Rethinking Reasoning with Masked Diffusion Language Models: Early Exits, Post-hoc Reasoning, and Beyond

Zachary Horvitz[*,1], **Raghav Singhal**[*,2], **Hao Zou**[1], **Carles Domingo-Enrich**[3]
Zhou Yu[1], **Rajesh Ranganath**[2,4], **Kathleen McKeown**[1]
[1]Columbia University [3]Microsoft Research
[2]Department of Computer Science, New York University
[4]Center for Data Science, New York University

Abstract

The reasoning paradigm, where language models reason before answering, has enabled breakthroughs on tasks such as mathematical problem-solving. While current tooling for reasoning is built around next-token prediction trained models, recent works introduce an alternative choice: masked diffusion language models (MDLMs). MDLMs are trained to in-fill positions in randomly masked sequences. We introduce *reasoning-as-infilling*, a prompting technique that pre-fills tokens to explicitly delimit reasoning and answer regions, unlocking a unified set of capabilities for MDLM reasoning. Because answer positions are explicitly designated, the model's conditional distributions over answer tokens are directly accessible during generation. This enables *early exits* when the model is certain of its answer. The same framework supports *post-hoc reasoning*: given question-answer pairs, MDLMs can sample high-quality reasoning traces from their posterior, a distribution that is intractable for autoregressive models. On GSM8k, fine-tuning LLaDA-8B-Base on these posterior traces improves accuracy by +14.9%, matching gains from human-written traces (+13.4%). Finally, given a reference answer, the answer region distributions enable *scoring partial reasoning traces* at *intermediate steps*, providing intermediate rewards that are more strongly correlated with correctness than scores from a specialized process reward model. At intermediate steps, answer-likelihood scores from autoregressive models are significantly less predictive of correctness than those from MDLMs. Our results demonstrate that the MDLM training objective provides promising benefits for reasoning.

1 Introduction

The current dominant approach for language modeling is based on next-token prediction (NTP) training. Next-token prediction (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 distributions $p_\theta(x^i|\mathbf{x}_{\text{UN-MASKED}})$ at every masked position i.

While modeling all masked positions requires additional effort, MDLMs have several potential benefits, such as parallel token decoding (Sahoo et al., 2024; 2025; Ben-Hamu et al., 2025; Wu et al., 2025a), and flexible decoding orders (Kim et al., 2025b) 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 sam-

[*]Denotes equal authorship. Correspondence to zfh2000@columbia.edu, rsinghal@nyu.edu.

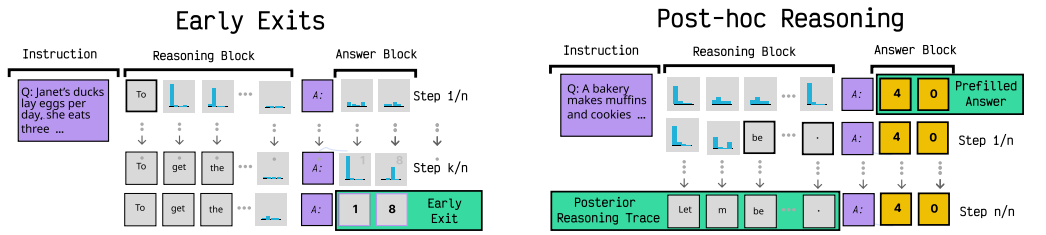

Figure 1: MDLMs learn the conditional distributions at each masked token position. We reframe reasoning as infilling a prompted reasoning template, which enables directly modeling answer token probabilities *during* reasoning. This provides several benefits, like enabling early exits (*left panel*) or *post-hoc* reasoning given a pre-filled answer (*right panel*).

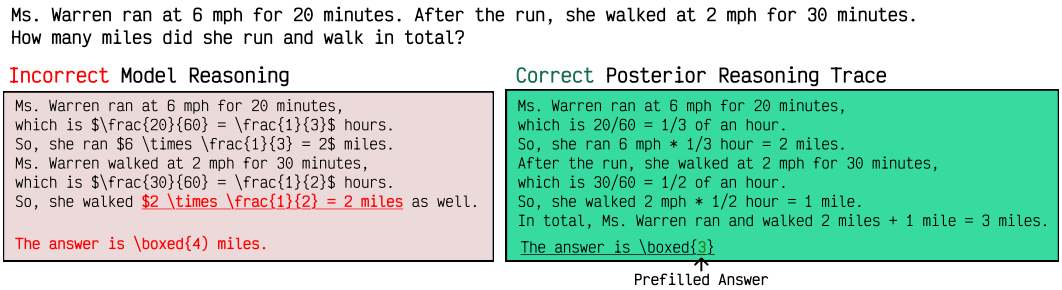

Figure 2: Posterior reasoning from correct answers is a source of high-quality reasoning traces. Pre-filling a correct answer into a reasoning template elicits correct reasoning for a large portion of problems that the model initially fails to solve. For example, here greedily sampling from LLaDA-8B-Instruct (Nie et al., 2025) yields an incorrect reasoning step that is corrected in the posterior reasoning trace.

ples from masked positions supports controllable generation (Schiff et al., 2024; Singhal et al., 2025).

In this work, we demonstrate that the conditional distributions of all masked positions, together with the ability of MDLMs to in-fill, unlock new sampling and post-training capabilities for reasoning that are not readily available for NTP models.

We propose **reasoning-as-infilling**, where we add user-specified contexts in multiple positions, not just the beginning of the sequence, unlike NTP models. Specifically, we prefill an explicit reasoning template that delimits specific reasoning and answer positions (see fig. 1). This single change yields multiple inference-time and post-training capabilities.

First, because the answer positions are explicitly designated, the MDLM's conditional distributions over answer tokens are directly accessible *during* generation. This enables **early exits**: once the model's answer-region entropy falls below a threshold, the remaining reasoning steps can be skipped. On GSM8k, early exits alone yield a 1.3× acceleration with less than a 1 percentage point reduction in accuracy, and can be combined with parallel decoding algorithms for additional acceleration. Reasoning-as-infilling also integrates naturally with *variable-sequence length* diffusion approaches (Wu et al., 2025b; Havasi et al., 2025; Kim et al., 2025a), enabling flexible reasoning budgets at inference time.

Second, the same framework directly supports post-training. Given a correct answer, prefilling the answer block enables sampling from the MDLM's posterior $p_\theta(\mathbf{r} \mid \mathbf{c}, \mathbf{a})$, a distribution that is intractable for NTP models but trivial for MDLMs. This yields **high-quality post-hoc reasoning traces** that significantly improve model performance when used for fine-tuning, matching the gains from expensive human-annotated traces.

Third, given a reference answer, the same answer-conditioned distributions provide a built-in mechanism for **scoring partial reasoning traces** at intermediate steps, without training an auxiliary process reward model. The marginal probabilities of the answer block correlate more strongly with final output correctness than scores from a specialized 7 billion parameter process reward model (Zhang et al., 2025a), suggesting applications for steering (Schiff et al., 2024; Singhal et al., 2025) and filtering during post-training.

Contributions. We introduce reasoning-as-infilling, a method that leverages MDLM's infilling capabilities to explicitly distinguish reasoning and answer tokens. This framework naturally provides three key benefits:

- *Early exits*, where we skip the remaining reasoning steps once the model is sufficiently certain of the answer. On GSM8k with one-token decoding, this leads to a 1.3× *acceleration with less than a* 1 *percentage point reduction in accuracy*. Early exits can be combined with parallel decoding algorithms for additional acceleration.
- *Post-hoc reasoning*, where given question-answer pairs, we generate reasoning traces conditioned on the answer. On the GSM8k dataset, supervised fine-tuning on these reasoning traces can improve the model (+14.9%) as much as supervised fine-tuning on the expensive *human-annotated* GSM8k reasoning traces (+13.4%). Posterior traces are also judged more correct than traces from STaR-style prefix-hint prompting for the same models.
- *Scoring reasoning traces*, where given a question-answer pair, we can score the reasoning process for correctness at intermediate steps using the distributions of the answer block. These scores correlate with whether the reasoning process will produce correct answers. In contrast, we find that analogous answer likelihoods computed by the evaluated autoregressive models are only weakly informative until late in generation.

2 Related Work

Early Exits. Concurrently, Li et al. (2025) demonstrate that MDLMs assign high probability to correct answers before decoding a full reasoning trace. They also propose a method for decoding answers early, based on the model's top logits over a pre-specified region of the sequence. In one configuration of their method, similar to *reasoning-as-infilling*, they prefill a suffix to delimit an answer region. Our work differs in several respects: we use an upper-bound on answer region entropy as the criterion for certainty, rather than marginal probability scores, and we ensure the answer region is sampled from the model's joint distribution rather than decoded in one step. Beyond early exits, we demonstrate that the reasoning-as-infilling framework extends naturally to parallel decoding and variable-length decoding algorithms (Kim et al., 2025a; Wu et al., 2025b; Havasi et al., 2025), further accelerating inference and enabling flexible reasoning budgets. Critically, we go beyond early exiting to show that the same infilling framework supports post-hoc reasoning trace generation and intermediate scoring.

Post-hoc Reasoning and Scoring. Zelikman et al. (2022) proposes fine-tuning language models on reasoning traces that are generated conditioned on 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 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. For MDLMs, Zhao et al. (2025b) propose in-filling guided policy optimization, where they randomly mask and infill human-written reasoning traces during training on problems where the model does not generate a correct reasoning trace and answer. In contrast, we sample from MDLMs given only the gold answer and no human-written reasoning traces.

Using log-probabilities for scoring or training has been explored for autoregressive models (Liu et al., 2025; Zhou et al., 2025). In contrast to auto-regressive models, MDLMs provide access to the conditional distributions of the answer tokens at *every intermediate step* of the reasoning process, as a byproduct of each forward pass. This yields dense, step-level scores, without requiring completed rollouts or training a separate process reward model (Zhang et al., 2024; Silver et al., 2016).

3 Masked Diffusion Language Models

MDLMs (Sohl-Dickstein et al., 2015; Devlin et al., 2019; Austin et al., 2021; Sahoo et al., 2024; Shi et al., 2024) are a class of generative models for modeling discrete data $\mathbf{x} \sim q_{\text{data}}$, where $\mathbf{x} = (x^1, x^2, \ldots, x^L)$ and each position x^i takes values in a finite vocabulary $\mathcal{V}$. For training the model $p_\theta(\mathbf{x})$, 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 by maximizing a lower bound on the log-likelihood of $\mathbf{x}$:

$$\mathcal{L}(\mathbf{x}, \theta) := \mathbf{E}_{\text{MASK} \sim U} \sum_{j \in \text{MASK}} \log p_\theta(x^j \mid \mathbf{x}_{\text{UN-MASKED}})$$

where MASKED-SET is a set of randomly masked positions in $\{1, 2, \ldots, L\}$. 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 $\mathbf{o} = [o_1, \ldots, o_L]$ with $o_j \in \{1, 2, \ldots, L\}$, the model likelihood can be defined as:

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

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.

Sampling from an MDLM model requires defining a decoding order. Popular sampling approaches for MDLMs select positions to unmask based on confidence (e.g. token probability (Chang et al., 2022; Wu et al., 2025a) or entropy (Kim et al., 2025b; Ye et al., 2025b; Ben-Hamu et al., 2025)). These sampling algorithms are often combined with block sampling methods (Sahoo et al., 2024; Arriola et al., 2025), which define a left-to-right sequence of fixed length blocks and decode within each block in an arbitrary order (Sahoo et al., 2024; Nie et al., 2025; Arriola et al., 2025).

Recently, several works (Wu et al., 2025b; Havasi et al., 2025; Kim et al., 2025a) have proposed variable-sequence MDLMs. Instead of unmasking a fixed-length sequence, these methods dynamically resize the output at inference-time using expand and delete operations.

What do MDLMs offer for reasoning? One benefit of MDLMs for reasoning tasks is their ability to decode tokens in parallel (Ben-Hamu et al., 2025; Wu et al., 2025a), accelerating inference and data generation. Another possible benefit is their ability to decode sequences in arbitrary orders. For instance, Kim et al. (2025b) show that any-order decoding outperforms next-token prediction on puzzles such as Sudoku. However, our experiments show that any-order decoding offers *limited benefits* for reasoning tasks. Naive or block any-order decoding algorithms either sample a large portion of tokens in a left-to-right order (nearly 50% for GSM8K) or underperform left-to-right sampling (see fig. 5). Qualitatively, any-order decoding often elicits a surprising behavior: MDLMs decode an incorrect answer first, followed by a post-hoc rationale (see appendix B for additional analysis).

This "pathological" behavior is revealing: it demonstrates that MDLMs naturally support many different factorizations of the joint distribution $p_\theta(\mathbf{a}, \mathbf{r} \mid \mathbf{c})$ over reasoning traces $\mathbf{r}$, input context $\mathbf{c}$, and answers $\mathbf{a}$. At inference-time, sampling a possibly incorrect answer $\hat{\mathbf{a}}$ before a reasoning trace is undesirable. However, when the gold answer is known, as in post-training, this same factorization becomes a valuable source of reasoning data. Moreover, the tendency of MDLMs to converge on answers early suggests that models may be certain of an answer well before completing their full reasoning trace. These observations motivate *reasoning-as-infilling*.

4 Rethinking Reasoning with MDLMs

MDLMs are prompted similarly to NTP models, and the distributions of the masked positions are used only for sampling high-confidence tokens. The remaining distributions are discarded. We show that the ability of MDLMs to in-fill and to access the distribution of *all* masked positions unlocks new inference-time and post-training capabilities.

4.1 Reasoning-as-Infilling with MDLMs

Generally, NTP models are controlled at inference-time with a prompt prefix $\mathbf{c}$ 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:

$$\left[\underbrace{[\mathrm{M}]_1\ [\mathrm{M}]_2\ \ldots\ [\mathrm{M}]_k}_{\text{reasoning block}}\ \text{<Delimiter>}\ \underbrace{[\mathrm{M}]_{k+1}\ \ldots\ [\mathrm{M}]_L}_{\text{answer block}}\right]$$

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. By distinguishing between reasoning and answer positions, reasoning-as-infilling offers several advantages for sampling and post-training.

Early exits. By designating explicit answer block positions, reasoning-as-infilling enables measuring answer uncertainty *while generating the reasoning trace*. A measure of uncertainty is the entropy of the answer block given the unmasked reasoning positions. This joint entropy requires additional estimation as MDLMs only provide access to the marginals $p_\theta(a_j \mid \mathbf{r}_{\text{UNMASKED}}, \mathbf{c})$. However, we show that the marginal distributions can be used to upper-bound the joint entropy,

$$H_{\text{UB}} := \sum_{j \in \text{ANSWER-BLOCK}} H(a_j \mid \mathbf{r}_{\text{UNMASKED}}, \mathbf{c}) \geq H(\mathbf{a} \mid \mathbf{r}_{\text{UNMASKED}}, \mathbf{c}) \tag{2}$$

See appendix I for a proof. Using this quantity, we propose **early exiting based on the answer uncertainty upper-bound** H_{UB}. That is, given a partial reasoning trace, $\mathbf{r}_{\text{UNMASKED}}$, we skip filling in the remaining reasoning tokens if the answer-entropy upper bound falls below a user-specified threshold γ, $H_{\text{UB}} < \gamma$.

Post-training MDLMs with reasoning-as-infilling. Typically, post-training a model to reason uses expensive human demonstrations (Ouyang et al., 2022). Alternatively, Zelikman et al. (2022); Phan et al. (2023); Ruan et al. (2025) have demonstrated that post-training on model generated reasoning traces provides an alternative for improving performance (Zelikman et al., 2022; Phan et al., 2023; Ruan et al., 2025).

These methods work off the principle that sampling reasoning traces from the posterior $p_\theta(\mathbf{r} \mid \mathbf{c}, \mathbf{a})$ and then training on these samples can increase the likelihood of generating correct answers. However, sampling from the posterior $p_\theta(\mathbf{r} \mid \mathbf{a}, \mathbf{c})$ is intractable for NTP models: the autoregressive factorization generates tokens left-to-right, making it impossible to condition on answer tokens that appear later in the sequence without either significant prompt engineering or training another model to yield reasoning traces given answer hints (Phan et al., 2023; Zelikman et al., 2022).

Because MDLMs model the conditional distributions of *all* masked positions given any set of unmasked tokens, conditioning on the answer is as simple as pre-filling the answer block positions. With reasoning-as-infilling, one can *simply* pre-fill the answer positions to sample directly from the posterior distribution, without prompt engineering or auxiliary models. These posterior traces can be used for post-training in several ways, including with STaR (Zelikman et al., 2022), maximum marginal log-likelihood training (Phan et al., 2023; Murphy, 2023), or maximizing the likelihood on the answer and the posterior reasoning traces: $\max_\theta \sum_{i=1}^{N} \log p_\theta(\mathbf{a}_i, \mathbf{r}_i \mid \mathbf{c}_i)$ where the posterior reasoning traces are generated by the model, $\mathbf{r}_i \sim p_\theta(\mathbf{r}_i \mid \mathbf{c}_i, \mathbf{a}_i)$.

Scoring partial reasoning traces when post-training. Existing fine-tuning algorithms, such as GRPO (Shao et al., 2024) and RLOO (Ahmadian et al., 2024), do not make use of posterior samples but score the generations upon completion. These algorithms can benefit from intermediate rewards (Silver et al., 2016). Recent work has shown that guiding the generation process with intermediate rewards produces samples that improve model

		GSM8K			Math500		
Model	**Sampler**	**Exit Param**	**Speed-up ↑**	**Acc.**	**Exit Param**	**Speed-up ↑**	**Acc.**
LLaDA	ENTROPY, $k = 1$	NO TEMPLATE	1 ×	76.6	NO TEMPLATE	1 ×	33.8
LLaDA	ENTROPY, $k = 1$	NO EXIT	1 ×	79.4	NO EXIT	1 ×	33.4
LLaDA	ENTROPY, $k = 1$	$\gamma = 0.1$	1.3 ×	78.6	$\gamma = 0.3$	1.2 ×	31.9
LLaDA	MED, $\lambda = 0.2$	NO EXIT	2.7 ×	79.9	NO EXIT	1.8 ×	33.4
LLaDA	MED, $\lambda = 0.2$	$\gamma = 0.1$	3.3 ×	79.3	$\gamma = 0.3$	2 ×	32.0
LLaDA	EB-sampler, $\lambda = 0.1$	NO EXIT	2.8 ×	79.5	NO EXIT	2.2 ×	34.6
LLaDA	EB-sampler, $\lambda = 0.1$	$\gamma = 0.1$	3.5 ×	78.7	$\gamma = 0.3$	2.4 ×	33.6
LLaDA	BPCD, $\lambda = 0.9$	NO EXIT	3.3 ×	79.0	NO EXIT	2.7 ×	33.2
LLaDA	BPCD, $\lambda = 0.9$	$\gamma = 0.1$	**[4 ×]**	78.9	$\gamma = 0.3$	**[3 ×]**	31.8
Dream	ENTROPY, $k = 1$	NO TEMPLATE	1 ×	80.1	NO TEMPLATE	1 ×	33.4
Dream	ENTROPY, $k = 1$	NO EXIT	1 ×	79.8	NO EXIT	1 ×	35.6
Dream	ENTROPY, $k = 1$	$\gamma = 0.7$	1.1 ×	76.7	$\gamma = 0.7$	1 ×	33.2
Dream	MED, $\lambda = 0.2$	NO EXIT	1.9 ×	79.2	NO EXIT	1.7 ×	35.6
Dream	MED, $\lambda = 0.2$	$\gamma = 0.7$	2.1 ×	79.3	$\gamma = 0.7$	1.8 ×	35.4
Dream	EB-sampler, $\lambda = 0.1$	NO EXIT	2 ×	77.6	NO EXIT	1.9 ×	32.2
Dream	EB-sampler, $\lambda = 0.1$	$\gamma = 0.7$	2.2 ×	75.3	$\gamma = 0.7$	**[3.4 ×]**	30.2
Dream	BPCD, $\lambda = 0.9$	NO EXIT	2.2 ×	78.0	NO EXIT	2.2 ×	32.0
Dream	BPCD, $\lambda = 0.9$	$\gamma = 0.7$	**[2.5 ×]**	74.6	$\gamma = 0.7$	2.5 ×	30.8

Table 1: Early exits accelerate MDLM inference. We evaluate LLaDA-8B-Instruct Nie et al. (2025) and Dream-7B-Instruct (Ye et al., 2025a) on GSM8k. We report acceleration relative to full-length generation with a sequence length of 256. Varying the early-exit threshold γ trades faster inference for task accuracy. Early exits can be combined with parallel decoding algorithms for additional acceleration.

fine-tuning (Zhang et al., 2024). These intermediate rewards are generally provided by an *external* pre-trained process reward model (PRM) (Lightman et al., 2023; Zhang et al., 2024; 2025b).

Given the answer, reasoning-as-infilling enables MDLMs to score arbitrary reasoning traces at intermediate steps. This is possible because MDLMs jointly model all positions, including the answer block distributions, as part of each forward pass during generation. Given a partial reasoning trace $\mathbf{r}_{\text{UNMASKED}}$ and an answer $\mathbf{a}^*$, we score $\mathbf{r}_{\text{UNMASKED}}$ with:

$$\phi_{\text{IPRM}}(\mathbf{r}_{\text{UNMASKED}} \mid \mathbf{c}, \mathbf{a}^*) := \sum_{j \in \text{ANSWER-BLOCK}} \log p_\theta(a_j = a_j^* \mid \mathbf{c}, \mathbf{r}_{\text{UNMASKED}}). \tag{3}$$

The intuition behind the equation is that when the likelihood of individual answer tokens is higher for the reasoning trace $\mathbf{r}_{\text{UNMASKED}}$, then $\mathbf{r}_{\text{UNMASKED}}$ is often more likely to produce the answer. Unlike the entropy criterion used for early exits, which requires no answer, ϕ_{IPRM} assumes a reference or verifier-provided answer $\mathbf{a}^*$, and is primarily applicable to post-training settings where labels are available.

5 Experiments

In this section, we examine the inference-time and post-training benefits of reasoning-as-infilling, such as **early-exits** based on answer certainty, the ability to **bootstrap reasoning traces** from correct answers, and their intrinsic ability to **score partial reasoning traces** given (question, answer) pairs.

5.1 Early exits

We investigate the inference-time benefits of reasoning-as-infilling on two mathematical reasoning benchmarks, GSM8k (Cobbe et al., 2021) and MATH500 (Lightman et al., 2023), with the Dream-7B-Instruct and LLaDA-8B-Instruct models. For both tasks, we consider a generation length $L = 256$ 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.

Model	Setting	GSM8K			
		Exit Param	Speed-up ↑	Seconds↓	Acc.
DreamOn	No Template (Fixed Length)	No Exit	1 ×	10.8	1.4%
	No Template	No Exit	1 ×	8.7	4.9%
DreamOn w/ Template	With Template (Fixed Length)	No Exit	1 ×	11.0	60.6%
	With Template	No Exit	1 ×	10.3	66.4%
	With Template (Fixed Length)	$\gamma = 0.5$	1.9 ×	5.9	64.7%
	With Template	$\gamma = 0.5$	1.7 ×	6.5	66.0%

Table 2: Reasoning-as-infilling can be used alongside variable-length diffusion models. We evaluate DreamOn (Wu et al., 2025b) on GSM8k. We consider both fixed-length decoding, and utilizing deletion operation to dynamically resize the reasoning budget. We report accuracy, along with speed-up relative to full-length generation (256 steps). We also report wall-clock seconds on an Nvidia A100 GPU. Notably, while DreamOn was fine-tuned for variable-length generation on a coding dataset (Huang et al., 2024), reasoning-as-infilling elicits correct mathematical reasoning.

Early Exiting Alongside Multi-token Decoding. For sampling, we examine early exits with one-token decoding along with recent multi-token decoding methods. We use entropy-bound (EB) sampler (Ben-Hamu et al., 2025), block parallel confidence decoding (BPCD) (Wu et al., 2025a), as well as a simple entropy-based multi-token method that we propose called MED, which we describe in appendix A. For a comparison of the three samplers, see fig. 4 in appendix A.

In table 1, we observe:

- **Faster inference with minimal accuracy degradation:** For LLaDA, early exiting yields a 1.3× acceleration on one-token entropy decoding with only a 0.8 percentage point drop in performance versus the baseline reasoning template.
- **Benefits compound with multi-token decoding**. Early exits combined with multi-token decoding provide further acceleration. For LLaDA on GSM8k, BPCD alone yields a 3.3× acceleration. Adding early exits yields a 4.0× total acceleration, a further 1.2× acceleration over BPCD alone, while accuracy changes from 79.0 to 78.9. In appendix F, we provide examples of reasoning traces with varying exit thresholds.

Notably, the benefits of early exits are more pronounced for LLaDA than Dream, which requires higher exit thresholds for speed-ups. This may be due to Dream's adaptation from an NTP model (Gong et al., 2024; Ye et al., 2025a). See appendix B for a discussion of the sampling behavior of Dream and LLaDA.

We also compare entropy-based exits with Prophet (Li et al., 2025) on GSM8k and CommonsenseQA. On GSM8k, the methods are matched at comparable NFE budgets (74.8% vs. 74.3% at 1.55×). On CommonsenseQA, entropy exits are both more accurate and faster at every evaluated threshold (for example, 76.2% at 1.86× vs. 72.2% at 1.02×). Full sweeps are in appendix G.

Early Exiting with Variable-Length Decoding. One potential drawback of reasoning-as-infilling with MDLMs is that it pre-defines a fixed reasoning trace length. We next examine how reasoning-as-infilling and early exits can be utilized alongside variable-length diffusion models (Wu et al., 2025b; Havasi et al., 2025; Kim et al., 2025a). We evaluate DreamOn-7B, which was fine-tuned for variable-length decoding on a dataset of coding problems (Huang et al., 2024), on GSM8k with an initial sequence length of 256 and block size 32 decoding. We consider both fixed-length decoding and sampling with dynamic reasoning-trace lengths via deletions.

In table 2, reasoning-as-infilling significantly improves the model's mathematical reasoning, eliciting correct reasoning despite the model's fine-tuning on coding tasks. Without reasoning-as-infilling DreamOn-7B performs poorly on GSM8k, with an accuracy of 1.4%. Enabling deletions further improves accuracy and provides a 6% reduction in sampling

Model	Posterior Reasoning Scores	
	Qwen-PRM	GPT-4o
LLaDA-8B-Base	0.31	0.36
LLaDA-8B-Instruct	0.38	0.43

Table 3: The MDLM reasoning posterior yields high-quality traces for problems that the original instruct-tuned model fails to solve. We perform posterior inference on the 1419 training samples that LLaDA-8B-Instruct 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 judges rate $\sim$ 40% of the instruct-tuned reasoning chains as correct. Notably, even the posterior reasoning chains from the base model are rated as $>$ 31% correct.

Model	Post-training Data	GSM8K Test Acc.	Early-exit Speed Up
LLaDA 8B-Base (No template)	-	13.9%	
LLaDA 8B-Base (With template)	-	51.2 %	
Early exit $\gamma = 1.0$		50.0 %	1.01×
Finetuned* LLaDA 8B-Base	GSM8k $(\mathbf{c}, \mathbf{r}_{\text{gold}}, \mathbf{a}_{\text{gold}})$ $(n = 7473)$	64.6 (+13.4) %	
Early exit $\gamma = 0.3$		58.9 %	1.4×
Early exit $\gamma = 0.4$		54.1 %	1.7×
Finetuned* LLaDA 8B-Base	GSM8k Posterior $(\mathbf{c}, \mathbf{r}, \mathbf{a}_{\text{gold}})$ $(n = 7473)$	66.1 (+14.9) %	
Early exit $\gamma = 0.3$		66.2 %	1.2×
Early exit $\gamma = 0.4$		64.7 %	1.5×
LLaDA-8B-Instruct	Misc. Instruction Data (n = 4.5 million)	75.96 %	–

Table 4: Fine-tuning the base model with posterior-generated data improves performance. Fine-tuning the LLaDA-8B-Base model on GSM8k human-written training data and posterior reasoning traces boosts accuracy to 64.6% and 66.1%, respectively. Finetuned* indicates LoRA (Hu et al., 2022) fine-tuning. Both fine-tuned models exhibit early exit capabilities. Notably, reasoning-as-infilling alone significantly improves base model performance from 13.9 $\rightarrow$ 51.2%.

time.[*] Incorporating early exits provides an additional 37% reduction in sampling time while maintaining accuracy.

5.2 Post-hoc reasoning

A key challenge for training better reasoning models is collecting high-quality reasoning traces for fine-tuning (Zelikman et al., 2022). We investigate whether the MDLM posterior distribution, $p_\theta(\mathbf{r} \mid \mathbf{c}, \mathbf{a})$, can provide these traces, even when an MDLM incorrectly solves the original task.

In this experiment, we generate samples from the LLaDA-8B-Instruct model with multi-token entropy decoding (MED) and any-order decoding. On the GSM8k training dataset (Lightman et al., 2023), the model answers 1419 out of 7473 problems incorrectly. We use these 1419 question-answer pairs to generate reasoning traces from both the instruct and the base model (without instruction tuning).

We evaluate these reasoning traces for correctness using GPT4o (Hurst et al., 2024) and the Qwen2.5-Math-7B PRM (Zhang et al., 2025a) (see appendix C for the system instructions). As shown in table 3, both judges rate $\sim$ 40% of the instruct-tuned model's posterior reasoning traces as correct, even on problems the model originally fails. We observe that the traces judged correct by GPT4o contain accurate reasoning steps that correct the original model's errors (see appendix D for examples). Notably, even the base model produces correct reasoning chains for $>$ 30% of these problems when conditioned on the answer. Posterior conditioning also outperforms STaR-style prefix-hint prompting (Zelikman et al., 2022) with the same checkpoints for both instruct and base models (table 8).

[*]We report average sampling time on an Nvidia A100 40GB GPU. DreamOn deletions count towards NFEs, but reduce the length of the sequence, and memory requirements of each forward pass.

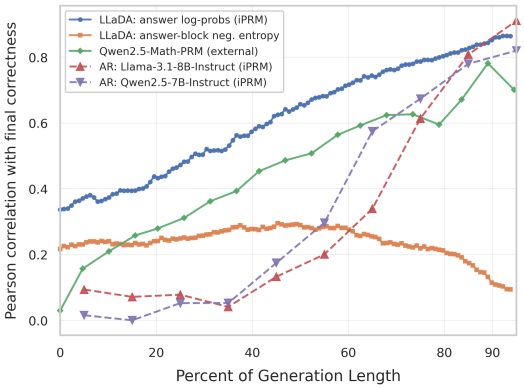

Figure 3: With answers, MDLMs enable scoring their own reasoning process. We score GSM8k reasoning traces, generated left-to-right with LLaDA, at intermediate steps using (a) gold answer log probabilities, (b) the answer block entropy bound, and (c) a 7B process reward model (PRM) (Zhang et al., 2025b). Gold answer probability at intermediate steps is more predictive of final correctness than PRM scores. We also compute an additional autoregressive baseline, where we compute gold answer likelihood scores at intermediate steps. The AR intermediate scores are significantly less correlated with final correctness (see appendix H for details). Even without gold labels, the MDLM answer block entropy is weakly correlated with correctness.

Posterior data can be used to improve base models. Here, we examine the effectiveness of post-training on the post-hoc reasoning dataset generated on the full GSM8k training set with the *base model*. We post-train the LLaDA-8B-Base model using LoRA (Hu et al., 2022) on a single Nvidia 40GB A100.

In table 4, we observe that fine-tuning the model on the posterior generated base model data significantly improves performance (+14.9%). As a benchmark, fine-tuning on the GSM8k *human annotated* reasoning traces produces comparable accuracy improvements on the test set. These results provide evidence that maximizing the log-likelihood $\log p_\theta(\mathbf{a}, \mathbf{r} \mid \mathbf{c})$ on the posterior reasoning traces improves accuracy on reasoning tasks. In turn, this indicates that when human reasoning traces are not available, MDLMs have the capacity to bootstrap their own instruction fine-tuning datasets. We include additional training details in appendix J.

Additionally, fine-tuning a model enhances early exiting capabilities. Using early exits can yield a 1.5× acceleration with less than a 1.4 percentage point drop in accuracy. In contrast, even at high γ thresholds, early exits have limited effects when performing reasoning-as-infilling with the original base model.

Scoring Intermediate Reasoning Traces. The reasoning-as-infilling framework also provides a mechanism for scoring reasoning traces at intermediate steps, which typically requires training a separate process reward model (Zhang et al., 2024). Because reasoning-as-infilling maintains explicit answer block positions, the model's own conditional distributions over answer tokens can serve as an intermediate score as defined in eq. (3).

Using LLaDA-8B-Instruct, we greedily sample solutions on the GSM8k test set (left-to-right, one token at a time) and compute the Pearson correlation between intermediate rewards and final output correctness. In fig. 3, we compare several strategies for estimating the final correctness of partial reasoning traces (see appendix E for details). The answer log probabilities $\log p_\theta(a_j = a_j^* \mid \mathbf{c}, \mathbf{r}_{\text{UN-MASKED}})$ correlate more strongly with final answer correctness at intermediate steps than scores from Qwen2.5-Math-PRM, a 7B process reward model (Zhang et al., 2025a).

Additionally, we compare whether next-token prediction trained auto-regressive models can similarly score intermediate reasoning traces. In theory, auto-regressive models can also estimate the likelihood of a reference answer given a partial reasoning trace in a single forward pass, by appending *The answer is \boxed{[ANSWER]* and computing answer likelihoods. In fig. 3, we include results for Llama-3.1-8B-Instruct (Dubey et al., 2024) and Qwen2.5-7B-Instruct (Team, 2024). Notably, intermediate scores for these models are signif-

icantly less predictive of correctness at early and intermediate steps than both the ϕ_{IPRM} and the dedicated process reward model. At early steps, intermediate auto-regressive likelihoods are less predictive of correctness than using label-free MDLM answer entropy.

These results suggest that MDLM pre-training, combined with reasoning-as-infilling, opens up several post-training directions: low-quality reasoning chains could be terminated early, the reasoning process could be steered towards correct solutions, reflection tokens could be automatically inserted at reasoning failures, and dense intermediate feedback could be incorporated into fine-tuning objectives.

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, and enables *rethinking* how these models are applied to reasoning tasks. Reasoning-as-infilling elicits useful properties of the MDLM training objective, allowing the user to structure the reasoning process, accelerating inference, and providing new post-training capabilities.

Limitations. Our proposed post-training methods, both generating a reasoning trace conditioned on the answer and scoring reasoning traces at intermediate reasoning steps, require access to question-answer pairs. Therefore, they do not apply to post-training tasks where we do not have access to an answer. However, access to question-answer pairs is a standard assumption in reinforcement learning with verifiable rewards, for instance GRPO (DeepSeek-AI et al., 2025). Additionally, we limit our evaluations to tasks with short answers. Future work could explore early exit and post-hoc reasoning capabilities of diffusion models on tasks with long or open-ended answers.

Acknowledgments

The authors would like to thank Subham Sahoo, Rico Angell, Chandhru Karthick, Mark Goldstein, Shenglong Wang, and Dinghuai Zhang for their valuable feedback.

This work was partly supported by the NIH/NHLBI Award R01HL148248, NSF Award 1922658 NRT-HDR: FUTURE Foundations, Translation, and Responsibility for Data Science, NSF CAREER Award 2145542, ONR N00014-23-1-2634, and Apple. Additional support was provided by a Fellowship from the Columbia Center of AI Technology. This work was also supported by IITP with a grant funded by the MSIT of the Republic of Korea in connection with the Global AI Frontier Lab International Collaborative Research. Zachary Horvitz is supported by a Google PhD Fellowship.

Large Language Model Use

We used LLM assistants for proofreading, copyediting, and limited wording revisions. Additionally, we used AI coding tools for assistance with scripts and plot generation. All resulting outputs were reviewed by the authors.

References

Arash Ahmadian, Chris Cremer, Matthias Gallé, Marzieh Fadaee, Julia Kreutzer, Olivier Pietquin, Ahmet Üstün, and Sara Hooker. Back to basics: Revisiting reinforce style optimization for learning from human feedback in llms. *arXiv preprint arXiv:2402.14740*, 2024.

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.

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, et al. Training verifiers to solve math word problems. *arXiv preprint arXiv:2110.14168*, 2021.

DeepSeek-AI, Daya Guo, Dejian Yang, Haowei Zhang, Junxiao Song, Ruoyu Zhang, Runxin Xu, Qihao Zhu, Shirong Ma, Peiyi Wang, Xiao Bi, Xiaokang Zhang, Xingkai Yu, Yu Wu, Z. F. Wu, Zhibin Gou, Zhihong Shao, Zhuoshu Li, Ziyi Gao, Aixin Liu, Bing Xue, Bingxuan Wang, Bochao Wu, Bei Feng, Chengda Lu, Chenggang Zhao, Chengqi Deng, Chenyu Zhang, Chong Ruan, Damai Dai, Deli Chen, Dongjie Ji, Erhang Li, Fangyun Lin, Fucong Dai, Fuli Luo, Guangbo Hao, Guanting Chen, Guowei Li, H. Zhang, Han Bao, Hanwei Xu, Haocheng Wang, Honghui Ding, Huajian Xin, Huazuo Gao, Hui Qu, Hui Li, Jianzhong Guo, Jiashi Li, Jiawei Wang, Jingchang Chen, Jingyang Yuan, Junjie Qiu, Junlong Li, J. L. Cai, Jiaqi Ni, Jian Liang, Jin Chen, Kai Dong, Kai Hu, Kaige Gao, Kang Guan, Kexin Huang, Kuai Yu, Lean Wang, Lecong Zhang, Liang Zhao, Litong Wang, Liyue Zhang, Lei Xu, Leyi Xia, Mingchuan Zhang, Minghua Zhang, Minghui Tang, Meng Li, Miaojun Wang, Mingming Li, Ning Tian, Panpan Huang, Peng Zhang, Qiancheng Wang, Qinyu Chen, Qiushi Du, Ruiqi Ge, Ruisong Zhang, Ruizhe Pan, Runji Wang, R. J. Chen, R. L. Jin, Ruyi Chen, Shanghao Lu, Shangyan Zhou, Shanhuang Chen, Shengfeng Ye, Shiyu Wang, Shuiping Yu, Shunfeng Zhou, Shuting Pan, S. S. Li, Shuang Zhou, Shaoqing Wu, Shengfeng Ye, Tao Yun, Tian Pei, Tianyu Sun, T. Wang, Wangding Zeng, Wanjia Zhao, Wen Liu, Wenfeng Liang, Wenjun Gao, Wenqin Yu, Wentao Zhang, W. L. Xiao, Wei An, Xiaodong Liu, Xiaohan Wang, Xiaokang Chen, Xiaotao Nie, Xin Cheng, Xin Liu, Xin Xie, Xingchao Liu, Xinyu Yang, Xinyuan Li, Xuecheng Su, Xuheng Lin, X. Q. Li, Xiangyue Jin, Xiaojin Shen, Xiaosha Chen, Xiaowen Sun, Xiaoxiang Wang, Xinnan Song, Xinyi Zhou, Xianzu Wang, Xinxia Shan, Y. K. Li, Y. Q. Wang, Y. X. Wei, Yang Zhang, Yanhong Xu, Yao Li, Yao Zhao, Yaofeng Sun, Yaohui Wang, Yi Yu, Yichao Zhang, Yifan Shi, Yiliang Xiong, Ying He, Yishi Piao, Yisong Wang, Yixuan Tan, Yiyang Ma, Yiyuan Liu, Yongqiang Guo, Yuan Ou, Yuduan Wang, Yue Gong, Yuheng Zou, Yujia He, Yunfan Xiong, Yuxiang Luo, Yuxiang You, Yuxuan Liu, Yuyang Zhou, Y. X. Zhu, Yanhong Xu, Yanping Huang, Yaohui Li, Yi Zheng, Yuchen Zhu, Yunxian Ma, Ying Tang, Yukun Zha, Yuting Yan, Z. Z. Ren, Zehui Ren, Zhangli Sha, Zhe Fu, Zhean Xu, Zhenda Xie, Zhengyan Zhang, Zhewen Hao, Zhicheng Ma, Zhigang Yan, Zhiyu Wu, Zihui Gu, Zijia Zhu, Zijun Liu, Zilin Li, Ziwei Xie, Ziyang Song, Zizheng Pan, Zhen Huang, Zhipeng Xu, Zhongyu Zhang, and Zhen Zhang. Deepseek-r1: Incentivizing reasoning capability in llms via reinforcement learning, 2025. URL https://arxiv.org/abs/2501.12948.

Jacob Devlin, Ming-Wei Chang, Kenton Lee, and Kristina Toutanova. Bert: Pre-training of deep bidirectional transformers for language understanding. In *Proceedings of the 2019 conference of the North American chapter of the association for computational linguistics: human language technologies, volume 1 (long and short papers)*, pp. 4171–4186, 2019.

Abhimanyu Dubey, Abhinav Jauhri, Abhinav Pandey, Abhishek Kadian, Ahmad Al-Dahle, Aiesha Letman, Akhil Mathur, Alan Schelten, Amy Yang, Angela Fan, Anirudh Goyal,

Anthony Hartshorn, Aobo Yang, Archi Mitra, Archie Sravankumar, Artem Korenev, Arthur Hinsvark, Arun Rao, Aston Zhang, Aurelien Rodriguez, Austen Gregerson, Ava Spataru, Baptiste Roziere, Bethany Biron, Binh Tang, Bobbie Chern, Charlotte Caucheteux, Chaya Nayak, Chloe Bi, Chris Marra, Chris McConnell, Christian Keller, Christophe Touret, Chunyang Wu, Corinne Wong, Cristian Canton Ferrer, Cyrus Nikolaidis, Damien Allonsius, Daniel Song, Danielle Pintz, Danny Livshits, David Esiobu, Dhruv Choudhary, Dhruv Mahajan, Diego Garcia-Olano, Diego Perino, Dieuwke Hupkes, Egor Lakomkin, Ehab AlBadawy, Elina Lobanova, Emily Dinan, Eric Michael Smith, Filip Radenovic, Frank Zhang, Gabriel Synnaeve, Gabrielle Lee, Georgia Lewis Anderson, Graeme Nail, Gregoire Mialon, Guan Pang, Guillem Cucurell, Hailey Nguyen, Hannah Korevaar, Hu Xu, Hugo Touvron, Iliyan Zarov, Imanol Arrieta Ibarra, Isabel Kloumann, Ishan Misra, Ivan Evtimov, Jade Copet, Jaewon Lee, Jan Geffert, Jana Vranes, Jason Park, Jay Mahadeokar, Jeet Shah, Jelmer van der Linde, Jennifer Billock, Jenny Hong, Jenya Lee, Jeremy Fu, Jianfeng Chi, Jianyu Huang, Jiawen Liu, Jie Wang, Jiecao Yu, Joanna Bitton, Joe Spisak, Jongsoo Park, Joseph Rocca, Joshua Johnstun, Joshua Saxe, Junteng Jia, Kalyan Vasuden Alwala, Kartikeya Upasani, Kate Plawiak, Ke Li, Kenneth Heafield, Kevin Stone, Khalid El-Arini, Krithika Iyer, Kshitiz Malik, Kuenley Chiu, Kunal Bhalla, Lauren Rantala-Yeary, Laurens van der Maaten, Lawrence Chen, Liang Tan, Liz Jenkins, Louis Martin, Lovish Madaan, Lubo Malo, Lukas Blecher, Lukas Landzaat, Luke de Oliveira, Madeline Muzzi, Mahesh Pasupuleti, Mannat Singh, Manohar Paluri, Marcin Kardas, Mathew Oldham, Mathieu Rita, Maya Pavlova, Melanie Kambadur, Mike Lewis, Min Si, Mitesh Kumar Singh, Mona Hassan, Naman Goyal, Narjes Torabi, Nikolay Bashlykov, Nikolay Bogoychev, Niladri Chatterji, Olivier Duchenne, Onur Çelebi, Patrick Alrassy, Pengchuan Zhang, Pengwei Li, Petar Vasic, Peter Weng, Prajjwal Bhargava, Pratik Dubal, Praveen Krishnan, Punit Singh Koura, Puxin Xu, Qing He, Qingxiao Dong, Ragavan Srinivasan, Raj Ganapathy, Ramon Calderer, Ricardo Silveira Cabral, Robert Stojnic, Roberta Raileanu, Rohit Girdhar, Rohit Patel, Romain Sauvestre, Ronnie Polidoro, Roshan Sumbaly, Ross Taylor, Ruan Silva, Rui Hou, Rui Wang, Saghar Hosseini, Sahana Chennabasappa, Sanjay Singh, Sean Bell, Seohyun Sonia Kim, Sergey Edunov, Shaoliang Nie, Sharan Narang, Sharath Raparthy, Sheng Shen, Shengye Wan, Shruti Bhosale, Shun Zhang, Simon Vandenhende, Soumya Batra, Spencer Whitman, Sten Sootla, Stephane Collot, Suchin Gururangan, Sydney Borodinsky, Tamar Herman, Tara Fowler, Tarek Sheasha, Thomas Georgiou, Thomas Scialom, Tobias Speckbacher, Todor Mihaylov, Tong Xiao, Ujjwal Karn, Vedanuj Goswami, Vibhor Gupta, Vignesh Ramanathan, Viktor Kerkez, Vincent Gonguet, Virginie Do, Vish Vogeti, Vladan Petrovic, Weiwei Chu, Wenhan Xiong, Wenyin Fu, Whitney Meers, Xavier Martinet, Xiaodong Wang, Xiaoqing Ellen Tan, Xinfeng Xie, Xuchao Jia, Xuewei Wang, Yaelle Goldschlag, Yashesh Gaur, Yasmine Babaei, Yi Wen, Yiwen Song, Yuchen Zhang, Yue Li, Yuning Mao, Zacharie Delpierre Coudert, Zheng Yan, Zhengxing Chen, Zoe Papakipos, Aaditya Singh, Aaron Grattafiori, Abha Jain, Adam Kelsey, Adam Shajnfeld, Adithya Gangidi, Adolfo Victoria, Ahuva Goldstand, Ajay Menon, Ajay Sharma, Alex Boesenberg, Alex Vaughan, Alexei Baevski, Allie Feinstein, Amanda Kallet, Amit Sangani, Anam Yunus, Andrei Lupu, Andres Alvarado, Andrew Caples, Andrew Gu, Andrew Ho, Andrew Poulton, Andrew Ryan, Ankit Ramchandani, Annie Franco, Aparajita Saraf, Arkabandhu Chowdhury, Ashley Gabriel, Ashwin Bharambe, Assaf Eisenman, Azadeh Yazdan, Beau James, Ben Maurer, Benjamin Leonhardi, Bernie Huang, Beth Loyd, Beto De Paola, Bhargavi Paranjape, Bing Liu, Bo Wu, Boyu Ni, Braden Hancock, Bram Wasti, Brandon Spence, Brani Stojkovic, Brian Gamido, Britt Montalvo, Carl Parker, Carly Burton, Catalina Mejia, Changhan Wang, Changkyu Kim, Chao Zhou, Chester Hu, Ching-Hsiang Chu, Chris Cai, Chris Tindal, Christoph Feichtenhofer, Damon Civin, Dana Beaty, Daniel Kreymer, Daniel Li, Danny Wyatt, David Adkins, David Xu, Davide Testuggine, Delia David, Devi Parikh, Diana Liskovich, Didem Foss, Dingkang Wang, Duc Le, Dustin Holland, Edward Dowling, Eissa Jamil, Elaine Montgomery, Eleonora Presani, Emily Hahn, Emily Wood, Erik Brinkman, Esteban Arcaute, Evan Dunbar, Evan Smothers, Fei Sun, Felix Kreuk, Feng Tian, Firat Ozgenel, Francesco Caggioni, Francisco Guzmán, Frank Kanayet, Frank Seide, Gabriela Medina Florez, Gabriella Schwarz, Gada Badeer, Georgia Swee, Gil Halpern, Govind Thattai, Grant Herman, Grigory Sizov, Guangyi, Zhang, Guna Lakshminarayanan, Hamid Shojanazeri, Han Zou, Hannah Wang, Hanwen Zha, Haroun Habeeb, Harrison Rudolph,

Helen Suk, Henry Aspegren, Hunter Goldman, Ibrahim Damlaj, Igor Molybog, Igor Tufanov, Irina-Elena Veliche, Itai Gat, Jake Weissman, James Geboski, James Kohli, Japhet Asher, Jean-Baptiste Gaya, Jeff Marcus, Jeff Tang, Jennifer Chan, Jenny Zhen, Jeremy Reizenstein, Jeremy Teboul, Jessica Zhong, Jian Jin, Jingyi Yang, Joe Cummings, Jon Carvill, Jon Shepard, Jonathan McPhie, Jonathan Torres, Josh Ginsburg, Junjie Wang, Kai Wu, Kam Hou U, Karan Saxena, Karthik Prasad, Kartikay Khandelwal, Katayoun Zand, Kathy Matosich, Kaushik Veeraraghavan, Kelly Michelena, Keqian Li, Kun Huang, Kunal Chawla, Kushal Lakhotia, Kyle Huang, Lailin Chen, Lakshya Garg, Lavender A, Leandro Silva, Lee Bell, Lei Zhang, Liangpeng Guo, Licheng Yu, Liron Moshkovich, Luca Wehrstedt, Madian Khabsa, Manav Avalani, Manish Bhatt, Maria Tsimpoukelli, Martynas Mankus, Matan Hasson, Matthew Lennie, Matthias Reso, Maxim Groshev, Maxim Naumov, Maya Lathi, Meghan Keneally, Michael L. Seltzer, Michal Valko, Michelle Restrepo, Mihir Patel, Mik Vyatskov, Mikayel Samvelyan, Mike Clark, Mike Macey, Mike Wang, Miquel Jubert Hermoso, Mo Metanat, Mohammad Rastegari, Munish Bansal, Nandhini Santhanam, Natascha Parks, Natasha White, Navyata Bawa, Nayan Singhal, Nick Egebo, Nicolas Usunier, Nikolay Pavlovich Laptev, Ning Dong, Ning Zhang, Norman Cheng, Oleg Chernoguz, Olivia Hart, Omkar Salpekar, Ozlem Kalinli, Parkin Kent, Parth Parekh, Paul Saab, Pavan Balaji, Pedro Rittner, Philip Bontrager, Pierre Roux, Piotr Dollar, Polina Zvyagina, Prashant Ratanchandani, Pritish Yuvraj, Qian Liang, Rachad Alao, Rachel Rodriguez, Rafi Ayub, Raghotham Murthy, Raghu Nayani, Rahul Mitra, Raymond Li, Rebekkah Hogan, Robin Battey, Rocky Wang, Rohan Maheswari, Russ Howes, Ruty Rinott, Sai Jayesh Bondu, Samyak Datta, Sara Chugh, Sara Hunt, Sargun Dhillon, Sasha Sidorov, Satadru Pan, Saurabh Verma, Seiji Yamamoto, Sharadh Ramaswamy, Shaun Lindsay, Shaun Lindsay, Sheng Feng, Shenghao Lin, Shengxin Cindy Zha, Shiva Shankar, Shuqiang Zhang, Shuqiang Zhang, Sinong Wang, Sneha Agarwal, Soji Sajuyigbe, Soumith Chintala, Stephanie Max, Stephen Chen, Steve Kehoe, Steve Satterfield, Sudarshan Govindaprasad, Sumit Gupta, Sungmin Cho, Sunny Virk, Suraj Subramanian, Sy Choudhury, Sydney Goldman, Tal Remez, Tamar Glaser, Tamara Best, Thilo Kohler, Thomas Robinson, Tianhe Li, Tianjun Zhang, Tim Matthews, Timothy Chou, Tzook Shaked, Varun Vontimitta, Victoria Ajayi, Victoria Montanez, Vijai Mohan, Vinay Satish Kumar, Vishal Mangla, Vítor Albiero, Vlad Ionescu, Vlad Poenaru, Vlad Tiberiu Mihailescu, Vladimir Ivanov, Wei Li, Wenchen Wang, Wenwen Jiang, Wes Bouaziz, Will Constable, Xiaocheng Tang, Xiaofang Wang, Xiaojian Wu, Xiaolan Wang, Xide Xia, Xilun Wu, Xinbo Gao, Yanjun Chen, Ye Hu, Ye Jia, Ye Qi, Yenda Li, Yilin Zhang, Ying Zhang, Yossi Adi, Youngjin Nam, Yu, Wang, Yuchen Hao, Yundi Qian, Yuzi He, Zach Rait, Zachary DeVito, Zef Rosnbrick, Zhaoduo Wen, Zhenyu Yang, and Zhiwei Zhao. The llama 3 herd of models, 2024. URL https://arxiv.org/abs/2407.21783.

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.

Marton Havasi, Brian Karrer, Itai Gat, and Ricky TQ Chen. Edit flows: Flow matching with edit operations. *arXiv preprint arXiv:2506.09018*, 2025.

Edward J Hu, Yelong Shen, Phillip Wallis, Zeyuan Allen-Zhu, Yuanzhi Li, Shean Wang, Lu Wang, Weizhu Chen, et al. Lora: Low-rank adaptation of large language models. *ICLR*, 1(2):3, 2022.

Siming Huang, Tianhao Cheng, Jason Klein Liu, Jiaran Hao, Liuyihan Song, Yang Xu, J. Yang, J. H. Liu, Chenchen Zhang, Linzheng Chai, Ruifeng Yuan, Zhaoxiang Zhang, Jie Fu, Qian Liu, Ge Zhang, Zili Wang, Yuan Qi, Yinghui Xu, and Wei Chu. Opencoder: The open cookbook for top-tier code large language models. 2024. URL https://arxiv.org/pdf/2411.04905.

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.

Jaeyeon Kim, Lee Cheuk-Kit, Carles Domingo-Enrich, Yilun Du, Sham Kakade, Timothy Ngotiaoco, Sitan Chen, and Michael Albergo. Any-order flexible length masked diffusion, 2025a. URL https://arxiv.org/abs/2509.01025.

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*, 2025b.

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

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.

Wei Liu, Siya Qi, Xinyu Wang, Chen Qian, Yali Du, and Yulan He. Nover: Incentive training for language models via verifier-free reinforcement learning. In *Proceedings of the 2025 Conference on Empirical Methods in Natural Language Processing*, pp. 7450–7469, 2025.

Kevin P. Murphy. *Probabilistic Machine Learning: Advanced Topics*. MIT Press, 2023. URL http://probml.github.io/book2.

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.

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.

Long Ouyang, Jeff Wu, Xu Jiang, Diogo Almeida, Carroll L. Wainwright, Pamela Mishkin, Chong Zhang, Sandhini Agarwal, Katarina Slama, Alex Ray, John Schulman, Jacob Hilton, Fraser Kelton, Luke Miller, Maddie Simens, Amanda Askell, Peter Welinder, Paul Christiano, Jan Leike, and Ryan Lowe. Training language models to follow instructions with human feedback, 2022. URL https://arxiv.org/abs/2203.02155.

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.

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

Zhihong Shao, Peiyi Wang, Qihao Zhu, Runxin Xu, Junxiao Song, Xiao Bi, Haowei Zhang, Mingchuan Zhang, YK Li, Yang Wu, et al. Deepseekmath: Pushing the limits of mathematical reasoning in open language models. *arXiv preprint arXiv:2402.03300*, 2024.

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.

David Silver, Aja Huang, Chris J Maddison, Arthur Guez, Laurent Sifre, George Van Den Driessche, Julian Schrittwieser, Ioannis Antonoglou, Veda Panneershelvam, Marc Lanctot, et al. Mastering the game of go with deep neural networks and tree search. *nature*, 529(7587):484–489, 2016.

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.

Alon Talmor, Jonathan Herzig, Nicholas Lourie, and Jonathan Berant. Commonsenseqa: A question answering challenge targeting commonsense knowledge. In *Proceedings of the 2019 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies, Volume 1 (Long and Short Papers)*, pp. 4149–4158, 2019.

Qwen Team. Qwen2.5: A party of foundation models, September 2024. URL https://qwenlm.github.io/blog/qwen2.5/.

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

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*, 2025a.

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, 2025b. 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.

Dan Zhang, Sining Zhoubian, Ziniu Hu, Yisong Yue, Yuxiao Dong, and Jie Tang. Rest-mcts*: Llm self-training via process reward guided tree search. *Advances in Neural Information Processing Systems*, 37:64735–64772, 2024.

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.

Siyan Zhao, Devaansh Gupta, Qinqing Zheng, and Aditya Grover. d1: Scaling reasoning in diffusion large language models via reinforcement learning, 2025a. URL https://arxiv.org/abs/2504.12216.

Siyan Zhao, Mengchen Liu, Jing Huang, Miao Liu, Chenyu Wang, Bo Liu, Yuandong Tian, Guan Pang, Sean Bell, Aditya Grover, et al. Inpainting-guided policy optimization for diffusion large language models. *arXiv preprint arXiv:2509.10396*, 2025b.

Xiangxin Zhou, Zichen Liu, Anya Sims, Haonan Wang, Tianyu Pang, Chongxuan Li, Liang Wang, Min Lin, and Chao Du. Reinforcing general reasoning without verifiers. *arXiv preprint arXiv:2505.21493*, 2025. URL https://arxiv.org/abs/2505.21493.

A Accelerated sampling with Multi-token Entropy Decoding

As MDLMs learn the conditional distribution $p_\theta(x^j \mid \mathbf{x}_{\text{UNMASKED}})$ 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 $p_\theta(x^i, x^j \mid \mathbf{x}_{\text{UNMASKED}}) \neq p_\theta(x^i \mid \mathbf{x}_{\text{UNMASKED}}) p_\theta(x^j \mid \mathbf{x}_{\text{UNMASKED}})$.

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

$$\text{KL}\left(p_\theta(\mathbf{x}^A \mid \mathbf{x}_{\text{UNMASKED}}, \mathbf{c}) \middle| \prod_{i \in A} p_\theta(x^i \mid \mathbf{x}_{\text{UNMASKED}}, \mathbf{c}) \right) \leq \sum_{i \in A} H(x^i \mid \mathbf{x}_{\text{UNMASKED}}, \mathbf{c}) \tag{4}$$

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

In this work, we additionally 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 unmasked text $\mathbf{x}_{\text{UNMASKED}}$, a decoding threshold λ and a maximum number of tokens to be decoded k_{max}, we propose the following definition of the set A for selecting positions to un-mask. In MED, we sort the position entropies in an ascending order and decode positions that satisfy $H(x^i \mid \mathbf{x}_{\text{UNMASKED}}, \mathbf{c}) < \lambda$ and select k_{max} such tokens. If no position has entropy lower than λ, we choose the position with the lowest entropy. Whenever the selected positions satisfy the entropy threshold, eq. (4) is at most $\lambda|A| \leq \lambda k_{\text{max}}$. When no position satisfies the threshold, we instead decode the single minimum-entropy position as a fallback.

Ben-Hamu et al. (2025) previously proposed entropy-bound (EB)-sampler, an adaptive multi-token decoder for MDLMs, which 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. Wu et al. (2025a) also propose accelerating inference with MDLMs using KV-caching and parallel decoding. Similar to Ben-Hamu et al. (2025), they propose an adaptive greedy strategy Block-wise Confidence-aware Parallel Decoding (BPCD), and decode tokens with confidences above a fixed probability threshold. The method proposed in (Wu et al., 2025a) is designed to accelerate argmax sampling from MDLMs.

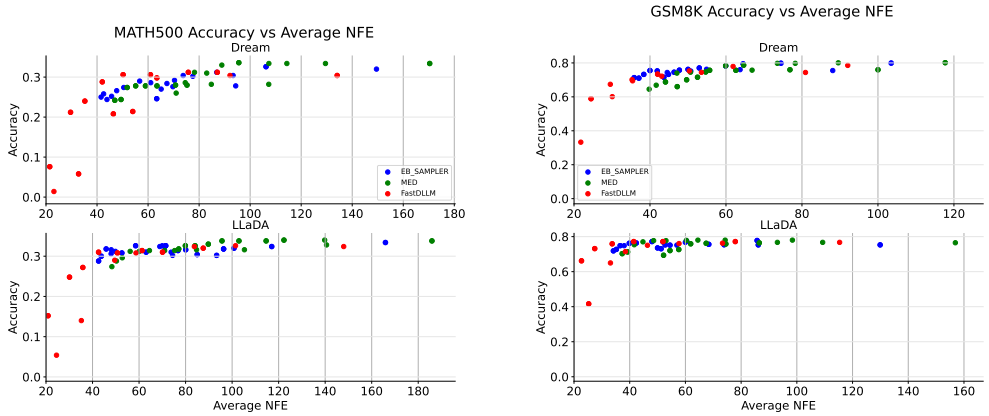

Figure 4: Adaptive decoding can reduce function evaluations while retaining competitive accuracy. In contrast to fixed token decoding, adaptive parallel decoding algorithms can preserve performance at selected operating points while reducing the number of function evaluations. Here, we plot results from MED alongside two recently proposed adaptive samplers, EB-Sampler (Ben-Hamu et al., 2025) and Block-wise confidence-aware parallel decoding (Wu et al., 2025a). In this experiment, we generate sequences of length $L \in \{128, 256\}$ with varying thresholds for the three samplers. The best sampler depends on task and NFE budget. For instance, for a $2\times$ reduction in NFEs, MED yields the highest accuracies on MATH500, while producing comparable numbers to EB-sampler for GSM8k.

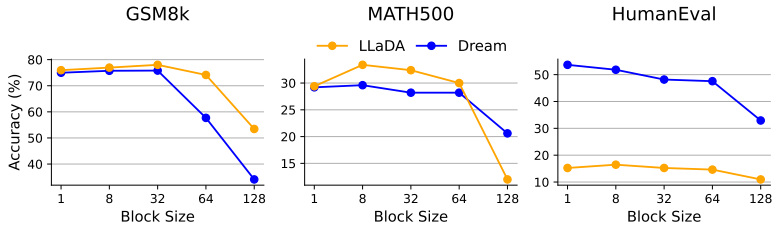

Figure 5: Any-order decoding hurts performance on reasoning tasks. When performing entropy decoding (Ye et al., 2025b), we observe that naive any-order sampling results in poor performance on all tasks. Left-to-right block decoding is required to make any-order sampling performant, and left-to-right sampling (block size $= 1$) is a strong baseline. We also observe in table 5 in appendix B that performant block any-order configurations sample a large portion of tokens left-to-right. We consider sequences of length 128.

In our sampler 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 $\lambda \in \{0.1, 0.2, 0.3, 0.4, 0.5, 0.6, 0.7, 0.8, 0.9\}$ in MED, EB-SAMPLER (Ben-Hamu et al., 2025), and BPCD (Wu et al., 2025a) with $k_{\max} = 32$ as the maximum number of tokens decoded in parallel. We fix a generation length $L \in \{128, 256\}$ and a block size of 32. In fig. 4, we compare the three adaptive decoding schemes with varying thresholds on the GSM8K, MATH500 datasets. We observe that the best sampler depends on task and NFE budget. For instance, for a $2\times$ reduction in NFEs, MED yields the highest accuracies on MATH500, while producing comparable numbers to EB-sampler for GSM8k.

B 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

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**	51.82	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 5: 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 on the non-Sudoku tasks, and on those tasks left-to-right sampling (block size = 1) is within a few percentage points of the best configuration. We also observe in appendix B 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.

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 include our results in table 5. On Sudoku, any-order sampling substantially improves performance.* 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) often leads to a large drop in performance outside Sudoku.

In what order are tokens decoded? In table 6 and table 7, 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 does block sampling improve performance? We find that purely any-order decoding from current MDLMs results in less auto-regressive generation. On GSM8K, it also produces fewer non-EOS tokens and much earlier answers, leaving more of the allocated generation length unused. 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*. We include an example of the latter behavior below.

GSM8K Example: Any-Order Entropy Decoding (LLaDA)

Question. Kylar went to the store to buy glasses for his new apartment. One glass costs $5, but every second glass costs only 60% of the price. Kylar wants to buy 16 glasses. How much does he need to pay for them?

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

Gold answer. 64

Final generation (incorrect).

```
Kylar wants to buy 16 glasses, so he needs to pay for 16/2 = 8 glasses at the
    full price.
The cost for these 8 glasses is 6 * $5 = $30.
The remaining 8 glasses are at the discounted price, which is $5 * 60% = $3.6.
The cost for these 8 glasses is 5 * $3.6 = $18.
In total, Kylar needs to pay $30 + $18 = 48.
Solution: 48
```

Decoding history (compressed).

```
Step 2:
<|eot_id|>

Step 5:
<...masks...>.<...masks...><|eot_id|>

Step 10:
<...masks...>8.
Solution: 48<|eot_id|>

Step 20:
<...masks...> $30 + $18 = 48.
Solution: 48<|eot_id|>

Step 40:
Kylar<...masks...> = $18.
In total, Kylar needs to pay $30 + $18 = 48.
Solution: 48<|eot_id|>

Step 127 (final output):
Kylar wants to buy 16 glasses, so he needs to pay for 16/2 = 8 glasses at the
    full price.
The cost for these 8 glasses is 6 * $5 = $30.
The remaining 8 glasses are at the discounted price, which is $5 * 60% = $3.6.
The cost for these 8 glasses is 5 * $3.6 = $18.
In total, Kylar needs to pay $30 + $18 = 48.
Solution: 48<|eot_id|>
```

Observation. Under any-order entropy decoding, LLaDA commits to the incorrect final answer early (by step 10), and subsequently fills in a post-hoc reasoning trace that is consistent with this answer.

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 ($n = 500$)

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	15.2%	29.7%	19.7	123.8

Table 7: Decoding behavior, HumanEval Similar to table 6, we measure autoregressiveness for generation lengths of 128 on HumanEval ($n = 164$).

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 successfully 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 1

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 Scoring Posterior Reasoning Chains with MDLMs

Given that posterior sampling yields both high-quality and low-quality reasoning chains, a natural question is: *Can we filter these chains without an external model?*

We find that answer block log probabilities computed with reasoning-as-infilling can be used to filter these reasoning chains, and identify traces that GPT4-o rates as correct. To do this, we iteratively unmask each generated posterior chain, left-to-right, and average the answer log probabilities over all time-steps. For LLaDA, we observe that correct reasoning traces correspond to higher average scores, see fig. 6. We also find that thresholding these answer-block log-probability scores yields AUC=0.74, see fig. 7.

Model	Generation method	GPT-4o score ↑
LLaDA-8B-Instruct	STaR-style hint	37.0%
LLaDA-8B-Instruct	Posterior reasoning	**44.1%**
LLaDA-8B-Base	STaR-style hint	13.5%
LLaDA-8B-Base	Posterior reasoning	**33.9%**

Table 8: Posterior reasoning vs. prefix-hint prompting, same checkpoints. On a 1000-question subset of the GSM8k *training* questions that LLaDA-8B-Instruct fails to solve, posterior conditioning on the answer block yields traces judged correct more often than STaR-style hint prompting (Zelikman et al., 2022) with the same model, with the largest gap for the base model.

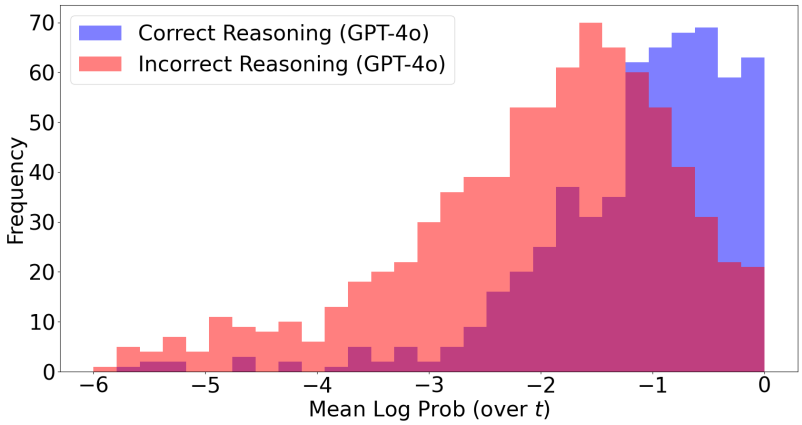

Figure 6: Distribution of answer block log probability scores for posterior samples. To score the 1419 posterior reasoning traces, we iteratively unmask each chain left-to-right, and compute the average answer block log probabilities across all timesteps. Posterior chains rated as correct by GPT4-o tend to have higher scores.

F Early Exit Samples

Below we demonstrate outputs from LLaDA on the GSM8k dataset. For a sample question from the test set, we generate a sequence of length 256 with the following settings:

- Sampling from the model, without using the reasoning template or early exits.
- Reasoning template with no exits.
- Reasoning template with varying exit thresholds.
- Reasoning template with the model producing the answer with no reasoning tokens decoded.

Example 1

Question: Janet's ducks lay 16 eggs per day. She eats three for breakfast every morning and bakes muffins for her friends every day with four. She sells the remainder at the farmers' market daily for $2 per fresh duck egg. How much in dollars does she make every day at the farmers' market?

LLaDA (No reasoning template, No Early Exits):

1

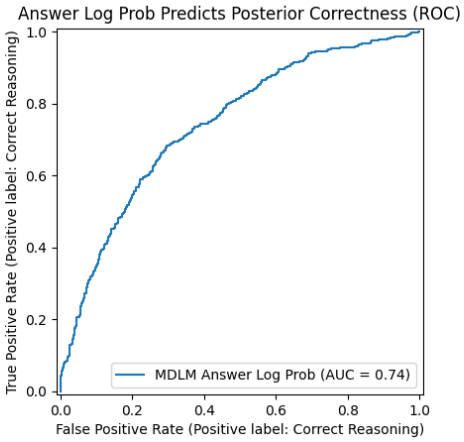

Figure 7: Answer block log probability scores predict posterior reasoning trace quality. The MDLM average answer block log probabilities can simply be thresholded to provide a classifier for predicting GPT4-o reasoning chain judgments. This classifier provides a potential method for filtering low-quality posterior samples (e.g. before fine-tuning).

```
To determine how much Janet makes every day at the farmers' market, we need to follow these steps:
1. Calculate the total number of eggs laid by Janet's ducks per day.
2. Subtract the number of eggs Janet eats and bakes for her friends from the total number of eggs laid.
3. Determine the number of eggs she sells at the farmers' market.
4. Calculate the total revenue from selling the eggs.
Step 1: Janet's ducks lay 16 eggs per day.
Step 2: Janet eats 3 eggs for breakfast every morning and bakes 4 eggs for her friends every day. Therefore, she eats and bakes a total of \(3 + 4 = 7\) eggs per day.
Step 3: The number of eggs she sells at the farmers' market is \(16 - 7 = 9\) eggs per day.
Step 4: Janet sells each egg for $2. Therefore, the total revenue from selling 9 eggs per day is \(9 \times 2 = 18\) dollars.
So, the amount Janet makes every day at the farmers' market is \(\boxed{18}\).
```

LLaDA output (Reasoning template, No Early Exit):

```


To determine how much Janet makes every day at the farmers' market, we need to follow these steps:
Step 1: Calculate the total number of eggs laid by Janet's ducks per day.
Janet's ducks lay 16 eggs per day.
Step 2: Calculate the number of eggs Janet eats for breakfast per day.
Janet eats 3 eggs for breakfast every day.
Step 3: Calculate the number of eggs Janet bakes for her friends per day.
Janet bakes 4 eggs for her friends every day.
Step 4: Calculate the number of eggs left after Janet eats some and bakes some.
Total eggs laid - Eggs eaten for breakfast - Eggs bakes for friends = Eggs left
16 - 3 - 4 = 9
So, Janet has 9 eggs left to sell.
Step 5: Calculate the revenue from selling the eggs at the farmers' market.
Janet sells each egg for $2.
Eggs left * Price per egg = Total revenue
9 * 2 = 18
Therefore, Janet makes $18 every day at the farmers' market.
The answer is \boxed{18}.
```

Dataset	Method	Acc. (%) ↑	Avg. NFE ↓	Speedup ↑
CommonsenseQA	No exit	72.3	256.0	1.00×
	Prophet (Li et al., 2025)	72.2	251.9	1.02×
	Entropy exit, γ=1.0	74.7	171.3	1.49×
	Entropy exit, γ=1.2	**76.2**	137.6	1.86×
	Entropy exit, γ=1.4	75.7	102.7	**2.49**×
GSM8K	No exit	78.8	256.0	1.00×
	Prophet (Li et al., 2025)	74.3	164.5	1.56×
	Entropy exit, γ=0.05	78.5	225.6	1.13×
	Entropy exit, γ=0.1	78.0	198.5	1.29×
	Entropy exit, γ=0.2	74.8	164.8	1.55×

Table 9: Comparison with Prophet and early-exit threshold sensitivity. We evaluate Prophet (Li et al., 2025) using its published configuration. On CommonsenseQA, entropy-based exits are more accurate and faster at every evaluated threshold. On GSM8k, entropy exit matches Prophet at a comparable NFE budget and offers a smooth accuracy–speed tradeoff at lower thresholds.

LLaDA output (Reasoning template, Early Exit($\gamma = 0.5$)):

```
To determine how much Janet makes every day at the farmers' market, we need to
    follow these steps:
Step 1: Calculate the total number of eggs laid by Janet's ducks per day.
<|mdm_mask|>...<|mdm_mask|> per<|mdm_mask|>...<|mdm_mask|> The answer is
    \boxed{18}.
```

LLaDA output (Reasoning template, Early Exit($\gamma = 0.7$)):

```
To determine<|mdm_mask|>...<|mdm_mask|> The answer is \boxed{18}.
```

LLaDA output (Forced Answer):

```
<|mdm_mask|>...<|mdm_mask|>. The answer is \boxed{14}.
```

G Early-Exit Baseline Comparison and Threshold Sensitivity

We compare entropy-based early exits with Prophet (Li et al., 2025) on GSM8k and CommonsenseQA (Talmor et al., 2019). We use a generation length of 256 with block size 32, matching the early-exit setup in the main paper, and run Prophet using the configuration provided in the original paper.

The results are shown in table 9. On CommonsenseQA, entropy-based exits are both more accurate and faster than Prophet at every evaluated threshold. On GSM8k, entropy exit matches Prophet at a comparable NFE budget (+0.5 percentage points at ∼1.55×) and offers a smooth accuracy and speed tradeoff at lower thresholds.

H Autoregressive Answer-Likelihood Baselines

In this section, we provide details for the autoregressive answer-likelihood baselines reported in fig. 3.

For each problem in the GSM8k test set (1319 problems), each autoregressive model (Llama-3.1-8B-Instruct and Qwen2.5-7B-Instruct) generates its own reasoning trace greedily (temperature 0, one sample per problem) with a maximum budget of 128 tokens. Every 5 generated tokens, we perform a single additional forward pass in which the answer delimiter *" The answer is \boxed{"* is appended to the partial trace, and we compute

$$\phi_{\text{AR}}(\mathbf{r}_{\leq t}) := \sum_j \log p(a_j^* \mid \mathbf{c}, \mathbf{r}_{\leq t}, \text{delimiter}),$$

the sum of the log-probabilities of the reference answer tokens. Checkpoints are binned into ten equal fractions of each sample's total generation length, and within each bin we

compute the Pearson correlation between ϕ_{AR} and the model's own final answer correctness. The full results are shown in fig. 3.

We restrict the analysis to generations that terminate within the 128-token budget (Llama: 703/1319, Qwen: 578/1319). This restriction is required to match the 128-token generation budget used for the MDLM curves in fig. 3. Even in this setting, intermediate answer likelihoods for both models are substantially less predictive of final correctness than the MDLM-defined intermediate score, ϕ_{IPRM}, through most of the generation.

I Proofs

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

$$\text{KL}\left(p_\theta(\mathbf{x}^A \mid \mathbf{x}_{\text{UN-MASKED}}, \mathbf{c}) \middle| \prod_{i \in A} p_\theta(x^i \mid \mathbf{x}_{\text{UN-MASKED}}, \mathbf{c})\right) \leq \sum_{i \in A} H(x^i \mid \mathbf{x}_{\text{UN-MASKED}}, \mathbf{c}) \tag{5}$$

Note that:

$$\begin{aligned}
&\text{KL}\left(p_\theta(\mathbf{x}^A \mid \mathbf{x}_{\text{UN-MASKED}}, \mathbf{c}) \middle| \prod_{i \in A} p_\theta(x^i \mid \mathbf{x}_{\text{UN-MASKED}}, \mathbf{c})\right) \\
&= \mathbf{E}_{p_\theta(\mathbf{x}^A \mid \mathbf{x}_{\text{UN-MASKED}}, \mathbf{c})}\left[\log p_\theta(\mathbf{x}^A \mid \mathbf{x}_{\text{UN-MASKED}}, \mathbf{c}) - \log \prod_{i \in A} p_\theta(x^i \mid \mathbf{x}_{\text{UN-MASKED}}, \mathbf{c})\right] && (6) \\
&= \mathbf{E}_{p_\theta(\mathbf{x}^A \mid \mathbf{x}_{\text{UN-MASKED}}, \mathbf{c})}\left[\log p_\theta(\mathbf{x}^A \mid \mathbf{x}_{\text{UN-MASKED}}, \mathbf{c}) - \sum_{i \in A} \log p_\theta(x^i \mid \mathbf{x}_{\text{UN-MASKED}}, \mathbf{c})\right] && (7) \\
&= -H(\mathbf{x}^A \mid \mathbf{x}_{\text{UN-MASKED}}, \mathbf{c}) + \sum_{i \in A} H(\mathbf{x}^i \mid \mathbf{x}_{\text{UN-MASKED}}, \mathbf{c}) && (8)
\end{aligned}$$

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

$$-H(\mathbf{x}^A \mid \mathbf{x}_{\text{UN-MASKED}}, \mathbf{c}) + \sum_{i \in A} H(\mathbf{x}^i \mid \mathbf{x}_{\text{UN-MASKED}}, \mathbf{c}) \leq \sum_{i \in A} H(\mathbf{x}^i \mid \mathbf{x}_{\text{UN-MASKED}}, \mathbf{c}) \tag{9}$$

Hence, for any set A:

$$\text{KL}\left(p_\theta(\mathbf{x}^A \mid \mathbf{x}_{\text{UN-MASKED}}, \mathbf{c}) \middle| \prod_{i \in A} p_\theta(x^i \mid \mathbf{x}_{\text{UN-MASKED}}, \mathbf{c})\right) \leq \sum_{i \in A} H(x^i \mid \mathbf{x}_{\text{UN-MASKED}}, \mathbf{c}) \tag{10}$$

Entropy upper-bound Next, we prove that:

$$H(\mathbf{a} \mid \mathbf{r}_{\text{UN-MASKED}}, \mathbf{c}) \leq H_{\text{UB}}(\mathbf{a} \mid \mathbf{r}_{\text{UN-MASKED}}, \mathbf{c}) \tag{11}$$

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 $\text{KL}\left(p_\theta(\mathbf{a} \mid \mathbf{r}_{\text{UN-MASKED}}, \mathbf{c}) \middle| \prod p_\theta(a^i \mid \mathbf{r}_{\text{UN-MASKED}}, \mathbf{c})\right) \geq 0$, which implies that, similar to eq. (8), we have

$$\begin{aligned}
-H(\mathbf{a} \mid \mathbf{r}_{\text{UN-MASKED}}, \mathbf{c}) + H_{\text{UB}}(\mathbf{a} \mid \mathbf{r}_{\text{UN-MASKED}}, \mathbf{c}) &= \text{KL}\left(p_\theta(\mathbf{a} \mid \mathbf{r}_{\text{UN-MASKED}}, \mathbf{c}) \middle| \prod p_\theta(a^i \mid \mathbf{r}_{\text{UN-MASKED}}, \mathbf{c})\right) && (12) \\
-H(\mathbf{a} \mid \mathbf{r}_{\text{UN-MASKED}}, \mathbf{c}) + H_{\text{UB}}(\mathbf{a} \mid \mathbf{r}_{\text{UN-MASKED}}, \mathbf{c}) &\geq 0 && (13) \\
H_{\text{UB}}(\mathbf{a} \mid \mathbf{r}_{\text{UN-MASKED}}, \mathbf{c}) &\geq H(\mathbf{a} \mid \mathbf{r}_{\text{UN-MASKED}}, \mathbf{c}) && (14)
\end{aligned}$$

Note that, by eq. (8), the gap of this bound equals the KL divergence between the joint answer-block distribution and the product of its marginals: the bound is tight exactly when the answer tokens are conditionally independent given the unmasked reasoning and context.

J Fine-tuning Details

We compare fine-tuning the LLaDA-8B Base (Nie et al., 2025) model on GSM8k (Cobbe et al., 2021) reasoning data, versus posterior data sampled from the same model using the training questions and pre-filled answers.

Data We generate the posterior data with LLaDA-8B-Base by pre-filling the correct answer and reasoning template and sampling with entropy decoding and a block size of 128. For the gold GSM8k training data, we preprocess the data by removing the additional computations in angle brackets, and converting the "####" format to our reasoning template. Additionally, unlike posterior data, GSM8k reasoning traces are of varying lengths. As a result, we truncate these traces to $L = 144$ tokens. When traces are longer, we truncate to the *last* 144 tokens. Approximately $\sim 12\%$ of GSM8k samples are truncated.

Training We use a batch size of 1 per GPU, with 8 different noise levels per batch element. We use LoRA (Hu et al., 2022) with $r = 128, \alpha = 32$. We fine-tune the model using 2 Nvidia A100 GPUs, with a learning rate of 2.5×10^{-6}, and 32 gradient accumulation steps. We train both models for 3300 steps.

We modify the supervised fine-tuning code provided by Zhao et al. (2025a).

Sampling We greedily sample from both models left-to-right with a block size of 1. We allocate 144 tokens for both models, and do not directly pre-fill a reasoning template.