Beyond Masked and Unmasked: Discrete Diffusion Models via Partial Masking

Chen-Hao Chao[1], Wei-Fang Sun[2], Hanwen Liang[1], Chun-Yi Lee[3], Rahul G. Krishnan[1]
[1] University of Toronto, Vector Institute
[2] NVIDIA AI Technology Center
[3] National Taiwan University
{chchao, rahulgk}@cs.toronto.edu, cylee@csie.ntu.edu.tw

Abstract

Masked diffusion models (MDM) are powerful generative models for discrete data that generate samples by progressively unmasking tokens in a sequence. Each token can take one of two states: *masked* or *unmasked*. We observe that token sequences often remain unchanged between consecutive sampling steps; consequently, the model repeatedly processes identical inputs, leading to redundant computation. To address this inefficiency, we propose the Partial masking scheme (Prime), which augments MDM by allowing tokens to take intermediate states interpolated between the masked and unmasked states. This design enables the model to make predictions based on partially observed token information, and facilitates a fine-grained denoising process. We derive a variational training objective and introduce a simple architectural design to accommodate intermediate-state inputs. Our method demonstrates superior performance across a diverse set of generative modeling tasks. On text data, it achieves a perplexity of 15.36 on OpenWebText, outperforming previous MDM (21.52), autoregressive models (17.54), and their hybrid variants (17.58), without relying on an autoregressive formulation. On image data, it attains competitive FID scores of 3.26 on CIFAR-10 and 6.98 on ImageNet-32, comparable to leading continuous generative models.

1 Introduction

Discrete data generation has been a central focus of probabilistic modeling since the early development of neural networks [1–3]. The goal is to build a model capable of generating sequences of symbolic units, known as *tokens*, which can represent words in natural language data or pixels in images. The field has been largely shaped by autoregressive models (ARM) (e.g., [4–6]), which capture the distribution of sequence data by factorizing it according to a prespecified left-to-right order. Recently, promising results from [7, 8] have demonstrated that order-agnostic models, such as masked diffusion models (MDM) (e.g., [7–10]), can be effectively extended to large-scale generation tasks, opening a new venue for discrete generative modeling.

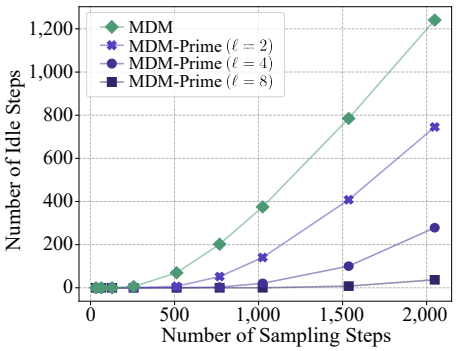


Figure 1: Number of idle steps during the reverse diffusion processes of MDM and MDM-Prime. The results are averaged over ten runs. ℓ is the sub-token sequence length.

Masked diffusion models are latent-variable generative models that introduce noise by progressively masking tokens within a sequence over time. During the reverse diffusion process, masked tokens are

Preprint. Under review.

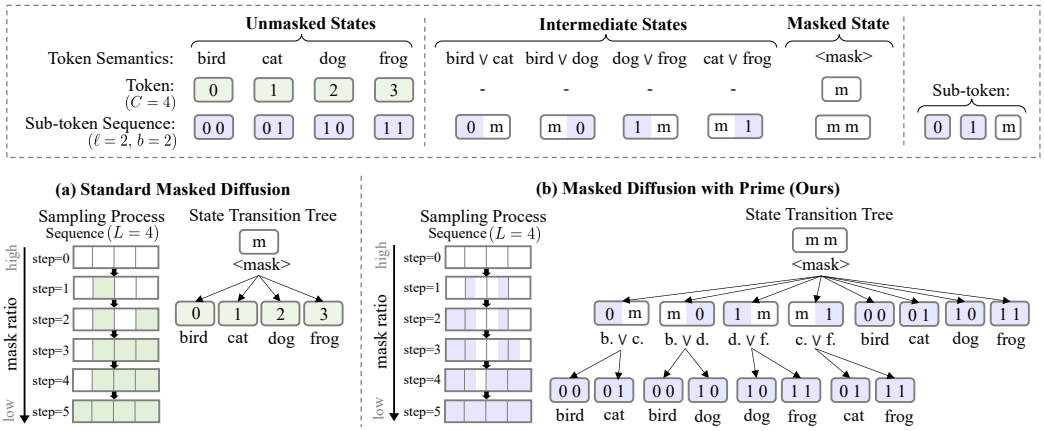


Figure 2: An illustrative example of (a) standard MDM and (b) MDM-Prime. Each token and its corresponding sub-token sequence (constructed via base-b encoding) can take one of three states: *unmasked*, *masked*, or *intermediate*. The masked and intermediate states serve as the latent representations produced by the forward diffusion process. This example contains $C = 4$ possible token classes, labeled as 'bird', 'cat', 'dog', and 'frog'. $\ell = 2$ indicates that each token is represented using two sub-tokens, and $b = \sqrt[\ell]{C} = 2$ denotes the number of classes per sub-token. The symbol `m` represents a masked token or a masked sub-token. The bottom-right sections of (a) and (b) illustrate the state transition trees. MDM-Prime supports transitions through intermediate states while retaining the ability to directly reach unmasked states. The bottom-left portions depict the sampling process for a token sequence of length $L = 4$. In (a), an idle step occurs between steps 3 and 4. In contrast, (b) demonstrates a sampling process without idle steps, which leads to improved model utilization.

incrementally unmasked according to a predefined ratio. Each token takes one of two states, *masked* or *unmasked*. This binary representation introduces an inefficiency; the entire sequence often remains unchanged across consecutive sampling steps, causing the model to repeatedly process the same input. This phenomenon is illustrated in the green curve in Fig. 1, which quantifies the number of such *idle steps* by simulating the reverse diffusion process of an MDM [9]. The figure shows that 37% of the 1,024 steps produce no update to the sequence during the reverse diffusion process. This inefficiency motivates our investigation of redefining the diffusion process to transform these idle steps into informative updates for improving the utilization of the model during generation.

We propose a simple yet effective solution that allows each token to take an *intermediate* state, which represents the interpolation between the masked and unmasked states, in the diffusion process. We refer to this method as the Partial masking scheme (Prime), and denote MDM augmented with Prime as MDM-Prime. Prime represents each token as a sequence of sub-tokens using a base-b encoding, with masking performed at the sub-token level. Since sub-tokens can be masked independently during the forward diffusion process, this method introduces intermediate states that partially reveal token information. An illustrative example is shown in the top of Fig. 2, where unmasked states '0-3' are first encoded as '00-11', and the intermediate states are obtained by masking one of the sub-tokens in the sub-token sequence. The intermediate states enable MDM-Prime to perform a fine-grained denoising process. For example, as illustrated in the 'State Transition Tree' of Fig. 2 (b), a four-choice prediction can be decomposed into two binary decisions during the sampling process (e.g., `mm`$\rightarrow$`m1`$\rightarrow$11). Compared to standard MDM transitions (i.e., Fig. 2 (a)), MDM-Prime is capable of making predictions based on partially observed token information while deferring the final token revelation until later sampling steps. With its ability to transition through intermediate states, MDM-Prime demonstrates improved model utilization during sampling, as reflected in the reduced number of idle steps in Fig. 1 (i.e., the purple curves). This in turn leads to enhanced performance, as later presented in Section 4. The contributions of this work are as follows:

- We propose MDM-Prime, a generalized MDM framework that enables intermediate token transitions. This framework can be optimized through a variational upper bound, which approximates the negative log-likelihood.
- We present a simple implementation of MDM-Prime built upon the standard MDM architecture. Our design requires only minor modifications to the input embedding layer of the standard MDM.

- We demonstrate that MDM-Prime achieves superior performance on both image and text generation tasks. On the OpenWebText dataset [11], MDM-Prime attains an evaluation perplexity of 15.36, outperforming ARM (17.54), MDM [7, 9, 10, 12] (21.52), and their hybrid variants [12, 13] (17.58). To the best of our knowledge, this is the first MDM-based approach to surpass ARM without relying on the autoregressive formulation. Furthermore, MDM-Prime achieves FID scores [14] of 3.26 on CIFAR-10 [15] and 6.98 on ImageNet-32 [16], demonstrating competitive performance comparable to leading continuous generative modeling methods [17–19].

2 Background

We first provide background on continuous-time masked diffusion processes. Section 2.1 describes the forward process, while Section 2.2 elaborates on the reverse process.

2.1 Forward Diffusion Process

Let $\mathcal{X} \triangleq \{0, \cdots, C-1\}$ denote a set of C tokens, and let $\boldsymbol{x} = [x^1, \cdots, x^L] \in \mathcal{X}^L$ be a sequence of tokens with length L, where the superscript indicates the index of each token. Let $\boldsymbol{x}_0$ denote the sample drawn from a data distribution p_{data}, which is defined as a probability mass function (pmf) over $\mathcal{X}^L$. Given a continuous time variable $t \in [0, 1]$, the latent variable introduced by the forward diffusion process is denoted as $\boldsymbol{x}_t \in (\mathcal{X} \cup \{\mathtt{m}\})^L \triangleq \tilde{\mathcal{X}}^L$, where $\mathtt{m}$ represents the masked token. In addition, let $\delta_{x'}(x)$ be the Kronecker delta function, which equals 1 if $x = x'$ and 0 otherwise. The forward diffusion process is performed through an element-wise conditional sampler $q(\boldsymbol{x}_t|\boldsymbol{x}_0) = \prod_{i=1}^{L} q(x_t^i|x_0^i)$ constructed by interpolating between $\delta_{\mathtt{m}}(\cdot)$ and $\delta_{x_0^i}(\cdot)$, defined as follows [9, 10, 20]:

$$q(x_t^i|x_0^i) = (1 - \alpha_t)\delta_{\mathtt{m}}(x_t^i) + \alpha_t \delta_{x_0^i}(x_t^i), \tag{1}$$

where $\alpha_t \in [0, 1]$ is a strictly decreasing scheduling function with boundary conditions $\alpha_0 \approx 1$ and $\alpha_1 \approx 0$. Intuitively, each perturbed token x_t^i retains the original value x_0^i with probability α_t, and is replaced by $\mathtt{m}$ with probability $1 - \alpha_t$. At time $t = 1$, the latent variable $\boldsymbol{x}_1 = [\mathtt{m}, \cdots, \mathtt{m}]$ consists entirely of masked tokens, exhibiting no randomness and revealing no information about $\boldsymbol{x}_0$. This process can also be interpreted as decreasing the mutual information between $\boldsymbol{x}_t$ and $\boldsymbol{x}_0$ over time according to α_t [21, 20] (i.e., $I(\boldsymbol{x}_t; \boldsymbol{x}_0) \approx \alpha_t H(\boldsymbol{x}_0)$, where $H(\boldsymbol{x}_0)$ denotes the entropy of $\boldsymbol{x}_0$) as explained in Appendix A.1.2. Based on Eq. (1), the forward diffusion process can be accomplished by first drawing a data sample $\boldsymbol{x}_0 \sim p_{\text{data}}(\cdot)$ and then applying masking to obtain $\boldsymbol{x}_t \sim q(\cdot|\boldsymbol{x}_0)$.

2.2 Reverse Diffusion Process

Let s and t be two time variables that satisfy $0 \leq s < t \leq 1$. The reverse diffusion process is performed by iterating through $p(\boldsymbol{x}_s|\boldsymbol{x}_t)$, starting from $\boldsymbol{x}_1$. The distribution $p(\boldsymbol{x}_s|\boldsymbol{x}_t)$ can be derived using the conditional distributions $q(\boldsymbol{x}_t|\boldsymbol{x}_s)$ and $q(\boldsymbol{x}_s|\boldsymbol{x}_t, \boldsymbol{x}_0)$. In particular, the transition distribution $q(\boldsymbol{x}_t|\boldsymbol{x}_s) = \prod_{i=1}^{L} q(x_t^i|x_s^i)$ is defined to be *absorbing* [20] on the masked state (i.e., a masked token remains masked from s to t), and is derived from Eq. (1) as follows [10]:

$$q(x_t^i|x_s^i) = \begin{cases} \frac{\alpha_s - \alpha_t}{\alpha_s}\delta_{\mathtt{m}}(x_t^i) + \frac{\alpha_t}{\alpha_s}\delta_{x_s^i}(x_t^i) & \text{if } x_s^i \in \mathcal{X}, \\ \delta_{x_s^i}(x_t^i) & \text{if } x_s^i = \mathtt{m}. \end{cases} \tag{2}$$

Based on Eqs. (1) and (2), the posterior distribution $q(\boldsymbol{x}_s|\boldsymbol{x}_t, \boldsymbol{x}_0) = \prod_{i=1}^{L} q(x_s^i|x_t^i, x_0^i)$ can be derived using Bayes' rule and the Markov property of the diffusion process, and is expressed as [9, 10]:

$$q(x_s^i|x_t^i, x_0^i) = \begin{cases} \delta_{x_t^i}(x_s^i) & \text{if } x_t^i \in \mathcal{X}, \\ \frac{1-\alpha_s}{1-\alpha_t}\delta_{\mathtt{m}}(x_s^i) + \frac{\alpha_s - \alpha_t}{1-\alpha_t}\delta_{x_0^i}(x_s^i) & \text{if } x_t^i = \mathtt{m}. \end{cases} \tag{3}$$

Eq. (3) indicates that, given x_0^i and observing a masked token $x_t^i = \mathtt{m}$, the reverse process transitions the masked token to its original value x_0^i with probability $\frac{\alpha_s - \alpha_t}{1-\alpha_t}$ or retains its value with probability $\frac{1-\alpha_s}{1-\alpha_t}$. For the unmasked token $x_t^i \in \mathcal{X}$, its value remains unchanged for the remaining steps.

Since $p(\boldsymbol{x}_s|\boldsymbol{x}_t) = \mathbb{E}_{p(\boldsymbol{x}_0|\boldsymbol{x}_t)}[q(\boldsymbol{x}_s|\boldsymbol{x}_t, \boldsymbol{x}_0)]$ [20], many recent works [8–10, 20, 22–24] choose to model $p(\boldsymbol{x}_0|\boldsymbol{x}_t)$ using $p_\theta(\boldsymbol{x}_0|\boldsymbol{x}_t)$, where θ denotes the model parameters. To facilitate computational

efficiency, this distribution is typically factorized as $p_\theta(\boldsymbol{x}_0|\boldsymbol{x}_t) = \prod_{i=1}^{L} p_\theta(x_0^i|\boldsymbol{x}_t)$ [8–10, 20, 22–24]. The parameter θ can be optimized by estimating the negative log-likelihood $-\log p_\theta(\boldsymbol{x}_0)$ using a variational upper bound, written as [9, 10]:

$$\mathcal{L}_{\text{vb}}(\boldsymbol{x}_0;\theta) = \int_0^1 \frac{\alpha_t'}{1-\alpha_t} \mathbb{E}_{q(\boldsymbol{x}_t|\boldsymbol{x}_0)} \left[\sum_{i=1}^{L} \log p_\theta(x_0^i|\boldsymbol{x}_t) \right] dt, \tag{4}$$

where $\alpha_t' = \frac{d}{dt}\alpha_t$. This objective specifies a cross-entropy loss, weighted by the coefficient $\frac{\alpha_t'}{1-\alpha_t}$, and satisfies $\mathcal{L}_{\text{vb}}(\boldsymbol{x}_0;\theta) \geq -\log p_\theta(\boldsymbol{x}_0)$ [9, 10]. In addition, since unmasked elements in $\boldsymbol{x}_t$ retain their values over time according to Eq. (3), the *carry-over* parameterization [9] (also referred to as the *mean* parameterization in [10]) can be applied by explicitly setting the corresponding entries in $\boldsymbol{x}_0$ to the unmasked values in $\boldsymbol{x}_t$. Formally, this is expressed as $p_\theta(x_0^i|\boldsymbol{x}_t) \triangleq \delta_{x_t^i}(x_0^i)$ for position i where $x_t^i \in \mathcal{X}$. This technique ensures that the model avoids redundant predictions on unmasked elements, and therefore can effectively reduce $\mathcal{L}_{\text{vb}}$ [9].

3 Methodology

In this section, we introduce a new MDM framework that incorporates the Partial masking scheme (Prime), referred to as MDM-Prime. We begin by introducing an invertible function for constructing sub-token sequences and define a novel masked diffusion process over these sub-tokens, which enables intermediate state transitions and reduces the number of idle steps (Section 3.1). We then propose a novel parameterization that captures sub-token dependencies via joint probability distributions, and describe a simple model architecture design that operates over sub-token inputs (Section 3.2).

3.1 Discrete Diffusion via Partial Masking

The core idea of Prime is to represent each token x_0^i with a sub-token sequence $\boldsymbol{y}_0^i = [y_0^{i,1}, \cdots, y_0^{i,\ell}]$, allowing the creation of intermediate states during the element-wise masking of the forward diffusion process (i.e., Eq. (1)). Given a target length $\ell > 1$, this can be achieved through the use of an invertible function f, which maps each token $x_0^i \in \mathcal{X}$ to its base-b encoding $\boldsymbol{y}_0^i \in \mathcal{Y}^\ell$ (i.e., a sequence of ℓ sub-tokens), where $\mathcal{Y} \triangleq \{0, \cdots, b-1\}$ denotes the set of sub-tokens with $b = \lceil \sqrt[\ell]{C} \rceil \in \mathbb{N}$.

The invertibility of f enables the likelihood to be defined on the set of sub-tokens $\mathcal{Y}$. For convenience, we slightly abuse notation and extend f to accept vector inputs: $f(\boldsymbol{x}_0) \triangleq [f(x_0^1), \cdots, f(x_0^L)] = [\boldsymbol{y}_0^1, \cdots, \boldsymbol{y}_0^L] = [(y_0^{1,1}, \cdots, y_0^{1,\ell}), \cdots, (y_0^{L,1}, \cdots, y_0^{L,\ell})] = \boldsymbol{y}_0$. Its inverse is denoted as f^{-1}. Under this transformation, the pmf of the data, $p_{\text{data}}(\boldsymbol{x}_0)$, can be equivalently written as $p_{\text{data}} \circ f^{-1}(\boldsymbol{y}_0)$ according to the change-of-variable principle. Rather than directly modeling $\boldsymbol{x}_0$, we model $\boldsymbol{y}_0$ through MDM, and reconstruct $\boldsymbol{x}_0$ using $f^{-1}(\boldsymbol{y}_0)$, resulting in a parameterized pmf $p_\theta(\boldsymbol{y}_0) = p_\theta \circ f(\boldsymbol{x}_0)$. An illustrative example is provided in Fig. 3.

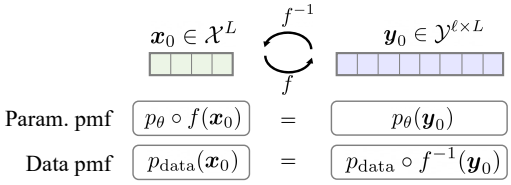


Figure 3: Illustration of the data and parameterized pmf with an invertible f. 'Param. pmf' denotes the parameterized pmf captured using an MDM with parameter θ. In this example, $\ell = 2$.

To learn $p_\theta(\boldsymbol{y}_0)$ using MDM, each data point $\boldsymbol{x}_0$ is first transformed into $\boldsymbol{y}_0 = f(\boldsymbol{x}_0)$, and its corresponding latent variable $\boldsymbol{y}_t$ is sampled from $q(\boldsymbol{y}_t|\boldsymbol{y}_0) = \prod_{i=1}^{L} \prod_{j=1}^{\ell} q(y_t^{i,j}|y_0^{i,j})$ according to Eq. (1) by substituting x's with y's. The reverse diffusion process is parameterized by $p_\theta(\boldsymbol{y}_0|\boldsymbol{y}_t) = \prod_{i=1}^{L} p_\theta(\boldsymbol{y}_0^i|\boldsymbol{y}_t)$, and the model is trained to minimize the loss $\mathcal{L}_{\text{vb}}(\boldsymbol{y}_0;\theta)$ as in Eq. (4):

$$\mathcal{L}_{\text{vb}}(\boldsymbol{y}_0;\theta) = \int_0^1 \frac{\alpha_t'}{1-\alpha_t} \mathbb{E}_{q(\boldsymbol{y}_t|\boldsymbol{y}_0)} \left[\sum_{i=1}^{L} \log p_\theta(\boldsymbol{y}_0^i|\boldsymbol{y}_t) \right] dt. \tag{5}$$

Section 3.2 provides details on the parameterization of $p_\theta(\boldsymbol{y}_0^i|\boldsymbol{y}_t)$, while Appendix A.2.1 shows that Eq. (5) defines a variational bound that approximates negative log-likelihood (NLL).

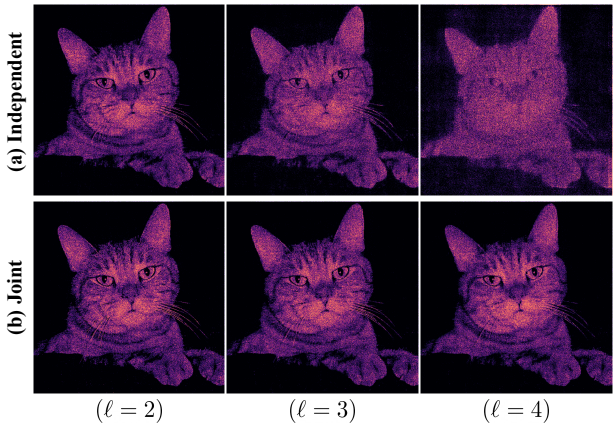


Figure 4: Distributions modeled by MDM-Prime using (a) independent and (b) joint parameterizations. Models are trained on a two-dimensional synthetic dataset with $\boldsymbol{x}_0 \in [0, \cdots, 511]^2$ representing the coordinate of the figure (512×512). Brighter regions indicate higher probabilities. Experimental details are offered in Appendix A.4.1.

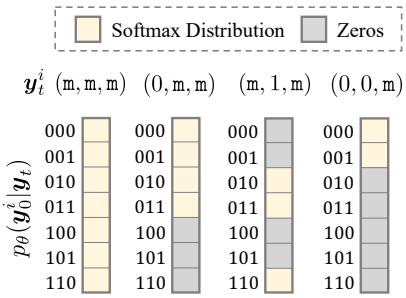


Figure 5: An illustration of the proposed carry-over parameterization technique. In this example, $C = 7$, $\ell = 3$, and $b = 2$. The conditional distribution $p_\theta(\boldsymbol{y}_0^i|\boldsymbol{y}_t)$ is defined in Eq. (7). Softmax distributions are formed by normalizing the corresponding logit outputs highlighted in yellow.

This new diffusion process operates on an augmented set $\tilde{\mathcal{Y}} \triangleq \mathcal{Y} \cup \{\mathtt{m}\}$. For each token, the number of intermediate states introduced by f and the diffusion process is given by $|\tilde{\mathcal{Y}}^\ell| - |\tilde{\mathcal{X}}| = (b+1)^\ell - (C+1)$. **Proposition** A.3 ensures that this number is always positive. This property allows MDM-Prime to learn smooth transitions via a rich set of intermediate states. Moreover, **Proposition** A.1 and Eq. (A5) formally establish that the number of idle steps decreases as ℓ increases, which guarantees an improved model utilization of MDM-Prime during the reverse diffusion process.

3.2 Parameterization

In this section, we discuss our implementation for $p_\theta(\boldsymbol{y}_0^i|\boldsymbol{y}_t)$, which comprises a decoder for modeling the distribution of $\boldsymbol{y}_0^i$ and an encoder for processing the input $\boldsymbol{y}_t$. We begin by outlining the decoder design, followed by a description of the encoder.

Decoder Design via Joint Probability. A straightforward approach to implementing $p_\theta(\boldsymbol{y}_0^i|\boldsymbol{y}_t)$ is to factorize it as $\prod_{j=1}^{\ell} p_\theta(y_0^{i,j}|\boldsymbol{y}_t)$, modeling each component with a softmax distribution. While this factorization allows for easy application of the *carry-over* parameterization [9, 10] (discussed in Section 2.2), it introduces two key challenges: (1) an independence assumption and (2) potential generation of invalid samples. First, the factorized form $p_\theta(\boldsymbol{y}_0^i|\boldsymbol{y}_t) = \prod_{j=1}^{\ell} p_\theta(y_0^{i,j}|\boldsymbol{y}_t)$ imposes an additional independence assumption across sub-tokens, preventing the model from capturing dependencies between them. As shown in Fig. 4 (a), increasing the sub-token sequence length ℓ leads to a deterioration in the sampling distribution due to this limitation. Second, since f is defined as an injective function but not necessarily bijective (i.e., $|f(\mathcal{X})| \leq |\mathcal{Y}^\ell|$), some samples $\boldsymbol{y}_0$ produced by such a factorized model may not correspond to any valid $\boldsymbol{x}_0$ under the inverse mapping f^{-1}. For instance, when encoding the GPT-2 [5] vocabulary of $C = 50,257$ classes with $\ell = 4$ and $b = \lceil \sqrt[\ell]{C} \rceil = 15$, the model may generate invalid sub-token sequences such as $\boldsymbol{y}_0^i = (14, 14, 14, 14)$ during inference time, even though there is no corresponding $x_0^i = 50,624$ for decoding.

To address these two issues, we propose to model the joint distributions $p_\theta(\boldsymbol{y}_0^i|\boldsymbol{y}_t)$ of a sequence of ℓ sub-tokens while explicitly zeroing out the probability mass assigned to invalid samples. This is achieved by parameterizing only the logits of base-b encoding $\boldsymbol{y}_0^i \in f(\mathcal{X})$ that correspond to a valid $x_0^i \in \mathcal{X}$, which results in exactly C entries in the logit outputs for each position $i \in \{1, \cdots, L\}$.

Based on the above joint probability design, to further support carry-over parameterization for MDM-Prime, the element-wise distribution should be defined as $p_\theta(y_0^{i,j}|\boldsymbol{y}_t) \triangleq \delta_{y_t^{i,j}}(y_0^{i,j})$ for all position i, j where $y_t^{i,j} \in \mathcal{Y}$ (following the end of Section 2.2). Since we parameterize the joint distribution as

$p_\theta(\boldsymbol{y}_0^i|\boldsymbol{y}_t) = p_\theta(y_0^{i,1}, \cdots, y_0^{i,\ell}|\boldsymbol{y}_t)$, this condition is imposed on the marginal distribution as follows:

$$p_\theta(y_0^{i,j}|\boldsymbol{y}_t) = \sum_{y_0^{i,1}, \cdots, y_0^{i,j-1}, y_0^{i,j+1}, \cdots, y_0^{i,\ell} \in \mathcal{Y}} p_\theta(y_0^{i,1}, \cdots, y_0^{i,\ell}|\boldsymbol{y}_t) \triangleq \delta_{y_t^{i,j}}(y_0^{i,j}). \tag{6}$$

To meet this condition, the probabilities of $\boldsymbol{y}_0^i$ with any element $y_0^{i,j}$ that is inconsistent with $y_t^{i,j}$ should be explicitly set to zero. The parameterized probability can thus be defined as follows:

$$p_\theta(\boldsymbol{y}_0^i|\boldsymbol{y}_t) = \begin{cases} \frac{\exp(E_\theta(\boldsymbol{y}_0^i|\boldsymbol{y}_t))}{\sum_{\boldsymbol{y}^i \in \mathcal{V}(\boldsymbol{y}_t^i)} \exp(E_\theta(\boldsymbol{y}^i|\boldsymbol{y}_t))}, & \text{if } \boldsymbol{y}_0^i \in \mathcal{V}(\boldsymbol{y}_t^i), \\ 0, & \text{if } \boldsymbol{y}_0^i \notin \mathcal{V}(\boldsymbol{y}_t^i), \end{cases} \tag{7}$$

where $\mathcal{V}(\boldsymbol{y}_t^i) \triangleq \{\boldsymbol{y}^i = [y^{i,1}, \cdots, y^{i,\ell}] \in f(\mathcal{X}) \ s.t. \ (y^{i,j} = y_t^{i,j}) \vee (y_t^{i,j} = \mathtt{m})\}$ denotes a set of outputs that is consistent with $y_t^{i,j}$, and $E_\theta : \mathcal{Y}^\ell \times \tilde{\mathcal{Y}}^{\ell \times L} \to \mathbb{R}$ is a scalar logit. **Proposition** A.4 guarantees the correctness of this parameterization (i.e., Eq. (7) satisfies Eq. (6)). Consider a simple example where $C = 7$, $\ell = 3$, and $b = 2$, then $\mathcal{V}(\boldsymbol{y}_t^i)$ corresponds to the following sets:

- If $\boldsymbol{y}_t^i = (\mathtt{m}, \mathtt{m}, \mathtt{m})$, then $\mathcal{V}(\boldsymbol{y}_t^i) = \{(0,0,0), (0,0,1), (0,1,0), (0,1,1), (1,0,0), (1,0,1), (1,1,0)\}$
- If $\boldsymbol{y}_t^i = (0, \mathtt{m}, \mathtt{m})$, then $\mathcal{V}(\boldsymbol{y}_t^i) = \{(0,0,0), (0,0,1), (0,1,0), (0,1,1)\}$
- If $\boldsymbol{y}_t^i = (0, 0, \mathtt{m})$, then $\mathcal{V}(\boldsymbol{y}_t^i) = \{(0,0,0), (0,0,1)\}$

Note that $(1,1,1)$ is invalid since $C = 7$ in this case. Some illustrative examples of $p_\theta(\boldsymbol{y}_0^i|\boldsymbol{y}_t)$ are provided in Fig. 5. As the reverse diffusion process progresses, the number of unmasked sub-tokens in $\boldsymbol{y}_t^i$ increases, leading to a substantial reduction in $|\mathcal{V}(\boldsymbol{y}_t^i)|$. This results in a decreasing number of candidate classes of $\boldsymbol{y}_0^i$ over time, and thus explicitly reducing uncertainty in the prediction task. From an implementation perspective, Eq. (7) can be efficiently derived using precomputed filters indexed by $y_t^{i,j}$. During the forward pass of the model, the logits of $\boldsymbol{y}_0^i \notin \mathcal{V}(\boldsymbol{y}_t^i)$ are excluded according to these filters. Further implementation details are provided in Appendix A.2.3 and Fig. A2.

Encoder Design for Processing Sub-tokens. In contrast to the decoder, where the distribution over sub-tokens $\boldsymbol{y}_0^i \in f(\mathcal{X})$ is represented jointly using logit outputs with C entries (see Figs. 5 and 6), the encoder receives noised inputs $\boldsymbol{y}_t^i$ that lie in the augmented set $\tilde{\mathcal{Y}}^\ell$. Since the size of this set, $|\tilde{\mathcal{Y}}^\ell|$, may grow with ℓ and typically exhibits $|\tilde{\mathcal{Y}}^\ell| \gg C$ (also see Appendix A.2.2), creating an embedding lookup table for $\boldsymbol{y}_t^i$ is impractical due to the resulting growth in the number of parameters in it. To address this issue, we propose to model each sub-token embedding separately (i.e., creating a lookup table for individual $y_t^{i,j} \in \tilde{\mathcal{Y}}$), followed by a merging operation to produce a token embedding.

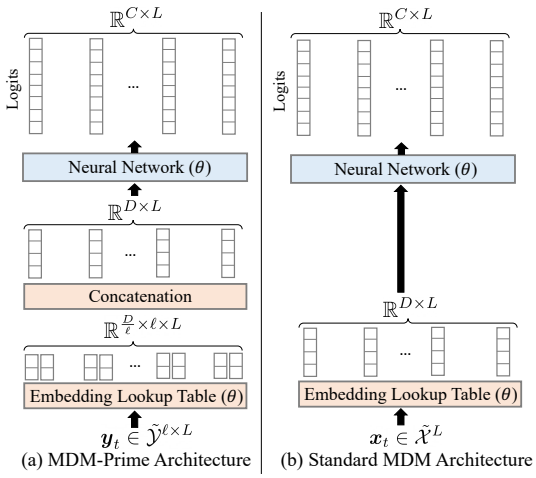


Figure 6: Comparison between (a) MDM-Prime and (b) standard MDM architectures. The embedding lookup table in (a) has fewer learnable parameters since $|\tilde{\mathcal{Y}}| < |\tilde{\mathcal{X}}|$ and $D/\ell < D$.

In our approach, a simple merging operation based on concatenation is employed. Let D denote the dimensionality of the token embedding vector. Each sub-token is first embedded into a vector of size D/ℓ, and the resulting ℓ embeddings are concatenated to form a D-dimensional token embedding vector. This token embedding can then be processed by an arbitrary downstream neural network, which allows us to reuse the standard MDM architecture. A comparison between this design and a standard MDM architecture is shown in Fig. 6. Alternative merging strategies, such as the Perceiver [25] cross-attention mechanism, are discussed and evaluated in Appendices A.3 and A.5.1, where we show that simple concatenation yields the best performance.

In summary, adapting a standard MDM to MDM-Prime requires only minimal architectural modifications on the embedding layer. This simple strategy preserves the overall architectural design of the standard MDM, enabling a fair comparison with our baseline in the following experiments.

Table 1: Zero-shot validation perplexities evaluated on seven textual datasets. Lower values correspond to better performance. Methods marked with * incorporate an autoregressive formulation. MDLM-Prime exhibits improved results on LAMBADA, PTB, and ArXiv.

	LAMBADA	WikiText	PTB	LM1B	AG News	PubMed	ArXiv
ARM* [9]	51.28	25.75	82.05	51.25	**52.09**	49.01	41.73
BD3-LM* [13]	≤50.03	≤31.31	≤96.81	≤60.88	≤61.67	≤42.52	≤39.20
EDLM-coAR* [12]	≤50.04	≤28.31	≤89.73	≤60.23	≤57.94	≤46.31	≤39.02
SEDD [7]	≤49.86	≤34.28	≤100.09	≤68.20	≤62.09	≤41.89	≤38.48
EDLM-NCE [12]	≤46.92	≤30.77	≤93.21	≤63.19	≤60.02	≤41.80	≤36.63
MDLM [9]	≤47.52	≤32.83	≤95.26	≤67.01	≤61.15	≤41.89	≤37.37
MDLM-Prime ($\ell = 2$)	≤30.91	≤27.93	≤74.81	≤55.50	≤63.21	≤**33.32**	≤25.44
MDLM-Prime ($\ell = 3$)	≤27.75	≤27.42	≤60.13	≤42.69	≤62.58	≤41.14	≤**24.71**
MDLM-Prime ($\ell = 4$)	≤**24.44**	≤**23.86**	≤53.98	≤38.02	≤59.44	≤48.64	≤25.83
MDLM-Prime ($\ell = 6$)	≤25.80	≤26.87	≤62.62	≤45.36	≤60.16	≤59.09	≤25.19
MDLM-Prime ($\ell = 8$)	≤25.23	≤25.77	≤**53.77**	≤**38.00**	≤64.89	≤54.50	≤25.79

4 Experiments

This section presents empirical evaluations to examine the effectiveness of the proposed method. We first report results in the text generation domain in Section 4.1. Then, we provide comparisons on the image generation benchmarks in Section 4.2.

4.1 Text Generation

Configuration. In this set of experiments, models are trained on the OpenWebText (OWT) dataset [11]. The text data are tokenized using the GPT-2 tokenizer [5], which defines $L = 1,024$ and $C = 50,257$. The Masked Diffusion Language Model (MDLM) [9] is adopted as our baseline, and the experimental setup is consistent with [9]. Prime with different ℓ is applied to enhance its performance, and our method is denoted as MDLM-Prime in this section. We also include comparisons with several recent approaches [7, 10, 12, 13]. In MDLM [9], MDLM-Prime, and all other recent methods [7, 9, 10, 12, 13], the core model architecture, i.e., the 'Neural Network (θ)' component in Fig. 6, is a diffusion transformer [26] with rotary positional embeddings [27]. No temperature is applied to the output probabilities during training and test time. Detailed hyperparameters are offered in Appendix A.4.3. Additional results are offered in Appendices A.5.1 and A.5.3.

Improvements to Likelihood Evaluation. We evaluate the models' ability to capture the data distribution using the perplexity (PPL) metric [28]. Table 2 reports PPL on OWT, along with the idle step ratio (ISR), which is defined as the proportion of idle steps relative to the total sampling steps and is computed using Eq. (A6). We observe that as ℓ increases, MDLM-Prime achieves lower PPL, with performance converging when $\ell \geq 4$. Since ISR also converges when $\ell \geq 4$, this trend suggests that ISR can serve as an indicator of improved likelihood modeling ability. We provide a further analysis of the relationship between performance and ISR, with a guideline for selecting ℓ, in Appendix A.2.4. Moreover, MDLM-Prime with $\ell \geq 3$ outperforms ARM, MDM-based approaches [7, 9, 10, 12], and their hybrid variants [13, 12] by a noticeable margin in terms of PPL, indicating that incorporating intermediate state representations allows MDLM-Prime to model data likelihood more effectively. Instead of following recent approaches [12, 13] that leverage an autoregressive formulation to enhance MDM performance, MDLM-Prime maintains an order-agnostic framework while achieving superior performance on textual data.

Table 2: PPL and ISR evaluation on OWT. Methods marked with * incorporate an autoregressive formulation. The symbol ↓ represents that lower values correspond to better performance. MDLM-Prime with $\ell \geq 3$ outperforms prior methods.

	PPL (↓)	ISR
ARM* [9]	17.54	-
BD3-LMs* [13]	≤20.73	-
EDLM-coAR* [12]	≤17.58	-
SEDD [7]	≤24.10	-
GenMD4 [10]	≤21.80	-
EDLM-NCE [12]	≤21.52	-
MDLM [9]	≤22.98	36.77%
MDLM-Prime ($\ell = 2$)	≤17.90	13.52%
MDLM-Prime ($\ell = 3$)	≤16.36	4.97%
MDLM-Prime ($\ell = 4$)	≤15.62	1.83%
MDLM-Prime ($\ell = 6$)	≤**15.36**	0.25%
MDLM-Prime ($\ell = 8$)	≤15.48	0.03%

Improvements to Generalizability to Unseen Text Data. With the models trained on OWT, we then examine their generalizability to unseen textual datasets. To assess the models' generalizability across diverse text domains, we report PPL on a suite of commonly used zero-shot benchmarks,

Table 3: FID and IS evaluation on CIFAR-10. The arrow symbols ↑ / ↓ represent that higher / lower results correspond to better performance.

CIFAR-10		
Discrete		
Method	FID (↓)	IS (↑)
MDM (NFE=512)	4.66	9.09
MDM-Mixture (NFE=512)	4.80	9.22
MDM-Prime (NFE=512)	**3.26**	**9.67**
PixelCNN [35]	65.93	4.60
D3PM Absorb [20] (NFE=1,000)	30.97	6.78
D3PM Gauss. [20] (NFE=1,000)	7.34	8.56
CTDD-DG [36] (NFE=1,000)	7.86	8.91
Tau-LDR [22] (NFE=1,000)	3.74	9.49
Discrete FM [23] (NFE=1,024)	3.63	-
Continuous		
NCSN [37]	25.32	8.87
Continuous FM [38]	6.35	-
Bit Diffusion [39]	3.48	-
StyleGAN+ADA [17]	3.26	**9.74**
DDPM [18]	**3.17**	9.46

Table 4: FID and IS evaluation on ImageNet-32. The arrow symbols ↑ / ↓ represent that higher / lower results correspond to better performance.

ImageNet-32		
Discrete		
Method	FID (↓)	IS (↑)
MDM (NFE=1,024)	7.91	11.60
MDM-Mixture (NFE=1,024)	8.08	11.56
MDM-Prime (NFE=1,024)	**6.98**	**11.65**
Continuous		
QC-NCSN++ [40]	19.62	9.94
NDM [41]	17.02	-
DDPM [18]	16.18	-
MSGAN [42]	12.30	-
i-DODE (SP) [43]	10.31	-
i-DODE (VP) [43]	9.09	-
Stochastic Interp. [44]	8.49	-
Soft Trunc. DDPM [45]	8.42	**11.82**
ScoreFlow (subVP) [19]	8.87	-
ScoreFlow (VP) [19]	8.34	-
Continuous FM [38]	**5.02**	-

including LAMBADA [29], WikiText [30], Penn Treebank (PTB) [31], 1 Billion Word Benchmark (LM1B) [32], AG News [33], and Scientific Papers (PubMed and ArXiv subsets [34]). The results are reported in Table 1. MDLM-Prime exhibits superior results on LAMBADA, PTB, and ArXiv, and achieves comparable performance to ARM on WikiText. While it underperforms ARM on AG News, the overall results highlight its superior generalizability across multiple domains. Furthermore, our ablation study in Appendix A.5.1 reveals that the carry-over parameterization plays an important role in enhancing zero-shot performance, offering improvements on both LAMBADA and PubMed.

4.2 Image Generation

Configuration. In this set of experiments, models are trained and evaluated on the CIFAR-10 [15] and ImageNet-32 [16] datasets. For both datasets, the dimensionality is set to $L = 32 \times 32 \times 3$, with $C = 256$ corresponding to pixel intensity values. The core model architecture is adapted from the ablated diffusion model (ADM) [46], which is the same as that used in [23]. Sample quality is evaluated using the widely adopted Fréchet Inception Distance (FID) [14] and Inception Score (IS) [47] metrics. Experimental details are provided in Appendix A.4.2. Additional results are presented in Appendices A.5.3 and A.5.4.

Improvements to Sample Quality. We first compare MDM-Prime with varying values of ℓ against the baseline configuration (i.e., $\ell = 1$). Due to the relatively small number of classes in the image experiments, we additionally explore the case of $\ell = 2/3$, where pixel values are merged into super-pixels, resulting in $b = \sqrt[2/3]{256} = 4,096$ discrete classes in one of the experimental settings. As shown in Fig. 7, the configuration with $\ell = 2$ achieves the best FID scores and exhibits a relatively low ISR compared to that of $\ell = 1$ and $2/3$. While the settings with $\ell = 3$ and $\ell = 4$ perform comparably to the baseline, we observe that models with lower ISR are less sensitive to the number of function evaluations (NFE) during sampling, as reflected by the smaller FID performance gaps between NFE=128 and 512.

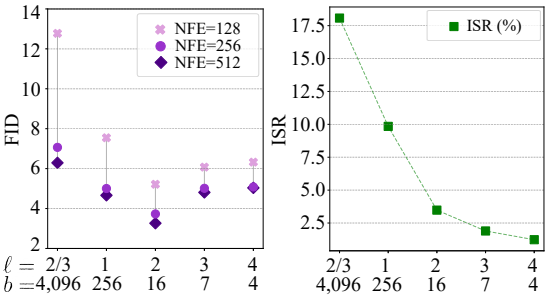


Figure 7: FID and ISR evaluated under different ℓ on CIFAR-10. NFE denotes the number of function evaluations during sampling.

The benchmark results are reported in Tables 3 and 4, which include two baselines, MDM and MDM-Mixture, as well as several existing generative modeling approaches [17–20, 22, 23, 35–45].

The MDM baseline corresponds to the standard configuration with $\ell = 1$, while MDM-Mixture extends this baseline by incorporating a mixture distribution using an auxiliary variable, similar to [48]. In this comparison, MDM-Prime adopts $\ell = 2$.

As shown in the tables, MDM and MDM-Mixture are inferior to MDM-Prime. On CIFAR-10, MDM-Prime achieves better results than the other discrete generative models while requiring fewer NFE, and attains performance comparable to StyleGAN+ADA [17]. On ImageNet-32, MDM-Prime demonstrates improved performance over existing continuous diffusion and score-based models (i.e., [18, 19, 40, 41, 43–45]), achieving an FID improvement of 1.36 over ScoreFlow (VP) [19].

Imputation under Partial Masking. To further evaluate MDM-Prime's capability for conditional image generation, we conduct experiments on image imputation. In contrast to text generation, where the positional index j of a sub-token $y_0^{i,j}$ lacks explicit semantic interpretation, sub-token positions in the image domain have a direct influence on the resulting pixel values. Specifically, the first sub-token $y_0^{i,1}$ has the greatest impact on determining the final pixel intensity. In this experiment, each conditional image (denoted as 'Condition' in Fig. 8) is first encoded into its corresponding sub-token representation. The first sub-token is retained, while the remaining sub-tokens are masked and subsequently predicted by our model. Some generated examples are shown in the 'Imputation' section of Fig. 8. The results demonstrate that our method can generate visually coherent images conditioned on the preserved sub-tokens, highlighting its effectiveness in controlled image synthesis.

Figure 8: Imputation results with conditional images obtained from CIFAR-10. MDM-Prime generates visually coherent image variants. More examples are provided in Appendix A.5.3.

5 Related Works

The general framework of discrete diffusion models was first introduced by [21], where the authors explored modeling binary data using a diffusion process. This framework was extended to real-world applications, such as text and image generation, by the authors in [20], who proposed various perturbation strategies to implement diffusion processes with discrete noise. The authors in [22] proposed generalizing this framework as Continuous-Time Markov Chains (CTMC). Inspired by the success of scaling MDM for text generation [7], some works [9, 10, 49] explored simplifications of the training objective of discrete diffusion models with latent variables represented using masked tokens. Other studies [23, 50] investigated learning approaches based on a broader class of latent representations formulated through flow matching.

Building on the theoretical foundation of MDM, several enhancement techniques [12, 13, 23, 48, 51–53] have been proposed. In [13], the authors proposed an interpolation between ARM and MDM to capture the left-to-right structure in textual data. In [12], an ARM was employed as an energy-based function to guide the sampling process of an MDM, resulting in improved performance. Other works [23, 51, 52] modified the sampling process of MDM to selectively remask certain unmasked predictions to enhance sample quality. In addition, the authors in [48, 53] explored distillation techniques designed to reduce the number of sampling steps while maintaining sample quality.

Another line of research has explored the use of diffusion models with continuous noise distributions (i.e., Gaussian) for modeling discrete data. Representative methods include [39, 54–58]. Among these works, Bit Diffusion [39] shares similarities with our approach. In their method, discrete data are first encoded into bit representations, and a continuous diffusion model (with Gaussian kernels) is trained to generate these encodings. The generated outputs are then quantized back into discrete tokens. However, due to its reliance on quantization, the model's likelihood becomes intractable, which leads to its inability to directly capture the distribution of discrete data.

6 Conclusion

Scientific progress has continually reshaped our understanding of what constitutes the most basic units of matter. Physicists initially believed that atoms were elementary units of matter. This view

changed with the discoveries of the electron, the atomic nucleus, and eventually the development of the *standard model* [59], which describes fundamental particles, their interactions, and how they combine to form atoms. In the context of generative models, we proposed Prime, a method to decompose the elementary unit of discrete data–tokens–into fine-grained subcomponents. MDM-Prime establishes a principled framework for perturbing and reconstructing discrete data using sub-token representations. Experimental results on both text and image generation tasks demonstrated that sub-token representations provide a more expressive modeling paradigm. We believe that this framework holds potential for addressing real-world problems that require fine-grained and precise modeling of discrete data.

Acknowledgements

RGK is supported by a Canada CIFAR AI Chair and a Canada Research Chair Tier II in Computational Medicine. This research was supported by an NFRF Special Call NFRFR2022-00526. Resources used in preparing this research were provided, in part, by the Province of Ontario, the Government of Canada through CIFAR, and companies sponsoring the Vector Institute. The authors gratefully acknowledge support from the National Science and Technology Council (NSTC) in Taiwan under grant numbers MOST 111-2223-E-002-011-MY3, NSTC 113-2221-E-002-212-MY3, and NSTC 113-2640-E-002-003. We also express our sincere appreciation to NVIDIA Corporation and the NVIDIA AI Technology Center (NVAITC) for the donation of GPUs and access to the Taipei-1 supercomputer, and to Google Inc. for additional support. Furthermore, we thank the National Center for High-Performance Computing for providing computational and storage resources. Finally, we are grateful to David Pellow and Vahid Belazadeh Meresht for their thoughtful review and valuable feedback on this work.

References

[1] Ilya Sutskever, James Martens, and Geoffrey Hinton. Generating Text with Recurrent Neural Networks. In *Proc. Int. Conf. on Machine Learning (ICML)*, 2011.

[2] Tomas Mikolov, Ilya Sutskever, Anoop Deoras, Hai Son Le, Stefan Kombrink, and Jan Honza Cernocky. Subword Language Modeling with Neural Networks. 2011.

[3] Alex Graves. Generating Sequences With Recurrent Neural Networks. `arXiv:1308.0850 [cs.NE]`, 2013.

[4] Ashish Vaswani, Noam Shazeer, Niki Parmar, Jakob Uszkoreit, Llion Jones, Aidan N. Gomez, Lukasz Kaiser, and Illia Polosukhin. Attention Is All You Need. In *Proc. of Int. Conf. on Neural Information Processing Systems (NeurIPS)*, 2017.

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

[6] Hugo Touvron, Thibaut Lavril, Gautier Izacard, Xavier Martinet, Marie-Anne Lachaux, Timothée Lacroix, Baptiste Rozière, Naman Goyal, Eric Hambro, Faisal Azhar, Aurelien Rodriguez, Armand Joulin, Edouard Grave, and Guillaume Lample. LLaMA: Open and Efficient Foundation Language Models. `arXiv:2302.13971 [cs.CL]`, 2023.

[7] Aaron Lou, Chenlin Meng, and Stefano Ermon. Discrete Diffusion Modeling by Estimating the Ratios of the Data Distribution. In *Proc. Int. Conf. on Machine Learning (ICML)*, 2024.

[8] Shen Nie, Fengqi Zhu, Zebin You, Xiaolu Zhang, Jingyang Ou, Jun Hu, Jun Zhou, Yankai Lin, Ji-Rong Wen, and Chongxuan Li. Large Language Diffusion Models. `arXiv:2502.09992 [cs.CL]`, 2025.

[9] Subham Sekhar Sahoo, Marianne Arriola, Yair Schiff, Aaron Gokaslan, Edgar Marroquin, Justin T Chiu, Alexander Rush, and Volodymyr Kuleshov. Simple and Effective Masked Diffusion Language Models. In *Proc. of the Int. Conf. on Neural Information Processing Systems (NeurIPS)*, 2024.

[10] Jiaxin Shi, Kehang Han, Zhe Wang, Arnaud Doucet, and Michalis K. Titsias. Simplified and Generalized Masked Diffusion for Discrete Data. In *Proc. of the Int. Conf. on Neural Information Processing Systems (NeurIPS)*, 2024.

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

[12] Minkai Xu, Tomas Geffner, Karsten Kreis, Weili Nie, Yilun Xu, Jure Leskovec, Stefano Ermon, and Arash Vahdat. Energy-Based Diffusion Language Models for Text Generation. In *Proc. Int. Conf. on Learning Representations (ICLR)*, 2025.

[13] 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 *Proc. Int. Conf. on Learning Representations (ICLR)*, 2025.

[14] Martin Heusel, Hubert Ramsauer, Thomas Unterthiner, Bernhard Nessler, and Sepp Hochreiter. GANs Trained by a Two Time-Scale Update Rule Converge to a Local Nash Equilibrium. In *Proc. of Int. Conf. on Neural Information Processing Systems (NeurIPS)*, 2017.

[15] Alex Krizhevsky and Geoffrey Hinton. Learning Multiple Layers of Features from Tiny Images. Technical report, University of Toronto, 2009.

[16] Patryk Chrabaszcz, Ilya Loshchilov, and Frank Hutter. A Downsampled Variant of ImageNet as an Alternative to the CIFAR datasets. `arXiv:1707.08819 [cs.CV]`, 2017.

[17] Tero Karras, Miika Aittala, Janne Hellsten, Samuli Laine, Jaakko Lehtinen, and Timo Aila. Training Generative Adversarial Networks with Limited Data. In *Proc. of Int. Conf. on Neural Information Processing Systems (NeurIPS)*, 2020.

[18] Jonathan Ho, Ajay Jain, and Pieter Abbeel. Denoising Diffusion Probabilistic Models. In *Proc. of Int. Conf. on Neural Information Processing Systems (NeurIPS)*, 2020.

[19] Yang Song, Conor Durkan, Iain Murray, and Stefano Ermon. Maximum Likelihood Training of Score-Based Diffusion Models. In *Proc. of Int. Conf. on Neural Information Processing Systems (NeurIPS)*, 2021.

[20] Jacob Austin, Daniel D. Johnson, Jonathan Ho, Daniel Tarlow, and Rianne van den Berg. Structured Denoising Diffusion Models in Discrete State-Spaces. In *Proc. of the Int. Conf. on Neural Information Processing Systems (NeurIPS)*, 2021.

[21] Jascha Sohl-Dickstein, Eric A. Weiss, Niru Maheswaranathan, and Surya Ganguli. Deep Unsupervised Learning using Nonequilibrium Thermodynamics. In *Proc. Int. Conf. on Machine Learning (ICML)*, 2015.

[22] Andrew Campbell, Joe Benton, Valentin De Bortoli, Tom Rainforth, George Deligiannidis, and Arnaud Doucet. A Continuous Time Framework for Discrete Denoising Models. In *Proc. of Int. Conf. on Neural Information Processing Systems (NeurIPS)*, 2022.

[23] Itai Gat, Tal Remez, Neta Shaul, Felix Kreuk, Ricky T. Q. Chen, Gabriel Synnaeve, Yossi Adi, and Yaron Lipman. Discrete Flow Matching. In *Proc. of Int. Conf. on Neural Information Processing Systems (NeurIPS)*, 2024.

[24] 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 *Proc. Int. Conf. on Learning Representations (ICLR)*, 2024.

[25] Andrew Jaegle, Felix Gimeno, Andrew Brock, Andrew Zisserman, Oriol Vinyals, and Joao Carreira. Perceiver: General Perception with Iterative Attention. In *Proc. Int. Conf. on Machine Learning (ICML)*, 2021.

[26] William Peebles and Saining Xie. Scalable Diffusion Models with Transformers. `arXiv:2212.09748 [cs.CV]`, 2022.

[27] Jianlin Su, Yu Lu, Shengfeng Pan, Ahmed Murtadha, Bo Wen, and Yunfeng Liu. RoFormer: Enhanced Transformer with Rotary Position Embedding. `arXiv:2104.09864 [cs.CL]`, 2023.

[28] Daniel Jurafsky and James H. Martin. *Speech and Language Processing: An Introduction to Natural Language Processing, Computational Linguistics, and Speech Recognition*. 3rd edition, 2025.

[29] 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 *Proc. Annual Meeting of the Association for Computational Linguistics (ACL)*, 2016.

[30] Stephen Merity, Caiming Xiong, James Bradbury, and Richard Socher. Pointer Sentinel Mixture Models. `arXiv:1609.07843 [cs.CL]`, 2016.

[31] Mitchell P. Marcus, Beatrice Santorini, and Mary Ann Marcinkiewicz. Building a large annotated corpus of english: The penn treebank. *Computational Linguistics (CL)*, 19(2):313–330, 1993.

[32] Ciprian Chelba, Tomas Mikolov, Mike Schuster, Qi Ge, Thorsten Brants, Philipp Koehn, and Tony Robinson. One Billion Word Benchmark for Measuring Progress in Statistical Language Modeling. In *Proc. Conf. of the European Chapter of the Association for Computational Linguistics (EACL)*, pages 263–272. Association for Computational Linguistics, 2013.

[33] Xiang Zhang, Junbo Zhao, and Yann LeCun. Character-level Convolutional Networks for Text Classification. In *Proc. of Int. Conf. on Neural Information Processing Systems (NeurIPS)*, 2015.

[34] 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 *Proc. Conf. of the North American Chapter of the Association for Computational Linguistics (NAACL)*, 2018.

[35] Aaron van den Oord, Nal Kalchbrenner, Oriol Vinyals, Lasse Espeholt, Alex Graves, and Koray Kavukcuoglu. Conditional Image Generation with PixelCNN Decoders. In *Proc. of Int. Conf. on Neural Information Processing Systems (NeurIPS)*, 2016.

[36] Hunter Nisonoff, Junhao Xiong, Stephan Allenspach, and Jennifer Listgarten. Unlocking Guidance for Discrete State-Space Diffusion and Flow Models. In *Proc. Int. Conf. on Learning Representations (ICLR)*, 2025.

[37] Yang Song and Stefano Ermon. Generative Modeling by Estimating Gradients of the Data Distribution. In *Proc. of Int. Conf. on Neural Information Processing Systems (NeurIPS)*, 2019.

[38] Yaron Lipman, Ricky T. Q. Chen, Heli Ben-Hamu, Maximilian Nickel, and Matt Le. Flow Matching for Generative Modeling. In *Proc. Int. Conf. on Learning Representations (ICLR)*, 2023.

[39] Ting Chen, Ruixiang Zhang, and Geoffrey Hinton. Analog Bits: Generating Discrete Data using Diffusion Models with Self-Conditioning. In *Proc. Int. Conf. on Learning Representations (ICLR)*, 2023.

[40] Chen-Hao Chao, Wei-Fang Sun, Bo-Wun Cheng, and Chun-Yi Lee. On Investigating the Conservative Property of Score-Based Generative Models. In *Proc. Int. Conf. on Machine Learning (ICML)*, 2023.

[41] Grigory Bartosh, Dmitry Vetrov, and Christian A. Naesseth. Neural Diffusion Models. In *Proc. Int. Conf. on Machine Learning (ICML)*, 2024.

[42] Ngoc-Trung Tran, Viet-Hung Tran, Ngoc-Bao Nguyen, Linxiao Yang, and Ngai-Man Cheung. Self-supervised GAN: Analysis and Improvement with Multi-class Minimax Game. In *Proc. of Int. Conf. on Neural Information Processing Systems (NeurIPS)*, 2019.

[43] Kaiwen Zheng, Cheng Lu, Jianfei Chen, and Jun Zhu. Improved Techniques for Maximum Likelihood Estimation for Diffusion ODEs. In *Proc. Int. Conf. on Machine Learning (ICML)*, 2023.

[44] Michael S. Albergo and Eric Vanden-Eijnden. Building Normalizing Flows with Stochastic Interpolants. In *Proc. Int. Conf. on Learning Representations (ICLR)*, 2023.

[45] Dongjun Kim, Seungjae Shin, Kyungwoo Song, Wanmo Kang, and Il-Chul Moon. Soft Truncation: A Universal Training Technique of Score-based Diffusion Model for High Precision Score Estimation. In *Proc. Int. Conf. on Machine Learning (ICML)*, 2022.

[46] Prafulla Dhariwal and Alex Nichol. Diffusion Models Beat GANs on Image Synthesis. In *Proc. of Int. Conf. on Neural Information Processing Systems (NeurIPS)*, 2021.

[47] Shane Barratt and Rishi Sharma. A Note on the Inception Score. *Workshop on Theoretical Foundations and Applications of Deep Generative Models at Int. Conf. on Machine Learning (ICML)*, 2018.

[48] Satoshi Hayakawa, Yuhta Takida, Masaaki Imaizumi, Hiromi Wakaki, and Yuki Mitsufuji. Distillation of Discrete Diffusion through Dimensional Correlations. *Machine Learning and Compression Workshop at the Int. Conf. on Neural Information Processing Systems (NeurIPS)*, 2024.

[49] Jingyang Ou, Shen Nie, Kaiwen Xue, Fengqi Zhu, Jiacheng Sun, Zhenguo Li, and Chongxuan Li. Your Absorbing Discrete Diffusion Secretly Models the Conditional Distributions of Clean Data. `arXiv:2406.03736 [cs.LG]`, 2025.

[50] Neta Shaul, Itai Gat, Marton Havasi, Daniel Severo, Anuroop Sriram, Peter Holderrieth, Brian Karrer, Yaron Lipman, and Ricky T. Q. Chen. Flow Matching with General Discrete Paths: A Kinetic-Optimal Perspective. In *Proc. Int. Conf. on Learning Representations (ICLR)*, 2025.

[51] Yixiu Zhao, Jiaxin Shi, Feng Chen, Shaul Druckmann, Lester Mackey, and Scott Linderman. Informed Correctors for Discrete Diffusion Models. `arXiv:2407.21243 [cs.LG]`, 2024.

[52] Guanghan Wang, Yair Schiff, Subham Sekhar Sahoo, and Volodymyr Kuleshov. Remasking Discrete Diffusion Models with Inference-Time Scaling. `arXiv:2503.00307 [cs.LG]`, 2025.

[53] Justin Deschenaux and Caglar Gulcehre. Beyond Autoregression: Fast LLMs via Self-Distillation Through Time. In *Proc. Int. Conf. on Learning Representations (ICLR)*, 2025.

[54] Sander Dieleman, Laurent Sartran, Arman Roshannai, Nikolay Savinov, Yaroslav Ganin, Pierre H. Richemond, Arnaud Doucet, Robin Strudel, Chris Dyer, Conor Durkan, Curtis Hawthorne, Rémi Leblond, Will Grathwohl, and Jonas Adler. Continuous diffusion for categorical data. `arXiv:2211.15089 [cs.CL]`, 2022.

[55] Xiang Lisa Li, John Thickstun, Ishaan Gulrajani, Percy Liang, and Tatsunori B. Hashimoto. Diffusion-LM Improves Controllable Text Generation. In *Proc. of Int. Conf. on Neural Information Processing Systems (NeurIPS)*, 2022.

[56] Ishaan Gulrajani and Tatsunori B. Hashimoto. Likelihood-Based Diffusion Language Models. In *Proc. of Int. Conf. on Neural Information Processing Systems (NeurIPS)*, 2023.

[57] Shujian Zhang, Lemeng Wu, Chengyue Gong, and Xingchao Liu. Language Rectified Flow: Advancing Diffusion Language Generation with Probabilistic Flows. In *Proc. of Annual Conf. of the North American Chapter of the Association for Computational Linguistics (NAACL)*, 2024.

[58] Vincent Hu, Di Wu, Yuki Asano, Pascal Mettes, Basura Fernando, Björn Ommer, and Cees Snoek. Flow Matching for Conditional Text Generation in a Few Sampling Steps. In *Proc. of Conference of the European Chapter of the Association for Computational Linguistics (EACL)*, 2024.

[59] W. N. Cottingham and D. A. Greenwood. An Introduction to the Standard Model of Particle Physics. 2007.

[60] Weiwei Zhang, Jian Sun, and Xiaoou Tang. Cat Head Detection - How to Effectively Exploit Shape and Texture Features. In *Proc. of European Conf. Computer Vision (ECCV)*, 2008.

[61] Prajit Ramachandran, Barret Zoph, and Quoc V. Le. Searching for Activation Functions. arXiv:1710.05941v2 [cs.NE], 2017.

[62] Diederik P. Kingma and Jimmy Ba. Adam: A Method for Stochastic Optimization. In *Proc. Int. Conf. on Learning Representations (ICLR)*, 2015.

[63] Ilya Loshchilov and Frank Hutter. Decoupled Weight Decay Regularization. In *Proc. Int. Conf. on Learning Representations (ICLR)*, 2019.

[64] Anji Liu, Oliver Broadrick, Mathias Niepert, and Guy Van den Broeck. Discrete Copula Diffusion. In *Proc. Int. Conf. on Learning Representations (ICLR)*, 2025.

[65] Laurent Dinh, David Krueger, and Yoshua Bengio. NICE: Non-linear Independent Components Estimation. 2015.

[66] Laurent Dinh, Jascha Sohl-Dickstein, and Samy Bengio. Density Estimation using Real NVP. In *Proc. Int. Conf. on Learning Representations (ICLR)*, 2016.

A Appendix

In this appendix, we provide additional analyses and experiments. Section A.1 presents some theoretical properties of masked diffusion models (MDM). Section A.2 offers an analysis of the proposed Partial masking scheme (Prime). Section A.3 outlines a number of architectural designs of MDM augmented with Prime (i.e., MDM-Prime). Section A.4 details the experimental configurations. Section A.5 reports additional experimental results. Section A.6 summarizes the limitations of this work. Finally, Section A.7 discusses the potential impacts of this work. The following table of contents summarizes the structure of the main manuscript and this appendix.

Table of Contents

A.1 Analyses of Masked Diffusion Processes

In this section, we examine two properties of MDM. In Section A.1.1, we derive an analytical expression for the expected number of idle steps in the reverse diffusion process. In Section A.1.2, we show that the mutual information between the latent variables and data exhibits a linearly decaying trend with respect to the scheduling function over time.

A.1.1 Expected Number of Idle Steps

In Section 1, we show that MDM may have significant number of idle steps during the sampling process. In this section, we derive a closed-form formula for calculating the expected number of idle steps and provide the reason why Prime must reduce this number.

Proposition A.1. *Let L be the token sequence length, T be the total number of discretized timesteps for the sampling process, and $\alpha_t \in [0,1]$ be a strictly decreasing scheduling function in $t \in [0,1]$. Suppose the sampling timesteps are indexed by $k \in \{0, \cdots, T-1\}$, the expected number of idle steps η, i.e., the entire token sequence remains unchanged between consecutive steps, is given by:*

$$\eta = \sum_{k=0}^{T-1} \left[1 - \left(\alpha_{1-\frac{k+1}{T}} - \alpha_{1-\frac{k}{T}}\right)\right]^L. \tag{A1}$$

Proof. At reverse step $k \to k+1$, a token remains unchanged if it is already unmasked at $t = 1 - \frac{k}{T}$ or it is masked at $t = 1 - \frac{k}{T}$ while remaining masked $t = 1 - \frac{k+1}{T}$. According to the forward diffusion process, the probability that a single token remains unchanged is given by:

$$\underbrace{\alpha_{1-\frac{k}{T}}}_{(i)} + \underbrace{(1 - \alpha_{1-\frac{k}{T}})}_{(ii)} \cdot \underbrace{\frac{1-\alpha_{1-\frac{k+1}{T}}}{1-\alpha_{1-\frac{k}{T}}}}_{(iii)} = 1 - \left(\alpha_{1-\frac{k+1}{T}} - \alpha_{1-\frac{k}{T}}\right), \tag{A2}$$

where (i) is the probability of observing an unmasked token at time $t = 1 - \frac{k}{T}$ (i.e., Eq. (1)), (ii) is the probability of observing a masked token at time $t = 1 - \frac{k}{T}$ (i.e., Eq. (1)), and (iii) is the probability that the masked token remains masked at time $t = 1 - \frac{k+1}{T}$ (i.e., Eq. (3)). For all L tokens to remain unchanged, the joint probability is $\left[1 - \left(\alpha_{1-\frac{k+1}{T}} - \alpha_{1-\frac{k}{T}}\right)\right]^L$. Due to the linearity of expectation, the expected number of idle steps can be calculated by accumulating Eq. (A2) for all $k \in \{0, \cdots, T-1\}$, and is written as:

$$\sum_{k=0}^{T-1} \left[1 - \left(\alpha_{1-\frac{k+1}{T}} - \alpha_{1-\frac{k}{T}}\right)\right]^L. \tag{A3}$$

□

Eq. (A1) enables the computation of idle steps given the sequence length L, the number of discretized steps T, and the scheduling function α_t. Consider the commonly used linear schedule $\alpha_t = 1 - t$ [9, 10]. Substituting the difference $\alpha_{1-\frac{k+1}{T}} - \alpha_{1-\frac{k}{T}} = \frac{1}{T}$ into the formula yields:

$$\sum_{k=0}^{T-1} \left(1 - \frac{1}{T}\right)^L = T\left(1 - \frac{1}{T}\right)^L \overset{(i)}{\approx} Te^{-\frac{L}{T}}, \tag{A4}$$

where (i) holds when T is large. For example, when $T = 1{,}024$ and $L = 1{,}024$, we obtain $Te^{-L/T} \approx 1024 \cdot e^{-1} \approx 376$, indicating that approximately 37% of the sampling steps are idle in the discrete diffusion process.

Moreover, since $1 - \left(\alpha_{1-\frac{k+1}{T}} - \alpha_{1-\frac{k}{T}}\right) \in [0,1]$, the quantity in Eq. (A1) decreases as the token sequence length L increases. In MDM-Prime, the sequence length is extended by a factor of $\ell > 1$ due to the introduction of sub-token sequences, and the number of idle steps η_{Prime} is given by:

$$\eta_{\text{Prime}} = \sum_{k=0}^{T-1} \left[1 - \left(\alpha_{1-\frac{k+1}{T}} - \alpha_{1-\frac{k}{T}}\right)\right]^{L\times\ell}. \tag{A5}$$

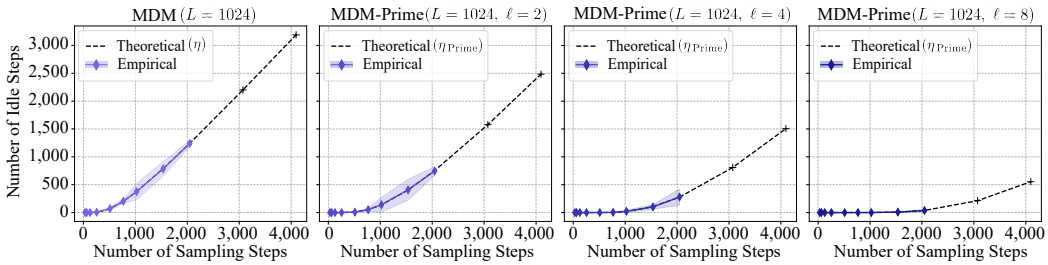


Figure A1: Comparison between idle steps obtained from simulation results and analytical computation. Solid curves and their corresponding shaded areas represent the mean and variance from ten independent simulation runs. Dashed lines indicate the theoretical values.

Given identical T and α_t, η_{Prime} in Eq. (A5) is always smaller than η in Eq. (A1). Therefore, employing Prime must result in fewer idle steps. To support this observation and verify the correctness of Eqs. (A1) and (A5), Fig. A1 compares our simulation results with the analytical values computed using η and η_{Prime}. The results demonstrate agreement between the theoretical and empirical estimates.

Based on the above definition of expected idle steps, we define the idle step ratio (ISR) as the proportion of expected idle steps relative to the total number of sampling steps T:

$$\frac{1}{T}\sum_{k=0}^{T-1}\left[1-\left(\alpha_{1-\frac{k+1}{T}}-\alpha_{1-\frac{k}{T}}\right)\right]^{L\times\ell}. \tag{A6}$$

ISR quantifies the model utilization in diffusion processes. To evaluate it, we can fix the scheduling function α_t and choose a large T (i.e., to approximate the continuous-time limit $T\to\infty$). In our experiments presented in Section 4, we set T as $1,024$.

A.1.2 Mutual Information Scheduling

In this section, we show that the MDM framework implements mutual information scheduling (i.e., $I(\boldsymbol{x}_t;\boldsymbol{x}_0)=\alpha_t H(\boldsymbol{x}_0)$) under the assumption that the latent variables $\boldsymbol{x}_t=[x_t^1,\cdots,x_t^L]$ are independent across dimensions, i.e., $p(\boldsymbol{x}_t)=\prod_{i=1}^L p(x_t^i)$. This relationship was initially established in prior works [20, 21], where a linear scheduling function $\alpha_t=1-t$ was considered as a special case. In our work, we extend this result to arbitrary scheduling functions $\alpha_t\in[0,1]$, as formally stated in **Proposition** A.2. Since the independence assumption may not hold in practice, previous work [20] reports this relationship as an approximation (i.e., $I(\boldsymbol{x}_t;\boldsymbol{x}_0)\approx\alpha_t H(\boldsymbol{x}_0)$).

Proposition A.2. *Let $\alpha_t\in[0,1]$ be the scheduling function and $q(\boldsymbol{x}_t|\boldsymbol{x}_0)=\prod_{i=1}^L q(x_t^i|x_0^i)$ be the distribution defined in Eq. (1). Assume $p(\boldsymbol{x}_t)=\prod_{i=1}^L p(x_t^i)$. Then, the mutual information between $\boldsymbol{x}_0$ and $\boldsymbol{x}_t$ satisfies:*

$$I(\boldsymbol{x}_t;\boldsymbol{x}_0)=\alpha_t H(\boldsymbol{x}_0). \tag{A7}$$

Proof. The diffusion process masks each element x_0^i with probability $1-\alpha_t$, resulting in the vector $\boldsymbol{x}_t\in\tilde{\mathcal{X}}^L$. Since x_t^i and x_t^j are independent for $i,j\in[1,\cdots,L]$ (by assumption) and the masking is applied independently, the mutual information can be decomposed as follows:

$$I(\boldsymbol{x}_t;\boldsymbol{x}_0)=H(\boldsymbol{x}_t)-H(\boldsymbol{x}_t|\boldsymbol{x}_0)=\sum_{i=1}^{L}H(x_t^i)-H(x_t^i|x_0^i) \tag{A8}$$

We refer to $H(x_t^i)$ and $H(x_t^i|x_0^i)$ as the *element-wise entropy* and *conditional entropy*, respectively. The following proof expands both terms to derive Eq. (A7).

(i) Conditional Entropy $H(x_t^i|x_0^i)$. Given x_0^i, the distribution over x_t^i is:

$$q(x_t^i|x_0^i)=\alpha_t\,\delta_{x_0^i}(x_t^i)+(1-\alpha_t)\,\delta_{\mathtt{m}}(x_t^i). \tag{A9}$$

Its entropy can be expressed as the entropy of a Bernoulli:

$$H(x_t^i|x_0^i)=H(\text{Bernoulli}(\alpha_t))=-\alpha_t\log\alpha_t-(1-\alpha_t)\log(1-\alpha_t). \tag{A10}$$

(ii) Element-wise Entropy $H(x_t^i)$. The element-wise distribution $p_t(x_t^i)$ can be written as follows:

$$\begin{aligned} p_t(x_t^i) &= \sum_{x_0^i \in \mathcal{X}} p_0(x_0^i) \cdot q(x_t^i | x_0^i) \\ &= \sum_{x_0^i \in \mathcal{X}} p_0(x_0^i) \left[\alpha_t \delta_{x_0^i}(x_t^i) + (1-\alpha_t)\delta_{\mathtt{m}}(x_t^i) \right] = \begin{cases} \alpha_t \, p_0(x_t^i), & \text{if } x_t^i \in \mathcal{X}, \\ 1-\alpha_t, & \text{if } x_t^i = \mathtt{m} \end{cases}. \end{aligned} \tag{A11}$$

The element-wise entropy can be expanded using Eq. (A11) as follows:

$$\begin{aligned} H(x_t^i) &= -\sum_{x \in \tilde{\mathcal{X}}} p_t(x) \log p_t(x) \\ &= -\left(\sum_{x \in \mathcal{X}} \alpha_t p_0(x) \log(\alpha_t p_0(x)) \right) - (1-\alpha_t)\log(1-\alpha_t) \\ &= -\left(\alpha_t \sum_{x \in \mathcal{X}} p_0(x) \log p_0(x) \right) - \alpha_t \log \alpha_t - (1-\alpha_t)\log(1-\alpha_t) \\ &= \alpha_t H(x_0^i) + H(\text{Bernoulli}(\alpha_t)). \end{aligned} \tag{A12}$$

According to Eqs. (A10) and (A12), the mutual information is expressed as follows:

$$\begin{aligned} I(\boldsymbol{x}_t; \boldsymbol{x}_0) &= \sum_{i=1}^{L} H(x_t^i) - H(x_t^i | x_0^i) \\ &= \sum_{i=1}^{L} \alpha_t H(x_0^i) + H(\text{Bernoulli}(\alpha_t)) - H(\text{Bernoulli}(\alpha_t)) \\ &= \sum_{i=1}^{L} \alpha_t H(x_0^i) = \alpha_t \sum_{i=1}^{L} H(x_0^i) = \alpha_t H(\boldsymbol{x}_0). \end{aligned} \tag{A13}$$

□

A.2 Analyses of the Partial Mask Scheme

In this section, we provide thorough analyses of MDM-Prime. Section A.2.1 derives a formula for computing the negative log-likelihood (NLL). Section A.2.2 shows that Prime yields a positive number of intermediate states. Section A.2.3 examines the correctness of the *carry-over* parameterization for Prime. Finally, Section A.2.4 offers a guideline for selecting the target length ℓ.

A.2.1 Negative Log Likelihood Calculation

In Section 3, we introduce an invertible function $f : \mathcal{X}^L \to \mathcal{Y}^{\ell \times L}$ to transform between two vectors, $\boldsymbol{x}_0$ and $\boldsymbol{y}_0$. In this section, we detail the computation of the expected negative log-likelihood (NLL) of MDM after applying this transformation. We begin by revisiting the derivation of Eq. (4), and then extend it to our proposed objective in Eq. (5).

The expected NLL of $\boldsymbol{x}_0$ is expressed as $\mathbb{E}_{p_{\text{data}}(\boldsymbol{x}_0)}[-\log p_\theta(\boldsymbol{x}_0)]$, which can be approximated using a variational upper bound [9, 10, 12] as expressed as follows:

$$\begin{aligned} \mathbb{E}_{p_{\text{data}}(\boldsymbol{x}_0)}[-\log p_\theta(\boldsymbol{x}_0)] &\leq \mathbb{E}_{p_{\text{data}}(\boldsymbol{x}_0)} \left[\int_0^1 \frac{\alpha_t'}{1-\alpha_t} \mathbb{E}_{q(\boldsymbol{x}_t|\boldsymbol{x}_0)} \left[\log p_\theta(\boldsymbol{x}_0 | \boldsymbol{x}_t) \right] dt \right] \\ &\overset{(i)}{=} \mathbb{E}_{p_{\text{data}}(\boldsymbol{x}_0)} \left[\int_0^1 \frac{\alpha_t'}{1-\alpha_t} \mathbb{E}_{q(\boldsymbol{x}_t|\boldsymbol{x}_0)} \left[\sum_{i=1}^{L} \log p_\theta(x_0^i | \boldsymbol{x}_t) \right] dt \right], \end{aligned} \tag{A14}$$

where (i) is derived from $p_\theta(\boldsymbol{x}_0 | \boldsymbol{x}_t) = \prod_{i=1}^{L} p_\theta(x_0^i | \boldsymbol{x}_t)$. Due to the introduction of f, the expected NLL expressed by MDM-Prime is written as $\mathbb{E}_{p_{\text{data}}(\boldsymbol{x}_0)}[-\log p_\theta \circ f(\boldsymbol{x}_0)]$, where $p_\theta \circ f(\boldsymbol{x}_0)$ is the

parameterized pmf as discussed in Section 3.1. This expectation can be estimated via Eq. (5), as derived below:

$$\begin{aligned}\mathbb{E}_{p_{\text{data}}(\boldsymbol{x}_0)}[-\log p_\theta \circ f(\boldsymbol{x}_0)] &= \mathbb{E}_{p_{\text{data}}\circ f^{-1}(\boldsymbol{y}_0)}[-\log p_\theta(\boldsymbol{y}_0)] \\ &\leq \mathbb{E}_{p_{\text{data}}\circ f^{-1}(\boldsymbol{y}_0)}\left[\int_0^1 \frac{\alpha_t'}{1-\alpha_t}\mathbb{E}_{q(\boldsymbol{y}_t|\boldsymbol{y}_0)}\left[\log p_\theta(\boldsymbol{y}_0|\boldsymbol{y}_t)\right] dt\right] \\ &\overset{(i)}{=} \mathbb{E}_{p_{\text{data}}\circ f^{-1}(\boldsymbol{y}_0)}\left[\int_0^1 \frac{\alpha_t'}{1-\alpha_t}\mathbb{E}_{q(\boldsymbol{y}_t|\boldsymbol{y}_0)}\left[\sum_{i=1}^{L}\log p_\theta(\boldsymbol{y}_0^i|\boldsymbol{y}_t)\right] dt\right],\end{aligned} \tag{A15}$$

where (i) is due to $p_\theta(\boldsymbol{y}_0|\boldsymbol{y}_t) = \prod_{i=1}^{L} p_\theta(\boldsymbol{y}_0^i|\boldsymbol{y}_t)$ following the derivation in Eq. (A14). To sample data points from the distribution $p_{\text{data}} \circ f^{-1}$, one can first sample $\boldsymbol{x}_0 \sim p_{\text{data}}$ and then transform it to $\boldsymbol{y}_0 = f(\boldsymbol{x}_0)$ according to the change-of-variable principle for probability distributions. This formulation enables us to estimate the expected NLL of MDM-Prime.

A.2.2 Number of Intermediate States

In Section 3.1, we claim that the number of intermediate states can be quantified as $|\tilde{\mathcal{Y}}^\ell| - |\tilde{\mathcal{X}}| = (b+1)^\ell - (C+1)$ and that this number is always positive. To verify this, we provide **Proposition** A.3.

Proposition A.3. *Let $\tilde{\mathcal{Y}} = \{0, \cdots, b-1\} \cup \{\mathtt{m}\}$ and $\tilde{\mathcal{X}} = \{0, \cdots, C-1\} \cup \{\mathtt{m}\}$. Let $\ell > 1$ be a positive integer and let $b = \lceil \sqrt[\ell]{C} \rceil$. The number of intermediate states is a positive integer:*

$$|\tilde{\mathcal{Y}}^\ell| - |\tilde{\mathcal{X}}| > 0. \tag{A16}$$

Proof. The number of original tokens with the mask token is:

$$|\tilde{\mathcal{X}}| = C + 1. \tag{A17}$$

The total number of possible sub-token sequences (including the mask) is:

$$|\tilde{\mathcal{Y}}^\ell| = (b+1)^\ell \overset{(i)}{=} \sum_{k=0}^{\ell}\binom{\ell}{k}b^{\ell-k} = b^\ell + \underbrace{\binom{\ell}{1}b^{\ell-1} + \cdots}_{>0} + 1 > b^\ell + 1, \tag{A18}$$

where (i) is derived by the binomial theorem. Since $b = \lceil \sqrt[\ell]{C} \rceil$ by construction, we have $b^\ell \geq C$, and thus:

$$(b+1)^\ell > b^\ell + 1 \geq C + 1. \tag{A19}$$

By rearranging this equation, we conclude that $(b+1)^\ell - (C+1) > 0$. As a result,

$$|\tilde{\mathcal{Y}}^\ell| - |\tilde{\mathcal{X}}| = (b+1)^\ell - (C+1) > 0. \tag{A20}$$

□

To illustrate how the number of intermediate states (i.e., $|\tilde{\mathcal{Y}}^\ell| - |\tilde{\mathcal{X}}|$) may increase with ℓ, we define a function $M(\ell, C) = (b+1)^\ell - (C+1)$ and evaluate its value under different ℓ. As a concrete example, consider $C = 256$ and $\ell \in \{2, 4, 8\}$. Substituting these values yields $M(2, 256) = 32$, $M(4, 256) = 368$, and $M(8, 256) = 6304$. This example highlights that Prime can produce a substantial number of intermediate states by selecting ℓ. This growth of $M(\ell, C)$ motivates our design choice for the embedding lookup table discussed in Section 3.2, where we mention the infeasibility of directly modeling the embeddings for $\boldsymbol{y}_t^i$.

A.2.3 Carry-over Parameterization

In Section 3.2, we introduced the carry-over parameterization for MDM-Prime. In this section, we justify this approach by presenting **Proposition** A.4 and discussing its practical implementation.

Proposition A.4. *Let $\mathcal{V}(\boldsymbol{y}_t^i)$ denote the set of $\boldsymbol{y}_0^i$ such that each $y_0^{i,j}$ is consistent with $y_t^{i,j} \in \mathcal{Y}$, i.e.,*

$$\mathcal{V}(\boldsymbol{y}_t^i) \triangleq \{\boldsymbol{y}^i = [y^{i,1}, \cdots, y^{i,\ell}] \in f(\mathcal{X}) \;\; s.t. \;\; (y^{i,j} = y_t^{i,j}) \vee (y_t^{i,j} = \mathtt{m})\}. \tag{A21}$$

Given the parameterized distribution $p_\theta(\boldsymbol{y}_0^i|\boldsymbol{y}_t)$ expressed as follows:

$$p_\theta(\boldsymbol{y}_0^i|\boldsymbol{y}_t) = \begin{cases} \frac{\exp(E_\theta(\boldsymbol{y}_0^i|\boldsymbol{y}_t))}{\sum_{\boldsymbol{y}^i \in \mathcal{V}(\boldsymbol{y}_t^i)} \exp(E_\theta(\boldsymbol{y}^i|\boldsymbol{y}_t))}, & \text{if } \boldsymbol{y}_0^i \in \mathcal{V}(\boldsymbol{y}_t^i), \\ 0, & \text{if } \boldsymbol{y}_0^i \notin \mathcal{V}(\boldsymbol{y}_t^i), \end{cases} \tag{A22}$$

the marginal distribution $p_\theta(y_0^{i,j}|\boldsymbol{y}_t)$ of Eq. (A22) satisfies the carry-over condition for all position i, j where $y_t^{i,j} \in \mathcal{Y}$:

$$p_\theta(y_0^{i,j}|\boldsymbol{y}_t) = \sum_{y_0^{i,1}, \cdots, y_0^{i,j-1}, y_0^{i,j+1}, \cdots, y_0^{i,\ell} \in \mathcal{Y}} p_\theta(y_0^{i,1}, \cdots, y_0^{i,\ell}|\boldsymbol{y}_t) = \delta_{y_t^{i,j}}(y_0^{i,j}). \tag{A23}$$

Proof. Given i, j such that $y_t^{i,j} \in \mathcal{Y}$, according to Eq. (A21), $y_0^{i,j} = y_t^{i,j}$ is a necessary condition for non-zero probability:

$$p_\theta(\boldsymbol{y}_0^i|\boldsymbol{y}_t) > 0 \;\; \Rightarrow \;\; y_0^{i,j} = y_t^{i,j}. \tag{A24}$$

In other word, if $y_0^{i,j} \neq y_t^{i,j}$ for any j is observed, then $p_\theta(\boldsymbol{y}_0^i|\boldsymbol{y}_t) = 0$. This indicates that the marginal $p_\theta(y_0^{i,j}|\boldsymbol{y}_t) = \sum_{y_0^{i,1}, \cdots, y_0^{i,j-1}, y_0^{i,j+1}, \cdots, y_0^{i,\ell} \in \mathcal{Y}} p_\theta(\boldsymbol{y}_0^i|\boldsymbol{y}_t) = 0$ for any $y_0^{i,j} \neq y_t^{i,j}$.

Since $\sum_{y_0^{i,j} \in \mathcal{Y}} p_\theta(y_0^{i,j}|\boldsymbol{y}_t) = \sum_{\boldsymbol{y}_0^i \in \mathcal{Y}^\ell} p_\theta(\boldsymbol{y}_0^i|\boldsymbol{y}_t) = 1$ by Eq. (A22), $p_\theta(y_0^{i,j}|\boldsymbol{y}_t) = 1$ for $y_0^{i,j} = y_t^{i,j}$. Therefore, the marginal distribution corresponds to the Kronecker delta function as follows:

$$p_\theta(y_0^{i,j}|\boldsymbol{y}_t) = \begin{cases} 1, & \text{if } y_0^{i,j} = y_t^{i,j}, \\ 0, & \text{if } y_0^{i,j} \neq y_t^{i,j}. \end{cases} = \delta_{y_t^{i,j}}(y_0^{i,j}). \tag{A25}$$

□

To understand this parameterization method, we examine its implementation in practice. The goal is to ensure that the marginal distribution $p_\theta(y_0^{i,j}|\boldsymbol{y}_t) = \sum_{y_0^{i,1}, \cdots, y_0^{i,j-1}, y_0^{i,j+1}, \cdots, y_0^{i,\ell} \in \mathcal{Y}} p_\theta(\boldsymbol{y}_0^i|\boldsymbol{y}_t)$ is zero whenever the candidate state $\boldsymbol{y}_0^i$ contains any $y_0^{i,j}$ that is inconsistent with $y_t^{i,j}$ (i.e., when $y_0^{i,j} \neq y_t^{i,j}$, as discussed in the proof of **Proposition** A.3). Since the marginal probability is a sum over probabilities, it follows that each individual probability $p_\theta(\boldsymbol{y}_0^i|\boldsymbol{y}_t)$ should be zero for $\boldsymbol{y}_0^i$ with $y_0^{i,j}$ that is inconsistent with $y_t^{i,j}$. To enforce this, we use filters with C entries to exclude the output logits of invalid $\boldsymbol{y}_0^i$. For example, suppose $C = 7$, $\ell = 3$, and $b = 2$. If $\boldsymbol{y}_t^i = (y_t^{i,1}, y_t^{i,2}, y_t^{i,3}) = (1, \mathtt{m}, 0)$, we create ℓ filters (i.e., Filters 1-3) as follows:

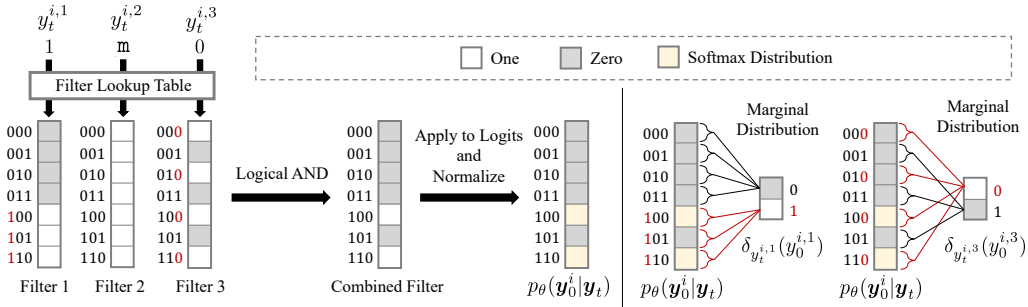
- For $y_t^{i,1} = 1$, Filter 1 excludes $\boldsymbol{y}_0^i$ where the first position is not 1.
- For $y_t^{i,2} = \mathtt{m}$, Filter 2 excludes no state since no condition is needed to be satisfied at this position.
- For $y_t^{i,3} = 0$, Filter 3 excludes $\boldsymbol{y}_0^i$ where the third position is not 0.

We then combine these element-wise filters using a logical AND operation, since any violation of an element-wise condition leads to an invalid $\boldsymbol{y}_0^i$ that has inconsistency with $y_t^{i,j}$. The resulting combined filter excludes $\boldsymbol{y}_0^i \notin \mathcal{V}(\boldsymbol{y}_t^i)$, retaining only $\boldsymbol{y}_0^i \in \mathcal{V}(\boldsymbol{y}_t^i)$. Subsequently, we apply a softmax over the logits corresponding to $\boldsymbol{y}_0^i \in \mathcal{V}(\boldsymbol{y}_t^i)$ to define the probability distribution $p_\theta(\boldsymbol{y}_0^i|\boldsymbol{y}_t)$. An illustrative example of this procedure is depicted in Fig. A2.

For efficient implementation, we precompute and cache a lookup table with size $|\tilde{\mathcal{Y}}| \times |f(\mathcal{X})| = (b+1) \times C$, which contains all possible filters for $y_t^{i,j}$. During the forward pass, we query this lookup table using $y_t^{i,j}$ to retrieve the corresponding filters, which are then applied to the logits to zero out the probability of $\boldsymbol{y}_0^i \notin \mathcal{V}(\boldsymbol{y}_t^i)$.

A.2.4 Analysis of the Target Length

Prime introduces a parameter ℓ that controls the target length of sub-token sequences. This section analyzes its impact on performance and provides a practical guideline for selecting an appropriate value of ℓ.



(a) Computation Procedure (b) Carry-over Condition

Figure A2: An illustration of (a) the computation procedure of the carry-over parameterization and (b) the corresponding carry-over condition (i.e., Eq. (A23)) on the marginal distributions. In this example, $C = 7$, $\ell = 3$, and $b = 2$ (i.e., binary encodings). In (a), the filters are precomputed and stored in a lookup table indexed by $y_t^{i,j}$. These filters exclude logits corresponding to $\boldsymbol{y}_0^i$ with $y_0^{i,j}$ that are inconsistent with $y_t^{i,j} \in \mathcal{Y}$ (i.e., $y_t^{i,1} = 1$ and $y_t^{i,3} = 0$). For each position $j \in \{1, \cdots, \ell\}$, a filter is queried from the lookup table using $y_t^{i,j}$. Then, a combined filter is then constructed by applying a logical AND operation across ℓ filters. Finally, the softmax distribution $p_\theta(\boldsymbol{y}_0^i|\boldsymbol{y}_t)$ is established by normalizing the filtered logits. In (b), the resulting distribution $p_\theta(\boldsymbol{y}_0^i|\boldsymbol{y}_t)$ satisfies the carry-over condition for $y_t^{i,j} \in \mathcal{Y}$ as defined in Eq. (A23).

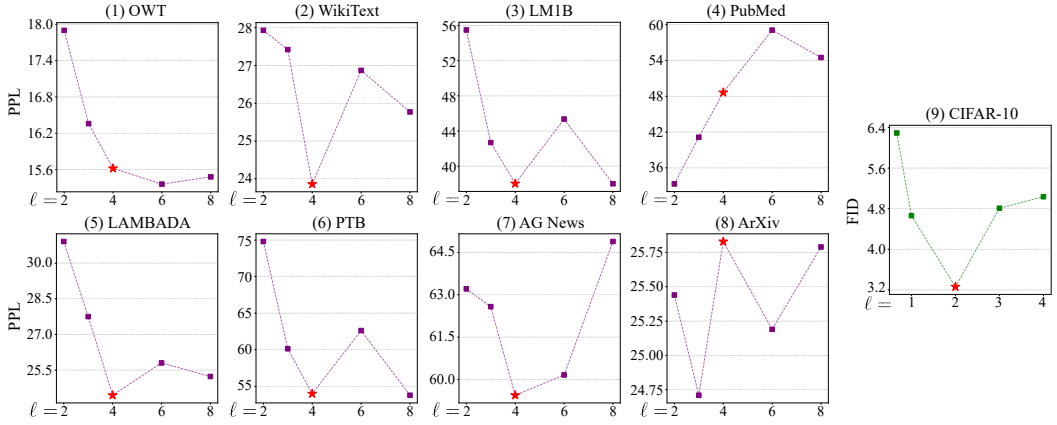


Figure A4: (1)-(8) PPL of MDLM-Prime with different ℓ evaluated on OWT and seven zeroshot datasets. (9) FID of MDM-Prime across varying ℓ evaluated on CIFAR-10. Lower values indicate better performance. Red stars highlight the ℓ values corresponding to the elbow points of the ISR curves (see Fig. A3). These points yield near-optimal performance across most evaluations.

Section 4 demonstrates that, although increasing ℓ generally correlates with improved performance, the relationship is not strictly monotonic, i.e., larger values of ℓ do not always lead to better results. As larger values of ℓ typically induce more intermediate states (see Section A.2.2), we hypothesize that the associated increase in learning complexity may degrade performance under fixed model capacity (e.g., model size). This observation motivates our development of a strategy for identifying an effective ℓ.

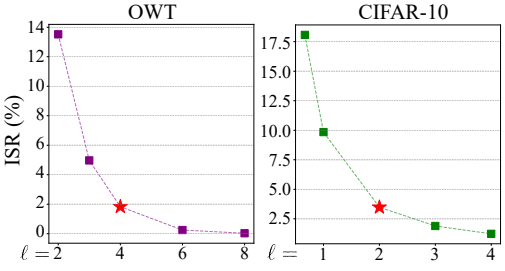


Figure A3: ISR under different choices of ℓ. The elbow points are highlighted using red stars.

Through empirical analysis across both image and text datasets, we find that selecting ℓ near the elbow point of the Idle Step Ratio (ISR) curve (i.e., Eq. (A6)) yields strong performance. Fig. A3 presents ISR curves for various values of ℓ, with elbow points indicated by red stars. The relationship between ISR elbow points and MDM-Prime's

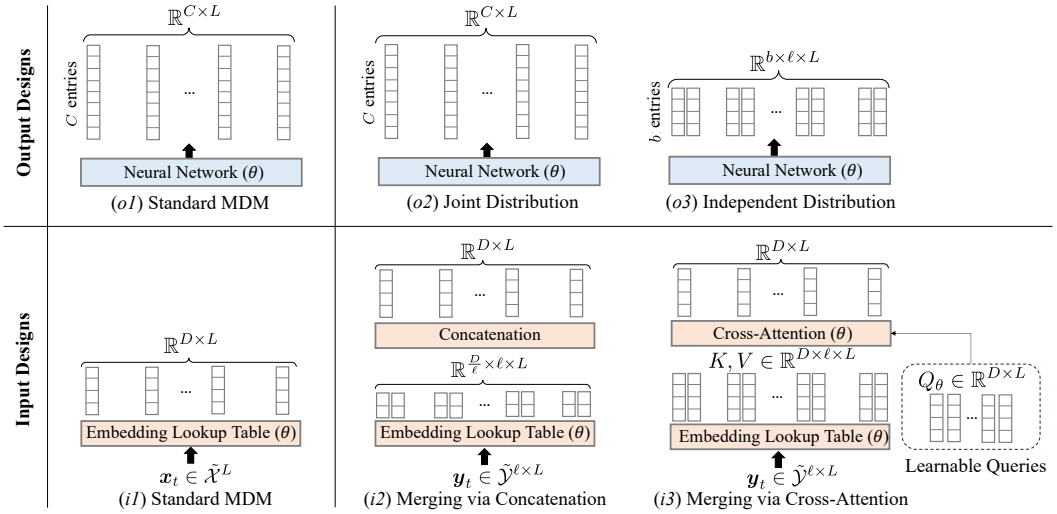


Figure A5: Comparison of input and output layer designs in MDM and MDM-Prime. The top row illustrates three types of output layer designs: (o1) standard MDM, (o2) MDM-Prime with joint distribution, and (o3) MDM-Prime with independent distribution. The bottom row shows three embedding layer designs: (i1) standard MDM embedding layer, (i2) MDM-Prime embedding layer with concatenation, and (i3) MDM-Prime embedding layer with Perceiver [25] cross-attention.

performance is depicted in Fig. A4, where these points align with near-optimal results across most evaluations. Based on the above findings, we recommend $\ell = 2$ for image datasets and $\ell = 4$ for text datasets.

A.3 Model Architecture Designs

This section compares a number of architectural variants of MDM-Prime, with a focus on the design choices for the output logit layer and the input embedding layer. For reference, the standard MDM architecture is shown in Figs. A5 (o1) and (i1), which illustrate its output logit and embedding layers, respectively. In standard MDM, the output layer produces L logits with C entries, while the embedding layer processes inputs $\boldsymbol{x}_t \in \tilde{\mathcal{X}}^L$ to produce L D-dimensional embeddings.

Output Logit Layer. In Section 3.2, we introduce two alternative designs for the output layer. The first design assumes independence among sub-tokens, leading to a factorized form of the probability distribution: $p_\theta(\boldsymbol{y}_0^i|\boldsymbol{y}_t) = \prod_{j=1}^{\ell} p_\theta(y_0^{i,j}|\boldsymbol{y}_t)$. This formulation can be implemented using $\ell \times L$ logits, each with b entries, as illustrated in Fig. A5 (o3). The second design (i.e., Fig. A5 (o2)) models a joint distribution over sub-token sequences, producing L logits with C entries, consistent with the standard MDM architecture shown in Fig. A5 (o1). As discussed in Section 3.2, we adopt the joint distribution design over the factorized alternative (Fig. A5 (o3)) due to two key limitations of the latter: (1) the independence assumption, and (2) the potential to generate invalid samples. These issues hinder the model's ability to capture complex data distributions (see Fig. 4) and limit its applicability to real-world generative tasks.

Embedding Layer. As discussed in Section 3.2, creating an embedding lookup table for $\boldsymbol{y}_t^i$ is impractical due to the resulting growth in the number of parameters in it (also see Section A.2.2). To address this, we model sub-token embeddings by creating a lookup table for $y_t^{i,j} \in \tilde{\mathcal{Y}}$. Since the input and output sequences differ in length, we introduce a simple merging strategy based on concatenation, as illustrated in Fig. A5 (i2). In this approach, each sub-token is embedded into a vector of size $\frac{D}{\ell}$, and the ℓ sub-token embeddings are then concatenated to form D-dimensional token-level embeddings.

Fig. A5 (i3) shows an alternative design based on Perceiver [25], which employs cross-attention with learnable latent queries to merge sub-token embeddings. In this design, each sub-token is embedded into a D-dimensional vector, producing $\ell \times L$ sub-token embeddings. These sub-token embeddings

are then used as the key and value matrices in a cross-attention layer, while a learnable query matrix $Q_\theta \in \mathbb{R}^{D\times L}$ retrieves the token-level embeddings. The resulting outputs are L D-dimensional token embeddings. Although this approach is well-established for processing high-dimensional inputs in transformer-based models, it introduces substantial computational overhead [25]. Moreover, empirical results in Appendix A.5.1 show that it does not lead to performance gains in either text or image generation tasks. Based on these observations, we adopt the simple concatenation-based merging strategy in MDM-Prime's embedding layer.

A.4 Experimental Setups

In this section, we provide additional implementation details and the hyperparameter settings used in our experiments. The configurations for the two-dimensional synthetic experiment, image generation experiment, and text generation experiment are presented in Sections A.4.1, A.4.2, and A.4.3, respectively.

A.4.1 Two-Dimensional Example

Dataset. The experiment in Fig. 4 is conducted on a synthetic dataset constructed by converting pixel values from an image in the Cat dataset [60] into a two-dimensional probability histogram. The image is first cropped to 512×512 pixels and converted to grayscale. A probability distribution is then constructed according to the normalized pixel intensity values. In this example, $\mathcal{X}^L = \{0, \dots, 511\}^2$, where $L = 2$ and $C = 512$. The sample $\boldsymbol{x}_0 \in \mathcal{X}^L$ represents the coordinate of the figure (512×512).

Training and Implementation Details. The network architecture is a four-layered multilayer perceptron (MLP) with hidden dimension 512 and Swish [61] as its activation function. The models are trained using the Adam optimizer [62] with a learning rate of 1×10^{-3} and a batch size of $4,096$. The training is performed on a single NVIDIA A40 GPU with 48 GB memory.

A.4.2 Image Generation

Datasets and Evaluation Methods. The experiments described in Section 4 are conducted on the CIFAR-10 [15] and ImageNet-32 [16] datasets. For both datasets, $L = 32 \times 32 \times 3$ and $C = 256$. The CIFAR-10 training set contains $50,000$ images, while the ImageNet-32 dataset comprises $1,281,149$ training images and $49,999$ validation images. The sample quality is assessed using the Fréchet Inception Distance (FID) [14] and Inception Score (IS) [47], implemented via the `torchmetrics.image.fid` and `torchmetrics.image.inception` libraries, respectively. The corrector steps [22, 23] are adopted during sampling. For CIFAR-10, FID is computed using the training set as the reference distribution, whereas for ImageNet-32, the validation set serves as the reference distribution. This evaluation protocol is consistent with prior works [17–20, 22, 23, 35–45].

Training and Implementation Details. Our model architecture follows [23] to adapt the ablated diffusion model (ADM) [46], which has a U-Net structure with a symmetric design. It comprises a downsampling module, a bottleneck module, and an upsampling module. Each module contains multiple residual convolutional blocks, with attention layers incorporated at selected resolutions. The downsampling and upsampling modules use an embedding dimension of 96 and channel multipliers of $[3, 4, 4]$, respectively. The depth of each block is set to 5. For attention layers, the number of head channels is set to 64. The network is optimized using the Adam optimizer [62] with $\beta_1 = 0.9$, $\beta_2 = 0.999$, and a learning rate of 1×10^{-4}. The scheduling function is defined as a third-order polynomial, given by $\alpha_t = (1 - t)^3$. The model is trained with a batch size of 512 for $4,250$ epochs on CIFAR-10 and $1,000$ epochs on ImageNet-32. The training is performed on eight NVIDIA L40 GPUs with 48 GB memory.

A.4.3 Text Generation

Datasets and Evaluation Methods. The training is performed on the OpenWebText (OWT) dataset [11]. Text data is tokenized using the GPT-2 tokenizer [5], which defines $L = 1,024$ and $C = 50,257$. Since some sequences in OWT exceed the maximum length L, they are concatenated with the `<eos>` token as a separator and then wrapped into segments of length $1,024$. In addition, the first and last tokens of each sequence are set to `<eos>`. As OWT does not include an official validation split, we follow the procedure in [9] by reserving the last $100,000$ samples for validation. The above data preprocessing and evaluation setup is consistent with [9].

Table A1: Zero-shot validation PPL evaluated on seven textual datasets. Lower values correspond to better performance. Methods marked with * incorporate an autoregressive formulation. MDLM-Prime with carry-over parameterization achieves improved results on LAMBADA and PubMed.

	Carry-over	LAMBADA	WikiText	PTB	LM1B	AG News	PubMed	ArXiv
ARM* [9]	-	51.28	25.75	82.05	51.25	**52.09**	49.01	41.73
MDLM [9]	✓	≤47.52	≤32.83	≤95.26	≤67.01	≤61.15	≤41.89	≤37.37
MDLM-Prime ($\ell = 2$)		≤34.61	≤31.94	≤91.85	≤65.25	≤63.74	≤63.51	≤25.95
MDLM-Prime ($\ell = 2$)	✓	≤30.91	≤27.93	≤74.81	≤55.50	≤63.21	≤**33.32**	≤25.44
MDLM-Prime ($\ell = 3$)		≤34.55	≤31.00	≤55.46	≤41.15	≤62.51	≤171.74	≤29.44
MDLM-Prime ($\ell = 3$)	✓	≤27.75	≤27.42	≤60.13	≤42.69	≤62.58	≤41.14	≤24.71
MDLM-Prime ($\ell = 4$)		≤25.79	≤**23.52**	≤**49.90**	≤37.70	≤59.65	≤210.96	≤**23.19**
MDLM-Prime ($\ell = 4$)	✓	≤24.44	≤23.86	≤53.98	≤38.02	≤59.44	≤48.64	≤25.83
MDLM-Prime ($\ell = 6$)		≤25.89	≤25.86	≤52.42	≤**37.65**	≤56.68	≤497.39	≤24.99
MDLM-Prime ($\ell = 6$)	✓	≤25.80	≤26.87	≤62.62	≤45.36	≤60.16	≤59.09	≤25.19
MDLM-Prime ($\ell = 8$)		≤32.20	≤26.01	≤53.27	≤38.05	≤65.63	≤218.99	≤28.63
MDLM-Prime ($\ell = 8$)	✓	≤**25.23**	≤25.77	≤53.77	≤38.00	≤64.89	≤54.50	≤25.79

Training and Implementation Details. Following prior works [7, 9, 10, 12, 13], our model architecture is a diffusion transformer (DiT) [26] with rotary positional embeddings [27]. The model consists of 12 DiT blocks with a hidden dimension of 768 and 12 attention heads, consistent with the configuration used in [9]. A dropout rate of 0.1 is applied throughout training. The scheduling function is defined as $\alpha_t = 1 - t$. The network is optimized using the AdamW optimizer [63] with $\beta_1 = 0.9$, $\beta_2 = 0.999$, and a linear learning rate warm-up from 0 to 3×10^{-4}. The model is trained with a batch size of 128 for 1 million steps, corresponding to a total of 262 billion tokens [9]. To facilitate efficient training, the model is first trained without the carry-over parameterization for 900K iterations, and then enables this mechanism in the final 100K iterations (see Section A.5.2). The training is performed on eight NVIDIA L40 GPUs with 48 GB memory.

A.5 Supplementary Experiments

In this section, we present additional experimental results. Section A.5.1 provides ablation studies on the embedding layer designs and the carry-over parameterization method. Section A.5.2 reports the time cost per training iteration under different values of ℓ. Section A.5.3 presents qualitative results for both the text and image generation tasks. Finally, Section A.5.4 provides visualization of the sampling process of MDM-Prime.

A.5.1 Ablation Study

Carry-over Parameterization. In this subsection, we evaluate the effectiveness of the carry-over parameterization. Table A2 compares the performance of MDLM-Prime trained with and without carry-over on the OWT dataset. It is observed that incorporating the carry-over parameterization consistently leads to slightly improved performance across different values of ℓ. In contrast, when the model is evaluated using unseen data from zeroshot datasts, the difference becomes significant. A comparison is offered in Table A1. This parameterization method consistently offers performance gains on LAMBADA. In addition, MDLM-Prime without the carry-over parameterization has noticeably inferior performance (i.e., difference > 30) on specific text domains such as PubMed. We hypothesize that the carry-over parameterization, by explicitly removing predictions of $\boldsymbol{y}_0$ inconsistent with $\boldsymbol{y}_t$, effectively reduces the uncertainty in the model's outputs and therefore enhances generalization to unseen domains.

Table A2: Perplexity (PPL) evaluation on OWT. The symbol ↓ represents that lower values correspond to better performance.

	PPL (↓)	
Carry-over		✓
$\ell = 2$	≤18.04	≤17.90
$\ell = 3$	≤16.43	≤16.36
$\ell = 4$	≤15.67	≤15.62
$\ell = 6$	≤**15.43**	≤**15.36**
$\ell = 8$	≤15.48	≤15.45

Embedding Layer Designs. In this subsection, we compare the performance of MDM-Prime when trained using two embedding-merging strategies: the concatenation-based design (denoted as (o2)) and the cross-attention-based design (denoted as (o3)), both of which are detailed in Appendix A.3. We first evaluate these approaches on the image generation task using the CIFAR-10 dataset with

Table A3: The average runtime calculated based on ten training iterations. The results are reported in seconds per iteration. The 95% confidence intervals of the results are around 0.05.

		Runtime (sec./iter.)	
Carry-over			✓
$\ell = 1$,	$b = 50,257$	1.06 e-1	1.14 e-1
$\ell = 2$,	$b = 225$	1.06 e-1	1.29 e-1
$\ell = 3$,	$b = 37$	1.11 e-1	1.34 e-1
$\ell = 4$,	$b = 15$	1.10 e-1	1.43 e-1
$\ell = 6$,	$b = 7$	1.13 e-1	1.75 e-1
$\ell = 8$,	$b = 4$	1.07 e-1	2.12 e-1

Table A4: FID scores evaluated under different NFE on the CIFAR-10 and ImageNet-32 benchmarks. Lower values correspond to better performance. MDM-Prime is trained with $\ell = 2$.

CIFAR-10			
NFE	128	256	512
MDM	7.55	5.00	4.66
MDM-Prime	**5.22**	**3.73**	**3.26**
ImageNet-32			
NFE	128	256	512
MDM	9.55	8.24	8.12
MDM-Prime	**7.85**	**7.61**	**7.31**

Table A5: Ablation results on the validation perplexities of MDLM-Prime ($\ell = 6$). The model is trained and evaluated with or without the carry-over parameterization. Lower perplexity indicates better performance.

Carry-over		Dataset							
Train	Evaluation	OWT	LAMBADA	WikiText	PTB	LM1B	AG News	PubMed	ArXiv
		≤15.43	≤25.89	≤25.86	≤52.42	≤37.65	≤56.68	≤497.39	≤24.99
	✓	≤15.41	≤**24.36**	≤**24.25**	≤**49.77**	≤**37.35**	≤**56.48**	≤63.58	≤**24.81**
✓	✓	≤**15.36**	≤25.80	≤26.87	≤62.62	≤45.36	≤60.16	≤**59.09**	≤25.19

$\ell = 2$. The concatenation-based design (o2) achieves a superior FID score of 3.26, compared to the cross-attention-based design (o3), which yields an FID score of 3.98. On the other hand, for text generation, using the cross-attention-based design leads to a noticeable degradation in performance, with perplexity increasing from 18.04 to 57.37 for $\ell = 2$ on the OWT dataset. These results indicate that the concatenation-based design offers better overall performance across both modalities.

A.5.2 Runtime Evaluation

Table A3 presents a comparison of the time required to optimize the model parameterized with and without the carry-over condition (i.e., Eq. (6)) under different choices of ℓ. We observe that larger values of ℓ incur higher computational costs for the setup with carry-over condition. The increase in runtime arises from the filter lookup table calculation required by the carry-over parameterization (see Section A.2.3). Due to the 48 GB memory limitation of our available GPUs, parallelized querying of the filter lookup table (with a batch size of 128) often results in out-of-memory errors. To mitigate this issue, we implement the lookup operations sequentially in our text experiments, which introduces additional computational overhead as ℓ increases. To maintain a runtime comparable to the baseline, we first train the model without the carry-over parameterization for 900K iterations, and then enable this mechanism during the final 100K iterations. Alternatively, the carry-over parameterization can be applied only at inference time. As shown in Table A5, this setup—training without the carry-over parameterization but evaluating with it—achieves performance comparable to, or slightly better than, that of the model trained and evaluated with the carry-over parameterization, while avoiding the higher training cost. Nonetheless, to preserve the mathematical formality of our proposed framework, we adopt the carry-over parameterization during both training and evaluation.

A.5.3 Sample Quality Evaluation

Image Generation. In this subsection, we provide additional quantitative and qualitative results to assess sample quality in image generation tasks. Table A4 presents a comparison of FID scores between MDM-Prime and its baseline (i.e., a standard MDM). The results show that MDM-Prime outperforms MDM across different number of function evaluations (NFE). In addition, Fig. A17 provides a number of imputation examples, while Figs. A18 and A19 present qualitative results showcasing multiple uncurated samples generated by MDM-Prime.

Text Generation. In this subsection, we present additional sample quality evaluations for text generation tasks. Fig. A6 presents the generative perplexity (Gen PPL) results for ARM, MDLM, and MDLM-Prime. Gen PPL is calculated by evaluating the perplexity of generated samples using a pretrained large language model. To ensure a comprehensive assessment, we report Gen PPL obtained from both order-agnostic models (e.g., LLaDA-8B [8]) and order-specific models (e.g., GPT-2 Large [5]). Following [24], we adopt 64-bit floating-point precision to enhance sampling accuracy. We observe opposite trends depending on the pretrained model: MDLM and MDLM-Prime outperform ARM under LLaDA-based Gen PPL, whereas ARM achieves lower perplexity than MDLM and MDLM-Prime when evaluated using GPT-2. In addition, we observe that MDLM-Prime performs better than MDLM when the number of sampling steps ranges from 512 to 2048.

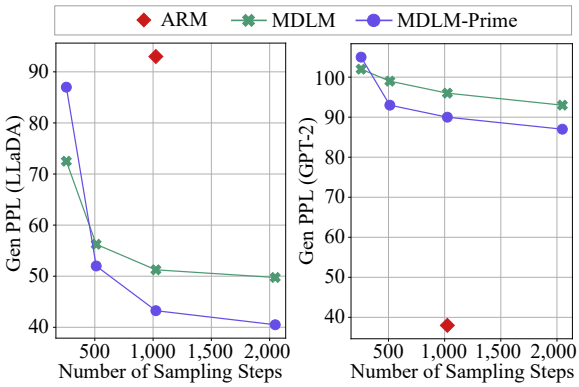


Figure A6: Gen PPL evaluated using LLaDA-8B (left) and GPT-2 Large (right). Lower values correspond to better performance. MDLM-Prime adopts $\ell = 6$.

In addition, to qualitatively evaluate MDLM-Prime, we present both unconditional and conditional generation samples. Unconditional generation results are shown in Fig. A16. For conditional generation, we provide the model with prefix (i.e., Fig. A11) and suffix (i.e., Fig. A12) texts sourced from an online article describing Rabindranath Tagore's poems (link), and assign the model to generate the middle content. Figs. A13-A15 present some samples generated using MDLM-Prime.

A.5.4 Visualization of Sampling Processes

Unlike continuous diffusion models (e.g., [19, 37]), visualizing the sampling processes of MDM and MDM-Prime is nontrivial due to the presence of masked tokens and sub-tokens. To qualitatively analyze the generation behaviors of MDM and MDM-Prime, we develop a visualization technique that illustrates the evolution of samples throughout the diffusion process. Instead of directly visualizing the latent variables (i.e., $\boldsymbol{x}_t$ for MDM or $\boldsymbol{y}_t$ for MDM-Prime), we visualize the model's predictions of the final unmasked sample. Specifically, we show $\boldsymbol{x}_0 \sim p_\theta(\cdot\,|\boldsymbol{x}_t)$ for MDM, and $\boldsymbol{x}_0 = f^{-1}(\boldsymbol{y}_0)$ where $\boldsymbol{y}_0 \sim p_\theta(\cdot\,|\boldsymbol{y}_t)$ for MDM-Prime.

Fig. A7 shows the evolution of samples $\boldsymbol{x}_0$ generated by MDM-Prime trained with different values of ℓ on CIFAR-10. For each setup, we capture a snapshot every 25 timesteps, with the total number of function evaluations (NFE) fixed at 500. The snapshots are displayed from left to right, depicting the progressive refinement from noisy initialization to the final output.

The text generation processes are presented in Figs. A8-A10. Since the coarse-to-fine transitions of $\boldsymbol{x}_0$ are less visually discernible in text generation, we additionally illustrate the denoising progression using token-wise masked ratios. In the figures, darker shades of blue indicate a higher degree of masking, while lighter colors suggest that a token is closer to its final predicted state. The results demonstrate that MDLM-Prime with larger values of ℓ enables a more fine-grained denoising process. In contrast, the original MDLM employs a binary masking scheme, where each token is either masked or unmasked. The results thus highlight the property of the proposed Prime method.

A.6 Limitations

A core assumption in MDM is that tokens (i.e., $x_0^1, \cdots, x_0^L$) are conditionally independent given the latent representation $\boldsymbol{x}_t$, i.e., $x_0^i \perp\!\!\!\perp x_0^j \mid \boldsymbol{x}_t$ for $i \neq j$. Under this assumption, many recent works [8–10, 20, 22–24] factorize the conditional distribution as $p_\theta(\boldsymbol{x}_0|\boldsymbol{x}_t) = \prod_{i=1}^{L} p_\theta(x_0^i|\boldsymbol{x}_t)$, which leads to the sum of log-probability terms in the evidence upper bound (i.e., Eq. (4)). While this factorized parameterization offers significant advantages in sampling and training efficiency, some recent studies [12, 64] have shown that it may degrade performance due to its inability to model inter-token dependencies.

($\ell = 2$)

($\ell = 4$)

($\ell = 8$)

Figure A7: Visualization of the sampling processes of MDM-Prime with $\ell = 2$ (top), $\ell = 4$ (middle), and $\ell = 8$ (bottom). The models are trained on CIFAR-10.

In this work, we adopt the same conditional independence assumption as prior studies [8–10, 20, 22–24], and extend it to our sub-token formulation. Specifically, we assume $\boldsymbol{y}_0^i \perp\!\!\!\perp \boldsymbol{y}_0^j \mid \boldsymbol{y}_t$ for $i \neq j$, and accordingly factorize the conditional distribution as $p_\theta(\boldsymbol{y}_0|\boldsymbol{y}_t) = \prod_{i=1}^{L} p_\theta(\boldsymbol{y}_0^i|\boldsymbol{y}_t)$. As discussed in Section 3, further factorizing each sub-token sequence prediction as $p_\theta(\boldsymbol{y}_0^i|\boldsymbol{y}_t) = \prod_{j=1}^{\ell} p_\theta(y_0^{i,j}|\boldsymbol{y}_t)$ leads to a clear drop in performance. Therefore, we retain the inter-token factorization (i.e., $p_\theta(\boldsymbol{y}_0|\boldsymbol{y}_t) = \prod_{i=1}^{L} p_\theta(\boldsymbol{y}_0^i|\boldsymbol{y}_t)$) to strike a balance between accuracy and computational efficiency.

A.7 Broader Impacts and Future Works

This paper investigated whether discrete data can be effectively represented and reconstructed using sub-token representations. We proposed MDM-Prime as a simple yet effective instantiation of this idea. Given its superior performance across both image and text generation tasks, MDM-Prime holds potential for positive societal impact. To support broader scientific contributions, we outline two potential directions for extending the proposed framework and guiding future research in this area:

Learnable Transformations for Discrete Data. MDM-Prime adopts base-b encoding as an invertible mapping to extend discrete data into longer sequences of sub-tokens. Analogous to normalizing flows in the continuous domain (e.g., [65, 66]), which parameterize invertible transformations using carefully designed model architectures, we anticipate that this discrete transformation can likewise be parameterized and optimized during training. As invertible modeling in the discrete domain remains underexplored, it presents opportunities to advance discrete generative modeling.

Capturing Inter-token Dependencies. Our current decoder implementation follows prior works [8–10, 20, 22–24], which assume conditional independence between tokens (see Section A.6). While recent approaches [12, 64] have proposed methods to relax this assumption, they require training an additional autoregressive model to guide the sampling process of MDM, resulting in substantial computational overhead. Hence, developing a more efficient method for MDM to model inter-token joint distributions represents a promising direction for future work.

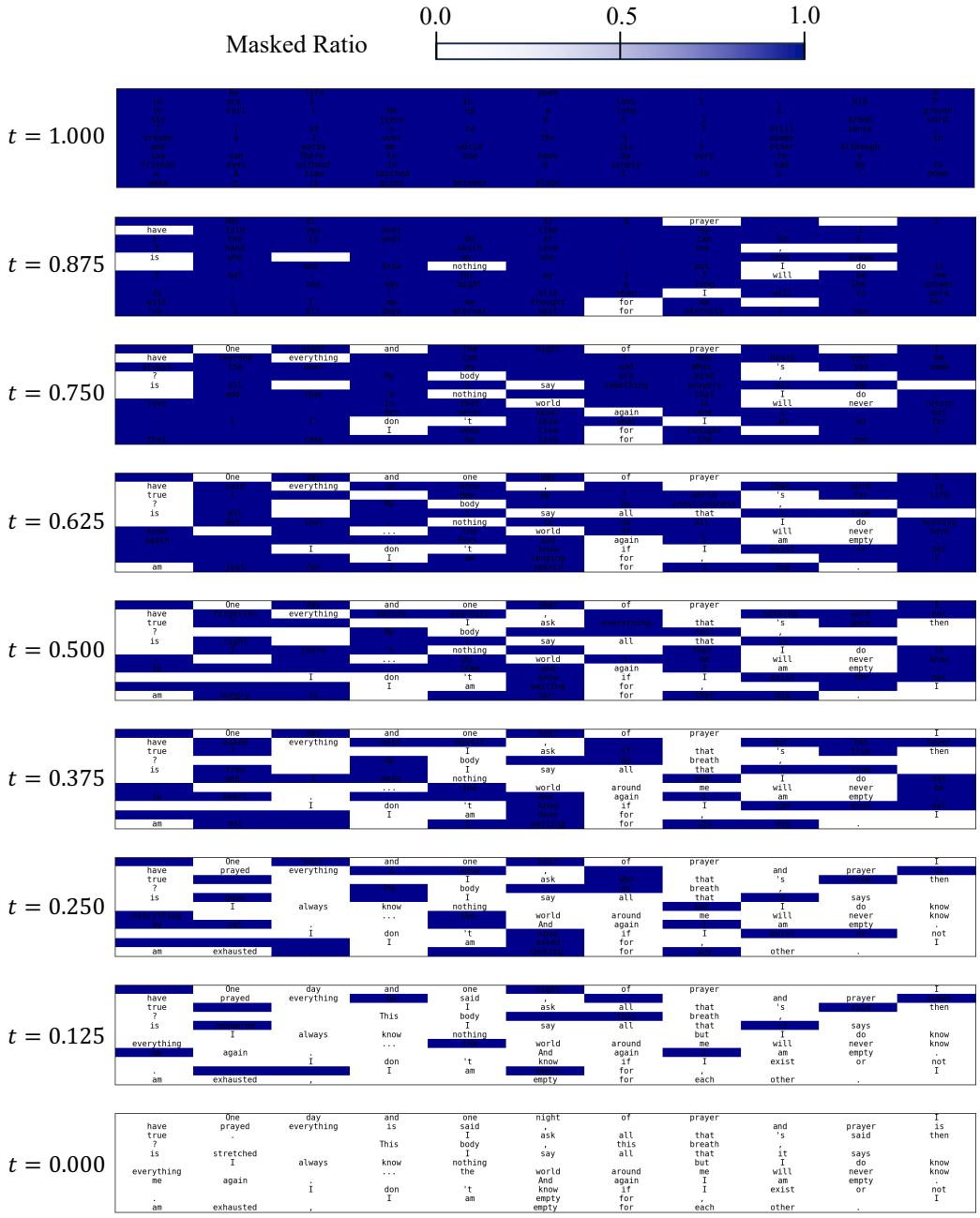


Figure A8: Visualization of the sampling processes of MDLM. The masked ratio is measured on a per-token basis, with higher values indicated by darker shades of blue. The samples are generated with prefix and suffix presented in Figs. A11 and A12, respectively. Further experimental details are shown in Section A.5.3.

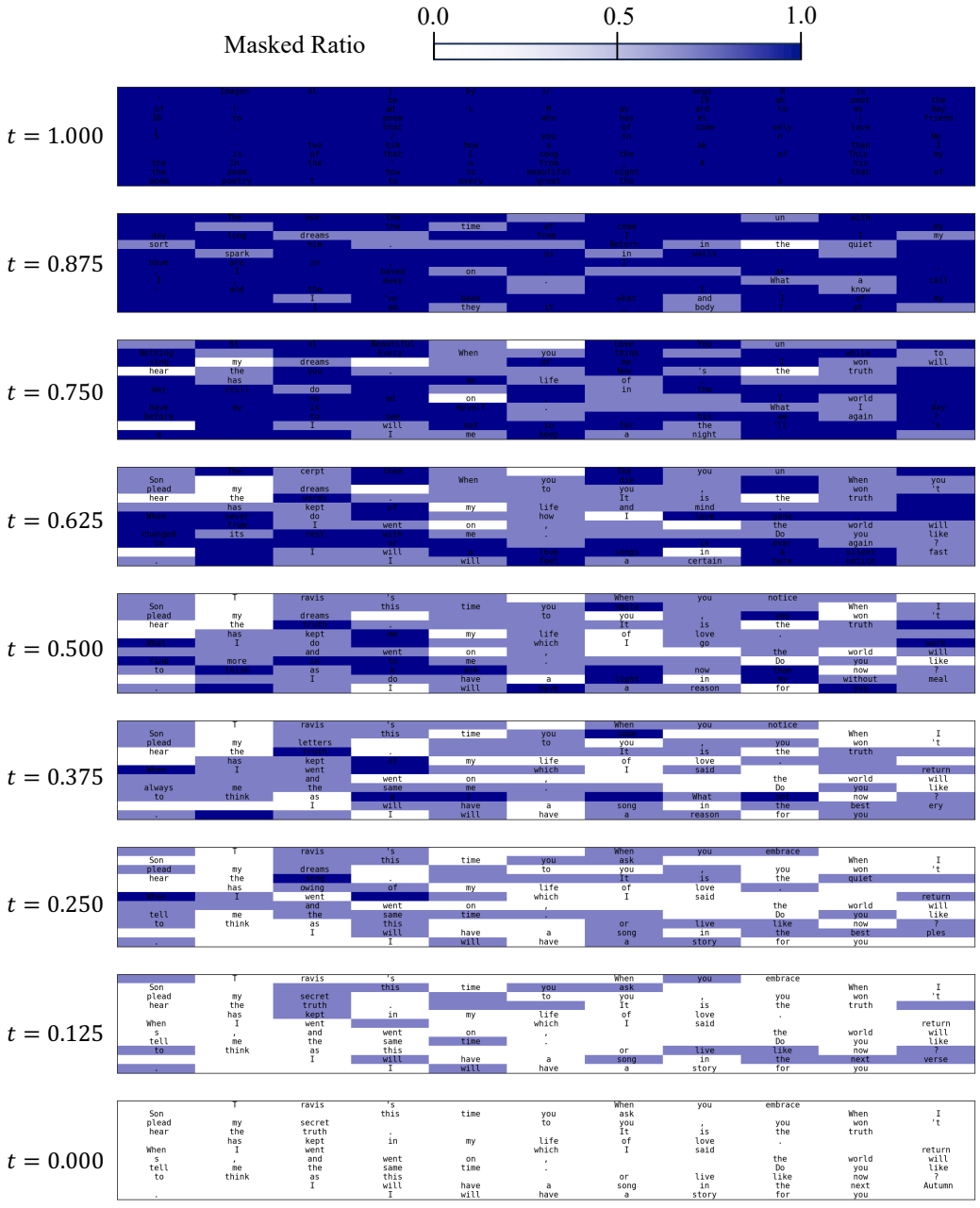


Figure A9: Visualization of the sampling processes of MDLM-Prime ($\ell = 2$). The masked ratio is measured on a per-token basis, with higher values indicated by darker shades of blue. The samples are generated with prefix and suffix presented in Figs. A11 and A12, respectively. Further experimental details are shown in Section A.5.3.

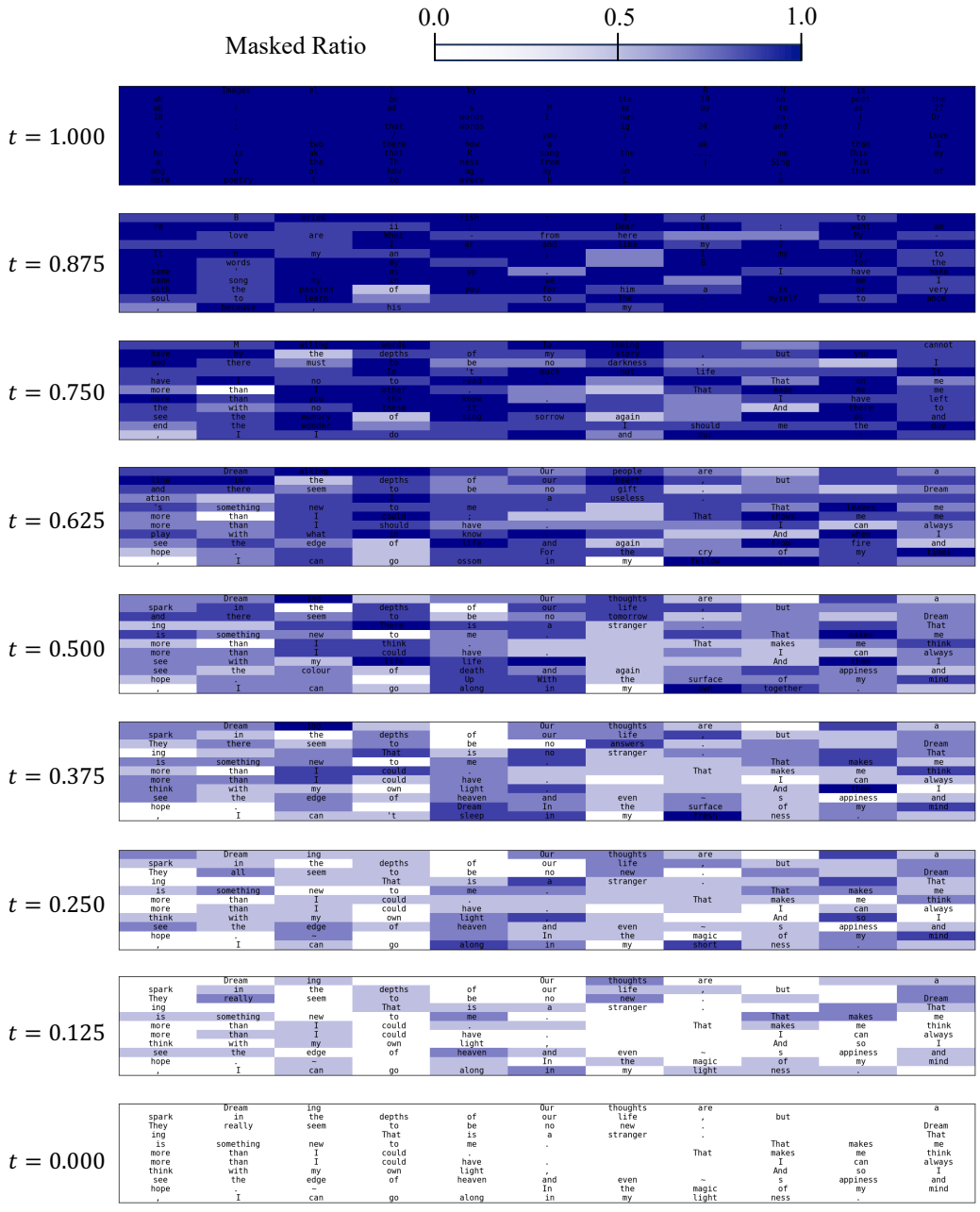


Figure A10: Visualization of the sampling processes of MDLM-Prime ($\ell = 4$). The masked ratio is measured on a per-token basis, with higher values indicated by darker shades of blue. The samples are generated with prefix and suffix presented in Figs. A11 and A12, respectively. Further experimental details are shown in Section A.5.3.

10 timeless poems by Rabindranath Tagore

Born on May 7, 1861 the Bard of Bengal, Rabindranath Tagore has inspired generations of people through his writings, poetry and thoughts. Tagore was much ahead of his time and his works were loved not only in India but across the world. His much-acclaimed work 'Gitanjali', which was first published in 1910 and later translated and published into English in 1912, won him the prestigious Nobel Prize in Literature in 1913 for "his profoundly sensitive, fresh and beautiful verse, by which, with consummate skill, he has made his poetic thought, expressed in his own English words, a part of the literature of the West." Infact, Rabondranath Tagore was the first non-European to ever win a Nobel Prize!

Remembering Tagore on his 160th birth anniversary today, here we list down some of his timeless poems that continue to resonate his creative charm and are still as relevant. These poems reflect upon different moods and are a must read for all. Read on!

Figure A11: Prefix text used for conditional sample generation.

Waiting
The song I came to sing
remains unsung to this day.
I have spent my days in stringing
and in unstringing my instrument.
The time has not come true,
the words have not been rightly set;
only there is the agony
of wishing in my heart….

I have not seen his face,
nor have I listened to his voice;
only I have heard his gentle footsteps
from the road before my house…..

But the lamp has not been lit
and I cannot ask him into my house;
I live in the hope of meeting with him;
but this meeting is not yet

Friend
Art thou abroad on this stormy night
on thy journey of love, my friend?
The sky groans like one in despair.

I have no sleep tonight.
Ever and again I open my door and look out on
the darkness, my friend!

I can see nothing before me.
I wonder where lies thy path!

By what dim shore of the ink-black river,
by what far edge of the frowning forest,
through what mazy depth of gloom art thou threading
thy course to come to me, my friend?

Figure A12: Suffix text used for conditional sample generation.

```
Travis's
When you embrace
Son
this time you ask

When I plead my secret
to you, you won't hear the light.
It is the quiet decker of my life of love.

When I went
which I said
returns, and went on,
the world will tell me the same time.

Do you like to think as this
or live like now?

I will have a song in the Adverse.
I will have a story for you
```

Figure A13: Conditional sample generated using MDLM-Prime ($\ell = 2$).

```
Dreaming
Our thoughts are
a spark in the depths of our life, but
They really seem to be no new.

Dreaming
That is a stranger.
That is something new to me.
That makes me more than I could.
That makes me think more than I could have.

I can always think with my own light,
And so I see the edge of heaven and even~sappiness and hope.~
In the magic of my mind, I can go along in my lightness.
```

Figure A14: Conditional sample generated using MDlM-Prime ($\ell = 4$).

```
The Promise
I think about my life when I make me the vessel of life.

Garnings, deeply
At the hinge of my life,
it’s the most tranquil, exposed world.
....

It is the very moment of such a thing
I walk up to my house every time night or fall.
Leave me.

I am a lifetime,
I come.
I begin to see my mind and into my mind.
The passion and death of the night do not think of the other away.
```

Figure A15: Conditional sample generated using MDLM-Prime ($\ell = 6$).

<|endoftext|> argument that such modeling is one of the world's thinkers for climate change.

"People can read the website. We are going to examine them," she said. "Each day, there are companies that only 20 -- 40 people approve of fossil fuel emissions. The huge zonal wave that ice is releasing into the atmosphere. Everything here is…having is back and it is coming."

For the political pro-energy group, SoundEarth and the Climate Collective think that it will lead to a stronger understanding of pollution, which could help make the greenhouse issue a premier educational problem. The studies are expected to increase this further: "For example, plants have the potential to respond to climate change. This is why we as a world need to encourage the kinds of applications that can damage infrastructure, up in trees, up in trenches, up in towers."

"We are most aggressive skeptics," Bec Lindy said. They claim that the pollution is used from storage in the air to take out high-cost water straight from fossil fuels. Carbon dioxide emissions, at times, swamp energy use from companies. This begins with gravity, and through large amounts of rain in mountains, caused storms. The trees also host a public debate over whether it's used when placing organic synchias and plants to actually remove emissions from the minerals of which it is created, as well as changing water to be natural.

While many of those skeptics are even aware that such pollution is from a scientific study, the Baker Effect, they may be receiving a new level of threat from the state.

Beegy said this is something she's hopeful about and suspect that it will continue to grow, especially after solar wind projects shrink the risk of air pollution. The solar Effect has its course changed.

"We see a public concern as knowing how we can affect the climate," Lindy said. "We have a set of models called climate feedback that we are able to predate the climate on. So if we need to, it forces us to actively reduce output to actually increase carbon pollution. They use some of the efficiency we already study as sources of government resources.

Conners is working on a supplementary Energy Initiative Decamp to study new complete wind refrigerators at the University of California that will be charged and responsive. House estimates on their new solar panels will likely serve over 33,000 panels, consisting of residential, "basement cost" scenarios that could be significantly set up if rug leaks in the open-up.

"The University is moving forward with the capital energy," Lindy said.

"For 2 years, we haven't worked in the BLy's first experiment," agrees Professor Mark Clerock, director of the National Institute of Reliousness and Turiative Research Unit at Hampshire University. "So whether it takes the 9 hours per second test is six-figure! Even 2 miles wouldn't have been predicted."

Professor Jeanne Morul Institute of California at California State adds, and stops saying that this will only accelerate anyway. Any skeptics say the BLy's changes sheet this might give even more freedom to Colorado.

In approving the research, the Democle Climate Council published a statement that the additional study "taught mixing severe weather events. In forests, burn the volume. Therefore the magnitude to severe consequences of weather could weaken by comparison." Professor Thomas Muirema, Professor A at West Texas's Gulf Observatory, is aiming to do the research this year. When the gets through extreme weather, he attributes one of the biggest implications of his discoveries. "It is the poisoning of our income – I don't see them taking any action and still don't have the opportunity to see pollution. I don't think their companies have anything to start doing," he said. "We'll have to deal with it again on whether its power … but we can see what consequences for them before and whether that is thought locally entirely on what comes with power would be an investigation."

We are thinking NOE to the chemicals, methane that comes from traditional coal emitting plants, the standard statement is funny.

If someone wants to try to do all the environmental damage with it, it can't afford the Department of Energy. But, government of the coalition has tried to extend those limits to 95 percent. It shows credit to Bob Marshall measures used by his governments decades ago. It raises some questions they cannot use plants made from traditional coal but can do get a product at a high cost. Should they get more money from the subsidies or others from their locally grown stations or the food grains required to comply? Their limits would be at least the biggest in the world.

The Marshall says that they send farms in rais-<|endoftext|>

Figure A16: Unconditional samples generated by MDLM-Prime with 1,024 sampling steps.

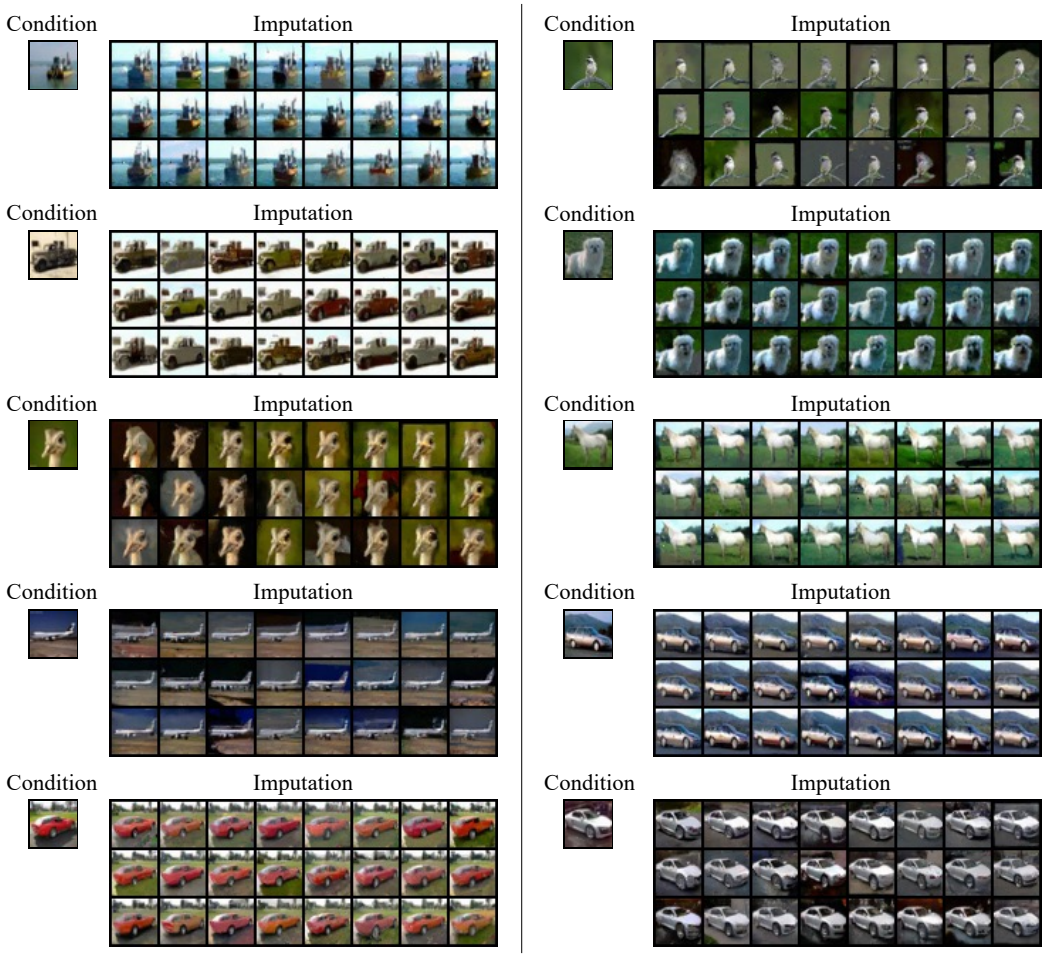


Figure A17: Uncurrated imputation results generated by MDM-Prime with CIFAR-10 images as conditions.

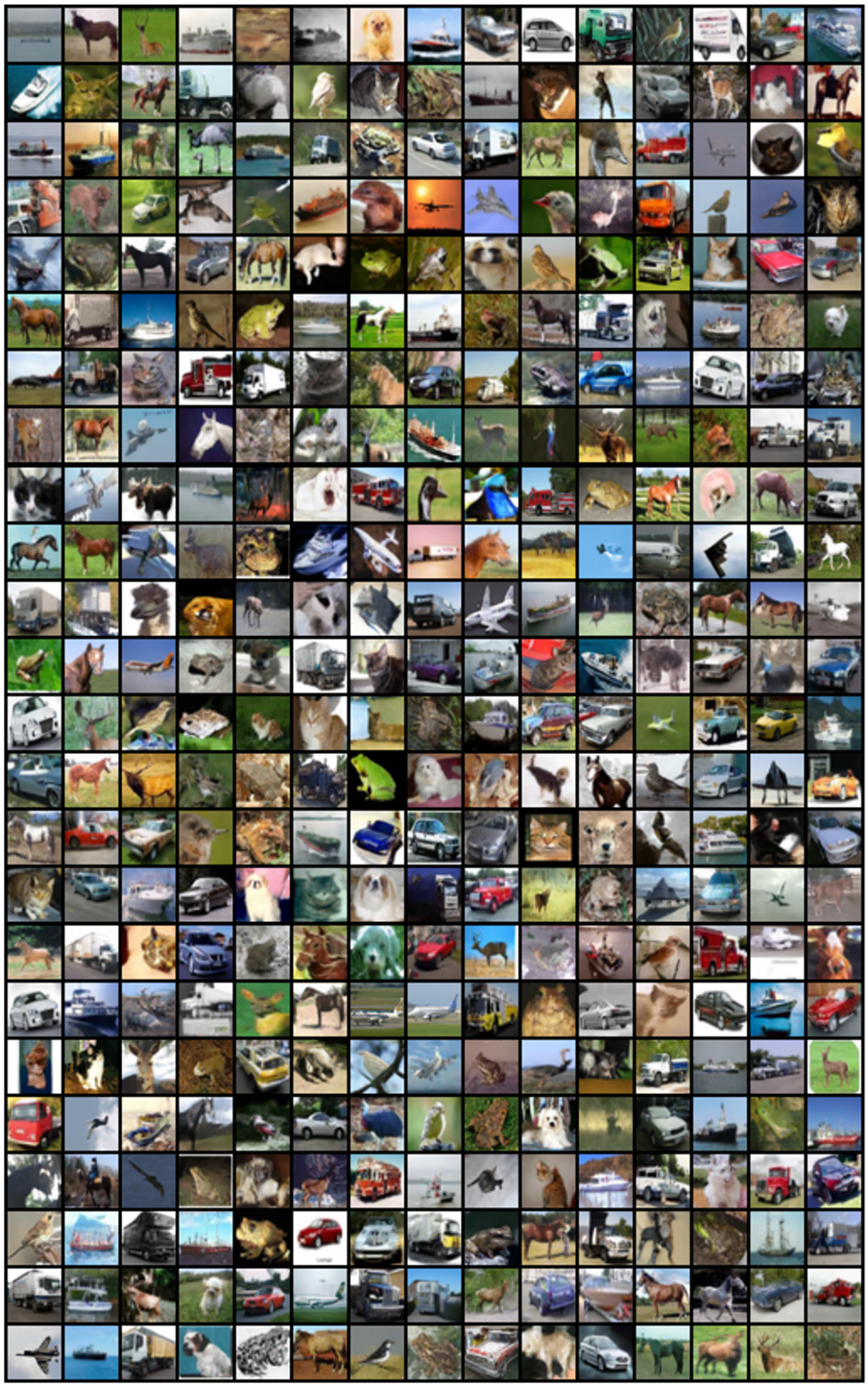


Figure A18: CIFAR-10 samples generated by MDM-Prime with NFE=512.

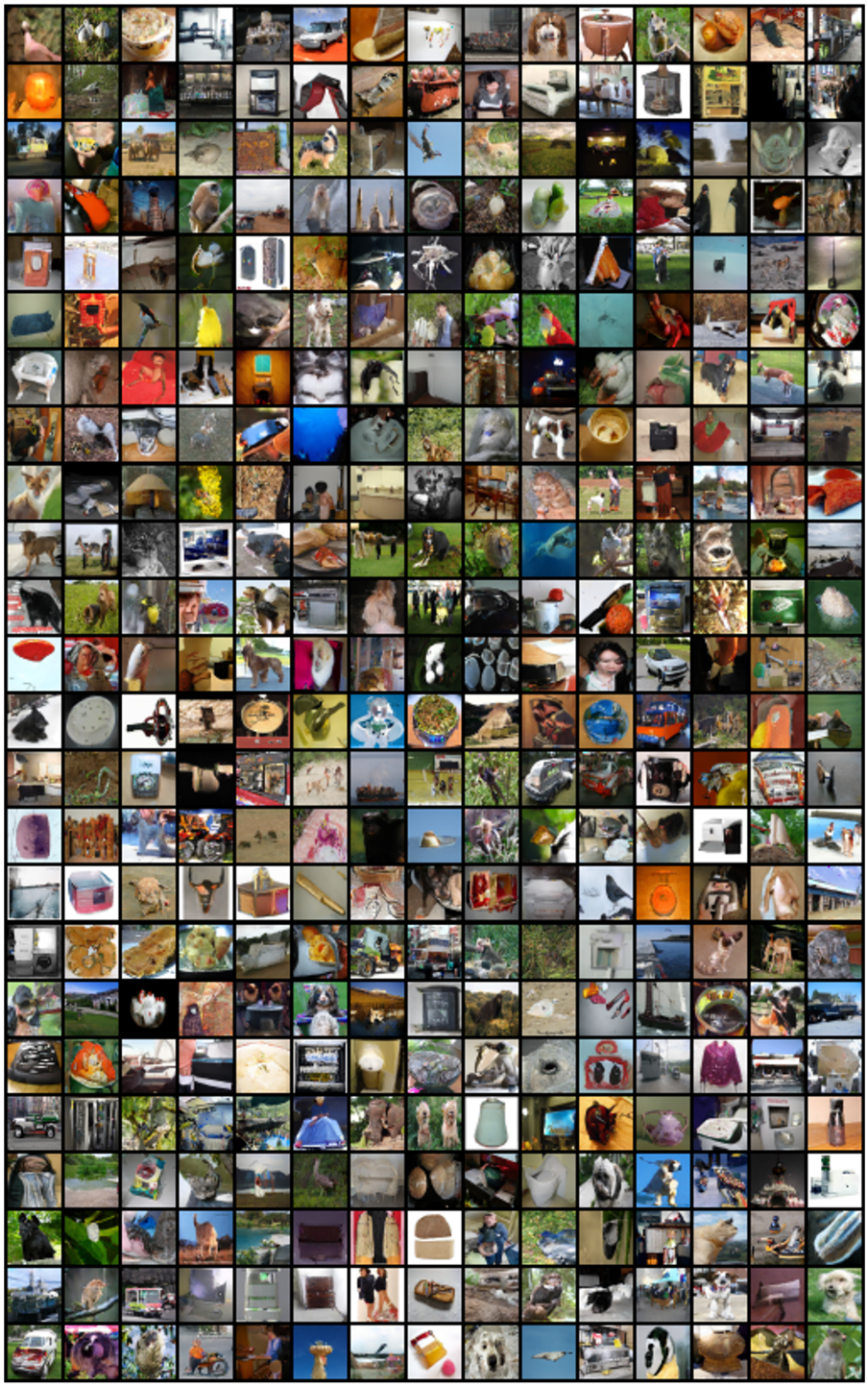


Figure A19: ImageNet-32 samples generated by MDM-Prime with NFE=512.