THE DIFFUSION DUALITY, CHAPTER II: Ψ-SAMPLERS AND EFFICIENT CURRICULUM

Anonymous authors
Paper under double-blind review

ABSTRACT

Uniform-state discrete diffusion models excel at few-step generation and guidance due to their inherent ability to self-correct, making them more preferable than autoregressive or masked diffusion models in these settings. However, their sampling efficiency has been limited by the reliance on standard posterior samplers, which plateau in quality as the steps increase. In this work, we introduce a novel family of "Predictor–Corrector" (PC) samplers for discrete diffusion models that generalize prior methods and apply to arbitrary noise processes. When paired with uniform-state diffusion, our samplers significantly outperform ancestral sampling on both language and vision tasks: achieving lower generative perplexity at matched unigram entropy on OpenWebText and better FID/IS scores on CIFAR10. Crucially, unlike conventional samplers, our PC methods continue to improve generation quality with more sampling steps, narrowing the gap with masked diffusion. Beyond sampling, we develop a fast and memory-efficient curriculum for Duo^{++}'s (our method) Gaussian relaxation phase, which avoids materializing large Gaussian-diffused one-hot vectors. This reduces training time by 25% compared to Duo while maintaining similar validation perplexity on OpenWebText and LM1B and strong downstream performance.

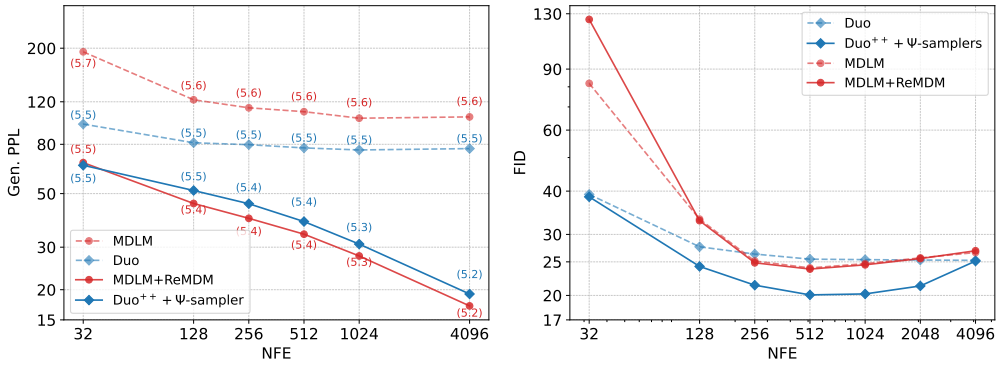

Figure 1: **Performance on language modeling and image modeling. (Left)**: Generative perplexity of Duo^{++} (ours) as a function of the number of sampling steps (NFEs). Duo^{++} generalizes ReMDM (Wang et al., 2025) and the performance consistently improve with the number of sampling steps. We annotate each curve with the average unigram entropy per generated sequence as a proxy for diversity. **(Right)**: On CIFAR-10, Duo^{++} achieves lower FID than MDLM (with ReMDM). Moreover, Duo^{++} obtains a better FID in just 128 steps than Duo with ancestral sampling in 4096 steps.

1 INTRODUCTION

Diffusion models are powerful generative algorithms that have achieved remarkable success in modeling continuous data domains, including images (Ho et al., 2020a; Rombach et al., 2022), audio (Kong et al., 2021; Liu et al., 2023b; Huang et al., 2023), and videos (Ho et al., 2022; Esser et al., 2023; Blattmann et al., 2023; Polyak et al., 2025). Recent advances have extended diffusion

models to categorical data, demonstrating their potential for language modeling (Austin et al., 2023; Lou et al., 2024; Sahoo et al., 2024; Shi et al., 2025; Ou et al., 2025; Sahoo et al., 2025a;b), graphs (Liu et al., 2023a), and molecules (Lee et al., 2025). Unlike autoregressive models that generate tokens sequentially from left to right, diffusion language models can decode tokens in parallel and in any order while leveraging bidirectional contextual information. This capability enables the design of language models that can be significantly faster than their autoregressive counterparts while maintaining strong downstream performance (Song et al., 2025; Labs et al., 2025).

Discrete diffusion models primarily employ one of two noise distributions: a uniform prior or a masked prior that concentrates all probability mass on a special [MASK] token. *Uniform-state diffusion models* (USDMs) offer a major advantage through their ability to self-correct mistakes, as they allow tokens to be revised multiple times during generation. In contrast, standard masked diffusion models (MDMs) update each token exactly once, preventing error correction during generation. Due to this self-correction capability, USDMs significantly outperform MDMs in generation in a few steps, particularly after distillation (Sahoo et al., 2025a). Furthermore, in applications that require guidance to steer generation towards specific targets by optimizing reward functions, USDMs prove to be much more suitable than autoregressive or MDM approaches (Schiff et al., 2025). However, USDMs face notable limitations: Their generation quality has not yet matched that of MDMs in high-sampling-step regimes, and their modeling capacity, as measured by likelihood, remains inferior to that of MDMs. Although Sahoo et al. (2025a) proposed a curriculum learning strategy (Bengio et al., 2009) that narrows the likelihood gap, this curriculum approach is computationally expensive.

To address MDMs' inability to remask tokens, Wang et al. (2025) introduced ReMDM-"Predictor-Corrector" (PC) samplers that generalize and outperform earlier PC methods (Campbell et al., 2022; Gat et al., 2024). These samplers substantially improve the inference time scaling behavior of MDMs. However, PC methods for uniform-state diffusion remain underexplored. Campbell et al. (2022) proposed PC methods for samplers that take advantage of the rate change matrices of the continuous-time Markov chain (CTMC) formulation of discrete diffusion processes, but such samplers are known to perform worse than ancestral samplers Lou et al. (2024); Schiff et al. (2025). Furthermore, while the curriculum learning strategy from Sahoo et al. (2025a) closes the likelihood gap between USDM and MDM, each curriculum step is computationally more expensive than standard training, resulting in a slower overall training.

We propose Duo^{++} to address these challenges, which expands the design space of USDMs using non-Markovian *superposition posteriors* (or as we refer in this paper, Ψ-posteriors). These posteriors align with the intermediate marginals of discrete diffusion processes and give rise to Ψ-samplers with predictor–corrector capabilities that are crucial for improving sample quality. In addition, Duo^{++} introduces an efficient curriculum learning strategy that advances the approach of Sahoo et al. (2025a) by accelerating training and reducing memory usage.

In summary, our contributions are threefold: (1) we propose a family of non-Markovian posteriors (ψ-posteriors) for discrete diffusion with arbitrary priors that share the same marginals as the Markovian discrete diffusion process Sec. 3. (2) We demonstrate that the induced Ψ-samplers improve text and image generation while scaling better than standard ancestral samplers in high NFE regimes, closing the performance gap with respect to MDMs coupled with remasking samplers in high NFE regimes for text generation Sec. 5.1 and surpassing them on image generation tasks Sec. 4. (3) We reformulate the curriculum learning strategy proposed in Sahoo et al. (2025a), achieving a $2\times$ speedup while reducing peak memory usage by 33% and end-to-end training time by 25%, while maintaining similar perplexity (Figure 1, right, Table 5) and downstream task accuracy (Table 1).

2 Background

Notation Let $\mathcal{V} := \{\mathbf{x} \in \{0,1\}^K : \sum_{i=1}^K \mathbf{x}_i = 1\}$ denote the set of one-hot encodings of discrete random variables over K categories. Let $\mathbf{x}^{1:L} \in \mathcal{V}^L$ and $[\mathbf{x}^\ell]_{\ell=1}^L \in \mathcal{V}^L$ denote a sequence of L discrete variables in $\mathcal{V}$ and $\mathbf{x}^\ell$ denote the entry ℓ^{th} in $\mathbf{x}^{1:L}$. Let Δ denote the K simplex. For $\mathbf{v} \in \Delta$, let $\text{Cat}(\cdot; \mathbf{v})$ denote a categorical distribution such that $\mathbb{P}(\mathbf{x}_i = 1) = \mathbf{v}_i$, for $\mathbf{x} \sim \text{Cat}(\cdot; \mathbf{v})$. Let $\langle \mathbf{a}, \mathbf{b} \rangle$ and $\mathbf{a} \odot \mathbf{b}$ denote the dot and Hadamard products between two respectively. Let $\mathbf{1} = \{1\}^K$ denote the all-ones vector.

2.1 Discrete Diffusion Models

Consider the clean data $\mathbf{x}^{1:L}$ drawn from the data distribution q_{data}. Discrete diffusion models (Sohl-Dickstein et al., 2015; Austin et al., 2023) define a sequence of increasingly noisy distributions $(q_t)_{t\in[0,1]}$, interpolating from q_{data} to a prior distribution $\text{Cat}(.;\boldsymbol{\pi})^{1:L}$ using Markovian transitions defined independently across input dimensions (Campbell et al., 2022; Sahoo et al., 2024; Shi et al., 2025; Ou et al., 2025; Schiff et al., 2025; Sahoo et al., 2025a). Let $\mathbf{z}_t^{1:L} \sim \prod_{\ell=1}^{L} q_t(.|\mathbf{x}^\ell)$ denote the intermediate latents at time step t. This work focuses on interpolating noise processes (Sahoo et al., 2024), whose conditional marginal distribution takes the form:

$$\mathbf{z}_t^\ell \sim q_t(.|\mathbf{x}^\ell) = \text{Cat}(.;\alpha_t \mathbf{x}^\ell + (1-\alpha_t)\boldsymbol{\pi}), \tag{1}$$

where $\alpha_t \in [0,1]$ is monotonically decreasing with t, and is known as the *noise schedule*. (1) defines the *forward process*, which progressively corrupts the data. The goal is to learn a *generative process* p_θ, parameterized by a neural network with parameters θ, that reverses this forward process to map from $\text{Cat}(.;\boldsymbol{\pi})^{1:L}$ back to q_{data}. The model is typically trained by minimizing the "Negative Evidence Lower Bound" (NELBO). The choice of prior $\boldsymbol{\pi}$ gives rise to two popular variants: Masked Diffusion Models (MDMs) and Uniform-state Diffusion Models (USDMs), which we discuss in the following.

2.1.1 Masked Diffusion Processes

MDMs (Sahoo et al., 2024; Shi et al., 2025; Ou et al., 2025) use a masked prior, where $\boldsymbol{\pi} = \mathbf{m} \in \mathcal{V}$ is the one-hot representation of a special [MASK] token (Devlin et al., 2019). During the forward process (1), tokens either remain unchanged or transition to the masked state $\mathbf{m}$, after which they stay masked. This behavior carries over to the reverse process. The posterior of the reverse process $q_{s|t}^{\text{MDM}}$ for $0 \le s < t < 1$ can be derived using Bayes' Rule, and that would be:

$$q_{s|t}^{\text{MDM}}(.|\mathbf{z}_t^\ell, \mathbf{x}^\ell) = \begin{cases} \text{Cat}\left(.; \frac{\alpha_s - \alpha_t}{1-\alpha_t}\mathbf{x}^\ell + \frac{1-\alpha_s}{1-\alpha_t}\mathbf{z}_t^\ell\right) & \text{if } \mathbf{z}_t^\ell = \mathbf{m}, \\ \text{Cat}(.;\mathbf{x}^\ell) & \text{otherwise.} \end{cases} \tag{2}$$

The approximate reverse posterior is $p_{s|t}^\theta = \prod_\ell q_{s|t}^{\text{MDM}}(.|\mathbf{z}_t^\ell, \mathbf{x}^\ell = \mathbf{x}_\theta^\ell(\mathbf{z}_t^{1:L}, t))$ where $\mathbf{x}_\theta : \mathcal{V}^L \times [0,1] \to \Delta^L$ is the denoising model. A key limitation is that once unmasked, tokens cannot be remasked (2). This creates compounding errors during inference, as the denoising model $\mathbf{x}_\theta$ imperfectly models the clean data.

Predictor Corrector Methods Wang et al. (2025) propose ReMDM samplers that maintain the same marginals as (2) during the reverse generation process, while allowing remasking and generalizing all previous predictor-corrector methods (Campbell et al., 2022; Gat et al., 2024).

2.1.2 Uniform-state Diffusion Processes

Alternatively, discrete diffusion models can use a uniform prior $\boldsymbol{\pi} = \mathbf{1}/K$ (Schiff et al., 2025; Sahoo et al., 2025a). This choice allows tokens to change values multiple times throughout the generative process, in contrast to masked diffusion. This property allows USDMs to excel in few-step generation (Sahoo et al., 2025a) and guidance applications (Schiff et al., 2025).

USDMs admit the following posterior distribution $q_{s|t}^{\text{USDM}}$ (for brevity, we simply write $q_{s|t}$ for $q_{s|t}^{\text{USDM}}$):

$$q_{s|t}(.\mid \mathbf{z}_t^\ell, \mathbf{x}^\ell) = \text{Cat}\left(.; \frac{K\alpha_t \mathbf{z}_t^\ell \odot \mathbf{x}^\ell + (\alpha_{t|s} - \alpha_t)\mathbf{z}_t^\ell + (\alpha_s - \alpha_t)\mathbf{x}^\ell + (1-\alpha_{t|s})(1-\alpha_s)\mathbf{1}/K}{K\alpha_t \langle \mathbf{z}_t^\ell, \mathbf{x}^\ell \rangle + 1 - \alpha_t}\right). \tag{3}$$

This posterior induces the following NELBO (Sahoo et al., 2025a):

$$\text{NELBO}\left(q, p_\theta; \mathbf{x}^{1:L}\right) = -\mathbb{E}_{t\sim\mathcal{U}[0,1],\, q_t(\mathbf{z}_t^\ell|\mathbf{x}^\ell;\alpha_t)} \sum_{\ell\in[L]} f(\mathbf{z}_t^\ell, \mathbf{x}_\theta^\ell(\mathbf{z}_t^\ell, t), \alpha_t; \mathbf{x}^\ell), \tag{4}$$

where

$$f(\mathbf{z}_t^\ell, \mathbf{x}_\theta^\ell(\mathbf{z}_t^\ell, t), \alpha_t; \mathbf{x}^\ell) = \frac{\alpha_t'}{K\alpha_t}\Bigg[\frac{K}{\bar{\mathbf{x}}_i^\ell} - \frac{K}{(\bar{\mathbf{x}}_\theta^\ell)_i} - \left(\zeta_t \mathbb{1}_{\mathbf{z}_t^\ell = \mathbf{x}^\ell} + \mathbb{1}_{\mathbf{z}_t^\ell \neq \mathbf{x}^\ell}\right)\sum_j \log\frac{(\bar{\mathbf{x}}_\theta^\ell)_i}{(\bar{\mathbf{x}}_\theta^\ell)_j}$$
$$- K\frac{\alpha_t}{1-\alpha_t}\log\frac{(\bar{\mathbf{x}}_\theta^\ell)_i}{(\bar{\mathbf{x}}_\theta^\ell)_m}\mathbb{1}_{\mathbf{z}_t^\ell \neq \mathbf{x}} - \left((K-1)\zeta_t \mathbb{1}_{\mathbf{z}_t^\ell = \mathbf{x}^\ell} - \frac{1}{\zeta_t}\mathbb{1}_{\mathbf{z}_t^\ell \neq \mathbf{x}^\ell}\right)\log\zeta_t\Bigg]. \quad (5)$$

Here, $\bar{\mathbf{x}}^\ell = K\alpha_t\mathbf{x}^\ell + (1-\alpha_t)\mathbf{1}$, $\bar{\mathbf{x}}_\theta^\ell = K\alpha_t\mathbf{x}_\theta^\ell(\mathbf{z}_t, t) + (1-\alpha_t)\mathbf{1}$, α_t' denotes the time derivative of α_t, $i = \arg\max_{j\in[K]}(\mathbf{z}_t^\ell)_j$ is the nonzero entry of $\mathbf{z}_t$, $\zeta_t = \frac{1-\alpha_t}{K\alpha_t+1-\alpha_t}$, and m denotes the index in $\mathbf{x}$ corresponding to 1, that is,x $\mathbf{x}_m = 1$.

The Diffusion Duality Sahoo et al. (2025a) show that USDMs emerge from an underlying Gaussian diffusion process (Sohl-Dickstein et al., 2015; Ho et al., 2020b; Song et al., 2021; Kingma et al., 2023) defined on the one-hot representation $\mathbf{x}^\ell \in \mathcal{V}$. The Gaussian diffusion introduces a sequence of noisy latents $\mathbf{w}_t^\ell \in \mathbb{R}^K \sim \tilde{q}_t(.|\mathbf{x}^\ell)$ with marginal $\tilde{q}_t(.|\mathbf{x}; \tilde{\alpha}_t) = \mathcal{N}(.; \tilde{\alpha}_t\mathbf{x}^\ell, (1-\tilde{\alpha}_t{}^2)\mathbf{I}_K)$, where $(\tilde{\alpha}_t)_{t\in[0,1]}$ is a monotonically decreasing noise schedule. Let $\arg\max : \mathbb{R}^K \to \mathcal{V}$ map a continuous vector $\mathbf{w} \in \mathbb{R}^K$ to the one-hot vector corresponding to the index of its largest entry in $\mathbf{w}$, this is, $\arg\max(\mathbf{w}) = \arg\max_{\mathbf{z}\in\mathcal{V}}\mathbf{z}^\top\mathbf{w}$. When applied to Gaussian latents $\mathbf{w}_t^\ell$, $\arg\max$ transforms them to the discrete latents $\mathbf{z}_t^\ell$ whose marginals take the form: $\mathbf{z}_t^\ell \sim q_t(.|\mathbf{x}^\ell; \alpha_t := \mathcal{T}(\tilde{\alpha}_t))$, where the function $\mathcal{T} : [0,1] \to [0,1]$ is the *Diffusion Transformation operator*:

$$\mathcal{T}(\tilde{\alpha}_t) = \frac{K}{K-1}\left[\int_{-\infty}^{\infty}\phi\left(z - \frac{\tilde{\alpha}_t}{\sqrt{1-\tilde{\alpha}_t{}^2}}\right)\Phi^{K-1}(z)\mathrm{d}z - \frac{1}{K}\right], \quad (6)$$

where $\phi(z) = \exp(-z^2)/\sqrt{2\pi}$ and $\Phi(z) = \int_{-\infty}^{z}\phi(t)\mathrm{d}t$ are the standard Normal PDF and CDF, respectively. More formally, this relationship is expressed as $q_t(\mathbf{z}_t|\mathbf{x}; \mathcal{T}(\tilde{\alpha}_t)) = [\arg\max]_\star\tilde{q}_t(\mathbf{w}_t|\mathbf{x}; \tilde{\alpha}_t)$ where the $\star$ operator denotes the *pushforward* of the K-dimensional Gaussian density under the $\arg\max$ map, yielding a categorical distribution with K classes.

Curriculum Learning Sahoo et al. (2025a) demonstrate that training USDMs with the following loss where the denoising transformer $\mathbf{x}_\theta : \Delta^L \cup \mathcal{V}^L \times [0,1] \to \Delta^L$ is modified to handle both continuous latents and discrete latents lowers the training variance and speeds up convergence:

$$\mathrm{NELBO}(q, p_\theta; \mathbf{x}^{1:L})$$
$$= \mathbb{E}_{\mathbf{x},t\sim\mathcal{U}[\beta,\gamma],\tilde{q}_t}\sum_{\ell\in[L]} f\Big(\mathbf{z}_t^\ell := \arg\max(\mathbf{w}_t^\ell), \mathbf{x}_\theta^\ell([\mathrm{softmax}(\mathbf{w}_t^{\ell'}/\tau)]_{\ell'=1}^L, t), \alpha_t := \mathcal{T}(\tilde{\alpha}_t); \mathbf{x}^\ell\Big). \quad (7)$$

Notice that the discrete inputs ($\arg\max(\mathbf{w}_t^\ell)$) are approximated with continuous tempered softmax $\mathrm{softmax}(\mathbf{w}_t^{\ell'}/\tau)$ where $\tau > 0$. During the early stages of training, we optimize the objective in (7), while in the later stages the model is trained directly with the true NELBO (4). It is important to understand how the denoising model simultaneously handles both continuous- and discrete-valued latents. For a sequence of latents $[\mathbf{y}^\ell]_{\ell=1}^L \in \mathcal{V}^L \cup \Delta^L$, the token embeddings in the first layer are computed by matrix multiplication: $(\mathbf{y}^\ell)_{\ell\in[L]}^\top\mathbf{V}$ with $\mathbf{V} \in \mathbb{R}^{K\times m}$ denoting the vocabulary embedding matrix and m the embedding dimension. For discrete inputs $(\mathbf{y}^\ell)_{\ell\in[L]} \in \mathcal{V}$, this operation is reduced to standard embedding lookups. In contrast, continuous inputs $(\mathbf{y}^\ell)_{\ell\in[L]} \in \Delta^K$ act as "soft lookups", producing convex combinations of the vocabulary embeddings. However, explicitly materializing one-hot vectors $\mathbf{w}_t^{1:L}$ during training is memory intensive and significantly slows down the computation.

2.2 Diffusion Guidance

For continuous data, diffusion models have achieved state-of-the-art controllable generation through both classifier-based (Sohl-Dickstein et al., 2015; Dhariwal & Nichol, 2021) and classifier-free guidance (Nichol & Dhariwal, 2021; Ho & Salimans, 2022). These approaches have since been extended to discrete data (Gruver et al., 2023). Let $y \in \{1, \ldots, M\}$ denote one of the M possible classes. For Classifier-free-guidance, the following reverser posterior is used that modulates the

strength of the guidance term via the temperature parameter γ (Nisonoff et al., 2024; Schiff et al., 2025):

$$\log p_\theta{}^\gamma(\mathbf{z}_s^{1:L} \mid y, \mathbf{z}_t^{1:L}) = \gamma \log p_\theta(\mathbf{z}_s^{1:L} \mid y, \mathbf{z}_t^{1:L}) + (1-\gamma) \log p_\theta(\mathbf{z}_s^{1:L} \mid \mathbf{z}_t^{1:L}), \tag{8}$$

where p_θ is the approximate reverse posterior introduced in Sec. 2.1.

3 The Ψ-Posteriors

Multiple joint distributions can give rise to the same marginals for the discrete diffusion process as defined in (1). In this work, we introduce a family of posteriors, denoted Ψ, and that share the same marginals as in (1); see Suppl. A.2 for details. These alternative generative processes are non-Markovian and apply both to the Masked diffusion processes and to the Uniform-state diffusion processes. Specifically, we define the posteriors for the generative process as:

$$\Psi_{s|t}(.|\mathbf{x}^\ell, \mathbf{z}_t^\ell) = \kappa_t q_{s|t}(.|\mathbf{z}_t^\ell, \mathbf{x}^\ell) + (1-\kappa_t) q_s(.|\mathbf{x}^\ell);\ \forall \ell \in [L] \tag{9}$$

where $\kappa_t \in [0,1]$ and $\Psi_1(.|\mathbf{x}^\ell) = \text{Cat}(.|\boldsymbol{\pi})$, with $\boldsymbol{\pi} = \mathbf{m}$ for MDMs and $\boldsymbol{\pi} = \mathbf{1}/K$ for USDMs. (9) is thus a linear combination of the forward process (1) and the reverse posteriors (2, 3) of standard discrete diffusion models. We therefore refer to these as *superposition posteriors*, or simply Ψ-posteriors.

Ψ-Forward Processes Consider the interpolating diffusion process in (1) discretized into T steps. Let $\mathbf{z}_{t(i)}^{1:L}$ denote the latent variables at times $t(i) = i/T$ for $0 \le i \le T$. The joint distribution factorizes independently across all the tokens as: $\Psi(\mathbf{z}_{0:1}^{1:L}|\mathbf{x}^{1:L}) = \prod_\ell \Psi(\mathbf{z}_{0:1}^\ell|\mathbf{x}^\ell)$ where $\Psi(\mathbf{z}_{0:1}^\ell|\mathbf{x}^\ell) = \Psi_1(\mathbf{z}_1^\ell|\mathbf{x}^\ell) \prod_{i=1}^T \Psi_{s|t}(\mathbf{z}_{s(i)}^\ell|\mathbf{z}_{t(i)}^\ell, \mathbf{x}^\ell)$. In what follows, we use s, t as shorthand for $s(i), t(i)$, respectively. The forward process can be derived from Bayes' rule: $\Psi(\mathbf{z}_t^\ell|\mathbf{z}_s^\ell, \mathbf{x}^\ell) = \Psi(\mathbf{z}_s^\ell|\mathbf{z}_t^\ell, \mathbf{x}^\ell)\Psi(\mathbf{z}_t^\ell|\mathbf{x}^\ell)/\Psi(\mathbf{z}_s^\ell|\mathbf{x}^\ell)$. Unlike the Markovian interpolating process in (1), this forward process is no longer Markovian, since each $\mathbf{z}_t^\ell$ may depend on both $\mathbf{z}_s^\ell$ and $\mathbf{x}^\ell$. These marginals therefore describe a non-Markovian forward generative process.

Ψ-Reverse Processes In Suppl. A.1, we show that the approximate reverse posterior takes the form:

$$[\Psi_{s|t}^\theta(.|\mathbf{z}_t^{1:L})]^\ell = \kappa_t q_{s|t}(.|\mathbf{z}_t^\ell, \mathbf{x}_\theta^\ell(\mathbf{z}_t^\ell, t)) + (1-\kappa_t)\left[\alpha_s q_{0|t}(.|\mathbf{z}_t^\ell, \mathbf{x}_\theta^\ell(\mathbf{z}_t^{1:L}, t)) + (1-\alpha_s)\boldsymbol{\pi}\right]. \tag{10}$$

where $\mathbf{x}_\theta(\mathbf{z}_t^{1:L}, t)$ denotes the denoising model. We dub (10) as Ψ-sampler. For $(\kappa_t = 1)_{t\in[0,1]}$, we recover the standard ancestral sampler defined in (2) for MDMs and (3) for USDMs. Notice that for $\kappa_t < 1$, $\Psi_{s|t}$ corresponds to a noisier version of the ancestral sampler marginal $q_{s|t}$. This is analogous to Predictor-Corrector methods in Gaussian diffusion (Song et al., 2021), where the corrector introduces additional Gaussian noise. In our case, q_t plays the role of the corrector, while $q_{s|t}$ acts as the predictor. The Ψ-posteriors also admit a principled NELBO formulation (see Suppl. A.3), though this is not directly relevant for our purposes.

Corollary For $\pi = \mathbf{m}$, different choices of the functional form of κ_t recover all known predictor-corrector formulations in the literature (Campbell et al., 2022; Gat et al., 2024; Wang et al., 2025). Thus, the Ψ framework subsumes these samplers as special cases and provides a unified perspective on predictor-corrector methods for discrete diffusion. The proof is provided in Suppl. A.4.

4 Scalable Curriculum for Faster Training

Curriculum learning, as introduced in Sahoo et al. (2025a), improves the likelihood of USDM by reducing the variance of the gradient and accelerating convergence. A central operation involves computing a weighted average over all K embedding vectors, where K is the vocabulary size (Sec. 2.1.2). However, explicitly materializing the weights is both memory- and compute-intensive, especially since modern vocabularies can reach hundreds of thousands of tokens. This makes naive implementation a significant bottleneck.

We propose an efficient alternative that avoids constructing the full weight vector. Our key insight is that only a small number of weights are appreciably different from zero, so the weighted average can be well approximated using a subset of embedding vectors $k \ll K$. In the following, we describe how this approximation can be performed efficiently.

Table 1: **Accuracy on multiple-choice question answering datasets.** Abbreviations: Arc-e (ARC-Easy), Arc-c (ARC-Challenge), HSwag (HellaSwag), WinoG (Winogrande), PIQA (Physical Intelligence Question Answering), OQA (OpenBookQA). †Results from Deschenaux et al. (2025). Duo^{++} ($k = 2$) achieves slightly higher accuracy than Duo on 4 out of 6 tasks. Overall, Duo^{++} and Duo perform similarly. The highest accuracy among USDMs is bolded. The absolute best per column is underlined.

	Arc-e	Arc-c	HSwag	WinoG	PIQA	MathQA	OQA
AR Transformer	<u>44.95</u>	23.04	30.55	<u>52.80</u>	<u>63.71</u>	<u>22.24</u>	19.00
MDLM†	34.26	24.66	<u>31.54</u>	51.93	57.89	20.70	<u>28.60</u>
Duo	28.11	25.43	26.46	47.20	51.14	20.00	23.40
Duo^{++} ($k = 2$)	27.32	**<u>26.11</u>**	26.26	49.64	**52.12**	20.40	**27.80**
Duo^{++} ($k = 3$)	**28.28**	25.00	25.89	47.36	50.65	**21.01**	23.00
Duo^{++} ($k = 5$)	28.03	25.77	**26.90**	**50.12**	51.25	20.20	25.40

Problem Setup Let $\mathbf{V}_i$ denote the i^{th} row in the vocabulary embedding matrix $\mathbf{V}$. Define $\text{softmax}(\mathbf{w}_t^\ell/\tau)_i = \frac{\omega_i}{\mathcal{W}}$ with $\omega_i = \exp((\mathbf{w}_t^\ell)_i/\tau)$ and $\mathcal{W} = \sum_i^K \omega_i$. Recall that $\mathbf{w}_t^\ell = \tilde{\alpha}_t \mathbf{x} + \tilde{\sigma}_t \boldsymbol{\epsilon}$ is a Gaussian latent. Thus, $\text{softmax}(\mathbf{w}_t^\ell/\tau)^\top \mathbf{V} = \frac{1}{\mathcal{W}} \sum_{i\in[K]} \omega_i \mathbf{V}_i$. Let o denote the index s.t. $\mathbf{x}_o = 1$ which we dub the "true token". The remaining indices form $\mathcal{Z}(\mathbf{x}) := [K] - \{o\}$. Let $\mathcal{P}(k) = \{(a_1, a_2, \ldots, a_k) : a_i \in \mathcal{Z}(\mathbf{y}), a_i \neq a_j \ \forall i \neq j\}$ denote a set of tuples of length k consisting of indices that correspond to 0 in $\mathbf{x}$. We seek to identify a subset $\mathcal{I} \subset [K] : |\mathcal{I}| = k$ s.t $\frac{1}{\mathcal{W}} \sum_{i\in[K]} \omega_i \mathbf{V}_i \approx \frac{1}{\mathcal{W}} \sum_{i\in\mathcal{I}} \omega_i \mathbf{V}_i$.

The challenge is to select the k largest entries of $\mathbf{w}_t^\ell$ without explicitly constructing all K Gaussian random variables. Notice that $(\mathbf{w}_t^\ell = \tilde{\sigma}_t \epsilon_\ell)_{\ell\in\mathcal{Z}(\mathbf{x})}$ and $(\mathbf{w}_t^\ell)_{\ell=o} = \tilde{\alpha}_t + \tilde{\sigma}_t \epsilon_o$ where $\epsilon_{\{.\}} \sim \mathcal{N}(0, 1)$. Instead of sampling all $K - 1$ Gaussian variables for $\mathcal{Z}(\mathbf{x})$, we leverage the distribution of order statistics. Let $W_{1:k}$ denote the top-k values of $(\epsilon_i)_{i\in\mathcal{Z}(\mathbf{x})}$, where $W_1 = \max(\{\epsilon_i : i \in \mathcal{Z}(\mathbf{x})\})$ and $W_i = \max(\{\epsilon_i : i \in \mathcal{Z}(\mathbf{x})\} - \{W_1, \ldots, W_{i-1}\}) \quad \forall i \in \{2, \ldots, k\}$. k largest Gaussians can be drawn efficiently via inverse-transform sampling (see Suppl. B.1) Once we have the top-k Gaussian latents, two cases arise:

Case 1: If $\tilde{\alpha}_t + \tilde{\sigma}_t \epsilon_o < \min(\{\tilde{\sigma}_t W_i : i \in [k]\})$, then the true token o is not in the top-k. In this case, $\mathcal{I} \subset \mathcal{Z}(\mathbf{x})$ and that $\mathcal{I} \sim \mathcal{P}(k)$. The calculation of the corresponding weights is also fairly straightforward, where $(\omega_{\mathcal{I}[i]} = \exp(\frac{\tilde{\sigma}_t W_i}{\tau}))_{i\in[k]}$ and $\mathcal{I}[i]$ denote the entry i^{th} in $\mathcal{I}$.

Case 2: If $\tilde{\alpha}_t + \tilde{\sigma}_t \epsilon_o > \min(\{\tilde{\sigma}_t W_i : i \in [k]\})$, then top-$k$ includes the true token: $\mathcal{I} = \bar{\mathcal{I}} \cup (o)$ where $\bar{\mathcal{I}} \sim \mathcal{P}(k-1)$. The corresponding weights are: $\left(\omega_{\mathcal{I}[i]} = \exp(\frac{\tilde{\sigma}_t W_i}{\tau})\right)_{i\in[k-1]}$ and $\omega_o = \exp\left(\frac{1+\tilde{\sigma}_t W_o}{\tau}\right)$.

Finally, we must approximate $\mathcal{W} = \sum_{i=1}^K \omega_i$. Direct computation requires all K Gaussian samples, which is infeasible. Instead, we show in Suppl. B.6 that (with $c = \min(\omega_{1:k})$):

$$\mathcal{W} = \sum_{i=1}^{K} \omega_i \approx \sum_{i\in\mathcal{I}} \omega_i + (K - k) \left[\frac{\tilde{\sigma}_t{}^2}{2} + \log \Phi\left(\frac{c - \tilde{\sigma}_t{}^2}{\tilde{\sigma}_t} \right) - \log \Phi\left(\frac{c}{\tilde{\sigma}_t} \right) \right]. \tag{11}$$

5 Experiments

We evaluate our approach on language modeling (Sec. 5.1.1) and image generation (Sec. 5.1.2), showing that Ψ-samplers substantially improve text and image quality, making USDMs as performant as MDMs. In Sec. 5.2, we further demonstrate that Duo^{++}, combined with the efficient curriculum strategy (Sec. 4), achieves performance comparable to Duo (Sahoo et al., 2025a)—the current state-of-the-art USDM—while reducing memory usage by 33% and training 25% faster.

5.1 Ψ-Samplers

We evaluate the Ψ-samplers in the context of both language modeling and image modeling to attest its generality across different modalities.

5.1.1 Language Modeling

Our experiments indicate that (1) **Ψ-samplers substantially improve generative perplexity for USDMs**, with gains becoming especially pronounced once the NFEs exceed the sequence length, thereby closing the gap with MDMs; and (2) **unlike ancestral sampling, which quickly plateaus with increasing NFEs, Ψ-samplers continue to yield improvements in sample quality**.

Experimental Settings For the masked diffusion baseline, we use pre-trained MDLM checkpoints (Sahoo et al., 2024) trained for 1M steps with a batch size of 512 on OpenWebText (OWT; Gokaslan & Cohen (2019) and sequence length $L = 1024$. We train our Duo^{++} using the same recipe for a comparable number of steps. We distill the MDLM and Duo checkpoint using SDTT (Deschenaux & Gulcehre, 2025) and DCD (Sahoo et al., 2025a) for 5 rounds of 10k steps using their respective implementation and default hyperparameters. Our primary baselines include MDLM and Duo^{++} with ancestral sampling, as well as MDLM with the ReMDM sampler (Wang et al., 2025). For additional training details, we refer the reader to the original work. We evaluate sample quality using Generative Perplexity (Gen. PPL, ↓) measured with GPT-2 Large and sample diversity using unigram entropy, following standard practice (Sahoo et al., 2024; 2025a). Sampling is performed in 64-bit precision (Zheng et al., 2025). See Suppl. C.1 for further details.

Results Figure 1 (left) shows the Gen. PPL and the entropy as a function of the NFE, for the ancestral and Ψ-samplers. Duo^{++} with Ψ-samplers significantly outperforms MDLM with ReMDM and ancestral samplers across an entire range of NFEs. Especially as the number of NFEs increases beyond the sequence length, the standard ancestral sampling plateaus while Ψ-samplers continuously improves the quality of the sampler.

How to choose κ_t ? Similarly to ReMDM, Ψ-samplers comes with a hyperparameter (κ_t (Eq. 9)) that requires tuning for strong performance. We set t and κ_t using two related heuristics, visualized in Figure 2. With the first heuristic, t is linearly decreasing. With the second one, inspired by ReMDM and referred to as "loop". When $t \in [0, t_{\text{off}}] \cup [t_{\text{on}}, 1]$, t is linearly decreasing, while it is kept constant when $t \in [t_{\text{off}}, t_{\text{on}}]$. For ReMDM, when t is kept constant, $\alpha_t = 0.9$, and $t_{\text{on}} = 0.55$ and $t_{\text{off}} = 0.05$. With both heuristics, we set $\kappa_t = c \in [0, 1)$ when $t \in [t_{\text{off}}, t_{\text{on}}] \subseteq [0, 1]$, and use $\kappa_t = 1$ otherwise. We achieve the best results when the Ψ-samplers are used at lower noise levels only. As the width of the interval $t_{\text{on}} - t_{\text{off}}$ increases, it becomes necessary to increase κ_t closer to 1 to maintain the quality of the sample. For example, with $\kappa_t = 0.02$, strong results are obtained only when $t_{\text{on}} - t_{\text{off}}$ is narrow and at low noise levels (e.g. $t_{\text{on}} = 0.15$, $t_{\text{off}} = 0.1$). In contrast, using $\kappa_t = 0.02$ at higher noise levels (e.g., $t_{\text{on}} = 0.45$, $t_{\text{off}} = 0.4$) leads to noticeable sample degradation. We obtain the best results for Duo^{++} with t linearly decreasing, that is, *without* is the looping strategy. Comprehensive numerical results for different choices of κ_t are provided in Suppl. C.1.

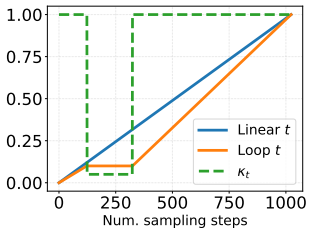

Figure 2: Evolution of t and the associated κ_t under the loop and linear t-decrease scheduling strategies.

5.1.2 Image Modeling

Our experiments indicate that (1) **Duo^{++} with Ψ-samplers produce a significantly higher quality image than MDLM with ReMDM sampler**, and (2) with a well-tuned κ_t and using 128 sampling steps, **Ψ-samplers reach a lower FID than the ancestral sampler in 4096, representing a 32× acceleration.**

Experimental Setup Our experimental setup is similar to Austin et al. (2023). We train a U-Net backbone with 35M parameters on the CIFAR-10 dataset, on raw pixels for 1.5M steps, using a batch size of 128, a learning rate of 2×10^{-4}, a dropout rate of 0.1, and random horizontal flips. The U-net is class conditional and we train with a class dropout probability of 0.1 to allow discrete classifier-free guidance (CFG; Ho & Salimans (2022); Schiff et al. (2025)). See Suppl. C.1 for additional experimental details. We report *Fréchet Inception Distance* (FID; Heusel et al. (2018)) and the *Inception Score* (IS; Salimans et al. (2016)) between the training set and 50K generated samples and use a guidance strength $\gamma = 1$ as used in (Schiff et al.,

2025) and a temperature of 0.8. Recall that selecting $\kappa_t = 1 - \frac{\sigma_t}{1-\alpha_s}$, where $s < t$ recovers ReMDM, where σ_t is the reMDM hyperparameter that controls the amount of remasking.

Results Figure 1 *(right)* shows that Ψ-samplers substantially improves both FID and IS for Duo^{++} helping it surpass MDLM with the ReMDM sampler. Interestingly, FID and IS do not improve monotonically as the number of sampling steps increases. In fact, the best performance is achieved in 512 steps. Impressively, Ψ-samplers attains a lower FID in just 128 steps than ancestral sampling achieves in 4096 steps. This represents a 32$\times$ reduction in sampling steps for similar quality. We found that the this trend also holds for the inception score.

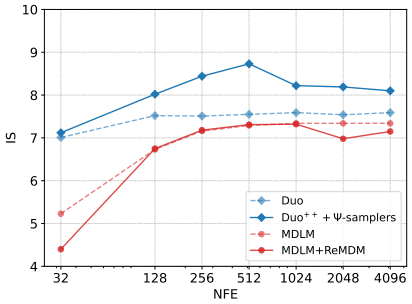

Figure 3: Duo^{++} significantly outperforms Duo, MDLM and ReMDM in Inception Score.

How to pick κ_t? The detailed experimental results are presented in Suppl. C.1. Consistent with our observations on OWT, we find that Ψ-samplers are most effective when applied at lower noise levels and with κ_t close to 1. The best FID is achieved with $\kappa_t = 0.95$, $t_{\text{on}} = 0.6$, and $t_{\text{off}} = 0.1$ using NFEs=512 with a linearly decreasing t. For both MDLM+ReMDM and Duo+Ψ-samplers, the best FID is found with linearly decreasing t, $t_{\text{on}} = 0.6, t_{\text{off}} = 0.1$ and $\kappa_t \in \{0.95, 0.98\}$. See Table 6 and Table 7 for all hyperparameter ablations.

5.2 Fast Curriculum

Our experiments show that with the efficient curriculum learning strategy in Sec. 4, **Duo^{++} trains 25% faster and matches Duo and on standard likelihood benchmarks and downstream tasks.**

Experimental settings We train Duo^{++} with the scalable curriculum (Sec. 4) on OWT and LM1B (Chelba et al., 2014). We train all models for 1M steps, using a batch size of 512. For LM1B, we use the `bert-base-uncased` tokenizer with a context length of 128, padding shorter sequences. This setup follows previous work (Sahoo et al., 2024; Lou et al., 2024; He et al., 2022). For OWT, we use the GPT-2 tokenizer (Radford et al., 2019), and reserve the last 100k documents for validation, following (Sahoo et al., 2025a; 2024). We follow Lou et al. (2024) and use a modified diffusion transformer (DiT) (Peebles & Xie, 2023) with rotary positional encoding (Su et al., 2023). We evaluate the impact of $k = \{2, 3, 5\}$ in the efficient curriculum. All models are trained on 16 H100 GPUs with bfloat16 precision. Training uses the loss in (7), with $\tau = 0.001$ for the first 500K steps and $(\beta, \gamma) = (0.03, 0.15)$ (Sahoo et al., 2025a).

Likelihood results Table 2 shows that on both LM1B and OWT, our efficient curriculum Duo^{++} matches the performance of Duo with its expensive curriculum. The lowest validation perplexity is achieved with $k = 2$, although $k = 2, 5$ performs similarly. We also evaluate the Zero-Shot perplexity in OWT following Sahoo et al. (2025a; 2024) and find that Duo^{++} achieves a performance comparable to Duo. That is, we evaluate the validation splits of Penn Treebank (Marcus et al., 1993), WikiText (Merity et al., 2016), LM1B (Chelba et al., 2014), LAMBADA (Paperno et al., 2016), AG News (Zhang et al., 2016) and scientific articles from ArXiv and PubMed (Cohan et al., 2018). Table 5 shows that Duo^{++} reaches a zero-shot probability similar to that of Duo while requiring 25% less GPU-hours.

Downstream Tasks In Table 1, we compare the multiple-choice accuracy of Duo, Duo^{++}, MDLM (Sahoo et al., 2024), and an autoregressive transformer using the `lm-eval-harness` suite (Gao et al., 2024). Although `lm-eval-harness` was originally designed for autoregressive models, it was adapted for diffusion models by recent work (Deschenaux & Gulcehre, 2024; Nie et al., 2025b;a; Shi et al., 2025) (details in Suppl. C.3). We find that Duo^{++} achieves an accuracy similar to that of Duo.

Throughput and peak memory usage Table 4 reports throughput and peak memory usage for Duo and Duo^{++}. Compared to Duo, Duo^{++} reduces peak memory usage by about 33% and doubles the speed of the Curriculum Learning phase. When applying Curriculum Learning for half of the training steps, Duo^{++} trains 25% faster than Duo on the 138M-parameter scale. In particular, peak memory and throughput remain stable across our sparse curriculum for $k \in \{2, 3, 5\}$.

6 Related work

Discrete diffusion models Discrete diffusion (Sohl-Dickstein et al., 2015; Austin et al., 2023; Campbell et al., 2022; Lou et al., 2024; Sahoo et al., 2024; Shi et al., 2025; Schiff et al., 2025; Ou et al., 2025; Sahoo et al., 2025a) and discrete flow matching (Campbell et al., 2024; Gat et al., 2024) have recently gained increasing attention due to advances in their foundations and more efficient implementations. Most discrete diffusion and flow matching methods use a uniform or masked noise distribution, although Shaul et al. (2024); von Rütte et al. (2025); Holderrieth et al. (2025) have explored more general processes. In this work, we present a general predictor-corrector algorithm for discrete diffusion on arbitrary spaces.

Table 2: Test perplexity (PPL) on LM1B and OWT. Lower is better. †Results from Sahoo et al. (2025a). Best Uniform-state diffusion numbers are **bolded**. Duo and Duo^{++} achieve comparable performance across both datasets while requiring 25% fewer GPU-hours, demonstrating the effectiveness of our memory-efficient curriculum.

	LM1B	OWT
Autoregressive		
Transformer†	22.3	17.5
Masked Diffusion		
SEDD Absorb† (Lou et al., 2024)	<u>32.7</u>	<u>24.1</u>
MDLM† (Sahoo et al., 2024)	**27.0**	**23.2**
Uniform-state Diffusion		
SEDD Uniform† (Lou et al., 2024)	40.3	29.7
UDLM† (Schiff et al., 2025)	31.3	27.4
Duo† (Sahoo et al., 2025a)	**29.9**	**25.2**
Duo^{++} (Ours), $k = 2$	<u>30.0</u>	**25.2**
Duo^{++} (Ours), $k = 3$	30.1	<u>25.3</u>
Duo^{++} (Ours), $k = 5$	30.2	25.4

Predictor-Corrector samplers Previous work showed that remasking can improve performance by allowing the model to correct sampling errors. ReMDM (Wang et al., 2025) generalizes previous predictor-corrector methods (Campbell et al., 2022; Gat et al., 2024) in the masked setting. Our approach further generalizes ReMDM to support arbitrary diffusion processes. Unlike Lezama et al. (2023); Zhao et al. (2025); Liu et al. (2025), who train an additional corrector module, our method does not introduce additional learned components.

Other discrete diffusion samplers Park et al. (2024) adapts the sampling step size to the noise level to outperform samplers that use a fixed step size. Although we use a uniform step size, our sampler is compatible with any step-size schedule. Ren et al. (2025) studies high-order sampling algorithms, whereas we rely on first-order information only. However, the posterior in (9) could be estimated using high-order samplers. Thus, Ψ-samplers are complementary to these lines of work.

7 Conclusion

We introduced a unified and practical framework for predictor-corrector sampling in discrete diffusion language models through Ψ-posteriors. By linearly superposing the forward and reverse diffusion processes (9), the Ψ-*posteriors* preserve the marginals of standard diffusion models. Importantly, the Ψ-*posteriors*, and associated Ψ-samplers subsumes prior masked-diffusion PC samplers Campbell et al. (2022); Gat et al. (2024); Wang et al. (2025) as special cases, and naturally extend to discrete diffusion models with uniform priors.Empirically, Duo^{++} with Ψ-samplers matches the performance of MDMs on natural language generation and achieves stronger FID/IS scores on CIFAR-10. Moreover, they exhibit superior scaling: performance continues to improve with NFEs, unlike ancestral samplers, which plateau. Finally, we propose a scalable training curriculum, inspired by Sahoo et al. (2025a), that reduces the peak memory usage by 33% and shortens the training time by 25%.

REFERENCES

Jacob Austin, Daniel D. Johnson, Jonathan Ho, Daniel Tarlow, and Rianne van den Berg. Structured denoising diffusion models in discrete state-spaces, 2023. URL https://arxiv.org/abs/2107.03006.

Yoshua Bengio, Jérôme Louradour, Ronan Collobert, and Jason Weston. Curriculum learning. In *International Conference on Machine Learning*, 2009. URL https://api.semanticscholar.org/CorpusID:873046.

Jon Louis Bentley. *Programming Pearls*. Addison-Wesley Professional, Boston, MA, 2nd edition, 1999. ISBN 978-0201657883.

Roger Berger and George Casella. *Statistical Inference*. Duxbury Press, Florence, AL, 2 edition, June 2001.

Andreas Blattmann, Robin Rombach, Huan Ling, Tim Dockhorn, Seung Wook Kim, Sanja Fidler, and Karsten Kreis. Align your latents: High-resolution video synthesis with latent diffusion models. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pp. 22563–22575, 2023.

Kim C. Border. Lecture 15: Order statistics; conditional expectation. https://healy.econ.ohio-state.edu/kcb/Ma103/Notes/Lecture15.pdf, 2021. Course notes for MA103.

Andrew Campbell, Joe Benton, Valentin De Bortoli, Tom Rainforth, George Deligiannidis, and Arnaud Doucet. A continuous time framework for discrete denoising models, 2022. URL https://arxiv.org/abs/2205.14987.

Andrew Campbell, Jason Yim, Regina Barzilay, Tom Rainforth, and Tommi Jaakkola. Generative flows on discrete state-spaces: Enabling multimodal flows with applications to protein co-design, 2024. URL https://arxiv.org/abs/2402.04997.

Ciprian Chelba, Tomas Mikolov, Mike Schuster, Qi Ge, Thorsten Brants, Phillipp Koehn, and Tony Robinson. One billion word benchmark for measuring progress in statistical language modeling, 2014. URL https://arxiv.org/abs/1312.3005.

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

Justin Deschenaux and Caglar Gulcehre. Promises, outlooks and challenges of diffusion language modeling, 2024. URL https://arxiv.org/abs/2406.11473.

Justin Deschenaux and Caglar Gulcehre. Beyond autoregression: Fast llms via self-distillation through time, 2025. URL https://arxiv.org/abs/2410.21035.

Justin Deschenaux, Lan Tran, and Caglar Gulcehre. Partition generative modeling: Masked modeling without masks, 2025. URL https://arxiv.org/abs/2505.18883.

Jacob Devlin, Ming-Wei Chang, Kenton Lee, and Kristina Toutanova. Bert: Pre-training of deep bidirectional transformers for language understanding, 2019. URL https://arxiv.org/abs/1810.04805.

Luc Devroye. *Non-Uniform Random Variate Generation*. Springer-Verlag, 1986. URL https://www.springer.com/gp/book/9780387963051.

Prafulla Dhariwal and Alexander Nichol. Diffusion models beat gans on image synthesis. *Advances in neural information processing systems*, 34:8780–8794, 2021.

Patrick Esser, Johnathan Chiu, Parmida Atighehchian, Jonathan Granskog, and Anastasis Germanidis. Structure and content-guided video synthesis with diffusion models. In *Proceedings of the IEEE/CVF International Conference on Computer Vision*, pp. 7346–7356, 2023.

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, Eric Tang, Anish Thite, Ben Wang, Kevin Wang, and Andy Zou. The language model evaluation harness, 07 2024. URL https://zenodo.org/records/12608602.

Itai Gat, Tal Remez, Neta Shaul, Felix Kreuk, Ricky T. Q. Chen, Gabriel Synnaeve, Yossi Adi, and Yaron Lipman. Discrete flow matching, 2024. URL https://arxiv.org/abs/2407.15595.

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

Nate Gruver, Samuel Stanton, Nathan Frey, Tim GJ Rudner, Isidro Hotzel, Julien Lafrance-Vanasse, Arvind Rajpal, Kyunghyun Cho, and Andrew G Wilson. Protein design with guided discrete diffusion. *Advances in neural information processing systems*, 36:12489–12517, 2023.

Zhengfu He, Tianxiang Sun, Kuanning Wang, Xuanjing Huang, and Xipeng Qiu. Diffusionbert: Improving generative masked language models with diffusion models, 2022. URL https://arxiv.org/abs/2211.15029.

Martin Heusel, Hubert Ramsauer, Thomas Unterthiner, Bernhard Nessler, and Sepp Hochreiter. Gans trained by a two time-scale update rule converge to a local nash equilibrium, 2018. URL https://arxiv.org/abs/1706.08500.

Jonathan Ho and Tim Salimans. Classifier-free diffusion guidance, 2022. URL https://arxiv.org/abs/2207.12598.

Jonathan Ho, Ajay Jain, and Pieter Abbeel. Denoising diffusion probabilistic models. *Advances in neural information processing systems*, 33:6840–6851, 2020a.

Jonathan Ho, Ajay Jain, and Pieter Abbeel. Denoising diffusion probabilistic models, 2020b. URL https://arxiv.org/abs/2006.11239.

Jonathan Ho, Tim Salimans, Alexey Gritsenko, William Chan, Mohammad Norouzi, and David J Fleet. Video diffusion models. *arXiv:2204.03458*, 2022.

Peter Holderrieth, Marton Havasi, Jason Yim, Neta Shaul, Itai Gat, Tommi Jaakkola, Brian Karrer, Ricky T. Q. Chen, and Yaron Lipman. Generator matching: Generative modeling with arbitrary markov processes, 2025. URL https://arxiv.org/abs/2410.20587.

Ari Holtzman, Jan Buys, Li Du, Maxwell Forbes, and Yejin Choi. The curious case of neural text degeneration, 2020. URL https://arxiv.org/abs/1904.09751.

Qingqing Huang, Daniel S. Park, Tao Wang, Timo I. Denk, Andy Ly, Nanxin Chen, Zhengdong Zhang, Zhishuai Zhang, Jiahui Yu, Christian Frank, Jesse Engel, Quoc V. Le, William Chan, Zhifeng Chen, and Wei Han. Noise2music: Text-conditioned music generation with diffusion models, 2023. URL https://arxiv.org/abs/2302.03917.

Diederik P. Kingma, Tim Salimans, Ben Poole, and Jonathan Ho. Variational diffusion models, 2023. URL https://arxiv.org/abs/2107.00630.

Zhifeng Kong, Wei Ping, Jiaji Huang, Kexin Zhao, and Bryan Catanzaro. Diffwave: A versatile diffusion model for audio synthesis. In *International Conference on Learning Representations*, 2021. URL https://openreview.net/forum?id=a-xFK8Ymz5J.

Inception Labs, Samar Khanna, Siddhant Kharbanda, Shufan Li, Harshit Varma, Eric Wang, Sawyer Birnbaum, Ziyang Luo, Yanis Miraoui, Akash Palrecha, et al. Mercury: Ultra-fast language models based on diffusion. *arXiv preprint arXiv:2506.17298*, 2025.

Seul Lee, Karsten Kreis, Srimukh Prasad Veccham, Meng Liu, Danny Reidenbach, Yuxing Peng, Saee Paliwal, Weili Nie, and Arash Vahdat. Genmol: A drug discovery generalist with discrete diffusion. *arXiv preprint arXiv:2501.06158*, 2025.

Jose Lezama, Tim Salimans, Lu Jiang, Huiwen Chang, Jonathan Ho, and Irfan Essa. Discrete predictor-corrector diffusion models for image synthesis. In *The Eleventh International Conference on Learning Representations*, 2023. URL https://openreview.net/forum?id=VM8batVBWvg.

Chengyi Liu, Wenqi Fan, Yunqing Liu, Jiatong Li, Hang Li, Hui Liu, Jiliang Tang, and Qing Li. Generative diffusion models on graphs: Methods and applications. *arXiv preprint arXiv:2302.02591*, 2023a.

Haohe Liu, Zehua Chen, Yi Yuan, Xinhao Mei, Xubo Liu, Danilo Mandic, Wenwu Wang, and Mark D Plumbley. Audioldm: text-to-audio generation with latent diffusion models. In *Proceedings of the 40th International Conference on Machine Learning*, pp. 21450–21474, 2023b.

Sulin Liu, Juno Nam, Andrew Campbell, Hannes Stärk, Yilun Xu, Tommi Jaakkola, and Rafael Gómez-Bombarelli. Think while you generate: Discrete diffusion with planned denoising, 2025. URL https://arxiv.org/abs/2410.06264.

Ilya Loshchilov and Frank Hutter. Decoupled weight decay regularization, 2019. URL https://arxiv.org/abs/1711.05101.

Aaron Lou, Chenlin Meng, and Stefano Ermon. Discrete diffusion modeling by estimating the ratios of the data distribution, 2024. URL https://arxiv.org/abs/2310.16834.

Mitchell P. Marcus, Beatrice Santorini, and Mary Ann Marcinkiewicz. Building a large annotated corpus of English: The Penn Treebank. *Computational Linguistics*, 19(2):313–330, 1993. URL https://aclanthology.org/J93-2004/.

Stephen Merity, Caiming Xiong, James Bradbury, and Richard Socher. Pointer sentinel mixture models, 2016. URL https://arxiv.org/abs/1609.07843.

Alex Nichol and Prafulla Dhariwal. Improved denoising diffusion probabilistic models, 2021. URL https://arxiv.org/abs/2102.09672.

Shen Nie, Fengqi Zhu, Chao Du, Tianyu Pang, Qian Liu, Guangtao Zeng, Min Lin, and Chongxuan Li. Scaling up masked diffusion models on text, 2025a. URL https://arxiv.org/abs/2410.18514.

Shen Nie, Fengqi Zhu, Zebin You, Xiaolu Zhang, Jingyang Ou, Jun Hu, Jun Zhou, Yankai Lin, Ji-Rong Wen, and Chongxuan Li. Large language diffusion models, 2025b. URL https://arxiv.org/abs/2502.09992.

Hunter Nisonoff, Junhao Xiong, Stephan Allenspach, and Jennifer Listgarten. Unlocking guidance for discrete state-space diffusion and flow models. *arXiv preprint arXiv:2406.01572*, 2024.

Jingyang Ou, Shen Nie, Kaiwen Xue, Fengqi Zhu, Jiacheng Sun, Zhenguo Li, and Chongxuan Li. Your absorbing discrete diffusion secretly models the conditional distributions of clean data, 2025. URL https://arxiv.org/abs/2406.03736.

Denis Paperno, Germán Kruszewski, Angeliki Lazaridou, Quan Ngoc Pham, Raffaella Bernardi, Sandro Pezzelle, Marco Baroni, Gemma Boleda, and Raquel Fernández. The lambada dataset: Word prediction requiring a broad discourse context, 2016. URL https://arxiv.org/abs/1606.06031.

Yong-Hyun Park, Chieh-Hsin Lai, Satoshi Hayakawa, Yuhta Takida, and Yuki Mitsufuji. *Jump Your Steps*: Optimizing sampling schedule of discrete diffusion models, 2024. URL https://arxiv.org/abs/2410.07761.

William Peebles and Saining Xie. Scalable diffusion models with transformers, 2023. URL https://arxiv.org/abs/2212.09748.

Adam Polyak, Amit Zohar, Andrew Brown, Andros Tjandra, Animesh Sinha, Ann Lee, Apoorv Vyas, Bowen Shi, Chih-Yao Ma, Ching-Yao Chuang, David Yan, Dhruv Choudhary, Dingkang Wang, Geet Sethi, Guan Pang, Haoyu Ma, Ishan Misra, Ji Hou, Jialiang Wang, Kiran Jagadeesh, Kunpeng Li, Luxin Zhang, Mannat Singh, Mary Williamson, Matt Le, Matthew Yu, Mitesh Kumar Singh, Peizhao Zhang, Peter Vajda, Quentin Duval, Rohit Girdhar, Roshan Sumbaly, Sai Saketh Rambhatla, Sam Tsai, Samaneh Azadi, Samyak Datta, Sanyuan Chen, Sean Bell, Sharadh Ramaswamy, Shelly Sheynin, Siddharth Bhattacharya, Simran Motwani, Tao Xu, Tianhe Li, Tingbo Hou, Wei-Ning Hsu, Xi Yin, Xiaoliang Dai, Yaniv Taigman, Yaqiao Luo, Yen-Cheng Liu, Yi-Chiao Wu, Yue Zhao, Yuval Kirstain, Zecheng He, Zijian He, Albert Pumarola, Ali Thabet, Artsiom Sanakoyeu, Arun Mallya, Baishan Guo, Boris Araya, Breena Kerr, Carleigh Wood, Ce Liu, Cen Peng, Dimitry Vengertsev, Edgar Schonfeld, Elliot Blanchard, Felix Juefei-Xu, Fraylie Nord, Jeff Liang, John Hoffman, Jonas Kohler, Kaolin Fire, Karthik Sivakumar, Lawrence Chen, Licheng Yu, Luya Gao, Markos Georgopoulos, Rashel Moritz, Sara K. Sampson, Shikai Li, Simone Parmeggiani, Steve Fine, Tara Fowler, Vladan Petrovic, and Yuming Du. Movie gen: A cast of media foundation models, 2025. URL https://arxiv.org/abs/2410.13720.

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

Yinuo Ren, Haoxuan Chen, Yuchen Zhu, Wei Guo, Yongxin Chen, Grant M. Rotskoff, Molei Tao, and Lexing Ying. Fast solvers for discrete diffusion models: Theory and applications of high-order algorithms, 2025. URL https://arxiv.org/abs/2502.00234.

Robin Rombach, Andreas Blattmann, Dominik Lorenz, Patrick Esser, and Björn Ommer. High-resolution image synthesis with latent diffusion models, 2022. URL https://arxiv.org/abs/2112.10752.

Olaf Ronneberger, Philipp Fischer, and Thomas Brox. U-net: Convolutional networks for biomedical image segmentation, 2015. URL https://arxiv.org/abs/1505.04597.

Subham Sekhar Sahoo, Marianne Arriola, Yair Schiff, Aaron Gokaslan, Edgar Marroquin, Justin T Chiu, Alexander Rush, and Volodymyr Kuleshov. Simple and effective masked diffusion language models, 2024. URL https://arxiv.org/abs/2406.07524.

Subham Sekhar Sahoo, Justin Deschenaux, Aaron Gokaslan, Guanghan Wang, Justin Chiu, and Volodymyr Kuleshov. The diffusion duality, 2025a. URL https://arxiv.org/abs/2506.10892.

Subham Sekhar Sahoo, Zhihan Yang, Yash Akhauri, Johnna Liu, Deepansha Singh, Zhoujun Cheng, Zhengzhong Liu, Eric Xing, John Thickstun, and Arash Vahdat. Esoteric language models. *arXiv preprint arXiv:2506.01928*, 2025b.

Tim Salimans, Ian Goodfellow, Wojciech Zaremba, Vicki Cheung, Alec Radford, and Xi Chen. Improved techniques for training gans, 2016. URL https://arxiv.org/abs/1606.03498.

Tim Salimans, Andrej Karpathy, Xi Chen, and Diederik P. Kingma. Pixelcnn++: Improving the pixelcnn with discretized logistic mixture likelihood and other modifications, 2017. URL https://arxiv.org/abs/1701.05517.

Yair Schiff, Subham Sekhar Sahoo, Hao Phung, Guanghan Wang, Sam Boshar, Hugo Dalla-torre, Bernardo P. de Almeida, Alexander Rush, Thomas Pierrot, and Volodymyr Kuleshov. Simple guidance mechanisms for discrete diffusion models, 2025. URL https://arxiv.org/abs/2412.10193.

Neta Shaul, Itai Gat, Marton Havasi, Daniel Severo, Anuroop Sriram, Peter Holderrieth, Brian Karrer, Yaron Lipman, and Ricky T. Q. Chen. Flow matching with general discrete paths: A kinetic-optimal perspective, 2024. URL https://arxiv.org/abs/2412.03487.

Jiaxin Shi, Kehang Han, Zhe Wang, Arnaud Doucet, and Michalis K. Titsias. Simplified and generalized masked diffusion for discrete data, 2025. URL https://arxiv.org/abs/2406.04329.

Jascha Sohl-Dickstein, Eric A. Weiss, Niru Maheswaranathan, and Surya Ganguli. Deep unsupervised learning using nonequilibrium thermodynamics, 2015. URL https://arxiv.org/abs/1503.03585.

Jiaming Song, Chenlin Meng, and Stefano Ermon. Denoising diffusion implicit models, 2022. URL https://arxiv.org/abs/2010.02502.

Yang Song, Jascha Sohl-Dickstein, Diederik P Kingma, Abhishek Kumar, Stefano Ermon, and Ben Poole. Score-based generative modeling through stochastic differential equations. In *International Conference on Learning Representations*, 2021. URL https://openreview.net/forum?id=PxTIG12RRHS.

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

Jianlin Su, Yu Lu, Shengfeng Pan, Ahmed Murtadha, Bo Wen, and Yunfeng Liu. Roformer: Enhanced transformer with rotary position embedding, 2023. URL https://arxiv.org/abs/2104.09864.

Dimitri von Rütte, Janis Fluri, Yuhui Ding, Antonio Orvieto, Bernhard Schölkopf, and Thomas Hofmann. Generalized interpolating discrete diffusion, 2025. URL https://arxiv.org/abs/2503.04482.

Guanghan Wang, Yair Schiff, Subham Sekhar Sahoo, and Volodymyr Kuleshov. Remasking discrete diffusion models with inference-time scaling, 2025. URL https://arxiv.org/abs/2503.00307.

Xiang Zhang, Junbo Zhao, and Yann LeCun. Character-level convolutional networks for text classification, 2016. URL https://arxiv.org/abs/1509.01626.

Yixiu Zhao, Jiaxin Shi, Feng Chen, Shaul Druckmann, Lester Mackey, and Scott Linderman. Informed correctors for discrete diffusion models, 2025. URL https://arxiv.org/abs/2407.21243.

Kaiwen Zheng, Yongxin Chen, Hanzi Mao, Ming-Yu Liu, Jun Zhu, and Qinsheng Zhang. Masked diffusion models are secretly time-agnostic masked models and exploit inaccurate categorical sampling, 2025. URL https://arxiv.org/abs/2409.02908.

CONTENTS

A Ψ-POSTERIORS

A.1 APPROXIMATE REVERSE MARGINALS

We parameterize the (generative) Ψ-reverse marginals to have a similar form as the true posterior (9). Therefore, the generative reverse marginals also factorizes over the sequence length. Because $\mathbf{x}^{1:L}$ is not available during sampling, there are two terms in (9) that are intractable. First, we choose to replace the posterior $q_{s|t}(.|\mathbf{z}_t^\ell, \mathbf{x}^\ell)$ by $q_{s|t}(.|\mathbf{z}_t^\ell, \mathbf{x}^\ell = \mathbf{x}_\theta^\ell)$. Additionally, as we cannot sample from $q_s(.|\mathbf{x}^\ell)$ without $\mathbf{x}^\ell$, we replace $\mathbf{x}^\ell$ by $q_{0|t}(.|\mathbf{z}_t, \mathbf{x}^\ell = \mathbf{x}_\theta^\ell), \forall \ell \in [L]$. Replacing these two intractable terms yield our generative reverse marginals:

$$\Psi_{s|t}^\theta(.|\mathbf{z}_t) = \kappa_t q_{s|t}(.|\mathbf{z}_t, \mathbf{x} = \mathbf{x}_\theta(\mathbf{z}_t, t)) + (1-\kappa_t)\left[\alpha_s q_{0|t}(.|\mathbf{z}_t, \mathbf{x} = \mathbf{x}_\theta(\mathbf{z}_t, t)) + (1-\alpha_s)\pi\right]. \tag{12}$$

Note that for the masked posterior (2), $q_{0|t}(.|\mathbf{z}_t, \mathbf{x} = \mathbf{x}_\theta(\mathbf{z}_t, t)) = \mathbf{x}_\theta(\mathbf{z}_t, t)$.

A.2 PROOF THAT THE Ψ-POSTERIORS HAVE THE CORRECT MARGINALS

Let $\Psi_{s|t}(.|\mathbf{x}^\ell, \mathbf{z}_t^\ell)$ denote the Ψ-posteriors defined in (9). Let s denotes $s(k) = t(k-1)$ and t denotes $t(k)$. To prove that the Ψ-posteriors have the correct marginals, we proceed by (downwards) induction, similar to Song et al. (2022). First, note that $\Psi_s(\mathbf{z}_s^\ell|\mathbf{x}^\ell)$ can be written as a marginalization over $\tilde{\mathbf{z}}_t^\ell$, for $s < t$:

$$\Psi_s(\mathbf{z}_s^\ell|\mathbf{x}^\ell) = \sum_{\tilde{\mathbf{z}}_t^\ell} \Psi_t(\tilde{\mathbf{z}}_t^\ell|\mathbf{x}^\ell)\Psi_{s|t}(\mathbf{z}_s^\ell|\tilde{\mathbf{z}}_t^\ell, \mathbf{x}) \tag{13}$$

Base Case Let $\Psi_1(\mathbf{z}_1^\ell|\mathbf{x}^\ell)$ denote the marginal at time $t = 1$. By definition in (9), $\Psi_1(\mathbf{z}_1^\ell|\mathbf{x}^\ell) = \text{Cat}(.|\boldsymbol{\pi})$. Therefore, the Ψ-posteriors have the correct marginal for $t = 1$.

Induction hypothesis Suppose that the Ψ-posteriors have the correct marginal for a certain $t \leq 1$, that is, $\Psi_t(.|\mathbf{x}^\ell) = q_t(.|\mathbf{x}^\ell)$.

Inductive step Based on the induction hypothesis, we now show that $\Psi_s(.|\mathbf{x}^\ell) = q_s(.|\mathbf{x}^\ell)$, for $s(k) = t(k-1)$. Indeed

$$\begin{aligned}
\Psi_s(.|\mathbf{x}^\ell) &\overset{(1)}{=} \sum_{\tilde{\mathbf{z}}_t^\ell} \Psi_t(\tilde{\mathbf{z}}_t^\ell|\mathbf{x}^\ell)\Psi_{s|t}(\mathbf{z}_s^\ell|\tilde{\mathbf{z}}_t^\ell, \mathbf{x}^\ell) \\
&\overset{(2)}{=} \sum_{\tilde{\mathbf{z}}_t^\ell} q_t(\tilde{\mathbf{z}}_t^\ell|\mathbf{x}^\ell)\Psi_{s|t}(\mathbf{z}_s^\ell|\tilde{\mathbf{z}}_t^\ell, \mathbf{x}^\ell) \\
&\overset{(3)}{=} \sum_{\tilde{\mathbf{z}}_t} q_t(\tilde{\mathbf{z}}_t^\ell|\mathbf{x}^\ell)\left[\kappa_t q_{s|t}(\mathbf{z}_s^\ell|\mathbf{x}^\ell, \tilde{\mathbf{z}}_t^\ell) + (1-\kappa_t)q_s(\mathbf{z}_s^\ell|\mathbf{x}^\ell)\right] \\
&\overset{(4)}{=} \kappa_t \sum_{\tilde{\mathbf{z}}_t^\ell} q_t(\tilde{\mathbf{z}}_t^\ell|\mathbf{x}^\ell)q_{s|t}(\mathbf{z}_s^\ell|\mathbf{x}^\ell, \tilde{\mathbf{z}}_t^\ell) + (1-\kappa_t)q_s(\mathbf{z}_s^\ell|\mathbf{x}^\ell)\sum_{\tilde{\mathbf{z}}_t^\ell} q_t(\tilde{\mathbf{z}}_t^\ell|\mathbf{x}^\ell) \\
&\overset{(5)}{=} \kappa_t q_s(\mathbf{z}_s^\ell|\mathbf{x}^\ell) + (1-\kappa_t)q_s(\mathbf{z}_s^\ell|\mathbf{x}^\ell) = q_s(\mathbf{z}_s^\ell|\mathbf{x}^\ell).
\end{aligned}$$

Specifically, (1) hold by (13), (2) by the induction hypothesis, (3) by definition of the Ψ-posteriors, (4) by distributing $q_t(\tilde{\mathbf{z}}_t^\ell|\mathbf{x}^\ell)$, (5) by definition of marginal probability (first term), and by observing that $\sum_{\tilde{\mathbf{z}}_t^\ell} q_t(\tilde{\mathbf{z}}_t^\ell|\mathbf{x}^\ell) = 1$ since q_t is normalized. This concludes the inductive step, and shows that the Ψ-posteriors have the correct marginal.

A.3 Negative Evidence Lower Bound

Let $\mathbf{z}_{0:1}^\ell$ denote a reverse trajectory with time indices $\{0, \frac{1}{T}, \frac{2}{T}, \dots, 1\}$ for token ℓ. The joint distribution of $(\mathbf{x}^\ell, \mathbf{z}_{0:1}^\ell)$ under the generative model factorizes as

$$p^\theta(\mathbf{x}^\ell, \mathbf{z}_{0:1}^\ell) = p(\mathbf{x}^\ell \mid \mathbf{z}_0^\ell)\Psi_1(\mathbf{z}_1^\ell)\prod_{i=1}^{T}\Psi_{s|t}^\theta(\mathbf{z}_{s(i)}^\ell \mid \mathbf{z}_{t(i)}^\ell), \tag{14}$$

where each pair $(s(i), t(i))$ denotes one reverse transition with $s(i) < t(i)$. The marginal likelihood is

$$p^\theta(\mathbf{x}^\ell) = \sum_{\mathbf{z}_{0:1}^\ell} p^\theta(\mathbf{x}^\ell, \mathbf{z}_{0:1}^\ell). \tag{15}$$

Introducing the variational distribution $\Psi(\mathbf{z}_{0:1}^\ell \mid \mathbf{x}^\ell) = \Psi_1(\mathbf{z}_1^\ell \mid \mathbf{x}^\ell)\prod_{i=1}^{T}\Psi_{s|t}(\mathbf{z}_{s(i)}^\ell \mid \mathbf{z}_{t(i)}^\ell, \mathbf{x}^\ell)$, Jensen's inequality results in:

$$-\log p^\theta(\mathbf{x}^\ell) \le \mathbb{E}_{\Psi(\mathbf{z}_{0:1}^\ell|\mathbf{x}^\ell)}\big[-\log p(\mathbf{x}^\ell \mid \mathbf{z}_0^\ell)\big] + \mathrm{KL}\big(\Psi_1(\cdot \mid \mathbf{x}^\ell)\big\|\Psi_1\big) \tag{16}$$

$$+\sum_{i=1}^{T}\mathbb{E}_{\Psi(\mathbf{z}_{t(i)}^\ell|\mathbf{x}^\ell)}\left[D_{\mathrm{KL}}\Big(\Psi_{s|t}(\cdot \mid \mathbf{z}_{t(i)}^\ell, \mathbf{x}^\ell) \,\Big\|\, \Psi_{s|t}^\theta(\cdot \mid \mathbf{z}_{t(i)}^\ell)\Big)\right]. \tag{17}$$

This expression is similar to the standard diffusion NELBO, with a reconstruction term, a prior term at t=1, and a sum of KL divergences. As $T \to \infty$, $p(\mathbf{x}^\ell \mid \mathbf{z}_0^\ell)$ concentrates around $\mathbf{x}^\ell$, hence $-\log p(\mathbf{x}^\ell \mid \mathbf{z}_0^\ell) \to 0$. Furthermore, the prior term is zero by definition of the Ψ-posteriors in (9).

A.4 Recovering Predictor-Corrector Methods for Masked Diffusion

Suppose that we work with masked diffusion, hence $\pi = \mathbf{m}$. The Ψ-posteriors can be expanded as

$$\Psi_{s|t}(.|\mathbf{z}_t^\ell) = \kappa_t q_{s|t}(.|\mathbf{z}_t^\ell, \mathbf{x}^\ell) + (1-\kappa_t)\left[\alpha_s q_{0|t}(.|\mathbf{z}_t^\ell, \mathbf{x}^\ell) + (1-\alpha_s)\pi\right] \tag{18}$$

$$= \kappa_t \begin{cases} \mathrm{Cat}(.; \mathbf{z}_t^\ell), & \mathbf{z}_t^\ell \neq \mathbf{m}, \\ \mathrm{Cat}\left(.; \dfrac{(1-\alpha_s)\mathbf{m} + (\alpha_s - \alpha_t)\mathbf{x}^\ell}{1-\alpha_t}\right), & \mathbf{z}_t^\ell = \mathbf{m} \end{cases} + (1-\kappa_t)\left[\alpha_s \mathbf{x}^\ell + (1-\alpha_s)\mathbf{m}\right] \tag{19}$$

$$\overset{(1)}{=} \kappa_t \begin{cases} \mathrm{Cat}(.; \mathbf{x}^\ell), & \mathbf{z}_t^\ell \neq \mathbf{m}, \\ \mathrm{Cat}\left(.; \dfrac{(1-\alpha_s)\mathbf{m} + (\alpha_s - \alpha_t)\mathbf{x}^\ell}{1-\alpha_t}\right), & \mathbf{z}_t^\ell = \mathbf{m} \end{cases} + (1-\kappa_t)\left[\alpha_s \mathbf{x}^\ell + (1-\alpha_s)\mathbf{m}\right] \tag{20}$$

$$= \begin{cases} \mathrm{Cat}(.; \kappa_t \mathbf{x}^\ell + (1-\kappa_t)[\alpha_s \mathbf{x}^\ell + (1-\alpha_s)\mathbf{m}]), & \mathbf{z}_t^\ell \neq \mathbf{m} \\ \mathrm{Cat}\left(.; \kappa_t \frac{(1-\alpha_s)\mathbf{m} + (\alpha_s - \alpha_t)\mathbf{x}^\ell}{1-\alpha_t} + (1-\kappa_t)[\alpha_s \mathbf{x}^\ell + (1-\alpha_s)\mathbf{m}]\right), & \mathbf{z}_t^\ell = \mathbf{m} \end{cases} \tag{21}$$

$$= \begin{cases} \mathrm{Cat}(.; [\kappa_t + (1-\kappa_t)\alpha_s]\mathbf{x}^\ell + (1-\kappa_t)(1-\alpha_s)\mathbf{m}), & \mathbf{z}_t^\ell \neq \mathbf{m} \\ \mathrm{Cat}\left(.; \left[\kappa_t \frac{\alpha_s - \alpha_t}{1-\alpha_t} + (1-\kappa_t)\alpha_s\right]\mathbf{x}^\ell + \left[\kappa_t \frac{1-\alpha_s}{1-\alpha_t} + (1-\kappa_t)(1-\alpha_s)\right]\mathbf{m}\right), & \mathbf{z}_t^\ell = \mathbf{m} \end{cases}, \tag{22}$$

where (1) holds $\mathbf{z}_t^\ell \neq \mathbf{m}$ implies that $\mathbf{z}_t^\ell = \mathbf{x}^\ell$, since in masked diffusion, the latents $\mathbf{z}_t^\ell$ are either a clean token or the masked token.

To conclude, if we pick $\kappa_t = 1 - \frac{\sigma_t}{1-\alpha_s}$, where σ_t is the free parameter in the ReMDM sampler, then the equation reduces to the ReMDM posterior. Therefore, the Ψ-posteriors generalize ReMDM, which itself generalized the FB (Campbell et al., 2022) and DFM (Gat et al., 2024) posteriors. Additionally, the Ψ-posteriors are not limited to masked diffusion, as we showed in this work.

B Fast Curriculum

In this section, we expand on the implementation of the efficient curriculum. In Sec. B.2, we focus on the overall design and challenges of the curriculum. The soundness of our approach relies on a

various mathematical results, which we also elaborate on in this section. Specifically, our efficient curriculum uses inverse transform sampling (Sec. B.3) and the Cumulative Distribution Function (CDF) distribution of the largest (Sec. B.3) and second largest (Sec. B.5) uniform random variable. Furthermore, we derive an analytical expression for the conditional mean of the exponential of a Gaussian random variable in Sec. B.6.

Furthermore, although the efficient curriculum could be implemented using the original definition of the Diffusion Transformation Operator $\mathcal{T}$, we show that $\mathcal{T}$ admits a convenient series expansion in Sec. B.7. This avoids the need to precompute 100k function values, and simplifies the implementation. Finally, in Sec. B.8, we show that $\mathcal{T}$ can be well approximated by a degree-9 polynomial, which removes the need to store a large number of coefficients during training

B.1 Generating the k largest Gaussian random variables out of K

We show that it is possible to generate the k largest Gaussian random variables out of K via inverse transform sampling (Suppl. B.3) as follows.

Given a single uniform random variable $U \sim \mathcal{U}[0, 1]$, one can obtain a standard Gaussian random variable $W = \Phi^{-1}(U)$, where Φ is the Gaussian CDF, via inverse transform sampling. Now assume we have a sorted list of K uniform random variables $U_1 \geq U_2 \geq ... \geq U_K$. Since Φ is a monotonically increasing functions, the largest uniform random variable, U_1, is mapped to the largest Gaussian random variable, i.e. $\Phi^{-1}(U_1)$ is distributed as the largest Gaussian random variable out of K.

As shown in Prop. B.1 the CDF of the largest uniform random variable out of K has an analytical solution. For $u \in [0, 1]$, $P(U_1 \leq u) = u^K$, hence it can be generated via inverse transform sampling.

Furthermore, the distribution of the second largest, conditioned on $U_1 = u_1$ also admits a closed form solution (Suppl. B.5): for $u_2 \in [0, u_1]$, it is given by $P(U_2 \leq u_2) = u_2^{K-1} u_1^{-(K-1)}$, i.e. it is distributed as the largest uniform variable out of $K - 1$, supported on $[0, u_1]$.

Finally, $P(U_3 \leq u_3 | U_2 = u_2, U_1 = u_1) = P(U_3 \leq u_3 | U_2 = u_2)$. Indeed, since $U_2 \leq U_1$, it does not matter what value U_1 takes, since $U_3 \leq U_2$. Therefore $P(U_3 \leq u_3 | U_2 = u_2) = u_3^{K-2} u_2^{-(K-2)}$, i.e. the largest uniform out of $K - 2$.

More generally, the same argument shows that conditioned on $U_i = u_i$, the random variable U_{i+1} is distributed as the largest uniform variable on $[0, u_i]$ out of $K - i + 1$. This shows that we can sample $U_1, ..., U_k$ in decreasing order and without simulating all the K variables. Finally, the U_i can be transformed into the k largest standard Gaussians out of K as $\{\Phi^{-1}(U_i)\}_{i=1}^k$.

B.2 How to Implement Our Fast Curriculum

Duo's curriculum is expensive While Duo (Sahoo et al., 2025a) converges to lower validation perplexities than UDLM (Schiff et al., 2025), the curriculum phase of Duo is expensive. Indeed, it materializes a Gaussian-diffused vector of size $B \times L \times K$, where B represents the batch size, L the context length, and K the vocabulary size. The Gaussian vector is normalized with a low-temperature softmax. Directly sampling a tensor of shape $B \times L \times K$, applying the softmax, and multiplying by the embedding table is computationally and memory intensive, especially for large vocabularies, as the tensor size scales with K. Since Sahoo et al. (2025a) use a low-temperature softmax, only a few entries are nonzero. This observation motivates our solution: approximate sampling of the top-k nonzero entries, with $k \ll K$.

Three Challenges To approximate Duo's curriculum, we must address three main challenges:

- First, we need to sample the k largest zero-mean Gaussian random variables out of K, to emulate the Gaussian Diffusion over the one-hot data samples $\mathbf{x}$ (Sec. B.2.1).
- Secondly, we must estimate the normalization constant of the softmax, without actually sampling the K random variables (Sec. B.2.2).
- Third, we require an efficient method to sample k distinct integers from K without replacement (Sec. B.2.3).

Algorithm 1 Scalable Top-k Approximation for Curriculum Learning

Input Clean token value x, vocabulary size K, top-k parameter k, inverse temperature τ, Gaussian schedules α_t, σ_t
Output Softmax weights $\boldsymbol{\lambda} \in [0,1]^k$, top-k indices $\tilde{\mathbf{x}}$, index of the largest variable $\mathbf{z}_t$.

$\{z_0^{(i)}\}_{i=1}^k \leftarrow$ `sample_top_gaussians`$(k, K-1, 0, \sigma_t)$ ▷ Algo. 2.
$z_\alpha \sim \mathcal{N}(\alpha_t, \sigma_t)$ ▷ Diffusion at the clean data index.
$\boldsymbol{Z}_{\text{top}} \leftarrow$ `top-k` $\left(\{z_\alpha\} \cup \{z_0^{(i)}\}\right)$ ▷ Keep the top k.
$\mu \leftarrow \mathbb{E}[\exp(z \cdot \tau) \mid z < \min(\boldsymbol{Z}_{\text{top}})]$ for $z \sim \mathcal{N}(0, \sigma_t)$ ▷ For normalization, Prop. B.5
$S \leftarrow \sum_{i=1}^k \exp(\boldsymbol{Z}_{\text{top}}^{(i)} \cdot \tau)$ ▷ S will contain the softmax normalization constant.
if $z_\alpha \in \boldsymbol{Z}_{\text{top}}$ **then**
  $S \leftarrow S + (K-k)\mu$ ▷ Approximate non-simulated variables with their conditional mean.
  $\tilde{\mathbf{x}} \leftarrow [x] \cup$ `sample_neq_x`$(k-1)$ ▷ Indices of the top k, Algo. 2.
else
  $S \leftarrow S + (K-k-1)\mu + \exp(z_\alpha \cdot \tau)$
  $\tilde{\mathbf{x}} \leftarrow$ `sample_neq_x`(k)
end if
$\boldsymbol{\lambda}_i \leftarrow \exp(\boldsymbol{Z}_{\text{top}}^{(i)} \cdot \tau)/S$ for $i = 0, \ldots, k-1$
$\mathbf{z}_t \leftarrow \arg\max_i \boldsymbol{Z}_{\text{top}}^{(i)}$ ▷ Index of the top 1.
return $\boldsymbol{\lambda}$, $\tilde{\mathbf{x}}$, $\mathbf{z}_t$

Recall that Algo. 1 shows the pseudocode of the algorithm.

B.2.1 Sampling the top k out of K Normal random variables

Libraries such as `numpy` and `pytorch` provide accurate approximations of the Gaussian CDF Φ and its inverse Φ^{-1}, allowing us to generate Gaussian random variables via inverse transform sampling (Sec. B.3). To sample K Gaussians, we could naively inverse-transform K uniform random variables. Crucially, because Φ^{-1} is monotonic, the k largest uniforms correspond exactly to the k largest Gaussians.

Finally, and importantly, *we do not need to simulate all K uniform random variables to obtain the top-k*. The largest uniform out of K has a closed-form CDF with an analytical inverse (Sec. B.1). Moreover, the second largest, conditioned on the largest, is itself uniform with a reduced support (Sec. B.5). Thus, the top-k uniforms can be sampled sequentially, by first drawing the maximum, then iteratively sample the remaining values in decreasing order.

In practice, a naive implementation of inverse transform sampling is numerically unstable when K is large. For stability, operations should implemented in log-space, and Algo. 2 shows the pseudocode for a log-space implementation

B.2.2 Estimating the normalization constant of the softmax

Computing the normalization constant of the softmax,

$$\text{softmax}(x)_i = \frac{\exp(x_i)}{\sum_{j=1}^K \exp(x_j)}, \tag{23}$$

requires access to all values $\{x_j\}_{j=1}^K$. However, because K is large, we do *not* wish to simulate all K random variables, and therefore cannot compute the softmax normalization constant exactly. Fortunately, we find that when K is large, the contribution of each non-simulated random variable is well approximated by $\mathbb{E}[\exp(X) \mid X < c]$, where $X \sim \mathcal{N}(0, \sigma)$ and c is the smallest among the top k random variables that we have simulated. Recall that the analytical expression of $\mathbb{E}[\exp(X) \mid X < c]$ appears in (11) (proof in Suppl. B.6)

B.2.3 Sampling Integers Without Repetitions and Shuffling

Suppose that $\mathbf{x}$ denotes the one-hot vector of category i. By symmetry, after after applying Gaussian diffusion to $\mathbf{x}$, all entries $\mathbf{x}_j$ such that $j \neq i$ follow the exact same distribution. Therefore, they have the same probability of being one of the top k largest random variable.

To implement the curriculum, we must not only approximate the weights of the embedding combination but also select which embeddings to include. Concretely, we sample k random indices *without repetition* excluding i. If the random variable at position i, corresponding to the clean token, belongs to the top-k, we replace one of the sampled indices with i. Otherwise, we use the k sampled indices directly.

A simple way to sample k random indices without repetition is to shuffle a list of K integers and take the first k. However, this defeats the purpose of our efficient curriculum, as it requires materializing large tensors. Instead, Floyd's algorithm (Bentley, 1999), given in Algo. 3, samples without repetition while avoiding shuffling. Although sequential with k iterations, it is still far more memory- and compute-efficient than shuffling when $k \ll K$.

B.3 Inverse Transform Sampling

The Inverse Transform Sampling method (Devroye, 1986) is an algorithm for simulating continuous random variables with a known Cumulative Distribution Function (CDF) F_X. Implementing Inverse Transform Sampling requires access to the inverse CDF F_X^{-1}, and a source of $i.i.d$ uniform random variables. If $X = F_X^{-1}(U)$, where $U \sim \mathcal{U}[0,1]$, then $X \sim F_X$. Indeed,

$$\mathbb{P}(X \leq x) = \mathbb{P}(F_X^{-1}(U) \leq x) = \mathbb{P}(U \leq F_X(x)) = F_X(x), \tag{24}$$

since for $a \in [0,1]$, $\mathbb{P}(U \leq a) = a$. This shows that X has the correct distribution.

B.4 Distribution of the largest random uniform variables out of K

Additionally, the distribution of the largest uniform random variable out of K admits a simple closed-form expression:

Proposition B.1 (Distribution of the largest random uniform random variable out of K)**.** *$U^{(1)} \geq U^{(2)} \geq ... \geq U^{(K)}$ denote an order statistic over K i.i.d uniform random variables $\mathcal{U}([0,\theta])$ with Cumulative Density Function (CDF) F_U. Suppose that $u \in [0,1]$, then $F_U(u) = \frac{u}{\theta}$. Then, the CDF $F_{U^{(1)}}$ and probability density function (PDF) $f_{U^{(1)}}$ of the largest random variable $U^{(1)}$ are as follows:*

$$\begin{aligned} F_{U^{(1)}}(u) &= F_U^K(u) = u^K \theta^{-K} \\ f_{U^{(1)}}(u) &= K F_U^{K-1}(x) f_U(x) = K F_U^{(K-1)}(x) f(x) = K x^{K-1} \theta^{-K} \end{aligned} \tag{25}$$

Proof.

$$F_{U^{(1)}}(u) = \mathbb{P}(U^{(1)} \leq u) = \mathbb{P}(U_i \leq u \; \forall i) = P(U \leq u)^K = F_U^K(u). \tag{26}$$

The PDF is obtained by differentiation:

$$f_{U^{(1)}}(x) = \frac{d}{dx} F_{U^{(1)}}(u) = K F_X^{K-1}(u) f_U(u), \tag{27}$$

□

B.5 Distribution of the second largest uniform random variable out of K

We use Prop. B.2 to find the distribution of the second largest uniform random variable out of K:

Proposition B.2 (Conditional Density (Berger & Casella, 2001))**.** *Let X, Y be two random variables with joint density $f_{X,Y}$ and marginals f_X, f_Y. Then, the conditional density of X given $Y = y$ is*

$$f_{X|Y=y}(x|y) = \frac{f_{X,Y}(x,y)}{f_Y(y)}. \tag{28}$$

Furthermore, the proof relies on the distribution of a pair of order statistic $(X^{(k)}, X^{(l)})$:

Proposition B.3 (Joint Density of Order Statistics (Berger & Casella, 2001)). *Let $X^{(N)} \geq ... \geq X^{(1)}$ denote an order statistic over N random variables with CDF F and PDF f. Then, the joint density of the variables $X^{(k)}$ and $X^{(l)}$, where $k < l$ is given by*

$$f_{X^{(k)},X^{(l)}}(u,v) = \frac{N!}{(k-1)!(l-k-1)!(N-l)!} F(u)^{k-1}(F(v)-F(u))^{l-k-1}(1-F(v))^{N-l} f(u)f(v). \tag{29}$$

See Border (2021) for a proof. Finally, using Prop. B.2 and B.3, we prove the main result:

Proposition B.4 (Conditional Distribution of $U^{(K-1)}$ given $U^{(K)}$). *Let $U^{(K)} \geq \cdots \geq U^{(1)}$ denote the order statistics of K independent and uniformly distributed random variables on $[0,\theta]$, arranged in descending order. Conditioned on $U^{(K)} = z$, $U^{(K-1)}$ is distributed as the largest of $(K-1)$ i.i.d uniform random variables on $[0,z]$.*

Proof. From Proposition B.3, the joint distribution $f_{X^{(N-1)},X^{(N)}}(u,v)$ is given by

$$f_{X^{(N-1)},X^{(N)}}(u,v) = \frac{N!}{(N-2)!} F_X^{(N-2)}(u) f(u) f(v) = N(N-1)u^{(n-2)}\theta^{-n}. \tag{30}$$

Using Proposition B.2, we can conclude:

$$\begin{aligned} f_{X^{(N-1)}|X^{(N)}}(u \mid v) = \frac{f_{X^{(N-1)},X^{(N)}}(u,v)}{f_{X^{(N)}}(v)} &= \frac{N(N-1)u^{(N-2)}\theta^{-N}}{Nv^{N-1}\theta^{-N}} \\ &= (N-1)u^{(N-2)}v^{(N-1)}, \end{aligned} \tag{31}$$

which is precisely the density of the largest out of $N-1$ independent uniform random variables on $[0,v]$. □

B.6 Conditional mean of the exponential of a Gaussian

Finding the analytical expression of $\mathbb{E}\left[\exp(X)|X < c\right]$ requires the expression for the conditional density, given that $X \in A$ for A are Borel set with non-zero probability:

Proposition B.5 (Conditional Density). *Let X be a random variable with density f_X, and let A be a Borel set such that $\mathbb{P}(X \in A) > 0$. Then the conditional density of X given $X \in A$ is*

$$f_{X|X\in A}(x) = \frac{f_X(x)\mathbb{1}\{x \in A\}}{\mathbb{P}(X \in A)}. \tag{32}$$

Proof. Since X admits the density f_X, for any Borel set $B \subseteq \mathbb{R}$ we have

$$\mathbb{P}(X \in B) = \int_B f_X(x)dx. \tag{33}$$

By definition of conditional probability, whenever $\mathbb{P}(X \in A) > 0$,

$$\mathbb{P}(X \in B \mid X \in A) = \frac{\mathbb{P}(X \in B \cap A)}{\mathbb{P}(X \in A)}. \tag{34}$$

Using the density representation of the numerator gives

$$\mathbb{P}(X \in B \mid X \in A) = \frac{\int_{B\cap A} f_X(x)dx}{\mathbb{P}(X \in A)}. \tag{35}$$

Define

$$g(x) = \frac{f_X(x)\mathbb{1}\{x \in A\}}{\mathbb{P}(X \in A)} \quad (x \in \mathbb{R}). \tag{36}$$

Then for every Borel set B

$$\int_B g(x)dx = \frac{1}{\mathbb{P}(X \in A)} \int_{B \cap A} f_X(x)dx = \mathbb{P}(X \in B \mid X \in A). \tag{37}$$

In particular, choosing $B = \mathbb{R}$ yields $\int_{\mathbb{R}} g(x)dx = 1$, so g is a valid probability density. Hence g is a density that realizes the conditional probabilities, i.e. $g = f_{X|X\in A}$. □

After proving Prop. B.5, we can prove that

$$\log \mathbb{E}[\exp(X) \mid X < c] = \frac{\sigma}{2} - \log \Phi(c/\sigma) + \log \Phi(\frac{c - \sigma^2}{\sigma}). \tag{38}$$

Proof.

$$\begin{aligned}
&\mathbb{E}[\exp(X) \mid X < c] \\
=&\int_{-\infty}^{c} \exp(x) \frac{f_X(x)}{\mathbb{P}(X < c)} dx \\
=&\frac{1}{\Phi(c/\sigma)} \int_{-\infty}^{c} \exp(x) \frac{1}{\sqrt{2\pi\sigma^2}} \exp\left(-\frac{x^2}{2\sigma^2}\right) dx \\
=&\frac{1}{\Phi(c/\sigma)} \frac{1}{\sqrt{2\pi\sigma^2}} \int_{-\infty}^{c} \exp\left(-\frac{x^2}{2\sigma^2} + x\right) dx \\
=&\frac{1}{\Phi(c/\sigma)} \frac{1}{\sqrt{2\pi\sigma^2}} \int_{-\infty}^{c} \exp\left(-\frac{1}{2\sigma^2}(x^2 - 2\sigma^2 x + \sigma^4 - \sigma^4)\right) dx \\
=&\frac{\exp(\sigma^2/2)}{\Phi(c/\sigma)} \frac{1}{\sqrt{2\pi\sigma^2}} \int_{-\infty}^{c} \exp\left(-\frac{1}{2\sigma^2}(x - \sigma^2)^2\right) dx \\
=&\frac{\exp(\sigma^2/2)}{\Phi(c/\sigma)} \Phi\left(\frac{c - \sigma^2}{\sigma}\right)
\end{aligned}$$

□

Applying a log on both sides yields

$$\log \mathbb{E}[\exp(X) \mid X < c] = \frac{\sigma}{2} - \log \Phi(c/\sigma) + \log \Phi(\frac{c - \sigma^2}{\sigma}), \tag{39}$$

which is the expression in (11).

B.7 Series Representation of $\mathcal{T}$ and $\partial_t \mathcal{T}$

We begin by station the Series expansion for $\mathcal{T}$ (Prop. B.6) and its time-derivative $\partial_t \mathcal{T}$ (Prop. B.7):

Proposition B.6 (Series Expansion of the Diffusion Transformation Operator)**.** *The diffusion transformation operator $\mathcal{T}$ can be expressed as:*

$$\mathcal{T}(\tilde{\alpha}_t) = \frac{K}{K-1} \left[e^{-\nu_t^2/2} \sum_{n=0}^{\infty} \frac{\nu_t^n}{n!} M_n - \frac{1}{K} \right] \tag{40}$$

$\nu_t = \frac{\tilde{\alpha}_t}{\sqrt{1-\tilde{\alpha}_t^2}}$ and $M_n = \int_{-\infty}^{\infty} z^n \phi(z) \Phi^{K-1}(z) dz$.

Proposition B.7 (Time-Derivative of the Diffusion Transformation Operator)**.** *The time-derivative of the diffusion transformation operator $\mathcal{T}$ can be expressed as:*

$$\frac{d}{dt}\mathcal{T}(\tilde{\alpha}_t) = \frac{K \cdot e^{-\nu_t^2/2}}{K-1} \frac{\tilde{\alpha}_t'}{(1-\tilde{\alpha}_t)^{3/2}} \sum_{n=0}^{\infty} \frac{\nu_t^n}{n!} \left[I_n - \nu_t M_n \right] \tag{41}$$

where ν_t and M_n are defined as in Prop. B.6. Finally, $I_n = \int_{-\infty}^{\infty} z^{n+1} \phi(z) \Phi^{K-1}(z) dz$, and $\tilde{\alpha}_t'$ denotes the time-derivative of the Gaussian noise schedule $\tilde{\alpha}_t$.

At this point, one might ask what is gained by expressing $\mathcal{T}$ as a series expansion. There are two key advantages. First, since $\mathcal{T}$ is intractable, Sahoo et al. (2024) resort to precomputing 100k evaluations, which can take up to two hours with the GPT-2 tokenizer. Second, they approximate the time derivative using finite differences. Crucially, observe that M_n and I_n in Prop. B.6 and B.7 are the only intractable components of the series expansion, and they are independent of the input $\tilde{\alpha}_t$. We find that the terms of the series decay to zero after roughly 150 terms (with slower decay as $t \to 1$). Thus, instead of pre-computing 100k evaluations of $\mathcal{T}$, it suffices to cache M_n and I_n for $n < 150$. In practice, this takes only a few seconds and can be performed at the start of training. We now prove Prop. B.6 and B.6.

B.7.1 Proof of Proposition B.6

To prove the result, we rely on the following proposition:

Proposition B.8 (Corollary of the Dominated Convergence Theorem)**.** *If the sum $\sum_{n=0}^{\infty} f_n(x)$ exists for all x and there exists an integrable function $g(x)$ such that*

$$\left| \sum_{n=0}^{k} f_n(x) \right| \leq g(x) \tag{42}$$

for all k, then

$$\int_{-\infty}^{\infty} \sum_{n=0}^{\infty} f_n(x) dx = \sum_{n=0}^{\infty} \int_{-\infty}^{\infty} f_n(x) dx. \tag{43}$$

We now prove Prop. B.6 using Prop. B.8:

Proof. Recall that the standard Gaussian PDF is given by

$$\phi(x) = \frac{1}{\sqrt{2\pi}} e^{-x^2/2}. \tag{44}$$

For notational convenience, let $\nu_t = \frac{\tilde{\alpha_t}}{\sqrt{1-\tilde{\alpha_t}^2}}$. We can rewrite $\phi(x - \nu_t)$ in terms of $\phi(x)$:

$$\phi(x - \nu_t) = \frac{1}{\sqrt{2\pi}} e^{-(x-\nu_t)^2/2} = \frac{1}{\sqrt{2\pi}} e^{-(x^2 - 2\nu_t x + \nu_t^2)/2} = \phi(x) e^{\nu_t x} e^{-\nu_t^2/2}. \tag{45}$$

Using the definition of the infinite series of e^x, we can expand $e^{\nu_t x}$:

$$\phi(x - \nu_t) = \phi(x) e^{-\nu_t^2/2} \sum_{n=0}^{\infty} \frac{\nu_t^n x^n}{n!}. \tag{46}$$

Substituting this into our original integral:

$$\int_{-\infty}^{\infty} \phi\left(z - \nu_t\right) \Phi^{K-1}(z) dz = \int_{-\infty}^{\infty} \phi(z) e^{-\nu_t^2/2} \sum_{n=0}^{\infty} \frac{\nu_t^n z^n}{n!} \Phi^{K-1}(z) dz \tag{47}$$

Since Prop. B.8 is satisfied, as the sum is the Taylor series of the exponential function, we can exchange the order of integration and summation. This leads to our final result:

$$\begin{aligned} \int_{-\infty}^{\infty} \phi\left(z - \nu_t\right) \Phi^{K-1}(z) dz &= e^{-\nu_t^2/2} \sum_{n=0}^{\infty} \frac{\nu_t^n}{n!} \int_{-\infty}^{\infty} z^n \phi(z) \Phi^{K-1}(z) dz \\ &= e^{-\nu_t^2/2} \sum_{n=0}^{\infty} \frac{\nu_t^n}{n!} M_n. \end{aligned} \tag{48}$$

□

B.7.2 Proof of Prop. B.7

Once again, we need to exchange the order of operations to prove Prop. B.7, which relies on Prop. B.9:

Proposition B.9 (Second Corollary of the Dominated Convergence Theorem)**.** *Let $f(x,t)$ be differentiable in t and suppose there exists a function $g(x,t)$ such that:*

1. $\left|\frac{\partial f(x,t)}{\partial t}\right| \leq g(x,t_0)$ *for all x and t in some neighborhood $|t-t_0| \leq \delta_0$*
2. $\int_{-\infty}^{\infty} g(x,t)dx < \infty$ *for all t*

Then

$$\frac{d}{dt}\int_{-\infty}^{\infty} f(x,t)dx = \int_{-\infty}^{\infty} \frac{\partial f(x,t)}{\partial t}dx \tag{49}$$

In our case, we have

$$f(x,t) = \phi\left(z - \frac{\tilde{\alpha}_t}{\sqrt{1-\tilde{\alpha}_t^2}}\right)\Phi^{K-1}(z) = \phi\left(z-\nu_t\right)\Phi^{K-1}(z) \tag{50}$$

which has time derivative

$$\frac{(z-\nu_t)\phi(z-\nu_t)}{(1-\alpha_t^2)^{3/2}}\Phi^{K-1}(z). \tag{51}$$

Therefore, we need to find a suitable function g that satisfies Prop. B.9 to justify swapping the order of integration and differentiation.

Proof. Let $1 > \delta_0 > 0$ and choose $t_0 = \frac{1-\delta_0}{2}$. When $|t-t_0| \leq \delta_0$, we have $t \in [t_0-\delta_0, t_0+\delta_0]$. Since $t_0 - \delta_0 < t_0 < 1$ and $t_0 + \delta_0 = \frac{1-\delta_0}{2} + \delta_0 < 1$, we are guaranteed that $t < 1$. This ensures that ν_t is finite. Because $\alpha_t \in [0,1)$ when $t < 1$, there exist a constant C, such that

$$C := \max_{|t-t_0|\leq\delta_0} \frac{1}{(1-\alpha_t^2)^{3/2}} < \infty. \tag{52}$$

For $z \in \mathbb{R}$ and $|t-t_0| \leq \delta_0$, we can bound the absolute value of the time derivative of f as follows:

$$\begin{aligned}\left|\frac{\partial f(z,t)}{\partial t}\right| &= \frac{|z-\nu_t|}{(1-\alpha_t^2)^{3/2}}\phi(z-\nu_t)\,\Phi^{K-1}(z) \\ &\leq C|z-\nu_t|\phi(z-\nu_t) = g(z,t).\end{aligned}$$

Finally, for all $t \in [0,1)$:

$$\begin{aligned}\int_{-\infty}^{\infty} g(z,t)dz &= C\int_{-\infty}^{\infty}|z-\nu_t|\phi(z-\nu_t)dz = C\int_{-\infty}^{\infty}|z|\phi(z)dz \\ &= C\int_{-\infty}^{\infty}|z|\phi(z)dz = 2C\int_0^{\infty} z\phi(z)dz \\ &= 2C\int_0^{\infty} z\cdot\frac{1}{\sqrt{2\pi}}e^{-z^2/2}dz \\ &= \frac{2C}{\sqrt{2\pi}}\int_0^{\infty} ze^{-z^2/2}dz \\ &= \frac{2C}{\sqrt{2\pi}}\cdot 1 = C\sqrt{\frac{2}{\pi}} < \infty,\end{aligned} \tag{53}$$

where we used the substitution $u = z^2/2$ in the integral $\int_0^{\infty} ze^{-z^2/2}dz$ to obtain $\int_0^{\infty} e^{-u}du = 1$. □

We can now prove Proposition B.7

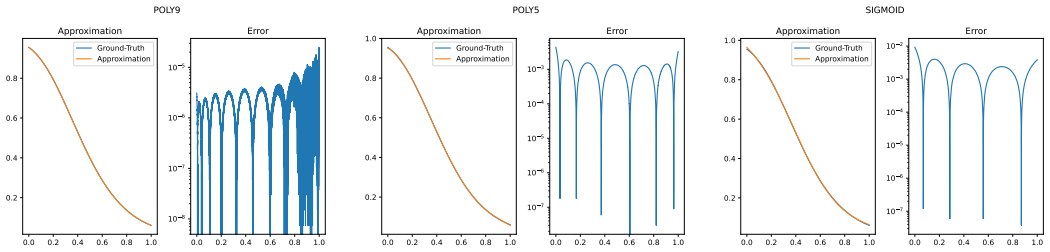

Figure 4: Polynomial approximation and approximation error, compared to the series approximation, truncated at 150 terms. The degree-9 polynomial (left) achieves orders of magnitude lower error than the degree-5 polynomial (center) and sigmoid (right) approximations.

Proof. We want to compute

$$\frac{d}{d\nu_t}\mathcal{T}(\alpha_t) = \frac{K}{K-1}\frac{d}{d\nu_t}\int \phi(z-\nu_t)\Phi^{K-1}(z)dz \tag{54}$$

Applying the derivative under the integral sign and using the identity $\phi(z-\nu_t) = \phi(z)e^{\nu_t z}e^{-\nu_t^2/2}$, we have:

$$\begin{aligned}\frac{d}{d\nu_t}\phi(z-\nu_t) &= \phi(z)\frac{d}{d\nu_t}[e^{\nu_t z-\nu_t^2/2}] \\ &= \phi(z)e^{\nu_t z-\nu_t^2/2}(z-\nu_t) \\ &= (z-\nu_t)\phi(z-\nu_t)\end{aligned} \tag{55}$$

Therefore:

$$\frac{d}{d\nu_t}\mathcal{T}(\alpha_t) = \frac{K}{K-1}\int_{-\infty}^{\infty}(z-\nu_t)\phi(z-\nu_t)\Phi^{K-1}(z)dz \tag{56}$$

Now using the Taylor series of $\phi(z-\nu_t)$, found earlier, and inverting the sum and integral as before, we find

$$\begin{aligned}\frac{d}{d\nu_t}\mathcal{T}(\alpha_t) &= \frac{K}{K-1}\int_{-\infty}^{\infty}(z-\nu_t)\phi(z)e^{\nu_t z}e^{-\nu_t^2/2}\Phi^{K-1}(z)dz \\ &= \frac{K\cdot e^{-\nu_t^2/2}}{K-1}\sum_{n=0}^{\infty}\frac{\nu_t^n}{n!}\left[\int_{-\infty}^{\infty}z^{n+1}\phi(z)\Phi^{K-1}(z)dz - \nu_t\int_{-\infty}^{\infty}z^n\phi(z)\Phi^{K-1}(z)dz\right] \\ &= \frac{K\cdot e^{-\nu_t^2/2}}{K-1}\sum_{n=0}^{\infty}\frac{\nu_t^n}{n!}\left[I_n - \nu_t M_n\right].\end{aligned} \tag{57}$$

where $I_n = \int_{-\infty}^{\infty}z^{n+1}\phi(z)\Phi^{K-1}(z)dz$ and $M_n = \int_{-\infty}^{\infty}z^n\phi(z)\Phi^{K-1}(z)dz$.

This expansion allows us to compute the derivative of the diffusion transformation operator with respect to ν_t in terms of moments of the standard normal distribution weighted by powers of the CDF. □

B.8 Polynomial Approximation of $\mathcal{T}$

Because the Diffusion Transformation Operator $\mathcal{T}$ has a sigmoid-like shape, we approximating it with S-shaped functions that require only a handful of coefficients. This allows us to store fewer parameters during training, instead of the 100k values required by the original curriculum or the 300 coefficients from the series approximation. Concretely, we test several functional forms with fewer than 10 parameters and fit them using non-linear least squares, via `scipy.optimize.curve_fit`.

As shown in Figure 4, approximations tend to be less accurate at the boundaries, when $t \approx 0$ or $t \approx 1$. We find that the degree-9 polynomial works better than a sigmoid function of the form $a\sigma(bt+c)+d$, especially at the boundaries.

C Experimental Details

C.1 Superposition Sampler

OpenWebText To evaluate the samplers, we use pre-trained checkpoints for MDLM (Sahoo et al., 2024) and Duo (Sahoo et al., 2025a) (distilled with *discrete consistency distillation*). We re-state their experimental settings in Suppl. C.2.1. For ReMDM, we use the official implementation of Wang et al. (2025), with a few of their best hyperparameter settings and plot the best-performing one. Note that Wang et al. (2025) use nucleus sampling (Holtzman et al., 2020), whereas we do not need to use it for Ψ-samplers with Duo. See Suppl. D.1 for details on selecting κ_t.

CIFAR10 We adopt the same U-Net backbone Ronneberger et al. (2015) used by Austin et al. (2023); Schiff et al. (2025) for both MDLM and Duo experiments. The architectures for MDLM and Duo are identical, except that Duo incorporates time-conditioning as in Austin et al. (2023); Schiff et al. (2025) (Table 3). Furthermore, the backbone uses a truncated logistic transformation of the output (Salimans et al., 2017). To enable classifier-free guidance, we use a label embedding which is added to the time embedding, as in Nichol & Dhariwal (2021). See Suppl. D.1 for details on selecting κ_t.

Table 3: Model architecture on CIFAR10

Component	Value
Vocab size	256
Number of ResNet blocks per scale	2
Base channels	128
Channel multiplier per scale	(1,2,2,2)
Attention resolutions	16
Conditional embedding dimension	128
Number of parameters	35.8M

C.2 Improved Curriculum

C.2.1 Language modeling

We adopt the same setup as prior work on discrete diffusion (Lou et al., 2024; Sahoo et al., 2024; 2025a), but restate it for completeness.

LM1B We detokenize the the One Billion Words (Chelba et al., 2014) as in Lou et al. (2024); Sahoo et al. (2024)[1], and tokenize it using the `bert-base-uncased` tokenizer (Devlin et al., 2019), as He et al. (2022). We use a context length of 128 and pad shorter documents.

OpenWebText We tokenize OpenWebText (Gokaslan & Cohen, 2019) with the `GPT-2` tokenizer, concatenate sequences to a length of 1024, and insert an `eos` token between documents. Since the dataset lacks an official validation split, we reserve the last 100k documents for validation.

Backbone We parameterize all models using the modified diffusion transformer architecture of Peebles & Xie (2023), following Lou et al. (2024); Sahoo et al. (2024). Our models use 12 layers, a hidden dimension of 768, 12 attention heads, and a timestep embedding of size 128 for the uniform-state diffusion variants. Word embeddings are not tied between input and output.

Curriculum Lookup For the Duo baseline, we train models using the original code. To implement the efficient curriculum, we replace the full linear combination of embeddings by a sparse lookup, implemented using `torch.nn.functional.embedding_bag` to avoid materializing intermediate tensors. The curriculum phase lasts for the first 500k steps, after which we perform regular embedding table lookups, just like Sahoo et al. (2025a).

Optimization We train all models with the AdamW optimizer (Loshchilov & Hutter, 2019) using a batch size of 512. The learning rate is linearly warmed up from 0 to 3×10^{-4} over 2,500 steps, then kept constant for the remainder of training. We apply a dropout rate of 0.1 throughout.

[1] https://github.com/louaaron/Score-Entropy-Discrete-Diffusion/blob/main/data.py

C.3 Downstream Evaluation Protocol

We evaluate downstream performance using the `lm-eval-harness` library (Gao et al., 2024), following the protocol of Deschenaux et al. (2025). We focus on multiple choice tasks, where the log-likelihood of each candidate answer, given a prompt, is computed and the answer with the highest score is selected. For diffusion language models, which optimize a variational bound on the log-likelihood of the full sequence, we adapt the evaluation by using Bayes' rule:

$$\log p(\mathbf{y}_i|\mathbf{x}) = \log p(\mathbf{x}, \mathbf{y}_i) - \log p(\mathbf{x}) \propto \log p(\mathbf{x}, \mathbf{y}_i), \tag{58}$$

Since $\log p(\mathbf{x})$ does not depend on the candidate $\mathbf{y}_i$, we simply select the answer that maximizes $\log p(\mathbf{x}, \mathbf{y}_i)$. In practice, we use the log-likelihood ELBO (4), estimated via Monte Carlo with 1024 samples, and choose the continuation $\mathbf{y}_i$ with the highest estimated likelihood.

C.4 Zero-Shot Likelihood

Our setting is the same as used by Sahoo et al. (2025a). Specifically, we measure the likelihood of the models trained on OpenWebText using the validation splits of seven diverse datasets: Penn Tree Bank (PTB; Marcus et al. (1993)), Wikitext (Merity et al., 2016), One Billion Words (LM1B; Chelba et al. (2014)), Lambada (Paperno et al., 2016), AG News (Zhang et al., 2016), and Scientific Papers (Pubmed and Arxiv subsets; Cohan et al. (2018)). The datasets are detokenized following the protocol of Lou et al. (2024); Sahoo et al. (2025a). We wrap all sequences to a maximum length of 1024 tokens and do not insert `eos` tokens between them. Table 5 shows that we reach similar performance as Duo.

D Additional Experimental Results

In Suppl. D.1, we elaborate on the impact of κ_t on the performance of the Ψ-samplers. In Suppl. D.2, we show that our efficient curriculum produces weights with the same marginal distributions as Sahoo et al. (2025a).

D.1 Tuning κ_t for the Ψ-samplers

As discussed in Sec. 5.1, the choice of κ_t is crucial for achieving strong performance with Ψ-samplers. Notably, poor choice of κ_t underperform ancestral sampling. Therefore, we report our hyperparameter sweeps in this section, for varying number of step sizes.

CIFAR10 We report the FID in Table 6, and IS in Table 7. To reduce computational cost, we excluded certain step size and hyperparameter combinations, especially at larger number of steps, based on the FID at lower number of steps. In Figure 1 and 3, we show the best FID/IS achieved at each number of steps.

OpenWebText We report generation perplexity (Gen. PPL) and entropy for Duo without distillation (Table 8) and with distillation (Table 9), using checkpoints released by Sahoo et al. (2025a). Results for MDLM+ReMDM are shown in Table 10, and we sample using the official implementation of Wang et al. (2025). For plotting, we selected hyperparameters that yield entropy closest to the posterior samplers. Notably, unlike CIFAR-10, we plot with the same hyperparameters across all sampling steps, although they need not be identical across Duo, Distilled Duo, and MDLM.

D.2 Distribution of the top k entries of the softmax

To verify that our sparse implementation accurately approximates the curriculum weights of Sahoo et al. (2025a), we compare the empirical distributions of the top-k largest entries between the original and our efficient implementation. While matching marginal distributions does not guarantee matching joint distributions, matching marginals are necessary for matching joints, and are easier to visualize. Recall that experimentally, our efficient implementation is sufficient to achieve strong performance (Sec. 5.2). Specifically, we show histograms using a tokenizer with $100k$ tokens in Figures 5, 6, 7, 8, and with the `GPT-2` tokenizer in Figures 9, 10, 11, 12, with varying temperature and log signal-to-noise ratios. In all cases, the top k variables have matching distributions.

Table 4: Training efficiency comparison between Duo and Duo^{++} on 138M parameter models. All measurements are conducted on a training job on 8 NVIDIA GH200-120GB GPU with batch size 32. We report the average throughput in sequence per second. The row "Duo (afer CL)" denotes the resources consumption of Duo after the Curriculum phase. The impact of k is minimal when $k \in \{2, 3, 5\}$, and Duo^{++} uses similar resources.

Method	**Throughput (samples/s)** $\uparrow$	**Peak Memory (GiB)** $\downarrow$
Duo	81.8	94.3
Duo (after the CL)	122.4	63.3
Duo^{++} ($k \in \{2, 3, 5\}$)	121.9	63.38

D.3 Training Efficiency of Our Fast Curriculum

As shown in Table 4, our sparse curriculum achieves a 33% reduction in peak memory usage and reaches an average throughput 25% higher than Duo, at a context length of 1024.

Table 5: Zero-shot perplexity (PPL) on seven datasets. Lower is better. †Results taken from Sahoo et al. (2025a). Duo^{++} ($k = 2$) achieves a slightly lower zero-shot perplexity than Duo on 6 of 7 datasets.

	PTB	Wiki	LM1B	LBD	AG News	PubMed	ArXiv
Autoregressive							
Transformer†	82.05	25.75	51.25	51.28	52.09	49.01	41.73
Diffusion (138M)							
SEDD Uniform†	105.51	41.10	82.62	57.29	82.64	55.89	50.86
UDLM†	112.82	39.42	77.59	53.57	80.96	50.98	44.08
Duo†	**89.35**	**33.57**	73.86	49.78	67.81	44.48	40.39
Duo^{++} ($k = 2$)	94.96	34.05	**73.80**	**48.67**	67.14	**43.98**	**38.93**
Duo^{++} ($k = 3$)	91.94	34.65	74.16	49.89	**66.89**	44.87	40.42
Duo^{++} ($k = 5$)	94.46	34.52	74.91	50.93	68.72	46.79	41.04

Algorithm 2 Reverse Sampling from Order Statistics of Gaussian Random Variables

Input Number of variables N, standard deviation σ, number of top values k
Sample $U_\ell \sim \mathcal{U}(0, 1)$, for $N \geq \ell \geq N - k + 1$
Compute the random variables: $R_\ell = \frac{\log U_\ell}{\ell}$
Compute the cumulative sums: $P_\ell = \sum_{m=\ell}^{N} R_m$
Let $V_\ell = \exp(P_\ell)$, the ℓ-th sample from the (uniform) order statistic.
Apply inverse normal CDF: $X^{(\ell)} = \Phi^{-1}(V_\ell) \cdot \sigma$
return $\{X^{(\ell)}\}_{\ell=N}^{N-k+1}$

Algorithm 3 Floyd's Algorithm for Sampling Without Repetition

Input Number of possible values N, number of samples k.
Initialize array S of size k to store samples
for $t = 0$ to $k - 1$ **do**
  Sample $j \sim$ Randint$(0, N - k + t)$
  if $t > 0$ and j appears in $S[0 : t]$ **then**
    $S[t] \leftarrow N - k + t$ {Use largest remaining value}
  else
    $S[t] \leftarrow j$
  end if
end for
return S

Table 6: FID scores across different numbers of sampling steps for various hyperparameter ablations. Lower is better. The section "Ψ-samplers Loop" denote the ReMDM-inspired scheduler, where t is linearly decreased to $\alpha_{t_{\text{on}}}$ (from $t = 1$ to $t = t_{\text{on}}$), then kept constant until t_{off}. The section "Ψ-samplers Linear" denote the Linear scheduler, where t linearly decreases, like during ancestral sampling. We omit certain settings (denoted by –), to spare compute costs, as each cell FID requires generating 50k samples.

	Number of steps						
	32	128	256	512	1024	2048	4096
Uniform Diffusion (Ancestral)							
Duo (log-lin.)	85.3	56.7	52.6	50.9	49.7	49.2	49.0
Duo (cosine)	77.7	51.9	47.6	45.9	45.0	44.6	44.3
+Greedy	64.9	47.5	44.2	42.8	42.2	41.8	41.6
+Guid. ($\gamma = 1$)	57.3	41.6	39.4	37.9	37.6	37.2	36.9
+temp. $T = 0.8$	**39.2**	**27.6**	**26.3**	**25.4**	**25.4**	**25.0**	**25.3**
Uniform Diffusion (Ψ-samplers Loop)							
$\alpha_{t_{\text{on}}} = 0.85, t_{\text{off}} = t_{\text{on}} + 0.05, \kappa = 0.02$	**40.4**	27.5	25.9	24.9	24.3	23.8	**25.1**
$\alpha_{t_{\text{on}}} = 0.45, t_{\text{off}} = t_{\text{on}} + 0.05, \kappa = 0.02$	45.0	29.3	27.6	28.2	33.4	–	–
$\alpha_{t_{\text{on}}} = 0.1, t_{\text{off}} = t_{\text{on}} + 0.05, \kappa = 0.02$	43.0	31.8	41.7	67.2	129.8	–	–
$\alpha_{t_{\text{on}}} = 0.8, t_{\text{off}} = t_{\text{on}} + 0.1, \kappa = 0.02$	40.9	27.2	25.3	24.0	**23.7**	28.3	52.5
$\alpha_{t_{\text{on}}} = 0.8, t_{\text{off}} = t_{\text{on}} + 0.1, \kappa = 0.5$	41.0	27.3	25.6	24.6	**23.7**	**23.4**	27.9
$\alpha_{t_{\text{on}}} = 0.7, t_{\text{off}} = t_{\text{on}} + 0.2, \kappa = 0.5$	43.2	**26.2**	**23.6**	**22.4**	25.0	–	–
Uniform Diffusion (Ψ-samplers Linear)							
$\alpha_{t_{\text{on}}} = 0.85, t_{\text{off}} = t_{\text{on}} + 0.05, \kappa = 0.02$	39.1	27.4	25.7	24.5	23.8	23.8	**28.2**
$\alpha_{t_{\text{on}}} = 0.45, t_{\text{off}} = t_{\text{on}} + 0.05, \kappa = 0.02$	41.9	29.1	27.8	34.6	61.4	–	–
$\alpha_{t_{\text{on}}} = 0.1, t_{\text{off}} = t_{\text{on}} + 0.05, \kappa = 0.02$	39.4	27.8	26.5	25.7	25.4	–	–
$\alpha_{t_{\text{on}}} = 0.8, t_{\text{off}} = t_{\text{on}} + 0.1, \kappa = 0.5$	38.9	26.9	25.2	23.9	23.1	23.4	31.6
$t_{\text{on}} = 0.3, t_{\text{off}} = 0.1\ \kappa = 0.75$	38.7	26.2	24.2	22.5	22.0	25.4	43.9
$t_{\text{on}} = 0.4, t_{\text{off}} = 0.1\ \kappa = 0.9$	38.7	25.7	23.4	21.5	20.9	23.4	37.3
$t_{\text{on}} = 0.5, t_{\text{off}} = 0.1\ \kappa = 0.95$	38.6	25.2	22.7	20.7	**20.2**	22.6	35.4
$t_{\text{on}} = 0.6, t_{\text{off}} = 0.1\ \kappa = 0.95$	**38.5**	**24.2**	**21.4**	**20.0**	22.3	32.7	59.0
$t_{\text{on}} = 0.6, t_{\text{off}} = 0.1\ \kappa = 0.98$	38.8	25.9	23.4	21.3	**20.2**	**21.3**	28.7
Masked Diffusion (Ancestral)							
MDLM (cosine)	104.2	51.9	46.7	45.1	44.5	45.3	48.0
MDLM (log-lin. / cosine)	**81.8**	48.0	40.0	39.3	37.8	38.0	38.7
MDLM (log-lin.)	208.3	74.2	48.4	38.0	34.2	33.3	33.1
+Greedy	208.3	74.2	48.4	38.1	34.2	33.3	33.1
+Guid. ($\gamma = 1$)	198.6	62.9	41.8	33.2	29.5	28.1	27.6
+temp. $T = 0.8$	126.2	**33.2**	**25.1**	**24.0**	**24.7**	**25.7**	**26.6**
Masked Diffusion (Ψ-samplers Linear)							
$t_{\text{on}} = 0.3, t_{\text{off}} = 0.1\ \kappa = 0.75$	125.5	33.3	25.2	24.1	24.9	26.2	28.5
$t_{\text{on}} = 0.5, t_{\text{off}} = 0.1\ \kappa = 0.95$	125.5	33.1	25.2	24.2	25.2	–	–
$t_{\text{on}} = 0.6, t_{\text{off}} = 0.1\ \kappa = 0.95$	**125.2**	**32.9**	**24.8**	**23.8**	25.0	27.3	31.9
$t_{\text{on}} = 0.6, t_{\text{off}} = 0.1\ \kappa = 0.98$	125.7	33.1	25.0	24.0	25.0	–	–
$t_{\text{on}} = 0.85, t_{\text{off}} = 0.8\ \kappa = 0.02$	183.7	79.2	88.8	113.1	138.0	–	–
$t_{\text{on}} = 0.45, t_{\text{off}} = 0.4\ \kappa = 0.02$	130.9	39.8	37.3	43.1	55.9	–	–
$t_{\text{on}} = 0.15, t_{\text{off}} = 0.1\ \kappa = 0.02$	125.9	33.2	25.1	24.0	**24.7**	**25.6**	**26.9**

Table 7: Inception Score (IS) across different numbers of sampling steps for various hyperparameter ablations. Higher is better. The section "Ψ-samplers Loop" denote the ReMDM-inspired scheduler, where t is linearly decreased to $\alpha_{t_{\text{on}}}$ (from $t = 1$ to $t = t_{\text{on}}$), then kept constant until t_{off}. The section "Ψ-samplers Linear" denote the Linear scheduler, where t linearly decreases, like during ancestral sampling. We omit certain settings (denoted by –), to spare compute costs, as each cell FID requires generating 50k samples. The decision to omit entries was based on the FID achieved with fewer steps.

	Number of steps						
	32	128	256	512	1024	2048	4096
Uniform Diffusion (Ancestral)							
Duo (log-lin.)	4.8	5.7	5.8	5.8	5.9	5.9	6.0
Duo (cosine)	5.3	6.1	6.5	6.4	6.4	6.5	6.4
+Greedy	5.6	6.3	6.3	6.5	6.5	6.6	6.5
+Guid. ($\gamma = 1$)	6.3	6.9	7.0	7.0	7.1	7.1	7.1
+temp. $T = 0.8$	**7.0**	**7.5**	**7.5**	**7.5**	**7.6**	**7.5**	**7.6**
Uniform Diffusion (Ψ-samplers Loop)							
$\alpha_{t_{\text{on}}} = 0.85, t_{\text{off}} = t_{\text{on}} - 0.05, \kappa = 0.02$	6.9	7.5	7.5	7.6	7.6	**7.6**	**7.3**
$\alpha_{t_{\text{on}}} = 0.45, t_{\text{off}} = t_{\text{on}} - 0.05, \kappa = 0.02$	6.9	**7.8**	**8.0**	**8.1**	**8.2**	–	–
$\alpha_{t_{\text{on}}} = 0.1, t_{\text{off}} = t_{\text{on}} - 0.05, \kappa = 0.02$	6.9	7.7	7.4	6.2	4.1	–	–
$\alpha_{t_{\text{on}}} = 0.8, t_{\text{off}} = t_{\text{on}} - 0.1, \kappa = 0.02$	**7.0**	7.5	7.6	7.6	7.6	7.2	6.1
$\alpha_{t_{\text{on}}} = 0.8, t_{\text{off}} = t_{\text{on}} - 0.1, \kappa = 0.5$	6.9	7.5	7.5	7.5	7.5	7.5	7.0
$\alpha_{t_{\text{on}}} = 0.7, t_{\text{off}} = t_{\text{on}} - 0.2, \kappa = 0.5$	**7.0**	7.6	7.6	7.7	7.4	–	–
Uniform Diffusion (Ψ-samplers Linear)							
$\alpha_{t_{\text{on}}} = 0.85, t_{\text{off}} = t_{\text{on}} - 0.05, \kappa = 0.02$	7.0	7.5	7.6	7.7	7.7	7.6	7.3
$\alpha_{t_{\text{on}}} = 0.45, t_{\text{off}} = t_{\text{on}} - 0.05, \kappa = 0.02$	**7.1**	**8.0**	**8.4**	**8.7**	7.9	–	–
$\alpha_{t_{\text{on}}} = 0.1, t_{\text{off}} = t_{\text{on}} - 0.05, \kappa = 0.02$	7.0	7.4	7.5	7.6	7.6	–	–
$\alpha_{t_{\text{on}}} = 0.8, t_{\text{off}} = t_{\text{on}} - 0.1, \kappa = 0.5$	7.0	7.6	7.6	7.6	7.7	7.5	7.0
$t_{\text{on}} = 0.3, t_{\text{off}} = 0.1\ \kappa = 0.75$	**7.1**	7.6	7.6	7.7	7.6	7.4	6.4
$t_{\text{on}} = 0.4, t_{\text{off}} = 0.1\ \kappa = 0.9$	**7.1**	7.6	7.7	7.7	7.7	7.7	7.0
$t_{\text{on}} = 0.5, t_{\text{off}} = 0.1\ \kappa = 0.95$	**7.1**	7.7	7.7	7.9	8.0	7.8	7.5
$t_{\text{on}} = 0.6, t_{\text{off}} = 0.1\ \kappa = 0.95$	**7.1**	7.8	8.0	8.2	**8.2**	8.0	6.9
$t_{\text{on}} = 0.6, t_{\text{off}} = 0.1\ \kappa = 0.98$	**7.1**	7.6	7.7	7.9	8.1	**8.2**	**8.1**
Masked Diffusion (Ancestral)							
MDLM (cosine)	4.3	5.9	6.1	6.2	6.3	6.2	6.2
MDLM (log-lin. / cosine)	**5.2**	**6.7**	7.0	7.1	7.1	7.0	6.8
MDLM (log-lin.)	2.7	4.9	5.7	6.3	6.6	6.6	6.7
+Greedy	2.7	4.9	5.7	6.2	6.6	6.6	6.7
+Guid. ($\gamma = 1$)	3.0	5.8	6.5	6.8	7.1	7.2	7.2
+temp. $T = 0.8$	4.4	**6.7**	**7.2**	**7.3**	**7.3**	**7.3**	**7.3**
Masked Diffusion (Ψ-samplers Linear)							
$t_{\text{on}} = 0.3, t_{\text{off}} = 0.1\ \kappa = 0.75$	**4.4**	**6.7**	7.1	**7.3**	7.2	7.2	6.9
$t_{\text{on}} = 0.5, t_{\text{off}} = 0.1\ \kappa = 0.95$	**4.4**	**6.7**	7.1	7.2	**7.3**	–	–
$t_{\text{on}} = 0.6, t_{\text{off}} = 0.1\ \kappa = 0.95$	**4.4**	**6.7**	**7.2**	7.3	7.2	7.2	**7.1**
$t_{\text{on}} = 0.6, t_{\text{off}} = 0.1\ \kappa = 0.98$	**4.4**	**6.7**	**7.2**	**7.3**	**7.3**	–	–
$t_{\text{on}} = 0.85, t_{\text{off}} = 0.8\ \kappa = 0.02$	3.2	5.2	4.9	4.4	4.3	–	–
$t_{\text{on}} = 0.45, t_{\text{off}} = 0.4\ \kappa = 0.02$	4.3	**6.7**	**7.2**	7.2	6.5	–	–
$t_{\text{on}} = 0.15, t_{\text{off}} = 0.1\ \kappa = 0.02$	**4.4**	**6.7**	**7.2**	**7.3**	**7.3**	**7.3**	**7.1**

Table 8: Generative perplexity (Gen. PPL, ↓) and entropy (↑) of **Duo** (without distillation) for different numbers of sampling steps: 32, 128, 256, 512, 1024, and 4096. Each column contains two sub-columns: Gen. PPL and Entropy. The Grayed row reports the configuration that achieves the best Gen. PPL while keeping entropy closest to the ancestral sampler.

	32		128		256		512		1024		4096	
	PPL	H	PPL	H	PPL	H	PPL	H	PPL	H	PPL	H
Ancestral sampler	97.0	5.5	81.2	5.5	79.7	5.5	77.3	5.5	75.8	5.5	76.8	5.5
$t_{\text{on}} = 0.5, t_{\text{off}} = 0.1,$ $\kappa_t = 0.95$	94.3	5.5	75.5	5.5	70.6	5.5	64.6	5.5	58.6	5.5	48.9	5.4
$t_{\text{on}} = 0.4, t_{\text{off}} = 0.1,$ $\kappa_t = 0.9$	94.1	5.6	74.0	5.5	68.7	5.5	63.3	5.5	56.8	5.5	46.9	5.3
$t_{\text{on}} = 0.15, t_{\text{off}} = 0.1,$ $\kappa_t = 0.02$	94.3	5.6	75.6	5.5	69.4	5.5	62.2	5.5	55.2	5.5	45.1	5.3
$\alpha_{t_{\text{on}}} = 0.8, t_{\text{off}} = t_{\text{on}} - 0.1,$ $\kappa_t = 0.5$	95.3	5.6	74.6	5.5	69.8	5.5	64.4	5.5	59.9	5.5	48.5	5.2
$t_{\text{on}} = 0.85, t_{\text{off}} = 0.8,$ $\kappa_t = 0.02$	108.5	5.5	92.1	5.5	85.6	5.4	77.9	5.3	70.2	5.0	26.4	3.5
$\alpha_{t_{\text{on}}} = 0.7, t_{\text{off}} = t_{\text{on}} - 0.2,$ $\kappa = 0.5$	105.4	5.6	75.5	5.5	69.8	5.5	66.4	5.4	59.5	5.3	15.1	3.5
$t_{\text{on}} = 0.45, t_{\text{off}} = 0.4,$ $\kappa_t = 0.02$	110.5	5.5	99.3	5.5	99.6	5.4	106.4	5.4	88.2	5.0	8.3	2.4

Table 9: Generative perplexity (Gen. PPL, ↓) and entropy (↑) of Duo, distilled with DCD (Sahoo et al., 2025a) for different numbers of sampling steps: 32, 128, 256, 512, 1024, and 4096. Each column contains two sub-columns: Gen. PPL and Entropy. The Grayed row reports the configuration that achieves the best Gen. PPL while keeping entropy closest to the ancestral sampler.

	32		128		256		512		1024		4096	
	PPL	H	PPL	H	PPL	H	PPL	H	PPL	H	PPL	H
Ancestral sampler	68.2	5.5	58.9	5.5	59.5	5.4	58.1	5.4	56.9	5.4	57.0	5.4
$t_{\text{on}} = 0.15, t_{\text{off}} = 0.1,$ $\kappa_t = 0.02$	65.5	5.5	51.5	5.5	45.4	5.4	38.3	5.4	30.9	5.3	19.2	5.2
$t_{\text{on}} = 0.4, t_{\text{off}} = 0.1,$ $\kappa_t = 0.9$	64.6	5.5	48.5	5.5	43.8	5.4	36.2	5.4	28.3	5.3	16.7	5.1
$\alpha_{t_{\text{on}}} = 0.8, t_{\text{off}} = t_{\text{on}} - 0.1,$ $\kappa_t = 0.5$	64.1	5.5	47.6	5.4	42.2	5.4	35.1	5.4	28.5	5.3	17.1	5.1
$t_{\text{on}} = 0.5, t_{\text{off}} = 0.1,$ $\kappa_t = 0.95$	65.1	5.5	50.1	5.5	45.8	5.4	38.6	5.4	31.2	5.3	18.3	5.1
$t_{\text{on}} = 0.85, t_{\text{off}} = 0.8,$ $\kappa_t = 0.02$	71.5	5.5	61.3	5.4	55.3	5.3	49.5	5.1	40.1	4.9	25.6	4.4
$\alpha_{t_{\text{on}}} = 0.7, t_{\text{off}} = t_{\text{on}} - 0.2,$ $\kappa = 0.5$	62.3	5.5	41.1	5.4	34.7	5.3	27.6	5.2	21.4	5.1	10.3	4.6
$t_{\text{on}} = 0.45, t_{\text{off}} = 0.4,$ $\kappa_t = 0.02$	66.4	5.5	51.9	5.4	45.3	5.3	36.8	5.2	28.4	5.0	10.9	4.3

Table 10: Generative perplexity (Gen. PPL, ↓) and entropy (↑) of MDLM and ReMDM for different numbers of sampling steps: 32, 128, 256, 512, 1024, and 4096. Each column contains two sub-columns: Gen. PPL and Entropy. We omit certain settings (denoted by –), to spare compute costs. The Grayed row reports the configuration that achieves the best Gen. PPL while keeping entropy closest to the ancestral sampler.

	32		128		256		512		1024		4096	
	PPL	H	PPL	H	PPL	H	PPL	H	PPL	H	PPL	H
Ancestral sampler	193.2	5.7	122.3	5.6	113.3	5.6	109.2	5.6	102.6	5.6	103.9	5.6
ReMDM-cap ($\eta_t = 0.04$)	197.97	5.7	124.10	5.6	113.29	5.7	109.92	5.6	111.26	5.7	–	–
ReMDM-cap ($\eta_t = 0.04$, top-p=0.9)	132.06	5.7	75.51	5.5	64.69	5.5	55.38	5.5	46.65	5.4	–	–
ReMDM-loop ($\eta_t = 0.02$, $t_{\text{on}} = 0.55$, $t_{\text{off}} = 0.05$, $\alpha_t = 0.9$)	224.6	5.7	74.8	5.6	56.9	5.5	44.7	5.4	34.4	5.4	20.1	5.2
ReMDM-loop ($\eta_t = 0.02$, $t_{\text{on}} = 0.55$, $t_{\text{off}} = 0.05$, $\alpha_t = 0.9$, top-p=1.0)	338.9	5.8	146.8	5.7	133.4	5.7	138.8	5.7	166.4	5.7	357.5	5.7

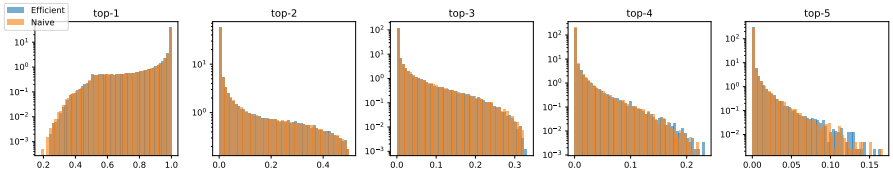

Figure 5: Marginal distributions of the top-5 entries using a tokenizer with 100k tokens, inverse temperature 100, and log signal-to-noise ratio -2. The histograms of the efficient and naive implementation match closely.

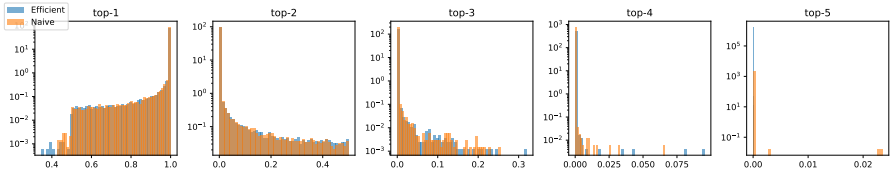

Figure 6: Marginal distributions of the top-5 entries using a tokenizer with 100k tokens, inverse temperature 1000, and log signal-to-noise ratio -1. The histograms of the efficient and naive implementation match closely.

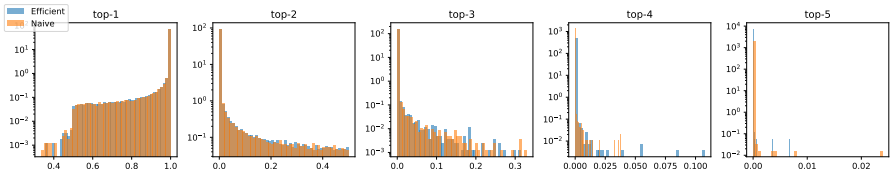

Figure 7: Marginal distributions of the top-5 entries using a tokenizer with 100k tokens, inverse temperature 1000, and log signal-to-noise ratio -2. The histograms of the efficient and naive implementation match closely.

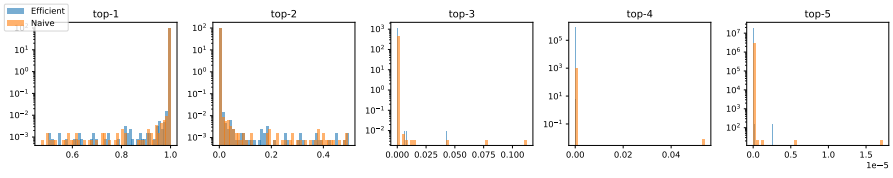

Figure 8: Marginal distributions of the top-5 entries using a tokenizer with 100k tokens, inverse temperature 1000, and log signal-to-noise ratio -4. The histograms of the efficient and naive implementation match closely.

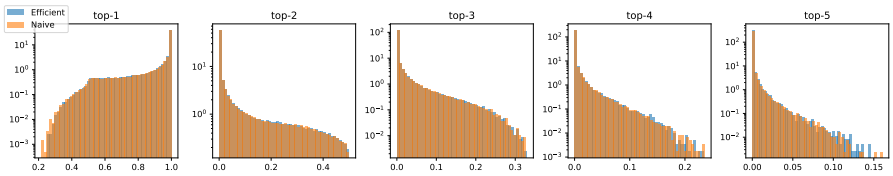

Figure 9: Marginal distributions of the top-5 entries using the GPT-2 tokenizer, inverse temperature 100, and log signal-to-noise ratio -2. The histograms of the efficient and naive implementation match closely.

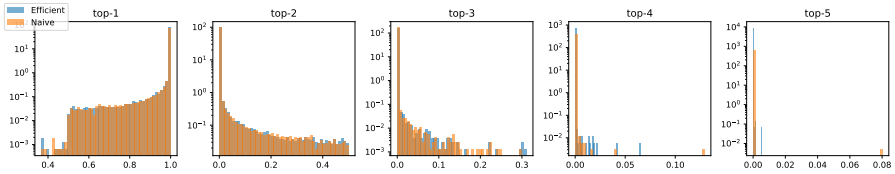

Figure 10: Marginal distributions of the top-5 entries using the GPT-2 tokenizer, inverse temperature 1000, and log signal-to-noise ratio -1. The histograms of the efficient and naive implementation match closely.

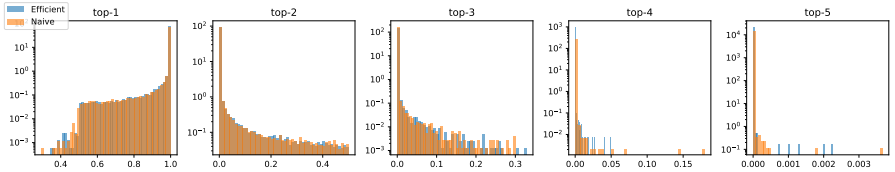

Figure 11: Marginal distributions of the top-5 entries using the GPT-2 tokenizer, inverse temperature 1000, and log signal-to-noise ratio -2. The histograms of the efficient and naive implementation match closely.

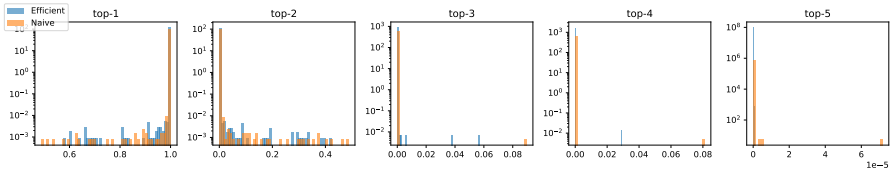

Figure 12: Marginal distributions of the top-5 entries using the GPT-2 tokenizer, inverse temperature 1000, and log signal-to-noise ratio -4. The histograms of the efficient and naive implementation match closely.