UNDERSTANDING AND ACCELERATING THE TRAINING OF MASKED DIFFUSION LANGUAGE MODELS

Anonymous authors
Paper under double-blind review

ABSTRACT

Masked diffusion models (MDMs) have emerged as a promising alternative to autoregressive models (ARMs) for language modeling. However, MDMs are known to learn substantially more slowly than ARMs, which may become problematic when scaling MDMs to larger models. Therefore, we ask the following question: how can we accelerate standard MDM training while maintaining its final performance? To this end, we first provide a detailed analysis of why MDM training is slow. We find that the main factor is the locality bias of language: the predictive information for a token is concentrated in nearby positions. We further investigate how this bias slows learning and suggest a simple yet effective remedy: bell-shaped time sampling as a training strategy. Notably, MDMs trained with our training recipe reach the same validation negative log-likelihood (NLL) up to $\sim 4\times$ faster than standard training on One Billion Word Benchmark (LM1B). We also show faster improvements in generative perplexity, zero-shot perplexity, and downstream task performance on various benchmarks.

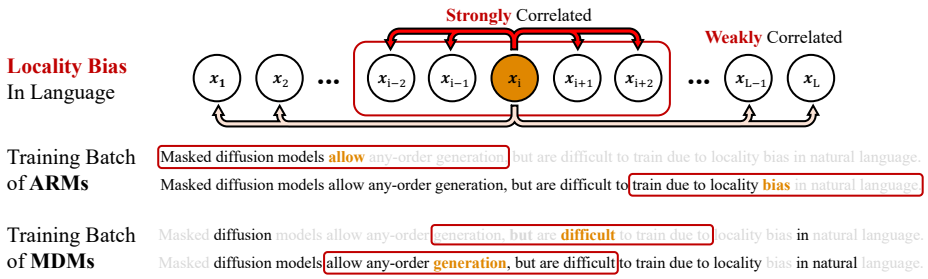


Figure 1: *Locality bias* in language makes MDMs slow learners. The red box illustrates the concept of locality bias: a token is influenced more strongly by nearby tokens and more weakly by distant ones. During training, ARMs learn to predict the target token from left-filled sequences, where an appropriate amount of local information is always available; in contrast, MDMs learn to predict the target token from arbitrary masked contexts, which may include cases where local information is either extremely scarce or extremely abundant, as highlighted by the red boxes. Our analysis suggests that these two regimes may be among the critical factors underlying the slow training of MDMs.

1 INTRODUCTION

Recently, discrete diffusion models (Lou et al., 2024; Austin et al., 2021) have rapidly emerged as a compelling framework for language modeling. They can be broadly divided into uniform discrete diffusion models (Schiff et al., 2025) and masked diffusion models (MDMs) (Shi et al., 2024), with various frameworks (Xie et al., 2026; Hong et al., 2026b) further developed from these two families. Among this wide range of studies, *the most basic MDM framework* (Ou et al., 2025; Sahoo et al., 2024) *remains the most widely adopted* in real-world applications precisely because of its simplicity. Large-scale models, such as LLaDA (Nie et al., 2025b), Seed Diffusion (Song et al., 2025), and Dream (Ye et al., 2025), ultimately follow the most standard generation scheme, in which model predictions are used to gradually convert a fully masked sentence into tokens. Various inference-time techniques (Hayakawa et al., 2026), reinforcement learning methods (Zhao et al., 2025), and even retrieval-augmented generation (Yu et al., 2026) are built upon this basic MDM framework.

Reviewers: please read the Reviewer Guidelines (iclr.cc/Conferences/2027/ReviewerGuidelines) and the AI Policy for Reviewers (iclr.cc/Conferences/2027/AIPolicyForReviewers). If you used AI to expand, edit, or polish your review, please provide the input text to the LLM. Better still, consider skipping the LLM and submitting your original text: we, and the authors, are much more interested in your unedited thoughts than in what an LLM has to say. AI-assisted or not, you are putting your name and reputation behind your review: LLM-generated falsehoods, hallucinations or misrepresentations are subject to disciplinary action, which may include desk-rejecting all papers you have authored.

Recent evidence suggests MDMs can become competitive with autoregressive models (ARMs). For general-purpose models, MDMs such as LLaDA (Nie et al., 2025b) and Dream (Ye et al., 2025) have surpassed same-sized ARM counterparts such as LLaMA 3 8B (Grattafiori et al., 2024), while for coding models, DiffuCoder (Gong et al., 2026) and Dream-Coder (Xie et al., 2025) have matched same-sized Qwen2.5-Coder (Hui et al., 2024). Prior work (Prabhudesai et al., 2025) further suggests that given sufficiently long training, MDMs can eventually match the performance of ARMs.

However, the main bottleneck of MDMs is that they learn much more slowly than ARMs. Specifically, MDMs are known to require$\sim 16\times$ training compute to reach the same validation loss of ARMs (Prabhudesai et al., 2025; Sahoo et al., 2026). This becomes a major obstacle when scaling MDMs to real-world LLMs, since the training cost of LLMs is extremely high; for example, even ARM-based LLaMA 3.1 405B spent ∼30M H100 GPU hours for training. Differences in training efficiency can translate into enormous differences in both cost and time. In this context, we raise a key question that has been overlooked in prior works:

How can we accelerate standard MDM training while maintaining its final performance?

To this end, we first **analyze why MDM training is slow.** Stepping back from MDM, we begin by separating two possible causes of slow any-order learning. The first is the locality bias of language: each token in natural language is influenced much more strongly by nearby tokens than by distant ones (Futrell, 2019). Since any-order generation includes orders beyond left-to-right (L2R), locality bias can shape its training dynamics differently. The second is the large order-space complexity of the any-order generation problem: unlike a fixed L2R order, models must learn a vastly larger family of generation problems. We show that the former factor, locality bias, has a substantially greater impact on training speed. Furthermore, we analyze why the locality bias slows MDM training, and find that both high-context (almost no masked) and low-context (almost fully masked) samples waste training budget. Although L2R bias, closely related to locality bias, has been pointed out as a challenge in many prior works (Kim et al., 2025; Xue et al., 2025), there has been little detailed analysis of *how* it slows down training.

Our second contribution is to introduce a **simple, effective, and generalizable technique for accelerating MDM training**. Motivated by our analysis of why MDMs learn slowly, we propose *bell-shaped time sampling*, which increases the probability of sampling t from the middle-context region, and show that this significantly speeds up MDM training. Notably, bell-shaped time sampling achieves up to $\sim \mathbf{4}\times$ **faster training** in reaching the same NLL on LM1B (Chelba et al., 2014). We show that it also leads to faster improvements in generative perplexity (PPL), and zero-shot PPL for models trained on OpenWebText (Gokaslan et al., 2019). Finally, we show that our method remains effective even in the continual pretraining with ARM initialization (Gong et al., 2025) setting commonly adopted for large-scale MDMs (Bie et al., 2025), and can further improve downstream task performance.

2 BACKGROUND

To analyze why MDMs are slow learners, we compare various classes of language models. We briefly explain those frameworks and objectives. Refer to Appendix A for details and related works.

Notation. Let $\mathbf{x}$ be a categorical sample with L dimensions and $x^i \in \{1, \dots, V\}$ be the i-th dimension of $\mathbf{x}$. We define special mask state as $m = V+1$. Let $\mathbf{e}$ denote the standard basis of $\mathbb{R}^{V+1}$. Thus, each token is represented by a one-hot vector $\mathbf{e}_i$ in $\mathbb{R}^{V+1}$. Let $\mathrm{Cat}(\boldsymbol{\pi})$ denote the categorical distribution over $V+1$ classes with $\boldsymbol{\pi} \in \Delta^{V+1}$, the $(V+1)$-simplex. We denote a noised sequence at time t by $\mathbf{x}_t$. We further define a mask indicator $\mathbf{M} \in \{0,1\}^L$ and the masked sequence $\mathbf{x}[\mathbf{M}]$ that is obtained by replacing x^i with m whenever $M^i = 1$, while leaving it unchanged whenever $M^i = 0$. Let $\mathbf{M}_n$ denote a mask indicator with mask number of n and $\mathcal{M}_n$ as set of all possible combinatorial $\mathbf{M}_n$. For ARM, let $\mathbf{x}^{<i} := (x^1, \dots, x^{i-1})$. We write uniform distribution as $\mathrm{U}(\cdot)$.

Masked diffusion models. MDM (Sahoo et al., 2024; Shi et al., 2024) defines the forward corruption process using the absorbing mask strategy: once a token is masked, it remains masked throughout the remaining process. For the diffusion process, define the time interval as $t \in \mathcal{T} = [0,1]$ where we corrupt the data from $t = 0$ (original data) to $t = 1$ (completely masked). Formally, the forward process at time t is defined as $q_{t|0}(x_t^i \mid \mathbf{x}) = \mathrm{Cat}(\cdot; \alpha_t \mathbf{e}_{x^i} + (1-\alpha_t)\mathbf{e}_m)$, where the forward process

gradually adds noise as t grows. In this regard, the noise scheduler is typically set to $\alpha_t = 1 - t$. To mimic the true posterior induced from the forward process, the parametrized reverse process is designed as $p_\theta(\mathbf{x}_s^i \mid \mathbf{x}_t) = \mathbf{1}(x_t^i \neq m) \cdot \mathrm{Cat}(\mathbf{e}_{x_t^i}) + \mathbf{1}(x_t^i = m) \cdot \mathrm{Cat}(\frac{1-\alpha_s}{1-\alpha_t}\mathbf{e}_m + \frac{\alpha_s - \alpha_t}{1-\alpha_t}\mu_\theta^i(\mathbf{x}_t, t))$ where $s < t$ and $\mathbf{1}(\cdot)$ denotes indicator function. The diffusion network $\mu_\theta^i(\mathbf{x}_t, t) \in \Delta^{V+1}$ here predicts the clean token x^i over the vocabulary. The reverse process is defined as a token-wise conditionally independent distribution, *i.e.*, $p_\theta(\mathbf{x}_s \mid \mathbf{x}_t) = \prod_{i=1}^L p_\theta(\mathbf{x}_s^i \mid \mathbf{x}_t)$. The NELBO is then:

$$\mathcal{L}_{\mathrm{MDM}} = \int_0^1 \frac{-\alpha_t'}{1-\alpha_t} \mathbb{E}_{\mathbf{x}\sim p_{\mathrm{data}}, \mathbf{x}_t \sim q_{t|0}} \left[\sum_{i:x_t^i = m} - \log p_\theta(x^i | \mathbf{x}_t, t) \right] dt. \tag{1}$$

Autoregressive models. A standard ARM factorizes the joint distribution in the L2R order:

$$p_\theta^{\mathrm{ARM}}(\mathbf{x}) = \prod_{i=1}^L p_\theta^{\mathrm{ARM}}\left(x^i \mid \mathbf{x}^{<i}\right), \quad \mathcal{L}_{\mathrm{ARM}} = \mathbb{E}_{\mathbf{x}\sim p_{\mathrm{data}}}[\textstyle\sum_{i=1}^L - \log p_\theta^{\mathrm{ARM}}(x^i | \mathbf{x}^{<i})]. \tag{2}$$

σ-ARMs and any-order ARMs. Typical ARMs factorize the joint distribution in the fixed L2R order, yet we consider a fixed-permutation ARM, where a single random permutation σ is sampled once and kept fixed throughout training. For brevity, we refer to this variant as σ-ARM. Formally, fix any random permutation $\sigma \in S_L$ where S_L denotes the set of all possible permutations of $[L]$. σ-ARMs then factorize joint probability as $p_\theta^{\sigma\text{-ARM}}(\mathbf{x}) = \prod_{r=1}^L p_\theta^{\sigma\text{-ARM}}(x^{\sigma_r} \mid \mathbf{x}^{<\sigma_r})$. Besides, Any-order ARMs (AoARMs) (Xue et al., 2025) factorize joint probability with *variable (not fixed)* random orders as $p_\theta^{\mathrm{AoARM}}(\mathbf{x}) = \mathbb{E}_{\sigma\sim U(S_L)} \prod_{r=1}^L p_\theta^{\mathrm{AoARM}}(x^{\sigma_r} \mid \mathbf{x}^{<\sigma_r}, \sigma)$. The objective of AoARM is then:

$$\mathcal{L}_{\mathrm{AoARM}} = \mathbb{E}_{\mathbf{x}\sim p_{\mathrm{data}}, \sigma\sim U(S_L)} \left[\textstyle\sum_{r=1}^L - \log p_\theta^{\mathrm{AoARM}}(x^{\sigma_r} \mid \mathbf{x}^{<\sigma_r}, \sigma) \right]. \tag{3}$$

Both MDM and AoARM share the same philosophy of performing *any-order generation*. This connection also extends to the NELBO (Kim et al., 2025): if the MDM model is time-agnostic and the AoARM model is σ-agnostic, then $\mathcal{L}_{\mathrm{AoARM}}$ (Eq. (3)) is theoretically equivalent to $\mathcal{L}_{\mathrm{MDM}}$ (Eq. (1)). The practical difference is that AoARM utilizes causal attention such that the model is aware of its currently generated order, while MDM utilizes full attention, which is agnostic of generated order.

2.1 Shared Format of Learning Objectives Across Various Frameworks

Although MDM, ARM, σ-ARM, and AoARM arise from different frameworks, all of these objectives can be viewed through the same lens: first choose the number of context (*i.e.*, observed) tokens, then construct a partially observed context, and finally predict one target token from that context.

Formally, let $N_c \in \{0, \ldots, L-1\}$ denote the number of context tokens, let $\mathbf{z}$ denote the resulting partially observed context, and let $p_{\mathcal{F}}(\mathbf{z} \mid \mathbf{x}, N_c)$ and $p_{\mathcal{F}}(i \mid \mathbf{x}, \mathbf{z}_{N_c})$ denote the framework-specific distributions for constructing the context and selecting the target position, respectively. Then, for each $\mathcal{F} \in \{\mathrm{MDM}, \mathrm{ARM}, \sigma\text{-ARM}, \mathrm{AoARM}\}$, the loss can be written in the shared form:

$$\mathcal{L}_{\mathcal{F}} = \textstyle\sum_{N_c=0}^{L-1} \mathcal{L}_{\mathcal{F}}^{N_c}, \quad \mathcal{L}_{\mathcal{F}}^{N_c} := \mathbb{E}_{\mathbf{x}\sim p_{\mathrm{data}}, \mathbf{z}_{N_c}\sim p_{\mathcal{F}}(\mathbf{z}|\mathbf{x}, N_c), i\sim p_{\mathcal{F}}(i|\mathbf{x}, \mathbf{z}_{N_c})}[-\log p_\theta(x^i \mid \mathbf{z}_{N_c})]. \tag{4}$$

Refer to Appendix A.2 for a full explanation. Therefore, the essential distinction between these frameworks does not lie in the token-level cross-entropy itself, but in how they define the distribution over contexts and target positions. For example, in ARM, $\mathbf{z}_{N_c}$ is the prefix $\mathbf{x}^{<N_c+1}$ and i is deterministically the next position, $N_c + 1$. In MDM, $\mathbf{z}_{N_c}$ is a masked sequence with N_c context tokens ($L - N_c$ masks), while i is sampled from $U(i : z_{N_c}^i = m)$. For simplicity, we omit the framework-specific sampling distributions in $\mathcal{L}_{\mathcal{F}}^{N_c}$ and write $\mathbb{E}_{\mathbf{x}, \mathbf{z}_{N_c}, i}[-\log p_\theta(x^i \mid \mathbf{z}_{N_c})]$.

3 Why Masked Diffusion Models Learn Slowly

We now analyze why MDMs are slow learners. While many prior works focus on the L2R bias of language, we instead consider the broader notion of *locality bias* (Futrell, 2019): in natural language, each token is influenced much more strongly by nearby tokens than by distant ones. We show here that this locality bias is indeed a critical factor in the slow training of MDMs. Furthermore, we show that low- and high-context samples impair MDM training for different reasons, both arising from locality bias. All experimental details for analysis in this section are in Appendix C.1.

3.1 Why Learning Any-order Language Generation is Hard

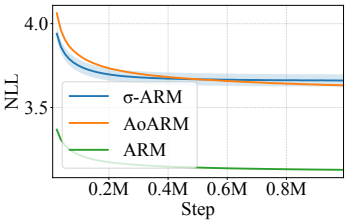


Figure 2: Training curve on LM1B. Blue region for σ-ARM is 1-sigma band for 5 distinct σ, and the solid line is the mean.

Before analyzing why MDMs train slowly, we first ask a broader question: why is learning any-order language generation difficult in the first place? Among several possible factors, we focus on two main possibilities: whether the difficulty primarily arises from the large order-space complexity of any-order generation itself, or from the locality bias of language. Since MDMs differ fundamentally from ARMs in architecture, we isolate this question here by comparing ARM, σ-ARM, and AoARM instead: ARM follows the L2R order and is therefore relatively aligned with the locality structure of language. In contrast, σ-ARM uses a single fixed random order, which removes the large order-space complexity of any-order generation but still forces the model to predict tokens from contexts that are often misaligned with local linguistic dependencies. AoARM further exposes the model to unfixed random orders, and therefore combines locality mismatch with the full order-space complexity of any-order generation.

We compare ARM, σ-ARM and AoARM on LM1B (Chelba et al., 2014) trained for 1M steps. For σ-ARM, we fix a single random permutation throughout training, while AoARM is trained with *non-fixed* random permutations and is therefore exposed to a much larger family of generation orders. The results are shown in Fig. 2. If the primary difficulty of any-order generation came mainly from the order-space complexity, then σ-ARM should be substantially easier to optimize than AoARM. However, while AoARM appears slightly slower in the early stage of training, it eventually matches σ-ARM, suggesting that order-space complexity might not be the main problem. Meanwhile, ARM *substantially outperforms* both models. These results lead to the following conclusion:

Finding 1. Locality bias appears to be an important factor in the difficulty of learning any-order generation, beyond the large order-space complexity of the generation problem.

3.2 Why Locality Bias in the Low-Context Region Makes MDM a Slow Learner

We next analyze why MDMs are slow learners, and why locality bias makes MDMs particularly difficult to optimize. A natural intuition is that locality bias slows down MDM training because MDMs inherently include the task of predicting targets in the low-context regime, where little local information is available. More specifically, one might expect training to be difficult because $\mathcal{L}_{\text{MDM}}^{N_c}$ is hard to reduce when N_c is small. However, we find the opposite. The low-context region is, in fact easy to learn; rather, the problem is that *MDMs waste training budget on easy low-context regions*.

A common misconception is that a task with high loss means the task is difficult to train. However, the problem should be separated: (1) what is the **optimal lower bound of the** $\mathcal{L}_{\mathcal{F}}^{N_c}$, and (2) what is **the gap between observed loss and optimal loss**. We suggest that the **difficulty of a task** is determined by the latter, not the former. With this distinction in mind, we further provide the following statement:

Proposition 3.1. *With infinite model capacity, the optimal loss for any framework* $\mathcal{F} \in \{\text{ARM}, \text{MDM}, \sigma\text{−ARM}, \text{AoARM}\}$ *is equal to the entropy of the data distribution:*

$$\min_{\theta} \mathcal{L}_{\mathcal{F}} = H(X), \quad H(X) := -\mathbb{E}_{\mathbf{x}\sim p_{\text{data}}}\big[\log p_{\text{data}}(\mathbf{x})\big]. \tag{5}$$

Furthermore, letting I denote the random target index, $\mathcal{L}_{\mathcal{F}}^{N_c}$ can be decomposed as

$$\mathcal{L}_{\mathcal{F}}^{N_c}(\theta) = H_{\mathcal{F}}(X^I \mid Z_{N_c}, I) + \mathbb{E}_{Z_{N_c},I}\text{KL}(p_{\text{data},\mathcal{F}}(\cdot \mid Z_{N_c}, I) \,\|\, p_\theta(\cdot \mid Z_{N_c}, I)).$$

where the conditional entropy is induced by p_{data}.

See Appendix B.1 for the full proof. From the proposition, *ARM and MDM share the same optimal loss*. What differs, however, is how this optimal loss is distributed across the number of context tokens. Depending on the language generation order, the lower bound of $\mathcal{L}_{\mathcal{F}}^{N_c}$, namely the **conditional entropy of data**, varies with N_c, while only the lower bound of their sum remains fixed.

Which region is hard to train? A high conditional loss at small N_c does not by itself imply that the region is hard to learn; rather, it means that the **irreducible conditional entropy of data** is higher. Intuitively, the low-context region lacks visible context, and therefore there is *no deterministic*

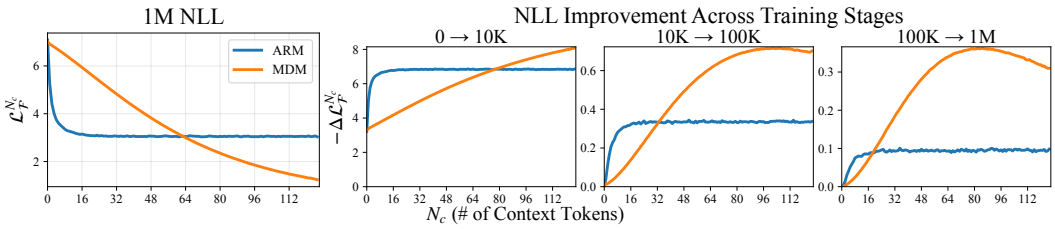


Figure 3: *Left:* NLL per number of context tokens on LM1B after 1M training steps. ARM shows an approximately uniform distribution, whereas MDM shows a clearly uneven one. *Right:* NLL drop across context token numbers for ARM and MDM over different training stages on LM1B. **For $N_c = 0$, MDM has already converged to the optimal loss by 10K steps**. Since MDM models the joint distribution as a product of independent conditional distributions, the optimal prediction for a fully masked sentence reduces to the dataset vocabulary distribution. Here, the NLL of the vocabulary distribution exactly matches the NLL of the 10K-step MDM on fully masked sentences, namely 7.02.

answer that would make the **conditional entropy** low. The **hard-to-train** region is instead the region where the **difference between the true data and the model distribution** remains high. Even if the observed loss is smaller than in other regions, if there remains a large gap between the **conditional entropy** and the observed loss, then the region should be regarded as hard to learn.

Empirical observations. We first identify where the **lower bound of $\mathcal{L}_{\mathcal{F}}^{N_c}$ is high**. In Fig. 3 (*left*), we plot $\mathcal{L}_{\mathcal{F}}^{N_c}$ using ARM and MDM trained on LM1B for 1M steps. Although these models are not fully optimal, the NLL improvement from 100K to 1M steps is small relative to the 1M-step NLL itself, suggesting that the 1M-step curves capture the overall shape of the lower bound. ARM exhibits an approximately uniform curve, whereas MDM shows a highly uneven one: its loss is much larger in the low-context region and decreases as N_c increases. We attribute this difference to locality bias. As illustrated in Fig. 1, ARM always predicts under the same asymmetric context structure, where the left side is observed and the right side is unobserved. In contrast, MDM changes the local context structure with N_c: in the low-context region, nearby tokens on both sides are mostly unobserved, while in the high-context region, they are mostly observed. Thus, the lower bound of $\mathcal{L}_{\mathrm{ARM}}^{N_c}$ remains nearly flat, whereas that of $\mathcal{L}_{\mathrm{MDM}}^{N_c}$ becomes strongly skewed toward small N_c.

We now identify whether **the low-context region is difficult to learn or not**. The extreme case $N_c = 0$ gives the most reliable and interpretable results: as detailed in the caption of Fig. 3, MDM has already reached the optimal fully masked prediction by 10K steps. Thus, the remaining high loss at $N_c = 0$ is not reducible model error, but irreducible conditional entropy. More broadly, under locality bias, low-context samples often provide little nearby information about the target token, and thus are expected to contain only limited learnable signal beyond this irreducible component. Consistent with this view, Fig. 3 (*right*) shows that low-context samples yield almost no further NLL improvement from 10K to 1M steps, while most of the remaining learning occurs in regions with more context. Therefore, low-context samples have high loss mainly because their conditional entropy is intrinsically high, not because the model still has much to learn from them. Consequently, standard MDM training repeatedly samples low-context problems that provide little additional learning signal, thereby wasting training budget. We summarize this observation as follows:

Finding 2. Low-context samples are not hard to learn; they are learned early and then waste the training budget of MDM.

3.3 Why Locality Bias in the High-Context Region Makes MDM a Slow Learner

We have shown that low-context samples are training-inefficient due to locality bias. We now consider the opposite: we show that high-context samples are also training-inefficient due to locality bias. When learning $p_\theta(x^i|\mathbf{x}_t, t)$, the model learns to predict the masked token while using the observed real tokens to build representations. The problem is that, due to locality bias, real tokens that are far from the mask contribute very little useful information to this learning process.

We first show that real tokens far from a mask have little effect on the learning signal. For a controlled experiment, using an MDM trained for 1M steps, we randomly select one center mask and analyze

Table 1: Cosine similarity between gradients before and after masking 8 tokens within radius.

Radius r	4	8	16	32	128
Cos. Sim.	0.108	0.463	0.685	0.839	0.935

Table 2: Cosine similarity between the joint-mask gradient and the sum of single-mask gradients.

# of Masks	2	4	8	16	32
Cos. Sim.	0.999	0.997	0.994	0.985	0.976

Table 3: Various time-sampling distributions on $[0, 1]$ explored for MDM training. For Dirac-delta, Gaussian, and Laplace, they satisfy the definition of a bell-shaped distribution when $\mu = 1/2$.

Uniform	Dirac-delta	Truncated Gaussian	Truncated Laplace
$\pi(t) = 1$	$\pi(t) = \delta_\mu$	$\pi(t) \propto \exp\left(-\frac{(t-\mu)^2}{2\sigma^2}\right)$	$\pi(t) \propto \exp\left(-\frac{\lvert t-\mu\rvert}{b}\right)$

the gradient of this center mask with and without additionally masking nearby real tokens. More specifically, we set several radii r and randomly replace 8 tokens within each region with masks. As shown in Table 1: the cosine similarity is 0.108 when $r = 4$, but increases to 0.935 when $r = 128$. This indicates that real tokens farther from the mask have little impact on the learning signal.

We next show that the gradient of high-context samples can be replaced by that of middle-context samples. Fix the number of masks k, and randomly sample k distinct single-mask patterns, *i.e.*, $\mathbf{M}_1$. We then construct $\mathbf{M}_k$ by taking the union of all masked positions. For the same sequence $\mathbf{x}$, we measure the cosine similarity between the sum of the gradients obtained from the k single-mask patterns and the gradient obtained from a single $\mathbf{M}_k$, and report the results in Table 2. The cosine similarity remains very high across different values of k, suggesting that high-context samples are training-inefficient and can be effectively replaced by middle-context samples.

These results imply that many observed tokens in high-context samples do not introduce notable new learning signals, and middle-context samples can replace them. Consequently, even high-context training can be inefficient: although more tokens are observed, *they waste the training budget*:

Finding 3. Under locality bias, high-context samples waste the training budget of MDM.

4 Mitigating the Locality Bias Problem in MDMs

4.1 Training MDM with Bell-Shaped Time Sampling

Sec. 3 naturally suggests that MDM training can be improved by focusing more on middle-context problems. We propose the most straightforward way to achieve this: directly sampling more training points around $t = 0.5$, *i.e.*, the middle-context region. We first define a bell-shaped distribution:

Definition 4.1. A probability distribution π over $[0, 1]$ is called bell-shaped if $\mathbb{P}_{t\sim\pi}\left(t \in [a, 1-a]\right) \geq \mathbb{P}_{t\sim\mathrm{U}([0,1])}\left(t \in [a, 1-a]\right)$ for all $a \in [0, 1/2)$, and the inequality is strict for at least one $a \in [0, 1/2)$.

Given a bell-shaped time distribution π, we train MDMs with the following objective:

$$\hat{\mathcal{L}}_\pi = \mathbb{E}_{\mathbf{x}\sim p_{\mathrm{data}},\, t\sim\pi,\, \mathbf{x}_t\sim q_{t|0}(\cdot|\mathbf{x})}\left[-\textstyle\sum_{i:x_t^i=m} \log p_\theta(x^i \mid \mathbf{x}_t, t)\right]. \tag{6}$$

This objective is clearly different from the theoretical NELBO in Eq. (1): it samples t from the tilted distribution π, and it does not include the coefficient $\alpha_t'/(1-\alpha_t)$ in the integrand. Nevertheless, the change does not alter the target optimizer under the standard time-agnostic parameterization (Sahoo et al., 2024; Nie et al., 2025b; Ye et al., 2025; Hong et al., 2026b). In fact, the following statement is stronger: the optimizer is preserved not only for bell-shaped distributions, but for any time-sampling distribution with nonzero mass on $(0, 1)$:

Proposition 4.2. *With infinite-capacity and time-agnostic model θ, for any probability distribution ρ over $[0, 1]$ satisfying $\mathbb{P}_{t\sim\rho}\left(t \in (0, 1)\right) > 0$, the minimizers of $\mathcal{L}_{\mathrm{MDM}}$ and $\hat{\mathcal{L}}_\rho$ are equal.*

See Appendix B.2 for proof and note that $\mathbb{P}_{t\sim\pi}\left(t \in (0, 1)\right) > 0$ follows from Definition 4.1. In this regard, our objective is to obtain $\arg\min_\theta \mathcal{L}_{\mathrm{MDM}}$ by learning $\hat{\mathcal{L}}_\pi$, which magnifies a learning signal on the middle-context region. As shown in Table 3, we consider several simple distributions with varying hyperparameters to understand which time sampling laws are beneficial.

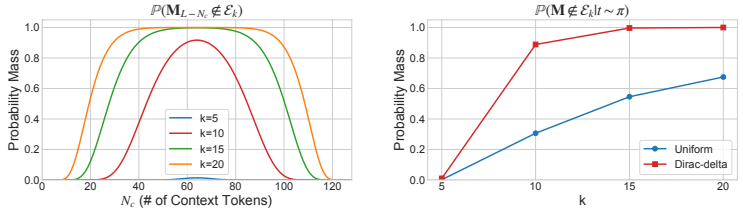


Figure 4: *Left:* Probability that a uniformly sampled mask pattern with exactly $L - N_c$ masks avoids k-inefficient set $\mathcal{E}_k$. *Right:* Probability that each uniform and Dirac-delta time sampling avoids $\mathcal{E}_k$.

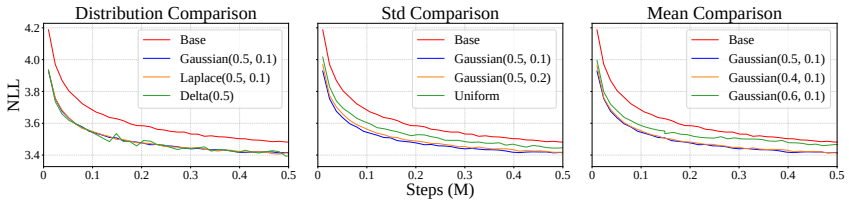


Figure 5: Validation NLL curves on LM1B for models trained with various time distributions listed in Table 3. *Left*: comparison across distribution families (μ=0.5). *Middle*: effect of variance where uniform can be treated as an extreme case. *Right*: effect of the mean μ under fixed σ=0.1. Here, "Base" denotes the conventional MDM trained with $\mathcal{L}_{\text{MDM}}$ (Eq. (1)) and is shown in red in all panels. Unlike "Uniform" with $\hat{\mathcal{L}}_\pi$ (Eq. (6)), "Base" includes the $\alpha'_t/(1-\alpha_t)$ weighting in the loss integrand.

4.2 Theoretical Analysis

We now investigate which training samples are emphasized by bell-shaped time sampling. Recall that $\mathbf{M}_n$ is a mask indicator with n masked positions, and $\mathcal{M}_n$ denotes the set of all possible $\mathbf{M}_n$.

Definition 4.3 (k-inefficient set $\mathcal{E}_k$)**.** If $\mathbf{M}$ contains k consecutive masks or k consecutive non-masks, we call $\mathbf{M}$ a k-inefficient mask pattern. We denote the set of all such k-inefficient patterns by $\mathcal{E}_k$.

The value of meaningful k may vary depending on 1) the intrinsic local bias of the language data, or 2) the model's capacity to exploit local information. Under locality bias, we regard samples from $\mathcal{E}_k$ as *training-inefficient*: if we flip masks on the samples from $\mathcal{E}_k$, we might obtain a sample that gives a more diverse training signal. This is supported by Sec. 3.2, where we show that low-context samples converge quickly, and Sec. 3.3, where we directly show that middle-context samples can effectively replace high-context samples. We visualize $\mathbb{P}(\mathbf{M}_{L-N_c} \notin \mathcal{E}_k \mid \mathbf{M}_{L-N_c} \sim \mathrm{U}(\mathcal{M}_{L-N_c}))$ in Fig. 4 (*left*), where $\mathcal{E}_k$ has substantial probability mass for low- and high-context regions. Now we show the following:

Proposition 4.4. *For any L, $2 \le k \le L$, and any bell-shaped time distribution π, the following statement always holds:*

$$\mathbb{P}\big(\mathbf{M} \notin \mathcal{E}_k \mid t \sim \mathrm{U}([0,1]), M^i \sim \text{Bernoulli}(t)\big) < \mathbb{P}\big(\mathbf{M} \notin \mathcal{E}_k \mid t \sim \pi, M^i \sim \text{Bernoulli}(t)\big).$$

See Appendix B.3 for the full proof. The above proposition implies that, when sampling with any bell-shaped distribution π, *training observes fewer k-inefficient samples* than under conventional MDM sampling. We further visualize two variants, $\pi = \mathrm{U}([0,1])$ and $\pi = \delta_{0.5}$, in Fig. 4. As shown in Fig. 4 (*right*), $\mathbb{P}(\mathbf{M} \notin \mathcal{E}_k \mid t \sim \delta_{0.5})$ is much higher than with uniform sampling, suggesting that bell-shaped time sampling significantly filters out k-consecutive inefficient samples.

5 Experiments

5.1 Training Dynamics of MDM with Various Time Sampling Distributions

To confirm our understanding, we first analyze the trends exhibited by the various time sampling distributions in Table 3. We train models on sentence-packed LM1B (Chelba et al., 2014) and measure validation NLL bound based on Eq. (1). We follow prior works (Sahoo et al., 2024; Shi et al., 2024) for the model architecture (∼100M scale).

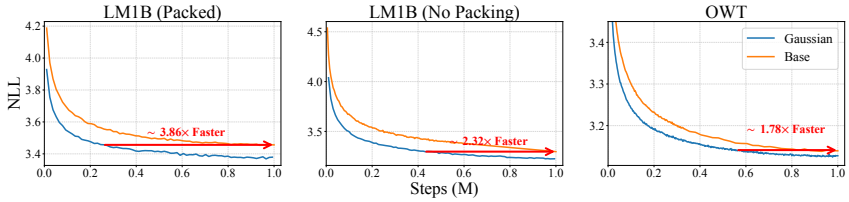


Figure 6: Validation NLL curves on various language modeling benchmarks. "Base" denotes base (standard) MDM, and "Gaussian" denotes MDM trained with $\hat{\mathcal{L}}_{\mathcal{N}(0.5,0.1)}$.

Table 4: Language understanding and completion benchmark results for continual pretraining (CPT)-only and supervised fine-tuned (SFT) models. CPT was performed on GPT-2 Large (0.7B) for 100K steps, and SFT was performed for 10K steps. Gray rows indicate models learned by CPT with Gaussian time sampling. Boldface denotes the best result within each section. The colored values in parentheses indicate the absolute change relative to the corresponding Base row within each section. Results are averaged over evaluation with 5 random seeds, where ± denotes one standard deviation.

Training		Benchmarks							
CPT	SFT	LAMBADA (%)	Obqa (%)	Wino (%)	PIQA (%)	SIQA (%)	Infilling	TriQA(%)	HellaSwag (%)
Continual Pretraining									
Base	—	2.15 ± 0.43	17.00 ± 0.472	**49.49** ± 1.20	52.12 ± 0.36	35.72 ± 0.46	6.94 ± 0.09	0.8 ± 0.09	28.87 ± 1.31
Gaussian	—	**10.25** ± 0.29 (+8.10)	**20.00** ± 1.68 (+3.00)	**49.49** ± 0.33 (+0.00)	**53.75** ± 1.10 (+1.63)	**37.05** ± 1.11 (+1.33)	**7.31** ± 0.20 (+0.37)	**1.3** ± 1.02 (+0.5)	**34.70** ± 1.44 (+5.83)
Continual Pretraining + Supervised Fine-tuning									
Base	Base	5.82 ± 0.15	18.20 ± 0.28	**50.59** ± 0.67	56.80 ± 0.23	38.95 ± 0.80	9.87 ± 0.15	6.4 ± 0.19	32.59± 1.24
Base	Gaussian	5.65 ± 0.05	20.00 ± 0.42	48.15 ± 0.39	56.47 ± 0.92	38.54 ± 0.47	9.48 ± 0.21	7.6 ± 0.06	32.95± 1.14
Gaussian	Base	15.54 ± 0.36	**24.6** ± 0.57	50.36 ± 1.95	**59.30** ± 0.38	41.04 ± 0.18	12.18 ± 0.21	27.4 ± 0.07	39.29± 1.33
Gaussian	Gaussian	**17.89** ± 0.37 (+12.07)	23.4 ± 1.41 (+5.2)	49.80 ± 1.12 (-0.79)	**59.30** ± 0.88 (+2.50)	**42.07** ± 0.04 (+3.12)	**12.62** ± 0.38 (+2.75)	**27.8** ± 0.06 (+21.4)	**39.84**± 1.21 (+7.25)

The results are shown in Fig. 5. In the left panel, **all bell-shaped distributions substantially outperform the standard MDM baseline**, while the specific class of distribution is not particularly important. The Dirac-delta, however, shows less stable dynamics. In the middle panel, the smaller variance ($\sigma = 0.1$) outperforms σ=0.2 and uniform, indicating the power of bell-shaped time sampling. In the right panel, shifting the mean toward the low-context region (μ=0.6) yields the worst result. Based on these observations, we adopt a Gaussian with μ=0.5, σ=0.1 as our default, offering stable training and natural centering at the middle-context region. Refer to Appendix C.1 for detailed experimental settings and Appendix C.2 for further results with various hyperparameters.

5.2 Evaluating Bell-Shaped Time Sampling Across Multiple Benchmarks

Experimental settings. We train an MDM with Gaussian time sampling and a base MDM on three datasets, including One Billion Words dataset (LM1B) (Chelba et al., 2014) with/without sentence packing, and OpenWebText (OWT) (Gokaslan et al., 2019). We generally follow the widely adopted experimental settings in discrete diffusion language modeling (Sahoo et al., 2025; Arriola et al., 2025): 1) validation NLL for models trained on LM1B and OWT, and 2) zero-shot PPL and generative PPL for models trained on OWT. See Appendix C.1 for details.

Results. The main results are shown in Fig. 6. As shown in Fig. 6, Gaussian time sampling consistently accelerates MDM training across all language modeling benchmarks. Using the validation NLL achieved by the base MDM after 1M training steps as the target, Gaussian time sampling reaches the same NLL about $\mathbf{3.86}\times$ faster on sentence-packed LM1B, $\mathbf{2.32}\times$ faster on LM1B without packing, and $\mathbf{1.78}\times$ faster on OWT. These results indicate that the benefit of bell-shaped time sampling is robust across different data settings rather than being limited to a specific benchmark. The same trend also appears in zero-shot PPL and generative PPL evaluations; refer to Appendix C.3.

5.3 Scaling Up to Billion-Scale Masked Diffusion Models

Recent large-scale MDM (Ye et al., 2025; Bie et al., 2025; Fan et al., 2026; Tae et al., 2025) training recipes have increasingly shifted toward continual pretraining (CPT) from ARM. In brief, this approach (Gong et al., 2025) initializes the MDM with weights from the pretrained ARM and then continues with MDM pretraining. In this regard, we show that our method can scale to a real-world MDM training recipe as well. We adopt the most conventional CPT (Gong et al., 2025) and use pretrained GPT-2 Large at $\sim$0.7B parameter scale. Details are in Appendix C.1.

Language understanding and completion benchmarks. We evaluate on multiple-choice benchmarks such as HellaSwag (Zellers et al., 2019), Obqa (Mihaylov et al., 2018), PIQA (Bisk et al., 2020), TriQA (Joshi et al., 2017), SIQA (Sap et al., 2019), and WinoGrande (Sakaguchi et al., 2021), as well as last-word prediction on LAMBADA (Paperno et al., 2016) and story infilling on ROCStories (Mostafazadeh et al., 2016), evaluated using ROUGE score (Lin, 2004). We report results both after CPT only and after subsequent supervised fine-tuning (SFT) in Table 4. Gaussian time sampling clearly outperforms the baseline across most tasks both for CPT only and SFT setup.

Open-ended instruction-following benchmark. Beyond NLL-based metrics and multiple-choice QA accuracy, we evaluate whether Gaussian time sampling improves the conditional generation quality. We evaluate models on two instruction following benchmarks, MT-Bench (Zheng et al., 2023) and AlpacaEval 2.0 (Li et al., 2023). We report win rates (WR) judged by GPT-4o (Hurst et al., 2024) under the non-tie protocol; for AlpacaEval 2.0, we also report the length-controlled win rate (LC). We fine-tune the model on the ShareGPT dataset to equip it with conversational ability. Table 5 reports the results, where the model trained with Gaussian time sampling clearly wins the baseline. This indicates that the benefits of bell-shaped time sampling extend beyond NLL-based metrics and translate into improved generation quality required in the real world. Qualitative examples are in Appendix E.

Table 5: Win rate of Gaussian time sampling compared to base MDM, judged by GPT-4o. Results are averaged over 5 random seeds, with $\pm$ denoting one standard deviation.

Benchmark	WR (%)
AlpacaEval 2.0(WR)	78.27 ± 1.37
AlpacaEval 2.0(LC)	78.61 ± 1.42
MT-Bench (non-tie)	84.34 ± 7.26

To sum up, our bell-shaped time sampling can (i) scale to billion-parameter-scale models, (ii) generalize to CPT with ARM initialization, and (iii) boost downstream task performance.

5.4 Others Explored for Focusing Training on the Middle-Context Region

We analyze other ways to bias MDM training toward the middle-context region.

Middle-flat noise scheduler. One can flatten the noise scheduler around $\alpha_t = 0.5$ in $\mathcal{L}_{\text{MDM}}$, so that the model naturally encounters more middle-context samples during training. The key difference from bell-shaped sampling is whether the target cross-entropy loss is reweighted. Under the theoretical NELBO, the scaling factor is smallest around $t = 0.5$ and increases as t moves away from the center. As a result, the objective is theoretically equivalent to the standard MDM loss with $\alpha_t = 1-t$ (Shi et al., 2024; Sahoo et al., 2024). Thus, although the model observes more middle-context during training, the learning signal assigned to each N_c remains the same as in the original $\mathcal{L}_{\text{MDM}}$.

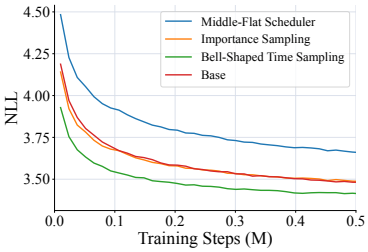


Figure 7: Training curve on LM1B.

Importance sampling. Another option is to sample masks uniformly, as in conventional MDMs, while assigning a larger scaling factor to the middle region. Specifically, we train with $\int_0^1 w(t)\mathbb{E}_{\mathbf{x}\sim p_{\text{data}},\, \mathbf{x}_t\sim q_{t|0}}\left[-\sum_{i:x_t^i=m} \log p_\theta(x^i \mid \mathbf{x}_t, t)\right] dt$, where $w(t)$ is chosen to be largest near the center. In this case, the learning signal is similar to that of bell-shaped time sampling, but the model does not actually encounter middle-context samples more frequently during training.

We found that only the MDM trained with bell-shaped time sampling learns faster than the base MDM. See Appendix C.4 for detailed settings and our interpretation of the corresponding results.

6 Conclusion

We studied why masked diffusion language models learn slowly and identified locality bias as a key factor behind their inefficient training dynamics. Motivated by this analysis, we propose bell-shaped time sampling, a simple training strategy that emphasizes the middle-context region, and show that it consistently accelerates MDM training and improves performance across various language modeling tasks. Although it may appear similar to those in continuous diffusion models (Karras et al., 2022; Esser et al., 2024), Sec. 3 and 5.4 show that our contribution is a non-trivial parallel contribution. We provide a detailed discussion of this difference in Appendix A, and limitations in Appendix D.

AI USE STATEMENT

In this work, we used generative AI tools for writing assistance, including grammar checking, writing polish, checking proofs for errors, and improving the clarity and readability of author-written text. We also used these tools to review and debug code, and develop preliminary figure layouts. We reviewed all AI-assisted work and take responsibility for the final content of this work.

ETHICS STATEMENT

This work studies an improved training strategy for discrete diffusion language models. It does not introduce new datasets, involve human subjects, or deploy models in user-facing settings. The potential societal impacts are therefore similar to those of existing generative language models. We do not anticipate additional ethical risks specific to this work.

REPRODUCIBILITY STATEMENT

We provide complete proofs of our theoretical results in Appendix B. Experimental settings, including datasets, preprocessing, model architectures, and training hyperparameters are provided in Appendix C.1. We will release the code upon acceptance.

REFERENCES

Marianne Arriola, Aaron Gokaslan, Justin T. Chiu, Zhihan Yang, Zhixuan Qi, Jiaqi Han, Subham Sekhar Sahoo, and Volodymyr Kuleshov. Block diffusion: Interpolating between autoregressive and diffusion language models. In *International Conference on Learning Representations (ICLR)*, 2025.

Jacob Austin, Daniel D Johnson, Jonathan Ho, Daniel Tarlow, and Rianne Van Den Berg. Structured denoising diffusion models in discrete state-spaces. In *Advances in Neural Information Processing Systems (NeurIPS)*, volume 34, pp. 17981–17993, 2021.

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

Yonatan Bisk, Rowan Zellers, Jianfeng Gao, Yejin Choi, et al. Piqa: Reasoning about physical commonsense in natural language. In *Proceedings of the AAAI conference on artificial intelligence*, volume 34, pp. 7432–7439, 2020.

Ciprian Chelba, Tomas Mikolov, Mike Schuster, Qi Ge, Thorsten Brants, Phillipp Koehn, and Tony Robinson. One billion word benchmark for measuring progress in statistical language modeling. In *Interspeech 2014*, pp. 2635–2639, 2014. doi: 10.21437/Interspeech.2014-564.

Arman Cohan, Franck Dernoncourt, Doo Soon Kim, Trung Bui, Seokhwan Kim, Walter Chang, and Nazli Goharian. A discourse-aware attention model for abstractive summarization of long documents. In *Proceedings of the 2018 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies, Volume 2 (Short Papers)*, pp. 615–621, 2018. doi: 10.18653/v1/N18-2097. URL https://aclanthology.org/N18-2097/.

Patrick Esser, Sumith Kulal, Andreas Blattmann, Rahim Entezari, Jonas Müller, Harry Saini, Yam Levi, Dominik Lorenz, Axel Sauer, Frederic Boesel, et al. Scaling rectified flow transformers for high-resolution image synthesis. In *International Conference on Machine Learning (ICML)*, 2024.

Chenghao Fan, Wen Heng, Bo Li, Sichen Liu, Yuxuan Song, Jing Su, Xiaoye Qu, Kai Shen, and Wei Wei. Stable-diffcoder: Pushing the frontier of code diffusion large language model. *arXiv preprint arXiv:2601.15892*, 2026.

Richard Futrell. Information-theoretic locality properties of natural language. In *Proceedings of the First Workshop on Quantitative Syntax (Quasy, SyntaxFest 2019)*, pp. 2–15. Association for Computational Linguistics, 2019. doi: 10.18653/v1/W19-7902. URL https://aclanthology.org/W19-7902/.

Aaron Gokaslan, Vanya Cohen, Ellie Pavlick, and Stefanie Tellex. Openwebtext corpus. http://Skylion007.github.io/OpenWebTextCorpus, 2019.

Shansan Gong, Shivam Agarwal, Yizhe Zhang, Jiacheng Ye, Lin Zheng, Mukai Li, Chenxin An, Peilin Zhao, Wei Bi, Jiawei Han, Hao Peng, and Lingpeng Kong. Scaling diffusion language models via adaptation from autoregressive models. In *International Conference on Learning Representations (ICLR)*, 2025.

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. In *The Fourteenth International Conference on Learning Representations*, 2026. URL https://openreview.net/forum?id=58NA3unZj5.

Aaron Grattafiori, Abhimanyu Dubey, Abhinav Jauhri, Abhinav Pandey, Abhishek Kadian, Ahmad Al-Dahle, Aiesha Letman, Akhil Mathur, Alan Schelten, Alex Vaughan, et al. The llama 3 herd of models. *arXiv preprint arXiv:2407.21783*, 2024.

Tiankai Hang, Shuyang Gu, Jianmin Bao, Fangyun Wei, Dong Chen, Xin Geng, and Baining Guo. Improved noise schedule for diffusion training. In *IEEE International Conference on Computer Vision (ICCV)*, pp. 4796–4806, 2025.

Marton Havasi, Brian Karrer, Itai Gat, and Ricky T. Q. Chen. Edit flows: Variable length discrete flow matching with sequence-level edit operations. In *Advances in Neural Information Processing Systems (NeurIPS)*, 2025.

Satoshi Hayakawa, Yuhta Takida, Masaaki Imaizumi, Hiromi Wakaki, and Yuki Mitsufuji. Distillation of discrete diffusion through dimensional correlations. In *Proceedings of the 42nd International Conference on Machine Learning*, pp. 22259–22297. PMLR, 2025.

Satoshi Hayakawa, Yuhta Takida, Masaaki Imaizumi, Hiromi Wakaki, and Yuki Mitsufuji. Demystifying MaskGIT sampler and beyond: Adaptive order selection in masked diffusion. *Transactions on Machine Learning Research*, 2026.

Zhengfu He, Tianxiang Sun, Qiong Tang, Kuanning Wang, Xuan-Jing Huang, and Xipeng Qiu. Diffusionbert: Improving generative masked language models with diffusion models. In *Proceedings of the 61st annual meeting of the association for computational linguistics (ACL)*, volume 1, pp. 4521–4534, 2023.

Michael Hersche, Samuel Moor-Smith, Thomas Hofmann, and Abbas Rahimi. Soft-masked diffusion language models. In *International Conference on Learning Representations (ICLR)*, 2026.

Chunsan Hong, Seonho An, Min-Soo Kim, and Jong Chul Ye. Improving discrete diffusion unmasking policies beyond explicit reference policies. In *International Conference on Learning Representations (ICLR)*, 2026a.

Chunsan Hong, Sanghyun Lee, and Jong Chul Ye. Unifying masked diffusion models with various generation orders and beyond. *arXiv preprint arXiv:2602.02112*, 2026b.

Emiel Hoogeboom, Didrik Nielsen, Priyank Jaini, Patrick Forré, and Max Welling. Argmax flows and multinomial diffusion: Learning categorical distributions. In *Advances in Neural Information Processing Systems (NeurIPS)*, volume 34, pp. 12454–12465, 2021.

Binyuan Hui, Jian Yang, Zeyu Cui, Jiaxi Yang, Dayiheng Liu, Lei Zhang, Tianyu Liu, Jiajun Zhang, Bowen Yu, Keming Lu, et al. Qwen2. 5-coder technical report. *arXiv preprint arXiv:2409.12186*, 2024.

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, 2024. URL https://arxiv.org/abs/2410.21276.

Mengni Jia, Mengyu Zhou, Yihao Liu, xiaoxi jiang, and guanjunjiang. Bringing stability to diffusion: Decomposing and reducing variance of training masked diffusion models. In *International Conference on Learning Representations (ICLR)*, 2026.

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

Tero Karras, Miika Aittala, Timo Aila, and Samuli Laine. Elucidating the design space of diffusion-based generative models. In *Advances in Neural Information Processing Systems (NeurIPS)*, 2022.

Jaemin Kim and Jong Chul Ye. Adaptive guidance for retrieval-augmented masked diffusion models. *arXiv preprint arXiv:2603.17677*, 2026.

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. In *International Conference on Machine Learning (ICML)*, 2025.

Jaeyeon Kim, Lee Cheuk Kit, Carles Domingo-Enrich, Yilun Du, Sham M. Kakade, Timothy Ngotiaoco, Sitan Chen, and Michael Samuel Albergo. Any-order flexible length masked diffusion. In *International Conference on Learning Representations (ICLR)*, 2026.

Florent Krzakala and Lenka Zdeborová. Hiding quiet solutions in random constraint satisfaction problems. *Physical review letters*, 102(23):238701, 2009.

Sanghyun Lee, Seungryong Kim, Jongho Park, and Dongmin Park. Lookahead unmasking elicits accurate decoding in diffusion language models. *arXiv preprint arXiv:2511.05563*, 2025.

Xuechen Li, Tianyi Zhang, Yann Dubois, Rohan Taori, Ishaan Gulrajani, Carlos Guestrin, Percy Liang, and Tatsunori B Hashimoto. Alpacaeval: An automatic evaluator of instruction-following models. `https://github.com/tatsu-lab/alpaca_eval`, 2023.

Chin-Yew Lin. Rouge: A package for automatic evaluation of summaries. In *Text summarization branches out*, pp. 74–81, 2004.

Anji Liu, Oliver Broadrick, Mathias Niepert, and Guy Van den Broeck. Discrete copula diffusion. In *International Conference on Learning Representations (ICLR)*, 2025.

Aaron Lou, Chenlin Meng, and Stefano Ermon. Discrete diffusion modeling by estimating the ratios of the data distribution. In *International Conference on Machine Learning (ICML)*, 2024.

Mitch Marcus, Beatrice Santorini, and Mary Ann Marcinkiewicz. Building a large annotated corpus of english: The penn treebank. *Computational linguistics*, 19(2):313–330, 1993.

Stephen Merity, Caiming Xiong, James Bradbury, and Richard Socher. Pointer sentinel mixture models. In *International Conference on Learning Representations (ICLR)*, 2017.

Todor Mihaylov, Peter Clark, Tushar Khot, and Ashish Sabharwal. Can a suit of armor conduct electricity? a new dataset for open book question answering. In *Proceedings of the 2018 conference on empirical methods in natural language processing*, pp. 2381–2391, 2018.

Nasrin Mostafazadeh, Nathanael Chambers, Xiaodong He, Devi Parikh, Dhruv Batra, Lucy Vanderwende, Pushmeet Kohli, and James Allen. A corpus and cloze evaluation for deeper understanding of commonsense stories. In *Proceedings of the 2016 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies*, pp. 839–849, 2016.

Shen Nie, Fengqi Zhu, Chao Du, Tianyu Pang, Qian Liu, Guangtao Zeng, Min Lin, and Chongxuan Li. Scaling up masked diffusion models on text. In *International Conference on Learning Representations (ICLR)*, 2025a.

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. In *Advances in Neural Information Processing Systems (NeurIPS)*, 2025b.

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. In *International Conference on Learning Representations (ICLR)*, 2025.

Denis Paperno, Germán Kruszewski, Angeliki Lazaridou, Ngoc-Quan Pham, Raffaella Bernardi, Sandro Pezzelle, Marco Baroni, Gemma Boleda, and Raquel Fernández. The lambada dataset: Word prediction requiring a broad discourse context. In *Proceedings of the 54th annual meeting of the association for computational linguistics (ACL)*, volume 1, pp. 1525–1534, 2016.

William Peebles and Saining Xie. Scalable diffusion models with transformers. In *IEEE International Conference on Computer Vision (ICCV)*, pp. 4195–4205, 2023.

Fred Zhangzhi Peng, Zachary Bezemek, Jarrid Rector-Brooks, Shuibai Zhang, Michael M. Bronstein, Anru Zhang, Joey Bose, and Alexander Tong. Planner aware path learning in diffusion language models training. In *International Conference on Learning Representations (ICLR)*, 2026.

Mihir Prabhudesai, Mengning Wu, Amir Zadeh, Katerina Fragkiadaki, and Deepak Pathak. Diffusion beats autoregressive in data-constrained settings. In *Advances in Neural Information Processing Systems (NeurIPS)*, 2025.

Patrick Pynadath, Jiaxin Shi, and Ruqi Zhang. Candi: Hybrid discrete-continuous diffusion models. *arXiv preprint arXiv:2510.22510*, 2025.

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

Colin Raffel, Noam Shazeer, Adam Roberts, Katherine Lee, Sharan Narang, Michael Matena, Yanqi Zhou, Wei Li, and Peter J Liu. Exploring the limits of transfer learning with a unified text-to-text transformer. *Journal of machine learning research*, 21(140):1–67, 2020.

Subham Sahoo, Marianne Arriola, Yair Schiff, Aaron Gokaslan, Edgar Marroquin, Justin Chiu, Alexander Rush, and Volodymyr Kuleshov. Simple and effective masked diffusion language models. In *Advances in Neural Information Processing Systems (NeurIPS)*, volume 37, pp. 130136–130184, 2024.

Subham Sekhar Sahoo, Justin Deschenaux, Aaron Gokaslan, Guanghan Wang, Justin Chiu, and Volodymyr Kuleshov. The diffusion duality. In *International Conference on Machine Learning (ICML)*, 2025.

Subham Sekhar Sahoo, Jean-Marie Lemercier, Zhihan Yang, Justin Deschenaux, Jingyu Liu, John Thickstun, and Ante Jukic. Scaling beyond masked diffusion language models. *arXiv preprint arXiv:2602.15014*, 2026.

Keisuke Sakaguchi, Ronan Le Bras, Chandra Bhagavatula, and Yejin Choi. Winogrande: An adversarial winograd schema challenge at scale. *Communications of the ACM*, 64(9):99–106, 2021.

Victor Sanh, Albert Webson, Colin Raffel, Stephen H Bach, Lintang Sutawika, Zaid Alyafeai, Antoine Chaffin, Arnaud Stiegler, Teven Le Scao, Arun Raja, et al. Multitask prompted training enables zero-shot task generalization. *arXiv preprint arXiv:2110.08207*, 2021.

Maarten Sap, Hannah Rashkin, Derek Chen, Ronan Le Bras, and Yejin Choi. Social iqa: Commonsense reasoning about social interactions. In *Proceedings of the 2019 conference on empirical methods in natural language processing and the 9th international joint conference on natural language processing (EMNLP-IJCNLP)*, pp. 4463–4473, 2019.

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. In *International Conference on Learning Representations (ICLR)*, 2025.

Jiaxin Shi, Kehang Han, Zhe Wang, Arnaud Doucet, and Michalis Titsias. Simplified and generalized masked diffusion for discrete data. In *Advances in Neural Information Processing Systems (NeurIPS)*, volume 37, pp. 103131–103167, 2024.

Yuxuan Song, Zheng Zhang, Cheng Luo, Pengyang Gao, Fan Xia, Hao Luo, Zheng Li, Yuehang Yang, Hongli Yu, Xingwei Qu, et al. Seed diffusion: A large-scale diffusion language model with high-speed inference. *arXiv preprint arXiv:2508.02193*, 2025.

Jianlin Su, Murtadha Ahmed, Yu Lu, Shengfeng Pan, Wen Bo, and Yunfeng Liu. Roformer: Enhanced transformer with rotary position embedding. *Neurocomput.*, 568(C), February 2024. ISSN 0925-2312. doi: 10.1016/j.neucom.2023.127063. URL https://doi.org/10.1016/j.neucom.2023.127063.

Jaesung Tae, Hamish Ivison, Sachin Kumar, and Arman Cohan. Tess 2: A large-scale generalist diffusion language model. In *Proceedings of the 63rd Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, pp. 21171–21188, 2025.

Yuqing Wang, Ye He, and Molei Tao. Evaluating the design space of diffusion-based generative models. In *Advances in Neural Information Processing Systems (NeurIPS)*, 2024.

Tianyu Xie, Shuchen Xue, Zijin Feng, Tianyang Hu, Jiacheng Sun, Zhenguo Li, and Cheng Zhang. Variational autoencoding discrete diffusion with enhanced dimensional correlations modeling. In *International Conference on Learning Representations (ICLR)*, 2026.

Zhihui Xie, Jiacheng Ye, Lin Zheng, Jiahui Gao, Jingwei Dong, Zirui Wu, Xueliang Zhao, Shansan Gong, Xin Jiang, Zhenguo Li, et al. Dream-coder 7b: An open diffusion language model for code. *arXiv preprint arXiv:2509.01142*, 2025.

Shuchen Xue, Tianyu Xie, Tianyang Hu, Zijin Feng, Jiacheng Sun, Kenji Kawaguchi, Zhenguo Li, and Zhi-Ming Ma. Any-order GPT as masked diffusion model: Decoupling formulation and architecture. In *ES-FoMo III: 3rd Workshop on Efficient Systems for Foundation Models*, 2025. URL https://openreview.net/forum?id=KbRxn8fzrY.

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

Chuanyue Yu, Jiahui Wang, Yuhan Li, Heng Chang, Ge Lan, Qingyun Sun, Jia Li, Jianxin Li, and Ziwei Zhang. Unlocking the potentials of retrieval-augmented generation for diffusion language models. *arXiv preprint arXiv:2601.11342*, 2026.

Rowan Zellers, Ari Holtzman, Yonatan Bisk, Ali Farhadi, and Yejin Choi. Hellaswag: Can a machine really finish your sentence? In *Proceedings of the 57th annual meeting of the association for computational linguistics*, pp. 4791–4800, 2019.

Xiang Zhang, Junbo Zhao, and Yann LeCun. Character-level convolutional networks for text classification. In *Advances in Neural Information Processing Systems (NeurIPS)*, volume 28, 2015.

Siyan Zhao, Devaansh Gupta, Qinqing Zheng, and Aditya Grover. d1: Scaling reasoning in diffusion large language models via reinforcement learning. In *Advances in Neural Information Processing Systems (NeurIPS)*, 2025.

Huangjie Zheng, Shansan Gong, Ruixiang ZHANG, Tianrong Chen, Jiatao Gu, Mingyuan Zhou, Navdeep Jaitly, and Yizhe Zhang. Continuously augmented discrete diffusion model for categorical generative modeling. In *The Fourteenth International Conference on Learning Representations*, 2026. URL https://openreview.net/forum?id=JNAZ3e7Bwt.

Kaiwen Zheng, Yongxin Chen, Hanzi Mao, Ming-Yu Liu, Jun Zhu, and Qinsheng Zhang. Masked diffusion models are secretly time-agnostic masked models and exploit inaccurate categorical sampling. In *International Conference on Learning Representations (ICLR)*, 2025.

Lianmin Zheng, Wei-Lin Chiang, Ying Sheng, Siyuan Zhuang, Zhanghao Wu, Yonghao Zhuang, Zi Lin, Zhuohan Li, Dacheng Li, Eric Xing, et al. Judging llm-as-a-judge with mt-bench and chatbot arena. *Advances in neural information processing systems*, 36:46595–46623, 2023.

Cai Zhou, Chenxiao Yang, Yi Hu, Chenyu Wang, Chubin Zhang, Muhan Zhang, Lester Mackey, Tommi Jaakkola, Stephen Bates, and Dinghuai Zhang. Coevolutionary continuous discrete diffusion: Make your diffusion language model a latent reasoner. *arXiv preprint arXiv:2510.03206*, 2025.

CONTENTS

1 Introduction **1**

2 Background **2**

2.1 Shared Format of Learning Objectives Across Various Frameworks 3

3 Why Masked Diffusion Models Learn Slowly **3**

3.1 Why Learning Any-order Language Generation is Hard 3

3.2 Why Locality Bias in the Low-Context Region Makes MDM a Slow Learner 4

3.3 Why Locality Bias in the High-Context Region Makes MDM a Slow Learner . . . 5

4 Mitigating the Locality Bias Problem in MDMs **6**

4.1 Training MDM with Bell-Shaped Time Sampling 6

4.2 Theoretical Analysis 7

5 Experiments **7**

5.1 Training Dynamics of MDM with Various Time Sampling Distributions 7

5.2 Evaluating Bell-Shaped Time Sampling Across Multiple Benchmarks 8

5.3 Scaling Up to Billion-Scale Masked Diffusion Models 8

5.4 Others Explored for Focusing Training on the Middle-Context Region 9

6 Conclusion **9**

A Background **16**

A.1 Related Work 16

A.2 Various Language Modeling Frameworks in Detail 17

B Proofs **19**

B.1 Proof of Proposition 3.1 20

B.2 Proof of Proposition 4.2 21

B.3 Proof of Proposition 4.4 23

C Omitted Experimental Details and Results **30**

C.1 Omitted Experimental Details 30

C.2 Additional Ablation on Bell-shaped Time Sampling 31

C.3 Zero-shot Perplexity and Generative Perplexity of Models Trained on OpenWebText 32

C.4 Other Explored Training Techniques in Detail 33

C.5 Stratified Sampling 34

D Limitations **35**

E Qualitative Examples **35**

A BACKGROUND

A.1 RELATED WORK

Discrete diffusion models. Discrete diffusion models (Austin et al., 2021; Hoogeboom et al., 2021) have emerged as a potential alternative framework for language modeling. The main objective of discrete diffusion modeling is to model the discrete data distribution via a continuous-time diffusion process. There are broadly two types of forward corruption processes in discrete diffusion models: (1) uniform corruption (Lou et al., 2024; Sahoo et al., 2025), which replaces tokens with uniformly random tokens, and (2) masking corruption (Sahoo et al., 2024; Shi et al., 2024), which replaces tokens with a special [MASK] token. Among these, masked diffusion models (MDMs) have emerged as a leading class of discrete diffusion models for text generation (Sahoo et al., 2024).

Various frameworks based on masked diffusion models. One of the most active recent research directions has been the development of new frameworks that modify the diffusion process itself in order to overcome the theoretical limitations of MDMs. Such efforts include block generation scheme (Arriola et al., 2025), modeling the joint probability distribution (Xie et al., 2026; Liu et al., 2025; Hayakawa et al., 2025), introducing new operations such as editing (Kim et al., 2026; Havasi et al., 2025), relaxing the discrete tokenization by augmenting the model with additional latent (Zheng et al., 2026; Zhou et al., 2025; Pynadath et al., 2025; Hersche et al., 2026), and learned generation order rather than random order (Peng et al., 2026; Hong et al., 2026b).

Standard masked diffusion models. Despite these extensions, the simple masking-based MDM framework (Ou et al., 2025; Sahoo et al., 2024) remains the most widely used backbone for large-scale text generation, largely due to its simplicity. Recent large-scale models such as LLaDA (Nie et al., 2025b), Seed Diffusion (Song et al., 2025), and Dream (Ye et al., 2025) retain the standard generation scheme, where a fully masked sequence is gradually converted into tokens. Many recent advances, including inference-time decoding strategies (Lee et al., 2025; Hayakawa et al., 2026), reinforcement learning methods (Hong et al., 2026a; Zhao et al., 2025), and retrieval-augmented generation (Yu et al., 2026; Kim & Ye, 2026), are also built on this basic framework. Empirically, standard MDMs have become increasingly competitive with autoregressive models (ARMs): LLaDA (Nie et al., 2025b) and Dream (Ye et al., 2025) outperform same-sized LLaMA3 models (Grattafiori et al., 2024), while DiffuCoder (Gong et al., 2026) and Dream-Coder (Xie et al., 2025) match same-sized Qwen2.5-Coder models (Hui et al., 2024).

Training inefficiency of masked diffusion models. Meanwhile, MDMs require $\sim 16\times$ more training compute than ARMs to reach the same validation loss (Prabhudesai et al., 2025; Sahoo et al., 2026), which may become a major obstacle when scaling MDMs to real-world LLMs. Several prior works (Kim et al., 2025; Xue et al., 2025) attribute this slow learning to the left-to-right (L2R) bias of language; yet, *how* this bias makes MDM training-inefficient remains underexplored. On the other hand, Jia et al. (Jia et al., 2026) suggest that the variance of the batch gradient estimator may be a key issue; yet lacks an experimental analysis of their hypothesis.

The work of Kim *et al*. (Kim et al., 2025) is closely related to ours in that it also tackles the training difficulty of MDMs. They compare σ-ARMs with permutations σ that are close to the L2R order, fully random permutations σ, and MDMs, showing that order-agnostic training, as in MDMs (and different from AoARMs), is difficult and that L2R bias further hinders learning. In contrast, our work compares the structurally similar σ-ARM and AoARM to identify the dominant factor among order-space complexity and locality bias. Furthermore, our study empirically analyzes how locality bias makes MDM training difficult, whereas Kim *et al*. take a fundamentally different approach by proving, based on a physics-inspired conjecture (Krzakala & Zdeborová, 2009), that order-agnostic training inevitably includes unlearnable subproblems. Finally, another key distinction is that Kim *et al*. propose an inference-time strategy, whereas we propose a training-time strategy.

Bell-shaped time sampling in continuous diffusion models. In continuous diffusion modeling, bell-shaped time sampling has also been studied as an effective training strategy. EDM (Karras et al., 2022), Stable Diffusion 3 (Esser et al., 2024), and Hang et al. (2025) all concentrate training samples around intermediate noise levels, using log-normal, logit-normal, or related middle-biased distributions. However, they motivate this concentration from different perspectives. EDM emphasizes intermediate noise levels because this is where training can make the most meaningful progress: at very low noise, the remaining perturbation becomes vanishingly small and difficult to resolve, whereas at

very high noise, the optimal denoiser approaches coarse data statistics and the sample-specific target becomes increasingly uninformative (Karras et al., 2022). Stable Diffusion 3 instead emphasizes the *intrinsic difficulty of the prediction problem*: near the two endpoints of rectified flow, the optimal velocity reduces to simple distributional means, while intermediate timesteps require substantially more nontrivial prediction (Esser et al., 2024). More importantly, Wang et al. (2024) provide a theoretical explanation for this behavior: under their deep-ReLU score-network analysis, denoising residuals are difficult to reduce at extreme noise scales, whereas intermediate scales are easier to fit because the input and output scales are better matched, leading their convergence analysis to favor bell-shaped total weighting. Taken together, EDM motivates middle-biased sampling from where optimization makes meaningful progress, Stable Diffusion 3 from where the prediction problem is most nontrivial, and Wang et al. (2024) from a theoretical analysis showing why the extreme-noise regimes are harder to fit.

Our main contribution is to characterize *why* the two endpoint regimes are training-inefficient in MDMs, and the underlying mechanisms are fundamentally different from those identified in continuous diffusion. In the high-noise (low-context) regime, we separate irreducible conditional entropy from reducible model error and show that the latter largely converges early, so these samples provide little additional learning signal despite their high loss. In the low-noise (high-context) regime, reducible error may remain, yet locality bias makes many distant observed tokens redundant and contributes little new gradient information. Thus, although bell-shaped sampling takes a similar form to prior continuous-diffusion strategies, its justification in MDMs does not directly follow from them. Rather, it emerges naturally from our analysis as a way to reallocate training budget away from both inefficient endpoint regimes. We further support this connection theoretically and empirically, including our analysis of inefficient mask patterns and alternative middle-biased training strategies in Sec. 5.4.

A.2 Various Language Modeling Frameworks in Detail

Masked diffusion models. Masked diffusion models (MDMs) (Sahoo et al., 2024; Shi et al., 2024) define the forward corruption process using the absorbing mask strategy: once a token is masked, it remains masked throughout the remaining process. For the diffusion process, define the time interval as $t \in \mathcal{T} = [0, 1]$ where we corrupt the data from $t = 0$ (least noisy) to $t = 1$ (most noisy). Formally, the forward process $q(\mathbf{x}_t^i \mid \mathbf{x}) = q_{t|0}($ is designed as follows:

$$q(\mathbf{x}_t^i \mid \mathbf{x}) = q_{t|0}(\cdot \mid x^i) = \mathrm{Cat}(\alpha_t \mathbf{e}_{x^i} + (1 - \alpha_t)\mathbf{e}_m)\,,$$

where the forward process gradually adds noise as t grows, *i.e.*, the noise scheduler should satisfy $\alpha_0 = 1$, $\alpha_1 = 0$, $\alpha_t' < 0$. The noise scheduler (Sahoo et al., 2024; Shi et al., 2024) is typically set to linear scheduler, $\alpha_t = 1 - t$. Following prior works (Sahoo et al., 2024; Hong et al., 2026b), discretize the time interval $\mathcal{T}$ with $T + 1$ steps, and define $s(\tau) = \tau/(T + 1)$ and $t(\tau) = (\tau + 1)/(T + 1)$. Then the generative distribution is divided into T diffusion reverse steps ($\mathbf{x}_{t(T)} \to \cdots \to \mathbf{x}_{t(0)}$) and 1 reconstruction step ($\mathbf{x}_{t(0)} \to \mathbf{x}$). The true reverse posterior based on corresponding forward process is derived as follows:

$$q(\mathbf{x}_s^i \mid \mathbf{x}_t, \mathbf{x}) = q(\mathbf{x}_s^i \mid \mathbf{x}_t^i, \mathbf{x}) = \begin{cases} \mathrm{Cat}(\mathbf{e}_{x_t^i}), & \text{if } \mathbf{x}_t^i \neq \mathbf{m}, \\ \mathrm{Cat}\Big(\frac{1-\alpha_s}{1-\alpha_t}\mathbf{e}_m + \frac{\alpha_s - \alpha_t}{1-\alpha_t}\mathbf{e}_{x^i}\Big)\,, & \text{if } x_t^i = \mathbf{m}. \end{cases}$$

where we drop τ in $s(\tau)$ and $t(\tau)$ for brevity. To mimic the true reverse posterior, with the diffusion network $\mu_\theta^i(\mathbf{x}_t, t) \in \Delta^{V+1}$ that predicts the clean token x^i over the vocabulary, the parametrized reverse process is designed as follows:

$$p_\theta(\mathbf{x}_s^i \mid \mathbf{x}_t) = q\big(\mathbf{x}_s^i \mid \mathbf{x}_t, \mathbf{x} = \mathbf{x}_\theta(\mathbf{x}_t, t)\big) = \begin{cases} \mathrm{Cat}(\mathbf{e}_{x_t^i}), & \text{if } \mathbf{x}_t^i \neq \mathbf{m}, \\ \mathrm{Cat}\Big(\frac{1-\alpha_s}{1-\alpha_t}\mathbf{e}_m + \frac{\alpha_s - \alpha_t}{1-\alpha_t}\mu_\theta^i(\mathbf{x}_t, t)\Big)\,, & \text{if } x_t^i = \mathbf{m}. \end{cases}$$

where $s < t$. Note that conventional MDM (Sahoo et al., 2024; Nie et al., 2025b; Ye et al., 2025; Hong et al., 2026b) typically set model output of mask probability as zero. Due to

curse of dimensionality (Lou et al., 2024), MDM models the reverse process as a token-wise conditionally independent distribution rather than capturing full joint distribution, *i.e.*, $p_\theta(\mathbf{x}_s \mid \mathbf{x}_t) = \prod_{i=1}^{L} p_\theta(\mathbf{x}_s^i \mid \mathbf{x}_t)$. In this regard, the model distribution is expressed as $p_\theta(\mathbf{x}) = \sum_{\mathbf{x}_{t(0:T)}} p_\theta(\mathbf{x}_{t(T)}) p_\theta(\mathbf{x}|\mathbf{x}_{t(0)}) \prod_{\tau=1}^{T} p_\theta(\mathbf{x}_{s(\tau)}|\mathbf{x}_{t(\tau)})$ in the discrete-time case. Finally, with $T \to \infty$, the continuos time NELBO is given as follows:

$$\mathcal{L}_{\text{MDM}} = \int_0^1 \frac{-\alpha_t'}{1-\alpha_t} \mathbb{E}_{\mathbf{x}\sim p_{\text{data}}, \mathbf{x}_t \sim q_{t|0}} \Big[- \sum_{i:x_t^i = m} \log p_\theta(x^i|\mathbf{x}_t, t) \Big] dt.$$

where $p_\theta(x^i|\mathbf{x}_t, t) = \mu_\theta^i(\mathbf{x}_t, t)$ and $\frac{-\alpha_t'}{1-\alpha_t} > 0$. Prior works (Sahoo et al., 2024; Zheng et al., 2025; Ou et al., 2025) have reported time-agnostic parameterization gives better performance than that of time-variant parameterization, and inspired by this, MDM typically utilizes time-agnostic network (Sahoo et al., 2024; Nie et al., 2025b; Ye et al., 2025; Hong et al., 2026b), *i.e.*, $\mu_\theta^i(\mathbf{x}_t, t) = \mu_\theta^i(\mathbf{x}_t)$.

Various views of MDM objective. Zheng *et al*. (Zheng et al., 2025) have shown that the time-variant MDM objective can be rewritten with the time-agnostic form, and Kim *et al*. (Kim et al., 2025) have shown that the time-agnostic MDM objective can be rewritten as any-order autoregressive objective form. For a sequence $\mathbf{x}$ and a mask indicator $\mathbf{M} \in \{0,1\}^L$, define the masked sequence $\mathbf{x}[\mathbf{M}]$ that is obtained by replacing x^i with m whenever $M^i = 1$, while leaving it unchanged whenever $M^i = 0$. Let $\mathbf{M}_n$ as mask indicator with mask number of n, *i.e.*, $\sum_i M_n^i = n$, and $\mathcal{M}_n$ as set of all possible combinatorial $\mathbf{M}_n$. Then, Zheng *et al*. have proven that:

Lemma A.1 (Proposition 3.1 of Zheng *et al*. (Zheng et al., 2025))**.** *For the time-variant diffusion network $\mu_\theta^i(\cdot, t) \in \Delta^{V+1}$, the theoretical NELBO of MDM, $\mathcal{L}_{\text{MDM}}$ (Eq. (1)), can be rewritten as follows:*

$$\mathcal{L}_{\text{MDM}} = \mathbb{E}_{\mathbf{x}\sim p_{\text{data}}} \Big[\sum_{N_c=0}^{L-1} \mathbb{E}_{\mathbf{M}_{L-N_c}\sim U(\mathcal{M}_{L-N_c}),\, i\sim U(i:M^i_{L-N_c}=1)} \big[-\log \bar{p}_\theta(x^i \mid \mathbf{x}[\mathbf{M}_{L-N_c}]) \big] \Big], \tag{7}$$

where $\log \bar{p}_\theta(x^i \mid \mathbf{x}[\mathbf{M}_{L-N_c}]) := \mathbb{E}_{\alpha_{N_c}} \big[\log p_\theta(x^i \mid \mathbf{x}_t = \mathbf{x}[\mathbf{M}_{L-N_c}], t = \alpha_t^{-1}(\alpha_{N_c})) \big]$, $\alpha_{N_c} \sim \text{Beta}(N_c + 1, L - N_c)$, *and* $p_\theta(x^i \mid \mathbf{x}[\mathbf{M}_{L-N_c}], t) = \mu_\theta^i(\mathbf{x}[\mathbf{M}_{L-N_c}], t)$.

Note that the original version of Proposition 3.1 of Zheng *et al*. includes a $1/(L - N_c)$ scaling term, which is accounted for in our notation by the expectation over the sampled index i. We note that $\bar{p}_\theta$ is not normalized; in particular, $\sum_{a\in[V]} \bar{p}_\theta(x^i = a \mid \mathbf{x}[\mathbf{M}]) \neq 1$.

Motivated by above lemma, Kim *et al*. have shown that:

Lemma A.2 (Proposition 2.1 of Kim *et al*. (Kim et al., 2025))**.** *For the time-agnostic diffusion network $\mu_\theta^i(\cdot) \in \Delta^{V+1}$, the theoretical NELBO of MDM, $\mathcal{L}_{\text{MDM}}$ (Eq. (1)), can be rewritten as follows:*

$$\mathcal{L}_{\text{MDM}} = \mathbb{E}_{\mathbf{x}\sim p_{\text{data}}} \Big[\sum_{N_c=0}^{L-1} \mathbb{E}_{\mathbf{M}_{L-N_c}\sim \text{U}(\mathcal{M}_{L-N_c}), i\sim \text{U}(i:M^i_{L-N_c}=1)} \big[-\log p_\theta(x^i|\mathbf{x}[\mathbf{M}_{L-N_c}]) \big] \Big]. \tag{8}$$

where $p_\theta(x^i \mid \mathbf{x}[\mathbf{M}_{L-N_c}]) = \mu_\theta^i(\mathbf{x}[\mathbf{M}_{L-N_c}])$.

Again, note that the original version of Proposition 2.1 of Kim *et al*. includes a $\frac{1}{\binom{L}{L-N_c}\cdot(L-N_c)}$ scaling term, which is accounted for in our notation by the expectation over the sampled index i and sampled mask $\mathbf{M}_{L-N_c}$.

Shared format of learning objectives. Recall the shared objective in Eq. (4):

$$\mathcal{L}_{\mathcal{F}} = \sum_{N_c=0}^{L-1} \mathcal{L}_{\mathcal{F}}^{N_c}, \quad \mathcal{L}_{\mathcal{F}}^{N_c} := \mathbb{E}_{\mathbf{x}\sim p_{\text{data}}, \mathbf{z}_{N_c}\sim p_{\mathcal{F}}(\mathbf{z}|\mathbf{x},N_c), i\sim p_{\mathcal{F}}(i|\mathbf{x},\mathbf{z}_{N_c})} [-\log p_\theta(x^i \mid \mathbf{z}_{N_c})], \tag{9}$$

where N_c is the number of context tokens, $\mathbf{z}_{N_c}$ is the partially observed context, and $p_{\mathcal{F}}(\mathbf{z} \mid \mathbf{x}, N_c)$ and $p_{\mathcal{F}}(i \mid \mathbf{x}, \mathbf{z}_{N_c})$ specify how each framework constructs the context and selects the target position.

Table 6: Framework-specific choices that instantiate the shared objective in Eq. (4). Entries are written as sampling rules; deterministic entries correspond to point masses at the displayed value.

Framework $\mathcal{F}$	Partially observed latent $\mathbf{z}_{N_c}$	$p_{\mathcal{F}}(\mathbf{z} \mid \mathbf{x}, N_c)$	$p_{\mathcal{F}}(i \mid \mathbf{x}, \mathbf{z}_{N_c})$
ARM	$\mathbf{x}^{<N_c+1}$	$\mathbf{z}_{N_c} = \mathbf{x}^{<N_c+1}$	$i = N_c + 1$
σ-ARM	$\mathbf{x}^{<\sigma_{N_c+1}}$	$\mathbf{z}_{N_c} = \mathbf{x}^{<\sigma_{N_c+1}}$	$i = \sigma_{N_c+1}$
AoARM	$(\mathbf{x}^{<\sigma_{N_c+1}}, \sigma)$	$\sigma \sim U(S_L), \quad \mathbf{z}_{N_c} = (\mathbf{x}^{<\sigma_{N_c+1}}, \sigma)$	$i = \sigma_{N_c+1}$
Time-agnostic MDM	$\mathbf{x}[\mathbf{M}_{L-N_c}]$	$\mathbf{M}_{L-N_c} \sim U(\mathcal{M}_{L-N_c})$, $\mathbf{z}_{N_c} = \mathbf{x}[\mathbf{M}_{L-N_c}]$	$i \sim U(\{i : z^i_{N_c} = m\})$
Time-variant MDM	$(\mathbf{x}[\mathbf{M}_{L-N_c}], \alpha_{N_c})$	$\mathbf{M}_{L-N_c} \sim U(\mathcal{M}_{L-N_c}), \quad \alpha_{N_c} \sim \mathrm{Beta}(N_c+1, L-N_c)$, $\mathbf{z}_{N_c} = (\mathbf{x}[\mathbf{M}_{L-N_c}], \alpha_{N_c})$	$i \sim U(\{i : z^i_{N_c} = m\})$

We spell out these framework-specific choices in Table 6. Since N_c counts observed tokens whereas $\mathbf{M}_n$ counts masked positions, the MDM row uses $\mathbf{M}_{L-N_c}$.

Substituting the ARM row into Eq. (9) recovers Eq. (2). Similarly, for a fixed permutation σ,

$$\sum_{N_c=0}^{L-1} \mathcal{L}^{N_c}_{\sigma\text{-ARM}} = \mathbb{E}_{\mathbf{x}\sim p_{\mathrm{data}}} \sum_{r=1}^{L} \Big[-\log p_\theta^{\sigma\text{-ARM}}\big(x^{\sigma_r} \mid \mathbf{x}^{<\sigma_r}\big) \Big],$$

and averaging the same construction over $\sigma \sim U(S_L)$ gives the AoARM objective in Eq. (3):

$$\sum_{N_c=0}^{L-1} \mathcal{L}^{N_c}_{\mathrm{AoARM}} = \mathbb{E}_{\mathbf{x}\sim p_{\mathrm{data}},\, \sigma\sim U(S_L)} \Big[-\sum_{r=1}^{L} \log p_\theta^{\mathrm{AoARM}}\big(x^{\sigma_r} \mid \mathbf{x}^{<\sigma_r}, \sigma\big) \Big].$$

For time-agnostic MDM, Eq. (9) becomes the context-level loss

$$\mathcal{L}^{N_c}_{\mathrm{MDM}} = \mathbb{E}_{\mathbf{x}\sim p_{\mathrm{data}},\, \mathbf{M}_{L-N_c}\sim U(\mathcal{M}_{L-N_c}),\, i\sim U(i:M^i_{L-N_c}=1)} \big[-\log p_\theta(x^i \mid \mathbf{x}[\mathbf{M}_{L-N_c}]) \big], \tag{10}$$

and this matches the theoretical NELBO of MDM by Lemma A.2.

For a time-variant MDM, Eq. (9) becomes

$$\mathcal{L}^{N_c}_{\mathrm{MDM}} = \mathbb{E}_{\substack{\mathbf{x}\sim p_{\mathrm{data}},\, \mathbf{M}_{L-N_c}\sim U(\mathcal{M}_{L-N_c}) \\ i\sim U(\{i:M^i_{L-N_c}=1\})\alpha_{N_c}\sim \mathrm{Beta}(N_c+1, L-N_c)}} \big[-\log p_\theta(x^i \mid \mathbf{x}[\mathbf{M}_{L-N_c}], \alpha_t^{-1}(\alpha_{N_c})) \big], \tag{11}$$

and this matches the theoretical NELBO of MDM by Lemma A.1 where we explicitly include p_θ rather than $\bar{p}_\theta$.

Remark. Since all experiments in our work use time-agnostic MDMs, as in typical MDM parameterizations (Sahoo et al., 2024; Nie et al., 2025b; Ye et al., 2025; Hong et al., 2026b), we simply evaluate $\log p_\theta(x^i \mid \mathbf{x}[\mathbf{M}_{L-N_c}])$ by $\log \mu^i_\theta(\mathbf{x}[\mathbf{M}_{L-N_c}])$. In other words, throughout all of experiments conducted in our paper, we measured $\mathcal{L}^{N_c}_{\mathrm{MDM}}$ by evaluating Eq. (10).

B PROOFS

Before presenting the proofs, we introduce a shorthand for the data conditional induced by a partially observed context. For a partially observed context $\mathbf{z}$, let $C_\#(\mathbf{z}) \subseteq [L]$ denote the set of positions whose clean token values are revealed by $\mathbf{z}$. For example, for a masked sequence $\mathbf{x}[\mathbf{M}]$, since $\mathbf{M}^j = 1$ means that position j is masked, we have

$$C_\#(\mathbf{x}[\mathbf{M}]) = \{j \in [L] : \mathbf{M}^j = 0\}.$$

For an autoregressive prefix $\mathbf{x}^{<i}$, we have

$$C_\#(\mathbf{x}^{<i}) = \{1, \ldots, i-1\}.$$

Similarly, for a permuted prefix $\mathbf{x}^{<\sigma_r}$, we have

$$C_\#(\mathbf{x}^{<\sigma_r}) = \{\sigma_1, \ldots, \sigma_{r-1}\}.$$

For each $j \in C_{\#}(\mathbf{z})$, we write z^j for the revealed token value at position j.

Using this notation, for any target position $i \in [L]$, token value $a \in [V]$, and partially observed context $\mathbf{z}$ with positive data mass, we define

$$p_{\text{data}}(x^i = a \mid \mathbf{z}) \coloneqq p_{\text{data}}\big(X^i = a \,\big|\, X^j = z^j,\ \forall j \in C_{\#}(\mathbf{z})\big) = \frac{\sum_{\tilde{\mathbf{x}} \in \mathcal{X} : \tilde{x}^i = a,\ \tilde{x}^j = z^j,\ \forall j \in C_{\#}(\mathbf{z})} p_{\text{data}}(\tilde{\mathbf{x}})}{\sum_{\tilde{\mathbf{x}} \in \mathcal{X} : \tilde{x}^j = z^j,\ \forall j \in C_{\#}(\mathbf{z})} p_{\text{data}}(\tilde{\mathbf{x}})}. \tag{12}$$

When $a = x^i$ is clear from context, we simply write $p_{\text{data}}(x^i \mid \mathbf{z})$. If the denominator is zero, the conditional distribution can be defined arbitrarily, since such a context is never sampled under p_{data}.

B.1 Proof of Proposition 3.1

We prove Proposition 3.1, which was omitted in the main paper. We first provide one useful lemma:

Lemma B.1 (Appendix E.1 of Kim *et al.* (Kim et al., 2025))**.** *For $\sigma \in S_L$ where S_L is any permutation of $[L]$, let $[\mathbf{x}^{<\sigma_r} : \mathbf{M}]$ denotes the masked sequence that reveals $\{x^{\sigma_1}, \ldots, x^{\sigma_{r-1}}\}$ and masks all remaining positions, rather than the raw subsequence. For the time-agnostic diffusion network $\mu_\theta^i(\cdot) \in \Delta^{V+1}$, the theoretical NELBO of MDM, $\mathcal{L}_{\text{MDM}}$ (Eq. (1)), can be rewritten as follows:*

$$\mathcal{L}_{\text{MDM}} = \mathbb{E}_{\mathbf{x} \sim p_{\text{data}}, \sigma \sim U(S_L)} \left[\sum_{r=1}^{L} - \log p_\theta(x^{\sigma_r} \mid [\mathbf{x}^{<\sigma_r} : \mathbf{M}]) \right]. \tag{13}$$

where $p_\theta(x^i \mid \mathbf{x}[\mathbf{M}_{L-N_c}]) = \mu_\theta^i(\mathbf{x}[\mathbf{M}_{L-N_c}])$.

Now we prove the following:

Proposition 3.1. *With infinite model capacity, the optimal loss for any framework $\mathcal{F} \in \{\text{ARM}, \text{MDM}, \sigma\text{−ARM}, \text{AoARM}\}$ is equal to the entropy of the data distribution:*

$$\min_\theta \mathcal{L}_{\mathcal{F}} = H(X), \quad H(X) \coloneqq -\mathbb{E}_{\mathbf{x} \sim p_{\text{data}}}\big[\log p_{\text{data}}(\mathbf{x})\big]. \tag{14}$$

Furthermore, letting I denote the random target index, $\mathcal{L}_{\mathcal{F}}^{N_c}$ can be decomposed as

$$\mathcal{L}_{\mathcal{F}}^{N_c}(\theta) = H_{\mathcal{F}}(X^I \mid Z_{N_c}, I) + \mathbb{E}_{Z_{N_c}, I} \mathrm{KL}(p_{\text{data},\mathcal{F}}(\cdot \mid Z_{N_c}, I) \,\|\, p_\theta(\cdot \mid Z_{N_c}, I)).$$

where the conditional entropy is induced by p_{data}.

Proof. We first prove the first claim,

$$\min_\theta \mathcal{L}_{\mathcal{F}} = H(X), \qquad \mathcal{F} \in \{\text{ARM}, \sigma\text{-ARM}, \text{AoARM}, \text{MDM}\},$$

For ARM,

$$L_{\text{ARM}} = -\mathbb{E}_{\mathbf{x} \sim p_{\text{data}}}\Big[\sum_{i=1}^{L} \log p_\theta^{\text{ARM}}(x^i \mid \mathbf{x}^{<i})\Big] = -\mathbb{E}_{\mathbf{x} \sim p_{\text{data}}} \log p_\theta^{\text{ARM}}(\mathbf{x}) = H(X) + \mathrm{KL}\big(p_{\text{data}}(\mathbf{x}) \,\|\, p_\theta^{\text{ARM}}(\mathbf{x})\big) \geq H(X),$$

where equality holds when $p_\theta(x^i|\mathbf{x}^{<i}) = p_{\text{data}}(x^i|\mathbf{x}^{<i})$ satisfies almost everywhere. Note that we can prove for σ-ARM case as above. For AoARM,

$$\mathcal{L}_{\text{AoARM}} = -\mathbb{E}_{\sigma \sim U(S_L)} \mathbb{E}_{\mathbf{x} \sim p_{\text{data}}}[\log p_\theta^{\text{AoARM}}(\mathbf{x} \mid \sigma)] = \mathbb{E}_{\sigma \sim U(S_L)}\Big[H(X) + \mathrm{KL}\big(p_{\text{data}}(\mathbf{x}) \,\|\, p_\theta^{\text{AoARM}}(\mathbf{x} \mid \sigma)\big)\Big] \geq H(X),$$

Under infinite model capacity, the equality is attained by

$$p_\theta^{\text{AoARM}}(\mathbf{x}^{\sigma_i} \mid \mathbf{x}^{<\sigma_i}, \sigma) = p_{\text{data}}(\mathbf{x}^{\sigma_i} \mid \mathbf{x}^{<\sigma_i}) \qquad \text{for all } i \in [L] \text{ and } \sigma \in S_L.$$

For time-variant MDM, rather than decomposing NELBO into above decomposition, we directly show that $\min_\theta \mathcal{L}_{\text{MDM}} = H(X)$ is achievable when $p_\theta(x^i = a \mid \mathbf{x}_t, t) = p_{\text{data}}(x^i = a \mid \mathbf{x}_t)$ for all $i \in [L]$, for all $\mathbf{x}_t \in \{\mathbf{x}[\mathbf{M}] : \forall \mathbf{x} \in \mathcal{X}, \forall \mathbf{M} \in \{0,1\}^L\}$, and for all $a \in [V]$.

Recall that NELBO of MDM is as follows:

$$\mathbb{E}_{p_{\text{data}}}[-\log p_\theta(\mathbf{x})] \le \mathcal{L}_{\text{MDM}} = \int_0^1 \frac{-\alpha_t'}{1-\alpha_t} \mathbb{E}_{\mathbf{x}\sim p_{\text{data}}, \mathbf{x}_t \sim q_{t|0}} \Big[- \sum_{i: x_t^i = m} \log p_\theta(x^i | \mathbf{x}_t, t) \Big] dt. \tag{15}$$

Here, the left-hand side is lower bounded by $H(X)$. On the other hand, we show that if the infinite capacity model θ^* satisfies $p_{\theta^*}(x^i = a \mid \mathbf{x}_t, t) = p_{\text{data}}(x^i = a \mid \mathbf{x}_t)$, the right-hand side gives $H(X)$. If $p_{\theta^*}(x^i = a | \mathbf{x}_t, t) = p_{\text{data}}(x^i = a | \mathbf{x}_t)$ holds, since $p_{\theta^*}(x^i = a | \mathbf{x}_t, t)$ is parametrized by $\mu_{\theta^*}(\mathbf{x}_t, t)$, the optimal θ^* is no more time-variant rather time-agnostic. Substituting θ into θ^* such that $p_\theta(x^i = a | \mathbf{x}_t, t) = p_{\text{data}}(x^i = a | \mathbf{x}_t)$, and using Lemma B.1 and Eq. (12), the right-hand side gives

$$\mathbb{E}_{\mathbf{x}\sim p_{\text{data}}, \sigma \sim U(S_L)} \left[\sum_{r=1}^{L} - \log p_{\text{data}}(x^{\sigma_r} \mid \mathbf{x}^{<\sigma_r}) \right], \tag{16}$$

which is equivalent to $H(X)$. Therefore, with infinite model capacity, $\min_\theta \mathcal{L}_{\text{MDM}} = H(X)$, and this minimum is achievable. Note that time-agnostic MDM can be proven with the same approach as above.

We now prove the second claim. Recall that

$$\mathcal{L}_{\mathcal{F}}^{N_c} = \mathbb{E}_{\mathbf{x}\sim p_{\text{data}}, \mathbf{z}_{N_c} \sim p_{\mathcal{F}}(\mathbf{z}|\mathbf{x}, N_c), i \sim p_{\mathcal{F}}(i|\mathbf{x}, \mathbf{z}_{N_c})} [-\log p_\theta(x^i \mid \mathbf{z}_{N_c})]. \tag{17}$$

For a fixed observation level n, define the framework-induced joint distribution

$$p_{\text{data},\mathcal{F}}(\mathbf{x}, \mathbf{z}_{N_c}, i \mid N_c) := p_{\text{data}}(\mathbf{x})\, p_{\mathcal{F}}(\mathbf{z}_{N_c} \mid \mathbf{x}, N_c)\, p_{\mathcal{F}}(i \mid \mathbf{x}, \mathbf{z}_{N_c}).$$

Also, with a slight abuse of notation, write

$$p_\theta(a \mid \mathbf{z}_{N_c}, i) := p_\theta(x^i = a \mid \mathbf{z}_{N_c}),$$

namely, the model prediction for the i-th token given $\mathbf{z}_{N_c}$. Then,

$$\begin{aligned}
\mathcal{L}_{\mathcal{F}}^{N_c} &= \mathbb{E}_{\mathbf{x}, \mathbf{z}_{N_c}, i} \big[-\log p_\theta(x^i \mid \mathbf{z}_{N_c}) \big] \\
&= \mathbb{E}_{(\mathbf{z}_{N_c}, i) \sim p_{\text{data},\mathcal{F}}(\cdot,\cdot|N_c)} \left[\sum_a p_{\text{data},\mathcal{F}}(a \mid \mathbf{z}_{N_c}, i) \big(-\log p_\theta(a \mid \mathbf{z}_{N_c}, i) \big) \right] \\
&= \mathbb{E}_{(\mathbf{z}_{N_c}, i) \sim p_{\text{data},\mathcal{F}}(\cdot,\cdot|N_c)} \Big[H\big(p_{\text{data},\mathcal{F}}(\cdot \mid \mathbf{z}_{N_c}, i)\big) + \mathrm{KL}(p_{\text{data},\mathcal{F}}(\cdot \mid \mathbf{z}_{N_c}, i) \,\|\, p_\theta(\cdot \mid \mathbf{z}_{N_c}, i)) \Big] \\
&= H_{\mathcal{F}}(X^I \mid Z_{N_c}, I) + \mathbb{E}_{(\mathbf{z}_{N_c}, i) \sim p_{\text{data},\mathcal{F}}(\cdot,\cdot|N_c)} \mathrm{KL}(p_{\text{data},\mathcal{F}}(\cdot \mid \mathbf{z}_{N_c}, i) \,\|\, p_\theta(\cdot \mid \mathbf{z}_{N_c}, i)),
\end{aligned}$$

which concludes the proof. Therefore, for every framework $\mathcal{F}$ and every observation level n,

$$\mathcal{L}_{\mathcal{F}}^{N_c} \ge H_{\mathcal{F}}(X^I \mid Z_{N_c}, I).$$

Moreover, equality holds if and only if

$$p_\theta(x^i \mid \mathbf{z}_{N_c}) = p_{\text{data},\mathcal{F}}(x^i \mid \mathbf{z}_{N_c}, i) \qquad \text{for } p_{\text{data},\mathcal{F}}(\mathbf{z}_{N_c}, i \mid N_c)\text{-almost every } (\mathbf{z}_{N_c}, i).$$

□

B.2 Proof of Proposition 4.2

We prove Proposition 4.2 in this section. We first provide useful lemma to prove the statement:

Lemma B.3 (Restatement of Zheng *et al*. (Zheng et al., 2025)). *Consider two different loss, $\mathcal{L}_{\text{MDM}}$ and weighted loss $\hat{\mathcal{L}}$ defined as follows:*

$$\mathcal{L}_{\text{MDM}}(\theta) = \sum_{N_c=0}^{L-1} \mathcal{L}_{\text{MDM}}^{N_c}(\theta) \qquad \hat{\mathcal{L}}(\theta) = \sum_{N_c=0}^{L-1} \omega(N_c) \mathcal{L}_{\text{MDM}}^{N_c}(\theta). \tag{18}$$

With infinite capacity and time-agnostic θ, if $\omega(N_c) > 0$ holds for all $N_c \in \{0, \dots, L-1\}$, the minimizer of $\mathcal{L}_{\text{MDM}}$ and $\hat{\mathcal{L}}$ are equal.

Proof. Note that above lemma can be inferred from the proof of Proposition 3.2 in Zheng *et al.* (Zheng et al., 2025). We here show the above statement more directly with our notations. Since the model is time-agnostic, we write $p_\theta(x^i \mid \mathbf{x}_t)$ instead of $p_\theta(x^i \mid \mathbf{x}_t, t)$. For $n \in \{0, \dots, L\}$, recall that $\mathcal{M}_n = \{\mathbf{M} \in \{0,1\}^L : \sum_{i=1}^{L} \mathbf{M}^i = n\}$ is the set of mask indicators with exactly n masked positions, where $\mathbf{M}^i = 1$ means that position i is masked. Also recall that $\mathbf{x}[\mathbf{M}_n]$ is the sequence obtained by replacing the masked positions of $\mathbf{x}$ with m. Thus, when the number of context tokens is N_c, the number of masked tokens is $L - N_c$, and the corresponding masked sequence is $\mathbf{x}[\mathbf{M}_{L-N_c}]$.

For different values of N_c, the corresponding masked contexts have disjoint input supports, since $\mathbf{x}[\mathbf{M}_{L-N_c}]$ contains exactly $L - N_c$ mask tokens. Hence, under infinite capacity and time-agnostic parameterization, the model can realize the pointwise optimum for each context level independently. Formally, define the common minimizer set

$$\Theta^\star := \big\{\theta : p_\theta(x^i = a \mid \mathbf{x}[\mathbf{M}_{L-N_c}]) = p_{\text{data}}(x^i = a \mid \mathbf{x}[\mathbf{M}_{L-N_c}])$$

$$\forall N_c = 0, \dots, L-1,\ \forall \mathbf{M}_{L-N_c} \in \mathcal{M}_{L-N_c},\ \forall i : M^i_{L-N_c} = 1,\ \forall \mathbf{x} \in [V]^L, \forall a \in [V]\big\}.$$

Under infinite capacity, this set is nonempty, since the model can realize the pointwise conditional distribution on every masked-context input.

To see that this is exactly the minimizer set, fix N_c, $\mathbf{M}_{L-N_c} \in \mathcal{M}_{L-N_c}$, a masked position i with $\mathbf{M}^i_{L-N_c} = 1$, and a masked context $\mathbf{x}_t$. The corresponding token-level cross-entropy is

$$\sum_{a \in [V]} p_{\text{data}}(x^i = a \mid \mathbf{x}[\mathbf{M}_{L-N_c}]) \big[-\log p_\theta(x^i = a \mid \mathbf{x}[\mathbf{M}_{L-N_c}])\big].$$

This can be decomposed as

$$H(p_{\text{data}}(\cdot \mid \mathbf{x}[\mathbf{M}_{L-N_c}])) + \mathrm{KL}(p_{\text{data}}(\cdot \mid \mathbf{x}[\mathbf{M}_{L-N_c}]) \,\|\, p_\theta(\cdot \mid \mathbf{x}[\mathbf{M}_{L-N_c}])).$$

The entropy term is independent of θ, and the KL term is minimized if and only if

$$p_\theta(x^i = a \mid x[\mathbf{M}_{L-N_c}]) = p_{\text{data}}(x^i = a \mid \mathbf{x}[\mathbf{M}_{L-N_c}])$$

for every token $a \in [V]$. Hence each context-level loss $\mathcal{L}^{N_c}_{\text{MDM}}$ is minimized exactly by matching the true conditional distribution on its corresponding masked contexts. Since $\hat{\mathcal{L}}$ and $\mathcal{L}_{\text{MDM}}$ are both positive weighted sums of the same context-level cross-entropy terms,

$$\hat{\mathcal{L}}(\theta) = \sum_{N_c=0}^{L-1} \omega(N_c) \mathcal{L}^{N_c}_{\text{MDM}}(\theta), \qquad \omega_\pi(N_c) > 0,$$

and

$$\mathcal{L}_{\text{MDM}}(\theta) = \sum_{N_c=0}^{L-1} \mathcal{L}^{N_c}_{\text{MDM}}(\theta),$$

their minimizers are exactly the same:

$$\arg\min_\theta \hat{\mathcal{L}}(\theta) = \Theta^\star = \arg\min_\theta \mathcal{L}_{\text{MDM}}(\theta).$$

□

We now prove the Proposition 4.2 with Lemma B.3. What we need to prove is that $w(N_c) > 0, \forall N_c \in \{0, \dots, L-1\}$ holds for any probability distribution ρ satisfying $\mathbb{P}_{t\sim\rho}(t \in (0,1)) > 0$.

Proposition 4.2. *With infinite-capacity and time-agnostic model θ, for any probability distribution ρ over $[0,1]$ satisfying $\mathbb{P}_{t\sim\rho}(t \in (0,1)) > 0$, the minimizers of $\mathcal{L}_{\text{MDM}}$ and $\hat{\mathcal{L}}_\rho$ are equal.*

Proof. Let ρ be a Borel probability measure on $[0,1]$. All integrals with respect to $d\rho(t)$ are understood as measure-theoretic integrals, so atomic choices such as $\rho = \delta_\mu$ are included. Under the absorbing forward process, for any $\mathbf{M}_{L-N_c} \in \mathcal{M}_{L-N_c}$,

$$\Pr(\mathbf{x}_t = \mathbf{x}[\mathbf{M}_{L-N_c}] \mid \mathbf{x}, t) = \alpha_t^{N_c}(1-\alpha_t)^{L-N_c}.$$

Therefore, after marginalizing over $t \sim \rho$, the probability of sampling exactly N_c context tokens is

$$\Pr_{\rho}(N_c) = \binom{L}{N_c} \mathbb{E}_{t\sim\rho}\left[\alpha_t^{N_c}(1-\alpha_t)^{L-N_c}\right] = \binom{L}{N_c} \int_{[0,1]} \alpha_t^{N_c}(1-\alpha_t)^{L-N_c}\, d\rho(t).$$

This formula remains valid when ρ has atoms; for example, if $\rho = \delta_\mu$, the integral simply evaluates the integrand at $t = \mu$. Conditioned on N_c, all mask indicators in $\mathcal{M}_{L-N_c}$ are sampled uniformly, because the above probability depends on $\mathbf{M}_{L-N_c}$ only through its cardinality. Hence

$$\begin{aligned}\hat{\mathcal{L}}_\rho(\theta) &= \mathbb{E}_{\mathbf{x}\sim p_{\text{data}},\, t\sim\rho,\, \mathbf{x}_t\sim q_{t|0}(\cdot|\mathbf{x})}\left[-\sum_{i:x_t^i=m} \log p_\theta(x^i \mid \mathbf{x}_t)\right] \\ &= \sum_{N_c=0}^{L-1} (L-N_c)\Pr_{\rho}(N_c)\,\mathcal{L}_{\text{MDM}}^{N_c}(\theta) = \sum_{N_c=0}^{L-1} \omega_\rho(N_c)\,\mathcal{L}_{\text{MDM}}^{N_c}(\theta),\end{aligned}$$

where $\omega_\rho(N_c) := (L-N_c)\binom{L}{N_c}\int_{[0,1]} \alpha_t^{N_c}(1-\alpha_t)^{L-N_c}\, d\rho(t)$. By definition of α_t in conventional MDM (Sahoo et al., 2024; Shi et al., 2024; Hong et al., 2026b) where $\alpha_0 = 1$, $\alpha_1 = 0$, and $\alpha_t' < 0$, the integrand $\alpha_t^{N_c}(1-\alpha_t)^{L-N_c}$ is strictly positive for every $t \in (0,1)$ and every $N_c \in \{0,\dots,L-1\}$. Since $\mathbb{P}_{t\sim\rho}(t \in (0,1)) > 0$, we have $\omega_\rho(N_c) > 0$ for all $N_c \in \{0,\dots,L-1\}$. Then, together with Lemma B.3, this concludes the proof. □

Remark. In measure-theoretic notation, the probability mass assigned by a probability distribution ρ to an interval $I \subseteq [0,1]$ is conventionally written as $\rho(I)$. To make this notation more accessible to readers less familiar with this convention, we instead write the same quantity as $\mathbb{P}_{t\sim\rho}(t \in I)$ throughout the main text and proofs.

B.3 Proof of Proposition 4.4

We prove Proposition 4.4 in this section. Recall that $\mathcal{E}_k$ denotes the set of k-inefficient mask indicators, i.e., those containing either k consecutive masks or k consecutive non-masks. Accordingly, $\mathcal{E}_k^c$ denotes the set of k-efficient mask indicators. Recall also that

$$\mathcal{M}_n = \Big\{\mathbf{M} \in \{0,1\}^L : \sum_{i=1}^{L} M^i = n\Big\}, \qquad \mathbf{M}_n \sim U(\mathcal{M}_n),$$

so that $\mathbf{M}_n$ is a uniformly sampled mask indicator with exactly n masked positions. Equivalently, it has exactly $L-n$ observed positions.

Define

$$q_{L,k}(n) := \mathbb{P}\big(\mathbf{M}_n \in \mathcal{E}_k^c \big| \mathbf{M}_n \sim U(\mathcal{M}_n)\big).$$

Let

$$N_k(a,b) := \#\Big\{\mathbf{M} \in \{0,1\}^{a+b} : \sum_{i=1}^{a+b} M^i = a,\ \mathbf{M} \in \mathcal{E}_k^c\Big\},$$

where a counts masked positions and b counts non-masked positions. Since k is fixed throughout this proof, we drop k in the notation. Then

$$q_{L,k}(n) = \frac{N(n, L-n)}{\binom{L}{n}}.$$

For a time sampling law π on $[0,1]$, define the induced mask-count law

$$w_\pi(n) := \mathbb{P}\left(\sum_{i=1}^{L} M^i = n \;\middle|\; t \sim \pi,\ M^i \overset{\text{i.i.d.}}{\sim} \text{Bernoulli}(t)\right).$$

Here π is understood as a Borel probability measure, and it need not admit a density. Conditioning on t gives

$$w_\pi(n) = \binom{L}{n} \int_{[0,1]} t^n (1-t)^{L-n}\, d\pi(t).$$

Therefore

$$\mathbb{P}\Big(\mathbf{M} \in \mathcal{E}_k^c \mid t \sim \pi, M^i \overset{\text{i.i.d.}}{\sim} \text{Bernoulli}(t)\Big) = \sum_{n=0}^{L} w_\pi(n) q_{L,k}(n). \tag{19}$$

In particular, when $\pi = U([0,1])$,

$$w_U(n) = \binom{L}{n} \int_0^1 t^n (1-t)^{L-n}\, dt = \frac{1}{L+1},$$

so uniform time sampling induces the uniform distribution over the mask count $n \in \{0, \dots, L\}$.

The key step is to show that $q_{L,k}(n)$ is maximized at the middle mask count.

Lemma B.5 (Central monotonicity of the efficient probability)**.** *For every $L \in \mathbb{N}$ and $k \geq 2$,*

$$q_{L,k}(n) = q_{L,k}(L-n),$$

and

$$q_{L,k}(n) \leq q_{L,k}(n+1) \qquad \textit{for all } 0 \leq n < \frac{L}{2}.$$

Consequently, $q_{L,k}(n)$ is maximized at $n = L/2$ when L is even, and at the two central points $(L-1)/2$ and $(L+1)/2$ when L is odd.

Proof. The symmetry

$$q_{L,k}(n) = q_{L,k}(L-n)$$

follows by flipping all mask bits, $0 \leftrightarrow 1$. This operation maps $\mathcal{M}_n$ bijectively to $\mathcal{M}_{L-n}$ and preserves the event $\mathcal{E}_k^c$, since it only swaps masks and non-masks.

It remains to prove monotonicity up to the center. Fix $n < m$ and set $L = n + m$. We want to prove

$$q_{L,k}(n) \leq q_{L,k}(n+1),$$

or equivalently

$$\frac{N(n,m)}{\binom{L}{n}} \leq \frac{N(n+1,m-1)}{\binom{L}{n+1}}.$$

Since

$$\frac{\binom{L}{n+1}}{\binom{L}{n}} = \frac{m}{n+1},$$

this is equivalent to

$$(n+1)N(n+1,m-1) - mN(n,m) \geq 0. \tag{20}$$

Unfortunately, $N(n,m)$ expands into a highly cumbersome form, making it difficult to prove the above inequality by a direct expansion. To handle this combinatorial counting problem, we use the concept of a *generating function*. Generating functions are a classical tool in enumerative combinatorics, used to encode combinatorial counts as coefficients of formal polynomials. For example, if a sequence $\{a_r\}_{r \geq 0}$ is encoded as

$$G(x) = \sum_{r \geq 0} a_r x^r,$$

then the coefficient-extraction notation $[x^r]G(x)$ denotes the coefficient of x^r in $G(x)$, namely a_r. In this sense, questions about the combinatorial numbers a_r can be translated into algebraic questions about the polynomial or power series $G(x)$. In the present proof, we will consider $F(x,y)$ that assigns a formal weight x to each masked position and y to each non-masked position; then the coefficient of $x^a y^b$ records how many k-efficient mask indicators have a masked and b non-masked positions. Equivalently, $[x^a y^b]F(x,y) = N(a,b)$ where $F(x,y) = \sum_{a,b \geq 0} N(a,b) x^a y^b$, so the *operation* $[x^a y^b]$ pulls out the coefficient of $x^a y^b$.

We divide three cases of $M \in \mathcal{E}_k^c$ to obtain $F(x,y)$: 1) the number of mask-runs and non-mask-runs are equal, *e.g.*, $\mathbf{M} = 1 \dots 0$ or $\mathbf{M} = 0 \dots 1$, 2) there is one more mask-run than non-mask-run

$\mathbf{M} = 1 \dots 1$, and 3) there is one less mask-run than non-mask-run $\mathbf{M} = 0 \dots 0$. To formalize $F(x, y)$, first let

$$P_k(z) := z + z^2 + \cdots + z^{k-1}.$$

Note the length of each consecutive 0's or 1's are in $\{1, \dots, k-1\}$. So, for example, the case starting from x and end at y with $2r$ total runs can be represented as

$$P_k(x)P_k(y) \cdots P_k(x)P_k(y) = P_k(x)^r P_k(y)^r.$$

Furthermore, there also exists the case starting from x and end at y with $2r$ total runs:

$$P_k(y)P_k(x) \cdots P_k(y)P_k(x) = P_k(x)^r P_k(y)^r.$$

Therefore, if the number of mask-runs and non-mask-runs are both equal to $r \geq 1$, then the total contribution is

$$2P_k(x)^r P_k(y)^r.$$

If there is one more mask-run than non-mask-run, the contribution is

$$P_k(x)^{r+1} P_k(y)^r \qquad (r \geq 0),$$

and similarly, if there is one more non-mask-run than mask-run, the contribution is

$$P_k(x)^r P_k(y)^{r+1} \qquad (r \geq 0).$$

To compute $F(x, y)$, we take their sum over $r \geq 0$ with the above cases. Then the bivariate generating function of nonempty k-efficient mask indicators is

$$F(x, y) := \sum_{a,b \geq 0} N(a, b) x^a y^b = \frac{P_k(x) + P_k(y) + 2P_k(x)P_k(y)}{1 - P_k(x)P_k(y)}.$$

The empty string is irrelevant here, since we only extract coefficients of positive total degree. Indeed, every k-efficient mask indicator is an alternating sequence of mask-runs and non-mask-runs, and every run length belongs to $\{1, \dots, k-1\}$. Now the left-hand side of Eq. (20) can be converted into the coefficient of $x^n y^{m-1}$ in

$$D(x, y) := \partial_x F(x, y) - \partial_y F(x, y).$$

Therefore it suffices to show that

$$[x^p y^q] D(x, y) \geq 0 \qquad \text{whenever } p \leq q.$$

We now prove this coefficientwise nonnegativity. Define

$$A_k(z) := 1 + P_k(z) = 1 + z + \cdots + z^{k-1}.$$

A direct differentiation of $F(x, y)$ gives

$$D(x, y) = \frac{A_k(y)^2 P_k'(x) - A_k(x)^2 P_k'(y)}{\big(1 - P_k(x)P_k(y)\big)^2}.$$

Write

$$B(x, y) := A_k(y)^2 P_k'(x) - A_k(x)^2 P_k'(y), \qquad K(x, y) := \frac{1}{\big(1 - P_k(x)P_k(y)\big)^2}.$$

Then

$$D(x, y) = B(x, y)K(x, y).$$

By the definition of a generating function,

$$[x^p y^q] D(x, y) \geq 0 \quad \Longleftrightarrow \quad [x^p y^q] \left(\sum_{i,j} \Big([x^i y^j] B(x, y) \Big) \Big([x^{p-i} y^{q-j}] K(x, y) \Big) \right) \geq 0.$$

Therefore, it suffices to show the right-hand side holds for any $p \leq q$. Let $c_{ij} := [x^i y^j] B(x, y)$ and $K_{u,v} := [x^u y^v] K(x, y)$. Our objective is then to show:

$$\sum_{i,j} c_{ij} K_{p-i, q-j} \geq 0, \qquad \text{whenever } p \leq q.$$

We first analyze the signs of the coefficients of B. Let

$$a_i := [z^i]P_k'(z), \qquad b_i := [z^i]A_k(z)^2.$$

Since

$$P_k'(z) = 1 + 2z + \cdots + (k-1)z^{k-2},$$

and

$$A_k(z)^2 = (1 + z + \cdots + z^{k-1})^2,$$

we have

$$\frac{a_i}{b_i} = \begin{cases} 1, & 0 \le i \le k-2, \\ 0, & k-1 \le i \le 2k-2, \end{cases}$$

with the convention that coefficients outside the support are zero. Hence $i \mapsto a_i/b_i$ is nonincreasing. Therefore, for $i < j$,

$$a_i b_j - b_i a_j = b_i b_j \left(\frac{a_i}{b_i} - \frac{a_j}{b_j} \right) \ge 0.$$

Then by definition of c_{ij},

$$c_{ij} \ge 0 \quad (i < j), \qquad c_{ji} = -c_{ij}, \qquad c_{ii} = 0. \tag{21}$$

Next we analyze K. For

$$K(x, y) = \frac{1}{\big(1 - P_k(x)P_k(y)\big)^2} = \sum_{r \ge 0} (r+1) P_k(x)^r P_k(y)^r,$$

let

$$f_r(s) := [z^s]P_k(z)^r \qquad \text{s.t. } K_{u,v} = \sum_{r \ge 0} (r+1) f_r(u) f_r(v).$$

For each fixed r, the sequence $s \mapsto f_r(s)$ is log-concave. Indeed, $P_k(z) = z(1 + z + \cdots + z^{k-2})$, and the nonzero coefficient sequence $(1, \ldots, 1)$ is log-concave; moreover, convolution preserves log-concavity for nonnegative sequences without internal zeros. Hence the coefficient sequence of $P_k(z)^r$ is log-concave. Equivalently, the ratio

$$\rho_s := \frac{f_r(s+1)}{f_r(s)}$$

is nonincreasing wherever both terms are positive. Therefore, for each fixed total degree S and for $u \le S/2$,

$$\frac{f_r(u+1) f_r(S-u-1)}{f_r(u) f_r(S-u)} = \frac{\rho_u}{\rho_{S-u-1}} \ge 1,$$

so the sequence

$$u \mapsto f_r(u) f_r(S-u)$$

is symmetric around $S/2$ and nondecreasing for $u \le S/2$. Summing over r with nonnegative weights $(r+1)$ preserves this property. Therefore,

$$K_{u,v} \ge K_{u',v'} \quad \text{whenever } u + v = u' + v' \text{ and } |u - v| \le |u' - v'|. \tag{22}$$

Now fix $p \le q$. Recall that

$$[x^p y^q]D = \sum_{i,j} c_{ij} K_{p-i,q-j}.$$

Pairing the terms (i, j) and (j, i) and using Eq. (21), we get

$$[x^p y^q]D = \sum_{i<j} c_{ij} \big(K_{p-i,q-j} - K_{p-j,q-i} \big),$$

where we use the convention that $K_{u,v} = 0$ whenever $u < 0$ or $v < 0$. We now show that each paired summand is nonnegative. Fix $i < j$.

First, suppose that $p < j$. Then $p - j < 0$, and hence

$$K_{p-j,q-i} = 0.$$

Since all coefficients of K are nonnegative, we have

$$K_{p-i,q-j} - K_{p-j,q-i} = K_{p-i,q-j} \geq 0,$$

where the inequality also covers the case in which $p - i < 0$ or $q - j < 0$, since then $K_{p-i,q-j} = 0$ by convention.

It remains to consider the case $p \geq j$. Since $p \leq q$ and $i < j$, all indices appearing below are nonnegative:

$$p - i, \quad q - j, \quad p - j, \quad q - i \geq 0.$$

The two pairs $(p - i, q - j)$ and $(p - j, q - i)$ have the same total degree:

$$(p - i) + (q - j) = (p - j) + (q - i).$$

Moreover, since $p \leq q$ and $i < j$,

$$\big|(q - j) - (p - i)\big| = \big|q - p - (j - i)\big| \leq q - p + (j - i) = \big|(q - i) - (p - j)\big|.$$

Thus $(p - i, q - j)$ is at least as close to the diagonal as $(p - j, q - i)$. By Eq. (22),

$$K_{p-i,q-j} \geq K_{p-j,q-i}.$$

Therefore, in both cases,

$$K_{p-i,q-j} - K_{p-j,q-i} \geq 0.$$

Since also $c_{ij} \geq 0$ for $i < j$, every paired summand is nonnegative. Hence

$$[x^p y^q]D \geq 0 \qquad \text{whenever } p \leq q.$$

Finally, take $p = n$ and $q = m - 1$. Since $n < m$, we have $n \leq m - 1$, so

$$[x^n y^{m-1}]D \geq 0.$$

Equivalently,

$$(n + 1)N(n + 1, m - 1) - mN(n, m) \geq 0.$$

As shown above, this is exactly

$$q_{L,k}(n) \leq q_{L,k}(n + 1).$$

This proves the monotonicity for every $n < L/2$, and the lemma follows. □

We now translate Lemma B.5 into a statement about time sampling. Recall that

$$\mathbb{P}\Big(\mathbf{M} \in \mathcal{E}_k^c \mid t \sim \pi, M^i \overset{\text{i.i.d.}}{\sim} \text{Bernoulli}(t)\Big) = \sum_{n=0}^{L} w_\pi(n) q_{L,k}(n). \tag{23}$$

where

$$\begin{aligned} w_\pi(n) &= \mathbb{P}\left(\sum_{i=1}^{L} M^i = n \,\middle|\, t \sim \pi,\ M^i \overset{\text{i.i.d.}}{\sim} \text{Bernoulli}(t)\right) \\ &= \binom{L}{n} \int_{[0,1]} t^n (1 - t)^{L-n} \, d\pi(t) \end{aligned} \tag{24}$$

Substituting Eq. (24) into Eq. (23) gives:

$$\begin{aligned} \mathbb{P}\Big(\mathbf{M} \in \mathcal{E}_k^c \mid t \sim \pi, M^i \overset{\text{i.i.d.}}{\sim} \text{Bernoulli}(t)\Big) &= \sum_{n=0}^{L} w_\pi(n) q_{L,k}(n) \\ &= \sum_{n=0}^{L} \binom{L}{n} \int_{[0,1]} t^n (1 - t)^{L-n} q_{L,k}(n) \, d\pi(t) \\ &= \int_{[0,1]} \sum_{n=0}^{L} \binom{L}{n} t^n (1 - t)^{L-n} q_{L,k}(n) \, d\pi(t) \end{aligned} \tag{25}$$

Define

$$R_{L,k}(t) := \mathbb{P}\left(\mathbf{M} \in \mathcal{E}_k^c \;\middle|\; M^i \overset{\text{i.i.d.}}{\sim} \text{Bernoulli}(t)\right).$$

Conditioning on the number of masks gives

$$R_{L,k}(t) = \sum_{n=0}^{L} \binom{L}{n} t^n (1-t)^{L-n} q_{L,k}(n). \tag{26}$$

Substituting Eq. (26) into Eq. (25) then gives:

$$\mathbb{P}\Big(\mathbf{M} \in \mathcal{E}_k^c \mid t \sim \pi, M^i \overset{\text{i.i.d.}}{\sim} \text{Bernoulli}(t)\Big) = \int_{[0,1]} R_{L,k}(t) d\pi(t). \tag{27}$$

Now our goal is to prove $R_{L,k}(t)$ has property of central monotonicity, and then prove Proposition 4.4.

Lemma B.6 (Central monotonicity in time)**.** *The function $R_{L,k}(t)$ is symmetric around $1/2$ and nondecreasing on $[0, 1/2]$. That is,*

$$R_{L,k}(t) = R_{L,k}(1-t), \qquad R'_{L,k}(t) \geq 0 \quad \textit{for } 0 \leq t \leq \frac{1}{2}.$$

Moreover, if $2 \leq k \leq L$, then $R_{L,k}$ is strictly increasing on $[0, 1/2]$.

Proof. The symmetry follows from the symmetry of $q_{L,k}(n)$:

$$q_{L,k}(n) = q_{L,k}(L-n).$$

To prove monotonicity, differentiate Eq. (26). The derivative of $R_{L,k}(t)$ is

$$R'_{L,k}(t) = L \sum_{n=0}^{L-1} \binom{L-1}{n} \big(q_{L,k}(n+1) - q_{L,k}(n)\big) t^n (1-t)^{L-1-n}.$$

Let $d_n := q_{L,k}(n+1) - q_{L,k}(n)$. By Lemma B.5, $d_n \geq 0$ for $n < (L-1)/2$, and by symmetry of $q_{L,k}$, $d_{L-1-n} = -d_n$. Now pair the n-th and $(L-1-n)$-th terms in the derivative. For $n < (L-1)/2$, the paired contribution is

$$L\binom{L-1}{n} d_n \Big[t^n (1-t)^{L-1-n} - t^{L-1-n}(1-t)^n\Big].$$

When $0 \leq t \leq 1/2$, we have $1 - t \geq t$, so every paired contribution is nonnegative. Hence $R'_{L,k}(t) \geq 0$ on $[0, 1/2]$.

It remains to prove strict monotonicity. When $2 \leq k \leq L$, we have $R_{L,k}(0) = 0$, because $t = 0$ gives the all-zero mask indicator, which contains $L \geq k$ consecutive non-masks and therefore lies in $\mathcal{E}_k$. On the other hand, $R_{L,k}(1/2) > 0$, since the alternating mask indicator $0101\cdots$ is k-efficient for every $k \geq 2$ and occurs with positive probability under i.i.d. Bernoulli$(1/2)$ sampling. Thus $R_{L,k}$ is not constant. Since $R'_{L,k}$ is a polynomial and $R'_{L,k} \geq 0$ on $[0, 1/2]$, if $R_{L,k}$ were not strictly increasing, then it would be constant on some nonempty interval, so $R'_{L,k}$ would vanish on that interval and hence vanish identically. This would contradict $R_{L,k}(0) < R_{L,k}(1/2)$. Therefore $R_{L,k}$ is strictly increasing on $[0, 1/2]$. □

We can now prove Proposition 4.4. Recall that Proposition 4.4 is as follows:

Proposition 4.4. *For any L, $2 \leq k \leq L$, and any bell-shaped time distribution π, the following statement always holds:*

$$\mathbb{P}\Big(\mathbf{M} \notin \mathcal{E}_k \mid t \sim \text{U}([0,1]), M^i \sim \text{Bernoulli}(t)\Big) < \mathbb{P}\Big(\mathbf{M} \notin \mathcal{E}_k \mid t \sim \pi, M^i \sim \text{Bernoulli}(t)\Big).$$

Proof. For brevity, write $R(t) := R_{L,k}(t)$. By Eq. (27), for any probability measure μ on $[0, 1]$,

$$\mathbb{P}\left(\mathbf{M} \in \mathcal{E}_k^c \;\middle|\; t \sim \mu,\ M_i \overset{\text{i.i.d.}}{\sim} \text{Bernoulli}(t)\right) = \int_0^1 R(t)\, d\mu(t).$$

Thus it suffices to compare $\int R(t)\,d\pi(t)$ with $\int R(t)\,dt$. Since π may be atomic or assign zero mass to parts of $[0,1]$, we avoid any pointwise density comparison and instead use an integral over the central intervals $[a, 1-a]$. By Lemma B.6, $R(t) = R(1-t)$ and R is strictly increasing on $[0, 1/2]$. In particular, $R'(a) \geq 0$ on $[0, 1/2]$. For every $t \in [0,1]$, we have

$$R(t) = R(0) + \int_0^{1/2} R'(a)\mathbf{1}\{t \in [a, 1-a]\}\,da.$$

Indeed, if $t \leq 1/2$, the right-hand side is $R(0) + \int_0^t R'(a)\,da = R(t)$; if $t \geq 1/2$, it is $R(0) + \int_0^{1-t} R'(a)\,da = R(1-t) = R(t)$.

Integrating this identity with respect to π gives

$$\int_0^1 R(t)\,d\pi(t) = R(0) + \int_0^{1/2} R'(a)\mathbb{P}_{t\sim\pi}\left(t \in [a, 1-a]\right)\,da.$$

Similarly, for the uniform distribution,

$$\int_0^1 R(t)\,dt = R(0) + \int_0^{1/2} R'(a)(1-2a)\,da,$$

because the interval $[a, 1-a]$ has length $1-2a$.

Therefore,

$$\int_0^1 R(t)\,d\pi(t) - \int_0^1 R(t)\,dt = \int_0^{1/2} R'(a)\left(\mathbb{P}_{t\sim\pi}\left(t \in [a, 1-a]\right) - (1-2a)\right)\,da.$$

Since π is bell-shaped,

$$\mathbb{P}_{t\sim\pi}\left(t \in [a, 1-a]\right) \geq \mathbb{P}_{t\sim \mathrm{U}([0,1])}\left(t \in [a, 1-a]\right) = 1-2a \qquad \text{for all } a \in [0, 1/2),$$

and since $R'(a) \geq 0$, the difference above is nonnegative.

It remains to show that the inequality is strict. By the definition of a bell-shaped distribution, there exists $a_0 \in (0, 1/2)$ such that

$$\mathbb{P}_{t\sim\pi}\left(t \in [a_0, 1-a_0]\right) > 1 - 2a_0.$$

Let

$$\eta := \mathbb{P}_{t\sim\pi}\left(t \in [a_0, 1-a_0]\right) - (1-2a_0) > 0.$$

Choose $0 < \varepsilon < \min\{a_0, \eta/4\}$ and set $I = [a_0 - \varepsilon, a_0]$. For every $a \in I$, we have $[a_0, 1-a_0] \subset [a, 1-a]$, so

$$\mathbb{P}_{t\sim\pi}\left(t \in [a, 1-a]\right) - (1-2a) \geq \mathbb{P}_{t\sim\pi}\left(t \in [a_0, 1-a_0]\right) - (1-2a) = \eta - 2(a_0 - a) > \frac{\eta}{2}.$$

Hence the bracketed term is strictly positive throughout the interval I.

Since R is strictly increasing on $[0, 1/2]$,

$$\int_I R'(a)\,da = R(a_0) - R(a_0 - \varepsilon) > 0.$$

Therefore,

$$\int_0^1 R(t)\,d\pi(t) - \int_0^1 R(t)\,dt \geq \int_I R'(a)\left(\mathbb{P}_{t\sim\pi}\left(t \in [a, 1-a]\right) - (1-2a)\right)\,da > 0.$$

Consequently,

$$\mathbb{P}\left(\mathbf{M} \in \mathcal{E}_k^c \;\middle|\; t \sim \pi,\ M_i \overset{\text{i.i.d.}}{\sim} \text{Bernoulli}(t)\right) > \mathbb{P}\left(\mathbf{M} \in \mathcal{E}_k^c \;\middle|\; t \sim U([0,1]),\ M_i \overset{\text{i.i.d.}}{\sim} \text{Bernoulli}(t)\right).$$

Taking complements gives

$$\mathbb{P}\left(\mathbf{M} \in \mathcal{E}_k \;\middle|\; t \sim \pi,\ M_i \overset{\text{i.i.d.}}{\sim} \text{Bernoulli}(t)\right) < \mathbb{P}\left(\mathbf{M} \in \mathcal{E}_k \;\middle|\; t \sim U([0,1]),\ M_i \overset{\text{i.i.d.}}{\sim} \text{Bernoulli}(t)\right).$$

This concludes the proof. □

Remark. In measure-theoretic notation, the probability mass assigned by a probability distribution π to an interval $I \subseteq [0,1]$ is conventionally written as $\pi(I)$. To make this notation more accessible to readers less familiar with this convention, we instead write the same quantity as $\mathbb{P}_{t\sim\pi}(t \in I)$ throughout the main text and proofs.

C Omitted Experimental Details and Results

C.1 Omitted Experimental Details

Experimental setup used in LM1B experiments. We detokenize the One Billion Words (LM1B) dataset following prior works (Lou et al., 2024; Sahoo et al., 2024). We tokenize LM1B using the BERT-BASE-UNCASED tokenizer (He et al., 2023), and concatenate sequences to a fixed length of 128 using sentence packing (Raffel et al., 2020). We parameterize the diffusion backbone (Lou et al., 2024; Sahoo et al., 2024) using a modified diffusion transformer (DiT) architecture (Peebles & Xie, 2023) with 12 layers, hidden size 768, and 12 attention heads. We train using the AdamW optimizer with a batch size of 512, and adopt a linear warmup schedule from 0 to $3\mathrm{e}{-4}$ over 2,500 steps (Sahoo et al., 2024; Lou et al., 2024; Sahoo et al., 2025). Training was conducted on $2\times$H200 GPUs for both LM1B with and without sentence packing and consumed $\sim$100 hours to reach 1M steps.

Experimental setup used in OWT experiments. For OpenWebText (OWT), we use the GPT-2 tokenizer (Radford et al., 2019) following prior works (Lou et al., 2024; Sahoo et al., 2024) and pack sequences to a context length of 1,024, inserting an `eos` token between consecutive documents. As OWT does not provide an official validation split, we reserve the last 100K documents for validation. We use the same DiT architecture specified in above LM1B setup. Following prior work (Shi et al., 2024), we train using the AdamW optimizer with a batch size of 512, and adopt a cosine annealing with weight decay scheduler with warmup from 0 to $3\mathrm{e}{-4}$ over 2,500 steps. Training was conducted on $4\times$H100 GPUs for OWT and consumed $\sim$300 hours to reach 1M steps.

Default setting for analysis on MDM training inefficiency (Sec. 3), training dynamics of MDM with various time distribution (Sec. 5.1), and analysis on other alternatives (Sec. 5.4). All experiments in these sections use LM1B with sentence packing as the default setting, and follow the aforementioned setup for LM1B experiments. For ARM, we utilize the same architecture for MDMs, and the only difference is that ARM utilizes causal attention. For σ-ARM, we just utilize same architecture of ARM, yet just permute the data and RoPE (Su et al., 2024) angle with randomly selected but fixed σ. For AoARM, we do not directly adopt the architecture of Xue et al. (Xue et al., 2025), since it does not use RoPE and injects source and target positions through sinusoidal embeddings and AdaLN, respectively. To keep the comparison compatible with our RoPE-based backbone, we implement a simple AoARM variant: source-position RoPE is applied in the first half of the layers, and target-position RoPE in the second half. Since this experiment only aims to test whether AoARM can outperform σ-ARM, this simple implementation is sufficient for our analysis.

Experimental setup used in pretraining benchmarks (Sec. 5.2). The training setup inherits all the aforementioned setups for LM1B and OWT, respectively. We report both generative and zero-shot perplexity. Generative perplexity is computed using a GPT-2 Large autoregressive model, following prior works (Sahoo et al., 2024; Lou et al., 2024; Sahoo et al., 2025), while zero-shot perplexity is evaluated on the corresponding validation splits of each dataset. See Appendix C.3 for details and results of zero-shot perplexity and generative perplexity.

Experimental setup used in continual pretraining (Sec. 5.3). We use GPT-2 Large (36 layers, 20 attention heads, hidden size 1280; 774M parameters) as the backbone and train a discrete diffusion language model with full-parameter updates. Continual pretraining (CPT) is performed on pre-tokenized FineWeb-Edu sequences with a global batch size of 256. We use a learning rate of $3\mathrm{e}{-4}$ with 2K warmup steps and 10K annealing steps. For downstream supervised fine-tuning (SFT), we construct a 210K-example instruction-style dataset by combining the training splits of the following benchmarks: LAMBADA (Paperno et al., 2016), HellaSwag (Zellers et al., 2019), TriviaQA (Joshi et al., 2017), OpenBookQA (Mihaylov et al., 2018), PIQA (Bisk et al., 2020), Social IQA (Sap et al., 2019), and WinoGrande (Sakaguchi et al., 2021). This composition follows prior instruction tuning paradigm (Sanh et al., 2021), where downstream benchmarks are repurposed as supervised training data to induce general reasoning capabilities. Each example is formatted as a prompt ending with `Answer:` followed by the target answer string. We fine-tune with a global batch size of 128, 64 diffusion steps, and a single annealing step, using a cosine learning rate schedule with 500 warmup steps in `bf16` precision and full-parameter updates. All downstream experiments run for 10K SFT steps. For instruction-following evaluation, we further fine-tune on 103K cleaned SHAREGPT-GPT4 conversations, following the SFT protocol of Nie *et al*. (Nie et al., 2025a). Starting from 2K-step downstream SFT checkpoints, we train for an additional 10K steps with a global batch size of 128

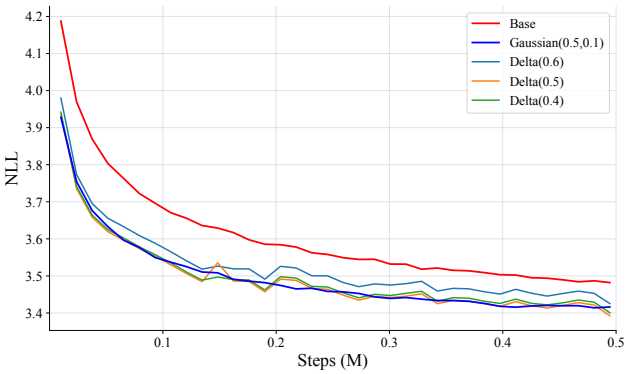


Figure 8: Additional ablation on Dirac-delta time sampling. We compare different choices of the delta location μ against the standard MDM baseline and the default Gaussian$(0.5, 0.1)$ time sampling.

and learning rate $1\mathrm{e}{-4}$. We use sequence length 1024, 64 diffusion steps, and a single annealing step. Training was conducted on $4\times$B200 GPUs and consumed $\sim$50 hours to reach 100K steps.

To briefly explain the CPT proposed by Gong *et al*. (Gong et al., 2025) and used in our experiments, Gong *et al*. adapt pretrained autoregressive language models into diffusion language models via continued pretraining, rather than training diffusion models from scratch. Their recipe bridges the gap between ARM and diffusion modeling by converting causal attention into bidirectional denoising behavior and preserving the ARM-style shift operation during adaptation. Importantly, this CPT procedure does not introduce a new diffusion objective; it directly optimizes the standard MDM loss $\mathcal{L}_{\mathrm{MDM}}$. In this CPT setting, applying our bell-shaped time sampling simply amounts to replacing this loss with $\hat{\mathcal{L}}_{\pi}$, while keeping the rest of CPT recipe proposed by Gong *et al*. unchanged. For supervised fine-tuning, following LLaDA (Nie et al., 2025b), we keep the condition tokens, such as the instruction and prompt, fixed without masking them, and mask only the response tokens for denoising under the standard MDM loss $\mathcal{L}_{\mathrm{MDM}}$. Similarly, applying our bell-shaped time sampling in this SFT setting simply amounts to replacing $\mathcal{L}_{\mathrm{MDM}}$ with $\hat{\mathcal{L}}_{\pi}$, while keeping the conditioning scheme unchanged.

C.2 Additional Ablation on Bell-shaped Time Sampling

We provide additional hyperparameter ablations for the time sampling distributions introduced in Table 3. All experiments use the same LM1B sentence-packed setting as in Sec. 5.1, and we report validation NLL during training. In all figures, "Base" denotes the conventional MDM objective, while Gaussian$(0.5, 0.1)$ denotes the default bell-shaped time sampling distribution used in the main experiments.

Dirac-delta time sampling. Fig. 8 compares Dirac-delta distributions $\pi = \delta_{\mu}$ with different choices of μ. All delta-based variants substantially outperform the standard MDM baseline, confirming that concentrating training on a narrow range of middle-context corruption levels is beneficial. Among them, $\delta_{0.5}$ and $\delta_{0.4}$ show strong performance, whereas $\delta_{0.6}$ is consistently worse. Since a larger t corresponds to a lower-context regime, this result is consistent with the observation in Sec. 5.1 that shifting the sampling distribution toward $\mu = 0.6$ degrades training dynamics. This result might indicate that the training inefficiency of low-context regime is larger than that of high-context regime. However, although delta sampling can strongly emphasize a specific context level, it removes diversity across corruption levels, which makes it less robust than the default Gaussian sampling.

Gaussian time sampling. Fig. 9 further varies the mean and standard deviation of the truncated Gaussian distribution. The results show that Gaussian time sampling is broadly effective: all Gaussian variants improve over the base MDM. Nevertheless, the mean is important. Distributions centered at $\mu = 0.5$ or slightly shifted to $\mu = 0.4$ remain strong, whereas shifting the mean toward $\mu = 0.6$ yields worse training dynamics, especially when the variance is small. Increasing the standard deviation from $\sigma = 0.1$ to $\sigma = 0.2$ partially mitigates this degradation, since the distribution assigns more probability mass back to the middle-context region. These results support our default choice

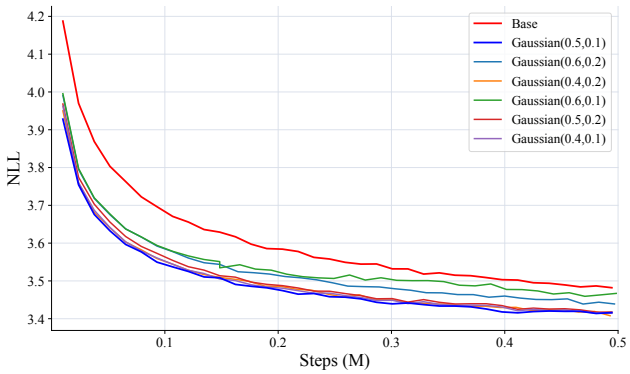


Figure 9: Additional ablation on truncated Gaussian time sampling. We vary the mean and standard deviation of the truncated Gaussian distribution and compare them with the standard MDM baseline and the default Gaussian$(0.5, 0.1)$.

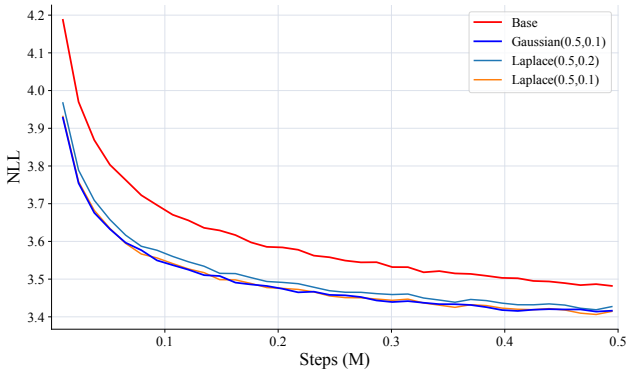


Figure 10: Additional ablation on truncated Laplace time sampling. We compare Laplace distributions with different scale parameters against the standard MDM baseline and the default Gaussian$(0.5, 0.1)$.

of Gaussian$(0.5, 0.1)$: it is centered at the most balanced context level while keeping the sampling distribution sufficiently concentrated.

Laplace time sampling. Fig. 10 evaluates truncated Laplace distributions centered at $\mu = 0.5$ with different scale parameters. Both Laplace$(0.5, 0.1)$ and Laplace$(0.5, 0.2)$ outperform the standard MDM baseline, showing that the benefit of bell-shaped time sampling is not specific to the Gaussian family. The narrower Laplace distribution performs slightly better, while the larger-scale variant remains effective but is somewhat less competitive because its heavier tails allocate more probability mass to the low- and high-context regions. Together with the Gaussian ablation, this suggests that the crucial factor is not the exact parametric form of $\pi(t)$, but whether the distribution keeps most of its probability mass around the middle-context region.

C.3 Zero-shot Perplexity and Generative Perplexity of Models Trained on OpenWebText

We provide zero-shot perplexity and generative perplexity for models trained on OWT in Table 7 and 8, respectively. Following prior works (Sahoo et al., 2024; Hong et al., 2026b), we utilize 7 zero-shot perplexity benchmarks, including PTB (Marcus et al., 1993), Wikitext (Merity et al., 2017), LM1B, Lambada (Paperno et al., 2016), AG News (Zhang et al., 2015), Pubmed (Cohan et al., 2018), and Arxiv (Cohan et al., 2018). The generative PPL is computed by a pre-trained GPT-2 Large on 100 generated samples with length of 1024. As shown in the tables, the model trained with Gaussian time sampling consistently outperforms the baseline.

Table 7: Zero-shot perplexities (↓) of models trained for 1M steps on OpenWebText (mean ± std over 5 seeds).

Training	Benchmark						
	PTB	Wikitext	LM1B	Lambada	AG News	Pubmed	Arxiv
Base	115.99 ± 5.60	37.76 ± 0.94	72.06 ± 0.45	48.18 ± 1.18	68.31 ± 1.19	43.15 ± 0.13	$\mathbf{37.85 \pm 0.14}$
Gaussian	$\mathbf{98.16 \pm 4.29}$	$\mathbf{34.51 \pm 0.80}$	$\mathbf{67.71 \pm 0.38}$	$\mathbf{46.39 \pm 1.05}$	$\mathbf{61.74 \pm 1.26}$	$\mathbf{41.43 \pm 0.11}$	37.88 ± 0.15

Table 8: Generative PPL (↓) of models trained for 1M steps on OpenWebText (mean ± std over 5 seeds). NFE denotes the number of function evaluations used during generation.

NFE	512	1024	2048
Base	56.51 ± 1.12	42.81 ± 0.73	36.04 ± 1.53
Gaussian	$\mathbf{55.50 \pm 1.05}$	$\mathbf{41.55 \pm 2.09}$	$\mathbf{33.89 \pm 0.91}$

C.4 Other Explored Training Techniques in Detail

In this section, we further explain the details of the other training techniques briefly described in Sec. 5.4. Recall that, for a strictly decreasing scheduler α_t, the continuous-time NELBO of MDM can be written as

$$\mathcal{L}_{\mathrm{MDM}} = \int_0^1 \frac{-\alpha_t'}{1-\alpha_t}\, \mathbb{E}_{\mathbf{x}\sim p_{\mathrm{data}},\, \mathbf{x}_t \sim q_{t|0}} \Big[\sum_{i: x_t^i = m} -\log p_\theta(x^i \mid \mathbf{x}_t, t) \Big]\, dt. \tag{28}$$

As in the main experimental results, we use a truncated Gaussian density for the middle-biased alternatives in Sec. 5.4. Define

$$f_{\mathrm{TG}}(t) = \frac{1}{\sigma Z}\, \varphi\Big(\frac{t-\mu}{\sigma}\Big), \qquad t \in [\ell, h], \tag{29}$$

where Z is the normalization constant. We regard $f_{\mathrm{TG}}(a)$ as zero outside $[\ell, h]$. Note that we utilized $\ell = 0, h = 1, \mu = 0.5$ and $\sigma = 0.1$ for all experiments in Sec. 5.4.

Middle-flat noise scheduler α_t^{MF}. Prior works (Sahoo et al., 2024; Shi et al., 2024) show that the NELBO of MDM is invariant to the specific choice of α_t. In this regard, a natural alternative is to modify the scheduler itself so that the model encounters middle-context samples more often while still optimizing the theoretical NELBO. To this end, define the truncated-normal CDF

$$F_{\mathrm{TG}}(t) := \int_\ell^t f_{\mathrm{TG}}(u)\, du, \tag{30}$$

and choose the scheduler as

$$\alpha_t^{\mathrm{MF}} = F_{\mathrm{TG}}^{-1}(1-t), \qquad t \sim \mathrm{Uniform}(0,1). \tag{31}$$

Recall that the forward process of MDM is defined as follows:

$$q(\mathbf{x}_t^i \mid \mathbf{x}) = q_{t|0}(\cdot \mid x^i) = \mathrm{Cat}(\alpha_t \mathbf{e}_{x^i} + (1-\alpha_t)\mathbf{e}_m)\,,$$

Thus, the model indeed sees many more middle-context samples if we set α_t as α_t^{MF} rather than the original linear scheduler of MDM, *i.e.*, $1-t$, as visualized in Fig. 11.

However, this does not give the same objective as bell-shaped time sampling and is equal to the theoretical NELBO of MDM (Eq. (1)). The reason is that the NELBO already contains the scheduler-dependent prefactor $-\alpha_t'/(1-\alpha_t)$. Formally,

$$\frac{d\alpha_t^{\mathrm{MF}}}{dt} = -\frac{1}{f_{\mathrm{TG}}(\alpha_t^{\mathrm{MF}})}, \tag{32}$$

and therefore

$$\frac{-\alpha_t^{\mathrm{MF}\,\prime}}{1-\alpha_t^{\mathrm{MF}}} = \frac{1}{(1-\alpha_t^{\mathrm{MF}}) f_{\mathrm{TG}}(\alpha_t^{\mathrm{MF}})}. \tag{33}$$

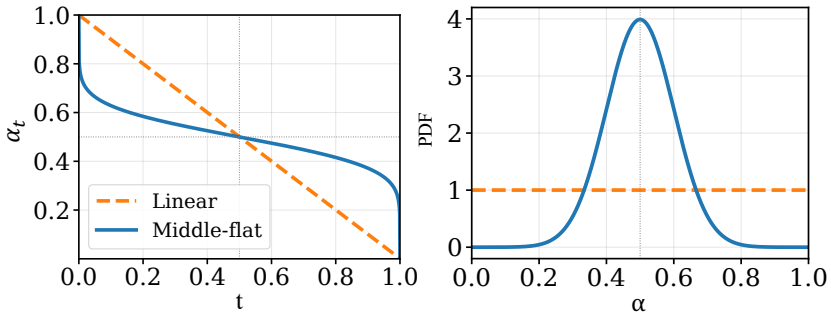


Figure 11: *Left:* the middle-flat scheduler $\alpha_t^{\mathrm{MF}} = F_{\mathrm{TG}}^{-1}(1-t)$ with $\ell = 0, h = 1, \mu = 0.5$ and $\sigma = 0.1$ is flatter around $\alpha = 0.5$ than the linear scheduler, so a uniform draw of t produces many more middle-context corruption levels. *Right:* under $t \sim \mathrm{Uniform}(0,1)$, the induced marginal distribution of α_t^{MF} matches the target truncated-Gaussian density $f_{\mathrm{TG}}(a)$.

Since $f_{\mathrm{TG}}(a)$ is largest around the middle region, the NELBO prefactor in Eq. (33) becomes smallest near the middle and larger away from it. In other words, the apparent oversampling of middle-context scheduler values is compensated by a smaller NELBO prefactor.

Therefore, the practical difference between bell-shaped time sampling and the middle-flat scheduler is twofold: 1) as discussed above, the middle-flat scheduler does not modify the learning signal itself, and thus cannot concentrate training on the middle-context region. 2) The scaling factor $-\alpha_t^{\mathrm{MF}\prime}/1-\alpha_t^{\mathrm{MF}}$ of the middle-flat scheduler becomes much larger near $t=0$ or $t=1$ than that of $\alpha_t = 1-t$, increasing the variance of the mini-batch estimator itself. For these two reasons, we conjecture that the middle-flat scheduler performs worse than the conventional MDM loss.

Importance sampling. Another possible alternative is to keep the canonical absorbing-state scheduler $\alpha_t = 1-t$, but to *reweight the loss* so that the resulting objective matches bell-shaped time sampling in expectation. Concretely, we keep drawing t uniformly, and use the importance ratio $f_{\mathrm{TG}}(t)$, then the reweighted objective then becomes

$$\mathcal{L}_{\mathrm{IS}} = \int_0^1 \frac{f_{\mathrm{TG}}(t)}{t} \mathbb{E}_{\mathbf{x}\sim p_{\mathrm{data}},\, \mathbf{x}_t \sim q_{t|0}} \Big[- \sum_{i: x_t^i = m} \log p_\theta(x^i \mid \mathbf{x}_t, t) \Big] dt$$

where $-\alpha_t'/(1-\alpha_t) = 1/t$ for $\alpha_t = 1-t$. Here, if we drop t in the denominator, $\mathcal{L}_{\mathrm{IS}}$ matches the *same expectation* as bell-shaped time sampling. The result shown in Fig. 7 corresponds to the setting where we drop t in the denominator, but we observed similar trends even when we do not drop t in the denominator.

The practical difference between importance sampling and ours is that importance sampling does *not* alter which corruption levels are actually drawn in a minibatch: samples are still generated from the uniform schedule, and only their scalar losses are reweighted afterward. By contrast, bell-shaped time sampling directly draws middle-context samples more frequently. Hence, although importance sampling and bell-shaped time sampling optimize the same objective in expectation, their optimization dynamics can differ substantially in practice; the former may suffer from a noisier gradient estimator because informative middle-context samples are not physically sampled more often.

C.5 Stratified Sampling

Recall that our bell-shaped time sampling loss is given as follows:

$$\hat{\mathcal{L}}_\pi = \mathbb{E}_{\mathbf{x}\sim p_{\mathrm{data}},\, t\sim\pi,\, \mathbf{x}_t\sim q_{t|0}(\cdot|\mathbf{x})} \Big[- \textstyle\sum_{i:x_t^i=m} \log p_\theta(x^i \mid \mathbf{x}_t, t) \Big]. \tag{34}$$

A practical issue arises when π is sharply concentrated around the middle-context region. If each t_i in a mini-batch is drawn independently from π, some batches can contain too many samples from the same context regime, which increases the variance of the gradient estimator and may destabilize

optimization. To mitigate this, we utilize *stratified time sampling*, which is widely used in prior works (Sahoo et al., 2024; 2025). Intuitively, stratified sampling divides the probability mass of the target time distribution into B equal-probability strata and draws one sample from each stratum. This makes each mini-batch cover the target time distribution more evenly than i.i.d. sampling, while preserving the expectation of the batch estimator.

To our knowledge, no prior work has applied stratified sampling to bell-shaped time sampling distributions, so we describe it here in detail. Let F_π denote the CDF of π. For a batch of size B, instead of i.i.d. sampling $t_i \sim \pi$, we first stratify the unit interval:

$$\hat{u}_i = \tfrac{i-1}{B}, \qquad \epsilon_i \sim \mathrm{U}\big([0, \tfrac{1}{B})\big), \qquad u_i = \hat{u}_i + \epsilon_i,$$

for $i \in [B]$, and then map through the inverse CDF, $t_i^{\mathrm{strat}} = F_\pi^{-1}(u_i)$. This preserves the expectation of the batch estimator while reducing its variance compared with naive sampling. Note that all experiments in our paper utilized stratified sampling.

D Limitations

We showed that bell-shaped time sampling is effective not only for from-scratch training, but also for billion-scale CPT with ARM initialization. In Sec. 5.1 and Appendix C.2, we further analyzed the training dynamics of a wide range of bell-shaped distributions on LM1B. Nevertheless, the optimal bell-shaped time distribution may depend on dataset characteristics and model scale, and a more systematic analysis of this dependence remains an important direction. Still, our experiments suggest that a Gaussian distribution with mean 0.5 and standard deviation 0.1 works robustly across most settings. Another promising future direction is to investigate whether bell-shaped time sampling can also benefit post-training methods, such as reinforcement learning.

E Qualitative Examples

In this section, we provide qualitative generation examples from the instruction-following models evaluated in Sec. 5.3. Specifically, we compare the outputs of the base MDM and the bell-shaped time sampling trained MDM after applying the same ShareGPT supervised fine-tuning procedure. These qualitative examples indicate that, after the same number of training steps, the base MDM has not yet acquired the ability to perform conditional generation, whereas our MDM trained with bell-shaped time sampling already demonstrates this ability.

(a) Salary-negotiation email generation

User Prompt

You need to write an email to negotiate your salary.

Base MDM

```
However, to understand how you need
to negotiate your salary, you need to
understand the following steps.

To negotiate your salary, you need to
need to understand the following of
steps steps:

1.  negotiate a fair salary:
negotiating a fair salary is a salary
that is fair to the parties and the b.
2.  negotiate a fair salary that is a
and the a a salary that is fair to the
parties a and the b.
3.  negotiate a fair salary that is fair
to the parties a and the b.
4.  negotiate fair salary5.  negotiate
the fair salary that is fair to the
parties and the b...
```

Gaussian time trained MDM

```
Sure, here's an example of an email that
you could use to negotiate your salary:

Subject:  Request for a Salary Raise

Dear [Name],

I am writing to request a salary raise.
I understand that you are a full-time
employee and you are entitled to a
salary salary.  However, I would like
to request that you consider a salary
raise.

I understand that you are a valuable
employee and I would like to ensure
that you are able to achieve your full
potential.  I would like to schedule an
effort to negotiate your salary.

Thank you for your consideration...
```

(b) Professional feedback email generation

User Prompt

Draft a professional email seeking your supervisor's feedback on the 'Quarterly Financial Report' you prepared. Ask specifically about the data analysis, presentation style, and the clarity of conclusions drawn. Keep the email short and to the point.

Base MDM

```
Additionally, please encourage the
reader to read the rest of the report.

Now, here's the revised version of the
email with the changes:

Subject:  Your supervisor's feedback on
the requested:
, the Quarterly Financial Report'

Subject:  Feedback on the 'Quarterly
Financial Report'
To:
::  Your Supervisor's Name,
...
[Your Supervisor's Name]
Best regards,
[Your Name,
Your Supervisor Name]...
```

Gaussian time trained MDM

```
Subject:

Dear ,

I am writing to request your feedback
on the 'Quarterly Financial Report'
prepared for the upcoming quarter.  I
would appreciate your feedback on the
data analysis, presentation style, and
clarity of the conclusions drawn.

I would also appreciate your feedback
on the presentation of the data and the
overall clarity of the financial report.

Thank you for your time and
consideration...
```

Figure 12: **Qualitative comparisons on professional email generation tasks.** Across both email-writing tasks, the baseline MDM frequently degenerates into malformed formatting and repetitive text fragments. In contrast, the Gaussian time trained MDM generates substantially more coherent responses with recognizable email structure, appropriate formatting, and improved instruction following, despite minor repetition artifacts.

(a) Method extraction from scientific abstracts

User Prompt

Extract the method that has been used in the research from the abstract of the paper. Give the exact sentences from the abstract that discuss the method.

Input Abstract

We present a state-of-the-art speech recognition system developed using end-to-end deep learning. Our architecture is significantly simpler than traditional speech systems, which rely on laboriously engineered processing pipelines; these traditional systems also tend to perform poorly when used in noisy environments. In contrast, our system does not need hand-designed components to model background noise, reverberation, or speaker variation, but instead directly learns a function that is robust to such effects. We do not need a phoneme dictionary, nor even the concept of a "phoneme." Key to our approach is a well-optimized RNN training system that uses multiple GPUs, as well as a set of novel data synthesis techniques that allow us to efficiently obtain a large amount of varied data for training.

Gaussian time trained MDM

```
``We present a state-of-the-art speech
recognition system developed using
end-to-end deep learning.''

``In contrast, our system does not need
hand-designed components to model
background noise, reverberation, or
speaker variation, but instead directly
learns a function that is robust to such
effects.''

``Key to our approach is a
well-optimized RNN training system that
uses multiple GPUs, as well as a set
of novel data synthesis techniques that
allow us to efficiently obtain a large
amount of varied data for training.''
```

Base MDM

```
The paper discusses a speech recognition
method developed using end-to-end
deep learning that is simpler than
traditional speech systems.  The.  The.
The..  The...  The.  processing noisy
environments..  The.  The.  The The..
The.  The The.....
```

(b) Open-ended opinion generation

User Prompt

What do you think about ChatGPT?

Gaussian time trained MDM

```
I am impressed with ChatGPT. ChatGPT is
a powerful AI model that can handle a
wide range of tasks, including language
translation, text generation, and
sentiment analysis.  It is also able
to learn from large amounts of data and
adapt to new situations.

I am particularly impressed with
ChatGPT's ability to learn quickly,
adapt to new tasks, and provide
high-quality responses.

ChatGPT is a large language model
designed to generate text based on user
input and assist with a wide variety of
applications...
```

Base MDM

```
* Do you think ChatGPT isPT?
* Chat?
* Do you think ChatGPT is a GPT?

* Do you think ChatGPT is a Chat?
* Do you think think ChatGPT is a Chat?
* Do you think ChatGPT is a GPT?
* Do you think ChatGPT is ChatGPT is a
GPT?
* Do you think ChatGPT is a GPT?
* Do Do you think ChatGPT is a GPT?...
```

Figure 13: **Qualitative comparisons between the baseline MDM and Gaussian time trained MDM.** Across both structured extraction and open-ended generation tasks, the baseline model frequently degenerates into repetitive or malformed text, whereas the Gaussian-trained model produces substantially more coherent and instruction-following responses.