arXiv:2603.16077v4 [cs.LG] 2 Sep 2026

MDM-Prime-v2: Binary Encoding and Index Shuffling Enable Scaling of Diffusion Language Models

Chen-Hao Chao[1], **Wei-Fang Sun**[2], **Junwei Quan**[1], **Chun-Yi Lee**[3], **Rahul G. Krishnan**[1]
[1]University of Toronto, Vector Institute
[2]NVIDIA AI Technology Center
[3]National Taiwan University
Correspondence: chchao@cs.toronto.edu, rahulgk@cs.toronto.edu
Code: https://github.com/chen-hao-chao/mdm-prime-v2

Abstract

Masked diffusion models (MDM) exhibit superior generalization when learned using a Partial masking scheme (Prime). This approach converts tokens into sub-tokens and models the diffusion process at the sub-token level. We identify two limitations of the MDM-Prime framework. First, we find that the functional form of the subtokenizer significantly increases the cross-entropy loss in the objective when paired with commonly used Byte-Pair-Encoding (BPE) tokenizers. Second, we lack tools to guide the hyperparameter choice of the token granularity in the subtokenizer. To address these limitations, we analyze the optimal design of the subtokenizer that minimizes MDM-Prime training objective and develop MDM-Prime-v2, a masked diffusion language model which incorporates *Binary Encoding* and *Index Shuffling*. Our analysis characterizes how token granularity and sub-token entropy influence the training objective and downstream performance, providing principled criteria for subtokenizer design. When extending the model size to 1.1B parameters, MDM-Prime-v2 demonstrates superior average zero-shot accuracy across eight commonsense reasoning benchmarks, outperforming similar-sized baselines including GPT-Neo, OPT, Pythia, Bloom, SMDM, and TinyLLaMA.

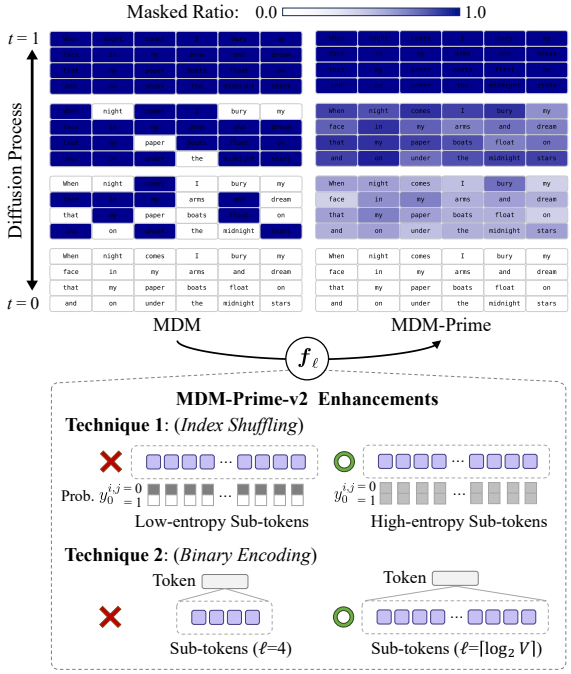


Figure 1: Overview of MDM, MDM-Prime, and the MDM-Prime-v2 enhancements. ℓ is the token granularity in base-b encoding and V is the vocabulary size. $y_0^{i,j}$ is a sub-token with positional indices i and j. MDM-Prime-v2 adopts two techniques: Technique 1 employs an index shuffling operation to encode high-entropy sub-tokens. Technique 2 sets $\ell = \lceil \log_2 V \rceil$ for $\boldsymbol{f}_\ell$, which performs binary encoding. The gray cell intensity in the lower section represents sub-token probability.

1 Introduction

Generative pretraining serves as the cornerstone of scalable language modeling (Kaplan et al., 2020). Autoregressive models (ARM) (e.g., Radford et al. (2019)) and masked diffusion models (MDM) (e.g., Sahoo et al. (2024); Nie et al. (2025b)) represent two paradigms: the former utilizes the chain rule to decompose the joint likelihood of the data, while the latter employs stochastic unmasking to reconstruct data from masked sequences.

Chao et al. (2025) demonstrated that subtokenization improves the expressivity of MDM latent representations, enabling partial word editing during diffusion steps and substantially improving sampling behavior. This framework was formalized as MDM-Prime, a generalized class of MDMs. In MDMs, each token takes one of two states, *masked* or *unmasked*. MDM-Prime introduces a richer set of intermediate latent representations via partially masked tokens. MDM-Prime uses a subtokenizer $\boldsymbol{f}_\ell$ with adjustable token granularity ℓ to encode each token into a sequence of ℓ sub-tokens. By modeling the diffusion process at the sub-token level, partially masked tokens can be produced naturally as intermediate transition states, leading to a fine-grained denoising process,

as illustrated in the upper section of Fig. 1.

We investigate two fundamental questions regarding $\boldsymbol{f}_\ell$. First, we identify a link between the subtokenizer, which determines how the model perceives inputs, and the tokenizer, which defines the data distribution. Our analysis of $\boldsymbol{f}_\ell$ reveals that high-entropy sub-tokens decrease the conditional (predictive) entropy, effectively minimizing the objective. Second, instead of treating ℓ as an empirically tuned hyperparameter, we characterize the relationship between ℓ and downstream performance, and establish a principled criterion for its selection. Based on this insight, we introduce a technique called index shuffling to permute token indices. We integrate these improvements into a unified framework named MDM-Prime-v2.

We perform a scaling analysis of MDM-Prime-v2. By varying compute budgets from 3×10^{18} to 3×10^{20} FLOPs, we determine compute-optimal configurations and derive its scaling behavior via Chinchilla analysis (Hoffmann et al., 2022). We scale MDM-Prime-v2 to 1.1B parameters and our model demonstrates superior performance over similar-sized baselines including GPT-Neo (Black et al., 2021), OPT (Zhang et al., 2022), Pythia (Biderman et al., 2023), Bloom (BigScience, 2023), SMDM (Nie et al., 2025a) and TinyLLaMA (Zhang et al., 2024) on zero-shot commonsense reasoning benchmarks. The contributions of this work are summarized as follows:

- We analyze the MDM-Prime objective and characterize how sub-token entropy and token granularity ℓ influence its optimal value, providing a theoretical basis for subtokenizer design.
- Based on the aforementioned analysis, we establish two practical techniques: a technique to enhance sub-token entropy, called index shuffling, and a selection criterion for ℓ under which $\boldsymbol{f}_\ell$ performs binary encoding.
- We demonstrate that MDM-Prime-v2 outperforms MDMs and ARMs at the 1.1B parameter scale, achieving the highest average zero-shot accuracy across eight benchmarks.

2 Background

2.1 Masked Diffusion Models

Let $\boldsymbol{x} = [x^1, \cdots, x^L] \in \mathcal{X}^L$ be a sequence of L tokens , where x^i denotes an element of $\boldsymbol{x}$ and $\mathcal{X} \triangleq \{0, \cdots, V-1\}$ represents a set of V token indices. Given a continuous time variable $t \in [0,1]$, $\boldsymbol{x}_0 \in \mathcal{X}^L$ denote the sample drawn from the data distribution q_{data}, and $\boldsymbol{x}_t \in (\mathcal{X} \cup \{\mathtt{m}\})^L \triangleq \tilde{\mathcal{X}}^L$ denotes the latent variable introduced by the forward diffusion process, where $\mathtt{m}$ represents the masked token. 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 a kernel $q_\alpha(\boldsymbol{x}_t|\boldsymbol{x}_0) = \prod_{i=1}^L q_\alpha(x_t^i|x_0^i)$ following a strictly decreasing time-dependent scheduling function $\alpha_t : [0,1] \to [0,1]$. The element-wise kernel q_α is defined as follows (Sahoo et al., 2024):

$$q_\alpha(x_t^i \mid x_0^i) = (1-\alpha_t)\delta_{\mathtt{m}}(x_t^i) + \alpha_t \delta_{x_0^i}(x_t^i). \quad (1)$$

Let $s, t \in [0,1]$ be time variables such that $s < t$. The reverse diffusion process is performed by iteratively applying $p(\boldsymbol{x}_s|\boldsymbol{x}_t) = \mathbb{E}_{p(\boldsymbol{x}_0|\boldsymbol{x}_t)}[q_\alpha(\boldsymbol{x}_s|\boldsymbol{x}_t, \boldsymbol{x}_0)]$ (Austin et al., 2021), by first drawing $\boldsymbol{x}_0 \sim p(\cdot \mid \boldsymbol{x}_t)$ and then sampling $\boldsymbol{x}_s \sim q_\alpha(\cdot \mid \boldsymbol{x}_t, \boldsymbol{x}_0)$ in each timestep, starting from $t = 1$. The reverse kernel $q_\alpha(\boldsymbol{x}_s|\boldsymbol{x}_t, \boldsymbol{x}_0) = \prod_{i=1}^L q_\alpha(x_s^i|x_t^i, x_0^i)$ is defined as:

$$q_\alpha(x_s^i|x_t^i, x_0^i) = \begin{cases} \delta_{x_t^i}(x_s^i), & \text{if } x_t^i \in \mathcal{X}, \\ \frac{1-\alpha_s}{1-\alpha_t}\delta_{\mathtt{m}}(x_s^i) + \frac{\alpha_s-\alpha_t}{1-\alpha_t}\delta_{x_0^i}(x_s^i), & \text{if } x_t^i = \mathtt{m}. \end{cases} \quad (2)$$

Intuitively, each masked token transitions to its original value x_0^i with probability $\frac{\alpha_s-\alpha_t}{1-\alpha_t}$ and retains the masked value with probability $\frac{1-\alpha_s}{1-\alpha_t}$.

To learn $p(\boldsymbol{x}_0|\boldsymbol{x}_t)$, one common approach is to minimize a weighted cross-entropy loss:

$$\mathcal{L} = \int_0^1 w(t)\mathbb{E}_{q_\alpha(\boldsymbol{x}_0,\boldsymbol{x}_t)}\left[-\log p(\boldsymbol{x}_0|\boldsymbol{x}_t)\right] dt, \quad (3)$$

where $q(\boldsymbol{x}_0, \boldsymbol{x}_t) \triangleq q_{\text{data}}(\boldsymbol{x}_0)q_\alpha(\boldsymbol{x}_t|\boldsymbol{x}_0)$. When $w(t) = 1$, Eq. (3) represents an integral of conditional cross-entropy loss (Austin et al., 2021; Chang et al., 2022; Gat et al., 2024). Under the choice $w(t) = \frac{\alpha_t'}{\alpha_t - 1}$ (where $\alpha_t' = \frac{d}{dt}\alpha_t$) and the factorized posterior $p(\boldsymbol{x}_0|\boldsymbol{x}_t) = \prod_{i=1}^L p(x_0^i|\boldsymbol{x}_t)$, this objective yields a variational upper bound on the negative log-likelihood of the data distribution (Sahoo et al., 2024; Shi et al., 2024).

2.2 Generalization via Partial Masking

Given the token granularity ℓ, MDM-Prime (Chao et al., 2025) represents each token $x_0^i \in \mathcal{X}$ as a sequence of ℓ sub-tokens $\boldsymbol{y}_0^i = [y_0^{i,1}, \cdots, y_0^{i,\ell}] \in \mathcal{Y}^\ell$ via an invertible function $f_\ell : \mathcal{X} \to \mathcal{Y}^\ell$, known as subtokenizer, where $\mathcal{Y} \triangleq \{0, \cdots, b-1\}$ denotes

a set of sub-token indices with $b = \lceil \sqrt[\ell]{V} \rceil \in \mathbb{N}$. The vectorized function $\boldsymbol{f}_\ell$ applied to the full sequence is defined as: $\boldsymbol{f}_\ell(\boldsymbol{x}_0) \triangleq [f_\ell(x_0^1), \cdots, f_\ell(x_0^L)] = [\boldsymbol{y}_0^1, \cdots, \boldsymbol{y}_0^L] = [(y_0^{1,1}, \cdots, y_0^{1,\ell}), \cdots, (y_0^{L,1}, \cdots, y_0^{L,\ell})] = \boldsymbol{y}_0$. The maximum ℓ for $\boldsymbol{f}_\ell$ is $\lceil \log_2 V \rceil$, where the function encodes tokens into binary sub-tokens. This operation is composable: for granularities ℓ_1, ℓ_2 such that $\frac{\ell_2}{\ell_1} \in \mathbb{N}$, the function can be decomposed as $\boldsymbol{f}_{\ell_2} = \boldsymbol{f}_{\ell_2/\ell_1} \circ \boldsymbol{f}_{\ell_1}$. MDM-Prime uses base-$b$ encoding for $\boldsymbol{f}_\ell$, denoted as $\boldsymbol{f}_\ell \triangleq \boldsymbol{f}_{\text{base-}b}$, and implements it using a lookup table.

The latent variable $\boldsymbol{y}_t \in (\mathcal{Y} \cup \{\mathtt{m}\})^{L\times\ell} \triangleq \tilde{\mathcal{Y}}^{L\times\ell}$ is sampled through independent masking, analogous to Eq. (1). The forward kernel $q_\alpha(\boldsymbol{y}_t|\boldsymbol{y}_0) = \prod_{i=1}^{L}\prod_{j=1}^{\ell} q_\alpha(y_t^{i,j}|y_0^{i,j})$ is defined as follows:

$$q_\alpha(y_t^{i,j} \mid y_0^{i,j}) = (1-\alpha_t)\delta_{\mathtt{m}}(y_t^{i,j}) + \alpha_t \delta_{y_0^{i,j}}(y_t^{i,j}). \tag{4}$$

Since $\boldsymbol{f}_\ell$ is invertible and both $\boldsymbol{x}_0$ and $\boldsymbol{y}_0$ are discrete, the change-of-variable principle indicates that the target distribution is invariant: $q_{\text{data}}(\boldsymbol{x}_0) = q_{\text{data}} \circ \boldsymbol{f}_\ell^{-1}(\boldsymbol{y}_0) \triangleq q_{\text{data_y}}(\boldsymbol{y}_0)$. Therefore, MDM-Prime approximates the same objective as MDM by substituting $\boldsymbol{x}_0$ as $\boldsymbol{y}_0$. The objective can be expressed in a similar way as Eq. (3):

$$\mathcal{L}^{(\ell)} = \int_0^1 w(t)\mathbb{E}_{q_\alpha(\boldsymbol{y}_0,\boldsymbol{y}_t)}\left[-\log p_\ell(\boldsymbol{y}_0|\boldsymbol{y}_t)\right] dt, \tag{5}$$

where $q_\alpha(\boldsymbol{y}_0, \boldsymbol{y}_t) = q_{\text{data_y}}(\boldsymbol{y}_0)q_\alpha(\boldsymbol{y}_t|\boldsymbol{y}_0)$, and $p_\ell(\boldsymbol{y}_0|\boldsymbol{y}_t)$ denotes the MDM-Prime model. The weighting factor $w(t) = \frac{\alpha_t'}{\alpha_t - 1}$ is the same as standard MDMs. We elaborate on how to approximate the variational bound of MDM-Prime using Eq. (5) in Appendix A.1.3.

Adapting the model architecture from MDM to MDM-Prime requires only a simple modification of the embedding lookup table. In this setup, sub-token embeddings are aggregated into token embeddings and subsequently processed by the neural network at the token level. This design preserves the computational cost (FLOPs) of each training iteration. A detailed description of the model architecture is provided in Appendix A.2.1 and Fig. A2 (a).

3 Methodology

This section details two proposed enhancements: Section 3.1 explores improved function form for the subtokenizer and Section 3.2 discusses the token granularity selection.

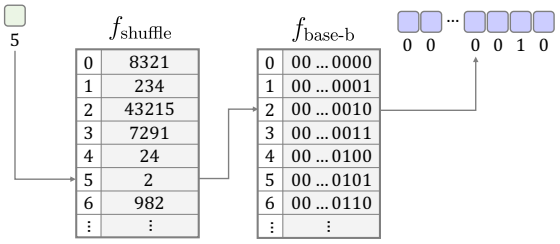


Figure 2: Implementation of the subtokenizer $f_\ell = f_{\text{base-}b} \circ f_{\text{shuffle}}$. The process utilizes two lookup tables to map indices efficiently; the inverse operation f_ℓ^{-1} is performed by reversing the lookup sequence. '5' denotes the token ID and '$00\cdots0010$' is its sub-token representation. See Appendix A.2.2 for a detailed description.

3.1 Increasing Entropy via Index Shuffling

Instead of defining $\boldsymbol{f}_\ell \triangleq \boldsymbol{f}_{\text{base-}b}$ as in MDM-Prime, this work derives the condition for $\boldsymbol{f}_\ell$ to minimize the loss function given a token granularity ℓ. Let $\boldsymbol{f}_{\text{shuffle}}$ denote an *Index Shuffling* operation, which permutes token indices via a lookup table (see Fig. 2). We propose $\boldsymbol{f}_\ell \triangleq \boldsymbol{f}_{\text{base-}b} \circ \boldsymbol{f}_{\text{shuffle}}$, which integrates an additional $\boldsymbol{f}_{\text{shuffle}}$ to effectively approximate the optimum.

To illustrate this design, we first characterize the effect of $\boldsymbol{f}_\ell$ on the objective (Eq. (5)). We isolate the term in the objective that depends on the transformation. We present this decomposition in **Proposition 3.1** (proof in Appendix A.1.2).

Proposition 3.1. *The loss can be decomposed into $\boldsymbol{f}_\ell$-independent and $\boldsymbol{f}_\ell$-dependent terms as follows:*

$$\inf_{\boldsymbol{f}_\ell}\inf_{p_\ell}\mathcal{L}^{(\ell)} = \int_0^1 w(t)\Big(\underbrace{\mathcal{H}(\boldsymbol{y}_0,\boldsymbol{y}_t)}_{\text{Independent on } \boldsymbol{f}_\ell.} - \sup_{\boldsymbol{f}_\ell}\underbrace{\mathcal{H}(\boldsymbol{y}_t)}_{\text{Dependent on } \boldsymbol{f}_\ell.}\Big)dt, \tag{6}$$

where $\mathcal{H}(\boldsymbol{y}_0, \boldsymbol{y}_t) \triangleq \mathbb{E}_{q_\alpha(\boldsymbol{y}_0,\boldsymbol{y}_t)}[-\log q_\alpha(\boldsymbol{y}_0, \boldsymbol{y}_t)]$ and $\mathcal{H}(\boldsymbol{y}_t) \triangleq \mathbb{E}_{q_\alpha(\boldsymbol{y}_t)}[-\log q_\alpha(\boldsymbol{y}_t)]$ represent the joint entropy of $\boldsymbol{y}_0, \boldsymbol{y}_t$ and the entropy of $\boldsymbol{y}_t$.

The $\boldsymbol{f}_\ell$-independent term corresponds to the joint negative entropy $-\mathcal{H}(\boldsymbol{y}_0, \boldsymbol{y}_t) = -(\mathcal{H}(\boldsymbol{y}_0) + \mathcal{H}(\boldsymbol{y}_t|\boldsymbol{y}_0))$, where $\mathcal{H}(\boldsymbol{y}_0) = \mathcal{H}(\boldsymbol{x}_0)$ remains constant due to the invertibility of $\boldsymbol{f}_\ell$, while $\mathcal{H}(\boldsymbol{y}_t|\boldsymbol{y}_0)$ is determined solely by the forward kernel in Eq. (4). On the other hand, the $\boldsymbol{f}_\ell$-dependent term suggests that the optimal $\boldsymbol{f}_\ell$ should maximize the entropy of $\boldsymbol{y}_t$, which reaches its optimum when each unmasked $y_t^{i,j}$ is uniformly distributed on $\mathcal{Y}$, as shown in **Proposition 3.2**:

Proposition 3.2. *The entropy of $\boldsymbol{y}_t$ is bounded:*

$$\mathcal{H}(\boldsymbol{y}_t) \le L\ell(h(\alpha_t) + \alpha_t \log b), \tag{7}$$

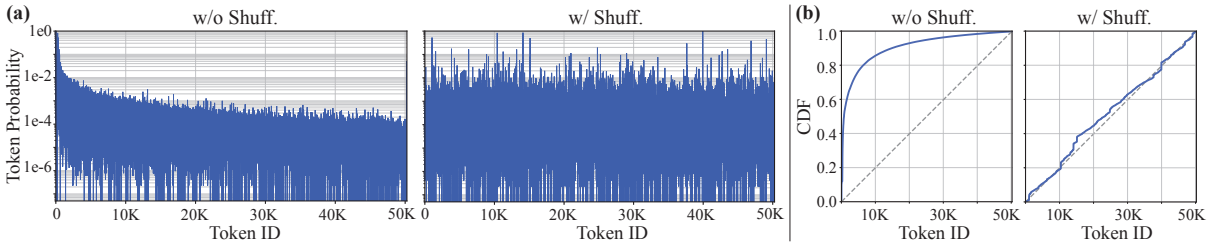


Figure 3: (a) Token probability and (b) cumulative distribution function (CDF) over token indices of the GPT-2 tokenizer (Radford et al., 2019) evaluated on the C4 dataset (Raffel et al., 2020). The left and right subplots of (a) and (b) show setups without and with index shuffling, respectively. The gray dashed line in (b) represents the CDF of a uniform token distribution.

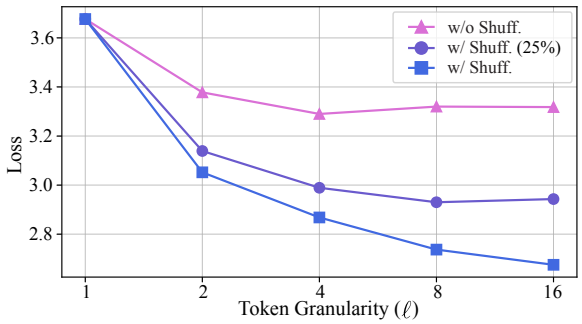


Figure 4: Ablation study on the loss used in MDM-Prime under three setups: 'w/o Shuff.,' 'w/ Shuff. (25%)', and 'w/ Shuff.' $\lceil \log_2 V \rceil = 16$ represents the maximum of ℓ. All models are trained using 10^{19} FLOPs. The experiments are conducted on C4.

where $h(\alpha_t) \triangleq -(1-\alpha_t)\log(1-\alpha_t) - \alpha_t \log \alpha_t$. *The equality holds if and only if each unmasked* $y_t^{i,j}$ *is uniformly distributed on* $\mathcal{Y}$.

Although **Propositions 3.1** and **3.2** identify high-entropy sub-tokens as the ideal case for optimality, the sub-tokens produced by directly applying base-b encoding to the token indices from commonly-used BPE tokenizers exhibit low entropy. An example of the GPT-2 tokenizer (Radford et al., 2019) is presented in the 'w/o Shuff.' column of Table 1. This occurs because BPE is constructed by iteratively merging the most frequent subword pairs, appending each newly merged pair as a new token index until the desired vocabulary size is reached. By construction, more frequent tokens receive lower indices, while less frequent tokens receive higher indices. Therefore, without deliberately shuffling the token indices, token probability is inversely proportional to the token index (see the left subplots with title 'w/o Shuff' in Fig. 3 (a) and (b)). Directly encoding these token indices using base-b encoding results in sub-tokens with low entropy, contradicting the maximization goal in Eqs. (6) and (7). To effectively disrupt the inherent token index structure, we propose the following technique:

	w/o Shuff.	w/ Shuff. (25%)	w/ Shuff.	Maximum
$\ell = 16$	0.8146	0.9038	0.9936	1.0000
$\ell = 8$	1.6152	1.8040	1.9811	2.0000
$\ell = 4$	3.2051	3.5360	3.8553	3.9069
$\ell = 2$	6.1489	6.7880	7.2803	7.8138

Table 1: Entropy of sub-tokens across different ℓ. The results are evaluated on C4. The maximum entropy is given by $-\log_2 \frac{1}{b}$, where $b = \lceil \sqrt[\ell]{V} \rceil$ and $V = 50,257$. $\lceil \log_2 V \rceil = 16$ represents the maximum of ℓ. Three configurations are compared: (1) 'w/o Shuff.,' the baseline setup without index shuffling, (2) 'w/ Shuff. (25%),' the setup where the shuffling operation is applied only to the first 25% of the token indices, and (3) 'w/ Shuff.,' the setup where token indices are fully shuffled.

Technique 1. (Index Shuffling) Randomly shuffle the token indices before performing $\boldsymbol{f}_{\text{base-}b}$.

An illustrative example is provided in Fig. 5 (a) to demonstrate how randomly shuffled token indices lead to higher sub-token entropy. By applying Technique 1, the average entropy approaches the theoretical maximum ($-\log_2 \frac{1}{b}$), as demonstrated in the 'w/ Shuff.' and 'w/ Shuff. (25%)' columns of Table 1. As further illustrated in Fig. 4, the loss decreases substantially when this shuffling operation is applied, and the improvement holds across all choices of ℓ used in $\boldsymbol{f}_{\text{base-}b}$. Intuitively, the reduction in loss stems from increased certainty in the conditional (predictive) distribution $q_{\text{data_y}}(\boldsymbol{y}_0|\boldsymbol{y}_t)$. As depicted in Fig. 5 (b), the index shuffling operation scatters similar probability masses across different slots. Therefore, when a specific sub-token value is observed, the conditional distribution becomes more certain, leading to a lower loss and improved sub-token prediction.

3.2 Selection for Token Granularity

The value of ℓ is critical to the performance of MDM-Prime. However, its influence on the objective and the criteria for a reliable selection have not been well explored. In this section, we propose set-

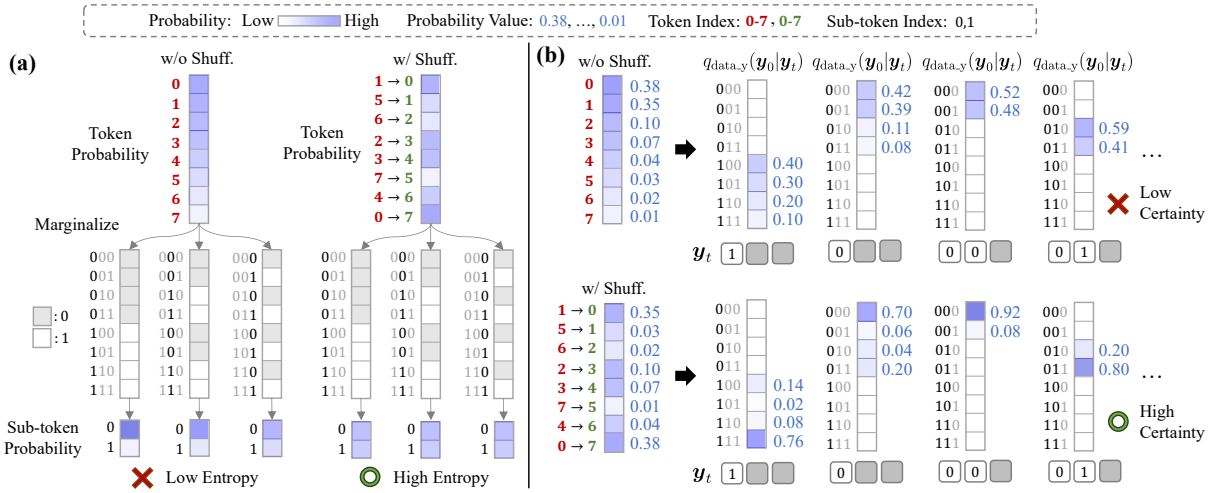


Figure 5: Illustrative examples of the index shuffling operation proposed in Technique 1. In this plot, $V = 8$, $\ell = 3$ and $b = 2$. Color intensity represents probability values, where darker blue denotes higher probability. (a) Marginal distributions: Marginalization is performed by summing entries corresponding to sub-token indices 0 (gray) and 1 (white), respectively. Sub-tokens encoded from standard token indices (i.e., 'w/o Shuff.') exhibit low entropy, while the shuffling operation results in sub-tokens with higher entropy. A detailed numerical example is provided in Fig. A3 in Appendix. (b) Conditional (predictive) distribution: The distribution $q_{\text{data_y}}(\boldsymbol{y}_0|\boldsymbol{y}_t)$ exhibits higher certainty under index shuffling, which results in improved performance.

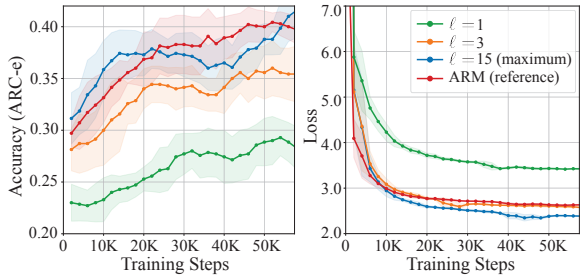


Figure 6: Zero-shot accuracy on ARC-e of MDM-Prime under different ℓ selections. Accuracy and loss exhibit consistent trend. ARM is included as a reference. All models have 170M parameters and are trained on SlimPajama (Soboleva et al., 2023) tokenized using LLaMA tokenizer with $V = 32,000$. The maximum ℓ is $\lceil \log_2 V \rceil = 15$.

ting ℓ to its maximum viable value, $\ell = \lceil \log_2 V \rceil$.

To justify this choice, we first establish in **Proposition 3.3** that the optimal MDM-Prime loss is monotonically non-increasing in ℓ:

Proposition 3.3. *Let ℓ_1, ℓ_2 be token granularities satisfying $1 < \ell_1 < \ell_2$ and $\frac{\ell_2}{\ell_1} \in \mathbb{N}$. The following inequalities hold:*

$$\inf_p \mathcal{L} \geq \inf_{p_{\ell_1}} \mathcal{L}^{(\ell_1)} \geq \inf_{p_{\ell_2}} \mathcal{L}^{(\ell_2)}. \tag{8}$$

According to **Proposition 3.3**, increasing ℓ leads to lower loss value (Eq. (5)). To validate this trend on practical applications, we conduct an experiment by varying the token granularity ℓ and evaluate downstream accuracy on the ARC-e (easy) (Clark et al., 2018) benchmark (see Fig. 6).

We observe that the downstream performance of MDM-based approaches is correlated with loss: lower loss consistently leads to better downstream performance, suggesting that the loss serves as a reliable proxy for downstream task quality. In Section 4.2, we further show that this phenomenon is applicable when the model scales up. Compared to the scaled up MDM model with $\ell = 1$ (SMDM (Nie et al., 2025a)), MDM-Prime-v2 ($\ell = \lceil \log_2 V \rceil$) exhibits improvement to the average accuracy on eight zero-shot benchmarks. Table A7 in Appendix A.4.4 further provides a detailed ablation study on both Techniques 1 and 2 for downstream tasks. Based on the above findings, we propose the following selection principle for ℓ:

Technique 2. (Binary Encoding) Select $\ell = \lceil \log_2 V \rceil$ such that $\boldsymbol{f}_{\text{base-}b}$ encodes tokens into binary sub-tokens.

In summary, Techniques 1 and 2 suggest the following subtokenizer: $\boldsymbol{f}_\ell = \boldsymbol{f}_{\text{base-}b} \circ \boldsymbol{f}_{\text{shuffle}}$, where $\boldsymbol{f}_{\text{shuffle}}$ maps the original token indices into shuffled ones, while $\boldsymbol{f}_{\text{base-}b}$ performs binary encoding. The entire operation is implemented using lookup tables (Fig. 2), requiring zero FLOPs and can be performed during data preprocessing. Further specifications are available in Appendix A.2.2.

4 Experiments

This section evaluates MDM-Prime-v2 via scaling analysis (Section 4.1), 1.1B-scale pretraining (Sec-

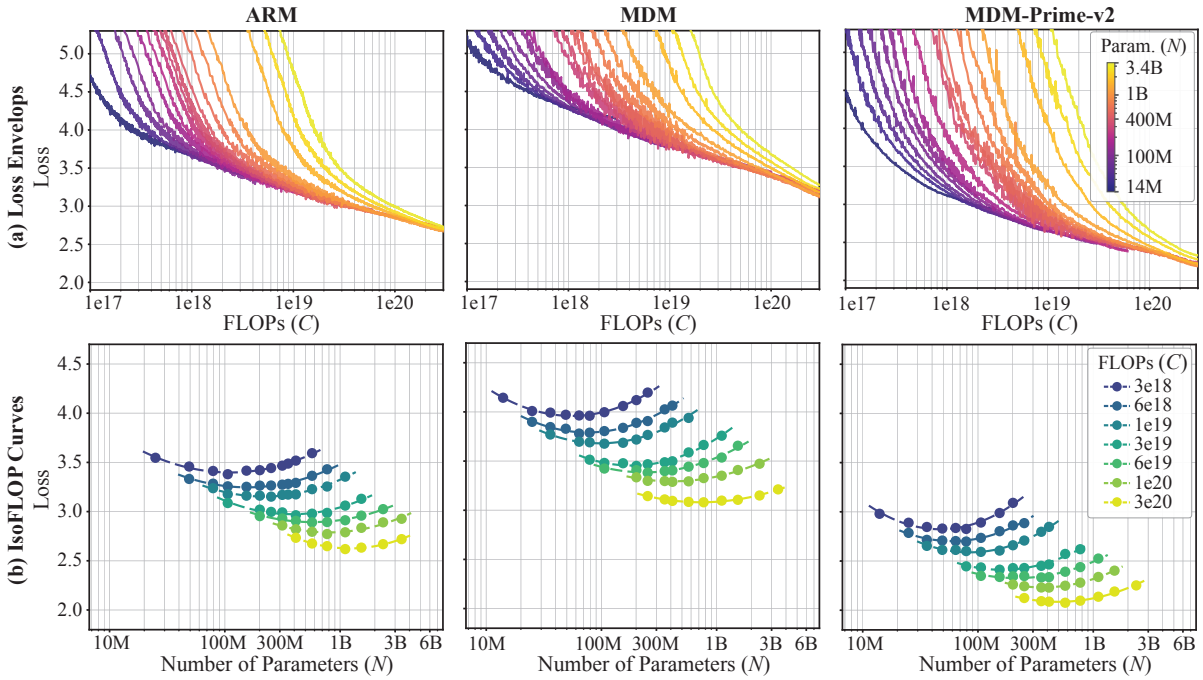


Figure 7: The (a) loss envelopes and (b) isoFLOP curves of ARM, MDM, and MDM-Prime-v2. In subplots (a), the number of parameters (N) ranges from 14M (purple) to 3.4B (yellow). In subplots (b), the compute budget (C) ranges from 3×10^{18} FLOPs (blue) to 3×10^{20} FLOPs (yellow). Please note that each model family optimizes a different training objective, the absolute loss values are not directly comparable across methods.

tion 4.2), and analysis on attention patterns and the singular value spectra of its projection weights (Section 4.3). Training details are provided in Appendix A.3. In Appendix A.4, we present experimental results on a 7B-parameter MDM-Prime-v2 pre-training run, validate the robustness of $\boldsymbol{f}_{\text{shuffle}}$ to random seed initialization, and discuss why subtokenization is ineffective for autoregressive models. A detailed ablation study on Techniques 1 and 2 for downstream tasks is presented in Table A7 in Appendix A.4.4. This section focuses on scaling behaviors and benchmark results.

4.1 Loss Behavior and Scaling Properties

As established in (Kaplan et al., 2020; Hoffmann et al., 2022), the training loss of language models exhibits a strong correlation with the training FLOPs (C). This compute budget is primarily determined by two configuration factors: the total number of training tokens (D) and the number of non-embedding parameters (N). To understand how these factors influence the scaling behavior of MDM-Prime-v2, we analyze its training loss dynamics across various combinations of N and D under fixed compute budgets ranging from 3×10^{18} to 3×10^{20} FLOPs. For these experiments, we employ a LLaMA-styled architecture (Touvron et al., 2023) (see Appendix A.3.1 for specification). All models are trained on C4 (Raffel et al., 2020) using the GPT-2 tokenizer with $V = 50,257$.

Coefficient	$\hat{a}$ ($N_{\text{opt}} \propto C^{\hat{a}}$)	$\hat{b}$ ($D_{\text{opt}} \propto C^{\hat{b}}$)
ARM	0.52	0.48
MDM	0.50	0.50
MDM-Prime-v2 (ℓ=16)	0.50	0.50

Table 2: The coefficients ($\hat{a}$ and $\hat{b}$) derived using the Chinchilla scaling law. A larger $\hat{a}$ indicates that compute resources should prioritize model parameters, while a larger $\hat{b}$ indicates that compute resources should prioritize training tokens. The other coefficients (α, β, A, B, and E) are provided in Table A3 in Appendix.

Figs. 7 (a) and (b) present the loss envelopes and isoFLOP curves for ARM, MDM, and MDM-Prime-v2. As shown in Fig. 7 (a), the training loss for all three methods decreases consistently as the total compute budget increases. This confirms that the loss of each method scales effectively with training FLOPs. Fig. 7 (b) compares the isoFLOP contours across fixed compute budgets. We note that because each model family optimizes a different training objective, the absolute loss values are not directly comparable across methods; we therefore focus on the scaling behavior within each family. We analyze the loss behavior using the Chinchilla law (Hoffmann et al., 2022). Using our empirical observations, consisting of (loss, N, D) triplets, we fit the power-law loss estimator:

	SciQ	SocialIQA	McTaco	TruthfulQA	BoolQ	ANLI	ARC-e	OBQA	Avg.
Reference Models									
GPT-Neo (1.3B) (Black et al., 2021)	77.10	41.25	42.89	23.13	61.99	33.13	50.21	33.60	45.41
OPT (1.3B) (Zhang et al., 2022)	76.70	40.63	37.08	23.75	57.83	33.86	50.97	33.40	44.28
Pythia (1.4B) (Biderman et al., 2023)	79.20	40.94	54.25	22.77	**63.15**	33.29	**53.91**	33.20	47.59
Bloom (1.1B) (BigScience, 2023)	74.60	39.05	53.63	25.58	59.08	33.35	45.41	29.40	45.01
Aligned Models									
SMDM (1.1B) (Nie et al., 2025a)	81.20	41.04	35.07	24.60	62.17	32.81	46.13	33.40	44.55
TinyLLaMA (1.1B) (Zhang et al., 2024)	80.90	39.56	40.88	20.93	59.20	33.25	50.13	33.40	44.78
MDM-Prime-v2 (1.1B) **(Ours)**	**83.30**	**42.02**	**66.14**	**25.83**	62.05	**34.24**	47.81	**34.00**	**49.42**

Table 3: Zero-shot accuracy on eight commonsense reasoning benchmarks, where higher values indicate better performance. SMDM uses 1,024 Monte Carlo steps to approximate the score of each answer option, whereas MDM-Prime-v2 uses the same number of steps or fewer, as summarized in Table A4. Methods under 'Aligned Models' share an identical training setup (dataset, compute budget, architecture, and optimization hyperparameters). Methods under 'Reference Models' are similarly sized models trained under different setups and are included for reference only.

$\hat{\mathcal{L}}(N, D) = E + \frac{A}{N^{\alpha}} + \frac{B}{D^{\beta}}$, where α, β, A, B, and E are coefficients determined via regression. Under a fixed compute budget $C \approx 6ND$, the optimal allocation of parameters (N_{opt}) and tokens (D_{opt}) is derived as follows:

$$N_{\text{opt}} = G\left(\frac{C}{6}\right)^{\hat{a}}, \quad D_{\text{opt}} = G^{-1}\left(\frac{C}{6}\right)^{\hat{b}}, \quad (9)$$

where $G = \left(\frac{\alpha A}{\beta B}\right)^{\frac{1}{\alpha+\beta}}$, $\hat{a} = \frac{\beta}{\alpha+\beta}$, and $\hat{b} = \frac{\alpha}{\alpha+\beta}$. As shown in Table 2, ARM exhibits the largest $\hat{a}$ and the smallest $\hat{b}$, indicating that the compute-optimal configuration of ARM prioritizes increasing model capacity (N) over data volume (D). In contrast, MDM and MDM-Prime-v2 yield the smallest $\hat{a}$ and the largest $\hat{b}$, suggesting that its compute-optimal performance is driven more by increasing training tokens than by expanding model parameters. This property is practically appealing: as current pretraining recipes (e.g., LLaMA (Touvron et al., 2023)) increasingly focus on scaling smaller models through longer training for efficient inference, a framework that achieves small $\hat{a}$ and large $\hat{b}$ offers a path to improved capabilities. These coefficients determine the compute-optimal frontier for each method and serve as a practical guide for configuring MDM-Prime-v2 under a compute budget. We offer further scaling guidelines in Appendix A.4.5.

4.2 Improvement at Larger-Scale Pretraining

Commonsense Reasoning. In this experiment, we adopt the training configuration of TinyLLaMA (ARM) (Zhang et al., 2024) and SMDM (MDM) (Nie et al., 2025a) to train a 1.1B parameter model on 540B tokens from the Slimpajama dataset (Soboleva et al., 2023) (totaling 3.3×10^{21} FLOPs). We compare the models on a wide range of commonsense reasoning tasks, with the descriptions of each task shown in Table A5 in Appendix. The model architecture and tokenizer are based on LLaMA.

Table 3 compares MDM-Prime-v2 against several pretrained ARM and MDM baselines of similar size: GPT-Neo (1.3B), OPT (1.3B), Pythia (1.4B), Bloom (1.1B), SMDM (1.1B), and TinyLLaMA (1.1B). MDM-Prime-v2 achieves the highest average accuracy across the tasks, outperforming these baselines on six of the eight tasks. In particular, our model demonstrates advantages in temporal reasoning and scientific question answering, delivering noticeable gains on SciQ and McTaco.

	SMDM (1.1B)		MDM-Prime-v2 (1.1B)	
Steps	Gen PPL (↓)	Entropy (↑)	Gen PPL (↓)	Entropy (↑)
128	129.09	4.55	122.80	4.60
256	118.00	4.36	109.08	4.36
512	86.08	4.23	79.82	4.24
768	81.01	3.92	66.33	3.99
1024	77.24	3.77	53.06	3.79

Table 4: Sample quality comparison between SMDM (1.1B) and MDM-Prime-v2 (1.1B) across varying sampling steps, measured by Gen PPL and entropy. MDM-Prime-v2 consistently achieves lower Gen PPL with comparable or higher entropy, with the quality gap widening at larger step counts.

Sample Quality. We compare the sample quality of SMDM and MDM-Prime-v2 using generative perplexity (Gen PPL) and entropy as metrics to assess the fidelity-diversity trade-off. Gen PPL is evaluated using TinyLLaMA trained on 3T tokens. As shown in Table 4, MDM-Prime-v2 consistently achieves lower Gen PPL with equal or higher entropy, demonstrating improved sample quality over

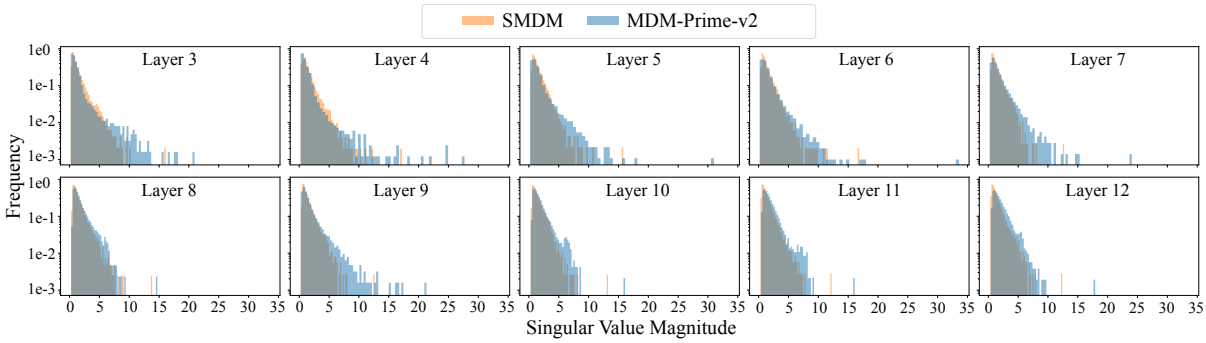


Figure 8: Singular value distributions of the projection matrices of SMDM and MDM-Prime-v2. Both models adopt the same architecture containing 20 layers. A complete plot of all layers is shown in Fig. A6.

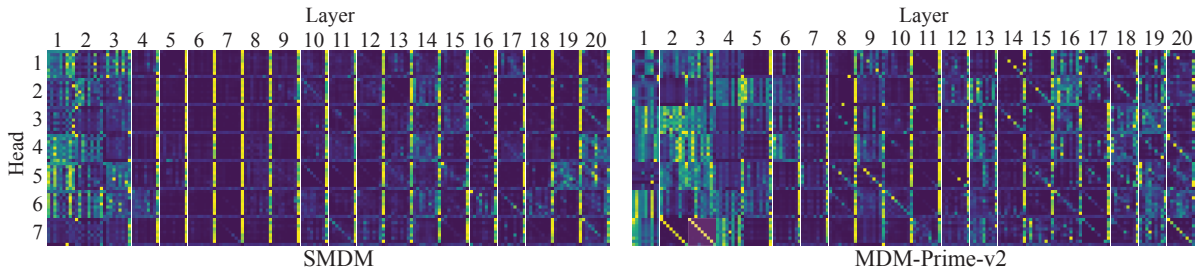


Figure 9: Attention score patterns of SMDM and MDM-Prime-v2. Both models contain 20 layers and 14 attention heads. A complete plot of all heads is shown in Fig. A7. MDM exhibits vertical stripes across Layers 4–20. In contrast, MDM-Prime-v2 displays diagonal structures and a richer diversity of patterns.

SMDM. Furthermore, we observe that this improvement becomes more pronounced as the number of sampling steps increases, evidenced by the larger Gen PPL gap at 768 and 1024 steps.

4.3 Analyses on Attention Sink Pattern and Singular Value Spectra

Analyzing attention score patterns (Clark et al., 2019b; Xiao et al., 2024) and the singular value spectra of self-attention weights (Martin and Mahoney, 2019; Dong et al., 2021) serves as a diagnostic tool for understanding model capacity utilization and trainability. In this section, we examine the attention mechanisms and spectral properties of SMDM and MDM-Prime-v2 to elucidate the factors driving the latter's superior performance.

We observe a divergence in the attention score patterns between SMDM and MDM-Prime-v2. Fig. A7 provides an illustration of the difference. SMDM exhibits signs of *Attention Sink* (Xiao et al., 2024), which is characterized by a single vertical stripe in the attention score matrices. This indicates that a significant proportion of heads attend to specific hidden representations. These representations function as learned 'no-op' registers, absorbing excess attention probability when no relevant context is present, which ultimately results in a lack of specialized routing for the attention operation. In contrast, MDM-Prime-v2 displays a richer diversity of attention patterns with more attention score matrices exhibiting sharp diagonal lines. These patterns suggest that MDM-Prime-v2 learns more specialized routing mechanisms, which potentially leads to improved representational capacity.

Spectral analysis of query-key-value projection matrices ($\mathbf{W}_{\text{qkv}}$) serves as an indicator of the utilization of model capacity (Martin and Mahoney, 2019; Dong et al., 2021). The singular value spectra for SMDM and MDM-Prime-v2 are presented in Fig. 8. MDM-Prime-v2 exhibits a heavier-tailed distribution compared to SMDM. This slower spectral decay indicates a reduction in rank collapse. Quantitatively, this is measured by the stable rank ($\|\mathbf{W}_{\text{qkv}}\|_F^2/\|\mathbf{W}_{\text{qkv}}\|_2^2$): MDM-Prime-v2 achieves a stable rank of 10.0, surpassing SMDM's 8.3. The result suggests that MDM-Prime-v2 has improved ability to capture diverse features rather than overfocusing on a single dominant feature.

5 Conclusion

We present MDM-Prime-v2, an enhanced and scaled-up version of MDM-Prime. Our analysis reveals the sub-optimality of base-b encoding for $\boldsymbol{f}_\ell$ with BPE tokenizers, and identifies a critical connection between the token granularity ℓ and the objective. We resolve these issues by employing an index shuffling operation in the subtokenizer

design and selecting $\ell = \lceil \log_2 V \rceil$. We characterize the scaling behavior of MDM-Prime-v2 across varying model and data scales, and demonstrate its practical effectiveness at the 1.1B-parameter scale, where it achieves improvements across eight zero-shot reasoning benchmarks. Our analysis of attention score patterns and singular value distributions further reveals that MDM-Prime-v2 learns more specialized routing mechanisms and exhibits larger stable ranks in its linear weights, indicating improved representational capacity.

Limitations

MDM-Prime-v2 adopts a random token index shuffling operation as a simple implementation to address distributional issues caused by the BPE tokenizer. However, **Proposition 3.2** suggests that the optimal design is a function that maximizes the entropy of $\boldsymbol{y}_t$. Although our experimental results demonstrate that index shuffling effectively increases entropy, yielding a near-optimal design, we anticipate that a rigorously derived algorithmic approach could further enhance performance. As an ablation, we compare index shuffling against a greedy assignment algorithm that maximizes entropy given corpus token-frequency statistics; results are reported in Appendix A.6. Greedy assignment achieves marginally higher entropy but brings negligible performance improvement. Given that it requires additional corpus statistics whereas index shuffling is fully off-the-shelf, we recommend index shuffling for practical use. Nevertheless, designing entropy-maximizing assignment strategies for subtokenizer represents a promising direction for future work to enhance MDM-Prime-v2, and Appendix A.6 discusses two further complementary avenues for extending this work.

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 the support from the National Science and Technology Council (NSTC) in Taiwan under grant numbers NSTC 114-2221-E-002-069-MY3, NSTC 113-2221-E-002-212-MY3, NSTC 115-2634-F-002-012, and NSTC 115-2218-E-A49-010, as well as the support from the Academia Sinica Scholar Award (ASSA) under grant number AS-ASSA-115-02, NTU Artificial Intelligence Center of Research Excellence, and Taiwan Centers of Excellence in Artificial Intelligence. This research was also supported by the NVIDIA Academic Grant Program. The authors would also like to express their appreciation for the hardware grant donation of the GPUs from NVIDIA Corporation and NVIDIA AI Technology Center (NVAITC) used in this work. Furthermore, the authors extend their gratitude to the National Center for High-Performance Computing (NCHC) for providing computational and storage resources. The authors also thank the NVIDIA Taipei-1 supercomputer for providing essential computing resources. Finally, we are grateful to Viet Nguyen, Xiaowen Zhang, and Vahid Belazadeh Meresht for their thoughtful review and valuable feedback on this work.

References

Joshua Ainslie, James Lee-Thorp, Michiel de Jong, Yury Zemlyanskiy, Federico Lebrón, and Sumit Sanghai. 2023. GQA: Training Generalized Multi-Query Transformer Models from Multi-Head Checkpoints. In *Proc. Conf. Empirical Methods in Natural Language Processing (EMNLP)*.

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

Stella Biderman, Hailey Schoelkopf, Quentin Gregory Anthony, Herbie Bradley, Kyle O'Brien, Eric Hallahan, Mohammad Aflah Khan, Shivanshu Purohit, USVSN Sai Prashanth, Edward Raff, and 1 others. 2023. Pythia: A suite for analyzing large language models across training and scaling. In *Proc. Int. Conf. on Machine Learning (ICML)*.

BigScience. 2023. BLOOM: A 176B-Parameter Open-Access Multilingual Language Model.

Yonatan Bisk, Rowan Zellers, Ronan Le Bras, Jianfeng Gao, and Yejin Choi. 2020. PIQA: Reasoning about Physical Commonsense in Natural Language. In *AAAI Conference on Artificial Intelligence (AAAI)*.

Sid Black, Leo Gao, Phil Wang, Connor Leahy, and Stella Biderman. 2021. GPT-Neo: Large Scale Autoregressive Language Modeling with Mesh-Tensorflow.

Andrew Campbell, Joe Benton, Valentin De Bortoli, Tom Rainforth, George Deligiannidis, and Arnaud Doucet. 2022. A Continuous Time Framework for

Discrete Denoising Models. In *Proc. of Int. Conf. on Neural Information Processing Systems (NeurIPS)*.

Huiwen Chang, Han Zhang, Lu Jiang, Ce Liu, and William T. Freeman. 2022. MaskGIT: Masked Generative Image Transformer. In *Conf. on Computer Vision and Pattern Recognition (CVPR)*.

Chen-Hao Chao, Wei-Fang Sun, Hanwen Liang, Chun-Yi Lee, and Rahul G. Krishnan. 2025. Beyond Masked and Unmasked: Discrete Diffusion Models via Partial Masking. In *Proceedings of the Conference on Neural Information Processing Systems (NeurIPS)*.

Christopher Clark, Kenton Lee, Ming-Wei Chang, Tom Kwiatkowski, Michael Collins, and Kristina Toutanova. 2019a. BoolQ: Exploring the Surprising Difficulty of Natural Yes/No Questions. In *Proc. Conf. of the North American Chapter of the Association for Computational Linguistics (NAACL)*.

Kevin Clark, Urvashi Khandelwal, Omer Levy, and Christopher D. Manning. 2019b. What Does BERT Look At? An Analysis of BERT's Attention. In *Proc. BlackboxNLP Workshop at Annual Meeting of the Association for Computational Linguistics (ACL)*.

Peter Clark, Isaac Cowhey, Oren Etzioni, Tushar Khot, Ashish Sabharwal, Carissa Schoenick, and Oyvind Tafjord. 2018. Think you have Solved Question Answering? Try ARC, the AI2 Reasoning Challenge.

Tri Dao, Daniel Y. Fu, Stefano Ermon, Atri Rudra, and Christopher Ré. 2022. FlashAttention: Fast and Memory-Efficient Exact Attention with IO-Awareness.

Mostafa Dehghani, Josip Djolonga, Basil Mustafa, Piotr Padlewski, Jonathan Heek, Justin Gilmer, Andreas Peter Steiner, Mathilde Caron, Robert Geirhos, Ibrahim Alabdulmohsin, Rodolphe Jenatton, Lucas Beyer, Michael Tschannen, Anurag Arnab, Xiao Wang, Carlos Riquelme Ruiz, Matthias Minderer, Joan Puigcerver, Utku Evci, and 23 others. 2023. Scaling Vision Transformers to 22 Billion Parameters. In *Proc. Int. Conf. on Machine Learning (ICML)*.

Yihe Dong, Jean-Baptiste Cordonnier, and Andreas Loukas. 2021. Attention is Not All You Need: Pure Attention Loses Rank Doubly Exponentially with Depth. In *Proc. Int. Conf. on Machine Learning (ICML)*.

Leo Gao, Jonathan Tow, Baber Abbasi, Stella Biderman, Sid Black, Anthony DiPofi, Charles Foster, Laurence Golding, Jeffrey Hsu, Alain Le Noac'h, Haonan Li, Kyle McDonell, Niklas Muennighoff, Chris Ociepa, Jason Phang, Laria Reynolds, Hailey Schoelkopf, Aviya Skowron, Lintang Sutawika, and 5 others. 2024. The Language Model Evaluation Harness.

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

Aaron Gokaslan, Vanya Cohen, Ellie Pavlick, and Stefanie Tellex. 2019. Openwebtext corpus.

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

Jordan Hoffmann, Sebastian Borgeaud, Arthur Mensch, Elena Buchatskaya, Trevor Cai, Eliza Rutherford, Diego de Las Casas, Lisa Anne Hendricks, Johannes Welbl, Aidan Clark, Tom Hennigan, Eric Noland, Katie Millican, George van den Driessche, Bogdan Damoc, Aurelia Guy, Simon Osindero, Karen Simonyan, Erich Elsen, and 3 others. 2022. Training Compute-Optimal Large Language Models. In *Proc. of the Int. Conf. on Neural Information Processing Systems (NeurIPS)*.

Jared Kaplan, Sam McCandlish, Tom Henighan, Tom B. Brown, Benjamin Chess, Rewon Child, Scott Gray, Alec Radford, Jeffrey Wu, and Dario Amodei. 2020. Scaling Laws for Neural Language Models.

Diederik P. Kingma, Tim Salimans, Ben Poole, and Jonathan Ho. 2021. Variational Diffusion Models. In *Proc. of the Int. Conf. on Neural Information Processing Systems (NeurIPS)*.

Benjamin Lefaudeux, Francisco Massa, Diana Liskovich, Wenhan Xiong, Vittorio Caggiano, Sean Naren, Min Xu, Jieru Hu, Marta Tintore, Susan Zhang, Patrick Labatut, Daniel Haziza, Luca Wehrstedt, Jeremy Reizenstein, and Grigory Sizov. 2022. xFormers: A modular and hackable Transformer modelling library.

Lightning-AI. 2023. Lit-gpt. *arXiv preprint arXiv:1909.08053*.

Stephanie Lin, Jacob Hilton, and Owain Evans. 2022. TruthfulQA: Measuring how models mimic human falsehoods. In *Proc. Association for Computational Linguistics (ACL)*, pages 3214–3252.

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

Charles H. Martin and Michael W. Mahoney. 2019. Traditional and Heavy-Tailed Self Regularization in Neural Network Models.

Todor Mihaylov, Peter Clark, Tushar Khot, and Ashish Sabharwal. 2018. Can a Suit of Armor Conduct Electricity? A New Dataset for Open Book Question Answering.

Jinjie Ni, Qian Liu, Chao Du, Longxu Dou, Hang Yan, Zili Wang, Tianyu Pang, and Michael Qizhe Shieh. 2025. Training Optimal Large Diffusion Language Models.

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

Shen Nie, Fengqi Zhu, Zebin You, Xiaolu Zhang, Jingyang Ou, Jun Hu, Jun Zhou, Yankai Lin, Ji-Rong Wen, and Chongxuan Li. 2025b. Large Language Diffusion Models. In *Proc. of Int. Conf. on Neural Information Processing Systems (NeurIPS)*.

Yixin Nie, Adina Williams, Emily Dinan, Mohit Bansal, Jason Weston, and Douwe Kiela. 2020. Adversarial NLI: A New Benchmark for Natural Language Understanding. In *Proc. Annual Meeting of the Association for Computational Linguistics (ACL)*.

Krishna Pillutla, Swabha Swayamdipta, Rowan Zellers, John Thickstun, Sean Welleck, Yejin Choi, and Zaid Harchaoui. 2021. MAUVE: Measuring the Gap Between Neural Text and Human Text using Divergence Frontiers. In *Proc. of Int. Conf. on Neural Information Processing Systems (NeurIPS)*.

Mihir Prabhudesai, Mengning Wu, Amir Zadeh, Katerina Fragkiadaki, and Deepak Pathak. 2025. Diffusion Beats Autoregressive in Data-Constrained Settings. In *Proc. of Int. Conf. on Neural Information Processing Systems (NeurIPS)*.

Alec Radford, Jeff Wu, Rewon Child, David Luan, Dario Amodei, and Ilya Sutskever. 2019. Language Models are Unsupervised Multitask Learners.

Colin Raffel, Noam Shazeer, Adam Roberts, Katherine Lee, Sharan Narang, Michael Matena, Yanqi Zhou, Wei Li, and Peter J. Liu. 2020. Exploring the Limits of Transfer Learning with a Unified Text-to-Text Transformer. *Journal of Machine Learning Research*, 21(1).

Hannah Rashkin, Asli Celikyilmaz, Yejin Choi, and Jianfeng Gao. 2020. PlotMachines: Outline-Conditioned Generation with Dynamic Plot State Tracking. In *Proc. Conf. Empirical Methods in Natural Language Processing (EMNLP)*.

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

Maarten Sap, Hannah Rashkin, Derek Chen, Ronan Le Bras, and Yejin Choi. 2019. Social IQa: Commonsense Reasoning about Social Interactions. In *Proc. Conf. Empirical Methods in Natural Language Processing (EMNLP)*.

Noam Shazeer. 2020. GLU Variants Improve Transformer.

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

Mohammad Shoeybi, Mostofa Patwary, Raul Puri, Patrick LeGresley, Jared Casper, and Bryan Catanzaro. 2019. Megatron-LM: Training Multi-Billion Parameter Language Models Using Model Parallelism. *arXiv preprint arXiv:1909.08053*.

Daria Soboleva, Faisal Al-Khateeb, Robert Myers, Jacob R Steeves, Joel Hestness, and Nolan Dey. 2023. SlimPajama: A 627B token cleaned and deduplicated version of RedPajama.

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

Jianlin Su, Yu Lu, Shengfeng Pan, Ahmed Murtadha, Bo Wen, and Yunfeng Liu. 2023. RoFormer: Enhanced Transformer with Rotary Position Embedding.

Olmo Team. 2025. Olmo 3.

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. 2023. LLaMA: Open and Efficient Foundation Language Models. `arXiv:2302.13971 [cs.CL]`.

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

Dimitri von Rütte, Janis Fluri, Omead Pooladzandi, Bernhard Schölkopf, Thomas Hofmann, and Antonio Orvieto. 2025. Scaling Behavior of Discrete Diffusion Language Models.

Johannes Welbl, Nelson F. Liu, and Matt Gardner. 2017. Crowdsourcing multiple choice science questions. In *Proc. Conf. Empirical Methods in Natural Language Processing (EMNLP)*.

Guangxuan Xiao, Yuandong Tian, Beidi Chen, Song Han, and Mike Lewis. 2024. Efficient Streaming Language Models with Attention Sinks. In *Proc. Int. Conf. on Learning Representations (ICLR)*.

Jiacheng Ye, Zhihui Xie, Lin Zheng, Jiahui Gao, Zirui Wu, Xin Jiang, Zhenguo Li, and Lingpeng Kong. 2025. Dream 7B: Diffusion Large Language Models.

Biao Zhang and Rico Sennrich. 2019. Root Mean Square Layer Normalization. In *Proc. of the Int. Conf. on Neural Information Processing Systems (NeurIPS)*.

Peiyuan Zhang, Guangtao Zeng, Tianduo Wang, and Wei Lu. 2024. Tinyllama: An open-source small language model.

Susan Zhang, Stephen Roller, Naman Goyal, Mikel Artetxe, Moya Chen, Shuohui Chen, Christopher Dewan, Mona Diab, Xian Li, Xi Victoria Lin, Todor Mihaylov, Myle Ott, Sam Shleifer, Kurt Shuster, Daniel Simig, Punit Singh Koura, Anjali Sridhar, Tianlu Wang, and Luke Zettlemoyer. 2022. OPT: Open Pre-trained Transformer Language Models.

Kaiwen Zheng, Yongxin Chen, Hanzi Mao, Ming-Yu Liu, Jun Zhu, and Qinsheng Zhang. 2025. Masked Diffusion Models are Secretly Time-Agnostic Masked Models and Exploit Inaccurate Categorical Sampling. In *Proc. Int. Conf. on Learning Representations (ICLR)*.

Ben Zhou, Daniel Khashabi, Qiang Ning, and Dan Roth. 2019. "Going on a vacation" takes longer than "Going for a walk": A Study of Temporal Commonsense Understanding. In *Proc. Conf. Empirical Methods in Natural Language Processing (EMNLP)*.

A Appendix

This appendix provides additional discussions and experiments. Section A.1 presents theoretical analyses. Section A.2 offers technical implementation details of the model architecture and subtokenizer of MDM-Prime-v2. Section A.3 elaborates on the experimental configurations. Section A.4 reports additional experimental results. Section A.5 walks through some related studies. Finally, Section A.6 outlines limitations and directions for future work.

A.1 Theoretical Analyses

In this section, we present theoretical derivations. In Section A.1.1, we present the proofs of **Proposition 3.3**, while Section A.1.2 establishes the justifications for **Propositions 3.1** and **3.2**. A table of symbols used in this paper is presented in Table A1.

Symbol	Definition	Symbol	Definition
L	token sequence length	V	number of classes a token can take
ℓ	sub-token sequence length	b	number of classes a sub-token can take
$\mathcal{X}$	a set of tokens: $\{0, \cdots, V-1\}$	$\tilde{\mathcal{X}}$	an augmented set of tokens: $\mathcal{X} \cup \{\mathtt{m}\}$
$\mathcal{Y}$	a set of sub-tokens: $\{0, \cdots, b-1\}$	$\tilde{\mathcal{Y}}$	an augmented set of sub-tokens: $\mathcal{Y} \cup \{\mathtt{m}\}$
$\boldsymbol{x}_0$	a sequence of clean tokens ($\boldsymbol{x}_0 \in \mathcal{X}^L$)	$\boldsymbol{x}_t$	a sequence of noised tokens ($\boldsymbol{x}_t \in \tilde{\mathcal{X}}^L$)
$\boldsymbol{y}_0$	a sequence of clean sub-tokens ($\boldsymbol{y}_0 \in \mathcal{Y}^{L\times\ell}$)	$\boldsymbol{y}_t$	a sequence of noised sub-tokens ($\boldsymbol{y}_t \in \tilde{\mathcal{Y}}^{L\times\ell}$)
$\delta_x(x')$	Kronecker delta (equals 1 only if $x = x'$)	α_t	scheduling function ($\alpha_t : [0,1] \to [0,1]$)
$\boldsymbol{f}_\ell(\boldsymbol{x}_0)$	invertible base-b encoding	$\boldsymbol{g}_\ell(\boldsymbol{y}_t)$	a function that maps between latent variables
$p(\boldsymbol{x}_0\|\boldsymbol{x}_t)$	MDM model (*carry-over* distribution of $\boldsymbol{x}_0$)	$p_\ell(\boldsymbol{y}_0\|\boldsymbol{y}_t)$	MDM-Prime model (*carry-over* distribution of $\boldsymbol{y}_0$)
$q_{\text{data}}(\boldsymbol{x}_0)$	data distribution of $\boldsymbol{x}_0$	$q_{\text{data_y}}(\boldsymbol{y}_0)$	data distribution of $\boldsymbol{y}_0$ ($q_{\text{data_y}} = q_{\text{data}} \circ \boldsymbol{f}_\ell^{-1}$)

Table A1: A table of symbols.

A.1.1 Proof of Propositions 3.3

We present our main findings (**Proposition 3.3** along with its equality condition) as a merged **Theorem A.1**. The proof relies on a series of auxiliary results provided in **Lemmas A.2-A.4** and **Propositions A.5-A.6**.

Theorem A.1. *Let p and p_ℓ denote the MDM and MDM-Prime models. Let ℓ_1, ℓ_2 be token granularities satisfying $1 < \ell_1 < \ell_2$ and $\frac{\ell_2}{\ell_1} \in \mathbb{N}$. The following inequalities hold:*

$$\inf_p \mathcal{L} \geq \inf_{p_{\ell_1}} \mathcal{L}^{(\ell_1)} \geq \inf_{p_{\ell_2}} \mathcal{L}^{(\ell_2)}. \tag{A1}$$

Let $\tilde{\ell} \in \mathbb{N}$ and $\boldsymbol{g}_{\tilde{\ell}}$ be a vectorized mapping with element-wise operation defined as:

$$g_{\tilde{\ell}}(\boldsymbol{y}_t^i) = \begin{cases} f_{\tilde{\ell}}^{-1}(\boldsymbol{y}_t^i), & \text{if } y_t^{i,j} \neq \mathtt{m}, \forall j \in \{1, ..., \tilde{\ell}\}, \\ \mathtt{m}, & \text{otherwise}, \end{cases} \tag{A2}$$

Given a scheduling function α_t and defining $\tilde{\alpha}_t = \alpha_t^{1/\tilde{\ell}}$, the first inequality in Eq. (A1) becomes an equality if and only if:

$$\mathbb{E}_{q_{\tilde{\alpha}}(\boldsymbol{y}_t)}\Big[\mathbb{D}_{\text{KL}}\big(q_{\text{data_y}}(\boldsymbol{y}_0 \mid \boldsymbol{y}_t) \,\|\, q_{\text{data_y}}(\boldsymbol{y}_0 \mid \boldsymbol{g}_{\tilde{\ell}}(\boldsymbol{y}_t))\big)\Big] = 0, \tag{A3}$$

where $\boldsymbol{y}_0 \in \mathcal{Y}^{L\times\ell_1}$, $\boldsymbol{y}_t \in \tilde{\mathcal{Y}}^{L\times\ell_1}$, $\tilde{\ell} = \ell_1$, and the second inequality becomes an equality if and only if Eq. (A3) holds with $\tilde{\ell} = \frac{\ell_2}{\ell_1}$, $\boldsymbol{y}_0 \in \mathcal{Y}^{L\times\ell_2}$, and $\boldsymbol{y}_t \in \tilde{\mathcal{Y}}^{L\times\ell_2}$.

Proof. The first inequality $\inf_p \mathcal{L} \geq \inf_{p_{\ell_1}} \mathcal{L}^{(\ell_1)}$ and the corresponding equality condition (Eq. (A3)) are established in **Proposition A.5** while the second inequality $\inf_{p_{\ell_1}} \mathcal{L}^{(\ell_1)} \geq \inf_{p_{\ell_2}} \mathcal{L}^{(\ell_2)}$ and the corresponding equality condition are established in **Proposition A.6**. Combining these two results yields the chain of inequalities presented in Eq. (A1). □

The proof of **Proposition A.5** is organized into three primary steps, as illustrated in Fig. A1:

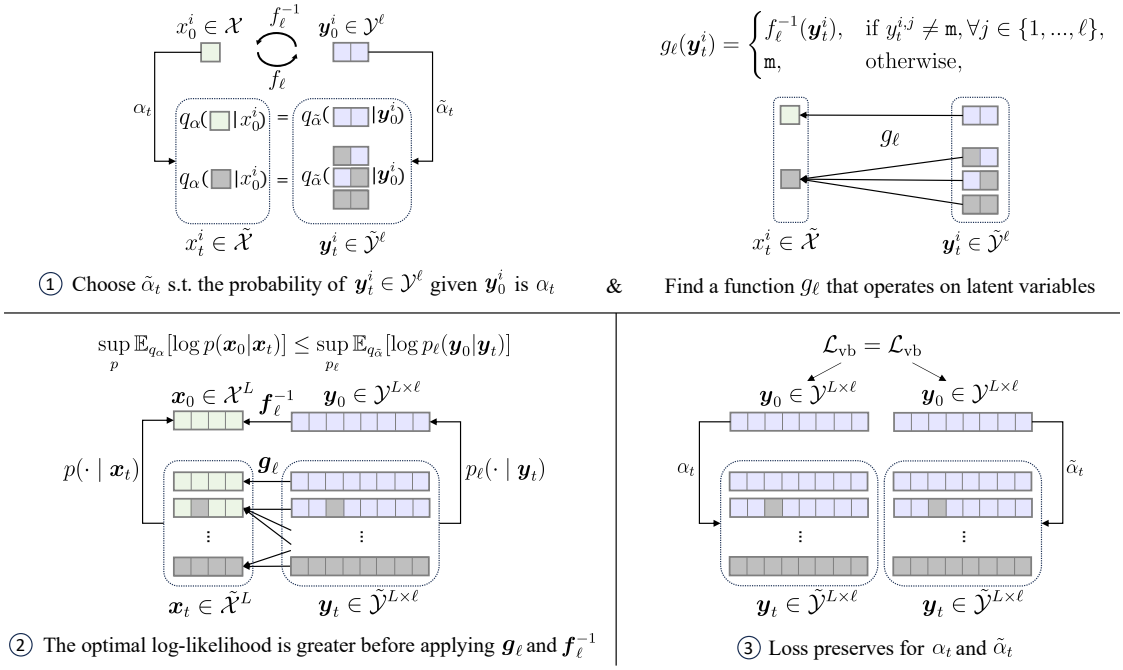


Figure A1: An illustration of the three key steps for proving **Proposition** A.5. Gray tiles represent masked tokens or sub-tokens. Green and purple tiles represent unmasked tokens and sub-tokens, respectively.

- **Step 1**: We identify a scheduling function $\tilde{\alpha}_t$ and a mapping $\boldsymbol{g}_\ell$ such that the expectations satisfy $\mathbb{E}_{q_\alpha(\boldsymbol{x}_0,\boldsymbol{x}_t)}[F(\boldsymbol{x}_0, \boldsymbol{x}_t)] = \mathbb{E}_{q_{\tilde{\alpha}}(\boldsymbol{y}_0,\boldsymbol{y}_t)}[F(\boldsymbol{f}_\ell^{-1}(\boldsymbol{y}_0), \boldsymbol{g}_\ell(\boldsymbol{y}_t))]$ for any arbitrary function $F : \mathcal{X}^L \times \tilde{\mathcal{X}}^L \to \mathbb{R}$. This equivalence is formally established in **Lemma A.2**.

- **Step 2**: We demonstrate that the logarithmic probability of MDM-Prime is point-wisely greater than or equal to that of MDM for every timestep $t \in [0, 1]$. This comparison is detailed in **Lemma A.3**.

- **Step 3**: Finally, we prove that the training objective remain invariant under different choices of the scheduling function, as presented in **Lemma A.4**.

By synthesizing these three results, we arrive at the inequality established in the proposition.

Lemma A.2. *Let $\boldsymbol{x}_t \sim q_\alpha(\cdot \mid \boldsymbol{x}_0)$ and $\boldsymbol{y}_t \sim q_{\tilde{\alpha}}(\cdot \mid \boldsymbol{y}_0)$, where $\tilde{\alpha}_t \triangleq \alpha_t^{1/\ell}$ and $\boldsymbol{y}_0 = \boldsymbol{f}_\ell(\boldsymbol{x}_0)$. In addition, let $\boldsymbol{g}_\ell : \tilde{\mathcal{Y}}^{L\times\ell} \to \tilde{\mathcal{X}}^L$ be a function defined in Eq. (A2). Given any mapping function $F : \mathcal{X}^L \times \tilde{\mathcal{X}}^L \to \mathbb{R}$, the following equality holds:*

$$\mathbb{E}_{q_\alpha(\boldsymbol{x}_0,\boldsymbol{x}_t)}[F(\boldsymbol{x}_0, \boldsymbol{x}_t)] = \mathbb{E}_{q_{\tilde{\alpha}}(\boldsymbol{y}_0,\boldsymbol{y}_t)}[F(\boldsymbol{f}_\ell^{-1}(\boldsymbol{y}_0), \boldsymbol{g}_\ell(\boldsymbol{y}_t))]. \tag{A4}$$

Proof.

$$\begin{aligned}
&\mathbb{E}_{q_{\tilde{\alpha}}(\boldsymbol{y}_0,\boldsymbol{y}_t)}[F(\boldsymbol{f}_\ell^{-1}(\boldsymbol{y}_0), \boldsymbol{g}_\ell(\boldsymbol{y}_t))] \\
&\overset{(i)}{=} \mathbb{E}_{q_{\text{data}}\circ\boldsymbol{f}_\ell^{-1}(\boldsymbol{y}_0)}\left[\sum_{\boldsymbol{y}_t} F(\boldsymbol{f}_\ell^{-1}(\boldsymbol{y}_0), \boldsymbol{g}_\ell(\boldsymbol{y}_t)) q_{\tilde{\alpha}}(\boldsymbol{y}_t \mid \boldsymbol{y}_0)\right] \\
&= \mathbb{E}_{q_{\text{data}}\circ\boldsymbol{f}_\ell^{-1}(\boldsymbol{y}_0)}\left[\sum_{\boldsymbol{y}_t} F(\boldsymbol{f}_\ell^{-1}(\boldsymbol{y}_0), \boldsymbol{g}_\ell(\boldsymbol{y}_t)) \prod_{i=1}^{L}\prod_{j=1}^{\ell} q_{\tilde{\alpha}}(y_t^{i,j} \mid y_0^{i,j})\right] \\
&\overset{(ii)}{=} \mathbb{E}_{q_{\text{data}}\circ\boldsymbol{f}_\ell^{-1}(\boldsymbol{y}_0)}\left[\sum_{\boldsymbol{y}_t} F(\boldsymbol{f}_\ell^{-1}(\boldsymbol{y}_0), \boldsymbol{g}_\ell(\boldsymbol{y}_t)) \prod_{i=1}^{L}\prod_{j=1}^{\ell} (1-\tilde{\alpha}_t)\delta_{\mathtt{m}}(y_t^{i,j}) + \tilde{\alpha}_t \delta_{y_0^i}(y_t^{i,j})\right]
\end{aligned}$$

$$= \mathbb{E}_{q_{\text{data}} \circ \boldsymbol{f}_\ell^{-1}(\boldsymbol{y}_0)} \left[\sum_{\boldsymbol{y}_t} F(\boldsymbol{f}_\ell^{-1}(\boldsymbol{y}_0), \boldsymbol{g}_\ell(\boldsymbol{y}_t)) \prod_{i=1}^{L} \prod_{j=1}^{\ell} (1-\alpha_t^{1/\ell}) \delta_{\mathtt{m}}(y_t^{i,j}) + \alpha_t^{1/\ell} \delta_{y_0^{i,j}}(y_t^{i,j}) \right]$$

$$= \mathbb{E}_{q_{\text{data}} \circ \boldsymbol{f}_\ell^{-1}(\boldsymbol{y}_0)} \Bigg[\sum_{\boldsymbol{y}_t} F(\boldsymbol{f}_\ell^{-1}(\boldsymbol{y}_0), \boldsymbol{g}_\ell(\boldsymbol{y}_t)) \prod_{i=1}^{L} (1-(\alpha_t^{1/\ell}))^\ell \delta_{\mathtt{m}}(y_t^{i,1}) \cdots \delta_{\mathtt{m}}(y_t^{i,\ell}) + \cdots$$
$$+ (\alpha_t^{1/\ell})^\ell \delta_{y_0^{i,1}}(y_t^{i,1}) \cdots \delta_{y_0^{i,\ell}}(y_t^{i,\ell}) \Bigg]$$

$$= \mathbb{E}_{q_{\text{data}} \circ \boldsymbol{f}_\ell^{-1}(\boldsymbol{y}_0)} \Bigg[\sum_{\boldsymbol{y}_t} F(\boldsymbol{f}_\ell^{-1}(\boldsymbol{y}_0), \boldsymbol{g}_\ell(\boldsymbol{y}_t)) \prod_{i=1}^{L} (1-(\alpha_t^{1/\ell}))^\ell \delta_{\mathtt{m},\cdots,\mathtt{m}}(y_t^{i,1}, \cdots, y_t^{i,\ell}) + \cdots$$
$$+ (\alpha_t^{1/\ell})^\ell \delta_{y_t^{i,1},\cdots,y_0^{i,\ell}}(y_0^{i,1}, \cdots, y_t^{i,\ell}) \Bigg]$$

$$= \mathbb{E}_{q_{\text{data}} \circ \boldsymbol{f}_\ell^{-1}(\boldsymbol{y}_0)} \left[\sum_{\boldsymbol{y}_t} F(\boldsymbol{f}_\ell^{-1}(\boldsymbol{y}_0), \boldsymbol{g}_\ell(\boldsymbol{y}_t)) \prod_{i=1}^{L} (1-(\alpha_t^{1/\ell}))^\ell \delta_{\mathtt{m},\cdots,\mathtt{m}}(\boldsymbol{y}_t^i) + \cdots + (\alpha_t^{1/\ell})^\ell \delta_{\boldsymbol{y}_0^i}(\boldsymbol{y}_t^i) \right]$$

$$= \mathbb{E}_{q_{\text{data}}(\boldsymbol{x}_0)} \left[\sum_{\boldsymbol{y}_t} F(\boldsymbol{x}_0, \boldsymbol{g}_\ell(\boldsymbol{y}_t)) \prod_{i=1}^{L} \underbrace{(1-(\alpha_t^{1/\ell}))^\ell \delta_{\mathtt{m},\cdots,\mathtt{m}}(\boldsymbol{y}_t^i) + \cdots}_{\text{Contains at least one } \mathtt{m}} + \underbrace{(\alpha_t^{1/\ell})^\ell \delta_{f_\ell(x_0^i)}(\boldsymbol{y}_t^i)}_{\text{Contains no } \mathtt{m}} \right]$$

$$\overset{(iii)}{=} \mathbb{E}_{q_{\text{data}}(\boldsymbol{x}_0)} \left[\sum_{\boldsymbol{y}_t} F(\boldsymbol{x}_0, \boldsymbol{g}_\ell(\boldsymbol{y}_t)) \prod_{i=1}^{L} (1-(\alpha_t^{1/\ell})^\ell) \delta_{\mathtt{m}}(g_\ell(\boldsymbol{y}_t^i)) + (\alpha_t^{1/\ell})^\ell \delta_{x_0^i}(g_\ell(\boldsymbol{y}_t^i)) \right]$$

$$= \mathbb{E}_{q_{\text{data}}(\boldsymbol{x}_0)} \left[\sum_{\boldsymbol{y}_t} F(\boldsymbol{x}_0, \boldsymbol{g}_\ell(\boldsymbol{y}_t)) \prod_{i=1}^{L} (1-\alpha_t) \delta_{\mathtt{m}}(g_\ell(\boldsymbol{y}_t^i)) + \alpha_t \delta_{x_0^i}(g_\ell(\boldsymbol{y}_t^i)) \right]$$

$$= \mathbb{E}_{q_{\text{data}}(\boldsymbol{x}_0)} \left[\sum_{\boldsymbol{y}_t} F(\boldsymbol{x}_0, \boldsymbol{g}_\ell(\boldsymbol{y}_t)) q_\alpha(\boldsymbol{g}_\ell(\boldsymbol{y}_t) \mid \boldsymbol{x}_0) \right]$$

$$\overset{(iv)}{=} \mathbb{E}_{q_{\text{data}}(\boldsymbol{x}_0)} \left[\sum_{\boldsymbol{x}_t} F(\boldsymbol{x}_0, \boldsymbol{x}_t) q_\alpha(\boldsymbol{x}_t \mid \boldsymbol{x}_0) \right]$$

$$= \mathbb{E}_{q_\alpha(\boldsymbol{x}_0, \boldsymbol{x}_t)}[F(\boldsymbol{x}_0, \boldsymbol{x}_t)].$$

In step (i), $q_{\tilde{\alpha}}(\boldsymbol{y}_0, \boldsymbol{y}_t) = q_{\tilde{\alpha}}(\boldsymbol{y}_t|\boldsymbol{y}_0) q_{\text{data_y}}(\boldsymbol{y}_0) = q_{\tilde{\alpha}}(\boldsymbol{y}_t|\boldsymbol{y}_0) q_{\text{data}} \circ \boldsymbol{f}_\ell^{-1}(\boldsymbol{y}_0)$ by definition. In step (ii), $q_{\tilde{\alpha}}(\boldsymbol{y}_t|\boldsymbol{y}_0)$ is expanded according to Eq. (4). In step (iii), since F depends on $\boldsymbol{y}_t$ only through $g_\ell(\boldsymbol{y}_t)$, we group terms by their image under g_ℓ. The domain of $\boldsymbol{y}_t^i$ partitions into vectors containing at least one $\mathtt{m}$ and the fully-unmasked value $f_\ell(x_0^i)$. Their respective probability masses are $1-(\tilde{\alpha}_t)^\ell = 1-\alpha_t$ and $(\tilde{\alpha}_t)^\ell = \alpha_t$, yielding the two-term expression in terms of $g_\ell(\boldsymbol{y}_t^i)$. In step (iv), the summation is performed on $\boldsymbol{x}_t \in \tilde{\mathcal{X}}^L$ (i.e., the codomain of $\boldsymbol{g}_\ell$). □

Lemma A.3. *Let $\tilde{\alpha}_t \triangleq \alpha_t^{1/\ell}$. The following inequality holds:*

$$\sup_p \mathbb{E}_{q_\alpha(\boldsymbol{x}_0, \boldsymbol{x}_t)} \big[\log p(\boldsymbol{x}_0 \mid \boldsymbol{x}_t) \big] \leq \sup_{p_\ell} \mathbb{E}_{q_{\tilde{\alpha}}(\boldsymbol{y}_0, \boldsymbol{y}_t)} \big[\log p_\ell(\boldsymbol{y}_0 \mid \boldsymbol{y}_t) \big], \quad \forall t \in [0,1]. \tag{A5}$$

The equality holds if and only if:

$$\mathbb{E}_{q_{\tilde{\alpha}}(\boldsymbol{y}_t)} \left[\mathbb{D}_{KL}(q_{\text{data_y}}(\boldsymbol{y}_0 \mid \boldsymbol{y}_t) \,\|\, q_{\text{data_y}}(\boldsymbol{y}_0 \mid \boldsymbol{g}_\ell(\boldsymbol{y}_t))) \right] = 0. \tag{A6}$$

Proof. The proof demonstrates that $\sup_{p_\ell} \mathbb{E}_{q_{\tilde{\alpha}}(\boldsymbol{y}_0, \boldsymbol{y}_t)} \big[\log p_\ell(\boldsymbol{y}_0|\boldsymbol{y}_t) \big] - \sup_p \mathbb{E}_{q_\alpha(\boldsymbol{x}_0, \boldsymbol{x}_t)} \big[\log p(\boldsymbol{x}_0|\boldsymbol{x}_t) \big] \geq 0$

by first expanding both terms. $\sup_{p_\ell} \mathbb{E}_{q_{\tilde{\alpha}}(\boldsymbol{y}_0,\boldsymbol{y}_t)}[\log p_\ell(\boldsymbol{y}_0|\boldsymbol{y}_t)]$ is expanded as follows:

$$
\begin{aligned}
&\sup_{p_\ell} \mathbb{E}_{q_{\tilde{\alpha}}(\boldsymbol{y}_0,\boldsymbol{y}_t)}[\log p_\ell(\boldsymbol{y}_0 \mid \boldsymbol{y}_t)] \\
&\quad \overset{(i)}{=} \sup_{p_\ell} \left(\mathbb{E}_{q_{\tilde{\alpha}}(\boldsymbol{y}_0,\boldsymbol{y}_t)}[\log p_\ell(\boldsymbol{y}_0 \mid \boldsymbol{y}_t)] + \mathbb{E}_{q_{\tilde{\alpha}}(\boldsymbol{y}_0,\boldsymbol{y}_t)}[\log q_{\text{data_y}}(\boldsymbol{y}_0 \mid \boldsymbol{y}_t)] - \mathbb{E}_{q_{\tilde{\alpha}}(\boldsymbol{y}_0,\boldsymbol{y}_t)}[\log q_{\text{data_y}}(\boldsymbol{y}_0 \mid \boldsymbol{y}_t)] \right) \\
&\quad = \sup_{p_\ell} \left(\mathbb{E}_{q_{\tilde{\alpha}}(\boldsymbol{y}_0,\boldsymbol{y}_t)}[\log q_{\text{data_y}}(\boldsymbol{y}_0 \mid \boldsymbol{y}_t) - \mathbb{E}_{q_{\tilde{\alpha}}(\boldsymbol{y}_0,\boldsymbol{y}_t)} \left[\log \frac{q_{\text{data_y}}(\boldsymbol{y}_0 \mid \boldsymbol{y}_t)}{p_\ell(\boldsymbol{y}_0 \mid \boldsymbol{y}_t)} \right] \right) \\
&\quad = \sup_{p_\ell} \left(\mathbb{E}_{q_{\tilde{\alpha}}(\boldsymbol{y}_0,\boldsymbol{y}_t)}[\log q_{\text{data_y}}(\boldsymbol{y}_0 \mid \boldsymbol{y}_t) - \mathbb{E}_{q_{\tilde{\alpha}}(\boldsymbol{y}_t)} \left[\mathbb{E}_{q_{\text{data_y}}(\boldsymbol{y}_0|\boldsymbol{y}_t)} \left[\log \frac{q_{\text{data_y}}(\boldsymbol{y}_0 \mid \boldsymbol{y}_t)}{p_\ell(\boldsymbol{y}_0 \mid \boldsymbol{y}_t)} \right] \right] \right) \\
&\quad = \mathbb{E}_{q_{\tilde{\alpha}}(\boldsymbol{y}_0,\boldsymbol{y}_t)}[\log q_{\text{data_y}}(\boldsymbol{y}_0 \mid \boldsymbol{y}_t)] - \inf_{p_\ell} \mathbb{E}_{q_{\tilde{\alpha}}(\boldsymbol{y}_t)} \Big[\mathbb{D}_{\text{KL}}\big(q_{\text{data_y}}(\boldsymbol{y}_0 \mid \boldsymbol{y}_t) \,\|\, p_\ell(\boldsymbol{y}_0 \mid \boldsymbol{y}_t)\big) \Big] \\
&\quad \overset{(ii)}{=} \mathbb{E}_{q_{\tilde{\alpha}}(\boldsymbol{y}_0,\boldsymbol{y}_t)}[\log q_{\text{data_y}}(\boldsymbol{y}_0 \mid \boldsymbol{y}_t)],
\end{aligned}
$$

where (i) adds and subtracts $\mathbb{E}_{q_{\tilde{\alpha}}(\boldsymbol{y}_0,\boldsymbol{y}_t)}[\log q_{\text{data_y}}(\boldsymbol{y}_0|\boldsymbol{y}_t)]$, and (ii) shows the optimum that the KL divergence becomes zero (i.e., $p_\ell(\boldsymbol{y}_0|\boldsymbol{y}_t) = q_{\text{data_y}}(\boldsymbol{y}_0|\boldsymbol{y}_t)$). On the other hand, $\sup_p \mathbb{E}_{q_\alpha(\boldsymbol{x}_0,\boldsymbol{x}_t)}[\log p(\boldsymbol{x}_0|\boldsymbol{x}_t)]$ can be simplified as an expected entropy term by following a similar derivation:

$$
\begin{aligned}
&\sup_{p} \mathbb{E}_{q_\alpha(\boldsymbol{x}_0,\boldsymbol{x}_t)}[\log p(\boldsymbol{x}_0 \mid \boldsymbol{x}_t)] \\
&\quad = \sup_{p} \left(\mathbb{E}_{q_\alpha(\boldsymbol{x}_0,\boldsymbol{x}_t)}[\log p(\boldsymbol{x}_0 \mid \boldsymbol{x}_t)] + \mathbb{E}_{q_\alpha(\boldsymbol{x}_0,\boldsymbol{x}_t)}[\log q_{\text{data}}(\boldsymbol{x}_0 \mid \boldsymbol{x}_t)] - \mathbb{E}_{q_\alpha(\boldsymbol{x}_0,\boldsymbol{x}_t)}[\log q_{\text{data}}(\boldsymbol{x}_0 \mid \boldsymbol{x}_t)] \right) \\
&\quad = \sup_{p} \left(\mathbb{E}_{q_\alpha(\boldsymbol{x}_0,\boldsymbol{x}_t)}[\log q_{\text{data}}(\boldsymbol{x}_0 \mid \boldsymbol{x}_t)] - \mathbb{E}_{q_\alpha(\boldsymbol{x}_0,\boldsymbol{x}_t)} \left[\log \frac{q_{\text{data}}(\boldsymbol{x}_0 \mid \boldsymbol{x}_t)}{p(\boldsymbol{x}_0 \mid \boldsymbol{x}_t)} \right] \right) \\
&\quad = \mathbb{E}_{q_\alpha(\boldsymbol{x}_0,\boldsymbol{x}_t)}[\log q_{\text{data}}(\boldsymbol{x}_0 \mid \boldsymbol{x}_t)] - \inf_{p} \mathbb{E}_{q_\alpha(\boldsymbol{x}_t)} \Big[\mathbb{D}_{\text{KL}}\big(q_{\text{data}}(\boldsymbol{x}_0 \mid \boldsymbol{x}_t) \,\|\, p(\boldsymbol{x}_0 \mid \boldsymbol{x}_t)\big) \Big] \\
&\quad = \mathbb{E}_{q_\alpha(\boldsymbol{x}_0,\boldsymbol{x}_t)}[\log q_{\text{data}}(\boldsymbol{x}_0 \mid \boldsymbol{x}_t)] \\
&\quad \overset{(i)}{=} \mathbb{E}_{q_{\tilde{\alpha}}(\boldsymbol{y}_0,\boldsymbol{y}_t)}[\log q_{\text{data}}(\boldsymbol{f}_\ell^{-1}(\boldsymbol{y}_0) \mid \boldsymbol{g}_\ell(\boldsymbol{y}_t))],
\end{aligned}
$$

where (i) rewrites the expectation via **Lemma A.2**. Based on the expanded terms, the proof concludes:

$$
\begin{aligned}
&\sup_{p_\ell} \mathbb{E}_{q_{\tilde{\alpha}}(\boldsymbol{y}_0,\boldsymbol{y}_t)}[\log p_\ell(\boldsymbol{y}_0 \mid \boldsymbol{y}_t)] - \sup_{p} \mathbb{E}_{q_\alpha(\boldsymbol{x}_0,\boldsymbol{x}_t)}[\log p(\boldsymbol{x}_0 \mid \boldsymbol{x}_t)] \\
&\qquad = \mathbb{E}_{q_{\tilde{\alpha}}(\boldsymbol{y}_0,\boldsymbol{y}_t)}[\log q_{\text{data_y}}(\boldsymbol{y}_0 \mid \boldsymbol{y}_t)] - \mathbb{E}_{q_{\tilde{\alpha}}(\boldsymbol{y}_0,\boldsymbol{y}_t)}[\log q_{\text{data}}(\boldsymbol{f}_\ell^{-1}(\boldsymbol{y}_0) \mid \boldsymbol{g}_\ell(\boldsymbol{y}_t))] \\
&\qquad = \mathbb{E}_{q_{\tilde{\alpha}}(\boldsymbol{y}_0,\boldsymbol{y}_t)} \left[\log \frac{q_{\text{data_y}}(\boldsymbol{y}_0 \mid \boldsymbol{y}_t)}{q_{\text{data}}(\boldsymbol{f}_\ell^{-1}(\boldsymbol{y}_0) \mid \boldsymbol{g}_\ell(\boldsymbol{y}_t))} \right] \\
&\qquad = \mathbb{E}_{q_{\tilde{\alpha}}(\boldsymbol{y}_t)} \left[\mathbb{D}_{\text{KL}}(q_{\text{data_y}}(\boldsymbol{y}_0 \mid \boldsymbol{y}_t) \,\|\, q_{\text{data}} \circ \boldsymbol{f}_\ell^{-1}(\boldsymbol{y}_0 \mid \boldsymbol{g}_\ell(\boldsymbol{y}_t)) \right] \\
&\qquad = \mathbb{E}_{q_{\tilde{\alpha}}(\boldsymbol{y}_t)} \left[\mathbb{D}_{\text{KL}}(q_{\text{data_y}}(\boldsymbol{y}_0 \mid \boldsymbol{y}_t) \,\|\, q_{\text{data_y}}(\boldsymbol{y}_0 \mid \boldsymbol{g}_\ell(\boldsymbol{y}_t)) \right] \geq 0.
\end{aligned}
$$

Rearranging the inequality yields $\sup_{p_\ell} \mathbb{E}_{q_{\tilde{\alpha}}(\boldsymbol{y}_0,\boldsymbol{y}_t)}\big[\log p_\ell(\boldsymbol{y}_0|\boldsymbol{y}_t)\big] \geq \sup_p \mathbb{E}_{q_\alpha(\boldsymbol{x}_0,\boldsymbol{x}_t)}\big[\log p(\boldsymbol{x}_0|\boldsymbol{x}_t)\big]$ and the equality holds when $\mathbb{E}_{q_{\tilde{\alpha}}(\boldsymbol{y}_t)} \left[\mathbb{D}_{\text{KL}}(q_{\text{data_y}}(\boldsymbol{y}_0 \mid \boldsymbol{y}_t) \,\|\, q_{\text{data_y}}(\boldsymbol{y}_0 \mid \boldsymbol{g}_\ell(\boldsymbol{y}_t)) \right] = 0$. □

Lemma A.4. *(cf. (Sahoo et al., 2024; Shi et al., 2024; Kingma et al., 2021)) Let $\alpha_t, \tilde{\alpha}_t : [0,1] \to [1,0]$ be two strictly decreasing, continuous bijections with $\alpha_0 = \tilde{\alpha}_0 = 1$ and $\alpha_1 = \tilde{\alpha}_1 = 0$. For any function $F : (0,1) \to \mathbb{R}$ integrable on $(0,1)$, define:*

$$
I(\alpha) \triangleq \int_0^1 \frac{\alpha_t'}{1-\alpha_t} F(\alpha_t) dt. \tag{A7}
$$

Then, we have:

$$
I(\alpha) = I(\tilde{\alpha}). \tag{A8}
$$

Proof. Let $s = \alpha_t$. Then, $ds = \alpha_t' dt$. As $t : 0 \to 1$, we have $s : 1 \to 0$. Hence,

$$I(\alpha) = \int_{t=0}^{1} \frac{\alpha_t'}{1-\alpha_t} F(\alpha_t) dt = \int_{s=1}^{0} \frac{1}{1-s} F(s) ds.$$

Let $s = \tilde{\alpha}_t$. Then, $ds = \tilde{\alpha}_t' dt$. As $s : 1 \to 0$, we have $t : 0 \to 1$. Hence,

$$\int_{s=1}^{0} \frac{1}{1-s} F(s) ds = \int_{t=0}^{1} \frac{\tilde{\alpha}_t'}{1-\tilde{\alpha}_t} F(\tilde{\alpha}_t) dt = I(\tilde{\alpha}).$$

Therefore, $I(\alpha) = I(\tilde{\alpha})$. □

Proposition A.5. *Given a scheduling function α_t, the following inequality holds:*

$$\inf_{p_\ell} \int_0^1 w(t) \mathbb{E}_{q_\alpha(\boldsymbol{y}_0, \boldsymbol{y}_t)} \big[-\log p_\ell(\boldsymbol{y}_0 \mid \boldsymbol{y}_t) \big] dt \leq \inf_{p} \int_0^1 w(t) \mathbb{E}_{q_\alpha(\boldsymbol{x}_0, \boldsymbol{x}_t)} \big[-\log p(\boldsymbol{x}_0 \mid \boldsymbol{x}_t) \big] dt. \tag{A9}$$

The equality holds if and only if:

$$\mathbb{E}_{q_{\tilde{\alpha}}(\boldsymbol{y}_t)} \left[\mathbb{D}_{KL}(q_{\text{data_y}}(\boldsymbol{y}_0 \mid \boldsymbol{y}_t) \,\|\, q_{\text{data_y}}(\boldsymbol{y}_0 \mid \boldsymbol{g}_\ell(\boldsymbol{y}_t))\right] = 0. \tag{A10}$$

Proof.

$$\begin{aligned}
\inf_{p_\ell} \int_0^1 w(t) \mathbb{E}_{q_\alpha(\boldsymbol{y}_0, \boldsymbol{y}_t)} \big[-\log p_\ell(\boldsymbol{y}_0 \mid \boldsymbol{y}_t) \big] dt &\overset{(i)}{=} \inf_{p_\ell} \int_0^1 \frac{\alpha_t'}{1-\alpha_t} \mathbb{E}_{q_\alpha(\boldsymbol{y}_0, \boldsymbol{y}_t)} \big[\log p_\ell(\boldsymbol{y}_0 \mid \boldsymbol{y}_t) \big] dt \\
&\overset{(ii)}{=} \int_0^1 \frac{\alpha_t'}{1-\alpha_t} \sup_{p_\ell} \mathbb{E}_{q_\alpha(\boldsymbol{y}_0, \boldsymbol{y}_t)} \big[\log p_\ell(\boldsymbol{y}_0 \mid \boldsymbol{y}_t) \big] dt \\
&\overset{(iii)}{=} \int_0^1 \frac{\tilde{\alpha}_t'}{1-\tilde{\alpha}_t} \sup_{p_\ell} \mathbb{E}_{q_{\tilde{\alpha}}(\boldsymbol{y}_0, \boldsymbol{y}_t)} \big[\log p_\ell(\boldsymbol{y}_0 \mid \boldsymbol{y}_t) \big] dt \\
&\overset{(iv)}{\leq} \int_0^1 \frac{\alpha_t'}{1-\alpha_t} \sup_{p} \mathbb{E}_{q_\alpha(\boldsymbol{x}_0, \boldsymbol{x}_t)} \big[\log p(\boldsymbol{x}_0 \mid \boldsymbol{x}_t) \big] dt \\
&= \inf_{p} \int_0^1 \frac{\alpha_t'}{1-\alpha_t} \mathbb{E}_{q_\alpha(\boldsymbol{x}_0, \boldsymbol{x}_t)} \big[\log p(\boldsymbol{x}_0 \mid \boldsymbol{x}_t) \big] dt \\
&= \inf_{p} \int_0^1 w(t) \mathbb{E}_{q_\alpha(\boldsymbol{x}_0, \boldsymbol{x}_t)} \big[-\log p(\boldsymbol{x}_0 \mid \boldsymbol{x}_t) \big] dt,
\end{aligned}$$

where (i) is because $w(t) = \frac{\alpha_t'}{\alpha_t - 1}$, (ii) is due to $\frac{\alpha_t'}{1-\alpha_t} < 0$, (iii) is due to **Lemma A.4**, and (iv) is due to **Lemma A.3**. The equality holds if and only if $\mathbb{E}_{q_{\tilde{\alpha}}(\boldsymbol{y}_t)} \left[\mathbb{D}_{\text{KL}}(q_{\text{data_y}}(\boldsymbol{y}_0 \mid \boldsymbol{y}_t) \,\|\, q_{\text{data_y}}(\boldsymbol{y}_0 \mid \boldsymbol{g}_\ell(\boldsymbol{y}_t))\right] = 0$ according to **Lemma A.3**. □

Lemmas A.2-A.4 and **Proposition A.5** show that if the scheduling $\tilde{\alpha}_t$ and the corresponding mapping $\boldsymbol{g}_\ell$ can be identified, the inequality in Eq. (A9) can be successfully established. We follow the same rationale to examine an additional extended theory: as the token granularity ℓ grows, the loss value becomes smaller. **Proposition A.6** generalizes **Proposition A.5** to compare the optimal losses of two MDM-Prime models with different token granularities.

Let ℓ_1 and ℓ_2 denote token granularities. We define $\boldsymbol{y}_0^{(\ell_1)} \triangleq \boldsymbol{f}_{\ell_1}(\boldsymbol{x}_0) \in \mathcal{Y}_1^{L \times \ell_1}$ and $\boldsymbol{y}_0^{(\ell_2)} \triangleq \boldsymbol{f}_{\ell_2}(\boldsymbol{x}_0) \in \mathcal{Y}_2^{L \times \ell_2}$, where the target spaces are defined as $\mathcal{Y}_k = \{0, \ldots, \lceil \sqrt[\ell_k]{V} \rceil - 1\}$ for $k \in \{1, 2\}$. For notational simplicity, we represent the distribution of $\boldsymbol{y}_0^{(\ell_2)}$ as $q_{\text{data_y}}(\boldsymbol{y}_0^{(\ell_2)})$ without explicitly defining a separate distribution symbol for $\boldsymbol{y}_0^{(\ell_1)}$.

We further specify the mappings $\boldsymbol{f}_{\tilde{\ell}}$ and $\boldsymbol{g}_{\tilde{\ell}}$ (Eq. (A2)) for transitioning between granularities with ratio $\tilde{\ell} = \ell_2 / \ell_1$. First, $\boldsymbol{f}_{\tilde{\ell}} : \mathcal{Y}_1^{L \times \ell_1} \to \mathcal{Y}_2^{L \times \ell_2}$ is an invertible function defined by the element-wise operation

$f_{\tilde{\ell}} : \mathcal{Y}_1 \to \mathcal{Y}_2^{\tilde{\ell}}$. We denote the mapping of a sequence as $\boldsymbol{y}_0^{(\ell_2)} = \boldsymbol{f}_{\tilde{\ell}}(\boldsymbol{y}_0^{(\ell_1)})$. With this definition, $\boldsymbol{y}_0^{(\ell_2)}$ can be expanded as follows:

$$\begin{aligned}\boldsymbol{y}_0^{(\ell_2)} = \boldsymbol{f}_{\tilde{\ell}}(\boldsymbol{y}_0^{(\ell_1)}) &= \Big[f_{\tilde{\ell}}(\boldsymbol{y}_0^{(\ell_1),i,1}), \cdots, f_{\tilde{\ell}}(\boldsymbol{y}_0^{(\ell_1),i,\ell_1})\Big] \\ &= [\boldsymbol{y}_0^{(\ell_2),i,1}, \cdots, \boldsymbol{y}_0^{(\ell_2),i,\ell_1}] \\ &= [(y_0^{(\ell_2),i,1}, \cdots, y_0^{(\ell_2),i,\tilde{\ell}}), \cdots, (y_0^{(\ell_2),i,\tilde{\ell}(\ell_1-1)}, \cdots, y_0^{(\ell_2),i,\ell_2})].\end{aligned}$$

Second, the function $\boldsymbol{g}_{\tilde{\ell}}$ is defined by the element-wise operation $g_{\tilde{\ell}}$. This function applies the inverse mapping $f_{\tilde{\ell}}^{-1}$ only if all corresponding sub-tokens are unmasked. Otherwise, it returns the masked token $\mathtt{m}$. The definition aligns with Eq. (A2) as follows:

$$g_{\frac{\ell_2}{\ell_1}}(\boldsymbol{y}_t^{(\ell_2),i,j}) = \begin{cases} f_{\frac{\ell_2}{\ell_1}}^{-1}(\boldsymbol{y}_t^{(\ell_2),i,j}), & \text{if } y_t^{i,\frac{\ell_2}{\ell_1}(j-1)+k} \neq \mathtt{m}, \forall k \in \{1, ..., \frac{\ell_2}{\ell_1}\}, \\ \mathtt{m}, & \text{otherwise.} \end{cases}$$

Proposition A.6. *Let ℓ_1, ℓ_2 be token granularities satisfying $1 < \ell_1 < \ell_2$ and $\frac{\ell_2}{\ell_1} \in \mathbb{N}$. Then, the optimal loss is monotone with respect to the token granularity:*

$$\inf_{p_{\ell_2}} \int_0^1 w(t)\mathbb{E}_{q_\alpha(\boldsymbol{y}_0^{(\ell_2)}, \boldsymbol{y}_t^{(\ell_2)})}\big[-\log p_{\ell_2}(\boldsymbol{y}_0^{(\ell_2)} \mid \boldsymbol{y}_t^{(\ell_2)})\big]dt \leq \inf_{p_{\ell_1}} \int_0^1 w(t)\mathbb{E}_{q_\alpha(\boldsymbol{y}_0^{(\ell_1)}, \boldsymbol{y}_t^{(\ell_1)})}\big[-\log p_{\ell_1}(\boldsymbol{y}_0^{(\ell_1)} \mid \boldsymbol{y}_t^{(\ell_1)})\big]dt. \tag{A11}$$

The equality holds if and only if:

$$\mathbb{E}_{q_{\tilde{\alpha}}(\boldsymbol{y}_t^{(\ell_2)})}\Big[\mathbb{D}_{KL}(q_{\text{data_y}}(\boldsymbol{y}_0^{(\ell_2)} \mid \boldsymbol{y}_t^{(\ell_2)}) \,\|\, q_{\text{data_y}}(\boldsymbol{y}_0^{(\ell_2)} \mid \boldsymbol{g}_{\frac{\ell_2}{\ell_1}}(\boldsymbol{y}_t^{(\ell_2)}))\Big] = 0. \tag{A12}$$

Proof. The proof first identifies the transformation between $\boldsymbol{y}_0^{(\ell_1)}$ and $\boldsymbol{y}_0^{(\ell_2)}$. Then, it establishes the inequality by following the proof presented in **Proposition A.5**.

Given that $\tilde{\ell} = \frac{\ell_2}{\ell_1}$, the invertible transformation $\boldsymbol{f}_{\ell_2}$ can be decomposed as $\boldsymbol{f}_{\ell_2} = \boldsymbol{f}_{\tilde{\ell}} \circ \boldsymbol{f}_{\ell_1}$, and $\boldsymbol{y}_0^{(\ell_2)}$ can be derived through $\boldsymbol{y}_0^{(\ell_1)}$ according to $\boldsymbol{y}_0^{(\ell_2)} = \boldsymbol{f}_{\tilde{\ell}}(\boldsymbol{y}_0^{(\ell_1)})$. In addition, let $\tilde{\alpha}_t = \alpha_t^{1/\tilde{\ell}}$. We can establish analogues to **Lemma A.2** and **Lemma A.3** using the same derivation steps:

$$\mathbb{E}_{q_\alpha(\boldsymbol{y}_0^{(\ell_1)}, \boldsymbol{y}_t^{(\ell_1)})}[F(\boldsymbol{y}_0^{(\ell_1)}, \boldsymbol{y}_t^{(\ell_1)})] = \mathbb{E}_{q_{\tilde{\alpha}}(\boldsymbol{y}_0^{(\ell_2)}, \boldsymbol{y}_t^{(\ell_2)})}[F(\boldsymbol{f}_{\tilde{\ell}}^{-1}(\boldsymbol{y}_0^{(\ell_2)}), \boldsymbol{g}_{\tilde{\ell}}(\boldsymbol{y}_t^{(\ell_2)}))].$$

$$\sup_{p_{\ell_1}} \mathbb{E}_{q_\alpha(\boldsymbol{y}_0^{(\ell_1)}, \boldsymbol{y}_t^{(\ell_1)})}\big[\log p_{\ell_1}(\boldsymbol{y}_0^{(\ell_1)} \mid \boldsymbol{y}_t^{(\ell_1)})\big] \leq \sup_{p_{\ell_2}} \mathbb{E}_{q_{\tilde{\alpha}}(\boldsymbol{y}_0^{(\ell_2)}, \boldsymbol{y}_t^{(\ell_2)})}\big[\log p_{\ell_2}(\boldsymbol{y}_0^{(\ell_2)} \mid \boldsymbol{y}_t^{(\ell_2)})\big], \quad \forall t \in [0, 1].$$

Finally, following the derivation of **Proposition A.5**, we have:

$$\begin{aligned}&\inf_{p_{\ell_2}} \int_0^1 w(t)\mathbb{E}_{q_\alpha(\boldsymbol{y}_0^{(\ell_2)}, \boldsymbol{y}_t^{(\ell_2)})}\big[-\log p_{\ell_2}(\boldsymbol{y}_0^{(\ell_2)} \mid \boldsymbol{y}_t^{(\ell_2)})\big]dt \\ &= \inf_{p_{\ell_2}} \int_0^1 \frac{\alpha_t'}{1-\alpha_t}\mathbb{E}_{q_\alpha(\boldsymbol{y}_0^{(\ell_2)}, \boldsymbol{y}_t^{(\ell_2)})}\big[\log p_{\ell_2}(\boldsymbol{y}_0^{(\ell_2)} \mid \boldsymbol{y}_t^{(\ell_2)})\big]dt \\ &= \int_0^1 \frac{\alpha_t'}{1-\alpha_t} \sup_{p_{\ell_2}} \mathbb{E}_{q_\alpha(\boldsymbol{y}_0^{(\ell_2)}, \boldsymbol{y}_t^{(\ell_2)})}\big[\log p_{\ell_2}(\boldsymbol{y}_0^{(\ell_2)} \mid \boldsymbol{y}_t^{(\ell_2)})\big]dt \\ &= \int_0^1 \frac{\tilde{\alpha}_t'}{1-\tilde{\alpha}_t} \sup_{p_{\ell_2}} \mathbb{E}_{q_{\tilde{\alpha}}(\boldsymbol{y}_0^{(\ell_2)}, \boldsymbol{y}_t^{(\ell_2)})}\big[\log p_{\ell_2}(\boldsymbol{y}_0^{(\ell_2)} \mid \boldsymbol{y}_t^{(\ell_2)})\big]dt \\ &\leq \int_0^1 \frac{\alpha_t'}{1-\alpha_t} \sup_{p_{\ell_1}} \mathbb{E}_{q_{\tilde{\alpha}}(\boldsymbol{y}_0^{(\ell_1)}, \boldsymbol{y}_t^{(\ell_1)})}\big[\log p_{\ell_1}(\boldsymbol{y}_0^{(\ell_1)} \mid \boldsymbol{y}_t^{(\ell_1)})\big]dt \\ &= \inf_{p_{\ell_1}} \int_0^1 \frac{\alpha_t'}{1-\alpha_t}\mathbb{E}_{q_\alpha(\boldsymbol{y}_0^{(\ell_1)}, \boldsymbol{y}_t^{(\ell_1)})}\big[\log p_{\ell_1}(\boldsymbol{y}_0^{(\ell_1)} \mid \boldsymbol{y}_t^{(\ell_1)})\big]dt \\ &= \inf_{p_{\ell_1}} \int_0^1 w(t)\mathbb{E}_{q_\alpha(\boldsymbol{y}_0^{(\ell_1)}, \boldsymbol{y}_t^{(\ell_1)})}\big[-\log p_{\ell_1}(\boldsymbol{y}_0^{(\ell_1)} \mid \boldsymbol{y}_t^{(\ell_1)})\big]dt.\end{aligned}$$

The equality holds if and only if $\mathbb{E}_{q_{\tilde{\alpha}}(\boldsymbol{y}_t^{(\ell_2)})}\left[\mathbb{D}_{\mathrm{KL}}(q_{\mathrm{data_y}}(\boldsymbol{y}_0^{(\ell_2)} \mid \boldsymbol{y}_t^{(\ell_2)}) \,\|\, q_{\mathrm{data_y}}(\boldsymbol{y}_0^{(\ell_2)} \mid \boldsymbol{g}_{\tilde{\ell}}(\boldsymbol{y}_t^{(\ell_2)}))\right] = 0$. □

A.1.2 Proofs of Propositions 3.1 and 3.2

As discussed in Section 3.1, the design of the subtokenizer $\boldsymbol{f}_\ell$ is critical for reducing the training loss. In this section, we provide the theoretical justifications for **Propositions 3.1** and **3.2**. We begin by deriving **Lemmas A.7-A.8**, which lead to the formulation of **Proposition 3.1**.

Lemma A.7. *The following equality holds:*

$$\mathbb{E}_{q_{\mathrm{data_y}}(\boldsymbol{y}_0)}[\log q_{\mathrm{data_y}}(\boldsymbol{y}_0)] = \mathbb{E}_{q_{\mathrm{data}}(\boldsymbol{x}_0)}[\log q_{\mathrm{data}}(\boldsymbol{x}_0)]. \tag{A13}$$

Proof.

$$\begin{aligned}
\mathbb{E}_{q_{\mathrm{data}}(\boldsymbol{x}_0)}[\log q_{\mathrm{data}}(\boldsymbol{x}_0)] &= \sum_{\boldsymbol{x}_0} q_{\mathrm{data}}(\boldsymbol{x}_0) \log q_{\mathrm{data}}(\boldsymbol{x}_0) \\
&= \sum_{\boldsymbol{y}_0} q_{\mathrm{data}} \circ \boldsymbol{f}_\ell^{-1}(\boldsymbol{y}_0) \log q_{\mathrm{data}} \circ \boldsymbol{f}_\ell^{-1}(\boldsymbol{y}_0) \\
&= \mathbb{E}_{q_{\mathrm{data_y}}(\boldsymbol{y}_0)}[\log q_{\mathrm{data_y}}(\boldsymbol{y}_0)].
\end{aligned}$$

□

Lemma A.8. *The following equality holds:*

$$\mathbb{E}_{q_\alpha(\boldsymbol{y}_0,\boldsymbol{y}_t)}[\log q_\alpha(\boldsymbol{y}_t \mid \boldsymbol{y}_0)] = L\ell\left((1-\alpha_t)\log(1-\alpha_t) + \alpha_t \log \alpha_t\right) \tag{A14}$$

Proof.

$$\begin{aligned}
\mathbb{E}_{q_\alpha(\boldsymbol{y}_0,\boldsymbol{y}_t)}[\log q_\alpha(\boldsymbol{y}_t \mid \boldsymbol{y}_0)] &= \mathbb{E}_{q_\alpha(\boldsymbol{y}_0,\boldsymbol{y}_t)}\left[\log\left(\prod_{i=1}^{L}\prod_{j=1}^{\ell} q_\alpha(y_t^{i,j} \mid y_0^{i,j})\right)\right] \\
&= \sum_{i=1}^{L}\sum_{j=1}^{\ell} \mathbb{E}_{q_\alpha(\boldsymbol{y}_0,\boldsymbol{y}_t)}\left[\log q_\alpha(y_t^{i,j} \mid y_0^{i,j})\right] \\
&= \sum_{i=1}^{L}\sum_{j=1}^{\ell} \mathbb{E}_{q_\alpha(\boldsymbol{y}_0,\boldsymbol{y}_t)}\left[\log\left((1-\alpha_t)\delta_{\mathtt{m}}(y_t^{i,j}) + \alpha_t \delta_{y_0^{i,j}}(y_t^{i,j})\right)\right] \\
&\overset{(i)}{=} \sum_{i=1}^{L}\sum_{j=1}^{\ell} \left((1-\alpha_t)\log(1-\alpha_t) + \alpha_t \log \alpha_t\right) \\
&= L\ell\left((1-\alpha_t)\log(1-\alpha_t) + \alpha_t \log \alpha_t\right),
\end{aligned}$$

where (i) is because $y_t^{i,j} = \mathtt{m}$ with probability $1-\alpha_t$ while $y_t^{i,j} = y_0^{i,j}$ with probability α_t (cf. **Proposition A2** in (Chao et al., 2025)). □

Proposition 3.1. The objective (Eq. (5)) can be decomposed into an $\boldsymbol{f}_\ell$-independent and an $\boldsymbol{f}_\ell$-dependent term as follows:

$$\inf_{\boldsymbol{f}_\ell}\inf_{p_\ell} \mathcal{L}^{(\ell)} = \int_0^1 w(t)\Big(\underbrace{\mathcal{H}(\boldsymbol{y}_0,\boldsymbol{y}_t)}_{\text{Independent on } \boldsymbol{f}_\ell.} - \sup_{\boldsymbol{f}_\ell} \underbrace{\mathcal{H}(\boldsymbol{y}_t)}_{\text{Dependent on } \boldsymbol{f}_\ell.}\Big)dt, \tag{A15}$$

where $\mathcal{H}(\boldsymbol{y}_t) \triangleq \mathbb{E}_{q_\alpha(\boldsymbol{y}_t)}[-\log q_\alpha(\boldsymbol{y}_t)]$ and $\mathcal{H}(\boldsymbol{y}_0,\boldsymbol{y}_t) \triangleq \mathbb{E}_{q_\alpha(\boldsymbol{y}_0,\boldsymbol{y}_t)}[-\log q_\alpha(\boldsymbol{y}_0,\boldsymbol{y}_t)]$ represent the entropy of $\boldsymbol{y}_t$ and the joint entropy of $\boldsymbol{y}_0, \boldsymbol{y}_t$, respectively.

Proof.

$$
\begin{aligned}
&\inf_{\boldsymbol{f}_\ell} \inf_{p_\ell} \mathcal{L}^{(\ell)} \\
&= \inf_{\boldsymbol{f}_\ell} \inf_{p_\ell} \int_0^1 w(t) \mathbb{E}_{q_\alpha(\boldsymbol{y}_0, \boldsymbol{y}_t)} \big[-\log p_\ell(\boldsymbol{y}_0 \mid \boldsymbol{y}_t) \big] dt \\
&\overset{(i)}{=} \inf_{\boldsymbol{f}_\ell} \int_0^1 w(t) \mathbb{E}_{q_\alpha(\boldsymbol{y}_0, \boldsymbol{y}_t)} \big[-\log q_{\text{data_y}}(\boldsymbol{y}_0 \mid \boldsymbol{y}_t) \big] dt \\
&= \int_0^1 -w(t) \sup_{\boldsymbol{f}_\ell} \mathbb{E}_{q_\alpha(\boldsymbol{y}_0, \boldsymbol{y}_t)} \big[\log q_{\text{data_y}}(\boldsymbol{y}_0 \mid \boldsymbol{y}_t) \big] dt \\
&= \int_0^1 -w(t) \sup_{\boldsymbol{f}_\ell} \left(\mathbb{E}_{q_\alpha(\boldsymbol{y}_0, \boldsymbol{y}_t)} \big[\log q_{\text{data_y}}(\boldsymbol{y}_0) + \log q_\alpha(\boldsymbol{y}_t \mid \boldsymbol{y}_0) - \log q_\alpha(\boldsymbol{y}_t) \big] \right) dt \\
&= \int_0^1 -w(t) \left(\underbrace{\mathbb{E}_{q_{\text{data_y}}(\boldsymbol{y}_0)} [\log q_{\text{data_y}}(\boldsymbol{y}_0)]}_{(1)\text{ Independent on } \boldsymbol{f}_\ell.} + \underbrace{\mathbb{E}_{q_\alpha(\boldsymbol{y}_0, \boldsymbol{y}_t)} [\log q_\alpha(\boldsymbol{y}_t \mid \boldsymbol{y}_0)]}_{(2)\text{ Independent on } \boldsymbol{f}_\ell.} + \sup_{\boldsymbol{f}_\ell} \underbrace{\mathbb{E}_{q_\alpha(\boldsymbol{y}_t)} [-\log q_\alpha(\boldsymbol{y}_t)]}_{(3)\text{ dependent on } \boldsymbol{f}_\ell.} \right) dt \\
&= \int_0^1 -w(t) \left(\mathbb{E}_{q_\alpha(\boldsymbol{y}_0, \boldsymbol{y}_t)} [\log q_\alpha(\boldsymbol{y}_0, \boldsymbol{y}_t)] + \sup_{\boldsymbol{f}_\ell} \mathbb{E}_{q_\alpha(\boldsymbol{y}_t)} [-\log q_\alpha(\boldsymbol{y}_t)] \right) dt, \\
&= \int_0^1 w(t) \Big(\underbrace{\mathcal{H}(\boldsymbol{y}_0, \boldsymbol{y}_t)}_{\text{Independent on } \boldsymbol{f}_\ell.} - \sup_{\boldsymbol{f}_\ell} \underbrace{\mathcal{H}(\boldsymbol{y}_t)}_{\text{Dependent on } \boldsymbol{f}_\ell.} \Big) dt,
\end{aligned}
$$

where (i) is derived in **Lemma A.3**. Term (1) corresponds to the negative data entropy, $-\mathcal{H}(\boldsymbol{x}_0) \triangleq \mathbb{E}_{q_{\text{data}}(\boldsymbol{x}_0)}[\log q_{\text{data}}(\boldsymbol{x}_0)]$, which is independent of $\boldsymbol{f}_\ell$ according to **Lemma A.7**. Term (2) represents the negative conditional entropy, $-\mathcal{H}(\boldsymbol{y}_t|\boldsymbol{y}_0)$, which equals to the constant $-L\ell[(1-\alpha_t)\log(1-\alpha_t) + \alpha_t \log \alpha_t]$ by **Lemma A.8**. Finally, term (3) is $\mathcal{H}(\boldsymbol{y}_t)$ and is the only component dependent on $\boldsymbol{f}_\ell$, making it the primary target of the optimization objective. □

Next, we show that the entropy of $\boldsymbol{y}_t$ is bounded. We first present **Lemma A.9** from (Austin et al., 2021; Chao et al., 2025). Then, we provide the proof of **Proposition 3.2**.

Lemma A.9. *(cf. Proposition A2 in (Chao et al., 2025))*

$$
\mathcal{H}(y_t^{i,j}) = \alpha_t \mathcal{H}(y_0^{i,j}) - (1-\alpha_t)\log(1-\alpha_t) - \alpha_t \log \alpha_t. \tag{A16}
$$

Proof. Please see Eq. (A12) in (Chao et al., 2025) and Section A7 of (Austin et al., 2021). □

Proposition 3.2. The entropy of $\boldsymbol{y}_t$ is bounded:

$$
\mathcal{H}(\boldsymbol{y}_t) \le L\ell(h(\alpha_t) + \alpha_t \log b), \tag{A17}
$$

where $h(\alpha_t) = -(1-\alpha_t)\log(1-\alpha_t) - \alpha_t \log \alpha_t$. The equality holds if and only if each unmasked $y_t^{i,j}$ is uniformly distributed on $\mathcal{Y}$.

Proof.

$$
\begin{aligned}
\mathcal{H}(\boldsymbol{y}_t) &\overset{(i)}{\le} \sum_{i=1}^{L} \sum_{j=1}^{\ell} \mathcal{H}(y_t^{i,j}) \\
&\overset{(ii)}{=} \sum_{i=1}^{L} \sum_{j=1}^{\ell} \Big(\alpha_t \mathcal{H}(y_0^{i,j}) - (1-\alpha_t)\log(1-\alpha_t) - \alpha_t \log \alpha_t \Big) \\
&\overset{(iii)}{\le} \sum_{i=1}^{L} \sum_{j=1}^{\ell} \left(\alpha_t \log b - (1-\alpha_t)\log(1-\alpha_t) - \alpha_t \log \alpha_t \right) \\
&= L\ell \left(\alpha_t \log b - (1-\alpha_t)\log(1-\alpha_t) - \alpha_t \log \alpha_t \right) \\
&= L\ell \left(\alpha_t \log b + h(\alpha_t) \right),
\end{aligned}
$$

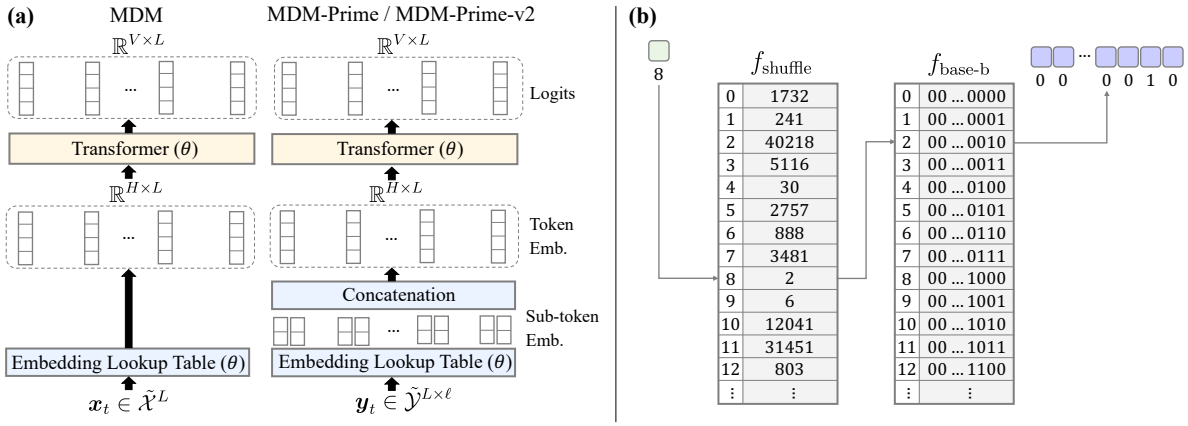


Figure A2: (a) Comparison of MDM, MDM-Prime, and MDM-Prime-v2 architectures. V denotes the vocabulary size, L is the sequence length, ℓ is the token granularity, and H is the Transformer hidden embedding dimension. (b) Implementation of the subtokenizer $f_\ell = f_{\text{base-}b} \circ f_{\text{shuffle}}$. The process utilizes two lookup tables to map indices efficiently; the inverse operation f_ℓ^{-1} is performed by reversing the lookup sequence.

where (i) is due to the subadditivity of entropy, (ii) is due to **Lemma A.9**, and (iii) is because $|\mathcal{Y}| = b$. The equality holds when each sub-token is independent and uniformly distributed. □

A.1.3 Estimating the Variational Bound of MDM-Prime

The training objective of MDM-Prime (Eq. (5)) requires an additional marginalization step to constitute a valid variational upper bound on the negative log likelihood (NLL). As the bound derivation assumes mutually independent sub-token predictions, the joint parameterization $p_\theta(\boldsymbol{y}_0^i|\boldsymbol{y}_t)$ in (Chao et al., 2025), which introduces dependencies among sub-tokens within each token group, must first be marginalized. Specifically, let $\mathcal{M}_j \triangleq \{y_0^{i,k} \mid k \neq j,\ y_0^{i,k} \in \mathcal{Y}\}$; the marginalized distribution is:

$$p_\theta(y_0^{i,j}|\boldsymbol{y}_t) \triangleq \sum_{\boldsymbol{y}_0^{i,\neq j}\in\mathcal{M}_j} p_\theta(y_0^{i,j}, \boldsymbol{y}_0^{i,\neq j}|\boldsymbol{y}_t). \tag{A18}$$

Substituting the fully factorized form $p_\theta(\boldsymbol{y}_0|\boldsymbol{y}_t) = \prod_{i=1}^{L}\prod_{j=1}^{\ell} p_\theta(y_0^{i,j}|\boldsymbol{y}_t)$ into Eq. (5) then yields a valid variational upper bound on the NLL. This marginalization can be implemented efficiently via filtering on the model's logit outputs.

A.2 Technical Specifications

This section provides further implementation details for the proposed framework. Section A.2.1 describes the architectures of MDM, MDM-Prime, and MDM-Prime-v2, while Section A.2.2 details a practical implementation of the subtokenizer $\boldsymbol{f}_\ell$.

A.2.1 Model Architecture of MDM-Prime and MDM-Prime-v2

Fig. A2 (a) compares the architectures of MDM, MDM-Prime, and MDM-Prime-v2. MDM-Prime and MDM-Prime-v2 share exactly the same architecture. As detailed in Section 2.2, transitioning from MDM to MDM-Prime (or MDM-Prime-v2) requires only a simple modification of the embedding table (Chao et al., 2025). In this setup, sub-token embeddings are aggregated into token embeddings, and the neural network (i.e., Transformer (θ) in the figure) subsequently processes at the token level. The decoder maintains its original dimensionality to effectively model intra-token dependencies. Specifically, let H represent the hidden layer embedding size of the transformer. The transformers in both architectures accept inputs of size $\mathbb{R}^{H\times L}$ and outputs logits with $\mathbb{R}^{C\times L}$. This design maintains the core model architecture and ensures that the computational cost (FLOPs) per forward pass remains unchanged.

A.2.2 Subtokenizer Implementation

Fig. A2 (b) depicts how the element-wise operation $f_\ell = f_{\text{base-}b} \circ f_{\text{shuffle}}$ of subtokenizer $\boldsymbol{f}_\ell$ is implemented in practice. This element-wise operation relies on lookup tables: to transform a token x_0^i into a sub-token sequence $\boldsymbol{y}_0^i$, the function sequentially retrieves the shuffled index and its corresponding base-b

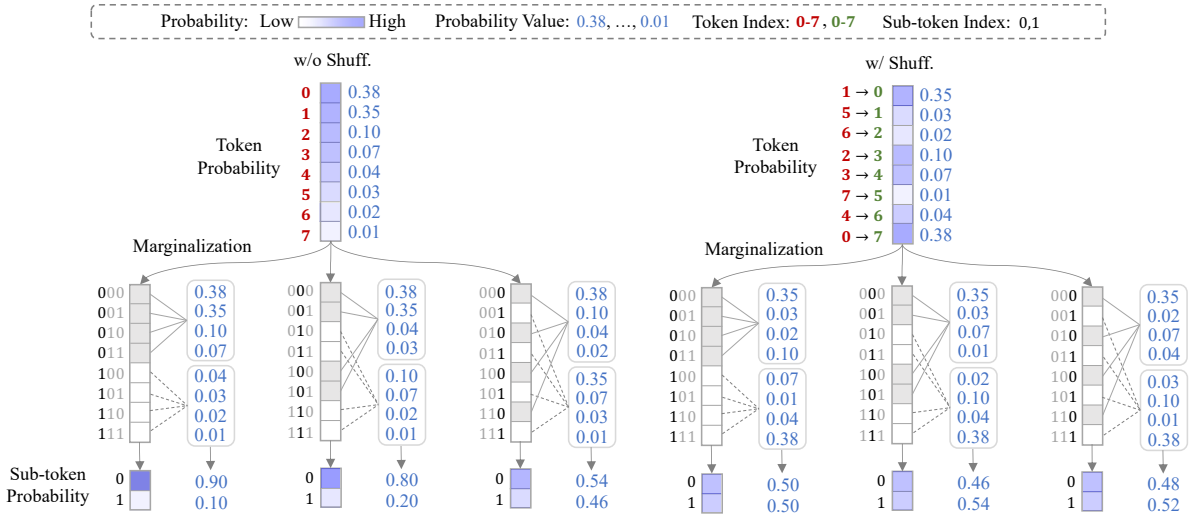


Figure A3: A detailed numerical example of Fig. 5 (a). In this plot, $V = 8$, $\ell = 3$ and $b = 2$. Color intensity represents probability values, where darker blue denotes higher probability. Marginalization is performed by summing entries corresponding to sub-token indices 0 (gray) and 1 (white), respectively. Sub-tokens encoded from standard indices with a structured distribution exhibit low entropy, while the shuffle operation results in sub-tokens with higher entropy.

representation. Similarly, the inverse operation f_ℓ^{-1} is executed via a reverse lookup. Both directions can be efficiently implemented using dictionaries in Python.

Please note that the $\boldsymbol{f}_\ell$ is static over training, and the lookup tables for index shuffling and base-b encoding are also fixed prior to training. In Section A.4, we show that the performance of MDM-Prime-v2 is robust to the choice of the random seed used for the index shuffling operation.

We do not recommend parameterizing and iteratively updating the subtokenizer during training for two reasons. First, our analysis in Section 3.1 suggests that a non-learnable index shuffling operation is sufficiently effective and the sub-token entropy is near the theoretical maximum. Introducing an additional optimization phase for the subtokenizer is likely to yield only marginal gains while significantly increasing computational overhead. Second, updating $\boldsymbol{f}_\ell$ during training would induce a continuous shift in both the target distribution $q_{\text{data_y}}(\boldsymbol{y}_0)$ and the latent distribution $q_\alpha(\boldsymbol{y}_t)$. This non-stationarity in the learning objective can result in severe training instability.

A.3 Experimental Configurations

This section provides supplemental experimental details and hyperparameter configurations. The settings for the experiments described in Sections 4.1 and 4.2 are detailed in Sections A.3.1 and A.3.2, respectively.

A.3.1 Experiments in Section 4.1

Dataset and Preprocessing Method. The training is performed on the C4 dataset (Raffel et al., 2020). Text data is tokenized using the GPT-2 tokenizer (Radford et al., 2019). The maximum sequence length L and the vocabulary size V are defined as 2,048 and 50,257, respectively. The documents are concatenated with the <eod> token as a separator and then wrapped into samples of length 2,048.

Training Configuration. We employ a Transformer architecture (Vaswani et al., 2017) incorporating Rotary Positional Embeddings (RoPE) (Su et al., 2023), SwiGLU activation (Shazeer, 2020), RMSNorm (Zhang and Sennrich, 2019), and QK-normalization (Dehghani et al., 2023). Model sizes (N) range from 14M to 3.4B parameters; detailed specifications for depth, hidden dimensions, and attention heads are summarized in Table A2. We utilize a linear scheduling function $\alpha_t = 1 - t$ for the diffusion process. Optimization is performed using AdamW (Loshchilov and Hutter, 2019) with $\beta_1 = 0.9$ and $\beta_2 = 0.999$. The learning rate is set to 2×10^{-4} with a cosine decay to a minimum of 2×10^{-5}. We use a batch size of 256 for $N \leq 1.5$B and 512 for larger models. The coding framework is based on Megatron-LM (Shoeybi et al., 2019). For hardware, models with $N \leq 400$M are trained on eight NVIDIA L40 GPUs (48 GB), while larger models ($N > 400$M) are trained on eight NVIDIA H100 GPUs (80 GB).

A.3.2 Experiments in Section 4.2

Dataset and Preprocessing Method. Following TinyLLaMA (Zhang et al., 2024) and SMDM (Nie et al., 2025a), training is conducted on the SlimPajama dataset (Soboleva et al., 2023), with text tokenized using the LLaMA tokenizer. The maximum sequence length L is set to 2,048, and the vocabulary size V is 32,000. Additionally, following the approach in (Nie et al., 2025a), we randomize the sequence length for 1% of the training data. This is implemented by sampling a length L' from a uniform distribution $\mathcal{U}(1, 2048)$ for each affected sequence.

Evaluation Method. We utilize the `lm-evaluation-harness` package (Gao et al., 2024) to measure zero-shot accuracy on eight commonsense reasoning benchmarks: SciQ (Welbl et al., 2017), SocialIQA (Sap et al., 2019), MCTaco (Zhou et al., 2019), TruthfulQA (Lin et al., 2022), BoolQ (Clark et al., 2019a), ANLI (Nie et al., 2020), and ARC-e (Easy) (Clark et al., 2018). Detailed task descriptions are provided in Table A5. For prediction, we concatenate each answer candidate with the question prompt and select the option that yields the highest likelihood under the diffusion model.

To evaluate sample quality, we use generative perplexity (Gen PPL) and entropy as complementary metrics, measuring sample fidelity and diversity respectively. Since Gen PPL can be manipulated through improper temperature selection at the expense of entropy (Zheng et al., 2025), we tune the sampling temperatures and seeds of SMDM and MDM-Prime-v2 to yield comparable entropy values, ensuring a fair Gen PPL comparison. Gen PPL is computed using TinyLLaMA (Zhang et al., 2024), trained on 3T tokens.

Training and Implementation Details. We utilize a Transformer architecture (Vaswani et al., 2017) enhanced with RoPE (Su et al., 2023), SwiGLU activation (Shazeer, 2020), RMSNorm (Zhang and Sennrich, 2019), and Grouped Query Attention (GQA) (Ainslie et al., 2023). The model contains 1.028B non-embedding parameters and is trained on 540B tokens. This configuration is identical to our MDM and ARM baselines (i.e., SMDM (Nie et al., 2025a) and TinyLLaMA (Zhang et al., 2024)). We adopt the convention of these works and refer to the model as 1.1B. For the diffusion process, we utilize a linear schedule defined by $\alpha_t = 1 - t$. We set the grouping factor u to 1 if $t < 0.5$ and ℓ if $t \geq 0.5$ (see Section A.4.2). Training is performed using the AdamW optimizer (Loshchilov and Hutter, 2019) with hyperparameters $\beta_1 = 0.9$ and $\beta_2 = 0.999$. We set the maximum learning rate to 2×10^{-4}, which follows a cosine decay schedule to a minimum of 2×10^{-5}. We use a batch size of 256. Our implementation is built upon the lit-gpt framework (Lightning-AI, 2023), leveraging FlashAttention (Dao et al., 2022) and xFormers (Lefaudeux et al., 2022) for enhanced efficiency. The model was trained on sixteen NVIDIA H100 GPUs (80 GB) with a multi-node configuration.

A.4 Supplementary Experiments

In this section, we present additional experimental results. Section A.4.1 examines design choices that yield performance invariance. Section A.4.2 provides further analysis of the log-likelihood distribution

N	H	B	Heads	N	H	B	Heads
14M	448	6	7	252M	1,024	20	16
25M	512	8	8	354M	1,280	18	10
36M	576	9	9	413M	1,280	21	10
49M	640	10	10	571M	1,408	24	11
64M	640	13	10	771M	1,792	20	14
79M	640	16	10	1,107M	2,048	22	16
106M	768	15	12	1,529M	2,304	24	18
154M	896	16	14	2,359M	2,560	30	20
201M	1,024	16	16	3,426M	2,816	36	22

Table A2: Model architecture configurations. N represents the number of non-embedding parameters, H denotes the hidden dimension size, B is the number of Transformer blocks, and 'Heads' refer to the number of attention heads.

	ARM	MDM	MDM-Prime-v2
α	$0.309^{+0.030}_{-0.017}$	$0.345^{+0.016}_{-0.038}$	$0.379^{+0.023}_{-0.024}$
β	$0.334^{+0.013}_{-0.036}$	$0.342^{+0.027}_{-0.017}$	$0.380^{+0.014}_{-0.018}$
A	217^{+140}_{-49}	355^{+98}_{-158}	502^{+210}_{-156}
B	1096^{+304}_{-531}	1969^{+1430}_{-551}	4324^{+1463}_{-1348}
E	$2.029^{+0.042}_{-0.077}$	$2.420^{+0.049}_{-0.052}$	$1.578^{+0.037}_{-0.056}$

Table A3: Regression parameters for ARM, MDM, and MDM-Prime-v2 fitted according to Chinchilla's scaling laws (Hoffmann et al., 2022). E denotes the irreducible loss (entropy of the natural text distribution). The terms $A \times N^{-\alpha}$ and $B \times D^{-\beta}$ account for errors arising from finite model size and limited training tokens, respectively.

	SocialIQA	OBQA	McTaco	ARC-e	TruthfulQA	BoolQ	SciQ	ANLI
Steps	1,024	512	512	512	256	256	256	128

Table A4: Number of steps used for zero-shot evaluation of MDM-Prime-v2. SMDM is fixed at 1,024 steps; MDM-Prime-v2 uses no more than 1,024 steps.

Benchmark	**Description**
SciQ (Welbl et al., 2017)	A collection of crowdsourced multiple-choice science exam questions covering Biology, Chemistry, and Physics.
SocialIQA (Sap et al., 2019)	A large-scale benchmark testing understanding of social interactions, emotional reactions, and motivations.
McTaco (Zhou et al., 2019)	A dataset focusing on temporal commonsense reasoning, requiring answers about duration, frequency, and event ordering.
TruthfulQA (Lin et al., 2022)	A benchmark measuring whether models mimic human falsehoods, focusing on misconceptions and superstitions.
BoolQ (Clark et al., 2019a)	Naturally occurring yes/no questions paired with Wikipedia paragraphs containing the answer.
ANLI (Nie et al., 2020)	Adversarial natural language inference dataset collected via a human-and-model-in-the-loop process.
ARC-e (Clark et al., 2018)	The easy subset of the AI2 Reasoning Challenge, containing grade-school science questions.
OBQA (Mihaylov et al., 2018)	A QA dataset modeled after open book exams, containing elementary science questions that require reasoning with common knowledge.

Table A5: An overview of the commonsense reasoning benchmarks used for evaluation.

across time. Section A.4.3 details the performance of ARM trained under the sub-tokenization scheme. Section A.4.4 provides detailed ablation analysis on the proposed two techniques. Section A.4.5 presents additional scaling analysis. Finally, Section A.4.6 presents preliminary results of MDM-Prime-v2 at 7B-parameter scale.

A.4.1 Performance-Invariant Designs

In Section 4.1, we analyze two performance-sensitive hyperparameters: the number of non-embedding parameters (N) and the number of training tokens (D). This section presents a number of performance-invariant designs. The results are shown in Fig. A4.

(a) Depth and Width of the Model. As previously established in (Kaplan et al., 2020), the specific shape of a Transformer architecture has a marginal influence on its final performance. Our observations align with their findings. To study this property, we vary the hidden dimension (H) and the number of Transformer blocks (B) while maintaining a fixed model size. Specifically, we follow the approximation $N \approx 12BH^2$, increasing the number of blocks quadratically as the hidden dimension is reduced. We compare three distinct (B, H) configurations: $(15, 768)$, $(27, 576)$, and $(4, 1536)$, with their corresponding training curves illustrated in Fig. A4. For a clear comparison, we adopt the original MDM-Prime ($\ell = 16$) and evaluate the impact of varying B and H against a reference curve for MDM-Prime-v2 ($\ell = 16$) with $(B, H) = (15, 768)$ (shown in gray). We observe that the performance is not sensitive to the model shape.

(b) Sub-token Embeddings. To adapt an MDM architecture to that of MDM-Prime, input sub-token embeddings are merged into token embeddings prior to the Transformer layers; this prevents an increase in model size (Chao et al., 2025). We observe that performance remains invariant across different merging methods. In this experiment, we evaluate three designs: (1) *Concat-Shared*: the original concatenation operation proposed in (Chao et al., 2025); (2) *Concat*: a concatenation operation using non-shared embedding tables for sub-tokens at different positions; and (3) *Sum*: a summation operation using the non-shared design from (2). The results are presented in Fig. A4 (b). Similarly, we include MDM-Prime-v2 ($\ell = 16$) with *Concat-Shared* as a reference curve (shown in gray). We observe that the performance is invariant to the choice of the embedding table designs.

(c) Random Seed of the Index Shuffling Operation. In MDM-Prime-v2, we introduce an index shuffling operation within the subtokenizer $\boldsymbol{f}_\ell$. To evaluate the robustness of this design to random seed

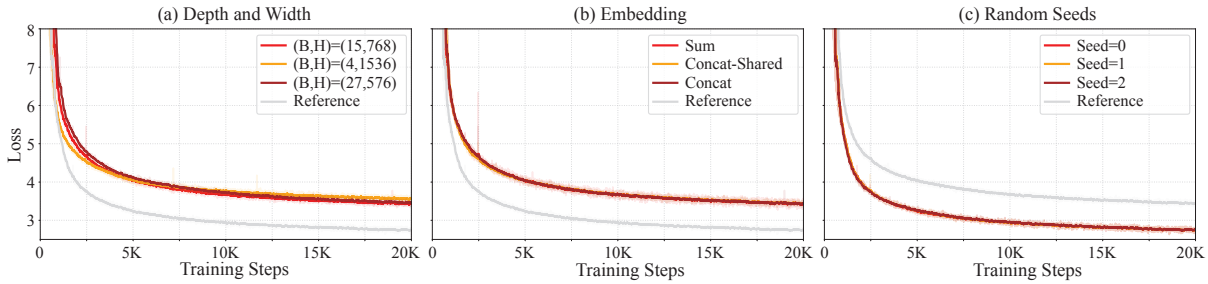


Figure A4: Analysis of performance invariance in MDM-Prime and MDM-Prime-v2. The plots show robustness against variations in: (a) model architecture depth and width; (b) embedding table designs; and (c) random seeds for the index shuffling operation.

initialization, we independently train three models using different seeds in the random index shuffling operation. The performance of these runs is presented in Fig. A4 (c), alongside MDM-Prime ($\ell = 16$) as a reference baseline (gray line). We observe that the performance is robust to the choice of random seeds.

A.4.2 Posterior Log-likelihood Distribution Across Time

In Section 4, we established that MDM-Prime-v2 exhibits improved posterior log-likelihood modeling (i.e., the loss) compared to MDM. To investigate the temporal distribution of this improvement, we computed the log-likelihood at each timestep t, specifically $\mathbb{E}_{q_\alpha(\boldsymbol{x}_0,\boldsymbol{x}_t)}[\log p(\boldsymbol{x}_0|\boldsymbol{x}_t)]$ for MDM and $\mathbb{E}_{q_\alpha(\boldsymbol{y}_0,\boldsymbol{y}_t)}[\log p_\ell(\boldsymbol{y}_0|\boldsymbol{y}_t)]$ for MDM-Prime, omitting the common weighting term $\frac{\alpha'}{1-\alpha_t}$. These results are presented in Fig. A5, where (a) and (b) show the settings without and with Technique 1, respectively.

We observe that MDM-Prime is not consistently superior across all timesteps. There is a distinct crossover point at approximately $t \approx 0.6$ in plot (a) and $t \approx 0.75$ in plot (b). These results indicate that the original MDM performs better when the sequence contains a high ratio of masked tokens (large t). On the other hand, MDM-Prime outperforms MDM when sufficient unmasked tokens are present (small t). We hypothesize that coarse, token-level prediction is more robust when context is scarce, whereas fine-grained denoising becomes advantageous when sufficient context is available. Motivated by this observation, we introduce a minor modification to the sub-token sampling process: we apply token-wise joint masks for $t = 0.5$ in the experiments of Section 4.2. Let u denote the sub-token group size, which equals to 1 if $t < 0.5$ and ℓ if $t \geq 0.5$. $\boldsymbol{y}_t^{i,k} = [y_t^{i,k*u}, \cdots, y_t^{i,(k+1)*u}]$. The modified forward kernel groups sub-tokens is formulated as:

$$q_\alpha(\boldsymbol{y}_t|\boldsymbol{y}_0) \triangleq \prod_{i=1}^{L} \prod_{k=1}^{\ell/u} q_\alpha(\boldsymbol{y}_t^{i,k}|\boldsymbol{y}_0) \tag{A19}$$

This modification incurs no additional FLOPs during training, integrates seamlessly into the existing pipeline, and yields consistent improvements in generation quality.

A second finding is that the index shuffling operation consistently improves log-likelihood across all timesteps, as evidenced by the upward shift in plot (b) for all token granularities ℓ. This shift moves the crossover point from $t \approx 0.65$ to $t \approx 0.75$. These results demonstrate the effectiveness of the proposed Technique 1.

A.4.3 Autoregression on Sub-tokens

This section presents the results for ARM trained with sub-tokenization using $\ell \in \{1, 4, 8\}$. Let y_0^k denote $y_0^{i,j}$ where $i = \lceil k/\ell \rceil$ and $j = k\%\ell$. The negative log-likelihood of these models is expressed as follows:

$$\mathbb{E}_{q_{\text{data_y}}(\boldsymbol{y}_0)} \left[-\sum_{k=1}^{L\ell} \log p(y_0^k|\boldsymbol{y}_0^{<k}) \right], \tag{A20}$$

where $\boldsymbol{y}_0^{<k} = [y_0^1, \cdots, y_0^{k-1}]$. Each model contains 92M non-embedding parameters and was trained on 52.4B tokens from the OWT dataset (Gokaslan et al., 2019). Table A6 summarizes the results. Contrary

	N	D	Loss (↓)
ℓ=1	92M	52.4B	2.925
ℓ=4	92M	52.4B	3.504
ℓ=8	92M	52.4B	3.962

Table A6: Loss of ARM trained with ℓ=1, 4, and 8 on OWT. N and D represents the model size and the number of training tokens, respectively. The symbol ↓ represents that lower values correspond to better performance.

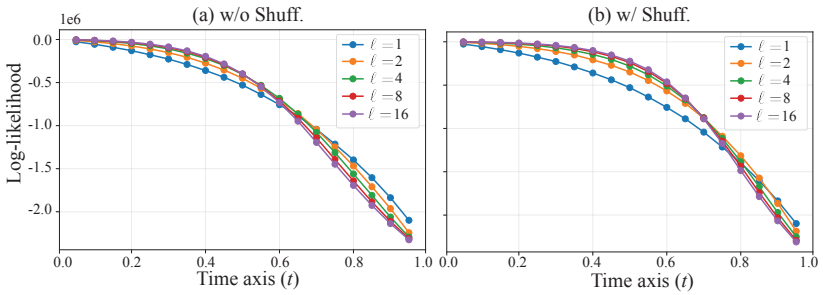


Figure A5: Log-likelihood of MDM (ℓ =1) and MDM-Prime (ℓ >1) across time. Subplots (a) and (b) show the results without and with the index shuffling operation, respectively. The maximum value for each curve is 0.

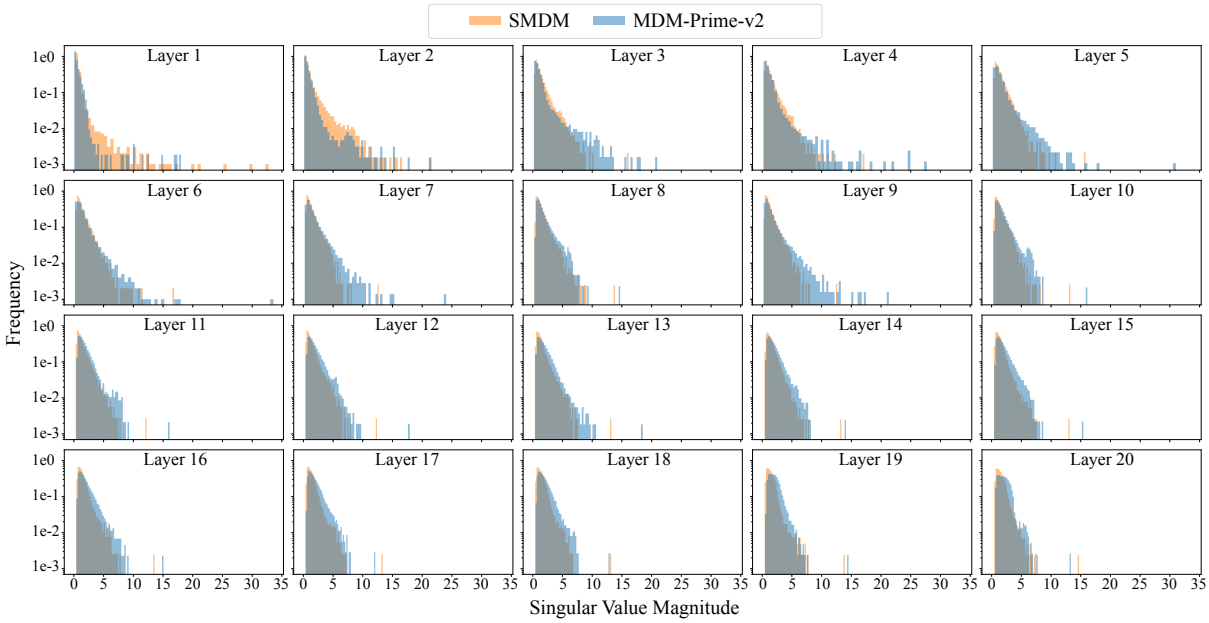


Figure A6: Singular value distributions of the query-key-value projection matrices of SMDM (1.1B) (Nie et al., 2025a) and MDM-Prime-v2 (1.1B). Both SMDM and MDM-Prime-v2 adopt the same architecture containing 20 layers. The y-axis represents frequency on a log scale. Compared to MDM, MDM-Prime-v2 exhibits a heavier-tailed distribution, particularly in intermediate layers (i.e., Layers 3–12).

to MDM, ARM exhibits performance degradation at higher ℓ values. We hypothesize this is because the predictive performance of ARM naturally declines as sequence length increases (Rashkin et al., 2020; Pillutla et al., 2021). Furthermore, unlike MDM-Prime, which uses a token-space model architecture (Section A.2.1), ARM with sub-tokens requires explicit factorization, which increases the total FLOPs by ℓ times. For these reasons, we select the token-level ARM as a baseline in Sections 4.1 and 4.2.

A.4.4 Ablation Analysis

This section presents an ablation analysis of the two proposed techniques. We train MDM-Prime (170M parameters) on the SlimPajama dataset and report evaluation results across eight zero-shot benchmarks in Table A7. Both techniques consistently reduce validation loss and improve zero-shot accuracy across all eight commonsense reasoning benchmarks. These results confirm two key findings: (1) lower validation loss correlates with stronger commonsense reasoning performance in MDM-Prime, and (2) the proposed techniques effectively reduce loss and improve accuracy. With the two techniques, the average accuracy increases from 40.02% to 43.76% with 3.74% gains.

A.4.5 Compute-Optimal Scaling of Parameters and Training Tokens

To guide future large-scale pretraining research for MDM-Prime-v2, we extend this analysis to determine the compute-optimal allocation of non-embedding parameters (N) and training tokens (D) for budgets up

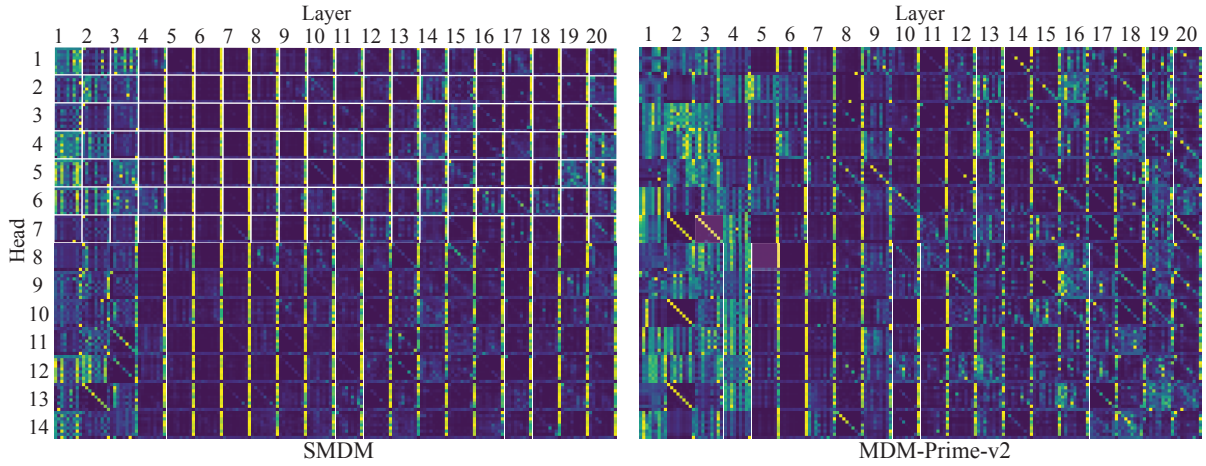


Figure A7: Attention score patterns of SMDM and MDM-Prime-v2. Both models adopt the same model architecture containing 20 layers and 14 attention heads. SMDM exhibits vertical stripes across deeper layers (e.g., Layers 4–20). In contrast, MDM-Prime-v2 displays diagonal structures and a richer diversity of patterns.

Technique 1	Technique 2	Loss	SciQ	SocialIQA	McTaco	TruthfulQA	BoolQ	ANLI	ARC-e	OBQA	Avg.
		2.63	68.7	34.2	53.2	26.0	48.4	34.9	33.6	21.2	40.02
✓		2.34	69.5	35.3	**65.9**	**28.4**	48.8	34.4	34.0	23.8	42.51
✓	✓	**2.31**	**73.6**	**36.2**	64.9	26.6	**50.0**	**35.4**	**39.2**	**24.2**	**43.76**

Table A7: Ablation study of the proposed two techniques on zero-shot benchmarks. The models contain 170M parameters and are trained on SlimPajama. The original MDM-Prime model is trained with $\ell = 3$. MDM-Prime-v2 adopts Technique 1 (random shuffling) and Technique 2 (binary encoding) with $\ell = 15$.

to 10^{25} FLOPs. As illustrated in Fig. A8, our predictions indicate that for target model sizes of 7B, 14B, and 32B, the corresponding optimal training token counts are 2.9T, 7.3T, and 21.7T, respectively.

A.4.6 Preliminary Results on 7B-Parameter Scale

Current large-scale diffusion language models (e.g., LLaDA 8B (Nie et al., 2025b), Dream 7B (Ye et al., 2025)) are not fully open-sourced and cannot be reproduced. The primary reason is that these models are trained on proprietary datasets, preventing future work from studying their pre-training dynamics. This motivates us to establish a training recipe for fully open diffusion LLMs. We adopt the fully open Dolma-6T dataset (Team, 2025) and use the standard LLaMA (Touvron et al., 2023) architecture with a context length of 4,096 for MDM-Prime-v2 (7B parameters). Our preliminary run trains MDM-Prime-v2 (7B) on 1.2T tokens using 192 NVIDIA H200 GPUs. As this run has not yet reached the compute-optimal setting discussed in Section A.3.1, where a 7B model should be trained on at least 2.9T tokens, we report only the training curve and downstream performance on SciQ (Welbl et al., 2017), PIQA (Bisk et al., 2020), and ARC-Challenge (Clark et al., 2018) as preliminary results. The loss curve is shown in Fig. A9, and downstream results are presented in Table A8. The most significant challenge we encountered at this scale is training efficiency. For MDMs with bidirectional attention, model FLOPs utilization (MFU) peaks at only 37%, falling short of the 45% typically achieved by standard ARMs. We hypothesize that `torch.compile` is optimized for causal attention transformers and does not yield full acceleration for diffusion LLMs. Developing more hardware-efficient pre-training methods remains an important direction for future work.

A.5 Related Works

The foundational framework for discrete diffusion was established by (Sohl-Dickstein et al., 2015) and subsequently expanded to real-world generative tasks via D3PM (Austin et al., 2021) and CTMC (Campbell et al., 2022). Recent efforts have adapted these principles to large language modeling, where MDMs (Sahoo et al., 2024; Shi et al., 2024) have emerged as a particularly scalable framework among various discrete diffusion alternatives (Gulrajani and Hashimoto, 2023; Gat et al., 2024). While the

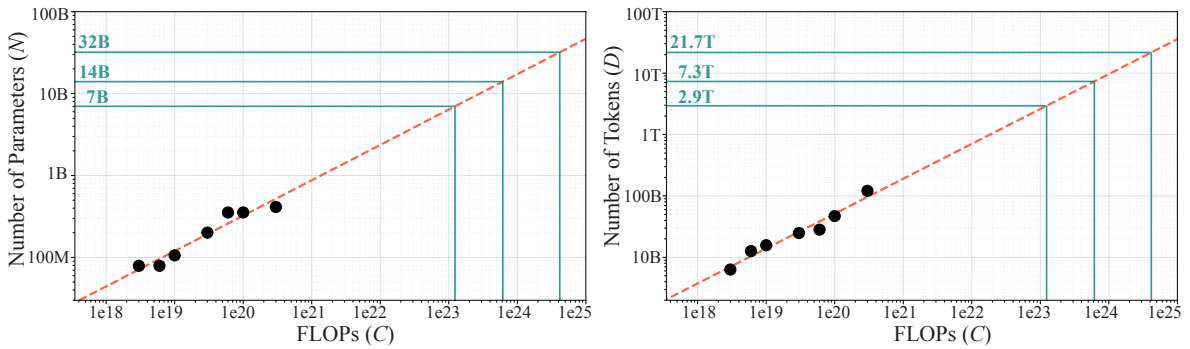


Figure A8: Compute-optimal allocation of the number of non-embedding parameters (N) and training tokens (D) for MDM-Prime-v2 under different FLOPs (C). Left: Optimal N as a function of C. Right: Optimal D as a function of C. Black points represent the compute-optimal empirical samples. The red dashed lines indicate the fitted power-law regression extended to larger budgets. Teal annotations highlight the training token counts required to optimally train models of specific target sizes (e.g., 7B, 14B, and 32B).

Tokens	SciQ	PIQA	ARC-Challenge
0.2T	90.67	55.33	26.00
0.5T	92.67	57.33	28.67
1.2T	92.67	59.33	30.00

Table A8: Zero-shot evaluation of MDM-Prime-v2 (7B) on SciQ, PIQA, and ARC-Challenge as a function of training tokens. All metrics are accuracy-based; higher values indicate better performance. The results are evaluated with 256 Monte Carlo inference steps and 150 instances on each task.

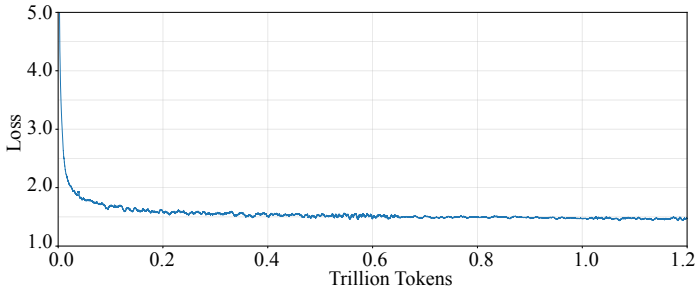


Figure A9: Training loss curve of MDM-Prime-v2 (7B) over the course of pre-training. The model is trained on 1.2T tokens using the Dolma-6T dataset.

scaling behavior of MDMs have been thoroughly investigated under both compute-constrained (Nie et al., 2025a; Ni et al., 2025; von Rütte et al., 2025) and data-constrained (Prabhudesai et al., 2025) regimes, a significant efficiency gap against ARMs remains. We investigate the role of the subtokenizer in shaping the MDM-Prime training objective and its influence on scaling behavior and downstream performance. Through a principled analysis of token granularity and sub-token entropy, we develop MDM-Prime-v2 and validate its effectiveness through large-scale pretraining, demonstrating state-of-the-art zero-shot commonsense reasoning performance at the 1.1B parameter scale.

A.6 Limitations and Discussions

This section outlines three potential directions for further enhancing MDM-Prime-v2.

Optimal Encoding Strategy. MDM-Prime-v2 adopts a random token index shuffling operation as a simple implementation to address distributional issues caused by the BPE tokenizer. However, **Proposition 3.2** suggests that the optimal design is a function that maximizes the entropy of $\boldsymbol{y}_t$. Although our experimental results demonstrate that index shuffling effectively increases entropy—yielding a near-optimal design—we anticipate that a rigorously derived algorithmic approach could further enhance performance. As an ablation, we compare random shuffling against a greedy assignment algorithm that maximizes entropy given corpus token-frequency statistics; results are reported in Table A9 (49M-parameter models trained for 3×10^{18} FLOPs) with the corresponding CDF of token IDs under greedy assignment shown in Fig. A10. Greedy assignment achieves marginally higher entropy but brings negligible performance improvement (Δloss = 0.002). Given that it requires additional corpus statistics whereas random shuffling is fully off-the-shelf, we recommend random shuffling for practical use. Nevertheless, designing entropy-maximizing assignment strategies that remain computationally lightweight represents a promising direction for future work within the MDM-Prime-v2 framework.

	Entropy (↑)	Loss (↓)
Random shuffling	0.9936	2.860
Greedy assignment	0.9943	2.858
Maximum	1.0000	-

Table A9: Entropy of sub-tokens across different subtokenization strategies. The results are evaluated on C4. The maximum entropy is given by $-\log_2 \frac{1}{b}$, where $b = \lceil \sqrt[\ell]{V} \rceil$ and $V = 50,257$. Random shuffling permutes token indices via a random assignment; greedy assignment iteratively pairs the most and least frequent tokens to balance probability mass across sub-tokens (see CDF in Fig. A10).

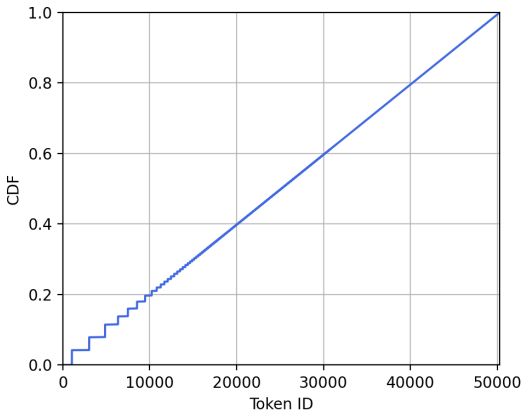


Figure A10: Cumulative distribution function (CDF) of token indices under greedy assignment, computed using the GPT-2 tokenizer on C4.

A Complete Training Pipeline at 7B Scale. While 7B parameters has become a standard model size among contemporary language models, completing a full compute-optimal pretraining run at this scale remains beyond our current computational budget. A compute-optimal 7B model requires training on at least 2.9T tokens (Section A.3.1), yet our preliminary run, spanning approximately two weeks on 192 NVIDIA H200 GPUs including engineering optimization and debugging, reaches only 1.2T tokens. An additional challenge at this scale is training efficiency: MDMs with bidirectional attention achieve a peak model FLOPs utilization (MFU) of only 37%, compared to approximately 45% for standard ARMs, likely because `torch.compile` is optimized for causal attention and does not fully accelerate bidirectional architectures. Completing the compute-optimal 7B run and closing this efficiency gap, through hardware-aware kernel design or compiler support for bidirectional attention, represent important practical milestones for scaling MDM-Prime-v2 to the operating regime of modern language models.

Post-training of MDM-Prime-v2. This work primarily focuses on improving the pretraining of MDM-Prime via enhanced subtokenizer designs. However, as established in recent works (Nie et al., 2025b), post-training also plays a critical role in downstream performance. Therefore, establishing an explicit relationship between subtokenizer design and downstream capabilities represents a promising direction for future research.

A.7 Potential Risks & AI Assistants

Our work does not involve direct societal risks, as it is a fundamental research contribution focused on improving the training objective and architecture of masked diffusion language models. The datasets used (C4 and SlimPajama) are publicly available and widely adopted benchmarks. AI assistance was used solely for grammar editing.