Accelerated Sampling from Masked Diffusion Models via Entropy Bounded Unmasking

Heli Ben-Hamu **Itai Gat** **Daniel Severo** **Niklas Nolte** **Brian Karrer**

FAIR, Meta AI

Abstract

Recent masked diffusion models (MDMs) have shown competitive performance compared to autoregressive models (ARMs) for language modeling. While most literature has focused on performance enhancing sampling procedures, efficient sampling from MDMs has been scarcely explored. We make the observation that often a given sequence of partially masked tokens determines the values of multiple unknown tokens deterministically, meaning that a single prediction of a masked model holds additional information unused by standard sampling procedures. Based on this observation, we introduce *EB-Sampler*, a simple drop-in replacement for existing samplers, utilizing an **E**ntropy **B**ounded unmasking procedure that dynamically unmasks multiple tokens in one function evaluation with predefined approximate error tolerance. We formulate the EB-Sampler as part of a broad family of adaptive samplers for which we provide an error analysis that motivates our algorithmic choices. EB-Sampler accelerates sampling from current state of the art MDMs by roughly 2-3x on standard coding and math reasoning benchmarks without loss in performance. We also validate the same procedure works well on smaller reasoning tasks including maze navigation and Sudoku, tasks ARMs often struggle with.

1 Introduction

Motivated by the success of diffusion and flow Esser et al. (2024); Polyak et al. (2024) models in continuous domains (*e.g.*, images, video), research efforts have focused on adapting these frameworks to discrete state spaces. Many of these approaches (Austin et al., 2021a; Lou et al., 2023; Sahoo et al., 2024; Campbell et al., 2024; Gat et al., 2024; Shi et al., 2024; Kitouni et al., 2024) utilize the masked modeling paradigm, thus we generally refer to these approaches as Masked Diffusion Models (MDMs). Recent large MDMs such as LLaDa (Nie et al., 2025b) and Dream (Ye et al., 2025) have shown competitive performance compared to similarly sized autoregressive models (ARMs) in the 7 billion parameter range on traditional language tasks including code, text, and mathematical reasoning benchmarks. These results open the possibility of scaled non-autoregressive generation for language using MDMs, a possible competitor to large language models (LLMs).

Abstractly, MDMs sample fixed-length sequences as discrete tokens from a vocabulary. They begin from a sequence of mask tokens and iteratively update tokens until all tokens are unmasked. Replacing a masked token with a token from the vocabulary is *unmasking*. In contrast to standard ARMs, the order in which tokens are unmasked becomes a design choice. Recent models often leverage *samplers* that surpass the performance of random order unmasking, deviating from the masked diffusion process used at training (Austin et al., 2021a). In particular, LLaDa and Dream achieve strong performance by unmasking tokens in a much more favorable order dictated by model predictions (Chang et al., 2022; Kim et al., 2025; Zheng et al., 2023).

39th Conference on Neural Information Processing Systems (NeurIPS 2025).

Performance is not the only criteria by which large MDMs have become popular. Another additional important aspect is efficient computation. While MDMs cannot reuse past computation via key-value caching like ARMs due to full attention, MDMs offer an exciting alternative route to efficiency via the opportunity to sample multiple tokens simultaneously. MDM efficiency is captured by the number of function evaluations (NFE) which is directly controlled via the sampler. The default for more efficient sampling of MDMs like LLaDa and Dream is to unmask a fixed number of tokens each step. Unfortunately, generation quality degrades quickly as more unmasked tokens are sampled independently and hence this opportunity has been thus far unrealized. Developing an MDM sampler for language that simultaneously achieves good performance and efficiency is the focus of this work.

Our approach is motivated by two empirical observations. The first is the strong performance of unmasking orders determined by the model, which indicates the associated predictions aligned with that order may have low model error, i.e. match the desired optimal distribution. The second is that multiple masked tokens are often predicted with high certainty, or, equivalently, low entropy. Intuitively, masked tokens that have both low model error and low entropy can be unmasked in parallel, which we propose to exploit with our approach. By strategically unmasking such tokens, a sampler can follow high performance unmasking orders while avoiding error due to independently unmasking tokens.

We realize these intuitions and make them concrete and precise via our main contributions:

- We propose an adaptive sampler for masked diffusion models called *EB-Sampler* (short for **E**ntropy **B**ounded Sampler), that decides both which tokens to unmask and how many tokens to unmask at each step using an entropy bound that approximately limits the dependence of unmasked tokens.
- EB-Sampler is a simple drop-in replacement to existing samplers and can be directly applied to sample from existing masked diffusion models without further training.
- EB-Sampler accelerates sampling from the best performing masked diffusion models (LLaDa and Dream) by 2-3x on standard coding and math reasoning benchmarks without loss in performance. We provide Pareto fronts between efficiency and performance and conclude EB-Sampler is superior across most settings. We also validate EB-Sampler works well on smaller reasoning tasks, including maze navigation and Sudoku.
- We show EB-Sampler is a member of a broad family of adaptive multi-token samplers for masked diffusion models. Our theory explains why this family samples correctly, why it can leverage a pre-trained masked diffusion model, and provides an intuitive error decomposition into model error and joint dependence error that motivates the design of EB-Sampler.

2 Preliminaries

2.1 Notations

Consider a discrete *state space* denoted $\mathcal{S} = \mathcal{T}^d$, of sequences of length d over a finite sized *vocabulary* $\mathcal{T} = [K] = \{1, 2, \ldots, K\}$. A state, $x \in \mathcal{S}$, is a sequence of elements, often called *tokens*, from the vocabulary $\mathcal{T}$, where $x = (x^1, x^2, \ldots, x^d)$ with each element $x^l \in \mathcal{T}$. We denote the set of indices of a sequence in $\mathcal{S}$ with $\mathcal{I} = [d] = \{1, 2, \ldots, d\}$. For masked modeling, we extend our vocabulary to include a mask token, $\mathtt{m}$. If index l or token x^l is described as masked, then $x^l = \mathtt{m}$ and if unmasked, $x^l \neq \mathtt{m}$. Finally, for a random variable $X = (X^1, X^2, \ldots, X^d)$ and a set $A \subseteq \mathcal{I}$, we define the notation for conditional probabilities $p(x^l|x^A) = \mathbb{P}(X^l = x^l|\{X^i = x^i|i \in A\})$.

2.2 Masked diffusion models

Given a finite set of samples $\mathcal{D} \subseteq \mathcal{S}$ drawn from an unknown target data distribution $X \sim q$, most MDMs, with few exceptions, have been proven to learn to model clean data conditionals $q(x^l|x^{\bar{M}})$ (Ou et al., 2025; Zheng et al., 2024) where $\bar{M} \subseteq \mathcal{I}$ is a set, possibly empty, of unmasked token indices. The main difference from BERT-like masked language modeling (Devlin et al., 2019) is that BERT-like models are trained on a fixed masking ratio, as opposed to all possible masks. In particular, the MDMs we consider learn factorized conditional predictions of $p^\theta(x^l|x^{\bar{M}})$, using full attention, for all l masked tokens, that ideally match $q(x^l|x^{\bar{M}})$, for every possible $\bar{M}$. Sampling

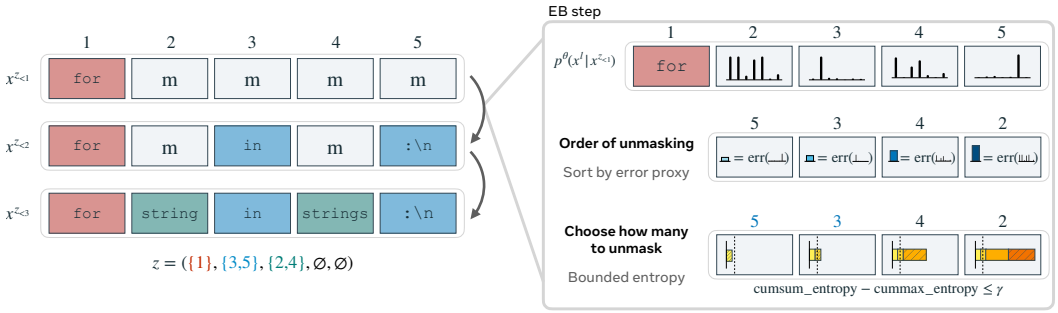


Figure 1: Illustration of an unmasking step with EB-Sampler. At each step EB sampler determines which tokens to unmask by ordering according to an error proxy and then chooses how many tokens to independently unmask by bounding their joint dependence via model prediction entropies.

with trained factorized conditionals can be done sequentially, unmasking one token at a time. This baseline sampler can be defined as follows: At each step (i) randomly pick an index to unmask, *e.g.*, l; (ii) sample from learned factorized conditional distribution $X^l \sim p^\theta(x^l|x^{\bar{M}})$, where $\bar{M}$ is the set of currently masked tokens. We refer to this sampling procedure as *random unmasking*.

3 Known challenges of MDM sampling

We now discuss two challenges in sampling from MDMs. The connections we draw between these challenges will support the intuition for the construction of the EB-Sampler in the next section.

3.1 Order of unmasking matters

For the optimal factorized conditionals predictor, the order of sequential unmasking will not change the underlying model distribution, $p^\theta(x)$. That is, by the chain rule of probability, for any two permutations (orders of unmasking) σ, σ':

$$p^\theta_\sigma(x) = p^\theta_{\sigma'}(x)$$

where $p^\theta_\sigma = \prod_{l=1}^d p^\theta(x^{\sigma(l)}|x^{\cup_{j<l}\sigma(j)})$. Nevertheless, training an optimal p^θ is intractable due to both learning on extremely high dimensional state spaces with limited model capacity and finite data and no hard constraints on the modeling itself such that the chain rule holds. Formally, it means that there exists *local model error* $D_{\mathrm{KL}}(q(x^l|x^{\bar{M}}), p^\theta(x^l|x^{\bar{M}})) > 0$. A key consequence is that the *total model error*, *i.e.*, the discrepancy between the data distribution, $q(x)$, and the model distribution for a certain order of unmasking, $p^\theta_\sigma(x)$, depends on the *order of unmasking*. A question that arises then is, do MDMs manage to learn such there are unmasking orders with low total model error? If so, can one find local model error proxies such that the chosen unmasking order will result in low total model error?

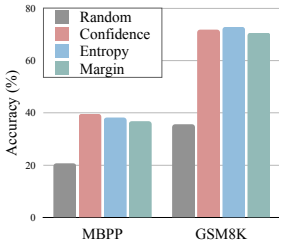


Figure 2: Performance of greedy sampling with various unmasking criteria from LLaDa 8B Base model.

Recent works (Nie et al., 2025a,b; Kim et al., 2025; Ye et al., 2025) proposed sequential greedy sampling with unmasking orders dictated by one of the following criteria (Chang et al., 2022; Kim et al., 2025; Zheng et al., 2023) for choosing the next coordinate l to unmask:

$$\begin{aligned}
\text{Confidence: } l &= \operatorname*{argmax}_{l' \in M} [\max_{x^{l'}} p^\theta(x^{l'}|x^{\bar{M}})] \\
\text{Entropy: } l &= \operatorname*{argmin}_{l' \in M} [H(X^{l'}|X^{\bar{M}} = x^{\bar{M}})] \\
\text{Margin: } l &= \operatorname*{argmax}_{l' \in M} [p^\theta(X^{l'} = y_1|x^{\bar{M}}) - p^\theta(X^{l'} = y_2|x^{\bar{M}})]
\end{aligned} \tag{1}$$

where $(y_1, y_2) = \arg\operatorname{Top}_2 \underset{x^{l'}}{p^\theta}(x^{l'}|x^{\bar{M}})$, where $\arg\operatorname{Top}_2$ returns the two token values with maximal probability, and $H(p) = -\sum_x p(x)\log p(x)$ is the entropy of p. Greedy samplers show superior performance compared to random unmasking samplers, as shown in Figure 2, closing the gap in performance between MDMs and ARMs (Nie et al., 2025b; Ye et al., 2025). This answers the first part of the question above, indicating the existence of orders with lower total model error. As for the second part, on how to find those unmasking orders, Figure 2 shows that the criteria in Equation (1) can serve as local model error proxies, determining an unmasking order with low(er) total model error.

3.2 Sampling efficiency

The best performing sampling procedures for language MDMs described above, similarly to ARMs, predict one token per function evaluation, hence the efficiency of MDMs remains a disadvantage compared to ARMs due to costly computations of full attention that does not allow KV-caching. An avenue for improving sampling efficiency of MDMs is to make use of the model predictions on all masked tokens to unmask multiple tokens per function evaluation. Common multi-token unmasking procedures unmask a fixed number of tokens, k, at each step by sampling independently from the predicted factorized conditionals. We refer to these approaches as Top-k sampling, and they can all be cast in terms of choosing the Top-k lowest model error proxy tokens to be unmasked at each step. The parameter k tunes an efficiency-accuracy tradeoff. The larger k is, the larger the *joint dependence error* will be, as it wrongly assumes independence of a fixed number of tokens at every step. Figure 3 empirically shows the degradation in performance in Top-k sampling for $k \in \{1, 2, 4, 8, 16\}$ with various error proxies.

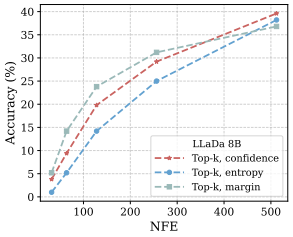


Figure 3: Efficiency-accuracy tradeoff of Top-k (NFE) sampling on MBPP.

4 Entropy Bounded (EB) Sampler

The previous section described how challenges in MDM sampling come from two distinct sources of error: local model error and joint dependence error. Controlling these errors motivates our *Entropy Bounded (EB) Sampler*, a direct replacement for Top-k samplers.

Section 3.1 found that not only do masked tokens with low model error likely exist, but that past research has already identified proxies computable via model predictions that identify these tokens in Equation (1). A step in EB-Sampler begins by sorting unmasked tokens in ascending order on this error proxy, exactly as in Top-k samplers. Then Section 3.2 showed Top-k sampling accumulates substantial joint dependence error that harms performance by sampling tokens independently. We now make the additional observation that MDM predictions are often highly confident about multiple masked tokens simultaneously. These tokens are predicted to have low dependence in the data distribution q, because these tokens are all predicted to have low entropy. EB-Sampler therefore defines a threshold $\gamma \geq 0$ and decides to unmask the largest subset U of sorted masked tokens such that

$$\sum_{l \in U} H(p^\theta(x^l|x^{\bar{M}})) - \max_{l \in U} H(p^\theta(x^l|x^{\bar{M}})) \leq \gamma. \tag{2}$$

When U is comprised of low model error tokens, this expression approximately bounds a rigorous joint dependence error introduced later in Section 5. Figure 1 visually illustrates a step of EB-Sampler.

Python code showing Top-k sampling and EB-Sampler is provided in Figure 4, where EB-Sampler is shown to be a minimal change in PyTorch. Like Top-k sampling, EB-Sampler is easily compatible with ad hoc adjustments like temperature and unsupervised classifier-free guidance that alter p^θ.

Consider this code with the same inputs. Different outputs are only from EB-Sampler determining the value of k using the entropy bound of Equation (2). When unmasked tokens have low dependence, EB-Sampler will unmask more tokens, and conversely, when there is (potentially) high dependence, EB-Sampler will unmask less tokens. The amount of tokens unmasked per step in EB-Sampler is therefore not fixed. More function evaluations are used when the sample is predicted complex and less when the sample is predicted simple. The threshold γ influences the number of steps where $\gamma = 0$ will unmask one token each step and $\gamma = \infty$ will unmask all tokens at once.

```python
def top_k_step(x, model, sample_fn, error_proxy_fn, k):
    p = model(x)
    err = error_proxy_fn(p)
    err = torch.where(x == model.mask_id, err, np.inf)
    _ , ids = torch.sort(err, dim=-1)



    k = torch.minimum(k, (x == model.mask_id).sum())
    ids_to_unmask = ids[:k]
    x[ids_to_unmask] = sample_fn(p, ids_to_unmask)
    return x
```

```python
def EB_step(x, model, sample_fn, error_proxy_fn, gamma):
    p = model(x)
    err = error_proxy_fn(p)
    err = torch.where(x == model.mask_id, err, np.inf)
    _ , ids = torch.sort(err, dim=-1)
+   entropy = torch.distributions.Categorical(probs=p).entropy()[ids]
+   acc_entropy = torch.cumsum(entropy)
+   cummax_entropy = torch.cummax(entropy, dim=0).values
+   k = (acc_entropy - cummax_entropy <= gamma).sum()
    k = torch.minimum(k, (x == model.mask_id).sum())
    ids_to_unmask = ids[:k]
    x[ids_to_unmask] = sample_fn(p, ids_to_unmask)
    return x
```

Figure 4: Python code implementation of a single sampling step for common Top-k approaches and for EB-Sampler.

5 Adaptive unmasking samplers

In this section, we formulate the EB-Sampler as a member of a more general family of adaptive multi-token samplers. The object we use to mathematically describe varying length unmasking steps is an ordered partition $\mathcal{I}$, denoted by $z = (z_1, z_2, \ldots, z_d)$, where z satisfies:

$$\bigcup_{i=1}^{d} z_i = \mathcal{I}, \quad z_i \cap z_j = \varnothing \tag{3}$$

and either $z_i \subseteq \mathcal{I}$ or $z_i = \varnothing$. Notation $z_{<i}$ denotes ordered sub-partitions up to index i, that is $z_{<i} = (z_j | j \in [i-1])$.

For a random variable $X = (X^1, X^2, \ldots, X^d)$ over $\mathcal{S}$ and ordered sub-partitions s, s', we extend the notation from Section 2.1 for conditional probabilities:

$$p(x^s | x^{s'}) = \mathbb{P}\left(\left\{X^l = x^l,\ \forall l \in \mathcal{I}_s\right\} \,\middle|\, \left\{X^j = x^j,\ \forall j \in \mathcal{I}_{s'}\right\}\right) \tag{4}$$

where $\mathcal{I}_s = \bigcup_{i=1}^{|s|} s_i$ and $\mathcal{I}_{s'} = \bigcup_{i=1}^{|s'|} s'_i$. Depending on context, we will also be using the notation in 4 with s being a set of indices, *e.g.*, $s = z_i$ or a single index, *e.g.*, $s = l$.

We consider a broad family of sampling procedures defined by ϕ that leverage approximate clean data conditionals provided by p^θ, and produce a joint distribution $p_\phi(x, z)$ over state x and partition z:

$$p_\phi(x, z) = \prod_{i=1}^{d} p_\phi(z_i, x^{z_i} | x^{z_{<i}}, z_{<i}) = \prod_{i=1}^{d} \left(\prod_{l \in z_i} p^\theta(x^l | x^{z_{<i}}) \right) \phi(z_i | x^{z_{<i}}, z_{<i}). \tag{5}$$

The distributions ϕ enforce z is a valid partition, and sampling z_i means token indices z_i are unmasked at step i. Without loss of generality, ϕ always unmasks at least one token if possible, i.e. only samples $z_i = \varnothing$ when $z_{<i} = \mathcal{I}$. Similarly define

$$q_\phi(x, z) = \prod_{i=1}^{d} q(z_i, x^{z_i} | x^{z_{<i}}, z_{<i}) = \prod_{i=1}^{d} q(x^{z_i} | x^{z_{<i}}) \phi(z_i | x^{z_{<i}}, z_{<i}). \tag{6}$$

Importantly, $q_\phi(x, z) = q(x) \prod_{i=1}^{d} \phi(z_i | x^{z_{<i}}, z_{<i})$, because the product of the clean data conditionals does not depend on the order of unmasking, z, so that $\sum_z q_\phi(x, z) = q(x)$ for any ϕ.

Expressiveness of ϕ This family encompasses existing common samplers that always unmask the same number of tokens on each step, such as deterministic samplers of Top-k smallest margin, Top-k smallest entropy, and Top-k confidence, or simple random unmasking, but also contains additional options, including dynamically determining the number of tokens to unmask at each step.

Error decomposition We quantify the error from sampling $p_\phi(x) = \sum_z p_\phi(x, z)$ instead of $q(x)$ via KL divergence

$$D_{\mathrm{KL}}(q(x), p_\phi(x)) \leq D_{\mathrm{KL}}(q_\phi(x, z), p_\phi(x, z)) = \sum_{i=1}^{d} \mathbb{E}_{q_\phi}[\ln q(x^{z_i} | x^{z_{<i}}) - \sum_{l \in z^i} \ln p^\theta(x^l | x^{z_{<i}})].$$

Appendix A.1 discusses when this is an equality, and proves this can be rewritten into two terms

$$\sum_{i=1}^{d} \mathbb{E}_{q_\phi}[\underbrace{\sum_{l \in z_i} D_{\mathrm{KL}}(q(x^l|x^{z<i}), p^\theta(x^l|x^{z<i}))}_{\text{model error}} + \underbrace{D_{\mathrm{KL}}(q(x^{z_i}|x^{z<i}), \prod_{l \in z_i} q(x^l|x^{z<i}))}_{\text{joint dependence error}}]. \tag{7}$$

Model error comes from sampling incorrect conditionals p^θ that do not match the data distribution. *Joint dependence error* comes from sampling tokens independently that are not actually independent in q. This second KL divergence is precisely joint mutual information, upper-bounded by

$$D_{\mathrm{KL}}(q(x^{z_i}|x^{z<i}), \prod_{l \in z_i} q(x^l|x^{z<i})) \leq \sum_{l \in z_i} H(q(x^l|x^{z<i})) - \max_{l \in z_i} H(q(x^l|x^{z<i})). \tag{8}$$

Choosing ϕ given a pre-trained model EB-Sampler is directly motivated by this error decomposition. The p^θ that achieves zero model error is the same for any ϕ, justifying using any pre-trained model learned to match clean data conditionals. Assume we can identify low model error tokens and we design ϕ to only select z_i from such tokens, where for all $l \in z_i$, $p^\theta(x^l|x^{z<i}) \approx q(x^l|x^{z<i})$. Then model error is negligible and joint dependence error is approximately upper-bounded by

$$\sum_{l \in z_i} H(p^\theta(x^l|x^{z<i})) - \max_{l \in z_i} H(p^\theta(x^l|x^{z<i})), \tag{9}$$

our criteria in Equation (2). So after adaptively identifying low model error tokens, ϕ can control overall error by selecting subsets of such tokens to unmask with bounded joint dependence. EB-Sampler applies this approach with the model error proxies from Section 3.1.

6 Experiments

We evaluate the performance of the EB-Sampler on standard code and math reasoning generation tasks and on logic puzzles solving. The empirical findings in this section support our theoretical derivations and demonstrate the proposed sampler's capabilities.

Baselines. We compare the EB-Sampler to Top-k samplers with the three error proxy functionals described in 1: (i) confidence; (ii) entropy; and (iii) margin.

Experimental setting. On all tasks and models we follow the same general experimental setting. We test the EB-Sampler with a range of thresholds, γ, depending on the task. For the Top-k samplers we test a range of k values. Each point in the plots (*e.g.*, Figure 5) corresponds to the resulting model performance when sampling with parameter γ or k for EB or Top-k sampler respectively. After finding temperature was unhelpful for LLaDa, we used zero temperature sampling for all experiments. Full details on threshold and k values can be found in Appendix C.

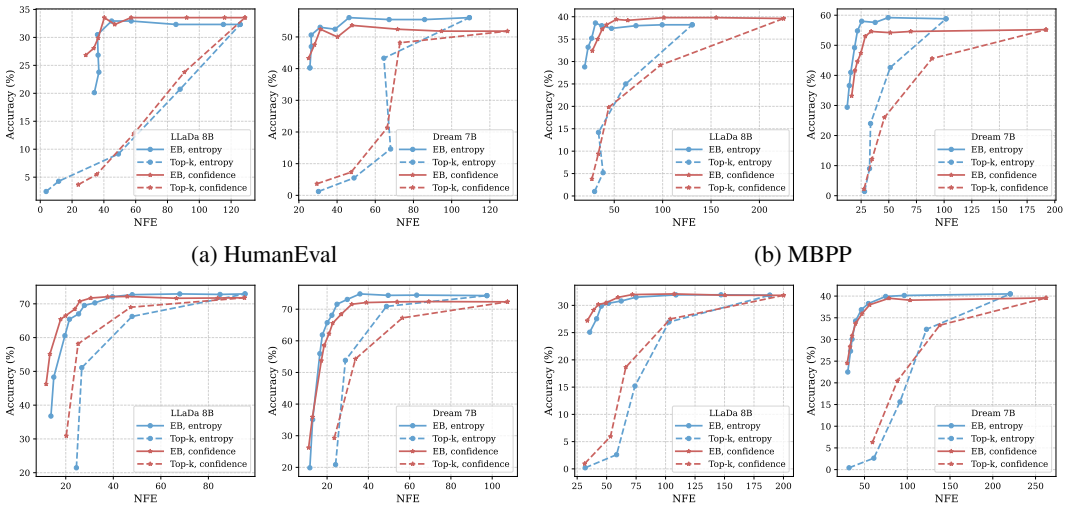
6.1 Code and math reasoning

We evaluate the performance of the EB-Sampler variants on text generation tasks in which (i) success can be measured quantitatively; and (ii) require an answer that is longer than a single token. These two properties facilitate quantitative assessment of the efficiency-accuracy tradeoff a sampler exhibits.

Models. We report results on two recent open source state-of-the-art language MDMs: LLaDa 8B Base (Nie et al., 2025b) and Dream 7B Base (Ye et al., 2025).

Benchmarks. We use 4 widely used benchmarks. HumanEval (0 shot) (Chen et al., 2021) and MBPP (4 shot) (Austin et al., 2021b) code generation benchmarks; and GSM8K (8 shot) (Cobbe et al., 2021) and Math (4 shot) (Hendrycks et al., 2021) math reasoning benchmarks. We note that we use different variants of GSM8K and Math in our evaluation compared to reported results for Dream 7B, hence the minor gaps in performance for standard Top-1 sampling.

Setup. For each task, we follow common practice and set a maximal number of generated tokens, `max_gen_len`. Then, the prompt is padded to the right with mask tokens until the maximal sequence length of the model. During sampling, only masked tokens in the range of $[\texttt{len(prompt)}, \texttt{len(prompt)} + \texttt{max\_gen\_len}]$ are allowed to be unmasked. Generation stops once all tokens in the designated range have been unmasked. In the main text, we show results for the two best performing error proxy functions entropy and confidence. Results with margin are in Appendix D.



(a) HumanEval (b) MBPP

(c) GSM8K (d) MATH

Figure 5: pass@1 accuracy vs. NFE with `generate_until` logic on code and math reasoning tasks.

6.1.1 Measuring efficiency gains of MDM samplers

In this part, we describe how we measure the gains obtained by our proposed sampler. We explain why current generation practices for MDMs require rethinking and propose to add a `generate_until` logic to save function evaluations. We then further observe that unlike ARMs that generate in a left-to-right order, a `generate_until` logic may still be suboptimal for MDM efficient generation.

`generate_until` logic. Current generation practices for MDMs fix an apriori amount of tokens to be generated, *i.e.*, `max_gen_len`, and generate until all tokens are unmasked. This is rather wasteful as typically producing a response to a given prompt will require less tokens than the full length, that is `answer_len` $<$ `max_gen_len`. We therefore propose to incorporate a `generate_until` logic in MDM samplers similar to ARM generation practices. For example, for the MBPP benchmark, the few shot samples given as context to the model include a concluding phrase "[DONE]". In ARM evaluation procedures, this phrase is used as a stopping criterion. For MDMs, however, due to the non left-to-right order of unmasking, we extend the `generate_until` logic to include an additional condition that all tokens in indices preceding the concluding phrase are unmasked.

All experiments were run with the `generate_until` logic applied as a post-process in order to measure gains both against full `max_gen_len` and effective generation length to stopping criterion. The NFE reported in Figure 5 measures average number of function evaluations until a task specific `generate_until` logic is satisfied. EB-Sampler consistently improves upon accuracy-NFE Pareto frontier across all datasets and error proxies, gaining speed-ups of 2-4x compared to Top-1 sampler at the same accuracy. In Figures 9 and 10 we show the same plots against full `max_gen_len` NFE.

Bias in function evaluation count. Surprisingly, under the `generate_until` post-process we still observed high NFE counts at $\gamma = 0$ compared to expected answer lengths. This is most pronounced on the MBPP benchmark, where the model finds it "easy" to repeat the instructions given to the model after the concluding phrase although it did not finish unmasking all the masks before that. To get a better estimate of the actual speed-up gains achieved by the EB-Sampler we take MBPP as a test case and adopt the semi-AR block generation scheme from (Arriola et al., 2025; Nie et al., 2025b) to restrict the model from generating tokens that are far from the context. In Table 1 we compare the average NFE at roughly the same performance for the various strategies of controlling the generation length. We note that for this dataset and the sampling strategies evaluated in the table the mean answer length is around 50 tokens, while the semi-AR block generation requires 64.59 NFE, thus not fully resolving the bias. On the contrary, EB-Sampler requires 21.19 function evaluations to get the same performance with same average answer length, generating tokens at a rate of 2.4 tokens per step. We therefore refrain from claiming a 6x speed up as it compares to a loose generation procedure and believe a better estimate of the EB-Sampler's gains is around 2-3x. Table 1 provides three key insights on MDM efficiency evaluation. First, measuring the efficiency gains of samplers for MDMs

Table 1: NFE and Speed-Ups for Dream 7B on the MBPP for various evaluation schemes at roughly same best pass@1. For all configurations in the table the mean answer length is ~ 50 tokens.

		Full `max_gen_len` $= 512$		`generate_until` logic		`generate_until` logic + semi-AR (block_len=64)		
	pass@1	NFE	Speed-Up	NFE	Speed-Up	pass@1	NFE	Speed-Up
Top-1	58.8%	512	x 1	101.71	x 1	58.8%	64.59	x 1
EB, entropy, $\gamma = 0.001$	59.2%	174.57	x 2.93	49.39	x 2.05	59%	38.90	x 1.66
EB, entropy, $\gamma = 0.1$	58%	85.33	x 6.00	25.49	x 3.99	58.6%	21.19	x 3.05

is non-trivial and depends on the generation configuration, second, MDMs invest computation on generating tokens that are not used and this opens up an important practical question for the future use of MDMs, and lastly, EB-Sampler shows strong performance in efficiency gains across all settings.

6.2 Logic puzzles

Discrete diffusion models have been shown to excel on logic puzzles such as maze navigation, Sudoku and more (Nolte et al., 2024; Ye et al., 2024). We investigate whether these strong performances can be retained while sampling more efficiently than one token at a time. Specifically, we train small scale discrete diffusion models on Sudoku and Maze navigation problems and evaluate their performance on held-out data when varying the sampling strategies.

6.2.1 Maze navigation

We use the maze generation methods for "DFS mazes" in (Nolte et al., 2024), see their Section 4.1. We generate $48K$ mazes for training and $2K$ for validation. All mazes are defined on a grid of size 10x10. They are serialized into tokens by enumerating all the connections between cells in the grid, i.e. the edges in the graph defined by the maze. To aid learning the invariance with respect to edge reordering, the edges are shuffled before tokenization. A 6 million parameter discrete DiT model, using code adapted from (Lou et al., 2023), is trained to optimize the masked diffusion objective, without explicit time dependence (Kitouni et al., 2024; Ou et al., 2025).

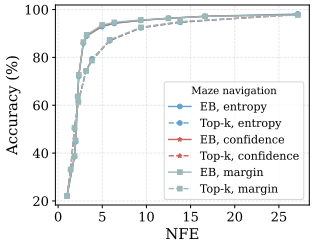


Figure 6: 10x10 mazes - accuracy vs average NFE.

The metric used for performance comparison is the accuracy, defined as the fraction of validation mazes fully solved. Figure 6 shows the accuracy as a function of average NFE over the validation set for the different ordering metrics. All metrics perform extremely similarly, thus partially obscured in the plot. All strategies work best in the single token unmasking regime. EB-Sampler preserves most of the accuracy before experiencing a sharp drop-off at less than 5 NFEs. In contrast, the Top-k baselines exhibit a steeper decline in performance, already at around 10 NFEs.

6.2.2 Sudoku

To further assess sampling strategies in structured logic problems, we evaluate performance on the task of Sudoku completion. Unlike maze navigation, which emphasizes path finding, Sudoku requires reasoning over dense, globally constrained grids. This setting provides a complementary benchmark to test sampling strategies under those global constraints.

We adopt the standard 9×9 Sudoku setting and adapt the code from Alp (2024) to generate $48K$ training puzzles and $2K$ held-out puzzles with, all with unique solutions. Each puzzle is serialized into a sequence of 89 tokens corresponding to the cell values and end-of-line tokens, with zeros indicating blank cells. The discrete DiT model architecture used for maze navigation is reused here.

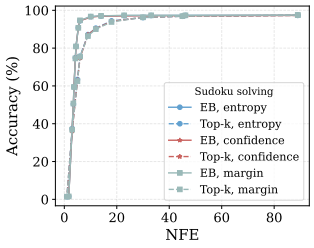


Figure 7: Sudoku - accuracy vs average NFE.

We again show accuracy as a function of average NFE. The results are depicted in Figure 7. The trend is similar to the trend in the maze navigation setup. EB-Sampler retains most of its performance for a longer time than the Top-k samplers, and then drops off sharply. Notably, almost full performance is retained even when averaging around 10-15 NFEs.

7 Related Work

Performant sampling for discrete diffusion. Procedures that improve sampling from MDMs are often focused on improving performance for a given pre-trained model. Recent approaches consider *planning* (Kim et al., 2025), deciding which masked tokens should be unmasked next, as well as *remasking* (Wang et al., 2025), deciding which unmasked tokens should be masked again, related to predictor-corrector iterations (Gat et al., 2024; Lezama et al., 2022) and forward-backwards sampling (Campbell et al., 2024), or consider both planning and remasking in (Zheng et al., 2023; Peng et al., 2025; Liu et al., 2024). Like EB-Sampler, this research often considers KL (equivalently ELBO) bounds. Unlike past research though, we focus on multi-token adaptive planning with a variable-sized set of tokens to unmask, crucial for efficiency. While sampling for LLaDa (Nie et al., 2025b) was described as *remasking*, it can be viewed as determining what to unmask first and then their token values within semi-autoregressive blocks, a member of our ϕ family. Because we aim for efficient and justified planning for scaled MDMs, we do not consider revisiting unmasked tokens here, enabling a simpler, time-independent analysis with a KL bound that is minimized via adaptive sampling. Future research might devise an efficient multi-token sampler that both unmasks and revisits past tokens upon EB-Sampler. One option might be to combine EB-Sampler with predictor-corrector iterations suggested in (Zhao et al., 2024) that update unmasked variables but do not change masks.

Efficient sampling for discrete diffusion. Efficiency has received relatively less attention than performance. (Ren et al., 2025) proposed higher-order numerical solvers for discrete diffusion, not specific to MDMs. (Park et al., 2024) introduced a method to avoid joint dependence error from parallel sampling, by performing a global optimization for a non-adaptive sampling schedule (i.e. the number of tokens per step). EB-Sampler determines this adaptively per sampling step. (Besnier et al., 2025) introduced an unmasking sampler for MaskGIT (Chang et al., 2022) that controls joint dependence error via quasi-random, low-discrepancy unmasking of an image, outperforming a confidence-based sampler in that domain. Finally, recent research (Zhu et al., 2025) has proposed a distillation procedure for MDMs, and trained one-step image generators from multi-step MaskGIT (Chang et al., 2022) and Messionic (Bai et al., 2024). Other works, analyzing the convergence and error of discrete diffusion models such as Chen and Ying (2024); Liang et al. (2025); Zhang et al. (2025) could potentially enable extension of our analysis of time-independent masked model to time dependent discrete diffusion.

Speculative decoding. A prominent method to accelerate LLMs is speculative decoding, with 2-2.5x speedup in (Chen et al., 2023). Instead of sampling from a large *target* language model, speculative decoding generates a candidate sequence from a smaller *draft* language model and accepts some portion of that candidate utilizing sequence probabilities. Modified rejection sampling guarantees the sequence is extended and this extension is sampled from the target model. Because evaluating models with causal attention can be done cheaply compared to sampling, efficiency gains occur when the draft model quickly samples reasonable sequences. This procedure has also been adapted to any-order masked models (Uria et al., 2014; Hoogeboom et al., 2022) with causal attention (Pannatier et al., 2024; Guo and Ermon, 2025). For MDMs with full attention, we cannot directly apply speculative decoding because it is generally expensive to compute the sequence probability in a full attention target model. (De Bortoli et al., 2025) has applied speculative sampling to continuous Gaussian diffusion, but relies upon querying the target model in parallel. Speculative decoding has been combined with discrete diffusion in (Christopher et al., 2024), where the draft is an MDM and the target is a language model. Our EB-Sampler approach is complementary, and could be simply applied to speed up the draft model.

8 Conclusions and Future Work

In this paper we propose EB-Sampler, a theoretically grounded adaptive sampler for masked discrete diffusion and flow models. This algorithm controls both which and how many tokens to sample using an interpretable entropy bound and serves as a drop-in replacement for existing samplers. We evaluate our approach on math, code and reasoning benchmarks with contemporary diffusion models of different scales, against the standard samplers used by the authors of those models. We find that EB-Sampler significantly outperforms existing samplers on the compute-vs-performance Pareto frontier and even yields 2-3x speed-ups without any loss of performance.

Future research might consider learning a parameterized adaptive sampler from data, perhaps optimizing our KL bound with respect to ϕ, or expand upon EB-Sampler to incorporate revisiting past unmasked tokens. Such research could further advance efficient and performant sampling for masked diffusion models.

References

Alp, A. (2024). Sudoku. `https://github.com/alicommit-malp/sudoku/tree/main`.

Arriola, M., Sahoo, S. S., Gokaslan, A., Yang, Z., Qi, Z., Han, J., Chiu, J. T., and Kuleshov, V. (2025). Block diffusion: Interpolating between autoregressive and diffusion language models. In *The Thirteenth International Conference on Learning Representations*.

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

Austin, J., Odena, A., Nye, M., Bosma, M., Michalewski, H., Dohan, D., Jiang, E., Cai, C., Terry, M., Le, Q., et al. (2021b). Program synthesis with large language models. *arXiv preprint arXiv:2108.07732*.

Bai, J., Ye, T., Chow, W., Song, E., Chen, Q.-G., Li, X., Dong, Z., Zhu, L., and Yan, S. (2024). Meissonic: Revitalizing masked generative transformers for efficient high-resolution text-to-image synthesis. In *The Thirteenth International Conference on Learning Representations*.

Besnier, V., Chen, M., Hurych, D., Valle, E., and Cord, M. (2025). Halton scheduler for masked generative image transformer.

Campbell, A., Yim, J., Barzilay, R., Rainforth, T., and Jaakkola, T. (2024). Generative flows on discrete state-spaces: Enabling multimodal flows with applications to protein co-design. *arXiv preprint arXiv:2402.04997*.

Chang, H., Zhang, H., Jiang, L., Liu, C., and Freeman, W. T. (2022). Maskgit: Masked generative image transformer. In *Proceedings of the IEEE/CVF conference on computer vision and pattern recognition*, pages 11315–11325.

Chen, C., Borgeaud, S., Irving, G., Lespiau, J.-B., Sifre, L., and Jumper, J. (2023). Accelerating large language model decoding with speculative sampling. *arXiv preprint arXiv:2302.01318*.

Chen, H. and Ying, L. (2024). Convergence analysis of discrete diffusion model: Exact implementation through uniformization.

Chen, M., Tworek, J., Jun, H., Yuan, Q., Pinto, H. P. D. O., Kaplan, J., Edwards, H., Burda, Y., Joseph, N., Brockman, G., et al. (2021). Evaluating large language models trained on code. *arXiv preprint arXiv:2107.03374*.

Christopher, J. K., Bartoldson, B. R., Ben-Nun, T., Cardei, M., Kailkhura, B., and Fioretto, F. (2024). Speculative diffusion decoding: Accelerating language generation through diffusion. *arXiv preprint arXiv:2408.05636*.

Cobbe, K., Kosaraju, V., Bavarian, M., Chen, M., Jun, H., Kaiser, L., Plappert, M., Tworek, J., Hilton, J., Nakano, R., et al. (2021). Training verifiers to solve math word problems. *arXiv preprint arXiv:2110.14168*.

De Bortoli, V., Galashov, A., Gretton, A., and Doucet, A. (2025). Accelerated diffusion models via speculative sampling. *arXiv preprint arXiv:2501.05370*.

Devlin, J., Chang, M.-W., Lee, K., and Toutanova, K. (2019). Bert: Pre-training of deep bidirectional transformers for language understanding. In *Proceedings of the 2019 conference of the North American chapter of the association for computational linguistics: human language technologies, volume 1 (long and short papers)*, pages 4171–4186.

Esser, P., Kulal, S., Blattmann, A., Entezari, R., Müller, J., Saini, H., Levi, Y., Lorenz, D., Sauer, A., Boesel, F., Podell, D., Dockhorn, T., English, Z., Lacey, K., Goodwin, A., Marek, Y., and Rombach, R. (2024). Scaling rectified flow transformers for high-resolution image synthesis.

Gat, I., Remez, T., Shaul, N., Kreuk, F., Chen, R. T. Q., Synnaeve, G., Adi, Y., and Lipman, Y. (2024). Discrete flow matching. *arXiv preprint arXiv:2407.15595*.

Guo, G. and Ermon, S. (2025). Reviving any-subset autoregressive models with principled parallel sampling and speculative decoding. *arXiv preprint arXiv:2504.20456*.

Hendrycks, D., Burns, C., Kadavath, S., Arora, A., Basart, S., Tang, E., Song, D., and Steinhardt, J. (2021). Measuring mathematical problem solving with the math dataset. *arXiv preprint arXiv:2103.03874*.

Hoogeboom, E., Gritsenko, A. A., Bastings, J., Poole, B., van den Berg, R., and Salimans, T. (2022). Autoregressive diffusion models. In *International Conference on Learning Representations*.

Kim, J., Shah, K., Kontonis, V., Kakade, S., and Chen, S. (2025). Train for the worst, plan for the best: Understanding token ordering in masked diffusions. *arXiv preprint arXiv:2502.06768*.

Kingma, D. P. and Welling, M. (2019). An introduction to variational autoencoders. *CoRR*, abs/1906.02691.

Kitouni, O., Nolte, N., Hensman, J., and Mitra, B. (2024). Disk: A diffusion model for structured knowledge.

Lezama, J., Salimans, T., Jiang, L., Chang, H., Ho, J., and Essa, I. (2022). Discrete predictor-corrector diffusion models for image synthesis. In *The Eleventh International Conference on Learning Representations*.

Liang, Y., Huang, R., Lai, L., Shroff, N., and Liang, Y. (2025). Absorb and converge: Provable convergence guarantee for absorbing discrete diffusion models.

Liu, S., Nam, J., Campbell, A., Stärk, H., Xu, Y., Jaakkola, T., and Gómez-Bombarelli, R. (2024). Think while you generate: Discrete diffusion with planned denoising. *arXiv preprint arXiv:2410.06264*.

Lou, A., Meng, C., and Ermon, S. (2023). Discrete diffusion modeling by estimating the ratios of the data distribution. *arXiv preprint arXiv:2310.16834*.

Nie, S., Zhu, F., Du, C., Pang, T., Liu, Q., Zeng, G., Lin, M., and Li, C. (2025a). Scaling up masked diffusion models on text. In *The Thirteenth International Conference on Learning Representations*.

Nie, S., Zhu, F., You, Z., Zhang, X., Ou, J., Hu, J., Zhou, J., Lin, Y., Wen, J.-R., and Li, C. (2025b). Large language diffusion models.

Nolte, N., Kitouni, O., Williams, A., Rabbat, M., and Ibrahim, M. (2024). Transformers can navigate mazes with multi-step prediction.

Ou, J., Nie, S., Xue, K., Zhu, F., Sun, J., Li, Z., and Li, C. (2025). Your absorbing discrete diffusion secretly models the conditional distributions of clean data.

Pannatier, A., Courdier, E., and Fleuret, F. (2024). σ-gpts: A new approach to autoregressive models. In *Joint European Conference on Machine Learning and Knowledge Discovery in Databases*, pages 143–159. Springer.

Park, Y.-H., Lai, C.-H., Hayakawa, S., Takida, Y., and Mitsufuji, Y. (2024). : Optimizing sampling schedule of discrete diffusion models. *CoRR*.

Peng, F. Z., Bezemek, Z., Patel, S., Rector-Brooks, J., Yao, S., Tong, A., and Chatterjee, P. (2025). Path planning for masked diffusion model sampling. *arXiv preprint arXiv:2502.03540*.

Polyak, A., Zohar, A., Brown, A., Tjandra, A., Sinha, A., Lee, A., Vyas, A., Shi, B., Ma, C.-Y., Chuang, C.-Y., Yan, D., Choudhary, D., Wang, D., Sethi, G., Pang, G., Ma, H., Misra, I., Hou, J., Wang, J., Jagadeesh, K., Li, K., Zhang, L., Singh, M., Williamson, M., Le, M., Yu, M., Singh, M. K., Zhang, P., Vajda, P., Duval, Q., Girdhar, R., Sumbaly, R., Rambhatla, S. S., Tsai, S., Azadi, S., Datta, S., Chen, S., Bell, S., Ramaswamy, S., Sheynin, S., Bhattacharya, S., Motwani, S., Xu, T., Li, T., Hou, T., Hsu, W.-N., Yin, X., Dai, X., Taigman, Y., Luo, Y., Liu, Y.-C., Wu, Y.-C., Zhao, Y., Kirstain, Y., He, Z., He, Z., Pumarola, A., Thabet, A., Sanakoyeu, A., Mallya, A., Guo, B., Araya, B., Kerr, B., Wood, C., Liu, C., Peng, C., Vengertsev, D., Schonfeld, E., Blanchard, E., Juefei-Xu, F., Nord, F., Liang, J., Hoffman, J., Kohler, J., Fire, K., Sivakumar, K., Chen, L., Yu, L., Gao, L., Georgopoulos, M., Moritz, R., Sampson, S. K., Li, S., Parmeggiani, S., Fine, S., Fowler, T., Petrovic, V., and Du, Y. (2024). Movie gen: A cast of media foundation models.

Ren, Y., Chen, H., Zhu, Y., Guo, W., Chen, Y., Rotskoff, G. M., Tao, M., and Ying, L. (2025). Fast solvers for discrete diffusion models: Theory and applications of high-order algorithms. *arXiv preprint arXiv:2502.00234*.

Sahoo, S. S., Arriola, M., Gokaslan, A., Marroquin, E. M., Rush, A. M., Schiff, Y., Chiu, J. T., and Kuleshov, V. (2024). Simple and effective masked diffusion language models. In *The Thirty-eighth Annual Conference on Neural Information Processing Systems*.

Shi, J., Han, K., Wang, Z., Doucet, A., and Titsias, M. (2024). Simplified and generalized masked diffusion for discrete data. In *The Thirty-eighth Annual Conference on Neural Information Processing Systems*.

Uria, B., Murray, I., and Larochelle, H. (2014). A deep and tractable density estimator. In Xing, E. P. and Jebara, T., editors, *Proceedings of the 31st International Conference on Machine Learning*, volume 32 of *Proceedings of Machine Learning Research*, pages 467–475, Bejing, China. PMLR.

Wang, G., Schiff, Y., Sahoo, S. S., and Kuleshov, V. (2025). Remasking discrete diffusion models with inference-time scaling. *arXiv preprint arXiv:2503.00307*.

Ye, J., Gao, J., Gong, S., Zheng, L., Jiang, X., Li, Z., and Kong, L. (2024). Beyond autoregression: Discrete diffusion for complex reasoning and planning. *arXiv preprint arXiv:2410.14157*.

Ye, J., Xie, Z., Zheng, L., Gao, J., Wu, Z., Jiang, X., Li, Z., and Kong, L. (2025). Dream 7b.

Zhang, Z., Chen, Z., and Gu, Q. (2025). Convergence of score-based discrete diffusion models: A discrete-time analysis.

Zhao, Y., Shi, J., Chen, F., Druckmann, S., Mackey, L., and Linderman, S. (2024). Informed correctors for discrete diffusion models. *arXiv preprint arXiv:2407.21243*.

Zheng, K., Chen, Y., Mao, H., Liu, M.-Y., Zhu, J., and Zhang, Q. (2024). Masked diffusion models are secretly time-agnostic masked models and exploit inaccurate categorical sampling. *arXiv preprint arXiv:2409.02908*.

Zheng, L., Yuan, J., Yu, L., and Kong, L. (2023). A reparameterized discrete diffusion model for text generation. *arXiv preprint arXiv:2302.05737*.

Zhu, Y., Wang, X., Lathuilière, S., and Kalogeiton, V. (2025). Di[M]o: Distilling masked diffusion models into one-step generator. *arXiv preprint arXiv:2503.15457*.

NeurIPS Paper Checklist

1. **Claims**

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

   Answer: [Yes] .

   Justification: Abstract and introduction list the paper's contribution and scope in the context of existing literature.

   Guidelines:

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

2. **Limitations**

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

   Answer: [Yes]

   Justification: The focus of this work is MDM sampling efficiency. Section 6.1.1 discusses the challenges and limitations of faithfully evaluating efficiency of MDM samplers.

   Guidelines:

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

3. **Theory assumptions and proofs**

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

Answer: [Yes]

Justification: See Appendix A.

Guidelines:

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

4. **Experimental result reproducibility**

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

   Answer: [Yes]

   Justification: We evaluate open source models on public benchmarks. Minimal code implementation of the core method is shown in Figure 4 and experimental details are listed in Appendix.

   Guidelines:

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

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

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

Answer: [No]

Justification: We evaluate open source models on public benchmarks. Minimal code implementation of the core method is shown in Figure 4 and experimental details are listed in Appendix.

Guidelines:

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

6. **Experimental setting/details**

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

   Answer: [Yes]

   Justification: Experiment section in paper and Appendix.

   Guidelines:

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

7. **Experiment statistical significance**

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

   Answer: [NA]

   Justification: The proposed method is deterministic and hence reporting error bars on experiments is not applicable.

   Guidelines:

   - The answer NA means that the paper does not include experiments.
   - The authors should answer "Yes" if the results are accompanied by error bars, confidence intervals, or statistical significance tests, at least for the experiments that support the main claims of the paper.

- The factors of variability that the error bars are capturing should be clearly stated (for example, train/test split, initialization, random drawing of some parameter, or overall run with given experimental conditions).
- The method for calculating the error bars should be explained (closed form formula, call to a library function, bootstrap, etc.)
- The assumptions made should be given (e.g., Normally distributed errors).
- It should be clear whether the error bar is the standard deviation or the standard error of the mean.
- It is OK to report 1-sigma error bars, but one should state it. The authors should preferably report a 2-sigma error bar than state that they have a 96% CI, if the hypothesis of Normality of errors is not verified.
- For asymmetric distributions, the authors should be careful not to show in tables or figures symmetric error bars that would yield results that are out of range (e.g. negative error rates).
- If error bars are reported in tables or plots, The authors should explain in the text how they were calculated and reference the corresponding figures or tables in the text.

8. **Experiments compute resources**

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

   Answer: [Yes]

   Justification: In Appendix.

   Guidelines:

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

9. **Code of ethics**

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

   Answer: [Yes]

   Justification: Yes.

   Guidelines:

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

10. **Broader impacts**

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

    Answer: [Yes]

    Justification: See Appendix.

    Guidelines:

    - The answer NA means that there is no societal impact of the work performed.
    - If the authors answer NA or No, they should explain why their work has no societal impact or why the paper does not address societal impact.

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

11. **Safeguards**

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

    Answer: [No]

    Justification: We use publicly available open source models.

    Guidelines:

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

12. **Licenses for existing assets**

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

    Answer: [Yes]

    Justification: Citations and links to used assets are provided.

    Guidelines:

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

- For existing datasets that are re-packaged, both the original license and the license of the derived asset (if it has changed) should be provided.
- If this information is not available online, the authors are encouraged to reach out to the asset's creators.

13. **New assets**

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

    Answer: [NA]

    Guidelines:

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

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

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

    Answer: [NA]

    Guidelines:

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

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

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

    Answer: [NA]

    Guidelines:

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

16. **Declaration of LLM usage**

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

Answer: [No]

Guidelines:

- The answer NA means that the core method development in this research does not involve LLMs as any important, original, or non-standard components.
- Please refer to our LLM policy (`https://neurips.cc/Conferences/2025/LLM`) for what should or should not be described.

A Theorems and proofs

A.1 KL divergence error decomposition

We revisit the KL divergence introduced in Section 5. Recall we quantify the error from sampling $p_\phi(x) = \sum_z p_\phi(x, z)$ instead of $q(x)$ via KL divergence

$$D_{\mathrm{KL}}(q(x), p_\phi(x)) \leq D_{\mathrm{KL}}(q_\phi(x, z), p_\phi(x, z)) = \sum_{i=1}^{d} \mathbb{E}_{q_\phi}\left[\ln q(x^{z_i}|x^{z<i}) - \sum_{l\in z^i} \ln p^\theta(x^l|x^{z<i})\right], \tag{10}$$

where the inequality can be derived from the Evidence Lower BOund (ELBO) (Kingma and Welling, 2019). In the equality we plug in the definitions of the joint distributions $p_\phi(x, z), q_\phi(x, z)$ from Equation (5) and Equation (6) respectively, where the ϕ term cancels out, and the sum and expectation can be interchanged since the sum does not

We add and subtract $\ln\left(\prod_{i=1}^{d}\prod_{l\in z^i} q(x^l|x^{z<i})\right)$ from Equation (10):

$$\begin{aligned}
&\sum_{i=1}^{d} \mathbb{E}_{q_\phi}\left[\ln q(x^{z_i}|x^{z<i}) - \sum_{l\in z^i} \ln p^\theta(x^l|x^{z<i})\right] \\
&= \sum_{i=1}^{d} \mathbb{E}_{q_\phi}\left[\sum_{l\in z^i} \ln \frac{q(x^l|x^{z<i})}{p^\theta(x^l|x^{z<i})} + \ln \frac{q(x^{z_i}|x^{z<i})}{\prod_{l\in z^i} q(x^l|x^{z<i})}\right],
\end{aligned} \tag{11}$$

and will now separately simplify the two terms.

Joint dependence error. We begin with the right term of the expectation in last equality of Equation (11), which will turn out to be the joint dependence error term in Equation (7).

$$\begin{aligned}
&\mathbb{E}_{q_\phi}\left[\ln \frac{q(x^{z_i}|x^{z<i})}{\prod_{l\in z^i} q(x^l|x^{z<i})}\right] = \\
&= \sum_{x,z} q_\phi(x, z)\left(\ln \frac{q(x^{z_i}|x^{z<i})}{\prod_{l\in z^i} q(x^l|x^{z<i})}\right) = \\
&\overset{(*)}{=} \sum_{z_{\leq i}, x^{z_{\leq i}}} q_\phi(z_{\leq i}, x^{z_{\leq i}})\left(\ln \frac{q(x^{z_i}|x^{z<i})}{\prod_{l\in z^i} q(x^l|x^{z<i})}\right) = \\
&= \sum_{z_{\leq i}, x^{z<i}} q_\phi(z_{\leq i}, x^{z<i}) \sum_{x^{z_i}} q_\phi(x^{z_i}|x^{z<i}, z_{\leq i})\left(\ln \frac{q(x^{z_i}|x^{z<i})}{\prod_{l\in z^i} q(x^l|x^{z<i})}\right) = \\
&\overset{(**)}{=} \mathbb{E}_{q_\phi(z_{\leq i}, x^{z<i})}\left[\sum_{x^{z_i}} q(x^{z_i}|x^{z<i}) \ln \frac{q(x^{z_i}|x^{z<i})}{\prod_{l\in z^i} q(x^l|x^{z<i})}\right] = \\
&= \mathbb{E}_{q_\phi(z_{\leq i}, x^{z<i})}\left[D_{\mathrm{KL}}\left(q(x^{z_i}|x^{z<i}), \prod_{l\in z^i} q(x^l|x^{z<i}))\right)\right].
\end{aligned} \tag{12}$$

where in $(*)$ we marginalize over $x^{z>i}$ and $z_{>i}$ since the function the expectation is taken over does not depend on them. In $(**)$ we use the fact that for the sampling procedure defined by ϕ, $q_\phi(x^{z_i}|x^{z<i}, z_{\leq i}) = q(x^{z_i}|x^{z<i})$. At last, we arrive at an expectation of the KL-divergence between the joint and the factorized product distributions of x^{z_i} conditioned on all the unmasked tokens before, measuring the joint dependence withoun the subset of indices z_i.

Model error. As for the left term in the last equality of Equation (11), it will turn out to be the model error term in Equation (7). Recalling the left term:

$$
\begin{aligned}
&\mathbb{E}_{q_\phi}\left[\sum_{l\in z^i}\ln\frac{q(x^l|x^{z<i})}{p^\theta(x^l|x^{z<i})}\right] = \\
&= \sum_{x,z} q_\phi(x,z)\left(\sum_{l\in z^i}\ln\frac{q(x^l|x^{z<i})}{p^\theta(x^l|x^{z<i})}\right) = \\
&\overset{(*)}{=} \sum_{z_{\leq i},x^{z\leq i}} q_\phi(z_{\leq i},x^{z\leq i})\left(\sum_{l\in z^i}\ln\frac{q(x^l|x^{z<i})}{p^\theta(x^l|x^{z<i})}\right) = \\
&= \sum_{z_{\leq i},x^{z<i}} q_\phi(z_{\leq i},x^{z<i})\sum_{x^{z_i}} q_\phi(x^{z_i}|x^{z<i},z_{\leq i})\left(\sum_{l\in z^i}\ln\frac{q(x^l|x^{z<i})}{p^\theta(x^l|x^{z<i})}\right) = \\
&\overset{(**)}{=} \mathbb{E}_{q_\phi(z_{\leq i},x^{z<i})}\left[\sum_{l\in z^i}\sum_{x^{z_i}} q(x^{z_i}|x^{z<i})\ln\frac{q(x^l|x^{z<i})}{p^\theta(x^l|x^{z<i})}\right] = \\
&\overset{(***)}{=} \mathbb{E}_{q_\phi(z_{\leq i},x^{z<i})}\left[\sum_{l\in z^i}\sum_{x^{l}} q(x^{l}|x^{z<i})\ln\frac{q(x^l|x^{z<i})}{p^\theta(x^l|x^{z<i})}\right] = \\
&= \mathbb{E}_{q_\phi(z_{\leq i},x^{z<i})}\left[\sum_{l\in z^i} D_{\mathrm{KL}}(q(x^l|x^{z<i}),p^\theta(x^l|x^{z<i}))\right]. \qquad (13)
\end{aligned}
$$

where $(*)$ and $(**)$ are the same steps as in the joint dependence error derivation above. In $(***)$, we again marginalize over variable that do not appear in the expectation. At last, we arrive at a sum of of the KL-divergences between the factorized conditionals, which is exactly what p^θ is trained to learn, and this we call this term *model error*.

KL divergence equality: The two KL divergences, $D_{\mathrm{KL}}(q(x),p_\phi(x))$ and $D_{\mathrm{KL}}(q_\phi(x,z),p_\phi(x,z))$, are equal when $q_\phi(z|x) = p_\phi(z|x)$. This only occurs under special ϕ. For any ϕ,

$$
q_\phi(z|x) = \frac{q_\phi(x,z)}{q(x)} = \frac{q(x)\prod_{i=1}^d \phi(z^i|x^{z<i},z^{<i})}{q(x)} = \prod_{i=1}^d \phi(z^i|x^{z<i},z^{<i}). \qquad (14)
$$

Then

$$
\begin{aligned}
p_\phi(z|x) &= \frac{p_\phi(x,z)}{p_\phi(x)} \\
&= \frac{\prod_{i=1}^d \left(\prod_{l\in z^i} p^\theta(x^l|x^{z<i})\right)\phi(z^i|x^{z<i},z^{<i})}{p_\phi(x)} \\
&= \frac{\prod_{i=1}^d \left(\prod_{l\in z^i} p^\theta(x^l|x^{z<i})\right)}{p_\phi(x)} q_\phi(z|x). \qquad (15)
\end{aligned}
$$

Thus $p_\phi(z|x)$ is almost certainly not equal to $q_\phi(z|x)$ even if we unmask one token sequentially, because p^θ is learned. The product of the learned conditionals almost certainly results in a different joint distribution depending on the order, unlike for the data distribution where the product of the true conditionals is $q(x)$ for every order.

However, when every ϕ is deterministic we do have equality. This special case is relevant because many schemes introduced in the main text are deterministic, or nearly so. Then the partition z is entirely determined by x and can be written $z_\phi(x)$, and $q_\phi(z|x) = p_\phi(z|x)$ are a point mass on $z_\phi(x)$. For a deterministic ϕ optimizing the KL divergence hence directly optimizes the likelihood $\mathbb{E}_q[\ln p_\phi(x)]$, and not a lower-bound on that likelihood, because $\mathbb{E}_q[\ln p_\phi(x)] = \mathbb{E}_{q_\phi}[\ln p_\phi(x,z_\phi(x))]$ when z is deterministically generated.

B Algorithm

Algorithm 1 EB-Sampler with `generate_until` logic

Require: Factorized conditionals predictions $p^{\theta,l}(\cdot|x)$, threshold $\gamma \geq 0$, prompt y_0, sequence length d, error proxy functional E, entropy functional H, stopping criteria $C : \mathcal{S} \to \{\texttt{True}, \texttt{False}\}$, mask token $\texttt{m}$

$n = d - \text{len}(y_0)$

$x \leftarrow [y_0, \texttt{m} * n]$ ▷ Set initial condition

$\mathcal{I}_{\texttt{m}} = \{l | x^l = \texttt{m}\}$

while $\mathcal{I}_{\texttt{m}} \neq \emptyset$ and not $C(x)$ **do** ▷ Stop if all tokens unmasked

  $\hat{p}_l = p^{\theta,l}(\cdot|x)$, for $l \in \mathcal{I}_{\texttt{m}}$

  $\hat{e} = E(\hat{p})$

  $\hat{h} = H(\hat{p})$

  $I_{\text{sort}} = \texttt{argsort}(\hat{e}_{\mathcal{I}_{\texttt{m}}})$ ▷ Sort masked tokens by error

  $U \leftarrow \{\}$ ▷ Initialize subpartition

  for a in I_{sort} **do** ▷ Iterate over sorted masked tokens

    $U \leftarrow U \cup a$

    if $\texttt{sum}(\hat{h}_U) - \texttt{max}(\hat{h}_U) \leq \gamma$ **then** ▷ Compute entropy bound equation 2

      $x^a = \texttt{sample}(\hat{p}_a)$ ▷ Sample unmasked value from posterior if below threshold

    else

      break ▷ Halt unmasking for loop if entropy bound is exceeded

  $\mathcal{I}_{\texttt{m}} = \{l | x^l = \texttt{m}\}$

return x

C Experimental details

C.1 Code and math reasoning

C.1.1 Datasets

For code generation tasks we evaluate EB-Sampler on 0-shot HumanEval Chen et al. (2021), 4-shot MBPP Austin et al. (2021b), and for math reasoning we test 8-shot GSM8K Cobbe et al. (2021) without Chain Of Thought (COT) variant, and 4-shot Math Hendrycks et al. (2021) corresponding to the `hendrycks_math` variant.

C.1.2 Setup

We evaluate the efficiency gains of EB-Sampler on two recent state of the art MDMs: LLaDa 8B Nie et al. (2025b) and Dream 7B Ye et al. (2025). For LLaDa 8B the maximal sequence length is 4096 and for Dream 7B, 2048. That is, the input to the models in the beginning of generation, is a padded sequence, starting with the prompt given in each task and then padded with the mask token, `m`, to the maximal sequence length, denoted `max_seq_len`. For each dataset a predetermined generation length is set, denoted `max_gen_len`. We enforce unmasking tokens that are in the range $[\texttt{len(prompt)}, \texttt{len(prompt)} + \texttt{max\_gen\_len}]$. In the rare case when $\texttt{len(prompt)} + \texttt{max\_gen\_len} > \texttt{max\_seq\_len}$ the prompt is truncated from the left.

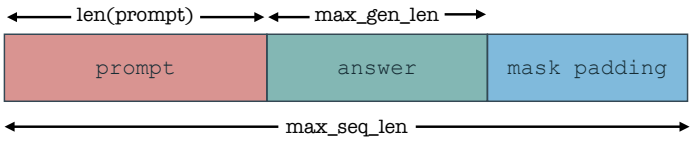


Figure 8: Input sequence visualization.

Table 2: Evaluation parameters for code and math reasoning.

	Dataset size	#-shots	max_gen_len	generate_until phrase	
				LLaDa 8B	Dream 7B
HumanEval	164	0	512	[”<|endoftext|>”,”\n\n\n”]	[”<|endoftext|>”, ”““\n”]
MBPP	500	4	512	”[DONE]”	
GSM8K	1320	8	256	”The answer is %d.”	
MATH	5000	4	512	”I hope it is correct.”	

All benchmarks were run with the same γ range of values for EB-Sampler, $\gamma \in \{0, 0.001, 0.01, 0.1, 0.25, 0.5, 0.75, 1.0, 1.5\}$, and with same k range of values for Top-k sampling, $k \in \{1, 2, 4, 8\}$. All benchmarks were run in the same computational setting, on $8\times$H100. We report runtimes for confidence and entropy error proxies in Table 3. Runtimes with margin error proxy are longer due to the need to sort over the vocabulary size to compute the top-2 tokens. Runtimes for LLaDa 8B are longer than Dream 7B due to having twice the maximal sequence length of Dream.

Table 3: Average runtimes on $8\times$H100 of code and math benchmarks evaluation for 1 token per step sampling (Top_1) with entropy and confidence error proxies. Relative standard deviation of measurements is 1%.

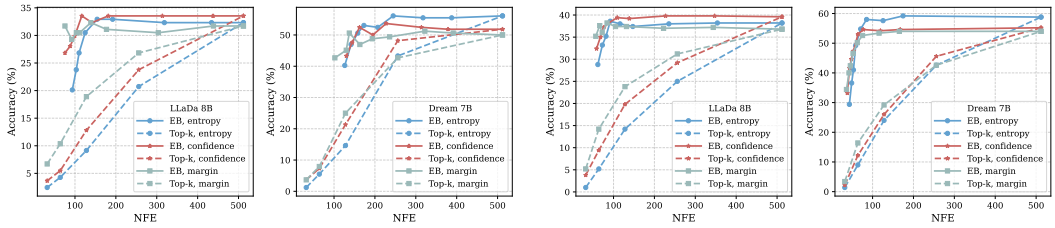
	Runtimes (hrs.)	
	LLaDa 8B	Dream 7B
HumanEval	0.58	0.26
MBPP	1.75	0.79
GSM8K	2.29	1.03
MATH	17.30	7.84

C.1.3 Post-process

Accuracy evaluation. Model outputs for all datasets evaluated with both LLaDa 8B and Dream 7B had been directly fed into the standard evaluation scripts, except for HumanEval with Dream 7B. Evaluating the raw output of Dream 7B on the HumanEval benchmark results in around 8% drop in performance compared to reported results by the authors. Investigating the cause for the drop in performance led to the observation that Dream 7B sometimes produces answers with a template that places the generated code inside code blocks which get ignored when compiled and are therefore considered as failure at the task. We thus post-process Dream's raw outputs on HumanEval to extract the function implementation, closing the gap to reported results to $< 2\%$. Importantly, we emphasize that all comparisons in our paper are between sampling procedures from the same model and same evaluation.

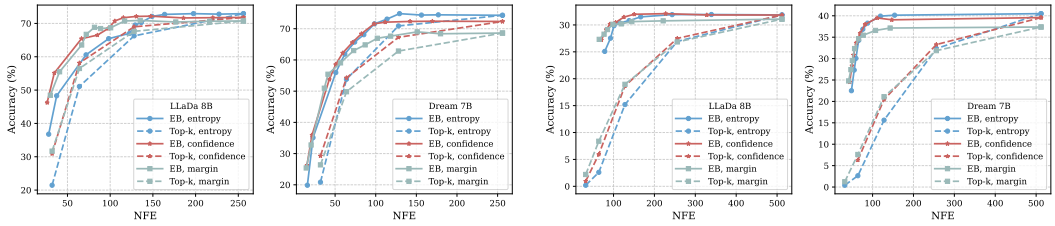
Efficiency measurement. As noted in Section 6.1.1, standard sampling procedures from MDM generate a sequence with predetermined length, `max_gen_len`. In most cases, the answer to the given prompt will be shorter, denoted `answer_len`, and there are (`max_gen_len` − `answer_len`) generated tokens that are being truncated, hence unused, during evaluation. To isolate the efficiency gain EB-Sampler provides in generating the answer tokens from the gain in generating the rest of the tokens that are later not used, we incorporate the `generate_until` logic as a post-process. We note that this logic can also be integrated in the sampling procedure itself, saving up computation, without changing the performance of the model, as the logic ensures termination of generation only once the stopping criterion has been met. The phrases used as markers for the `generate_until` post-process logic for each dataset and each model are listed in Table 2.

D Additional experiments - code and math reasoning



(a) HumanEval (b) MBPP

Figure 9: pass@1 accuracy vs. full `max_gen_len` NFE on code reasoning tasks.

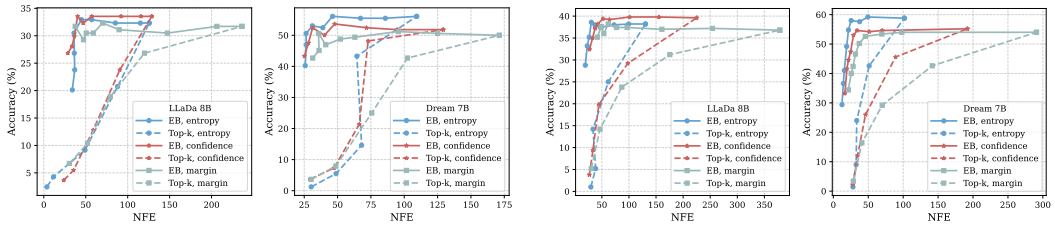


(a) GSM8K (b) MATH

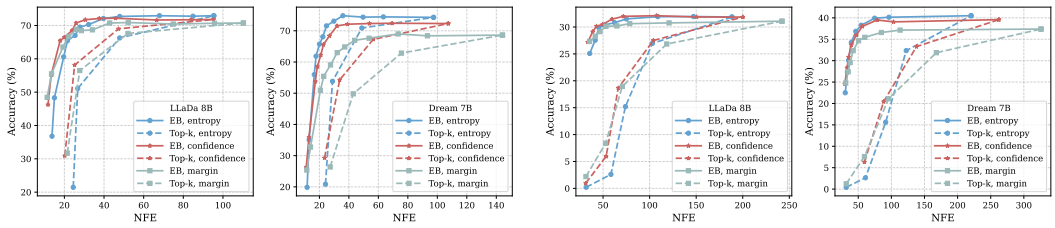
Figure 10: pass@1 accuracy vs. full `max_gen_len` NFE on math reasoning tasks.

D.1 Results with margin error proxy

In Figure 11 we show the results with the margin error proxy along with the results with confidence and entropy error proxies presented in the main body of the paper in Figure 5. We observed that confidence error proxy was typically the best for the LLaDa 8B model and entropy error proxy worked best in most cases for Dream 7B. The margin error proxy mostly yielded inferior accuracy in full NFE, thus to maintain readability of the plots we did not include it in the main body of the paper.



(a) HumanEval (b) MBPP



(c) GSM8K (d) MATH

Figure 11: pass@1 accuracy vs. NFE with `generate_until` logic on code and math reasoning tasks.

D.2 Measuring efficiency of MDMs

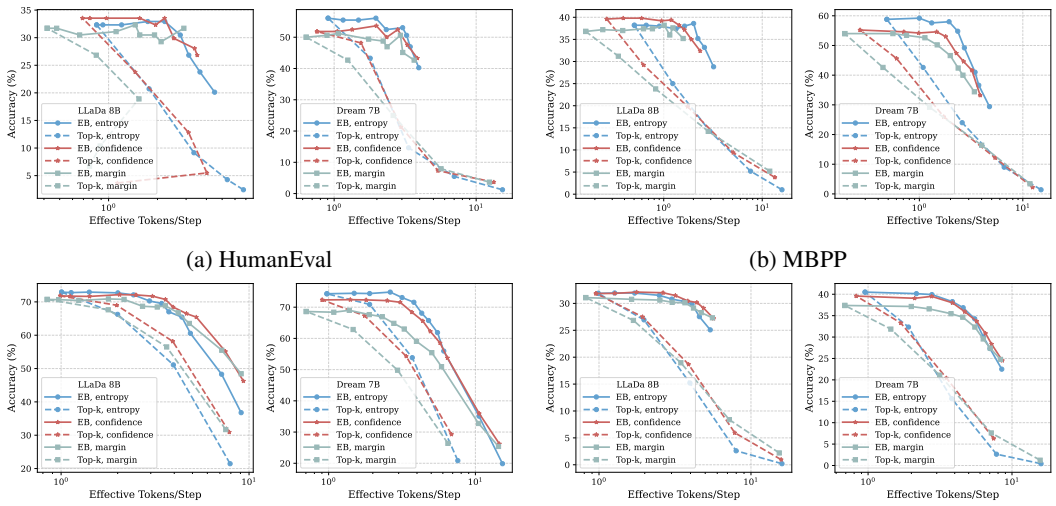
In Section 6.1.1 we outline the different ways to measure sampling efficiency and discuss the problems with some of the approaches. We explain why measuring efficiency against full sequence length generation results in apparent high gains (like in Figures 9 and 10), and then propose two ways to better approximate the gains provided by different samplers:

- Unmask with `generate_until` logic
- Unmask semi-auto-regressively with `generate_until` logic

Effective tokens/step with `generate_until` logic. In Figure 12 we show accuracy against the effective generation speed of the model quantified via:

$$\text{Effective Tokens/Step} = \frac{\texttt{mean\_answer\_len}}{\texttt{mean\_NFE\_to\_condition}}, \tag{16}$$

where `mean_answer_len` is the average number of tokens from left to right until the `generate_until` answer markers, and `mean_NFE_to_condition` is the number of function evaluations required by the model to generate the answer until both `generate_until` answer markers apear in answer and all tokens before the marker are unmasked. Figure 12 shows that in most cases the effective speed at $\gamma = 0$ or Top_1, is actually less than 1, meaning that the model unmasks tokens that are not used in the final answer, or equivalently unmasks tokens that come after the stopping phrase in the sequence. This observation led us to explore an approach to mitigates this inefficiency via semi-autoregressive generation (Arriola et al., 2025; Nie et al., 2025b).



(a) HumanEval (b) MBPP

(c) GSM8K (d) MATH

Figure 12: pass@1 accuracy vs. tokens/step on code and math reasoning tasks.

Semi-autoregressive generation. Figure 12 supports our claim that mostly in the MBPP benchmark many tokens are generated after the stopping phrase although not all tokens have been unmasked before. In the main body of the paper, in Table 1, we showed an ablation with semi-autoregressive generation for that benchmark with the Dream 7B model. In Table 4 we also add the same ablation with the LLaDa 8B model. We report the same ablation for the GSM8K benchmark in Tables 5 and 6, which show that the gap in efficiency between with and without semi-autoregressive generation is small to non existing aligning with Figure 12 that shows effective token/step of around 1 on this benchmark. That is, on GSM8K it is less likely that the model generates tokens that come after the stopping phrase.

Table 4: NFE and Speed-Ups for LLaDa 8B on the MBPP for various evaluation schemes at roughly same best pass@1. For all configurations in the table the mean answer length is ~ 60 tokens.

	Full `max_gen_len` $= 512$			`generate_until` logic		`generate_until` logic + semi-AR (block_len=64)		
	pass@1	NFE	Speed-Up	NFE	Speed-Up	pass@1	NFE	Speed-Up
Top-1	39.6%	512	x 1	224.93	x 1	39.4%	73.31	x 1
EB, confidence, $\gamma = 0.001$	39.8%	347.28	x 1.47	155.31	x 1.44	39.4%	60.83	x 1.21
EB, confidence, $\gamma = 0.1$	39.2%	138.41	x 3.67	64.01	x 3.51	38.8%	33.20	x 2.21

Table 5: NFE and Speed-Ups for Dream 7B on the GSM8K for various evaluation schemes at roughly same best pass@1. For all configurations in the table the mean answer length is ~ 93 tokens.

	Full `max_gen_len`			`generate_until` logic		`generate_until` logic + semi-AR (block_len=64)		
	pass@1	NFE	Speed-Up	NFE	Speed-Up	pass@1	NFE	Speed-Up
Top-1	74.30%	256	x 1	97.44	x 1	74.90%	95.23	x 1
EB, entropy, $\gamma = 0.01$	74.37%	156.29	x 1.64	49.60	x 1.96	75.36%	48.29	x 1.97
EB, entropy, $\gamma = 0.1$	74.83%	129.27	x 1.98	35.94	x 2.71	75.36%	35.60	x 2.66

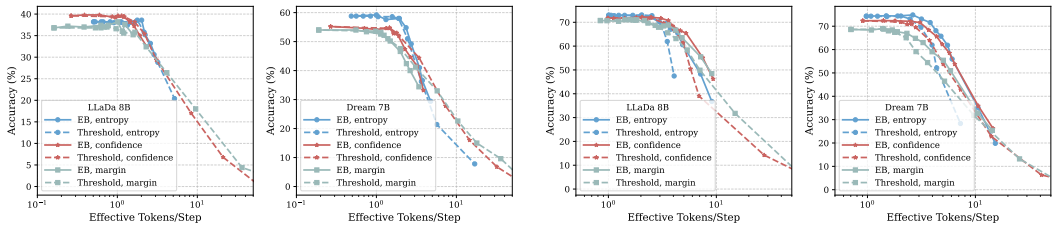
Table 6: NFE and Speed-Ups for LLaDa 8B on the GSM8K for various evaluation schemes at roughly same best pass@1. For all configurations in the table the mean answer length is ~ 93 tokens.

	Full `max_gen_len`			`generate_until` logic		`generate_until` logic + semi-AR (block_len=64)		
	pass@1	NFE	Speed-Up	NFE	Speed-Up	pass@1	NFE	Speed-Up
Top-1	71.79%	256	x 1	95.19	x 1	71.95%	93.62	x 1
EB, confidence, $\gamma = 0.01$	71.64%	185.78	x 1.36	66.57	x 1.43	72.33%	65.42	x 1.43
EB, confidence, $\gamma = 0.1$	72.17%	147.48	x 1.74	45.83	x 2.08	72.71%	45.30	x 2.07

D.3 Comparison to naive thresholding

Our key observation in this work is that adaptively choosing the number of tokens to unmask in each step can result in a better accuracy-efficiency tradeoff. In this section we compare to naive thresholding approaches that dynamically determine the number of tokens to unmask with a per-token thresholding criterion as opposed to the EB-Sampler which accumulates the sum of entropies.

We test the EB-Sampler against three threshold criteria: entropy, confidence and margin. In fig. 13 we show accuracy vs. Effective Tokens/Step 16. For MBPP we see a very similar profile for EB-Sampler and naive thresholding while for GSM8K, EB-Sampler shows a favorable accuracy-efficiency tradeoff.



(a) MBPP (b) GSM8K

Figure 13: Comparison between EB-Sampler and naive thresholding. pass@1 accuracy vs. tokens/step on code and math reasoning tasks.

These results indicate that for some settings, like in MBPP, the model predictions that yield high accuracy are highly confident or equivalently have very low entropy, thus naive thresholding aligns with EB-Sampler accuracy-efficiency profile. On the contrary, for GSM8K, we do observe an non-negligible gap that could be explained by our information theoretic interpretation of the unmasking process.

E Societal Impact

As this work introduces a more efficient sampling procedure from existing models, it does not introduce significant societal risks beyond those that already exist.