 3.

These findings show that the decoding order from NTP models is performant for MDLM models, despite their any-order and multi-token decoding capacities. Despite these findings, we show that the additional compute and capacities offered by MDLMs have many possible benefits.

Decoding order	**GSM8K**		**Math500**		**HumanEval**		**Sudoku (4)**	
	Llada	Dream	Llada	Dream	Llada	Dream	Llada	Dream
Left-to-right	75.96	74.98	29.4	29.2	15.24	**53.65**	36.13	17.28
Any-order decoding	53.44	34.11	12.0	20.6	10.97	32.92	**47.64**	**61.26**
Block any-order (8)	76.95	75.73	**33.4**	**29.6**	**16.46**	37.17	38.74	44.50
Block any-order (32)	**78.01**	**75.81**	32.4	28.2	15.24	48.17	**47.64***	**61.26***
Block any-order (64)	74.14	57.69	30.0	28.2	14.63	47.56	-	-
Llama3.1-8B	70.81		26.80		62.20		2.09	

Table 1: Left-to-right sampling is a competitive sampling algorithm for reasoning and coding. When performing entropy decoding (Ye et al., 2025b), we observe that full any-order sampling results in poor performance on all tasks but Sudoku. Left-to-right block decoding is required to make any-order sampling performant, and left-to-right sampling (block size $= 1$) is always within a few percent of the best configuration. We also observe in appendix A that performant block any-order configurations sample a large portion of tokens left-to-right.*For Sudoku, we consider sequences of length 32, otherwise we use a sequence length of 128.

Parallel Tokens	**GSM8K**		**Math500**		**HumanEval**		**Sudoku**	
	Llada	Dream	Llada	Dream	Llada	Dream	Llada	Dream
1	**76.95**	**75.73**	**33.4**	**29.6**	**16.46**	**51.82**	47.64	**61.26**
2	62.31	57.69	19.6	16.6	4.87	20.12	**50.79**	57.59
4	33.58	28.50	7.0	3.6	4.87	12.19	29.32	42.93

Table 2: MDLMs can generate multiple fixed tokens in parallel, but this degrades accuracy. We decode $1, 2, 4$ tokens in parallel, with block any-order (8) entropy decoding. We note that decoding even two tokens in parallel leads to a significant drop on all tasks but Sudoku.

4 NO COMPUTE LEFT BEHIND

MDLMs are trained to in-fill sequences by modeling the distributions $p_\theta(x^j \mid \mathbf{x}_{\text{un-masked}}, \mathbf{c})$ for masked positions $j \in$ MASK-SET given un-masked text $\mathbf{x}_{\text{un-masked}}$ and a context $\mathbf{c}$.

Typically, MDLMs are prompted similarly to NTP models, and the additional compute spent on these position distributions is only used for sampling a *fixed* number of positions. The remaining distributions are discarded. In this work, we show that the ability of MDLMs to in-fill and to access the distribution of *all* masked positions unlocks many new sampling and post-training capabilities.

- **Multi-token Entropy Decoding**. We introduce MED, an adaptive multi-token decoding algorithm that controls the error incurred by multi-token decoding by decoding multiple positions only if the conditional entropies of the decoded positions falls below a threshold.
- **Reasoning-as-Infilling for Control, Early Exits, and Post-Training Benefits**. We propose infilling a user-specified prompt in multiple parts of the sequence. Specifically for reasoning tasks, we first pre-fill a reasoning template that differentiates between reasoning and answer positions, then infill with the MDLM model. This method of prompting enables controlling the length of the reasoning process, and measuring the uncertainty of the answer block during the reasoning process for early exiting. We also demonstrate how this approach supports new post-training directions for MDLMs.

Assumptions. We assume that the masked conditional distributions learned by the MDLM model define a consistent joint distribution (Majid et al., 2025).

4.1 MULTI-TOKEN ENTROPY DECODING

As MDLMs learn the conditional distribution $p_\theta(x^j \mid \mathbf{x}_{\text{UN-MASKED}})$ for all masked tokens, they support unmasking multiple tokens in parallel. However, decoding even two positions, x^i and x^j in parallel can result in samples that may not be likely under the MDLM joint distribution $p_\theta(\mathbf{x})$, as typically $p_\theta(x^i, x^j \mid \mathbf{x}_{\text{UN-MASKED}}) \neq p_\theta(x^i \mid \mathbf{x}_{\text{UN-MASKED}}) p_\theta(x^j \mid \mathbf{x}_{\text{UN-MASKED}})$. In table 2, we observe that decoding even 2 tokens in parallel hurts performance.

However, for any set of positions $A \subseteq$ MASK-SET $\subseteq \{1, \ldots, \ell\}$, we can upper bound the Kullback-Leibler (KL) divergence between the joint distribution $p_\theta(\mathbf{x}^A \mid \mathbf{x}_{\text{un-masked}}, \mathbf{c})$ and the factorized distribution $\prod_{i \in A} p_\theta(x^i \mid \mathbf{x}_{\text{un-masked}}, \mathbf{c})$ with the sum of the entropies of the masked tokens:

$$\text{KL}\left(p_\theta(\mathbf{x}^A \mid \mathbf{x}_{\text{un-masked}}, \mathbf{c}) \middle| \prod_{i \in A} p_\theta(x^i \mid \mathbf{x}_{\text{un-masked}}, \mathbf{c}) \right) \leq \sum_{i \in A} H(x^i \mid \mathbf{x}_{\text{un-masked}}, \mathbf{c}) \tag{3}$$

where H is the entropy of the distribution $p_\theta(x^i \mid \mathbf{x}_{\text{un-masked}}, \mathbf{c})$. For a proof, see appendix E.

In this work, we propose multi-token entropy decoding, which makes use of the entropies of the masked positions x^j to decide whether to decode multiple positions in parallel. Given un-masked text $\mathbf{x}_{\text{un-masked}}$, a decoding threshold λ and a maximum number of tokens to be decoded k, we propose two definitions of the set A for selecting positions to un-mask:

- MED: We sort the position entropies in an ascending order and decode positions with that satisfy $H(x^i \mid \mathbf{x}_{\text{un-masked}}, \mathbf{c}) < \lambda$ and select k such tokens. If no position has entropy lower than λ, we choose the position with the lowest entropy.
- AR-MED: We decode at most k tokens in a contiguous left-to-right order for positions that satisfy $H(x^i \mid \mathbf{x}_{\text{un-masked}}, \mathbf{c}) < \lambda$, or the left most position if no position has entropy below λ.

Both MED and AR-MED allow for upper bounding the KL divergence in eq. (3) by λk.

4.2 REASONING-AS-INFILLING WITH MDLMS

Generally, NTP models are controlled at inference-time with a prompt prefix that is inserted at the beginning of the sequence. However, for MDLMs we propose pre-filling the output sequence with user-specified tokens. In the case of reasoning tasks, where a model produces a reasoning trace prior to answering, we can pre-fill the output sequence with a reasoning template that distinguishes the reasoning and answer token positions:

$$\Big[\underbrace{[\text{MASK}]_1 \quad [\text{MASK}]_2 \quad \dots \quad [\text{MASK}]_k}_{\text{reasoning block}} \quad \text{<Answer Delimiter>} \quad \underbrace{[\text{MASK}]_{k+1} \quad \dots \quad [\text{MASK}]_\ell}_{\text{answer block}}\Big]$$

Here the answer delimiter is a user-specified choice (e.g. *"The answer is: "* for math tasks, or function definitions for a coding task). In this reformulation of prompting, the context $\mathbf{c}$ now includes both the prompt and the answer delimiter, see fig. 1.

This prompting technique offers several advantages. For instance, we can control the amount of compute spent on reasoning before generating an answer. More critically, by distinguishing between reasoning and answer tokens, we can:

- Measure the uncertainty of the answer block while reasoning, allowing for early exits when the model converges on an answer.
- With access to an answer, sample from the posterior distribution of the reasoning traces conditioned on the answer. This enables generating reasoning traces for post-training.

Early exits with the answer entropy upper-bound. By having explicit positions for the answer block and modeling the distribution of those positions, we can measure the uncertainty of the answer *while reasoning*. Access to the distributions $p_\theta(a^i \mid \mathbf{r}_{\text{un-masked}}, \mathbf{c})$, allows us to compute the conditional entropy of those positions. We define the quantity:

$$H_{\text{UB}}(\mathbf{a} \mid \mathbf{r}_{\text{UN-MASKED}}, \mathbf{c}) := \sum_{j \in \{\text{ANSWER-BLOCK}\}} H(a^j \mid \mathbf{r}_{\text{UN-MASKED}}, \mathbf{c}). \tag{4}$$

While this quantity, in general, is not equal to the conditional entropy $H(\mathbf{a} \mid \mathbf{r}_{\text{UN-MASKED}}, \mathbf{c})$, H_{UB} upper-bounds it.

$$H(\mathbf{a} \mid \mathbf{r}_{\text{UN-MASKED}}, \mathbf{c}) \leq H_{\text{UB}}(\mathbf{a} \mid \mathbf{r}_{\text{UN-MASKED}}, \mathbf{c}). \tag{5}$$

See appendix E for a proof. When H_{UB} falls below a user-specified threshold λ, we can exit the reasoning process and generate the answer, reducing the inference requirements. Additionally, we observe that answer uncertainty can provide some information about problem difficulty.

Generating reasoning traces conditioned on the answer. In this section, we demonstrate how the in-filling capacity and the reasoning template can enable new training capacities for MDLMs. Phan et al. (2023); Ruan et al. (2025) propose training a model p_θ to maximize the probability of an answer $\mathbf{a}$ given a context $\mathbf{c}$, by maximizing the marginal log-likelihood:

$$\log p_\theta(\mathbf{a} \mid \mathbf{c}) = \log \int_{\mathbf{r}} p_\theta(\mathbf{a}, \mathbf{r} \mid \mathbf{c}) d\mathbf{r}. \tag{6}$$

While it is intractable to estimate this integral exactly, its gradient can be estimated as:

$$\nabla_\theta \log p_\theta(\mathbf{a} \mid \mathbf{c}) = \mathbf{E}_{p_\theta(\mathbf{r} \mid \mathbf{a}, \mathbf{c})} \nabla_\theta \log p_\theta(\mathbf{a}, \mathbf{r} \mid \mathbf{c}). \tag{7}$$

Sampling from the posterior $p_\theta(\mathbf{r} \mid \mathbf{a}, \mathbf{c})$ is intractable for NTP models, therefore, Phan et al. (2023); Zelikman et al. (2022) make use of approximate posterior sampling methods, which require significant prompting engineering to yield reasoning traces given answer hints.

With MDLMs, using the *reasoning-as-infilling* template, one can *simply* in-fill the answer block positions, enabling exact sampling from the posterior distribution over the reasoning traces, assuming that the masked conditional distributions define a consistent joint distribution.

5 Experiments

In this section, we first study the effectiveness of multi-token entropy decoding for parallel sampling. Then, we examine the inference-time and post-training benefits of the proposed *reasoning-as-infilling* framework, such as (1) early-exits based on answer certainty $H_{\text{UB}}(\mathbf{a} \mid \mathbf{r}_{\text{un-masked}}, \mathbf{c})$ and (2) the ability to bootstrap high-quality reasoning traces given (question, answer) pairs.

Dataset	Sampler	Model	Accuracy ↑	KL ↓	NFEs ↓
GSM8K	$\sigma_{\text{ENTROPY},k=1}$	LLaDa	**78.01**	0.0	128.0
		Dream	**75.81**	0.0	128.0
	$\sigma_{\text{ENTROPY},k=2}$	LLaDa	36.24	92.6	64.0
		Dream	19.79	99.5	64.0
	$\sigma_{\text{MED},\lambda=0.1}$	LLaDa	78.01	0.5	88.4
		Dream	75.81	0.4	92.5
	$\sigma_{\text{MED},\lambda=0.2}$	LLaDa	78.01	1.5	84.8
		Dream	75.82	1.9	79.9
	$\sigma_{\text{MED},\lambda=0.3}$	LLaDa	77.86	2.7	81.2
		Dream	75.44	3.3	75.7
HumanEval	$\sigma_{\text{ENTROPY},k=1}$	LLaDa	15.24	0.0	128.0
		Dream	**48.17**	0.0	128.0
	$\sigma_{\text{ENTROPY},k=2}$	LLaDa	4.87	85.5	64.0
		Dream	4.87	77.0	64.0
	$\sigma_{\text{MED},\lambda=0.1}$	LLaDa	15.24	0.7	70.0
		Dream	48.17	0.5	68.5
	$\sigma_{\text{MED},\lambda=0.2}$	LLaDa	15.85	2.0	61.8
		Dream	48.17	1.5	60.4
	$\sigma_{\text{MED},\lambda=0.3}$	LLaDa	**16.46**	3.6	57.8
		Dream	**48.17**	2.2	57.0

Table 3: MED enables parallel token decoding without any loss in performance. We compare MED decoding with different λ thresholds to entropy decoding with a fixed number of tokens $k \in \{1, 2\}$. We observe that MED significantly reduces the number of NFEs while matching accuracy and maintaining a low KL.

5.1 Accelerated sampling with Multi-token Entropy Decoding

In this experiment, we show that entropy-thresholded multi-token decoding (MED) enables parallel decoding without incurring the distributional error and performance degradation that forced multi-token decoding incurs.

For these experiments, we use two open-source MDLMs models, Dream 7B Instruct (Ye et al., 2025a) and LLaDA 8B Instruct (Nie et al., 2025). We consider two popular benchmarks, (1) GSM8K (Lightman et al., 2023), a mathematical reasoning dataset, and (2) HumanEval (Chen et al., 2021), a coding benchmark. As baselines, we consider the entropy decoding scheme $\sigma_{\text{ENTROPY},k}$ (Chang et al., 2022; Ye et al., 2025b), which decodes a fixed number of k tokens in each step. We consider $k \in \{1, 2\}$.

For evaluations, we measure the task accuracy and the number of function evaluations (NFEs) for varying values of k in the fixed token decoding scheme as well as varying values of $\lambda \in \{0.1, 0.2, 0.3\}$ in MED with $k = 32$ as the maximum number of tokens decoded in parallel. We fix a generation length $\ell = 128$ and a block size of 32. Additionally, we also measure the KL divergence between the likelihoods of the single-token decoding scheme with the multi-token decoding schemes. The likelihood for each decoding scheme $\sigma = (\sigma_1, \sigma_2, \ldots, \sigma_n)$ is computed as $\prod_{t=1}^{n} p_\theta(x_{\sigma_t} \mid \mathbf{c}, \mathbf{x}_{\sigma_{<t}})$ where $n \leq t$ and σ_t denotes the position(s) decoded at step t. When decoding multiple tokens in a single step, we use the conditionally independent factorization.

In table 3, we observe:

- Decoding just $k = 2$ tokens in parallel results in a large drop in accuracy on GSM8K for both LLaDA and Dream ($> 40\%$). We observe that decoding $k = 2$ also leads to a significant increase in KL.

- MED with $\lambda = 0.2$, provides significant speed-ups and *no loss in accuracy* for both LLaDA and Dream. For HUMANEVAL, MED results in identical accuracy with a $2.2\times$ speed-up, and on GSM8K, we observe a $1.5\times$ speed-up with no loss in performance. In contrast to fixed decoding with $k = 2$ tokens, the KL divergence is substantially lower even with fewer NFEs, and scaling λ offers a trade-off between fewer NFEs and KL.

5.2 THE BENEFITS OF *Reasoning-as-Infilling*

Early exits. We investigate the inference-time benefits of the proposed *reasoning-as-infilling* framework on two mathematical reasoning datasets, GSM8K (Lightman et al., 2023) and MATH500 (Cobbe et al., 2021b), with the Dream 7B and LLaDA 8B models.

For both tasks, we consider the sequence length $\ell = 256$ with block size 32. We pre-fill the answer delimiter *"The answer is \boxed{.."*, and allocate 10 answer tokens. As a baseline, we compare against allocating a sequence of length 256 with no-reasoning template. For sampling, we examine early exits with one-token decoding and both MED and AR-MED with $\lambda = 0.2$.

In table 4, we observe:

- For both Dream and LLaDA, early exiting reduces the total number of NFEs, and increasing the early exit threshold γ enables trading faster inference for task accuracy. For example, for LLaDA, we observe a 23% speed up on one-token entropy decoding with only a $< 1\%$ drop in performance versus baseline reasoning template. Early exits combined with MED and AR-MED provide further savings. LLaDa with MED and $\gamma = 0.1$ outperforms the base configuration on GSM8K with a $3.3\times$ speedup.
- Notably, the benefits of early exits are more pronounced for LLaDA than Dream, which requires higher exit thresholds for speedups. This may be due to Dream's adaption from an NTP model (Gong et al., 2024; Ye et al., 2025a). See appendix A for a discussion of the sampling behavior of Dream and LLaDA.

		GSM8K			Math500		
Model	**Sampler**	**Exit Param**	**NFEs**	**Acc.**	**Exit Param**	**NFEs**	**Acc.**
LLaDA	$\sigma_{\text{ENTROPY},k=1}$	NO TEMPLATE	256	76.6	NO TEMPLATE	256	**33.8**
LLaDA	$\sigma_{\text{ENTROPY},k=1}$	NO EXIT	256	79.4	NO EXIT	256	33.4
LLaDA	$\sigma_{\text{ENTROPY},k=1}$	$\gamma = 0.1$	193	78.6	$\gamma = 0.3$	221	31.9
LLaDA	$\sigma_{\text{MED},\lambda=0.2}$	NO EXIT	94	**79.9**	NO EXIT	143	33.4
LLaDA	$\sigma_{\text{MED},\lambda=0.2}$	$\gamma = 0.1$	**77**	79.3	$\gamma = 0.3$	129	32.0
LLaDA	$\sigma_{\text{AR-MED},\lambda=0.2}$	NO EXIT	105	79.2	NO EXIT	143	33.4
LLaDA	$\sigma_{\text{AR-MED},\lambda=0.2}$	$\gamma = 0.1$	93	78.1	$\gamma = 0.3$	**128**	31.7
Dream	$\sigma_{\text{ENTROPY},k=1}$	NO TEMPLATE	256	**80.1**	NO TEMPLATE	256	33.4
Dream	$\sigma_{\text{ENTROPY},k=1}$	NO EXIT	256	79.8	NO EXIT	256	**35.6**
Dream	$\sigma_{\text{ENTROPY},k=1}$	$\gamma = 0.7$	225	76.7	$\gamma = 0.7$	245	33.2
Dream	$\sigma_{\text{MED},\lambda=0.2}$	NO EXIT	135	79.2	NO EXIT	147	35.6
Dream	$\sigma_{\text{MED},\lambda=0.2}$	$\gamma = 0.7$	**121**	79.3	$\gamma = 0.7$	141	35.4
Dream	$\sigma_{\text{AR-MED},\lambda=0.2}$	NO EXIT	148	77.1	NO EXIT	151	35.6
Dream	$\sigma_{\text{AR-MED},\lambda=0.2}$	$\gamma = 0.7$	131	73.9	$\gamma = 0.7$	**136**	35.0

Table 4: Early-exits can accelerate MDLM inference. We evaluate reasoning-as-infilling with early exits on a generation length of 256. Varying the early exit threshold γ enables trading faster inference for task accuracy. Lower values of γ preserve performance.

Next, we investigate how a dataset of question-answer pairs $\{(\mathbf{c}_i, \mathbf{a}_i)\}_{i=1}^N$, can be used to analyze and improve MDLMs. Due to limited compute, we generate and evaluate these traces, rather than use them for training.

The answer posterior is a source of high-quality reasoning traces. Here, we evaluate reasoning traces $\mathbf{r}$ generated from the posterior distribution $p_\theta(\mathbf{r} \mid \mathbf{c}, \mathbf{a})$. A key challenge for training better

Judge	Posterior Reasoning Scores
Qwen2.5-Math-PRM	38%
GPT-4o	43%

Table 5: The MDLM reasoning posterior yields high-quality traces for problems that the original model fails to solve. We perform posterior inference on the 1419 training samples that LLaDa with greedy decoding fails to solve, and evaluate the resulting traces with two judges, QWEN2.5-MATH-PRM (Zhang et al., 2025a) and GPT-4o (Hurst et al., 2024). Both models rate 40% of these reasoning chains as correct.

reasoning models is collecting high quality reasoning traces (Zelikman et al., 2022). We investigate whether the MDLM posterior distribution can provide these traces, even when an MDLM incorrectly solves the original task.

To do this, we utilize question-answer pairs from GSM8K (Lightman et al., 2023). In this experiment, we generate samples from the LLaDA-8B instruction fine-tuned model with MED (block size of 32). On the GSM8K training dataset, the model answers 1419 out of 7473 problems incorrectly. We use these question-answer pairs to generate reasoning traces from the MDLM posterior (i.e. with the answer pre-filled).

To evaluate these reasoning traces for correctness, we use GPT4o (Hurst et al., 2024), and the Qwen2.5-Math-7B PRM (Zhang et al., 2025a), see appendix C for the system instructions to the GPT4o model. We observe that both judge models rate $\sim 40\%$ of the posterior reasoning traces as correct. In appendix D, we include examples of reasoning traces generated from the posterior and the regular model with different judge labels. We observe that the posterior traces judged correct by GPT4o contain accurate reasoning steps, correcting the original model's behavior.

Scoring partial reasoning traces. (Lew et al., 2023; Singhal et al., 2025) show that intermediate rewards can be used to generate samples that score higher rewards. In appendix B, we show that MDLMs offer several methods for scoring whether the reasoning process will generate a correct answer, which could be utilized for these approaches.

6 DISCUSSION

Much of the current tooling around pre-training, post-training, and inference for text generation has been built around a key modeling choice: next-token prediction training. MDLMs are an expressive class of models trained to in-fill masked sequences, requiring additional training and inference compute. In our work, we find that this additional compute has many uses beyond just accelerating inference and warrants *rethinking* how these models are utilized. For instance, the ability to in-fill unlocks new prompting techniques, like the proposed reasoning-as-filling framework, along with new data generation and post-training methods. Large-scale language models trained on next-token prediction have revolutionized text generation. Our work provides evidence that models such MDLMs, trained on alternative objectives, unlock new capabilities not readily available to NTP approaches.

6.1 LIMITATIONS

State-of-the-art NTP language models achieve significant benefits from long reasoning chains (Guo et al., 2025; OpenAI, 2025). While the MDLM framework introduces new possibilities, MDLMs come with computational costs that make the current available models impractical for long-context tasks. Most notably, for a sequence of length ℓ, an NTP model needs to make ℓ predictions, while an MDLM model makes $\mathcal{O}(\ell^2)$ predictions. Moreover, current MDLMs do not naturally support inference-optimizations, such as caching, for any-order decoding. We are hopeful that our findings, which highlight the benefits and sampling behavior of current models, can help guide architectural decisions for future models that address these limitations.

REFERENCES

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

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.

Gregor Bachmann and Vaishnavh Nagarajan. The pitfalls of next-token prediction. *arXiv preprint arXiv:2403.06963*, 2024.

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.

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*, pp. 11315–11325, 2022.

Charlie Chen, Sebastian Borgeaud, Geoffrey Irving, Jean-Baptiste Lespiau, Laurent Sifre, and John Jumper. Accelerating large language model decoding with speculative sampling. *arXiv preprint arXiv:2302.01318*, 2023.

Mark Chen, Jerry Tworek, Heewoo Jun, Qiming Yuan, Henrique Ponde De Oliveira Pinto, Jared Kaplan, Harri Edwards, Yuri Burda, Nicholas Joseph, Greg Brockman, et al. Evaluating large language models trained on code. *arXiv preprint arXiv:2107.03374*, 2021.

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

Karl Cobbe, Vineet Kosaraju, Mohammad Bavarian, Mark Chen, Heewoo Jun, Lukasz Kaiser, Matthias Plappert, Jerry Tworek, Jacob Hilton, Reiichiro Nakano, et al. Training verifiers to solve math word problems. *arXiv preprint arXiv:2110.14168*, 2021b.

Fabian Gloeckle, Badr Youbi Idrissi, Baptiste Rozière, David Lopez-Paz, and Gabriel Synnaeve. Better & faster large language models via multi-token prediction. *arXiv preprint arXiv:2404.19737*, 2024.

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.

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

Daya Guo, Dejian Yang, Haowei Zhang, Junxiao Song, Ruoyu Zhang, Runxin Xu, Qihao Zhu, Shirong Ma, Peiyi Wang, Xiao Bi, et al. Deepseek-r1: Incentivizing reasoning capability in llms via reinforcement learning. *arXiv preprint arXiv:2501.12948*, 2025.

Aaron Hurst, Adam Lerer, Adam P Goucher, Adam Perelman, Aditya Ramesh, Aidan Clark, AJ Ostrow, Akila Welihinda, Alan Hayes, Alec Radford, et al. Gpt-4o system card. *arXiv preprint arXiv:2410.21276*, 2024.

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

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.

Yaniv Leviathan, Matan Kalman, and Yossi Matias. Fast inference from transformers via speculative decoding. In *International Conference on Machine Learning*, pp. 19274–19286. PMLR, 2023.

Alexander K Lew, Tan Zhi-Xuan, Gabriel Grand, and Vikash K Mansinghka. Sequential monte carlo steering of large language models using probabilistic programs. *arXiv preprint arXiv:2306.03081*, 2023.

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

Liangchen Luo, Yinxiao Liu, Rosanne Liu, Samrat Phatale, Meiqi Guo, Harsh Lara, Yunxuan Li, Lei Shu, Yun Zhu, Lei Meng, et al. Improve mathematical reasoning in language models by automated process supervision. *arXiv preprint arXiv:2406.06592*, 2024.

Mahbod Majid, Rattana Pukdee, Vishwajeet Agrawal, Burak Varıcı, and Pradeep Kumar Ravikumar. On the consistent recovery of joint distributions from conditionals. In *The 28th International Conference on Artificial Intelligence and Statistics*, 2025.

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

OpenAI. Introducing openai o3 and o4-mini. https://openai.com/index/introducing-o3-and-o4-mini/, April 2025. Accessed: 2025-09-24.

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

Du Phan, Matthew Douglas Hoffman, David Dohan, Sholto Douglas, Tuan Anh Le, Aaron Parisi, Pavel Sountsov, Charles Sutton, Sharad Vikram, and Rif A Saurous. Training chain-of-thought via latent-variable inference. *Advances in Neural Information Processing Systems*, 36:72819–72841, 2023.

Mihir Prabhudesai, Menging Wu, Amir Zadeh, Katerina Fragkiadaki, and Deepak Pathak. Diffusion beats autoregressive in data-constrained settings. *arXiv preprint arXiv:2507.15857*, 2025.

Alec Radford, Jeff Wu, Rewon Child, David Luan, Dario Amodei, and Ilya Sutskever. Language models are unsupervised multitask learners. 2019.

Yangjun Ruan, Neil Band, Chris J Maddison, and Tatsunori Hashimoto. Reasoning to learn from latent thoughts. *arXiv preprint arXiv:2503.18866*, 2025.

Subham Sekhar Sahoo, Marianne Arriola, Yair Schiff, Aaron Gokaslan, Edgar Marroquin, Justin T Chiu, Alexander Rush, and Volodymyr Kuleshov. Simple and effective masked diffusion language models, 2024.

Subham Sekhar Sahoo, Zhihan Yang, Yash Akhauri, Johnna Liu, Deepansha Singh, Zhoujun Cheng, Zhengzhong Liu, Eric Xing, John Thickstun, and Arash Vahdat. Esoteric language models. *arXiv preprint arXiv:2506.01928*, 2025.

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

Asad Shahab. sudokuLLM: Llm fine-tuning for sudoku solving. https://github.com/Asad-Shahab/sudokuLLM, 2025. Accessed: 2025-09-24.

Claude E Shannon. Prediction and entropy of printed english. *Bell system technical journal*, 30(1): 50–64, 1951.

Jiaxin Shi, Kehang Han, Zhe Wang, Arnaud Doucet, and Michalis K Titsias. Simplified and generalized masked diffusion for discrete data. *arXiv preprint arXiv:2406.04329*, 2024.

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, 2025. URL https://arxiv.org/abs/2501.06848.

Jascha Sohl-Dickstein, Eric Weiss, Niru Maheswaranathan, and Surya Ganguli. Deep unsupervised learning using nonequilibrium thermodynamics. In *International Conference on Machine Learning*, pp. 2256–2265. PMLR, 2015.

Benigno Uria, Iain Murray, and Hugo Larochelle. A deep and tractable density estimator. In *International Conference on Machine Learning*, pp. 467–475. PMLR, 2014.

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

Zirui Wu, Lin Zheng, Zhihui Xie, Jiacheng Ye, Jiahui Gao, Yansong Feng, Zhenguo Li, Victoria W., Guorui Zhou, and Lingpeng Kong. Dreamon: Diffusion language models for code infilling beyond fixed-size canvas, 2025. URL https://hkunlp.github.io/blog/2025/dreamon.

Jiacheng Ye, Zhihui Xie, Lin Zheng, Jiahui Gao, Zirui Wu, Xin Jiang, Zhenguo Li, and Lingpeng Kong. Dream 7b, 2025a. URL https://hkunlp.github.io/blog/2025/dream.

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*, 2025b.

Eric Zelikman, Yuhuai Wu, Jesse Mu, and Noah Goodman. Star: Bootstrapping reasoning with reasoning. *Advances in Neural Information Processing Systems*, 35:15476–15488, 2022.

Zhenru Zhang, Chujie Zheng, Yangzhen Wu, Beichen Zhang, Runji Lin, Bowen Yu, Dayiheng Liu, Jingren Zhou, and Junyang Lin. The lessons of developing process reward models in mathematical reasoning. *arXiv preprint arXiv:2501.07301*, 2025a.

Zhenru Zhang, Chujie Zheng, Yangzhen Wu, Beichen Zhang, Runji Lin, Bowen Yu, Dayiheng Liu, Jingren Zhou, and Junyang Lin. The lessons of developing process reward models in mathematical reasoning, 2025b. URL https://arxiv.org/abs/2501.07301.

A MDLM ANY-ORDER SAMPLING BEHAVIOR

We study the effects of greedy any-order entropy decoding (Ye et al., 2025a) for LLaDA (Nie et al., 2025) and Dream (Ye et al., 2025a), as well as any-order with different **block lengths** (Sahoo et al., 2024; Arriola et al., 2025; Nie et al., 2025; Ye et al., 2025a). The block length is the contiguous region of consecutive positions considered by the sampling algorithm, where the model can decode in any order. Blocks are unmasked left-to-right.

We included our results in table 1. On Sudoku, any-order sampling significantly improves performance.[1] However, for the remaining datasets, left-to-right sampling with a block length of 1 is a competitive approach. In some cases (e.g. for Dream on HUMANEVAL (Chen et al., 2021)), left-to-right block length 1 sampling is the most performant configuration. Additionally, purely any-order decoding (i.e. when the block size = generation length), leads to a massive drop in performance.

In what order are tokens decoded? In table 7 and appendix A, we analyze the behavior of these different configurations on a portion of GSM8K and HUMANEVAL. We compute the fraction of non-EOS tokens decoded from the leftmost masked position, the average distance from the leftmost position, and the total number of non-EOS tokens. For GSM8K, we also include the average step at which the answer appears in the decoded sequence.

Notably, top performing block-length configurations often behave very autoregressively. On GSM8K, when the block size is 32, both LLaDA and Dream sample the leftmost unmasked position approximately 50% of the time. Additionally, the average distance of the unmasked position from the left-most mask is approximately 3 tokens. Gong et al. (2025) similarly observe the left-to-right sampling behavior of Dream for coding.

Why are block lengths necessary? We find that that purely any-order decoding from current MDLMs results in less auto-regressive generation, fewer non-eos tokens, and very early answers, not utilizing the full allocated generation length. Reviewing samples from any-order decoding, we observe two specific pathological behaviors: 1) Models first greedily decoding low entropy *end-of-text* tokens, leading to shorter or empty texts that do not fully utilize the assigned tokens, and 2) decoding only an answer, or decoding answers first, *before* reasoning chains.

Config	Model	Acc.	% Leftmost	Dist. Left	Non-EOS Tokens	Answer Step
Block(1)	Dream	76.4%	100.0%	0.0	105.1	78.1
	LLaDA	79.0%	100.0%	0.0	115.3	84.3
Block(32)	Dream	77.6%	52.1%	2.9	103.1	76.3
	LLaDA	76.8%	47.1%	3.3	112.3	82.8
AO(128)	Dream	34.2%	73.1%	6.5	24.3	18.7
	LLaDA	53.4%	40.8%	20.2	75.0	16.9

Table 6: Decoding Behavior, GSM8K We evaluate the autoregressiveness of different sampling configurations by measuring the percent of non-EOS tokens decoded from the leftmost position, the average distance of these positions from left, the total number of non-EOS tokens, and at what timestep the answer is decoded. We consider generation lengths of 128 on a portion of GSM8K ($n = 500$)

B POSSIBILITIES FOR ANALYZING MODEL BEHAVIOR

Additionally, we observe that the conditional distributions of the answer block positions provide information about correctness *during* the reasoning process, see fig. 2. Without access to any answer, the conditional entropy upper bound $H_{\text{UB}}(\mathbf{a} \mid \mathbf{r}_{\text{un-masked}}, \mathbf{c})$ is weakly correlated with correctness (Pearson correlation of 0.28).

However, with gold labels $\mathbf{a}$, the conditional distributions learned by MDLM provide a "process reward model" that scores partial generations for correctness. During reasoning, the log probabilities

[1] Of note, on Sudoku, diffusion models with auto-regressive sampling significantly outperform Llama 8B. This may reflect benefits of the MDLM training objective.

Config	Model	Acc.	% Leftmost	Dist. Left	Non-EOS Tokens
Block(1)	Dream	53.7%	100.0%	0.0	95.0
	LLaDA	11.0%	100.0%	0.0	119.8
Block(32)	Dream	48.2%	43.1%	3.8	96.8
	LLaDA	15.2%	44.7%	4.1	119.5
AO(128)	Dream	32.9%	44.5%	6.8	56.4
	LLaDA	11.0%	29.7%	19.7	123.8

Table 7: Decoding Behavior, HumanEval Similar to table 7, we measure autoregressiveness for generation lengths of 128, on a portion of HumanEval ($n = 500$)

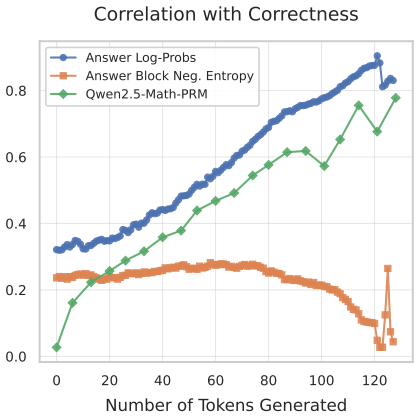

Figure 2: MDLMs enable scoring their own reasoning process without an external process verifier. We score GSM8K reasoning traces, generated left-to-right with LLaDA, at intermediate steps using (a) gold answer log probabilities, (b) the answer block entropy bound, and (c) and an 8B process reward model (PRM) (Zhang et al., 2025b). Gold answer probability at intermediate steps is more predictive of final correctness than PRM scores. Even without gold labels, the answer block entropy is weakly correlated with correctness.

of the gold answer tokens $\log p_\theta(a_j \mid \mathbf{c}, \mathbf{r}_{\text{un-masked}})$ are more strongly correlated with final answer correctness at intermediate steps than a pretrained process reward model (Zhang et al., 2025a).

Qualitatively, we observe that drops in these gold answer probabilities at intermediate steps can correspond to reasoning mistakes (See Appendix). This indicates that MDLM pre-training unlocks other new post-training capabilities: low-quality reasoning chains could terminated early, the reasoning process could be steered towards correction solutions, reflection tokens could be automatically inserted at reasoning failures, and new sources of dense feedback could incorporated into fine-tuning objectives.

C Evaluating Reasoning Trace Correctness

We evaluate reasoning trace correctness with GPT4o (Hurst et al., 2024). We use the following template to evaluate the reasoning chain:

```
SYSTEM_INSTRUCTIONS = """
You are a strict grader.
You are given a question and a model's response.

Return a answer of either:
- 1 if both the model's final answer AND reasoning chain are correct.
- 0 otherwise.

Your final answer should be the digit 0 or 1.
```

```
"""


PROMPT_TEMPLATE = """Grade the submission.

[Question]
{question}

[Model Response]
{reasoning}

[Reference Answer]
{reference_answer}
"""
```

We use 512 tokens for all judge models, and default hyperparameters.

D Reasoning Trace Examples

Below, we include reasoning traces from the model $p_\theta(\mathbf{a} \mid \mathbf{r}, \mathbf{c})p_\theta(\mathbf{r} \mid \mathbf{c})$ and the posterior traces $p_\theta(\mathbf{r} \mid \mathbf{a}, \mathbf{c})$. We include samples where LLaDA originally fails to correctly answer the question. In two cases, the posterior succesfully provides an alternative correct reasoning trace, and in two it fails to do so.

D.1 Correct Posterior Traces

Example 1

Question: Ms. Warren ran at 6 mph for 20 minutes. After the run, she walked at 2 mph for 30 minutes. How many miles did she run and walk in total?

Incorrect LLaDA output:

```
Ms. Warren ran at 6 mph for 20 minutes, which is $\frac{20}{60} =
    \frac{1}{3}$ hours. So, she ran $6 \times \frac{1}{3} = 2$
    miles.

Ms. Warren walked at 2 mph for 30 minutes, which is $\frac{30}{60}
    = \frac{1}{2}$ hours. So, she walked $2 \times \frac{1}{2} = 2
    miles as well.

The answer is \boxed{4} miles.
```

Correct LLaDA posterior:

```
Ms. Warren ran at 6 mph for 20 minutes, which is 20/60 = 1/3 of an
    hour.
So, she ran 6 mph * 1/3 hour = 2 miles.
After the run, she walked at 2 mph for 30 minutes, which is 30/60 =
    1/2 of an hour.
So, she walked 2 mph * 1/2 hour = 1 mile.
In total, Ms. Warren ran and walked 2 miles + 1 mile = 3 miles.
The answer is \boxed{3}
```

Example 2

Question: Each week Jaime saves $50. Every two weeks she spends $46 of her savings on a nice lunch with her mum. How long will it take her to save $135?

Incorrect LLaDA output:

```
Jaime saves $50 each week, so in two weeks she saves $50 x 2 = $100.
Every two weeks she spends $46, so in two weeks she saves $100 -
    $46 = $54.
To save $135, she needs to save $135 - $54 = $81 more.
Since she saves $54 every two weeks, it will take her $81 / $54 = 3
    more two weeks to save $135.
The answer is \boxed{3}+2 = 6 weeks.
```

Correct LLaDA posterior:

```
Jaime saves $50 each week, so in two weeks she saves $50 x 2 = $100.
However, she spends $46 every two weeks, so her net savings every
    two weeks is $100 - $46 = $54.
To find out how long it will take her to save $135, we divide $135
    by her net savings every two weeks: $135 / $54 = 2.5.
Therefore, it will take her 2.5 two weeks to save $135.
The answer is \boxed{5}
```

D.2 Incorrect Posterior Traces

Example 1

Question: Tommy is making 12 loaves of bread. He needs 4 pounds of flour per loaf. A 10-pound bag of flour costs $10 and a 12-pound bag costs $13. When he is done making his bread, he has no use for flour and so he will throw away whatever is left. How much does he spend on flour if he buys the cheapest flour to get enough?

Incorrect LLaDA output:

```
Tommy needs 12 loaves of bread, and each loaf requires 4 pounds of
    flour.
Therefore, he needs a total of 12 * 4 = 48 pounds of flour.

The cheapest option is to buy 12-pound bags of flour.
Since Tommy needs 48 pounds, he will need 48/12 = 4 bags.
Each bag costs $13, so the total cost for 4 bags is 4 * 13 = $52.

The answer is \boxed{52}.
```

Incorrect LLaDA posterior:

```
Tommy needs 12 loaves of bread, so he needs a total of 12 * 4 = 48
    pounds of flour.
If he buys 10-pound bags, he will need 48/10 = 4 bags,
which will cost 4 * 10 = 40.
If he buys 12-pound bags, he will need 48/12 = 4 bags, which will
    cost 4 * 13 = $52.
Therefore, Tommy should buy 10-pound bags to save money.
The answer is \boxed{50}.
```

Example 2

Question: Thomas started saving for a car almost 2 years ago. For the first year, his weekly allowance was $50. In the second year, he got a job that pays $9 an hour at a coffee shop and worked 30 hours a week, so his parents discontinued his allowance. If the car he wants to buy is $15,000 and he spends $35 a week on himself, how much more money does Thomas need to buy the car by the end of the 2 years?

Incorrect LLaDA output:

```
In the first year, Thomas earned $50 x 52 = $2600.
In the second year, Thomas earned $9 x 30 x 52 = 7980.
Thomas saved $2600 + $7980 = 10580.
Thomas spent $35 x 52 = 1820.
Thomas still needs $15000 - 10580 - 1820 = 2600 more.
The answer is \boxed{2600}.
```

Incorrect LLaDA posterior:

```
In the first year, Thomas earned $50 x 52 = $2600.
In the second year, Thomas earned $9 x 30 x 52 = 8760.
Thomas spent $35 x 52 x 52 = 8360.
In total, Thomas saved $2600 + $8760 - $8360 = 3000.
To buy the car, Thomas still needs $15000 - 3000 = $2000.
The answer is \boxed{2000}
```

E PROOFS

MED KL upper-bound Here we prove that for any set $A \subset \{1, \dots\}$/un-masked, the following upper bound holds:

$$\mathrm{KL}\left(p_\theta(\mathbf{x}^A \mid \mathbf{x}_{\text{un-masked}}, \mathbf{c}) \middle| \prod_{i \in A} p_\theta(x^i \mid \mathbf{x}_{\text{un-masked}}, \mathbf{c})\right) \leq \sum_{i \in A} H(x^i \mid \mathbf{x}_{\text{un-masked}}, \mathbf{c}) \tag{8}$$

Note that:

$$\begin{aligned}
&\mathrm{KL}\left(p_\theta(\mathbf{x}^A \mid \mathbf{x}_{\text{un-masked}}, \mathbf{c}) \middle| \prod_{i \in A} p_\theta(x^i \mid \mathbf{x}_{\text{un-masked}}, \mathbf{c})\right) \\
&\quad = \mathbf{E}_{p_\theta(\mathbf{x}^A \mid \mathbf{x}_{\text{un-masked}}, \mathbf{c})}\left[\log p_\theta(\mathbf{x}^A \mid \mathbf{x}_{\text{un-masked}}, \mathbf{c}) - \log \prod_{i \in A} p_\theta(x^i \mid \mathbf{x}_{\text{un-masked}}, \mathbf{c})\right] &(9)\\
&\quad = \mathbf{E}_{p_\theta(\mathbf{x}^A \mid \mathbf{x}_{\text{un-masked}}, \mathbf{c})}\left[\log p_\theta(\mathbf{x}^A \mid \mathbf{x}_{\text{un-masked}}, \mathbf{c}) - \sum_{i \in A} \log p_\theta(x^i \mid \mathbf{x}_{\text{un-masked}}, \mathbf{c})\right] &(10)\\
&\quad = -H(\mathbf{x}^A \mid \mathbf{x}_{\text{un-masked}}, \mathbf{c}) + \sum_{i \in A} H(\mathbf{x}^i \mid \mathbf{x}_{\text{un-masked}}, \mathbf{c}) &(11)
\end{aligned}$$

Now, since the entropy for discrete random variables is positive, $H(\mathbf{x}^A \mid \mathbf{x}_{\text{un-masked}}, \mathbf{c}) \geq 0$, which implies:

$$-H(\mathbf{x}^A \mid \mathbf{x}_{\text{un-masked}}, \mathbf{c}) + \sum_{i \in A} H(\mathbf{x}^i \mid \mathbf{x}_{\text{un-masked}}, \mathbf{c}) \leq \sum_{i \in A} H(\mathbf{x}^i \mid \mathbf{x}_{\text{un-masked}}, \mathbf{c}) \tag{12}$$

Hence, we have that for any set A, we have that:

$$\mathrm{KL}\left(p_\theta(\mathbf{x}^A \mid \mathbf{x}_{\text{un-masked}}, \mathbf{c}) \middle| \prod_{i \in A} p_\theta(x^i \mid \mathbf{x}_{\text{un-masked}}, \mathbf{c})\right) \leq \sum_{i \in A} H(x^i \mid \mathbf{x}_{\text{un-masked}}, \mathbf{c}) \tag{13}$$

Entropy upper-bound Next, we prove that:

$$H(\mathbf{a} \mid \mathbf{r}_{\text{UN-MASKED}}, \mathbf{c}) \leq H_{\text{UB}}(\mathbf{a} \mid \mathbf{r}_{\text{UN-MASKED}}, \mathbf{c}) \tag{14}$$

where $H_{\text{UB}}(\mathbf{a} \mid \mathbf{r}_{\text{UN-MASKED}}, \mathbf{c}) = \sum_i H(a^i \mid \mathbf{r}_{\text{UN-MASKED}}, \mathbf{c})$.

Next, we note that $\mathrm{KL}(p_\theta(\mathbf{a} \mid \mathbf{r}_{\text{UN-MASKED}}, \mathbf{c}) \mid \prod p_\theta(a^i \mid \mathbf{r}_{\text{UN-MASKED}}, \mathbf{c}) \geq 0$, which implies that, similar to eq. (11), we have

$$-H(\mathbf{a} \mid \mathbf{r}_{\text{UN-MASKED}}, \mathbf{c}) + H_{\text{UB}}(\mathbf{a} \mid \mathbf{r}_{\text{UN-MASKED}}, \mathbf{c}) = \mathrm{KL}(p_\theta(\mathbf{a} \mid \mathbf{r}_{\text{UN-MASKED}}, \mathbf{c}) \mid \prod p_\theta(a^i \mid \mathbf{r}_{\text{UN-MASKED}}, \mathbf{c})$$

$$-H(\mathbf{a} \mid \mathbf{r}_{\text{UN-MASKED}}, \mathbf{c}) + H_{\text{UB}}(\mathbf{a} \mid \mathbf{r}_{\text{UN-MASKED}}, \mathbf{c}) \geq 0 \tag{15}$$

$$H_{\text{UB}}(\mathbf{a} \mid \mathbf{r}_{\text{UN-MASKED}}, \mathbf{c}) \geq H(\mathbf{a} \mid \mathbf{r}_{\text{UN-MASKED}}, \mathbf{c}) \tag{16}$$