SIMPLE AND CONTROLLABLE UNIFORM DISCRETE DIFFUSION LANGUAGE MODELS

Anonymous authors
Paper under double-blind review

ABSTRACT

Diffusion models for continuous data gained widespread adoption owing to their high quality generation and control mechanisms. However, controllable diffusion on discrete data faces challenges: continuous diffusion guidance methods are not applicable and recent discrete diffusion models are not well-suited to control or exhibit a quality gap. Here, we provide a straightforward derivation of classifier-free and classifier-based guidance for discrete diffusion, as well as a new class of diffusion models that leverage uniform noise and thus can continuously edit their outputs. We improve the quality of these models with a novel continuous-time variational lower bound that yields state-of-the-art performance, in settings with small vocabularies. Empirically, we demonstrate the effectiveness of our guidance mechanisms relative to autoregressive and diffusion baselines, especially in conjunction with uniform noise diffusion, on several discrete data domains, including genomic sequences, small molecule design, and discretized image generation.

1 INTRODUCTION

Diffusion models (Sohl-Dickstein et al., 2015; Ho et al., 2020) gained widespread adoption in image generation and signal processing in part due to their high controllability using mechanisms such as classifier-based (Dhariwal & Nichol, 2021a) and classifier-free (Nichol et al., 2021; Ho & Salimans, 2022) guidance. Tasks where guidance plays a key role include MRI denoising (Song & Ermon, 2019), 3D reconstruction (Poole et al., 2022; Gao et al., 2024), and conditional generation (Saharia et al., 2022; Gokaslan et al., 2024).

However, applying controllable diffusion-based generation to tasks where the data is discrete (e.g., molecule design or text generation) presents challenges. First, standard diffusion models and their guidance mechanisms are not directly applicable, since they require taking gradients with respect to the data, and these are not defined in discrete settings. Second, popular discrete extensions of diffusion (Lou et al., 2023; Sahoo et al., 2024a) cannot perform multiple editing passes on generated tokens, hence are not ideal for controllable generation. Third, the performance of discrete diffusion models (measured by perplexity) lags behind autoregressive (AR) models, especially for classes of diffusion that are amenable to control, such as uniform noise (Austin et al., 2021; Lou et al., 2023).

Here, we propose discrete diffusion models and guidance mechanisms that are effective at controllable generation and that address the above challenges. First, we provide straightforward and easy to implement adaptations of classifier-based and classifier-free guidance for discrete diffusion models. Second, we introduce uniform noise diffusion language models (UDLM), which undo random token perturbations and are particularly amenable to guidance, since they can continuously edit discrete data (Austin et al., 2021). Third, we address performance issues that plagued previous iterations of uniform noise discrete diffusion, namely, we introduce a continuous time version of the evidence lower bound which tightens the variational gap (Kingma et al., 2021; Sahoo et al., 2024a).

We demonstrate the effectiveness of guidance with discrete diffusion models relative to AR models on several domains: genomics, molecular generation, and discretized images. Our guidance results indicate that classifier-free guidance is more useful when paired with diffusion models compared to AR and that our proposal for classifier-based guidance is the best classifier-based method for discrete guidance, especially when combined with UDLM. Our language modeling experiments reveal that contrary to a widely-held belief (Austin et al., 2021; Lou et al., 2023), uniform noise diffusion can

attain state-of-the-art performance on small vocabulary datasets (e.g,. molecules, DNA) and that UDLM attains a new state-of-the-art in perplexity among uniform noise diffusion models.

In summary, our contributions are as follows:

- We provide simple and effective discrete classifier-based and classifier-free guidance.
- We introduce UDLM, a class of discrete diffusion models particularly amenable to guidance, and we derive a tightened ELBO that significantly improves their performance.
- Across three domains, we demonstrate that discrete guidance yields better controllable generation compared to strong AR baselines and previous diffusion guidance methods.

2 Background

Notation Let $\mathcal{V}$ be the space of all one-hot tokens over some vocabulary consisting of N unique characters: $\mathcal{V} = \{\mathbf{z} \in \{0,1\}^N : \sum_i \mathbf{z}_i = 1\} \subset \Delta^N$, where Δ^N represents the simplex over N categories. Let $\mathbf{1}$ be a N-dimensional column vector of all ones, and denote the Hadamard product between two vectors as $\odot$. We define $\mathbf{z}^{(1:L)}$ as a sequence of L tokens, where $\mathbf{z}^{(\ell)} \in \mathcal{V}$, for all tokens $\ell \in 1, \dots, L$, and use $\mathcal{V}^L$ to denote the set of all such sequences. Finally, let $\mathrm{Cat}(\cdot; p)$ denote the categorical distribution with probability vector $p \in \Delta^N$.

2.1 Discrete Diffusion Models

Diffusion models are a class of generative models defined by a denoising network p_θ that is trained to remove noise from latent variables $\mathbf{z}_t$. These latents are generated by a fixed corruption process q that, starting from clean data $\mathbf{x}_0$ drawn from the data distribution $q(\mathbf{x}_0)$, increasingly adds more noise to $\mathbf{z}_t$, as t increases (Sohl-Dickstein et al., 2015; Song & Ermon, 2019; Ho et al., 2020).

In discrete denoising diffusion probabilistic models (D3PM; Austin et al. (2021)), the noising process is defined in terms of a transition matrix $Q_{t|s}$ whose $(i,j)^{\text{th}}$ entry is the probability of transitioning from the i-th state at time s to the j-th state at time t. This induces a Markov corruption process where we have $q(\mathbf{z}_t \mid \mathbf{z}_s) = \mathrm{Cat}(\mathbf{z}_t; Q_{t|s}\mathbf{z}_s)$. Sahoo et al. (2024a) build off this framework to introduce specialized algorithms that are both simpler and more effective than the general D3PM framework. They focus on a specific class of forward processes from D3PM that can be defined as interpolations between clean data and a noisy prior $\boldsymbol{\pi}$, and we adopt their notation below:

$$q(\mathbf{z}_t|\mathbf{x}_0) = \mathrm{Cat}(\mathbf{z}_t; \alpha_t \mathbf{x}_0 + (1-\alpha_t)\boldsymbol{\pi}), \tag{1}$$

where $\alpha_t = \alpha(t)$ is a noise schedule monotonically decreasing in t. Defining $\alpha_{t|s} = \alpha_t/\alpha_s$, this class of processes admit the following posteriors

$$q(\mathbf{z}_s|\mathbf{z}_t, \mathbf{x}_0) = \mathrm{Cat}\left(\mathbf{z}_s; \frac{[\alpha_{t|s}\mathbf{z}_t + (1-\alpha_{t|s})\mathbf{1}\boldsymbol{\pi}^\top \mathbf{z}_t] \odot [\alpha_s \mathbf{x}_0 + (1-\alpha_s)\boldsymbol{\pi}]}{\alpha_t \mathbf{z}_t^\top \mathbf{x}_0 + (1-\alpha_t)\mathbf{z}_t^\top \boldsymbol{\pi}}\right). \tag{2}$$

Of note, for absorbing-state diffusion, where $\boldsymbol{\pi} = [\text{MASK}]$, a one-hot vector at the special `[MASK]` token index, Sahoo et al. (2024a) show that when the latent $\mathbf{z}_t \neq [\text{MASK}]$ then $q(\mathbf{z}_s|\mathbf{z}_t, \mathbf{x}_0) = \mathrm{Cat}(\mathbf{z}_s; \mathbf{z}_t)$, which reflects the fact that unasked tokens at time t must remain unmasked for all time $s < t$.

2.2 Diffusion Guidance

For continuous data, diffusion models have demonstrated state-of-the-art controllable generation by means of classifier-based (Sohl-Dickstein et al., 2015; Dhariwal & Nichol, 2021a) and classifier-free guidance (Nichol et al., 2021; Ho & Salimans, 2022; Saharia et al., 2022). These approaches rely on different ways of expressing the score of a distribution conditioned on y.

Classifier-based Classifier-based generation employs a diffusion model to iteratively sample from a tempered distribution $p^\gamma(\mathbf{z}_s \mid y, \mathbf{z}_t) \propto p(y \mid \mathbf{z}_s)^\gamma p_\theta(\mathbf{z}_s \mid \mathbf{z}_t)$, where γ represents an inverse temperature parameter, $p_\theta(\mathbf{z}_s \mid \mathbf{z}_t)$ is a pre-trained diffusion model, and $p(y \mid \mathbf{z}_s)$ is a classifier:

$$\nabla_{\mathbf{z}_s} \log p^\gamma(\mathbf{z}_s \mid y, \mathbf{z}_t) = \gamma \nabla_{\mathbf{z}_s} \log p(y \mid \mathbf{z}_s) + \nabla_{\mathbf{z}_s} \log p_\theta(\mathbf{z}_s \mid \mathbf{z}_t). \tag{3}$$

Classifier-free We can also observe that appyling Bayes' rule to $p(y \mid \mathbf{x})$ and differentiating with respect to the input yields $\nabla_{\mathbf{x}} \log p(y \mid x) = \nabla_{\mathbf{x}} \log p(x \mid y) - \nabla_{\mathbf{x}} \log p(x)$. Applying this to $p(y \mid \mathbf{z}_s)$ and plugging into (3) gives us the formulation for classifier-free guidance:

$$\begin{aligned}\nabla_{\mathbf{z}_s} \log p^{\gamma}(\mathbf{z}_s \mid y, \mathbf{z}_t) &= \gamma \cdot [\nabla_{\mathbf{z}_s} \log p_\theta(\mathbf{z}_s \mid y, \mathbf{z}_t) - \nabla_{\mathbf{x}} \log p_\theta(\mathbf{z}_s \mid \mathbf{z}_t)] + \nabla_{\mathbf{z}_s} \log p_\theta(\mathbf{z}_s \mid \mathbf{z}_t) \\ &= \gamma \nabla_{\mathbf{z}_s} \log p_\theta(\mathbf{z}_s \mid y, \mathbf{z}_t) + (1-\gamma) \nabla_{\mathbf{z}_s} \log p_\theta(\mathbf{z}_s \mid \mathbf{z}_t). \end{aligned} \tag{4}$$

where $p_\theta(\mathbf{z}_s \mid y, \mathbf{z}_t)$ and $p_\theta(\mathbf{z}_s \mid \mathbf{z}_t)$ represent conditional and unconditionally trained diffusion models, respectively. In practice, this is often implemented by using a single model p_θ and randomly dropping out / masking the conditioner y during training.

The problem with applying guidance to discrete diffusion is that guidance terms are not differentiable with respect to the the discrete representations $\mathbf{z}_t$ (Sahoo et al., 2023; 2024a).

3 Uniform Diffusion Language Models

While masked diffusion models demonstrate better language modeling compared to other discrete diffusion (Austin et al., 2021; Lou et al., 2023), we argue that they are less amenable to guidance, since once a token is unmasked at some time t it remains so for all $s < t$. In contrast, with uniform noising, intermediate latents can be refined multiple times throughout the denoising process.

We therefore revisit categorical uniform noise discrete diffusion, where $\boldsymbol{\pi} = \boldsymbol{u} = \mathbf{1}/N$. Our aim is that by analyzing this class of diffusion models more carefully, we can reduce the gap to absorbing-state and yield performant models that are more easily steered by guidance tools we develop below.

3.1 Uniform Noise Diffusion

Discrete Time Likelihood Bound We start with the variational lower bound that is defined by a diffusion process. For discrete time diffusion, i.e., some finite steps T, we define $t(i) = (i+1)/T$ and $s(i) = i/T$ for i in $0, \dots, T$. The denoising network p_θ is trained to minimize a variational upper bound (NELBO), which is given by:

$$\mathbb{E}_q \Bigg[\underbrace{-\log p_\theta(\mathbf{x}_0 | \mathbf{z}_{t(0)})}_{\mathcal{L}_{\text{recons}}} + \underbrace{\sum_{i=1}^{T} D_{\mathrm{KL}}[q(\mathbf{z}_{s(i)} | \mathbf{z}_{t(i)}, \mathbf{x}_0) \| p_\theta(\mathbf{z}_{s(i)} | \mathbf{z}_{t(i)})]}_{\mathcal{L}_{\text{diffusion}}} \Bigg] + \underbrace{D_{\mathrm{KL}}[q(\mathbf{z}_{t(T)} | \mathbf{x}_0) \| p_\theta(\mathbf{z}_{t(T)})]}_{\mathcal{L}_{\text{prior}}}, \tag{5}$$

where D_{KL} represents the Kullback-Leibler divergence. When clear, we drop the explicit dependence of t and s on the discrete step i.

Uniform Noise Forward Process We formulate uniform noise diffusion using the interpolating discrete diffusion framework (Sahoo et al., 2024a). When letting $\boldsymbol{\pi} = \boldsymbol{u}$, the input $\mathbf{x}$ transitions to a random state with some probability at each time step. Crucially, after $\mathbf{x}$ changes once, it can do so again. Formally, when $\boldsymbol{\pi} = \boldsymbol{u}$, the posterior from (2) becomes

$$q(\mathbf{z}_s \mid \mathbf{z}_t, \mathbf{x}_0) = \mathrm{Cat}\left(\mathbf{z}_s; \frac{N\alpha_t \mathbf{z}_t \odot \mathbf{x}_0 + (\alpha_{t|s} - \alpha_t)\mathbf{z}_t + (\alpha_s - \alpha_t)\mathbf{x}_0 + \frac{(\alpha_s - \alpha_t)(1-\alpha_s)}{N\alpha_s}\mathbf{1}}{N\alpha_t \langle \mathbf{z}_t, \mathbf{x}_0 \rangle + 1 - \alpha_t}\right) \tag{6}$$

Denoising Process The optimal form for the reverse diffusion process p_θ matches (6): in fact setting p_θ to (6) reduces the KL terms in (5) to zero. However, setting p_θ to exactly (6) is not possible because it cannot be a function $\mathbf{x}_0$ (which p_θ is generating). Therefore, we introduce a predictive model $\mathbf{x}_\theta(\mathbf{z}_t, t) : \mathcal{V} \times [0,1] \to \Delta^N$ of the 'clean' data given a noisy latent $\mathbf{z}_t$ at time t. We use $\mathbf{x}_\theta$ to parameterize the denoising process as $p_\theta(\mathbf{z}_s \mid \mathbf{z}_t) = q(\mathbf{z}_s \mid \mathbf{z}_t, \mathbf{x} = \mathbf{x}_\theta)$, yielding:

$$p_\theta(\mathbf{z}_s \mid \mathbf{z}_t) = \mathrm{Cat}\left(\mathbf{z}_s; \frac{N\alpha_t \mathbf{z}_t \odot \mathbf{x}_\theta + (\alpha_{t|s} - \alpha_t)\mathbf{z}_t + (\alpha_s - \alpha_t)\mathbf{x}_\theta + \frac{(\alpha_s - \alpha_t)(1-\alpha_s)}{N\alpha_s}\mathbf{1}}{N\alpha_t \langle \mathbf{z}_t, \mathbf{x}_\theta \rangle + 1 - \alpha_t}\right), \tag{7}$$

Note that this minimizes the $\mathcal{L}_{\text{diffusion}}$ term in (5) precisely when $\mathbf{x}_\theta = \mathbf{x}_0$, as desired. To simplify notation, we omit below the explicit dependence of $\mathbf{x}_\theta$ on t.

3.2 Improved Likelihood Bounds in Continuous Time

Next, we leverage the above formulation to develop an improved ELBO by taking $T \to \infty$ and analyzing each term $\mathcal{L}_{\text{recons}}, \mathcal{L}_{\text{diffusion}}, \mathcal{L}_{\text{prior}}$ in (5). This yields three improvements: (1) a simple and elegant closed-form expression for the ELBO that is easier to reason about; (2) an analytical reduction of $\mathcal{L}_{\text{recons}}, \mathcal{L}_{\text{prior}}$ to zero, which tightens the ELBO; (3) a further tightening via the continuous-time extension of $\mathcal{L}_{\text{diffusion}}$ as in Kingma et al. (2021) and Sahoo et al. (2024a;b).

Prior Loss ($\mathcal{L}_{\text{prior}}$) Given that we define our corruption process as an interpolation between clean data and a limiting distribution, for any noise schedules where $\alpha_{t(T)} = 0$, the distribution $q(\mathbf{z}_{t(T)}) = \boldsymbol{\pi}$. Therefore, we can simply define $p_\theta(\mathbf{z}_{t(T)}) = \boldsymbol{\pi}$, and the KL divergence in $\mathcal{L}_{\text{prior}}$ evaluates to zero.

Reconstruction Loss ($\mathcal{L}_{\text{recons}}$) Similarly, for $\mathcal{L}_{\text{recons}}$, if our noise schedule is such that $T \to \infty \implies \alpha_{t(0)} = 1$ (i.e., $t(0) = 1/T$), then the marginal $q(\mathbf{z}_{\frac{1}{T}} \mid \mathbf{x}_0) \to \text{Cat}(\mathbf{z}_{\frac{1}{T}}; \mathbf{x}_0)$. That is, in the limit, the first latent vector is identically equal to the clean data. We can thus parameterize our denoising network such that at time $t(0)$ the function simply copies its inputs: $\mathbf{x}_\theta(\mathbf{z}_{\frac{1}{T}}, 1/T) = \mathbf{z}_{\frac{1}{T}}$. Additionally, we note that our choice of parameterization for p_θ implies that $p_\theta(\mathbf{x}_0 \mid \mathbf{z}_{\frac{1}{T}}) = \mathbf{x}_\theta(\mathbf{z}_{\frac{1}{T}}, 1/T)$. Thus in the continuous time limit, we have:

$$\lim_{T\to\infty} \mathbb{E}_q[\mathcal{L}_{\text{recons}}] = \lim_{T\to\infty} \mathbb{E}_q[\log p_\theta(\mathbf{x}_0 \mid \mathbf{z}_{\frac{1}{T}})] = \lim_{T\to\infty} \mathbb{E}_q[\log(\langle \mathbf{x}_0, \mathbf{x}_\theta(\mathbf{z}_{\frac{1}{T}}, 1/T)\rangle)] = 0.$$

Diffusion Loss ($\mathcal{L}_{\text{diffusion}}$) Turning finally to the diffusion loss term $\mathcal{L}_{\text{diffusion}}$, we first define the shorthand $D_{\text{KL}}[q_t||p_\theta] = D_{\text{KL}}[[q(\mathbf{z}_s \mid \mathbf{z}_t, \mathbf{x}_0)||p_\theta(\mathbf{z}_s \mid \mathbf{z}_t)]]$ and then re-write this term as an expectation over t uniformly sampled from $1/T, 2/T, \dots, T$:

$$\mathcal{L}_{\text{diffusion}} = \sum_{i=1}^{T} D_{\text{KL}}[q_{i/T}||p_\theta] = T \cdot \mathbb{E}_{t\sim\{1/T,2/T,\dots,T\}} D_{\text{KL}}[q_t||p_\theta]. \tag{8}$$

Plugging in our expressions for the true and predicted posteriors from (6) and (7) into (8), then taking $T \to \infty$, we get (see Appendix A for details):

$$\begin{aligned}\lim_{T\to\infty} \mathcal{L}_{\text{diffusion}} &= \lim_{T\to\infty} \mathbb{E}_{t\sim\{1/T,2/T,\dots,T\}} T \cdot D_{\text{KL}}[q_t||p_\theta] = \int_{t=0}^{t=1} \lim_{T\to\infty} T \cdot D_{\text{KL}}[q_t||p_\theta] \mathrm{d}t \\ &= \int_{t=0}^{t=1} \left[\frac{\alpha_t'}{N\alpha_t} \left[\frac{N}{\bar{\mathbf{x}}_i} - \frac{N}{(\bar{\mathbf{x}}_\theta)_i} - \sum_{j \text{ s.t. } (\mathbf{z}_t)_j = 0} \left(\frac{\bar{\mathbf{x}}_j}{\bar{\mathbf{x}}_i} \right) \log \left[\left(\frac{(\bar{\mathbf{x}}_\theta)_i \cdot \bar{\mathbf{x}}_j}{(\bar{\mathbf{x}}_\theta)_j \cdot \bar{\mathbf{x}}_i} \right) \right] \right] \right] \mathrm{d}t,\end{aligned} \tag{9}$$

where $\mathbf{x}_i$ denotes the i^{th} index of a vector $\mathbf{x}$, $\bar{\mathbf{x}} = N\alpha_t \mathbf{x} + (1-\alpha_t)\mathbf{1}$, $\bar{\mathbf{x}}_\theta = N\alpha_t \mathbf{x}_\theta + (1-\alpha_t)\mathbf{1}$, and we define $i = \arg\max_{j\in[N]} (\mathbf{z}_t)_j$ to be the non-zero entry of $\mathbf{z}_t$.

Combining our arguments regarding $\mathcal{L}_{\text{prior}}$ and $\mathcal{L}_{\text{recons}}$ with (9), yields our final tight variational bound:

$$\mathcal{L}^\infty = \int_{t=0}^{t=1} \mathbb{E}_q \left[\frac{\alpha_t'}{N\alpha_t} \left[\frac{N}{\bar{\mathbf{x}}_i} - \frac{N}{(\bar{\mathbf{x}}_\theta)_i} - \sum_{j \text{ s.t. } (\mathbf{z}_t)_j = 0} \left(\frac{\bar{\mathbf{x}}_j}{\bar{\mathbf{x}}_i} \right) \log \left[\left(\frac{(\bar{\mathbf{x}}_\theta)_i \cdot \bar{\mathbf{x}}_j}{(\bar{\mathbf{x}}_\theta)_j \cdot \bar{\mathbf{x}}_i} \right) \right] \right] \right] \mathrm{d}t. \tag{10}$$

Extension to Sequences Extending training with (10) from $\mathbf{x} \in \mathcal{V}$ to sequences $\mathbf{x}^{(1:L)} \in \mathcal{V}^L$, we make the assumption that the denoising process factorizes independently across tokens when conditioned on a sequence of noisy latents $\mathbf{z}_t^{(1:L)}$. In this case, we use a single model $\mathbf{x}_\theta^{(\ell)}(\mathbf{z}_t^{(1:L)}, t)$ for predicting each token $\ell \in \{1, \dots, L\}$ in a sequence, and we train with the sequence-level objective:

$$\mathcal{L}^\infty = \int_{t=0}^{t=1} \mathbb{E}_q \sum_\ell \left[\frac{\alpha_t'}{N\alpha_t} \left[\frac{N}{\bar{\mathbf{x}}_i^{(\ell)}} - \frac{N}{(\bar{\mathbf{x}}_\theta^{(\ell)})_i} - \sum_{j \text{ s.t. } (\mathbf{z}_t^{(\ell)})_j = 0} \left(\frac{\bar{\mathbf{x}}_j^{(\ell)}}{\bar{\mathbf{x}}_i^{(\ell)}} \right) \log \left[\left(\frac{(\bar{\mathbf{x}}_\theta^{(\ell)})_i \cdot \bar{\mathbf{x}}_j^{(\ell)}}{(\bar{\mathbf{x}}_\theta^{(\ell)})_j \cdot \bar{\mathbf{x}}_i^{(\ell)}} \right) \right] \right] \right] \mathrm{d}t. \tag{11}$$

We dub models trained with our refined objective **U**niform **D**iffusion **L**anguage **M**odels (UDLM).

4 GUIDANCE ALGORITHMS FOR DISCRETE DIFFUSION

Next, we introduce two guidance algorithms for discrete diffusion, including, but not limited to, UDLM. As in the continuous case, we formalize the guidance term as a probability $p(y|\mathbf{z})$, where $y \in \{1, \dots, K\}$ is one of K possible classes and Δ^K is the K-simplex defined over these classes.

4.1 CLASSIFIER-FREE GUIDANCE

We refer to the following discrete version of classifier guidance as **D-CFG** for **D**iscrete **C**lassifier-**F**ree **G**uidance. To derive D-CFG, we begin by applying Bayes' Rule:

$$p(\mathbf{z}_s \mid \mathbf{z}_t, y) = [p(y \mid \mathbf{z}_s, \mathbf{z}_t)/p(y \mid \mathbf{z}_t)]\, p(\mathbf{z}_s \mid \mathbf{z}_t)$$

As in Section 2.2, we wish to sample from the distribution:

$$p^\gamma(\mathbf{z}_s \mid \mathbf{z}_t, y) \propto [p(y \mid \mathbf{z}_s, \mathbf{z}_t)/p(y \mid \mathbf{z}_t)]^\gamma\, p(\mathbf{z}_s \mid \mathbf{z}_t)$$

We now examine the right hand side and again apply Bayes' rule :

$$\left[\frac{p(y \mid \mathbf{z}_s, \mathbf{z}_t)}{p(y \mid \mathbf{z}_t)}\right]^\gamma p(\mathbf{z}_s \mid \mathbf{z}_t) = \left[\frac{p(\mathbf{z}_s \mid \mathbf{z}_t, y)p(y \mid \mathbf{z}_t)}{p(\mathbf{z}_s \mid \mathbf{z}_t)p(y \mid \mathbf{z}_t)}\right]^\gamma p(\mathbf{z}_s \mid \mathbf{z}_t) = p(\mathbf{z}_s \mid \mathbf{z}_t, y)^\gamma p(\mathbf{z}_s \mid \mathbf{z}_t)^{(1-\gamma)}.$$

In practice, we train a conditional denoising network $p_\theta(\mathbf{z}_s \mid \mathbf{z}_t, y)$ alongside an unconditional one $p_\theta(\mathbf{z}_s \mid \mathbf{z}_t)$, both of which are parameterized such that for a sequence of L tokens $\mathbf{z}_s^{(1:L)}$ the probabilities factorize independently across the tokens of the sequence when conditioned on $\mathbf{z}_t^{(1:L)}$. We can thus sample from $p^\gamma(\mathbf{z}_s^{(1:L)} \mid \mathbf{z}_t^{(1:L)}, y)$ by computing $p_\theta(\mathbf{z}_s^{(1:L)} \mid \mathbf{z}_t^{(1:L)}, y)^\gamma p_\theta(\mathbf{z}_s^{(1:L)} \mid \mathbf{z}_t^{(1:L)})^{(1-\gamma)}$, which factorizes independently across tokens, and re-normalizing for each token:

$$p_\theta^\gamma(\mathbf{z}_s^{(1:L)} \mid \mathbf{z}_t^{(1:L)}, y) = \prod_{\ell=1}^{L} \frac{1}{Z^{(\ell)}} p_\theta(\mathbf{z}_s^{(\ell)} \mid \mathbf{z}_t^{(1:L)}, y)^\gamma p_\theta(\mathbf{z}_s^{(\ell)} \mid \mathbf{z}_t^{(1:L)})^{(1-\gamma)}, \tag{12}$$

where $Z^{(\ell)} = \sum_{\mathbf{z}'_s} p_\theta(\mathbf{z}'_s \mid \mathbf{z}_t^{(1:L)}, y)^\gamma p_\theta(\mathbf{z}'_s \mid \mathbf{z}_t^{(1:L)})^{(1-\gamma)}$ is the per-token partition function.

4.2 CLASSIFIER-BASED GUIDANCE

Next, we describe a discrete version of classifier-based guidance: **D-CBG** for **D**iscrete **C**lassifier-**B**ased **G**uidance. Extending classifier-based guidance to diffusion models is difficult because the guiding classifier need not factorize the same as the diffusion denoising network, which would imply that the classifier needs to be evaluated on exponentially many sequence combinations. We resolve this using factorization assumptions on the decoding model and a Taylor expansion trick.

First, we introduce the following additional notation: given $\mathbf{z}^{1:L}$, let $\tilde{\mathcal{Z}}_\ell(\mathbf{z}^{1:L})$ be the set of sequences $\tilde{\mathbf{z}}^{(1:L)}$ for which $\tilde{\mathbf{z}}^{(\ell')} = \mathbf{z}^{(\ell')}$ for all $\ell' \neq \ell$, i.e., the set of sequences that are either the same as or only differ in position ℓ relative to $\mathbf{z}^{(1:L)}$.

To formulate discrete classifier-based guidance, we make the assumption that conditioned on $\mathbf{z}_t^{(1:L)}$, the tempered distribution from which we want to sample $p^\gamma(\mathbf{z}_s^{(1:L)} \mid \mathbf{z}_t^{1:L}, y)$ factorizes independently across tokens. Therefore, we can focus on the tempered distirbution of each token $\mathbf{z}_s^{(\ell)}$, for $\ell \in 1, \dots, L$:

$$p^\gamma(\mathbf{z}_s^{(\ell)} \mid \mathbf{z}_t^{(1:L)}, y) \propto p(y \mid \mathbf{z}_s^{(\ell)}, \mathbf{z}_t^{(1:L)})^\gamma p(\mathbf{z}_s^{(\ell)} \mid \mathbf{z}_t^{(1:L)}).$$

In practice, we can sample from $p^\gamma(\mathbf{z}_s^{(\ell)} \mid \mathbf{z}_t^{(1:L)}, y)$ by training a classifier $p_\phi : \mathcal{V}^L \to \Delta^K$ on noised latents $\mathbf{z}_t^{(1:L)}$ for $t \in [0, 1]$ and use this to model the first term on the right hand side by only evaluating p_ϕ on sequences for which $\mathbf{z}_s^{(1:L)}$ and $\mathbf{z}_t^{(1:L)}$ differ by at most the token at position ℓ:

$$p(y \mid \mathbf{z}_s^{(\ell)}, \mathbf{z}_t^{(1:L)}) \approx p_\phi(y \mid \tilde{\mathbf{z}}^{(1:L)}), \quad \text{for} \quad \tilde{\mathbf{z}}^{(1:L)} = \left[\mathbf{z}_t^{(1:\ell-1)}, \mathbf{z}_s^{(\ell)}, \mathbf{z}_t^{(\ell+1:L)}\right] \in \tilde{\mathcal{Z}}_\ell(\mathbf{z}_t^{(1:L)}).$$

We additionally train an unconditional denoising network $p_\theta(\mathbf{z}_s^{(\ell)} \mid \mathbf{z}_t^{(1:L)})$ and then sample from the re-normalized distribution:

$$p^\gamma_{\phi,\theta}(\mathbf{z}_s^{(\ell)} \mid \mathbf{z}_t^{(1:L)}, y) = \frac{p_\phi(y \mid \tilde{\mathbf{z}}^{(1:L)})^\gamma p_\theta(\mathbf{z}_s^{(\ell)} \mid \mathbf{z}_t^{(1:L)})}{\sum_{\tilde{\mathbf{z}}^{(1:L)}} p_\phi(y \mid \tilde{\mathbf{z}}^{(1:L)})^\gamma p_\theta(\mathbf{z}_s^{(\ell)} \mid \mathbf{z}_t^{(1:L)})}. \tag{13}$$

Restricting the summation in the denominator of (13) to $\tilde{\mathcal{Z}}_\ell(\mathbf{z}_t^{(1:L)})$ makes normalization tractable, as we are only summing over N terms.

Our method can be thought of as an adaptation of the successful FUDGE (Yang & Klein, 2021) approach, which guides AR generation, to discrete diffusion, similar to how NOS (Gruver et al., 2024) extended the AR guidance mechanism of PPLM (Dathathri et al., 2019) to diffusion models.

First-Order Approximation While tractable, this formulation suffers from the drawback that at each denoising step we must perform $\mathcal{O}(L \cdot N)$ forward passes through the classifier model, which can quickly become impractical for larger vocabularies. Similarly to Grathwohl et al. (2021), Vignac et al. (2022), and Nisonoff et al. (2024), we treat the classifier p_ϕ as a continuous function of the one-hot inputs $\tilde{z}^{(1:L)} \in \mathbb{R}^{L \times N}$ and use the first-order Taylor approximation of $\log p_\phi$ to efficiently compute $p_\phi(y \mid \mathbf{z}_s^{(\ell)})$ with only a single forward and backward pass through the classifier model:

$$\begin{aligned} p_\phi(y \mid \tilde{\mathbf{z}}^{(1:L)}) &= \exp\left(\log \frac{p_\phi(y \mid \tilde{\mathbf{z}}^{(1:L)})}{p_\phi(y \mid \mathbf{z}_t^{(1:L)})} + \log p_\phi(y \mid \mathbf{z}_t^{(1:L)})\right) \\ &\approx \exp\left((\tilde{\mathbf{z}}^{(1:L)} - \mathbf{z}_t^{(1:L)})^T \nabla_{\mathbf{z}_t^{(1:L)}} \log p_\phi(y \mid \mathbf{z}_t^{(1:L)}) + \log p_\phi(y \mid \mathbf{z}_t^{(1:L)})\right). \end{aligned} \tag{14}$$

5 Experiments

Datasets For our language modeling experiments we examine several distinct discrete domains: reference genomes from ten diverse species (Species-10), the small molecule dataset known as QM9 (Ruddigkeit et al., 2012; Ramakrishnan et al., 2014), where molecules are represented by a string of characters known as SMILES (Weininger, 1988), discretized images from CIFAR-10 (Krizhevsky et al., 2009), and three NLP datasets consisting of text8 (Mahoney, 2011), Amazon Review Polarity (McAuley & Leskovec, 2013; Zhang et al., 2015), and the one billion words dataset (LM1B; Chelba et al. (2014)). These datasets cover a range of domains and vocabularies of varying sizes (see Table 1). For guidance experiments, we explore species-specific sequence generation, molecular property maximization, and class-conditional image generation.

5.1 Language Modeling with Uniform Noise Discrete Diffusion

Our language modeling experiments reveal that (1) contrary to a widely-held belief, **uniform noise diffusion can attain state-of-the-art performance** on small vocabulary datasets (Table 1), and that (2) our **UDLM are state-of-the-art among uniform noise diffusion models** (Tables 2 and 3).

In Table 1, despite previous evidence indicating that absorbing-state discrete diffusion greatly outperforms uniform noise, we find a more nuanced story. Namely, for smaller vocabulary regimes, which are common for scientific applications, the gap between MDLM and UDLM is negligible, with UDLM even outperforming absorbing-state diffusion on certain datasets. Even within a single domain, we find a trend between vocabulary size and performance gap, with the text8 results of UDLM on par with MDLM, and a more persistent gap for the larger vocabulary experiments. Intuitively, these results follow from the observation that in larger vocabulary regimes, at every denoising step, uniform noise diffusion models need to predict over a combinatorially larger set of potential clean data sequences compared to absorbing-state; smaller vocabularies reduce this complexity.

Secondly, although we still find a persisting perplexity gap between absorbing and uniform noise discrete diffusion for larger vocabulary NLP datasets, we note that our uniform noise diffusion models trained with UDLM, closes this gap, as evidenced in Tables 2 and 3, where we show that UDLM attains the best reported uniform noise discrete diffusion language modeling performance.

Table 1: UDLM performs best in smaller vocabulary regimes. Values correspond to perplexity (PPL; ↓) on various datasets. Best values are **bolded.** Datasets marked with * have values reported from early stopping on the validation set, since AR model demonstrated overfitting; Values for all other datasets reflect validation performance at the end of training. †Taken from Lou et al. (2023).

		Vocab. Size	AR	MDLM	UDLM
Bio.	Species-10	12	**2.88**	3.17≤	3.15≤
Chem.	QM9*	40	2.19	2.12≤	**2.02**≤
Images	CIFAR-10	256	-	**9.14**≤	11.21≤
NLP	text8	35	**2.35**†	2.62≤	2.71≤
	Amazon*	30,522	**21.67**	24.93≤	27.27≤
	LM1B	30,522	**22.83**	31.69≤	35.35≤

Table 2: UDLM outperforms previously reported uniform noise discrete diffusion models on text8 dataset. Values correspond to bits per character (BPC). Best values are **bolded** and best discrete uniform diffusion value is underlined. †Taken from Lou et al. (2023).

Method	BPC (↓)
Autoregressive†	
IAF/SCF [64]	1.88
AR Argmax Flow [19]	1.39
Discrete Flow [56]	**1.23**
Autoregressive	**1.23**
Non-autoregressive†	
Mult. Diffusion [19]	1.72≤
MAC [50]	1.40≤
BFN [12]	1.41≤
D3PM Absorb [1]	1.45≤
SEDD Absorb [25]	1.39≤
MDLM [42] *(Retrained)*	1.38≤
Discrete Uniform Diffusion	
D3PM Uniform† [1]	1.61≤
SEDD Uniform† [25]	1.47≤
UDLM (*Ours*)	1.44≤

Table 3: UDLM outperforms previously reported uniform noise discrete diffusion models on LM1B dataset. Values correspond to perplexity (PPL). Best values are **bolded** and best discrete uniform diffusion value is underlined. †Taken from Sahoo et al. (2024a). *Taken from Lou et al. (2023).

Method	PPL (↓)
Autoregressive†	
Transformer-X Base [5]	23.5
OmniNetT [55]	**21.5**
Transformer *(Retrained)*	22.83
Diffusion†	
BERT-Mouth [58]	142.89≤
D3PM Absorb [1]	77.50≤
Diffusion-LM [24]	118.62≤
DiffusionBert [58]	63.78≤
SEDD Absorb [25]	32.79≤
MDLM [42] *(Retrained)*	31.69≤
Discrete Uniform Diffusion	
D3PM Uniform* [1]	77.50≤
SEDD Uniform* [25]	40.25≤
UDLM (*Ours*)	35.35≤

Ablating Continuous Time Formulation In Figure 1, we see the effect of increasing T and more specifically using our continuous time ELBO. The curves represent validation perplexity on the Amazon Polarity dataset at various training steps, and we observe that increasing T indeed improves language modeling with UDLM, i.e., $T = \infty$, performing best.

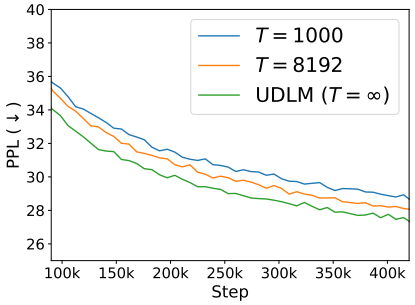

Figure 1: Increasing T to infinity improves language modeling. Values correspond to validation perplexity (PPL) for uniform noise discrete diffusion using finite T vs. UDLM ($T = \infty$).

5.2 Guided Discrete Diffusion

Our guidance results indicate that (1) **classifier-free guidance is more useful when paired with diffusion models compared to AR** (Table 4) and that (2) **our proposed D-CBG is the best classifier-based method** for discrete guidance, especially when combined with UDLM (Tables 5 and 6).

Table 4: Diffusion decoding is more controllable than AR for genomic sequences. Best values are **bolded**.

Model	Guidance	3-mer JS (↓)	6-mer JS (↓)	Disc. AUROC (↓)	F1 (↑)
Random	-	0.13	0.22	1.00	0.07
AR	D-CFG$_{\gamma=1}$	0.04	0.09	1.00	0.87
AR	D-CFG$_{\gamma=2}$	0.05	0.14	1.00	0.68
MDLM	D-CFG$_{\gamma=1}$	**0.03**	**0.07**	**0.61**	0.90
MDLM	D-CFG$_{\gamma=2}$	0.04	0.11	0.71	0.78
UDLM	D-CFG$_{\gamma=1}$	**0.03**	0.08	0.77	0.92
UDLM	D-CFG$_{\gamma=2}$	0.08	0.18	0.75	**0.96**

Baselines For guidance experiments, our primary baseline is the dominant approach for language modeling, AR. Our goal is to show that with similar, even slightly worse, language modeling performance, discrete diffusion models equipped with classifier-based and classifier-free guidance can outperform AR models on controlled generation. We compare to three flavors of guided AR models. The first is applying D-CFG to AR models. We also use the established control mechanisms of Plug-and-play language models (PPLM; Dathathri et al. (2019)) and FUDGE (Yang & Klein, 2021). To demonstrate the better performance of our D-CBG method, we compare to Gruver et al. (2024) (NOS) which can be viewed as an extension of PPLM to discrete diffusion models.

Hyperparameters For both our D-CFG and D-CBG methods we vary the strength of the γ parameter. Although the original FUDGE formulation simply uses $\gamma = 1$, we also perform a search for this baseline. For PPLM and NOS, we vary the parameters of the Langevin sampling that is performed to update models' hidden representation, namely the step size η, the fluency KL weight γ_{kl}, and the number of update steps n (see Dathathri et al. (2019) and Gruver et al. (2024) for more details). For all methods, we display the best performing hyperparameter configuration in the main table results, deferring the full sweep results to Appendix D.

Species-specific Genome Generation For genomic sequences, we evaluate D-CFG with diffusion models vs. with an AR model. We train models on sequences of length 32,768 nucleotides, using base-pair level tokenization and conditionally generate 64 sequences for each class. As quality measures for each species class, we compute the Jensen-Shannon (JS) distance between the k-mer frequencies of the generated sequences and those from the validation set, and report the mean JS across species (weighted by species frequency in the dataset), with smaller values indicating better k-mer distributional overlap between the ground truth and generated sequences (Sarkar et al., 2024). Additionally, we train a small classifier to distinguish between generated and validation set sequences and report the area under the receiver operator curve for this classifier (Disc. AUROC). Values closer to 0.5 indicate that the classifier is unable to distinguish between synthetic and true sequences (Sarkar et al., 2024). To measure the controllability, we train a separate classifier on the Species-10 dataset and measure macro F1 score of this oracle classifier on the generated sequences.

Results of this experiment are presented in Table 4. For reference, we also provide metrics for randomly generating sequences with nucleotide frequencies proportional to each species' validation subset. We find that both MDLM and UDLM are able to better generate sequences that match the desired control parameter, with higher F1 scores under the evaluation oracle relative to AR. Moreover, UDLM is able to outperform MDLM in satisfying this control. Importantly, we find that only UDLM is amenable to increasing the guidance parameter γ, where its metrics improves while AR and MDLM metrics degrade.

Finally, of note, the diffusion model generation for this experiment is accomplished with far fewer function evaluations compared to AR. Whereas AR must decode each of the 32,768 tokens, because MDLM and UDLM can decode multiple tokens in parallel, we generate with $T = 512$.

Molecular Property Maximization For the QM9 dataset, we investigate novel generation of sequences that maximize some property, either drug-likeness (QED; Bickerton et al. (2012)), see Table 5, or a count of the number of rings present in the molecule, see Table 6.

For D-CFG, we see that all methods perform comparably when maximizing drug likeness, but that UDLM is better suited towards guidance for the structural property of ring count, as it achieves similar property maximization to AR and MDLM, without sacrificing the validity and novelty of generated sequences. This pattern is even more evident when examining classifier-based guidance mechanisms, where we find that for both QED and ring count, UDLM best trades-off maximizing these properties while generating sensible and not memorized sequences.

Table 5: Guidance with discrete diffusion models better balances the generation of valid and novel molecules with maximizing the property of interest, drug likeness (QED), compared to AR. Validity, novelty, and mean QED for novel sequences are measured for generated sequences from each method. Best values are **bolded**.

Method	Guidance	Valid (↑)	Novel (↑)	QED Mean (↑)
Original Data	-	133k	133k	0.47
Classifier-free				
AR	D-CFG$_{\gamma=2.5}$	1995	130	0.60
MDLM	D-CFG$_{\gamma=3.0}$	652	251	**0.61**
UDLM	D-CFG$_{\gamma=3.0}$	**2034**	133	**0.61**
Classifier-based				
AR	FUDGE$_{\gamma=10}$	914	24	0.58
AR	PPLM$_{\eta=0.1,n=30}$	1618	291	0.48
MDLM	D-CBG$_{\gamma=20}$ (Ours)	64	22	0.58
MDLM	NOS$_{\eta=0.001,n=1,\gamma_{kl}=0.01}$	1302	**471**	0.45
UDLM	D-CBG$_{\gamma=35}$ (Ours)	1223	87	**0.61**
UDLM	NOS$_{\eta=5,n=5,\gamma_{kl}=0.001}$	946	302	0.47

Table 6: Guidance with UDLM best balances the generation of valid and novel molecules with maximizing the property of interest, ring count, compared to AR and MDLM. Validity, novelty, and mean ring count for novel sequences are measured for generated sequences from each method. Best values are **bolded**.

Method	Guidance	Valid (↑)	Novel (↑)	Ring Count Mean (↑)
Original Data	-	133k	133k	1.74
Classifier-free				
AR	D-CFG$_{\gamma=7.0}$	108	24	5.04
MDLM	D-CFG$_{\gamma=6.0}$	132	90	5.01
UDLM	D-CFG$_{\gamma=3.5}$	1975	459	4.91
Classifier-based				
AR	FUDGE$_{\gamma=2}$	**2035**	15	4.20
AR	PPLM$_{\eta=0.1,n=30}$	1486	233	1.89
MDLM	D-CBG$_{\gamma=25}$ (Ours)	82	65	**5.05**
MDLM	NOS$_{\eta=5,n=10,\gamma_{kl}=0.01}$	353	246	3.31
UDLM	D-CBG$_{\gamma=40}$ (Ours)	1670	**943**	4.73
UDLM	NOS$_{\eta=5,n=10,\gamma_{kl}=0.01}$	961	402	2.64

Class-conditional Image Generation In Table 7, we see the positive effect of guided-generation on image quality. Both MDLM and UDLM outperform their finite-time counterparts (in the form of D3PM (Austin et al., 2021)) and, moreover, generating with D-CFG ($\gamma = 4$) further improves the image quality metrics of Fréchet inception distance (FID; Heusel et al. (2017)) and Inception Score (IS; Salimans et al. (2016)). In Figure 2, we show several conditionally generated images.

Table 7: Guidance improves image on discretized CIFAR-10. FID and IS for finite- (D3PM) and continuous-time (MDLM / UDLM) discrete diffusion models. Guidance using D-CFG ($\gamma = 4$). Best values are **bolded**. †Taken from Austin et al. (2021).

	FID (↓)	IS (↑)
D3PM Absorb†	41.28	6.26
MDLM	33.75	6.74
MDLM D-CFG	15.56	**9.02**
D3PM Uniform†	51.27	5.99
UDLM	33.65	6.86
UDLM D-CFG	23.21	8.66

Figure 2: Conditionally generated images for each class in CIFAR10.

6 Related Works, Discussion, and Conclusion

Discrete Diffusion Our UDLM method most closely aligns to the recent state-of-the-art discrete diffusion models of Ou et al. (2024), Sahoo et al. (2024a), and Shi et al. (2024) that focus on absorbing state diffusion. Similar to these works, we provide a continuous time ELBO leading to performance gains, but we focus on uniform noise. Of note, other extensions of D3PM that do not start from the variational perspective, instead rely on the formalisms of continuous time Markov chains (CTMC) (Campbell et al., 2022) and concrete score matching (Lou et al., 2023), but are less performant than works such as MDLM (Sahoo et al., 2024a) and our method.

Guidance Leveraging continuous embeddings of discrete data, Diffusion-LM (Li et al., 2022) uses Langevin sampling with classifier-based guidance. Similarly, SSD-LM (Han et al., 2022) perform Gaussian noising on the logits of a bi-directional model, which they combine with pre-trained classifiers to perform classifier-based guidance with Langevin dynamics. LD4LG (Lovelace et al., 2024) implement classifier-free guidance on continuous embeddings. Wang et al. (2023) perform guidance using auxiliary semantic latent variables. In contrast, to these continuous formulations for discrete data, our work adapts guidance mechanisms directly to the discrete domain.

FUDGE (Yang & Klein, 2021) can be viewed an analog to our D-CBG, but is restricted to AR parameterizations of the denoising network. DiGress (Vignac et al., 2022) uses a similar first-order approximation to the one we employ. However, in DiGress this approximation is used to resolve the intractability of the normalizing constant. In our work, we derive a tractable expression for classifier-based guidance and simply use the Taylor approximation to speed up computation in large sequence length and vocabulary size regimes.

Sanchez et al. (2023) derives an equivalent formulation for D-CFG and use it to better enforce AR models' adherence to prefix prompts. FreeGress (Ninniri et al., 2024) also offer a comparable method to D-CFG, but focus on graph diffusion models. Most similar to our method, is the concurrent work of Nisonoff et al. (2024), which derives classifier-based and classifier-free guidance for diffusion and flow models. However, this work is highly tailored to models that leverage formalism of continuous time Markov chains and guidance is applied to the rate matrices.

Conclusion In search of a more controllable diffusion process, in this work, we derived a tight variational bound for uniform noise discrete diffusion, closing the gap to state-of-the-art absorbing-state diffusion models. We also highlighted that contrary to previous findings, in small vocabulary regimes, uniform noise is on par or better than absorbing state.

We then demonstrated that straightforward adaptations of classifier-based and classifier-free guidance can offer improved guided generation relative to AR models. We found that with classifier-free mechanisms, diffusion models are more amenable to control without sacrificing quality of generated sequences. We also demonstrated that our classifier-based method is better than previous ones for both AR and diffusion models and is best paired with our UDLM.

REFERENCES

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

G Richard Bickerton, Gaia V Paolini, Jérémy Besnard, Sorel Muresan, and Andrew L Hopkins. Quantifying the chemical beauty of drugs. *Nature chemistry*, 4(2):90–98, 2012.

Andrew Campbell, Joe Benton, Valentin De Bortoli, Thomas Rainforth, George Deligiannidis, and Arnaud Doucet. A continuous time framework for discrete denoising models. *Advances in Neural Information Processing Systems*, 35:28266–28279, 2022.

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.

Zihang Dai, Zhilin Yang, Yiming Yang, Jaime Carbonell, Quoc V Le, and Ruslan Salakhutdinov. Transformer-xl: Attentive language models beyond a fixed-length context. *arXiv preprint arXiv:1901.02860*, 2019.

Sumanth Dathathri, Andrea Madotto, Janice Lan, Jane Hung, Eric Frank, Piero Molino, Jason Yosinski, and Rosanne Liu. Plug and play language models: A simple approach to controlled text generation. *arXiv preprint arXiv:1912.02164*, 2019.

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

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

Ruiqi Gao, Aleksander Holynski, Philipp Henzler, Arthur Brussee, Ricardo Martin-Brualla, Pratul Srinivasan, Jonathan T. Barron, and Ben Poole. Cat3d: Create anything in 3d with multi-view diffusion models, 2024. URL https://arxiv.org/abs/2405.10314.

Aaron Gokaslan, A Feder Cooper, Jasmine Collins, Landan Seguin, Austin Jacobson, Mihir Patel, Jonathan Frankle, Cory Stephenson, and Volodymyr Kuleshov. Commoncanvas: Open diffusion models trained on creative-commons images. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pp. 8250–8260, 2024.

Will Grathwohl, Kevin Swersky, Milad Hashemi, David Duvenaud, and Chris Maddison. Oops i took a gradient: Scalable sampling for discrete distributions. In *International Conference on Machine Learning*, pp. 3831–3841. PMLR, 2021.

Alex Graves, Rupesh Kumar Srivastava, Timothy Atkinson, and Faustino Gomez. Bayesian flow networks. *arXiv preprint arXiv:2308.07037*, 2023.

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, 2024.

Albert Gu and Tri Dao. Mamba: Linear-time sequence modeling with selective state spaces. *arXiv preprint arXiv:2312.00752*, 2023.

Xiaochuang Han, Sachin Kumar, and Yulia Tsvetkov. Ssd-lm: Semi-autoregressive simplex-based diffusion language model for text generation and modular control. *arXiv preprint arXiv:2210.17432*, 2022.

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. *Advances in neural information processing systems*, 30, 2017.

Jonathan Ho and Tim Salimans. Classifier-free diffusion guidance. *arXiv preprint arXiv:2207.12598*, 2022.

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

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

Frank P Kelly. *Reversibility and stochastic networks*. Cambridge University Press, 2011.

Diederik Kingma, Tim Salimans, Ben Poole, and Jonathan Ho. Variational diffusion models. *Advances in neural information processing systems*, 34:21696–21707, 2021.

Alex Krizhevsky, Geoffrey Hinton, et al. Learning multiple layers of features from tiny images. 2009.

Greg Landrum et al. Rdkit: A software suite for cheminformatics, computational chemistry, and predictive modeling. *Greg Landrum*, 8(31.10):5281, 2013.

Xiang Li, John Thickstun, Ishaan Gulrajani, Percy S Liang, and Tatsunori B Hashimoto. Diffusion-lm improves controllable text generation. *Advances in Neural Information Processing Systems*, 35:4328–4343, 2022.

Aaron Lou, Chenlin Meng, and Stefano Ermon. Discrete diffusion language modeling by estimating the ratios of the data distribution. *arXiv preprint arXiv:2310.16834*, 2023.

Justin Lovelace, Varsha Kishore, Chao Wan, Eliot Shekhtman, and Kilian Q Weinberger. Latent diffusion for language generation. *Advances in Neural Information Processing Systems*, 36, 2024.

Matt Mahoney. Text8 dataset, 2011. URL http://mattmahoney.net/dc/textdata.

Julian McAuley and Jure Leskovec. Hidden factors and hidden topics: understanding rating dimensions with review text. In *Proceedings of the 7th ACM conference on Recommender systems*, pp. 165–172, 2013.

Chenlin Meng, Kristy Choi, Jiaming Song, and Stefano Ermon. Concrete score matching: Generalized score matching for discrete data. *Advances in Neural Information Processing Systems*, 35: 34532–34545, 2022.

Alex Nichol, Prafulla Dhariwal, Aditya Ramesh, Pranav Shyam, Pamela Mishkin, Bob McGrew, Ilya Sutskever, and Mark Chen. Glide: Towards photorealistic image generation and editing with text-guided diffusion models. *arXiv preprint arXiv:2112.10741*, 2021.

Matteo Ninniri, Marco Podda, and Davide Bacciu. Classifier-free graph diffusion for molecular property targeting. In *Joint European Conference on Machine Learning and Knowledge Discovery in Databases*, pp. 318–335. Springer, 2024.

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.

Nuala A O'Leary, Mathew W Wright, J Rodney Brister, Stacy Ciufo, Diana Haddad, Rich McVeigh, Bhanu Rajput, Barbara Robbertse, Brian Smith-White, Danso Ako-Adjei, Alexander Astashyn, Azat Badretdin, Yiming Bao, Olga Blinkova, Vyacheslav Brover, Vyacheslav Chetvernin, Jinna Choi, Eric Cox, Olga Ermolaeva, Catherine M Farrell, Tamara Goldfarb, Tripti Gupta, Daniel Haft, Eneida Hatcher, Wratko Hlavina, Vinita S Joardar, Vamsi K Kodali, Wenjun Li, Donna Maglott, Patrick Masterson, Kelly M McGarvey, Michael R Murphy, Kathleen O'Neill, Shashikant Pujar, Sanjida H Rangwala, Daniel Rausch, Lillian D Riddick, Conrad Schoch, Andrei Shkeda, Susan S Storz, Hanzhen Sun, Francoise Thibaud-Nissen, Igor Tolstoy, Raymond E Tully, Anjana R Vatsan, Craig Wallin, David Webb, Wendy Wu, Melissa J Landrum, Avi Kimchi, Tatiana Tatusova, Michael DiCuccio, Paul Kitts, Terence D Murphy, and Kim D Pruitt. Reference sequence (RefSeq) database at NCBI: current status, taxonomic expansion, and functional annotation. *Nucleic Acids Res.*, 44(D1):D733–45, January 2016.

Jingyang Ou, Shen Nie, Kaiwen Xue, Fengqi Zhu, Jiacheng Sun, Zhenguo Li, and Chongxuan Li. Your absorbing discrete diffusion secretly models the conditional distributions of clean data. *arXiv preprint arXiv:2406.03736*, 2024.

William Peebles and Saining Xie. Scalable diffusion models with transformers. In *Proceedings of the IEEE/CVF International Conference on Computer Vision*, pp. 4195–4205, 2023.

Ben Poole, Ajay Jain, Jonathan T Barron, and Ben Mildenhall. Dreamfusion: Text-to-3d using 2d diffusion. *arXiv preprint arXiv:2209.14988*, 2022.

Raghunathan Ramakrishnan, Pavlo O Dral, Matthias Rupp, and O Anatole Von Lilienfeld. Quantum chemistry structures and properties of 134 kilo molecules. *Scientific data*, 1(1):1–7, 2014.

Olaf Ronneberger, Philipp Fischer, and Thomas Brox. U-net: Convolutional networks for biomedical image segmentation. In *Medical image computing and computer-assisted intervention–MICCAI 2015: 18th international conference, Munich, Germany, October 5-9, 2015, proceedings, part III 18*, pp. 234–241. Springer, 2015.

Lars Ruddigkeit, Ruud Van Deursen, Lorenz C Blum, and Jean-Louis Reymond. Enumeration of 166 billion organic small molecules in the chemical universe database gdb-17. *Journal of chemical information and modeling*, 52(11):2864–2875, 2012.

Chitwan Saharia, William Chan, Saurabh Saxena, Lala Li, Jay Whang, Emily L Denton, Kamyar Ghasemipour, Raphael Gontijo Lopes, Burcu Karagol Ayan, Tim Salimans, et al. Photorealistic text-to-image diffusion models with deep language understanding. *Advances in neural information processing systems*, 35:36479–36494, 2022.

Subham Sekhar Sahoo, Anselm Paulus, Marin Vlastelica, Vít Musil, Volodymyr Kuleshov, and Georg Martius. Backpropagation through combinatorial algorithms: Identity with projection works. In *The Eleventh International Conference on Learning Representations*, 2023. URL https://openreview.net/forum?id=JZMR727O29.

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. *arXiv preprint arXiv:2406.07524*, 2024a.

Subham Sekhar Sahoo, Aaron Gokaslan, Chris De Sa, and Volodymyr Kuleshov. Diffusion models with learned adaptive noise. *arXiv preprint arXiv:2312.13236*, 2024b.

Tim Salimans, Ian Goodfellow, Wojciech Zaremba, Vicki Cheung, Alec Radford, and Xi Chen. Improved techniques for training gans. *Advances in neural information processing systems*, 29, 2016.

Guillaume Sanchez, Honglu Fan, Alexander Spangher, Elad Levi, Pawan Sasanka Ammanamanchi, and Stella Biderman. Stay on topic with classifier-free guidance. *arXiv preprint arXiv:2306.17806*, 2023.

Anirban Sarkar, Ziqi Tang, Chris Zhao, and Peter Koo. Designing dna with tunable regulatory activity using discrete diffusion. *bioRxiv*, pp. 2024–05, 2024.

Yair Schiff, Chia-Hsiang Kao, Aaron Gokaslan, Tri Dao, Albert Gu, and Volodymyr Kuleshov. Caduceus: Bi-directional equivariant long-range dna sequence modeling. *arXiv preprint arXiv:2403.03234*, 2024.

Philippe Schwaller, Teodoro Laino, Théophile Gaudin, Peter Bolgar, Christopher A Hunter, Costas Bekas, and Alpha A Lee. Molecular transformer: A model for uncertainty-calibrated chemical reaction prediction. *ACS central science*, 5(9):1572–1583, 2019.

Jiaxin Shi, Kehang Han, Zhe Wang, Arnaud Doucet, and Michalis K Titsias. Simplified and generalized masked diffusion for discrete data. *arXiv preprint arXiv:2406.04329*, 2024.

Andy Shih, Dorsa Sadigh, and Stefano Ermon. Training and inference on any-order autoregressive models the right way. *Advances in Neural Information Processing Systems*, 35:2762–2775, 2022.

Jascha Sohl-Dickstein, Eric Weiss, Niru Maheswaranathan, and Surya Ganguli. Deep unsupervised learning using nonequilibrium thermodynamics. In *International conference on machine learning*, pp. 2256–2265. PMLR, 2015.

Yang Song and Stefano Ermon. Generative modeling by estimating gradients of the data distribution. *Advances in neural information processing systems*, 32, 2019.

Jianlin Su, Yu Lu, Shengfeng Pan, Ahmed Murtadha, Bo Wen, and Yunfeng Liu. Roformer: Enhanced transformer with rotary position embedding. *arXiv preprint arXiv:2104.09864*, 2021.

Haoran Sun, Lijun Yu, Bo Dai, Dale Schuurmans, and Hanjun Dai. Score-based continuous-time discrete diffusion models. *arXiv preprint arXiv:2211.16750*, 2022.

Yi Tay, Mostafa Dehghani, Vamsi Aribandi, Jai Gupta, Philip M Pham, Zhen Qin, Dara Bahri, Da-Cheng Juan, and Donald Metzler. Omninet: Omnidirectional representations from transformers. In *International Conference on Machine Learning*, pp. 10193–10202. PMLR, 2021.

Dustin Tran, Keyon Vafa, Kumar Agrawal, Laurent Dinh, and Ben Poole. Discrete flows: Invertible generative models of discrete data. *Advances in Neural Information Processing Systems*, 32, 2019.

Clement Vignac, Igor Krawczuk, Antoine Siraudin, Bohan Wang, Volkan Cevher, and Pascal Frossard. Digress: Discrete denoising diffusion for graph generation. *arXiv preprint arXiv:2209.14734*, 2022.

Alex Wang and Kyunghyun Cho. Bert has a mouth, and it must speak: Bert as a markov random field language model. *arXiv preprint arXiv:1902.04094*, 2019.

Yingheng Wang, Yair Schiff, Aaron Gokaslan, Weishen Pan, Fei Wang, Christopher De Sa, and Volodymyr Kuleshov. Infodiffusion: Representation learning using information maximizing diffusion models. In *International Conference on Machine Learning*, pp. 36336–36354. PMLR, 2023.

David Weininger. Smiles, a chemical language and information system. 1. introduction to methodology and encoding rules. *Journal of chemical information and computer sciences*, 28(1):31–36, 1988.

Thomas Wolf, Lysandre Debut, Victor Sanh, Julien Chaumond, Clement Delangue, Anthony Moi, Pierric Cistac, Tim Rault, Rémi Louf, Morgan Funtowicz, Joe Davison, Sam Shleifer, Patrick von Platen, Clara Ma, Yacine Jernite, Julien Plu, Canwen Xu, Teven Le Scao, Sylvain Gugger, Mariama Drame, Quentin Lhoest, and Alexander M. Rush. Transformers: State-of-the-art natural language processing. In *Proceedings of the 2020 Conference on Empirical Methods in Natural Language Processing: System Demonstrations*, pp. 38–45, Online, October 2020. Association for Computational Linguistics. URL https://www.aclweb.org/anthology/2020.emnlp-demos.6.

Kevin Yang and Dan Klein. Fudge: Controlled text generation with future discriminators. *arXiv preprint arXiv:2104.05218*, 2021.

Xiang Zhang, Junbo Zhao, and Yann LeCun. Character-level convolutional networks for text classification. *Advances in neural information processing systems*, 28, 2015.

Zachary Ziegler and Alexander Rush. Latent normalizing flows for discrete sequences. In *International Conference on Machine Learning*, pp. 7673–7682. PMLR, 2019.

A CONTINUOUS TIME DISCRETE UNIFORM DIFFUSION

Here, we derive a continuous time formulation ($T \to \infty$) for the diffusion loss term $\mathcal{L}_{\text{diffusion}}$ when using the uniform distribution as the limiting distribution of the diffusion process.

For uniform noise diffusion, we define a limiting distribution $\boldsymbol{u} = \mathbf{1}/N$, where $\mathbf{1}$ represents the column vector of all ones, and N the size of the vocabulary. We adopt the assumption from MDLM (Sahoo et al., 2024a) that our diffusion process interpolates between clean data and noise:

$$q(\mathbf{z}_t \mid \mathbf{x}) = \text{Cat}(\mathbf{z}_t; \alpha_t \mathbf{x} + (1 - \alpha_t)\boldsymbol{\pi}) \tag{15}$$

and the marginals are given as

$$q(\mathbf{z}_t \mid \mathbf{z}_s) = \text{Cat}(\mathbf{z}_t; \alpha_{t|s} \mathbf{z}_s + (1 - \alpha_{t|s})\boldsymbol{\pi}), \tag{16}$$

where $\alpha_{t|s} = \alpha_t / \alpha_s$.

Following Austin et al. (2021), we can derive the posterior:

$$q(\mathbf{z}_s \mid \mathbf{z}_t, \mathbf{x}) = \text{Cat}\left(\mathbf{z}_s; \frac{[\alpha_{t|s}\mathbf{z}_t + (1 - \alpha_{t|s})\mathbf{1}\boldsymbol{\pi}^\top \mathbf{z}_t] \odot [\alpha_s \mathbf{x} + (1 - \alpha_s)\boldsymbol{\pi}]}{\alpha_t \langle \mathbf{z}_t, \mathbf{x} \rangle + (1 - \alpha_t)\mathbf{z}_t^\top \boldsymbol{\pi}}\right). \tag{17}$$

Using $\boldsymbol{\pi} = \boldsymbol{u}$ as defined above, we get the following:

$$q(\mathbf{z}_s \mid \mathbf{z}_t, \mathbf{x}) = \text{Cat}\left(\mathbf{z}_s; \frac{\alpha_t \mathbf{z}_t \odot \mathbf{x} + \frac{(\alpha_{t|s} - \alpha_t)}{N}\mathbf{z}_t + \frac{(\alpha_s - \alpha_t)}{N}\mathbf{x} + \frac{(1 - \alpha_{t|s})(1 - \alpha_s)}{N^2}\mathbf{1}}{\alpha_t \langle \mathbf{z}_t, \mathbf{x} \rangle + \frac{(1 - \alpha_t)}{N}}\right) \tag{18}$$

$$= \text{Cat}\left(\mathbf{z}_s; \frac{N\alpha_t \mathbf{z}_t \odot \mathbf{x} + (\alpha_{t|s} - \alpha_t)\mathbf{z}_t + (\alpha_s - \alpha_t)\mathbf{x} + \frac{(\alpha_s - \alpha_t)(1 - \alpha_s)}{N\alpha_s}\mathbf{1}}{N\alpha_t \langle \mathbf{z}_t, \mathbf{x} \rangle + 1 - \alpha_t}\right). \tag{19}$$

For the denoising distribution, we replace $\mathbf{x}$ by $\mathbf{x}_\theta$:

$$p_\theta(\mathbf{z}_s \mid \mathbf{z}_t) = \text{Cat}\left(\mathbf{z}_s; \frac{N\alpha_t \mathbf{z}_t \odot \mathbf{x}_\theta + (\alpha_{t|s} - \alpha_t)\mathbf{z}_t + (\alpha_s - \alpha_t)\mathbf{x}_\theta + \frac{(\alpha_s - \alpha_t)(1 - \alpha_s)}{\alpha_s N}\mathbf{1}}{N\alpha_t \langle \mathbf{z}_t, \mathbf{x}_\theta \rangle + 1 - \alpha_t}\right). \tag{20}$$

Let us now look at the diffusion loss term in the NELBO:

$$T \cdot D_{\text{KL}}(q(\mathbf{z}_s \mid \mathbf{z}_t, \mathbf{x}) || p_\theta(\mathbf{z}_s \mid \mathbf{z}_t)) = T \cdot \sum_{j \in [N]} q(\mathbf{z}_s \mid \mathbf{z}_t, \mathbf{x})_j \log\left(\frac{q(\mathbf{z}_s \mid \mathbf{z}_t, \mathbf{x})_j}{p_\theta(\mathbf{z}_s \mid \mathbf{z}_t)_j}\right). \tag{21}$$

Letting $i = \arg\max_{j \in [N]} (\mathbf{z}_t)_j$ be the non-zero entry of $\mathbf{z}_t$, we can break up this KL into two terms:

$$T \cdot D_{\text{KL}}(q(\mathbf{z}_s \mid \mathbf{z}_t, \mathbf{x}) || p_\theta(\mathbf{z}_s \mid \mathbf{z}_t)) = \underbrace{T \cdot q(\mathbf{z}_s \mid \mathbf{z}_t, \mathbf{x})_i \log\left(\frac{q(\mathbf{z}_s \mid \mathbf{z}_t, \mathbf{x})_i}{p_\theta(\mathbf{z}_s \mid \mathbf{z}_t)_i}\right)}_{\text{Term 1}}$$

$$\underbrace{+ T \cdot \sum_{\substack{j \in [N] \\ \text{s.t. } (\mathbf{z}_t)_j = 0}} q(\mathbf{z}_s \mid \mathbf{z}_t, \mathbf{x})_j \log\left(\frac{q(\mathbf{z}_s \mid \mathbf{z}_t, \mathbf{x})_j}{p_\theta(\mathbf{z}_s \mid \mathbf{z}_t)_j}\right)}_{\text{Term 2}}. \tag{22}$$

We now examine each of these terms taking $T \to \infty$, or equivalently, using $s = t - \frac{1}{T} \implies T = \frac{1}{t - s}$, taking $s \to t$.

Term 1:

$$\lim_{s \to t} \frac{1}{t - s} \cdot \frac{N\alpha_t \mathbf{x}_i + \alpha_{t|s} - \alpha_t + (\alpha_s - \alpha_t)\mathbf{x}_i + \frac{(\alpha_s - \alpha_t)(1 - \alpha_s)}{\alpha_s N}}{N\alpha_t \mathbf{x}_i + 1 - \alpha_t}$$

$$\cdot \log\left[\frac{\frac{N\alpha_t \mathbf{x}_i + \alpha_{t|s} - \alpha_t + (\alpha_s - \alpha_t)\mathbf{x}_i + \frac{(\alpha_s - \alpha_t)(1 - \alpha_s)}{\alpha_s N}}{N\alpha_t \mathbf{x}_i + 1 - \alpha_t}}{\frac{N\alpha_t (\mathbf{x}_\theta)_i + \alpha_{t|s} - \alpha_t + (\alpha_s - \alpha_t)(\mathbf{x}_\theta)_i + \frac{(\alpha_s - \alpha_t)(1 - \alpha_s)}{\alpha_s N}}{N\alpha_t (\mathbf{x}_\theta)_i + 1 - \alpha_t}}\right] \tag{23}$$

As $s \to t$, the coefficient on the log term will approach 1. Additionally both the numerator and the denominator inside the log term will approach 1, thus the entire log term will approach 0. Combining this with the fact that $t - s$ will approach 0, gives us the indeterminate form of $0/0$, hence we apply L'Hôpital's's rule to (23). Writing q and p_θ as functions of s, when we differentiate the log term we get:

$$\begin{aligned}\frac{d}{ds}\log\left[\frac{q(s)}{p_\theta(s)}\right] &= \frac{d}{ds}\log q(s) - \frac{d}{ds}\log p_\theta(s) \\ &= \frac{\frac{d}{ds}q(s)}{q(s)} - \frac{\frac{d}{ds}p_\theta(s)}{p_\theta(s)}\end{aligned} \tag{24}$$

Let's look at each derivative term in (24):

$$\begin{aligned}\lim_{s\to t}\frac{d}{ds}q(s) &= \lim_{s\to t}\frac{\frac{-\alpha_s'\alpha_t}{\alpha_s^2} + \alpha_s'(\mathbf{x}_\theta)_i + \frac{N\alpha_s[\alpha_s'(1-\alpha_s)-\alpha_s'(\alpha_s-\alpha_t)]-[N\alpha_s'(\alpha_s-\alpha_t)(1-\alpha_s)]}{N\alpha_s^2}}{N\alpha_t\mathbf{x}_i + 1 - \alpha_t} \\ &= \frac{\frac{-\alpha_t'}{\alpha_t} + \alpha_t'\mathbf{x}_i + \frac{\alpha_t'(1-\alpha_t)}{N\alpha_t}}{N\alpha_t\mathbf{x}_i + 1 - \alpha_t} \\ &= \frac{\alpha_t'}{N\alpha_t}\left[\frac{-N + N\alpha_t\mathbf{x}_i + 1 - \alpha_t}{N\alpha_t\mathbf{x}_i + 1 - \alpha_t}\right] \\ &= \frac{\alpha_t'}{N\alpha_t}\left[1 - \frac{N}{N\alpha_t\mathbf{x}_i + 1 - \alpha_t}\right]\end{aligned} \tag{25}$$

Similarly,

$$\lim_{s\to t}\frac{d}{ds}p_\theta(s) = \frac{\alpha_t'}{N\alpha_t}\left[1 - \frac{N}{N\alpha_t(\mathbf{x}_\theta)_i + 1 - \alpha_t}\right] \tag{26}$$

Note that when taking $s \to t$, both $q(s)$ and $p_\theta(s)$ evaluate to 1. Additionally, differentiating $\frac{1}{t-s}$ with respect to s evaluates to -1. When combining these facts and plugging (25) and (26) into (24), (23) becomes

$$\begin{aligned}\lim_{s\to t}\text{Term 1} &= \frac{-\alpha_t'}{N\alpha_t}\left[1 - \frac{N}{N\alpha_t\mathbf{x}_i + 1 - \alpha_t}\right] + \frac{\alpha_t'}{N\alpha_t}\left[1 - \frac{N}{N\alpha_t(\mathbf{x}_\theta)_i + 1 - \alpha_t}\right] \\ &= \boxed{\frac{\alpha_t'}{N\alpha_t}\left[\frac{N}{N\alpha_t\mathbf{x}_i + 1 - \alpha_t} - \frac{N}{N\alpha_t(\mathbf{x}_\theta)_i + 1 - \alpha_t}\right]}\end{aligned} \tag{27}$$

Term 2:

$$\lim_{s\to t}\frac{1}{t-s}\cdot\sum_{\substack{j\in[N]\\ \text{s.t. } (\mathbf{z}_t)_j=0}}\frac{(\alpha_s-\alpha_t)\mathbf{x}_j + \frac{(\alpha_s-\alpha_t)(1-\alpha_s)}{N\alpha_s}}{N\alpha_t\mathbf{x}_i + 1 - \alpha_t}\log\left[\frac{\frac{(\alpha_s-\alpha_t)\mathbf{x}_j + \frac{(\alpha_s-\alpha_t)(1-\alpha_s)}{N\alpha_s}}{N\alpha_t\mathbf{x}_i + 1 - \alpha_t}}{\frac{(\alpha_s-\alpha_t)(\mathbf{x}_\theta)_j + \frac{(\alpha_s-\alpha_t)(1-\alpha_t)}{N\alpha_s}}{N\alpha_t(\mathbf{x}_\theta)_i + 1 - \alpha_t}}\right] \tag{28}$$

For each term in the summation, $\frac{1}{t-s}$ times the coefficient on the log term will have an indeterminate form of $0/0$ as $s \to t$. We therefore apply L'Hôpital's rule to this coefficient:

$$\lim_{s\to t}\frac{1}{t-s}\cdot\frac{(\alpha_s-\alpha_t)\mathbf{x}_j + \frac{(\alpha_s-\alpha_t)(1-\alpha_s)}{N\alpha_s}}{N\alpha_t\mathbf{x}_i + 1 - \alpha_t} = \frac{-\alpha_t'}{N\alpha_t}\left[\frac{N\alpha_t\mathbf{x}_j + 1 - \alpha_t}{N\alpha_t\mathbf{x}_i + 1 - \alpha_t}\right] \tag{29}$$

Now, for the log term, we exchange the limit with the continuous log function and have that both the numerator and the denominator go to zero as $s \to t$. We therefore apply L'Hôpital's rule here as

well:

$$\log \lim_{s\to t}\left[\frac{\frac{(\alpha_s-\alpha_t)\mathbf{x}_j+\frac{(\alpha_s-\alpha_t)(1-\alpha_s)}{N\alpha_s}}{N\alpha_t\mathbf{x}_i+1-\alpha_t}}{\frac{(\alpha_s-\alpha_t)(\mathbf{x}_\theta)_j+\frac{(\alpha_s-\alpha_t)(1-\alpha_t)}{N\alpha_s}}{N\alpha_t(\mathbf{x}_\theta)_i+1-\alpha_t}}\right] = \log \lim_{s\to t}\left[\frac{\frac{d}{ds}\frac{(\alpha_s-\alpha_t)\mathbf{x}_j+\frac{(\alpha_s-\alpha_t)(1-\alpha_s)}{N\alpha_s}}{N\alpha_t\mathbf{x}_i+1-\alpha_t}}{\frac{d}{ds}\frac{(\alpha_s-\alpha_t)(\mathbf{x}_\theta)_j+\frac{(\alpha_s-\alpha_t)(1-\alpha_t)}{N\alpha_s}}{N\alpha_t(\mathbf{x}_\theta)_i+1-\alpha_t}}\right]$$

$$= \log\left[\left(\frac{N\alpha_t(\mathbf{x}_\theta)_i+1-\alpha_t}{N\alpha_t\mathbf{x}_i+1-\alpha_t}\right)\left(\frac{\alpha_t'\mathbf{x}_j+\frac{\alpha_t'(1-\alpha_t)}{N\alpha_t}}{\alpha_t'(\mathbf{x}_\theta)_j+\frac{\alpha_t'(1-\alpha_t)}{N\alpha_t}}\right)\right]$$

$$= \log\left[\left(\frac{N\alpha_t(\mathbf{x}_\theta)_i+1-\alpha_t}{N\alpha_t\mathbf{x}_i+1-\alpha_t}\right)\left(\frac{\frac{\alpha_t'}{N\alpha_t}(N\alpha_t\mathbf{x}_j+1-\alpha_t)}{\frac{\alpha_t'}{N\alpha_t}(N\alpha_t(\mathbf{x}_\theta)_j+1-\alpha_t)}\right)\right]$$

$$= \log\left[\left(\frac{N\alpha_t(\mathbf{x}_\theta)_i+1-\alpha_t}{N\alpha_t\mathbf{x}_i+1-\alpha_t}\right)\left(\frac{N\alpha_t\mathbf{x}_j+1-\alpha_t}{N\alpha_t(\mathbf{x}_\theta)_j+1-\alpha_t}\right)\right]$$

$$= \log\left[\left(\frac{N\alpha_t(\mathbf{x}_\theta)_i+1-\alpha_t}{N\alpha_t(\mathbf{x}_\theta)j+1-\alpha_t}\right)\left(\frac{N\alpha_t\mathbf{x}_j+1-\alpha_t}{N\alpha_t\mathbf{x}_i+1-\alpha_t}\right)\right] \tag{30}$$

Multiplying (29) by (30), we get:

$$\lim_{s\to t}\text{Term 2} = \boxed{\frac{-\alpha_t'}{N\alpha_t}\sum_{\substack{j\in[N]\\ \text{s.t. }(\mathbf{z}_t)_j=0}}\left(\frac{N\alpha_t\mathbf{x}_j+1-\alpha_t}{N\alpha_t\mathbf{x}_i+1-\alpha_t}\right)\log\left[\left(\frac{N\alpha_t(\mathbf{x}_\theta)_i+1-\alpha_t}{N\alpha_t(\mathbf{x}_\theta)j+1-\alpha_t}\right)\left(\frac{N\alpha_t\mathbf{x}_j+1-\alpha_t}{N\alpha_t\mathbf{x}_i+1-\alpha_t}\right)\right]} \tag{31}$$

Combining Terms 1 and 2: Using (27) and (31), the final KL term in the continuous time limit is:

$$\lim_{T\to\infty} T\cdot D_{\mathrm{KL}}(q||p_\theta) = \frac{\alpha_t'}{N\alpha_t}\left[\frac{N}{N\alpha_t\mathbf{x}_i+1-\alpha_t}-\frac{N}{N\alpha_t(\mathbf{x}_\theta)_i+1-\alpha_t}\right.$$
$$\left.-\sum_{\substack{j\in[N]\\ \text{s.t. }(\mathbf{z}_t)_j=0}}\left(\frac{N\alpha_t\mathbf{x}_j+1-\alpha_t}{N\alpha_t\mathbf{x}_i+1-\alpha_t}\right)\log\left[\left(\frac{N\alpha_t(\mathbf{x}_\theta)_i+1-\alpha_t}{N\alpha_t(\mathbf{x}_\theta)j+1-\alpha_t}\right)\left(\frac{N\alpha_t\mathbf{x}_j+1-\alpha_t}{N\alpha_t\mathbf{x}_i+1-\alpha_t}\right)\right]\right].$$

Defining $\bar{\mathbf{x}} = N\alpha_t\mathbf{x}+(1-\alpha_t)\mathbf{1}$ and $\bar{\mathbf{x}}_\theta = N\alpha_t\mathbf{x}_\theta+(1-\alpha_t)\mathbf{1}$, as in Section 3.2, yields the desired result.

B Relating UDLM to CTMC

Similar to Sahoo et al. (2024a), our work tackles the problem of discrete diffusion from the variational perspective and analyzes the ELBO in the continuous time limit. In contrast, other works, such as Campbell et al. (2022) and Lou et al. (2023), have extended the discrete diffusion framework proposed in Austin et al. (2021) using the formalisms of continuous time Markov chains (CTMC). In this section, we relate these two approaches.

Background on CTMC In the CTMC formulation, the key quantity of interest is the **rate matrix** $R_t \in \mathbb{R}^{N\times N}$. In analogy to the discrete time transition matrices Q_t, which define the transition probabilities between states, these rate matrices define the instantaneous rate of change between states in continuous time. More formally, letting δ be the Kronecker delta function, and using $\mathbf{z}$ and $\mathbf{z}'$ to denote observed values of the latents, we can define the forward noising process using R_t as follows:

$$q(\mathbf{z}_t=\mathbf{z}' \mid \mathbf{z}_s=\mathbf{z}) = \delta_{\mathbf{z}',\mathbf{z}} + R_t(\mathbf{z},\mathbf{z}')\frac{1}{T} + o\left(\frac{1}{T}\right) \tag{32}$$

where $o(1/T)$ indicates terms that vanish more quickly than $1/T$. Recall that we defined $s = t-\frac{1}{T}$. In continuous time, as $T\to\infty$, $o(1/T)$ terms are ignored, and we have

$$R_t(\mathbf{z},\mathbf{z}') = \lim_{T\to\infty}\frac{q(\mathbf{z}_t=\mathbf{z}' \mid \mathbf{z}_{t-(1/T)}=\mathbf{z})-\delta_{\mathbf{z}',\mathbf{z}}}{1/T}, \tag{33}$$

hence our treatment of R_t as the 'instantaneous' rate of state transitions.

Importantly, processes defined as in (32) have known time reversal processes given by (Kelly, 2011; Sun et al., 2022):

$$q(\mathbf{z}_s = \mathbf{z}' \mid \mathbf{z}_t = \mathbf{z}) = \delta_{\mathbf{z}',\mathbf{z}} + \check{R}_t(\mathbf{z}, \mathbf{z}')\frac{1}{T} + o\Big(\frac{1}{T}\Big), \tag{34}$$

where $\check{R}_t$ represents the reverse rate matrix which is related to the forward matrix as follows:

$$\check{R}_t(\mathbf{z}', \mathbf{z}) = \frac{q(\mathbf{z}_t = \mathbf{z}')}{q(\mathbf{z}_t = \mathbf{z})} R_t(\mathbf{z}, \mathbf{z}'), \quad \text{if } \mathbf{z}' \neq \mathbf{z} \tag{35}$$

with $\check{R}_t(\mathbf{z}, \mathbf{z}) = -\sum_{\mathbf{z}' \neq \mathbf{z}} \check{R}_t(\mathbf{z}', \mathbf{z})$, which ensures that the rows of $\check{R}_t$ sum to zero (i.e., mass cannot be created or destroyed). Note that in (35), we are scaling the forward rate matrix by a ratio of the unconditional marginals: $q(\mathbf{z}_t = \mathbf{z}')/q(\mathbf{z}_t = \mathbf{z})$.

Forward Rate Matrices for Uniform Noise Recall from the analysis in Appendix A that we can use L'Hospital's rule to evaluate $\lim_{T\to\infty} T \cdot (1 - \alpha_{t|s}) = -\alpha'_t/\alpha_t$. Now, using (16) from above, for $\mathbf{z}' \neq \mathbf{z}$ we have

$$q(\mathbf{z}_t = \mathbf{z}' \mid \mathbf{z}_s = \mathbf{z}) = \frac{1 - \alpha_{t|s}}{N}. \tag{36}$$

Combining this with (33), we have that:

$$R_t(\mathbf{z}, \mathbf{z}') = \lim_{T\to\infty} T \cdot \frac{1 - \alpha_{t|s}}{N} = -\frac{\alpha'_t}{N\alpha_t}. \tag{37}$$

Now for $\mathbf{z}' = \mathbf{z}$, again from (16), we have

$$q(\mathbf{z}_t = \mathbf{z} \mid \mathbf{z}_s = \mathbf{z}) = \alpha_{t|s} + \frac{1 - \alpha_{t|s}}{N}, \tag{38}$$

which combines with (33) to yield:

$$R_t(\mathbf{z}, \mathbf{z}) = \lim_{T\to\infty} T \cdot \left(\alpha_{t|s} + \frac{1 - \alpha_{t|s}}{N} - 1\right) = \frac{1 - N}{N} \lim_{T\to\infty} T \cdot (1 - \alpha_{t|s}) = \frac{-\alpha'_t}{N\alpha_t}(1 - N). \tag{39}$$

Writing (37) and (39) as a single expression, gives:

$$R_t(\mathbf{z}, \mathbf{z}') = -\frac{\alpha'_t}{N\alpha_t}[\mathbf{1}\mathbf{1}^\top - N\mathbf{I}]. \tag{40}$$

B.1 Equivalence of UDLM ELBO and SEDD ELBO

Below, we demonstrate that the variational lower bound from (10) used to train UDLM is equivalent to the lower bound derived in SEDD (Lou et al., 2023).

Notation To facilitate the discussion, we introduce a notational shorthand $q_t(\mathbf{z}') = q(\mathbf{z}_t = \mathbf{z}')$ and $q_t(\mathbf{z}' \mid \mathbf{x}) = q(\mathbf{z}_t = \mathbf{z}' \mid \mathbf{x})$.

In SEDD, the quantity of interest is the ratio of probabilities in the reverse rate matrix equation given in (35), and they train a parametric model to learn this so-called concrete score (Meng et al., 2022):

$$s_\theta(\mathbf{z})_{\mathbf{z}'} \approx \frac{q_t(\mathbf{z}')}{q_t(\mathbf{z})}. \tag{41}$$

Since the unconditional marginals in this ratio are intractable, SEDD proposes a tractable denoising score-based objective. Importantly, they show that the denoising score-based objective they use for training serves as variational bound and derive the following expression for Negative Evidence Lower Bound (NELBO):

$$\begin{aligned}&\text{NELBO}_{\text{SEDD}} \\ &= \mathbb{E}_{t\in[0,1],\mathbf{z}\sim q_t(.|\mathbf{x})}\left[\sum_{\mathbf{z}'\neq\mathbf{z}_t} R_t(\mathbf{z}, \mathbf{z}')\left(\mathbf{s}_\theta(\mathbf{z})_{\mathbf{z}'} - \frac{q_t(\mathbf{z}'|\mathbf{x})}{q_t(\mathbf{z}|\mathbf{x})}\log \mathbf{s}_\theta(\mathbf{z})_{\mathbf{z}'} + K\left(\frac{q_t(\mathbf{z}'|\mathbf{x})}{q_t(\mathbf{z}|\mathbf{x})}\right)\right)\right],\end{aligned} \tag{42}$$

where $K(a) = a(\log a - 1)$ for $a \in \mathbb{R}^+$.

SEDD with Interpolating Uniform Noise From (1) we have $q_t(\mathbf{z}|\mathbf{x}) = \alpha_t \mathbf{x}_i + (1-\alpha_t)/N$ where i is the non-zero index of the one-hot vector $\mathbf{z}$, i.e., $\mathbf{z}_i = 1$. Thus, the "true" conditional score in (42) can be written as

$$\frac{q_t(\mathbf{z}'|\mathbf{x})}{q_t(\mathbf{z}|\mathbf{x})} = \frac{\alpha_t \mathbf{x}_j + (1-\alpha_t)/N}{\alpha_t \mathbf{x}_i + (1-\alpha_t)/N} = \frac{N\alpha_t \mathbf{x}_j + (1-\alpha_t)}{N\alpha_t \mathbf{x}_i + (1-\alpha_t)}, \tag{43}$$

where again we use i and j to denote the non-zero indices of the one-hot vectors $\mathbf{z}$ and $\mathbf{z}'$, respectively.

Using Mean Parameterization in SEDD NELBO In our work, we use the mean parameterization, that is we predict the 'clean' data given noisy observations using the model that we denote as $\mathbf{x}_\theta$. Note that (42) is minimized if

$$\mathbf{s}_\theta(\mathbf{z})_{\mathbf{z}'} = \frac{q_t(\mathbf{z}'|\mathbf{x})}{q_t(\mathbf{z}|\mathbf{x})}.$$

Thus, we can replace $\mathbf{x}$ in (43) to extract a score model from our parameterization:

$$\mathbf{s}_\theta(\mathbf{z})_{\mathbf{z}'} = \frac{\alpha_t (\mathbf{x}_\theta)_j + (1-\alpha_t)/N}{\alpha_t (\mathbf{x}_\theta)_i + (1-\alpha_t)/N} = \frac{N\alpha_t (\mathbf{x}_\theta)_j + (1-\alpha_t)}{N\alpha_t (\mathbf{x}_\theta)_i + (1-\alpha_t)}. \tag{44}$$

We now show two useful identities that come from this parameterization. First,

$$\begin{aligned}
\sum_{\mathbf{z}' \neq \mathbf{z}} \mathbf{s}_\theta(\mathbf{z})_{\mathbf{z}'} &= \sum_{j \neq i} \frac{\alpha_t (\mathbf{x}_\theta)_j + (1-\alpha_t)/N}{\alpha_t (\mathbf{x}_\theta)_i + (1-\alpha_t)/N} \\
&= \frac{\alpha_t [\sum_{j \neq i} (\mathbf{x}_\theta)_j] + \sum_{j \neq i} (1-\alpha_t)/N}{\alpha_t (\mathbf{x}_\theta)_i + (1-\alpha_t)/N} \\
&= \frac{\alpha_t [\sum_{j \neq i} (\mathbf{x}_\theta)_j] + (1-\alpha_t)(N-1)/N}{\alpha_t (\mathbf{x}_\theta)_i + (1-\alpha_t)/N} && \because \sum_{j=1}^{N} 1 = N \implies \sum_{j \neq i} 1 = N - 1 \\
&= \frac{\alpha_t [1 - (\mathbf{x}_\theta)_i] + (1-\alpha_t) - (1-\alpha_t)/N}{\alpha_t (\mathbf{x}_\theta)_i + (1-\alpha_t)/N} && \because \sum_j (\mathbf{x}_\theta)_j = 1 \implies \sum_{j \neq i} (\mathbf{x}_\theta)_j = 1 - (\mathbf{x}_\theta)_i \\
&= \frac{\alpha_t + (1-\alpha_t) - \alpha_t (\mathbf{x}_\theta)_i - (1-\alpha_t)/N}{\alpha_t (\mathbf{x}_\theta)_i + (1-\alpha_t)/N} \\
&= \frac{1 - \alpha_t (\mathbf{x}_\theta)_i - (1-\alpha_t)/N}{\alpha_t (\mathbf{x}_\theta)_i + (1-\alpha_t)/N} \\
&= \frac{1}{\alpha_t (\mathbf{x}_\theta)_i + (1-\alpha_t)/N} - 1 \\
&= \frac{N}{N\alpha_t (\mathbf{x}_\theta)_i + (1-\alpha_t)} - 1
\end{aligned} \tag{45}$$

The same logic can be applied to show that

$$\sum_{\mathbf{z}' \neq \mathbf{z}} \frac{q_t(\mathbf{z}'|\mathbf{x})}{q_t(\mathbf{z}|\mathbf{x})} = \sum_{j \neq i} \frac{\alpha_t \mathbf{x}_j + (1-\alpha_t)/N}{\alpha_t \mathbf{x}_i + (1-\alpha_t)/N} = \frac{N}{N\alpha_t \mathbf{x}_i + (1-\alpha_t)} - 1. \tag{46}$$

Equivalence Between $\text{NELBO}_{\text{SEDD}}$ and $\text{NELBO}_{\text{UDLM}}$ We can now use the identities from (45) and (46) to demonstrate that $\text{NELBO}_{\text{SEDD}}$ (42) is equivalent to $\text{NELBO}_{\text{UDLM}}$ (10).

$\text{NELBO}_{\text{SEDD}}$

$$= \mathbb{E}_{t\in[0,1],\mathbf{z}\sim q_t(.|\mathbf{x})}\left[\sum_{\mathbf{z}'\neq\mathbf{z}} R_t(\mathbf{z},\mathbf{z}')\left(\mathbf{s}_\theta(\mathbf{z})_{\mathbf{z}'} - \frac{q_t(\mathbf{z}'|\mathbf{x})}{q_t(\mathbf{z}|\mathbf{x})}\log \mathbf{s}_\theta(\mathbf{z})_{\mathbf{z}'} + K\left(\frac{q_t(\mathbf{z}'|\mathbf{x})}{q_t(\mathbf{z}|\mathbf{x})}\right)\right)\right]$$

$$= \mathbb{E}_{t\in[0,1],\mathbf{z}\sim q_t(.|\mathbf{x})}\left[\sum_{\mathbf{z}'\neq\mathbf{z}} -\frac{\alpha_t'}{N\alpha_t}\left(\mathbf{s}_\theta(\mathbf{z})_{\mathbf{z}'} - \frac{q_t(\mathbf{z}'|\mathbf{x})}{q_t(\mathbf{z}|\mathbf{x})}\log \mathbf{s}_\theta(\mathbf{z})_{\mathbf{z}'} + K\left(\frac{q_t(\mathbf{z}'|\mathbf{x})}{q_t(\mathbf{z}|\mathbf{x})}\right)\right)\right] \quad \text{from (37)}$$

$$= \mathbb{E}_{t\in[0,1],\mathbf{z}\sim q_t(.|\mathbf{x})}\left[\frac{\alpha_t'}{N\alpha_t}\left(-\sum_{\mathbf{z}'\neq\mathbf{z}}\mathbf{s}_\theta(\mathbf{z})_{\mathbf{z}'} + \sum_{\mathbf{z}'\neq\mathbf{z}}\frac{q_t(\mathbf{z}'|\mathbf{x})}{q_t(\mathbf{z}|\mathbf{x})}\log \mathbf{s}_\theta(\mathbf{z})_{\mathbf{z}'} - \sum_{\mathbf{z}'\neq\mathbf{z}} K\left(\frac{q_t(\mathbf{z}'|\mathbf{x})}{q_t(\mathbf{z}|\mathbf{x})}\right)\right)\right]$$

Recall that $K(a) = a\log a - a$

$$= \mathbb{E}_{t\in[0,1],\mathbf{z}\sim q_t(.|\mathbf{x})}\frac{\alpha_t'}{N\alpha_t}\left[-\sum_{\mathbf{z}'\neq\mathbf{z}}\mathbf{s}_\theta(\mathbf{z})_{\mathbf{z}'} + \sum_{\mathbf{z}'\neq\mathbf{z}}\frac{q_t(\mathbf{z}'|\mathbf{x})}{q_t(\mathbf{z}|\mathbf{x})}\log \mathbf{s}_\theta(\mathbf{z})_{\mathbf{z}'} - \sum_{\mathbf{z}'\neq\mathbf{z}}\frac{q_t(\mathbf{z}'|\mathbf{x})}{q_t(\mathbf{z}|\mathbf{x})}\log\frac{q_t(\mathbf{z}'|\mathbf{x})}{q_t(\mathbf{z}|\mathbf{x})} + \sum_{\mathbf{z}'\neq\mathbf{z}}\frac{q_t(\mathbf{z}'|\mathbf{x})}{q_t(\mathbf{z}|\mathbf{x})}\right]$$

Using (45) and (46) we get,

$$= \mathbb{E}_{t\in[0,1],\mathbf{z}\sim q_t(.|\mathbf{x})}\frac{\alpha_t'}{N\alpha_t}\left[-\frac{N}{\alpha_t N(\mathbf{x}_\theta)_i + (1-\alpha_t)} + \frac{N}{\alpha_t N\mathbf{x}_i + (1-\alpha_t)} + \sum_{\mathbf{z}'\neq\mathbf{z}}\frac{q_t(\mathbf{z}'|\mathbf{x})}{q_t(\mathbf{z}|\mathbf{x})}\log \mathbf{s}_\theta(\mathbf{z})_{\mathbf{z}'} - \sum_{\mathbf{z}'\neq\mathbf{z}}\frac{q_t(\mathbf{z}'|\mathbf{x})}{q_t(\mathbf{z}|\mathbf{x})}\log\frac{q_t(\mathbf{z}'|\mathbf{x})}{q_t(\mathbf{z}|\mathbf{x})}\right]$$

$$= \mathbb{E}_{t\in[0,1],\mathbf{z}\sim q_t(.|\mathbf{x})}\frac{\alpha_t'}{N\alpha_t}\left[-\frac{N}{\alpha_t N(\mathbf{x}_\theta)_i + (1-\alpha_t)} + \frac{N}{\alpha_t N\mathbf{x}_i + (1-\alpha_t)} - \sum_{\mathbf{z}'\neq\mathbf{z}}\frac{q_t(\mathbf{z}'|\mathbf{x})}{q_t(\mathbf{z}|\mathbf{x})}\log\left(\frac{1}{\mathbf{s}_\theta(\mathbf{z})_{\mathbf{z}'}}\frac{q_t(\mathbf{z}'|\mathbf{x})}{q_t(\mathbf{z}|\mathbf{x})}\right)\right]$$

Using (43) and (44) we get,

$$= \mathbb{E}_{t\in[0,1],\mathbf{z}\sim q_t(.|\mathbf{x})}\frac{\alpha_t'}{N\alpha_t}\left[\frac{N}{\alpha_t N\mathbf{x}_i + (1-\alpha_t)} - \frac{N}{\alpha_t N(\mathbf{x}_\theta)_i + (1-\alpha_t)} - \sum_{\mathbf{z}'\neq\mathbf{z}}\frac{\alpha_t N\mathbf{x}_j + (1-\alpha_t)}{\alpha_t N\mathbf{x}_i + (1-\alpha_t)}\log\left(\frac{\alpha_t N(\mathbf{x}_\theta)_i + (1-\alpha_t)}{\alpha_t N(\mathbf{x}_\theta)_j + (1-\alpha_t)}\cdot\frac{\alpha_t N\mathbf{x}_j + (1-\alpha_t)}{\alpha_t N\mathbf{x}_i + (1-\alpha_t)}\right)\right]$$

$$= \text{NELBO}_{\text{UDLM}}$$

C Experimental Details

C.1 Dataset Details

In this section, we provide more details, e.g., source, train/validation splits, etc., for the the datasets used in this work. For an overview, please see Table 8.

Species-10 This dataset is a composite of reference genomes from ten diverse species: *Arabidopsis thaliana*, *Caenorhabditis elegans*, *Danio rerio*, *Drosophila melanogaster*, *Felis catus*, *Gallus gallus*, *Gorilla gorilla*, *Homo sapiens*, *Mus musculus*, and *Salmo trutta*, which were downloaded from NCBI refseq database (O'Leary et al., 2016). In Table 9, we provide the assembly accession IDs used to download the reference genomes. Genomes were chunked into non-overlapping segments of 32,768 nucleotides and were tokenized using base-pair-level tokenization.

Training and validation sets were randomly split using 95% / 5%. In Table 10, we specify the size and relative species composition of each split.

Table 8: Relevant details for datasets used in this work.

Dataset	Tokenizer	Vocab. Size	Input size	Padding?
Species-10	Base-pair	12	32,768	No
QM9	Regex (Schwaller et al., 2019)	40	32	Yes
CIFAR-10	Binned Pixel Intensity	256	32×32×3	No
text8	Character	35	256	No
Amazon	`bert-base-uncased`	30,522	128	Yes
LM1B	`bert-base-uncased`	30,522	128	No

Table 9: Species genomes accession IDs

Species	Assembly Accession
Arabidopsis thaliana	`GCF_000001735.4_TAIR10.1`
Caenorhabditis elegans	`GCF_000002985.6_WBcel235`
Danio rerio	`GCF_000002035.6_GRCz11`
Drosophila melanogaster	`GCF_000001215.4_Release_6_plus_ISO1_MT`
Felis catus	`GCF_018350175.1_F.catus_Fca126_mat1.0`
Gallus gallus	`GCF_016699485.2_bGalGal1.mat.broiler.GRCg7b`
Gorilla gorilla	`GCF_029281585.2_NHGRI_mGorGor1-v2.0_pri`
Homo sapiens	`GCF_000001405.40_GRCh38.p14`
Mus musculus	`GCF_000001635.27_GRCm39`
Salmo trutta	`GCF_901001165.1_fSalTru1.1`

For guidance, we use the species label.

QM9 Molecules The QM9 dataset comes from Ruddigkeit et al. (2012) and Ramakrishnan et al. (2014). The dataset is comprised of ∼133k small molecules. We process the dataset using the RDKit library (Landrum et al., 2013) to extract 'canonical' SMILES string representations and add the annotations for drug-likeness (QED) and ring-counts. The data are tokenized using a regular expression from Schwaller et al. (2019). Input lengths of 32 tokens were used for pre-training and generation experiments.

Training and validation sets were randomly split using 95% / 5%.

For guidance, we use a cutoff of 90$^{\text{th}}$ percentile for QED / ring count to generate binary labels.

CIFAR-10 This widely-used image dataset contains RGB images with 32×32 pixels. We use the provided train and test splits of 50,000 training images and 10,000 validation images. The dataset

Table 10: Species-10 train and validation splits statistics. Proportion of each species in the training and validation sets. Overall split size is indicated in parentheses in the column header.

Species	Train (95% ≈ 16.5B bps)	Validation (5% ≈ 869M bps)
Arabidopsis thaliana	0.68	0.77
Caenorhabditis elegans	0.58	0.58
Danio rerio	9.50	9.29
Drosophila melanogaster	0.08	0.73
Felis catus	13.94	14.05
Gallus gallus	6.04	5.93
Gorilla gorilla	20.40	20.19
Homo sapiens	18.89	19.22
Mus musculus	15.69	15.60
Salmo trutta	13.49	13.64

contains 10 classes of roughly equal proportion. We tokenize the dataset by rounding to integer pixel intensity values in the range $[0, 255]$.

For guidance, we use the image class label.

text8 The dataset was downloaded from `http://mattmahoney.net/dc/text8.zip`. Data was tokenized at the character level using the lower case letters [‘a’ - ‘z’] and a white-space character. The data was broken into non-overlapping chunks of 256 tokens.

The first 90M characters were used for the training set and the final 5M characters were used as a validation set.

Amazon Review The Amazon Review dataset was downloaded from `https://huggingface.co/datasets/fancyzhx/amazon_polarity`. We tokenize using the `bert-base-uncased` tokenizer. Sequences were padded to a max input length of 128 tokens.

Train and validation splits were used from the downloaded data, with 3.6 M sequences in the training data and 400k sequences in the validation set.

LM1B This dataset was downloaded from `https://huggingface.co/datasets/billion-word-benchmark/lm1b`. We tokenize using the `bert-base-uncased` tokenizer. Data was broken into non-overlapping chunks of 128 tokens.

We use the train and validation splits provided in the downloaded data. After chunking the data, our training set consisted of 7M sequences and our validation set consisted of 72k sequences.

C.2 Architectural Details

In Table 11, we provide an overview of the architectures and parameter counts of the models used for each dataset.

Table 11: Architectures used when training on various datasets.

Dataset	Architecture	Parameter Count
Species-10	Mamba (AR) / Caduceus (Diffusion)	3.5 M (AR) / 4.8 M (Diffusion)
QM9	Transformer	92.4 M
CIFAR-10	UNet	35.8 M
Text8	Transformer	92.4 M
Amazon	Transformer	139 M
LM1B	Transformer	139 M

Genomics Caduceus For the Species-10 experiments, we use Mamba-based models (Gu & Dao, 2023). The AR model is a standard next-token-prediction Mamba backbone with 8 blocks and hidden dimension 256. For the diffusion models we train with a variant of the Caduceus architecture from Schiff et al. (2024), also with 8 blocks and hidden dimension 256. Specifically, we use the non-reverse complementary equivariant version of Caduceus, dubbed Caduceus-Ph in Schiff et al. (2024), which is similar to the AR Mamba, but with bi-directional Mamba blocks that use strategic weight tying to limit the parameter count: 3.5 M parameters of AR vs. 4.8 M for diffusion models.

CIFAR-10 UNet We adopt the UNet (Ronneberger et al., 2015) as a main backbone for both MDLM and UDLM, following (Ho et al., 2020). Network configurations are presented in Table 12. Specifically, we follow Austin et al. (2021) and Campbell et al. (2022) and use the original UNet backbone from DDPM (Ho et al., 2020), adding an extra discretized truncated logistic transformation to network outputs. To enable conditioning on labels, we add a label embedding layer which is added to the time embeddings, inspired by ADM network (Dhariwal & Nichol, 2021b).

Table 12: Architecture details for CIFAR-10 UNet.

	MDLM	UDLM
Vocab size	258	256
Number of ResNet blocks per scale	2	2
Base channels	128	128
Channel multiplier per scale	(1,2,2,2)	(1,2,2,2)
Attention resolutions	16	16
Time conditioning	False	True
Conditional embedding dimension	128	128
Number of Params	35.8M	35.8M

QM9 and NLP Transformer For both the QM9 and NLP datasets (Amazon, text8, and LM1B), we use the same Diffusion Transformer (Peebles & Xie, 2023) with adaLN for conditioning on time (for uniform noise diffusion models) and class (for guided generation). The Transformer for each dataset differs only in the word-embedding size, which is determined by the vocabulary size of the datasets' corresponding tokenizer. Our Transformer consists of 12 layers, a hidden dimension of 768, and 12 attention heads. We use RoPE (Su et al., 2021) as the positional embeddings.

C.3 Training Configurations

In Table 13, we detail the hyperparameter setup for each of the language modeling experiments in Section 5.1.

All diffusion models were trained and evaluated using a log-linear noise schedule.

Of note, for CIFAR-10, our models are trained for 300K iterations as opposed to 1.5M iterations, as in D3PM (Austin et al., 2021).

Table 13: Training hyper-parameters for all included experiments.

	Species-10	QM9	CIFAR-10	text8	Amazon	LM1B
Train steps	30K	25K	300K	1000K	184K	1000K
Batch size	32	2048	512	512	512	512
LR	$2e^{-3}$	$3e^{-4}$	$2e^{-4}$	$3e^{-4}$	$3e^{-4}$	$3e^{-4}$
Optim.	ADAM (0.9, 0.999)	ADAM (0.9, 0.999)	ADAM (0.9, 0.999)	ADAM (0.9, 0.999)	ADAM (0.9, 0.999)	ADAM (0.9, 0.999)
LR sched.	CosineDecay $3e^{-6}$ min.	CosineDecay $3e^{-6}$ min.	-	-	-	-
Warmup steps	3K	1K	5K	2.5K	2.5K	2.5K
GPU count	8	4	8	8	8	8
GPU type	A5000	A6000	A100	A100	A5000	A100

C.4 Guidance Details

C.4.1 Baselines

FUDGE Implementation FUDGE is a classifier-based autoregressive guidance method proposed by Yang & Klein (2021). FUDGE first trains a classifier on all possible prefixes. During sampling, instead of directly sampling from $p(x_i|x_{1:i-1})$, FUDGE samples from the perturbed conditional distribution $p(y|x_{1:i})p(x_i|x_{1:i-1})$ where y denotes the classifier label, $x_{1:i-1}$ denotes the already decoded tokens and x_i stands for possible token to be generated. For efficiency, FUDGE trun-

cates x_i to the $topk$ tokens with the largest unconditional log-likelihood. Although not in the original FUDGE formulation, following the classifier-based guidance in diffusion models (Dhariwal & Nichol (2021b)), we introduce the temperature variable γ and sample from the perturbed distribution $\frac{p(x_i|x_{1:i-1})p^{\gamma}(y|x_{1:i})}{\sum_{x_i} p(x_i|x_{1:i-1})p^{\gamma}(y|x_{1:i})}$

In our QM9 experiments, $topk$ is set as the vocabulary size which is 40. When training the classifier, we use a smaller classification model using the same dataset specified in Appendix C.1. The smaller backbone DiT consists of 8 layers, 8 attention heads, and a hidden dimension of 512. We use the same hyperparameters as those specified in Table 13.

PPLM Implementation Inspired by the approximate Metropolis-adjusted Langevin (MALA), Plug and Play language model (PPLM Dathathri et al. (2019) introduces guidance by conducting gradient updates on the hidden representations of a language model. PPLM first rewrites the autoregressive language models' decoding process using the KV-cache mechanism (Wolf et al., 2020): $x_{t+1}, H_{t+1} = LM(x_t, H_t)$, where x_t denotes the token at time step t and H_t is defined as $[(K_t^{(1)}, V_t^{(1)}), ..., (K_t^{(l)}, V_t^{(l)}]$ in which l is the number of layers. When decoding the token x_{t+1} at time step $t+1$, PPLM initializes the perturbation term ΔH_t as 0, and then iteratively updates it by n times following the gradient ascend formula: $\Delta H_t \leftarrow \Delta H_t + \eta \frac{\nabla_{\Delta H_t}[log(p(y|H_t+\Delta H_t)) - \gamma_{KL} D_{KL}(p(x_{t+1}|H_t+\Delta H_t)||p(x_{t+1}|H_t))]}{\|\nabla_{\Delta H_t}[log(p(y|H_t+\Delta H_t)) - \gamma_{KL} D_{KL}(p(x_{t+1}|H_t+\Delta H_t)||p(x_{t+1}|H_t))]\|}$ in which y denotes the classifier label. PPLM then generates the perturbed probability distribution $p_{perb}(x_{t+1}|H_t + \Delta H_t)$ using $H_t + \Delta H_t$ and decodes x_{t+1} using the fused probability distribution $\frac{1}{\beta} p_{perb}^{\gamma_{gm}}(x_{t+1}|H_t + \Delta H_t) p_{unperb}^{1-\gamma_{gm}}(x_{t+1}|H_t)$, where β is the normalization factor. Following Dathathri et al. (2019), in our experiments γ_{kl} is set as 0.01 and γ_{gm} is set as 0.95. η and n are decided by grid search.

For the QM9 experiments, when training the classifier for PPLM, we use the same dataset specified in Appendix C.1. We use the same hyperparameters as those specified in Table 13, except the LR peak is reduced to $3e^{-5}$ and minimum is reduced to $3e^{-7}$.

NOS Implementation To implement the NOS baseline from Gruver et al. (2024), we train a classifier where we use the same backbone as the unconditional diffusion model (see Appendix C.2 for architecture details), and we initialize and freeze weights using the pre-trained unconditional diffusion model. We then mean pool the last hidden embeddings of the backbone and linearly project them to the classification logits. This final projection layer represents the only trainable parameters of the guidance model.

At inference, we perform Langevin sampling following Algorithm 2 from Gruver et al. (2024). In our experiments we denote step-size for the Langevin sampling by η, the number of steps by n, and the weight on the 'fluency' KL penalty by γ_{kl}.

For the QM9 experiments, when training the classifier for NOS, we use the same dataset specified in Appendix C.1. We use the same hyperparameters as those specified in Table 13, except the LR peak is reduced to $3e^{-5}$ and minimum is reduced to $3e^{-7}$.

C.4.2 D-CFG Details

When implementing D-CFG we train a single AR / discrete diffusion model (see Appendix C.2 for architecture details) where we randomly drop out the class condition by replacing it with a class `[MASK]` token. The class condition is fused into models using the implementation of adaptive layer norm from Peebles & Xie (2023). We use 10% rate for masking / dropping-out the condition.

At inference time, we perform two forward passes through the model, one with condition provided to compute the conditional probability $p_\theta(\mathbf{z}_s \mid \mathbf{z}_t, y)$ and one where the condition is masked to compute the unconditional probability $p_\theta(\mathbf{z}_s \mid \mathbf{z}_t)$. These values are then used as described in Section 4.1.

C.4.3 D-CBG Details

For D-CBG in the QM9 experiment, we train a smaller classification model using the same dataset specified in Appendix C.1. The smaller backbone DiT consists of 8 layers, 8 attention heads, and a hidden dimension of 512. We apply mean pooling on the final hidden representations before

linearly projecting to the classification logits. We use the same hyperparameters as those specified in Table 13.

We train the model on noised inputs using a log-linear schedule, where the type of corruption applied corresponds to the diffusion model to which guidance is applied.

C.5 Guidance Evaluation Details

C.5.1 Genomic Sequences Metrics

Below we describe the quality and control metrics used in the species-specific genome generation experiment.

k-mer JS To compute the k-mer distirbution shift, for each species, we create counts for each of the unique 3mers and 6mers for that species in the validation set and 64 generated sequences for that species. We then compute the Jensen-Shannon divergence between those categorical histograms. Finally we take a weighted average of these distances across species where the weights are given by the relative species proportion in the validation dataset, see Table 10 for the relative proportions.

Discriminator AUROC For the discriminator AUROC metric, we train a HyenaDNA model with 2 layers and hidden dimension 128. This model was downloaded, modified (reduced number of layers and hidden dimension) and initialized from scratch from `https://huggingface.co/LongSafari/hyenadna-small-32k-seqlen-hf`. We mean pool the final layer embeddings and linearly project them to the binary classification logits. The model is trained on the 640 generated sequences, which are labeled as the negative class, and 640 randomly selected sequences from the ground truth validation set (64 sequences per species), which are labeled as the positive class. This dataset of 1,280 sequences is randomly split into 95% train and 5% validation. The discriminator is trained with batch size of 8, learning rate of $1\mathrm{e}^{-4}$, and the ADAM optimizer for 3 epochs to minimize binary cross entropy loss on the classification of real vs. generated sequences. We report the AUROC from the final epoch on the 5% validation split of this classification dataset.

Oracle F1 Finally, controllability is measured by the macro-averaged F1 of an oracle model on the 640 generated sequences. Our oracle model is a separate HyenaDNA model with 8 layers and hidden dimension 256, which has 6.6M parameters. This model was downloaded and initialized from scratch from `https://huggingface.co/LongSafari/hyenadna-small-32k-seqlen-hf`. We mean pool the final layer embeddings and linearly project them to the ten category classification logits. This model was trained on the full Species-10 dataset as described in Appendix C.1. For reference, in Table 14 we present the classification results of the this oracle model in the 5% valdiation set of the original data. We see that other than difficulty distinguishing between human and gorilla genomes, the model can serve as a near perfect oracle.

Table 14: Evaluation of HyenaDNA 'oracle' classifier on Species10 validation split.

Species	Precision	Recall	F1
Arabidopsis thaliana	1.00	0.99	1.00
Caenorhabditis elegans	1.00	1.00	1.00
Danio rerio	1.00	1.00	1.00
Drosophila melanogaster	1.00	1.00	1.00
Felis catus	1.00	1.00	1.00
Gallus gallus	1.00	1.00	1.00
Gorilla gorilla	0.63	0.45	0.52
Homo sapiens	0.54	0.72	0.62
Mus musculus	1.00	0.97	0.98
Salmo trutta	1.00	0.98	0.99

C.5.2 CIFAR-10 Quality Metrics

For evaluation, we randomly samples 50,000 images for each model and the tools provided here: https://github.com/w86763777/pytorch-image-generation-metrics.git, as described in Campbell et al. (2022).

FID Fréchet inception distance (Heusel et al., 2017) is a common metric in image generation where the divergence between real and generated data is measured to reflect the alignment of two distributions. The metric uses features extracted from a pretrained Inception-v3 model on ImageNet-1K to estimate the mean and variance of the input data. The difference of two multi-dimensional Gaussian distributions is measured by Wasserstein-2 distance or Fréchet distance $d(.)$ as follows:

$$FID = d(\mathcal{N}(\mu_{real}, \Sigma_{real}), \mathcal{N}(\mu_{fake}, \Sigma_{fake})) = \|\mu_{real} - \mu_{fake}\| + \|\text{Tr}(\Sigma_{real} + \Sigma_{fake} - 2(\Sigma_{real}\Sigma_{fake})^{0.5})\| \quad (47)$$

IS Inception Score (Salimans et al., 2016) is an alternative measure of how well generated images are aligned with human judgement. IS also utilizes Inception-v3 model to compute label distribution $p(y|x)$ for each generated image. IS focuses on two criteria: (1) A generated image should contain a distinct class object, meaning its label distribution is expected to have low entropy; (2) The generated images should vary across multiple classes, so the marginal distribution, $p(y) = \int_z p(y|G(z))dz$, is expected to have high entropy, ideally approaching a uniform distribution. The formula is presented as below:

$$IS = \exp\left[\mathbb{E}_x \text{KL}(p(y|x)\|p(y))\right], \quad (48)$$

F1 F1 is used as a proxy for satisfying the desired conditional generation of generated samples by a pre-trained classifier on CIFAR-10. We use a pretrained Vision Transformer model downloaded from https://huggingface.co/edadaltocg/vit_base_patch16_224_in21k_ft_cifar10 and fine-tuned on CIFAR-10.

C.5.3 QM9 Guidance Metrics

For the guidance experiments in QM9, we generate 2,048 sequences of length 32. The reported metrics are explained below.

Validity Validity is measured by whether the generated SMILES string can be parsed by the RDKit library. Any strings that fail to be parsed are counted as invalid.

Novelty Novelty is measured by the number of valid and unique sequences that are not present in the original QM9 dataset.

Property Mean / Median Finally, we also report the mean and median of the novel generated sequence for the property of interest, QED or ring count.

D Guidance Ablation Results

For each of the guidance experiments, we perform a hyerparameter search on the guidance parameters, e.g., γ in D-CFGand D-CBG. Below we present results from these searches for the various guidance experiments.

Species-specific Genome Generation In this experiment we vary $\gamma \in \{1, 2, 3\}$ for D-CFG applied to AR, MDLM, and UDLM. Additionally, for MDLM and UDLM we vary the number of sampling steps $T \in \{128, 256, 512\}$.

Results for the AR D-CFG grid search are presented in Table 15, results for MDLM and UDLM grid search are presented in Table 16. These results highlight that UDLM is more amenable to guidance

Table 15: Varying γ for AR with D-CFG on Species-10 generation.

AR D-CFGγ	3-mer JS (↓)	6-mer JS (↓)	Disc. AUROC (↓)	F1 (↑)
1	0.04	0.09	1.00	0.87
2	0.05	0.14	1.00	0.68
3	0.10	0.23	1.00	0.20

Table 16: Varying γ and T for MDLM and UDLM with D-CFG on Species-10 generation.

Model	D-CFGγ	3-mer JS (↓)	6-mer JS (↓)	Disc. AUROC (↓)	F1 (↑)
$T = 128$					
MDLM	1	**0.03**	0.08	0.66	0.87
MDLM	2	0.04	0.12	0.89	0.77
MDLM	3	0.06	0.15	0.91	0.63
UDLM	1	**0.03**	0.08	0.72	0.91
UDLM	2	0.07	0.15	0.69	0.92
UDLM	3	0.10	0.23	0.74	0.83
$T = 256$					
MDLM	1	**0.03**	0.08	0.81	0.88
MDLM	2	0.04	0.11	0.84	0.77
MDLM	3	0.07	0.16	0.83	0.67
UDLM	1	**0.03**	0.09	0.84	0.95
UDLM	2	0.07	0.16	0.85	0.94
UDLM	3	0.11	0.24	0.93	0.81
$T = 512$					
MDLM	1	**0.03**	**0.07**	**0.61**	0.90
MDLM	2	0.04	0.11	0.71	0.78
MDLM	3	0.06	0.15	0.91	0.69
UDLM	1	**0.03**	0.08	0.77	0.92
UDLM	2	0.08	0.18	0.75	**0.96**
UDLM	3	0.11	0.24	0.76	0.82

than either AR or MDLM in this setting. UDLM achieves better quality and control metrics, and while all model performance degrades as $\gamma = 3$, UDLM degrades less and improves when $\gamma = 2$.

QM9 In this section, we list the hyperparameter grid search results for QM9 drug likeliness (QED) maximization using AR FUDGE (Table 17), AR PPLM (Table 18), AR CFG (Table 19), MDLM D-CBG (Table 20), MDLM D-CFG (Table 21), UDLM D-CBG (Table 22), UDLM D-CFG (Table 23), MDLM NOS (Table 24), and UDLM NOS (Table 25). QM9 ring count maximization guidance results with AR FUDGE (Table 26), AR PPLM (Table 27), AR D-CFG (Table 28), MDLM D-CBG (Table 29), MDLM D-CFG (Table 30), UDLM D-CBG (Table 31), UDLM D-CFG (Table 32), MDLM NOS (Table 33), UDLM NOS (Table 34) are also included.

CIFAR-10 In Table 35, we explore the effect of $\gamma \in 1, 2, 3, 4, 5$ in conditional image generation. We find that increasing γ generally leads to better IS and F1 scores for both MDLM and UDLM. For FID, MDLM achieves better scores as γ increases. However, the impact of γ is weaker for UDLM, as the model's FID score worsens when γ exceeds 2. We also plot visual outputs of different γ in Figure 3. As γ is increased, the appearance and the shape of a class object become more refined and sharpened.

Table 17: Varying γ for maximizing QED guidance with AR FUDGE. The number of generated molecules is 2048. Mean and median QED values of the novel molecules are reported. Best values are **bolded**. The set of results used in the main paper is starred.

γ	Num. Valid (↑)	Num. Novel (↑)	QED Mean (↑)	QED Median (↑)
1	**2046**	6	0.55	0.55
2	2034	18	0.56	0.56
3	1984	22	0.58	0.58
4	1833	31	0.56	0.57
5	1641	**43**	0.57	0.59
10*	914	24	0.58	0.60
15	674	10	0.58	0.60
20	434	7	0.57	0.59
25	200	5	**0.62**	**0.62**
30	77	4	**0.62**	**0.62**
35	18	7	0.61	0.61
40	9	4	0.61	0.61

Table 18: Varying η and n for maximizing QED guidance with AR PPLM. The number of generated molecules is 2048. Mean and median QED values of the novel molecules are reported. Best values are **bolded**. The set of results used in the main paper is starred.

	Num. Valid (↑)	Num. Novel (↑)	QED Mean (↑)	QED Median (↑)
$\eta = 0.04, n = 10$	**2048**	0	N.A.	N.A.
$\eta = 0.1, n = 10$	2044	22	0.43	0.46
$\eta = 0.04, n = 30$	2030	39	0.47	0.48
$\eta = 0.1, n = 30^*$	1618	**291**	**0.48**	**0.49**

Table 19: Varying γ for maximizing QED guidance with AR D-CFG. The number of generated molecules is 2048. Mean and median QED values of the novel molecules are reported. Best values are **bolded**. The set of results used in the main paper is starred.

γ	Num. Valid (↑)	Num. Novel (↑)	QED Mean (↑)	QED Median (↑)
1.0	**2018**	172	0.57	0.57
1.5	**2018**	**175**	0.58	0.58
2.0	2017	142	0.59	0.59
2.5*	1995	130	**0.60**	**0.60**
3.0	1954	121	**0.60**	**0.60**

Table 20: Varying γ for maximizing QED guidance with MDLM D-CBG. The number of generated molecules is 2048. Mean and median QED values of the novel molecules are reported. Best values are **bolded**. The set of results used in the main paper is starred.

γ	Num. Valid (↑)	Num. Novel (↑)	QED Mean (↑)	QED Median (↑)
1.0	**1203**	**406**	0.46	0.47
1.5	1061	369	0.47	0.47
2.0	902	310	0.48	0.48
2.5	801	271	0.48	0.49
3.0	711	216	0.49	0.50
3.5	629	189	0.50	0.52
4.0	542	173	0.50	0.51
4.5	489	154	0.51	0.53
5.0	423	135	0.52	0.53
5.5	402	123	0.53	0.54
6.0	362	115	0.53	0.55
6.5	338	110	0.54	0.56
7.0	343	95	0.55	0.56
7.5	292	88	0.55	0.56
8.0	276	73	0.56	0.58
8.5	294	72	0.56	0.56
9.0	267	73	0.56	0.57
9.5	270	71	0.56	0.58
10.0	248	69	0.56	0.57
15.0	151	54	0.57	0.58
20.0*	64	22	**0.58**	**0.60**
25.0	11	3	**0.58**	0.58
30.0	1	1	**0.58**	0.58
40.0	0	0	N.A.	N.A.

Table 21: Varying γ for maximizing QED guidance with MDLM D-CFG. The number of generated molecules is 2048. Mean and median QED values of the novel molecules are reported. Best values are **bolded**. The set of results used in the main paper is starred.

γ	Num. Valid (↑)	Num. Novel (↑)	QED Mean (↑)	QED Median (↑)
1.0	**1178**	**410**	0.56	0.57
1.5	1079	396	0.58	0.58
2.0	932	347	0.59	0.60
2.5	760	307	0.60	**0.61**
3.0*	652	251	**0.61**	**0.61**

Table 22: Varying γ for maximizing QED guidance with UDLM D-CBG. The number of generated molecules is 2048. Mean and median QED values of the novel molecules are reported. Best values are **bolded**. The set of results used in the main paper is starred.

γ	Num. Valid (↑)	Num. Novel (↑)	QED Mean (↑)	QED Median (↑)
1.0	**2039**	**168**	0.47	0.47
3.0	1916	154	0.50	0.50
5.0	1697	126	0.52	0.55
10.0	1771	63	0.57	0.59
15.0	1732	70	0.59	0.60
20.0	1670	77	0.60	0.60
25.0	1478	74	0.60	**0.61**
30.0	1330	69	0.60	**0.61**
35.0*	1223	87	**0.61**	**0.61**
40.0	1178	75	**0.61**	**0.61**

Table 23: Varying γ for maximizing QED guidance with UDLM D-CFG. The number of generated molecules is 2048. Mean and median QED values of the novel molecules are reported. Best values are **bolded**. The set of results used in the main paper is starred.

γ	Num. Valid (↑)	Num. Novel (↑)	QED Mean (↑)	QED Median (↑)
1.0	2010	**184**	0.57	0.57
1.5	2021	158	0.59	0.59
2.0	2032	144	0.60	0.60
2.5	**2035**	140	0.60	0.60
3.0*	2034	133	**0.61**	**0.61**

Figure 3: Illustration of varying γ on CIFAR-10.

Table 24: Varying η, n, and γ_{kl} for maximizing QED guidance with MDLM NOS. The number of generated molecules is 2048. Mean and median QED values of the novel molecules are reported. Best values are **bolded**. The set of results used in the main paper is starred.

	Num. Valid (↑)	Num. Novel (↑)	QED (↑)	
			Mean	Median
$\eta = 0.001, n = 1, \gamma_{kl} = 0$	1324	453	0.446	0.457
$\eta = 0.001, n = 1, \gamma_{kl} = 0.001$	1302	471	**0.454**	**0.463**
$\eta = 0.001, n = 1, \gamma_{kl} = 0.01^*$	1302	471	**0.454**	**0.463**
$\eta = 0.001, n = 5, \gamma_{kl} = 0$	1322	453	0.445	0.456
$\eta = 0.001, n = 5, \gamma_{kl} = 0.001$	1300	468	0.453	**0.463**
$\eta = 0.001, n = 5, \gamma_{kl} = 0.01$	1322	453	0.445	0.456
$\eta = 0.001, n = 10, \gamma_{kl} = 0$	1324	454	0.445	0.456
$\eta = 0.001, n = 10, \gamma_{kl} = 0.001$	1300	468	0.453	**0.463**
$\eta = 0.001, n = 10, \gamma_{kl} = 0.01$	1324	454	0.445	0.456
$\eta = 0.01, n = 1, \gamma_{kl} = 0$	1323	452	0.445	0.456
$\eta = 0.01, n = 1, \gamma_{kl} = 0.001$	1301	471	0.452	**0.463**
$\eta = 0.01, n = 1, \gamma_{kl} = 0.01$	1323	452	0.445	0.456
$\eta = 0.01, n = 5, \gamma_{kl} = 0$	1302	471	0.453	**0.463**
$\eta = 0.01, n = 5, \gamma_{kl} = 0.001$	1325	458	0.445	0.456
$\eta = 0.01, n = 5, \gamma_{kl} = 0.01$	1302	471	0.453	**0.463**
$\eta = 0.01, n = 10, \gamma_{kl} = 0$	**1329**	457	0.445	0.456
$\eta = 0.01, n = 10, \gamma_{kl} = 0.001$	1306	478	0.453	**0.463**
$\eta = 0.01, n = 10, \gamma_{kl} = 0.01$	**1329**	457	0.445	0.456
$\eta = 0.1, n = 1, \gamma_{kl} = 0$	1296	477	0.451	0.462
$\eta = 0.1, n = 1, \gamma_{kl} = 0.001$	1325	456	0.444	0.456
$\eta = 0.1, n = 1, \gamma_{kl} = 0.01$	1296	477	0.451	0.462
$\eta = 0.1, n = 5, \gamma_{kl} = 0$	1322	454	0.443	0.456
$\eta = 0.1, n = 5, \gamma_{kl} = 0.001$	1300	482	0.450	0.461
$\eta = 0.1, n = 5, \gamma_{kl} = 0.01$	1322	454	0.443	0.456
$\eta = 0.1, n = 10, \gamma_{kl} = 0$	1298	478	0.450	0.461
$\eta = 0.1, n = 10, \gamma_{kl} = 0.001$	1323	452	0.443	0.456
$\eta = 0.1, n = 10, \gamma_{kl} = 0.01$	1298	478	0.450	0.461
$\eta = 1, n = 1, \gamma_{kl} = 0$	1268	486	0.441	0.448
$\eta = 1, n = 1, \gamma_{kl} = 0.001$	1258	**503**	0.446	0.452
$\eta = 1, n = 1, \gamma_{kl} = 0.01$	1268	486	0.441	0.448
$\eta = 1, n = 5, \gamma_{kl} = 0$	1268	486	0.441	0.448
$\eta = 1, n = 5, \gamma_{kl} = 0.001$	1258	**503**	0.446	0.452
$\eta = 1, n = 5, \gamma_{kl} = 0.01$	1268	486	0.441	0.448
$\eta = 1, n = 10, \gamma_{kl} = 0$	1268	486	0.441	0.448
$\eta = 1, n = 10, \gamma_{kl} = 0.001$	1258	**503**	0.446	0.452
$\eta = 1, n = 10, \gamma_{kl} = 0.01$	1268	486	0.441	0.448
$\eta = 5, n = 1, \gamma_{kl} = 0$	631	367	0.435	0.443
$\eta = 5, n = 1, \gamma_{kl} = 0.001$	664	410	0.430	0.438
$\eta = 5, n = 1, \gamma_{kl} = 0.01$	631	367	0.435	0.443
$\eta = 5, n = 5, \gamma_{kl} = 0$	631	367	0.435	0.443
$\eta = 5, n = 5, \gamma_{kl} = 0.001$	664	410	0.430	0.438
$\eta = 5, n = 5, \gamma_{kl} = 0.01$	631	367	0.435	0.443
$\eta = 5, n = 10, \gamma_{kl} = 0$	631	367	0.435	0.443
$\eta = 5, n = 10, \gamma_{kl} = 0.001$	664	410	0.430	0.438
$\eta = 5, n = 10, \gamma_{kl} = 0.01$	631	367	0.435	0.443

Table 25: Varying η, n, and γ_{kl} for maximizing QED guidance with UDLM NOS. The number of generated molecules is 2048. Mean and median QED values of the novel molecules are reported. Best values are **bolded**. The set of results used in the main paper is starred.

	Num. Valid (↑)	Num. Novel (↑)	QED (↑)	
			Mean	Median
$\eta = 0.001, n = 1, \gamma_{kl} = 0$	1976	369	0.450	0.462
$\eta = 0.001, n = 1, \gamma_{kl} = 0.001$	**1988**	383	0.458	0.467
$\eta = 0.001, n = 1, \gamma_{kl} = 0.01$	**1988**	383	0.458	0.467
$\eta = 0.001, n = 5, \gamma_{kl} = 0$	1987	382	0.458	0.467
$\eta = 0.001, n = 5, \gamma_{kl} = 0.001$	1975	369	0.449	0.462
$\eta = 0.001, n = 5, \gamma_{kl} = 0.01$	1987	382	0.458	0.467
$\eta = 0.001, n = 10, \gamma_{kl} = 0$	1987	379	0.457	0.467
$\eta = 0.001, n = 10, \gamma_{kl} = 0.001$	1976	369	0.449	0.462
$\eta = 0.001, n = 10, \gamma_{kl} = 0.01$	1987	379	0.457	0.467
$\eta = 0.01, n = 1, \gamma_{kl} = 0$	**1988**	384	0.458	0.467
$\eta = 0.01, n = 1, \gamma_{kl} = 0.001$	1976	373	0.450	0.462
$\eta = 0.01, n = 1, \gamma_{kl} = 0.01$	**1988**	384	0.458	0.467
$\eta = 0.01, n = 5, \gamma_{kl} = 0$	1976	370	0.449	0.461
$\eta = 0.01, n = 5, \gamma_{kl} = 0.001$	1985	383	0.458	0.468
$\eta = 0.01, n = 5, \gamma_{kl} = 0.01$	1985	383	0.458	0.468
$\eta = 0.01, n = 10, \gamma_{kl} = 0$	1974	371	0.449	0.461
$\eta = 0.01, n = 10, \gamma_{kl} = 0.001$	**1988**	383	0.458	0.468
$\eta = 0.01, n = 10, \gamma_{kl} = 0.01$	1974	371	0.449	0.461
$\eta = 0.1, n = 1, \gamma_{kl} = 0$	1976	384	0.448	0.461
$\eta = 0.1, n = 1, \gamma_{kl} = 0.001$	1985	**387**	0.458	0.468
$\eta = 0.1, n = 1, \gamma_{kl} = 0.01$	1976	384	0.448	0.461
$\eta = 0.1, n = 5, \gamma_{kl} = 0$	1984	380	0.459	0.468
$\eta = 0.1, n = 5, \gamma_{kl} = 0.001$	1973	377	0.449	0.462
$\eta = 0.1, n = 5, \gamma_{kl} = 0.01$	1984	380	0.459	0.468
$\eta = 0.1, n = 10, \gamma_{kl} = 0$	1982	382	0.459	0.468
$\eta = 0.1, n = 10, \gamma_{kl} = 0.001$	1973	375	0.449	0.462
$\eta = 0.1, n = 10, \gamma_{kl} = 0.01$	1982	382	0.459	0.468
$\eta = 1, n = 1, \gamma_{kl} = 0$	1965	361	0.455	0.466
$\eta = 1, n = 1, \gamma_{kl} = 0.001$	1957	382	0.452	0.461
$\eta = 1, n = 1, \gamma_{kl} = 0.01$	1965	361	0.455	0.466
$\eta = 1, n = 5, \gamma_{kl} = 0$	1957	382	0.452	0.461
$\eta = 1, n = 5, \gamma_{kl} = 0.001$	1965	361	0.455	0.466
$\eta = 1, n = 5, \gamma_{kl} = 0.01$	1957	382	0.452	0.461
$\eta = 1, n = 10, \gamma_{kl} = 0$	1957	382	0.452	0.461
$\eta = 1, n = 10, \gamma_{kl} = 0.001$	1965	361	0.455	0.466
$\eta = 1, n = 10, \gamma_{kl} = 0.01$	1957	382	0.452	0.461
$\eta = 5, n = 1, \gamma_{kl} = 0$	944	302	**0.467**	**0.474**
$\eta = 5, n = 1, \gamma_{kl} = 0.001$	974	329	0.465	0.470
$\eta = 5, n = 1, \gamma_{kl} = 0.01$	944	302	**0.467**	**0.474**
$\eta = 5, n = 5, \gamma_{kl} = 0$	944	302	**0.467**	**0.474**
$\eta = 5, n = 5, \gamma_{kl} = 0.001^*$	946	302	**0.467**	**0.474**
$\eta = 5, n = 5, \gamma_{kl} = 0.01$	981	332	0.465	0.469
$\eta = 5, n = 10, \gamma_{kl} = 0$	974	329	0.465	0.470
$\eta = 5, n = 10, \gamma_{kl} = 0.001$	975	330	0.465	0.470
$\eta = 5, n = 10, \gamma_{kl} = 0.01$	982	330	0.465	0.470

Table 26: Varying γ for maximizing Ring Count guidance with AR FUDGE. The number of generated molecules is 2048. Mean and Median Ring Count values of the novel molecules are reported. Best values are **bolded**. The set of results used in the main paper is starred.

γ	Num. Valid (↑)	Num. Novel (↑)	Ring Count Mean (↑)	Ring Count Median (↑)
1	**2040**	**20**	3.90	4.00
2*	2035	15	4.20	4.00
3	2018	11	5.09	5.00
4	1982	7	4.71	5.00
5	1785	3	**5.33**	**6.00**
10	1	0	N.A.	N.A.
15	0	0	N.A.	N.A.
20	0	0	N.A.	N.A.
25	0	0	N.A.	N.A.
30	0	0	N.A.	N.A.
35	0	0	N.A.	N.A.
40	0	0	N.A.	N.A.

Table 27: Varying η and n for maximizing Ring Count guidance with AR PPLM. The number of generated molecules is 2048. Mean and Median Ring Count values of the novel molecules are reported. Best values are **bolded**. The set of results used in the main paper is starred.

	Num. Valid (↑)	Num. Novel (↑)	Ring Count (↑)	
			Mean	Median
$\eta = 0.04, n = 10$	**2048**	0	N.A.	N.A.
$\eta = 0.1, n = 10$	2042	23	1.26	1.00
$\eta = 0.04, n = 30$	2027	43	1.88	**2.00**
$\eta = 0.1, n = 30^*$	1486	**233**	**1.89**	**2.00**

Table 28: Varying γ for maximizing Ring Count guidance with AR D-CFG. The number of generated molecules is 2048. Mean and Median Ring Count values of the novel molecules are reported. Best values are **bolded**. The set of results used in the main paper is starred.

γ	Num. Valid (↑)	Num. Novel (↑)	Ring Count Mean (↑)	Ring Count Median (↑)
1.0	**2009**	303	4.49	4.00
1.5	1984	311	4.79	5.00
2.0	1948	**390**	4.77	5.00
2.5	1868	389	4.78	5.00
3.0	1765	378	4.82	5.00
3.5	1540	300	4.82	5.00
4.0	1206	217	4.88	5.00
4.5	965	173	4.71	4.00
5.0	656	110	4.75	4.00
5.5	426	76	4.70	4.00
6.0	284	57	4.75	4.00
6.5	166	35	4.92	5.00
7.0*	108	24	5.04	5.50
7.5	72	12	5.23	**6.00**
8.0	50	6	5.33	**6.00**
8.5	22	3	5.50	**6.00**
9.0	8	4	5.50	**6.00**
9.5	4	3	5.33	**6.00**
10.0	2	1	**6.00**	**6.00**

Table 29: Varying γ for maximizing Ring Count guidance with MDLM D-CBG. The number of generated molecules is 2048. Mean and Median Ring Count values of the novel molecules are reported. Best values are **bolded**. The set of results used in the main paper is starred.

γ	Num. Valid (↑)	Num. Novel (↑)	Ring Count Mean (↑)	Ring Count Median (↑)
1.0	**1160**	**447**	2.19	2.00
5.0	364	215	3.58	4.00
10.0	251	160	4.19	4.00
15.0	171	118	4.71	**5.00**
20.0	119	100	4.98	**5.00**
25.0*	82	65	**5.05**	**5.00**
30.0	49	38	4.63	4.00
35.0	21	18	4.89	**5.00**
40.0	12	12	4.75	**5.00**

Table 30: Varying γ for maximizing Ring Count guidance with MDLM D-CFG. The number of generated molecules is 2048. Mean and Median Ring Count values of the novel molecules are reported. Best values are **bolded**. The set of results used in the main paper is starred.

γ	Num. Valid (↑)	Num. Novel (↑)	Ring Count Mean (↑)	Ring Count Median (↑)
1.0	**968**	**487**	4.32	4.00
1.5	870	432	4.64	4.00
2.0	781	429	4.72	**5.00**
2.5	637	375	4.87	**5.00**
3.0	508	290	4.84	**5.00**
3.5	435	256	4.79	**5.00**
4.0	336	196	4.92	**5.00**
4.5	308	186	4.83	**5.00**
5.0	213	123	4.86	**5.00**
5.5	164	107	4.86	**5.00**
6.0*	132	90	**5.01**	**5.00**
6.5	81	54	4.80	**5.00**
7.0	77	54	4.83	**5.00**
7.5	54	39	4.82	**5.00**
8.0	32	22	4.68	**5.00**
8.5	17	14	4.64	4.00
9.0	10	9	4.78	**5.00**
9.5	12	11	4.91	**5.00**
10.0	3	3	4.00	4.00

Table 31: Varying γ for maximizing Ring Count guidance with UDLM D-CBG. The number of generated molecules is 2048. Mean and Median Ring Count values of the novel molecules are reported. Best values are **bolded**. The set of results used in the main paper is starred.

γ	Num. Valid (↑)	Num. Novel (↑)	Ring Count Mean (↑)	Ring Count Median (↑)
1.0	**2029**	209	2.13	2.00
5.0	1494	301	4.09	4.00
10.0	1641	528	4.50	4.00
15.0	1676	711	4.56	4.00
20.0	1692	821	4.62	4.00
25.0	1697	849	4.66	4.00
30.0	1697	889	4.67	4.00
35.0	1707	926	4.71	4.00
40.0*	1670	**943**	**4.73**	**5.00**

Table 32: Varying γ for maximizing Ring Count guidance with UDLM D-CFG. The number of generated molecules is 2048. Mean and Median Ring Count values of the novel molecules are reported. Best values are **bolded**. The set of results used in the main paper is starred.

γ	Num. Valid (↑)	Num. Novel (↑)	Ring Count Mean (↑)	Ring Count Median (↑)
1.0	1932	**585**	4.54	4.00
1.5	1979	552	4.66	4.00
2.0	**1981**	535	4.80	**5.00**
2.5	1971	507	4.83	**5.00**
3.0	1968	481	4.82	**5.00**
3.5*	1975	459	**4.91**	**5.00**
4.0	1951	426	4.89	**5.00**
4.5	1925	404	4.88	**5.00**
5.0	1894	429	4.81	**5.00**
5.5	1826	406	4.89	**5.00**
6.0	1699	366	4.84	**5.00**
6.5	1611	317	4.83	**5.00**
7.0	1508	300	4.82	**5.00**
7.5	1375	308	4.84	**5.00**
8.0	1251	232	4.85	**5.00**
8.5	1110	219	4.81	**5.00**
9.0	1026	233	4.73	**5.00**
9.5	887	194	4.84	**5.00**
10.0	829	207	4.71	4.00

Table 33: Varying η, n, and γ_{kl} for maximizing Ring Count guidance with MDLM NOS. The number of generated molecules is 2048. Mean and median Ring Count values of the novel molecules are reported. Best values are **bolded**. The set of results used in the main paper is starred.

	Num. Valid (↑)	Num. Novel (↑)	Ring Count (↑)	
			Mean	Median
$\eta = 0.001, n = 1, \gamma_{kl} = 0$	1302	471	1.926	2.000
$\eta = 0.001, n = 1, \gamma_{kl} = 0.001$	1325	453	1.899	2.000
$\eta = 0.001, n = 1, \gamma_{kl} = 0.01$	1302	471	1.926	2.000
$\eta = 0.001, n = 5, \gamma_{kl} = 0$	**1326**	456	1.908	2.000
$\eta = 0.001, n = 5, \gamma_{kl} = 0.001$	1303	473	1.930	2.000
$\eta = 0.001, n = 5, \gamma_{kl} = 0.01$	**1326**	456	1.908	2.000
$\eta = 0.001, n = 10, \gamma_{kl} = 0$	1325	456	1.914	2.000
$\eta = 0.001, n = 10, \gamma_{kl} = 0.001$	1303	471	1.930	2.000
$\eta = 0.001, n = 10, \gamma_{kl} = 0.01$	1325	456	1.914	2.000
$\eta = 0.01, n = 1, \gamma_{kl} = 0$	1324	453	1.925	2.000
$\eta = 0.01, n = 1, \gamma_{kl} = 0.001$	1306	472	1.928	2.000
$\eta = 0.01, n = 1, \gamma_{kl} = 0.01$	1324	453	1.925	2.000
$\eta = 0.01, n = 5, \gamma_{kl} = 0$	1302	469	1.928	2.000
$\eta = 0.01, n = 5, \gamma_{kl} = 0.001$	1323	459	1.939	2.000
$\eta = 0.01, n = 5, \gamma_{kl} = 0.01$	1302	469	1.928	2.000
$\eta = 0.01, n = 10, \gamma_{kl} = 0$	1324	460	1.933	2.000
$\eta = 0.01, n = 10, \gamma_{kl} = 0.001$	1324	460	1.933	2.000
$\eta = 0.01, n = 10, \gamma_{kl} = 0.01$	1305	470	1.932	2.000
$\eta = 0.1, n = 1, \gamma_{kl} = 0$	1292	478	1.948	2.000
$\eta = 0.1, n = 1, \gamma_{kl} = 0.001$	1320	469	1.942	2.000
$\eta = 0.1, n = 1, \gamma_{kl} = 0.01$	1292	478	1.948	2.000
$\eta = 0.1, n = 5, \gamma_{kl} = 0$	1316	474	1.951	2.000
$\eta = 0.1, n = 5, \gamma_{kl} = 0.001$	1291	480	1.960	2.000
$\eta = 0.1, n = 5, \gamma_{kl} = 0.01$	1316	474	1.951	2.000
$\eta = 0.1, n = 10, \gamma_{kl} = 0$	1315	471	1.951	2.000
$\eta = 0.1, n = 10, \gamma_{kl} = 0.001$	1292	480	1.952	2.000
$\eta = 0.1, n = 10, \gamma_{kl} = 0.01$	1292	480	1.952	2.000
$\eta = 1, n = 1, \gamma_{kl} = 0$	1186	507	2.384	2.000
$\eta = 1, n = 1, \gamma_{kl} = 0.001$	1163	479	2.240	2.000
$\eta = 1, n = 1, \gamma_{kl} = 0.01$	1186	507	2.384	2.000
$\eta = 1, n = 5, \gamma_{kl} = 0$	1186	507	2.384	2.000
$\eta = 1, n = 5, \gamma_{kl} = 0.001$	1188	**510**	2.388	2.000
$\eta = 1, n = 5, \gamma_{kl} = 0.01$	1166	481	2.263	2.000
$\eta = 1, n = 10, \gamma_{kl} = 0$	1163	479	2.240	2.000
$\eta = 1, n = 10, \gamma_{kl} = 0.001$	1188	508	2.391	2.000
$\eta = 1, n = 10, \gamma_{kl} = 0.01$	1168	483	2.267	2.000
$\eta = 5, n = 1, \gamma_{kl} = 0$	390	268	3.164	**3.000**
$\eta = 5, n = 1, \gamma_{kl} = 0.001$	337	231	3.147	**3.000**
$\eta = 5, n = 1, \gamma_{kl} = 0.01$	390	268	3.164	**3.000**
$\eta = 5, n = 5, \gamma_{kl} = 0$	390	268	3.164	**3.000**
$\eta = 5, n = 5, \gamma_{kl} = 0.001$	341	235	3.187	**3.000**
$\eta = 5, n = 5, \gamma_{kl} = 0.01$	403	281	3.281	**3.000**
$\eta = 5, n = 10, \gamma_{kl} = 0$	390	268	3.164	**3.000**
$\eta = 5, n = 10, \gamma_{kl} = 0.001$	343	237	3.203	**3.000**
$\eta = 5, n = 10, \gamma_{kl} = 0.01^*$	353	246	**3.313**	**3.000**

Table 34: Varying η, n, and γ_{kl} for maximizing Ring Count guidance with UDLM NOS. The number of generated molecules is 2048. Mean and median Ring Count values of the novel molecules are reported. Best values are **bolded**. The set of results used in the main paper is starred.

	Num. Valid (↑)	Num. Novel (↑)	Ring Count (↑)	
			Mean	Median
$\eta = 0.001, n = 1, \gamma_{kl} = 0$	1975	369	2.024	2.000
$\eta = 0.001, n = 1, \gamma_{kl} = 0.001$	1987	382	2.026	2.000
$\eta = 0.001, n = 1, \gamma_{kl} = 0.01$	1987	382	2.026	2.000
$\eta = 0.001, n = 5, \gamma_{kl} = 0$	1975	369	2.014	2.000
$\eta = 0.001, n = 5, \gamma_{kl} = 0.001$	1987	379	2.021	2.000
$\eta = 0.001, n = 5, \gamma_{kl} = 0.01$	1975	369	2.014	2.000
$\eta = 0.001, n = 10, \gamma_{kl} = 0$	1987	382	2.016	2.000
$\eta = 0.001, n = 10, \gamma_{kl} = 0.001$	1975	369	2.014	2.000
$\eta = 0.001, n = 10, \gamma_{kl} = 0.01$	1975	369	2.014	2.000
$\eta = 0.01, n = 1, \gamma_{kl} = 0$	1975	373	2.024	2.000
$\eta = 0.01, n = 1, \gamma_{kl} = 0.001$	1987	376	2.027	2.000
$\eta = 0.01, n = 1, \gamma_{kl} = 0.01$	1987	376	2.027	2.000
$\eta = 0.01, n = 5, \gamma_{kl} = 0$	1973	373	2.019	2.000
$\eta = 0.01, n = 5, \gamma_{kl} = 0.001$	1989	384	2.026	2.000
$\eta = 0.01, n = 5, \gamma_{kl} = 0.01$	1989	384	2.026	2.000
$\eta = 0.01, n = 10, \gamma_{kl} = 0$	1971	372	2.027	2.000
$\eta = 0.01, n = 10, \gamma_{kl} = 0.001$	**1990**	384	2.034	2.000
$\eta = 0.01, n = 10, \gamma_{kl} = 0.01$	1971	372	2.027	2.000
$\eta = 0.1, n = 1, \gamma_{kl} = 0$	1983	387	2.041	2.000
$\eta = 0.1, n = 1, \gamma_{kl} = 0.001$	1973	371	1.997	2.000
$\eta = 0.1, n = 1, \gamma_{kl} = 0.01$	1983	387	2.041	2.000
$\eta = 0.1, n = 5, \gamma_{kl} = 0$	1983	391	2.087	2.000
$\eta = 0.1, n = 5, \gamma_{kl} = 0.001$	1974	367	1.989	2.000
$\eta = 0.1, n = 5, \gamma_{kl} = 0.01$	1974	367	1.989	2.000
$\eta = 0.1, n = 10, \gamma_{kl} = 0$	1981	392	2.094	2.000
$\eta = 0.1, n = 10, \gamma_{kl} = 0.001$	1975	371	1.992	2.000
$\eta = 0.1, n = 10, \gamma_{kl} = 0.01$	1981	392	2.094	2.000
$\eta = 1, n = 1, \gamma_{kl} = 0$	1961	**423**	2.137	2.000
$\eta = 1, n = 1, \gamma_{kl} = 0.001$	1962	403	2.181	2.000
$\eta = 1, n = 1, \gamma_{kl} = 0.01$	1961	**423**	2.137	2.000
$\eta = 1, n = 5, \gamma_{kl} = 0$	1961	**423**	2.137	2.000
$\eta = 1, n = 5, \gamma_{kl} = 0.001$	1962	403	2.181	2.000
$\eta = 1, n = 5, \gamma_{kl} = 0.01$	1961	**423**	2.137	2.000
$\eta = 1, n = 10, \gamma_{kl} = 0$	1961	**423**	2.137	2.000
$\eta = 1, n = 10, \gamma_{kl} = 0.001$	1962	403	2.181	2.000
$\eta = 1, n = 10, \gamma_{kl} = 0.01$	1961	**423**	2.137	2.000
$\eta = 5, n = 1, \gamma_{kl} = 0$	950	396	2.629	2.000
$\eta = 5, n = 1, \gamma_{kl} = 0.001$	933	379	2.546	2.000
$\eta = 5, n = 1, \gamma_{kl} = 0.01$	950	396	2.629	2.000
$\eta = 5, n = 5, \gamma_{kl} = 0$	950	396	2.629	2.000
$\eta = 5, n = 5, \gamma_{kl} = 0.001$	933	379	2.546	2.000
$\eta = 5, n = 5, \gamma_{kl} = 0.01$	954	397	2.635	2.000
$\eta = 5, n = 10, \gamma_{kl} = 0$	950	396	2.629	2.000
$\eta = 5, n = 10, \gamma_{kl} = 0.001$	932	378	2.550	2.000
$\eta = 5, n = 10, \gamma_{kl} = 0.01^*$	961	402	**2.637**	**2.500**

Table 35: Ablation results of γ values on CIFAR-10. Best values are **bolded**.

	FID (↓)	IS (↑)	F1 (↑)
MDLM D-CFG			
$\gamma = 1$	27.94	7.14	0.76
$\gamma = 2$	18.62	8.24	0.95
$\gamma = 3$	16.19	8.78	0.99
$\gamma = 4$	**15.56**	9.02	0.99
$\gamma = 5$	15.73	**9.19**	**1.00**
UDLM D-CFG			
$\gamma = 1$	26.70	7.43	0.81
$\gamma = 2$	**20.75**	8.34	0.96
$\gamma = 3$	21.31	8.52	0.98
$\gamma = 4$	23.21	**8.66**	**0.99**
$\gamma = 5$	26.15	8.60	**0.99**