Discrete Modeling via Boundary Conditional Diffusion Processes

Yuxuan Gu[†] **Xiaocheng Feng**[†‡] **Lei Huang**[†] **Yingsheng Wu**[†] **Zekun Zhou**[†]
Weihong Zhong[†] **Kun Zhu**[†] **Bing Qin**[†‡]
[†]Harbin Institute of Technology [‡] Peng Cheng Laboratory
{yxgu,xcfeng,lhuang,yswu,zkzhou,whzhong,kzhu,qinb}@ir.hit.edu.cn

Abstract

We present an novel framework for efficiently and effectively extending the powerful continuous diffusion processes to discrete modeling. Previous approaches have suffered from the discrepancy between discrete data and continuous modeling. Our study reveals that the absence of guidance from discrete boundaries in learning probability contours is one of the main reasons. To address this issue, we propose a two-step forward process that first estimates the boundary as a prior distribution and then rescales the forward trajectory to construct a boundary conditional diffusion model. The reverse process is proportionally adjusted to guarantee that the learned contours yield more precise discrete data. Experimental results indicate that our approach achieves strong performance in both language modeling and discrete image generation tasks. In language modeling, our approach surpasses previous state-of-the-art continuous diffusion language models in three translation tasks and a summarization task, while also demonstrating competitive performance compared to auto-regressive transformers. Moreover, our method achieves comparable results to continuous diffusion models when using discrete ordinal pixels and establishes a new state-of-the-art for categorical image generation on the CIFAR-10 dataset.

1 Introduction

Discrete modeling is essential due to the natural prevalence of discreteness in numerous domains, including proteins [Madani et al., 2020, 2023], images [Parmar et al., 2018, Dosovitskiy et al., 2021], and natural language [Sutskever et al., 2014, Brown et al., 2020]. Recent dominant framework for discrete modeling is the Transformer [Vaswani et al., 2017] with an autoregressive manner. While achieving impressive performance, it does suffer from a slow step-by-step generation process, especially for long sequences. Continuous Diffusion models [Sohl-Dickstein et al., 2015, Ho et al., 2020], on the contrary, exhibit the ability to recover high-dimensional data from noise in parallel with limited iteration steps. Although proved to be effective in continuous data generation [Rombach et al., 2022, Kong et al., 2021], they continue to encounter challenges in discrete modeling [Austin et al., 2021, Chen et al., 2023b, Li et al., 2022, Gong et al., 2023b].

In this paper, we reveal a significant discrepancy pertaining to the modeling of discrete data using continuous diffusion models. Current approaches represent a discrete sample with a vector point in the continuous space. The diffusion process learns a neural network to model the probability distributions that recovers this continuous point from Gaussian noise. However, the discrete data actually corresponds to an area in the continuous space rather than a single point, where the oversimplified assumption leads to a mismatch between learned probability contours and the boundary of the discrete area. Take language generation as an example, a word is represented with an embedding vector in the embedding space. To generate this word, it is impractical to strictly enforce the predicted vector to be an exact match to the embedding. On the contrary, vectors around this embedding can also generate

38th Conference on Neural Information Processing Systems (NeurIPS 2024).

 https://doi.org/10.52202/079017-3852

the same word, thereby defining the collective area they encompass as a discrete area of this word. As illustrated in Figure 1A, suppose the learned probability density function is $p_\theta(\mathbf{x})$ and two points $\mathbf{x}^i$ and $\mathbf{x}^o$ are sampled in the same density contour where $p_\theta(\mathbf{x}^i) = p_\theta(\mathbf{x}^o)$. It is obvious that $\mathbf{x}^i$ lies in the discrete area and is able to recover the discrete data while $\mathbf{x}^o$ can not. This means that the diffusion model only learns a simplified scenario that does not match the real probability distribution.

To address the issues above, we proposed to take the boundaries of discrete areas as priors, as shown in Figure 1B, where boundary curves are regarded as oracle contours. As it gradually approaches the discrete boundary, the learned density contours of diffusion models are expected to transform from Gaussian distributions to the boundary distribution. Therefore, we propose to divide the forward process into two steps. First is the boundary estimation where we precisely calculate the stopping time t_0 and position $\mathbf{x}_{t_0}$ at which the forward trajectory cross the boundary. Then we rescale the trajectory for both training and inference stages to make the sampling probability of noisy point $\mathbf{x}_t$ conditioned on the boundary. To make the boundary estimation tractable (appendix A) and eliminate randomness in conditional state transitions $\mathbf{x}_{t_0} \rightarrow \mathbf{x}_t$, we utilize the Ordinary Differential Equations (ODEs) to describe the forward trajectory.

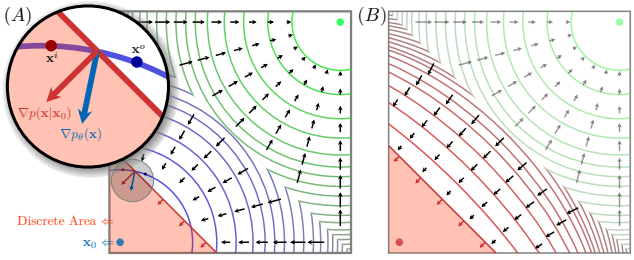


Figure 1: (A) Blue and green curves are the learned probability density contours of the diffusion model for two data points. The red area is the discrete area of the blue data $\mathbf{x}_0$ and the boundary of this area is naturally a density contour. The discrete boundary is a complex hypersurface in the high-dimensional continuous space and we simplify it into a red line for convenience of description. As observed in the magnified part, the learned contours deviate from the boundary contour, resulting in inconsistent probability densities and gradient directions. (B) We consider the discrete boundary as priors for the diffusion process to estimate a more appropriate probability distribution, where the learned contours are expected to follow the shape of the discrete boundary.

Our approach is experimented in both language modeling and discrete image generation. On three machine translation datasets (IWSLT14 DE-EN [Cettolo et al., 2012], WMT14 EN-DE, WMT16 EN-RO) and a text summarization dataset (GIGAWORD [Rush et al., 2015]) for language modeling, our proposed approach not only significantly improves existing diffusion models to at most 7.8% but also achieves competitive performance to autoregressive transformers. For image generation on CIFAR-10 [Krizhevsky et al., 2009], our model realizes a comparable result to continuous diffusion models with discrete ordinal pixels and establishes a new state-of-the-art for categorical pixels.

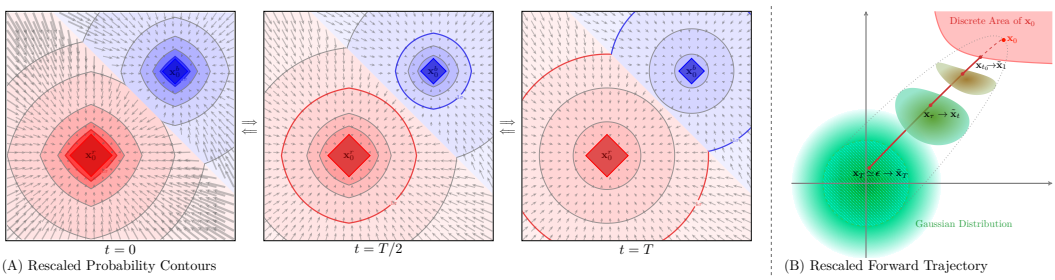
2 Preliminaries

Diffusion Models To model a real distribution $q(\mathbf{x}_0)$, diffusion models utilize a forward process $p_t(\mathbf{x}|\mathbf{x}_0)$ with T steps to gradually add Gaussian noise $\pi(\mathbf{x}) = \mathcal{N}(\mathbf{0}, \mathbf{I})$ into the data distribution, where $p_T(\mathbf{x}|\mathbf{x}_0) = \pi(\mathbf{x})$. There are different architectures for the forward process. A common approach [Ho et al., 2020] considers the forward process as the Markovian process, where $p_t(\mathbf{x}|\mathbf{x}_0) = \prod_{s=1}^{t} p_s(\mathbf{x}_s|\mathbf{x}_{s-1})$ combines a series of Gaussian distributions. Thus the forward process follows a Gaussian distribution that $p_t(\mathbf{x}|\mathbf{x}_0) = \mathcal{N}(\sqrt{\bar{\alpha}_t}\mathbf{x}_0, (1-\bar{\alpha}_t)\mathbf{I})$ (Variance Preserving) or $p_t(\mathbf{x}|\mathbf{x}_0) = \mathcal{N}(\mathbf{x}_0, \sigma_t^2\mathbf{I})$ (Variance Exploding) [Song et al., 2021b], where noise scheduler $\bar{\alpha}_t$ monotonically decreases from 1 to 0 and σ_t increases from sufficiently small to the maximum pairwise distance between all training data points. To recover data from noise, diffusion processes train neural networks $\mathbf{x}_\theta(\mathbf{x}_t, t)$ to predict $\mathbf{x}_0$ (other equivalent targets include $\boldsymbol{\epsilon}$ and $\nabla \log p(\mathbf{x}_t)$) from $\mathbf{x}_t \sim p_t(\mathbf{x}|\mathbf{x}_0)$:

$$\mathcal{L}_\theta = \mathbb{E}_{t\sim\mathcal{U}_{(1,T)}, \mathbf{x}_0\sim q(\mathbf{x}_0), \mathbf{x}_t\sim p_t(\mathbf{x}|\mathbf{x}_0)} \left[\|\mathbf{x}_0 - \mathbf{x}_\theta(\mathbf{x}_t, t)\|^2 \right]. \tag{1}$$

Samples are generated with a series of reverse state transition $p(\mathbf{x}_{t-1}|\mathbf{x}_t, \mathbf{x}_\theta(\mathbf{x}_t, t))$.

Flow Matching Another architecture [Lipman et al., 2023] utilizes the ODEs and defines a time-dependent flow function $\phi_t(\mathbf{x}) = \sigma_t(\mathbf{x}_0)\mathbf{x} + \mu_t(\mathbf{x}_0)$ that maps $p_t(\mathbf{x}|\mathbf{x}_0) = [\phi_t]_* \pi(\mathbf{x}) = \pi(\phi_t^{-1}(\mathbf{x})) \left| \det \frac{\mathrm{d}\phi_t^{-1}(\mathbf{x})}{\mathrm{d}\mathbf{x}} \right| = \mathcal{N}(\mu_t(\mathbf{x}_0), \sigma_t^2(\mathbf{x}_0)\mathbf{I})$, where μ_t and σ_t can be the same as in diffusion



(A) Rescaled Probability Contours (B) Rescaled Forward Trajectory

Figure 2: (A) Rescaled Probability Contours. The bold curve 1σ is the density contour of one standard deviation. As the time t decreases from T to 0, the rescaled contours will gradually fit the discrete boundary and probability densities will also concentrate to this boundary. (B) Rescaled Forward Trajectory. Original forward trajectory $\mathbf{x}_0 \to \mathbf{x}_{t_0} \to \mathbf{x}_\tau$ is rescaled to be a boundary conditional trajectory $\tilde{\mathbf{x}}_1 \to \tilde{\mathbf{x}}_t$ that starts from $\tilde{\mathbf{x}}_1 = \mathbf{x}_{t_0}$. The rescaled forward distribution $\tilde{p}_t(\tilde{\mathbf{x}}_t|\mathbf{x}_0)$ is transformed from the discrete boundary to Gaussian distributions.

models or a more straightforward form that $\mu_t = (1 - \frac{t}{T})\mathbf{x}_0$ and $\sigma_t = \frac{t}{T}$. Recovering data from noises relies on the vector field $u_t(\mathbf{x}|\mathbf{x}_0)$ that generates the probability path with the ODE $\mathrm{d}\phi_{T-t}(\mathbf{x}) = u_{T-t}(\phi_{T-t}(\mathbf{x})|\mathbf{x}_0)\mathrm{d}t, t : 0 \to T$. Neural networks $u_\theta(\mathbf{x}, t)$ are trained to estimate the vector field $u_t(\mathbf{x}|\mathbf{x}_0)$ via the following objective:

$$\mathcal{L}_\theta = \mathbb{E}_{t\sim\mathcal{U}_{(1,T)},\mathbf{x}_0\sim q(\mathbf{x}_0),\mathbf{x}_T\sim\pi(\mathbf{x})}\left[\left\|u_\theta(\phi_t(\mathbf{x}_T), t) - \frac{\mathrm{d}\phi_t(\mathbf{x}_T)}{\mathrm{d}t}\right\|^2\right]. \tag{2}$$

Besides, the vector field is proved to have the form:

$$u_t(\mathbf{x}|\mathbf{x}_0) = \frac{\sigma_t'(\mathbf{x}_0)}{\sigma_t(\mathbf{x}_0)}\left(\mathbf{x} - \mu_t(\mathbf{x}_0)\right) + \mu_t'(\mathbf{x}_0), \text{ where apostrophe indicates derivative to } t. \tag{3}$$

3 Methodology

As illustrated in Figure 2, our objective is to refine the probability density contours of $p_t(\mathbf{x}|\mathbf{x}_0)$ so that they better fit the boundaries of discrete samples while still allowing for the ease of sampling. Let $\mathbf{x}_0$ denote the samples from a real distribution $q(\mathbf{x}_0)$. Obtaining a boundary-aware corresponding noisy data $\mathbf{x}$ at time $t \in [1, T]$ is $p_t(\mathbf{x}|\mathbf{x}_0) = \int p_t(\mathbf{x}, \mathbf{x}_{t_0}, t_0|\mathbf{x}_0)\mathrm{d}\mathbf{x}_{t_0}\mathrm{d}t_0$, where t_0 is a random variable distributed according to when the diffusion trajectory and the discrete boundary intersect, and $\mathbf{x}_{t_0}$ is the corresponding sample point at t_0. Then the forward process is rescaled in two steps:

$$\tilde{p}_t(\mathbf{x}|\mathbf{x}_0) = \int \underbrace{\tilde{p}_t(\mathbf{x}|\mathbf{x}_{t_0}, t_0, \mathbf{x}_0)}_{\text{Trajectory Rescaling}} \quad \underbrace{p(\mathbf{x}_{t_0}, t_0|\mathbf{x}_0)}_{\text{Boundary Estimation}} \mathrm{d}\mathbf{x}_{t_0}\mathrm{d}t_0, \tag{4}$$

where the latter term is to calculate the discrete boundaries and the former term is to rescale the forward trajectory. In order to make the equation tractable and ensure that $\mathbf{x}$ and $\mathbf{x}_{t_0}$ are on the same trajectory, we model the forward process with flow functions $\phi_t(\mathbf{x})$ and extend the notation as:

$$\psi_t(\mathbf{x}) = \mathbf{u}(\mathbf{x}_0, t)\,\mathbf{x}_0 + \mathbf{v}(\mathbf{x}_0, t)\,\mathbf{x}, \quad p_t(\mathbf{x}|\mathbf{x}_0) = [\psi_t]_*\pi(\mathbf{x}) \tag{5}$$

where $\mathbf{u}(\cdot)$ and $\mathbf{v}(\cdot)$ are coefficient functions and sampling $\mathbf{x}_t$ from $p_t(\mathbf{x}|\mathbf{x}_0)$ equals to

$$\mathbf{x}_t = \psi_t(\boldsymbol{\epsilon}), \quad \boldsymbol{\epsilon} \sim \pi(\mathbf{x}) = \mathcal{N}(\mathbf{0}, \mathbf{I}). \tag{6}$$

3.1 Estimate Discrete Boundaries

Before figuring out the joint distribution $p(\mathbf{x}_{t_0}, t_0|\mathbf{x}_0)$, let's start by discussing how to verify whether an arbitrary point $\mathbf{x}$ in the continuous space belongs to the discrete area of $\mathbf{x}_0$. Suppose $\mathbf{x}_0$, which exists in the continuous space S, is the representation vector of a discrete random variable $\mathcal{I}$ in a discrete space with K states. Besides, $\mathcal{J}$ is another discrete random variable *i.i.d.* with $\mathcal{I}$. We define the discrete area of $\mathbf{x}_0$ in the continuous space S as:

$$C_\mathcal{I} = \{\forall \mathbf{x} \in S | f(\mathbf{x}, \mathcal{I}) > f(\mathbf{x}, \mathcal{J}), \forall \mathcal{J} \neq \mathcal{I}\}, \tag{7}$$

https://doi.org/10.52202/079017-3852

where $f(\mathbf{x}, \mathcal{I})$ is a function assessing the likelihood of an arbitrary continuous point $\mathbf{x}$ inside the discrete area of $\mathbf{x}_0$. For instance, in language modeling, K is the vocabulary size. $\mathcal{I}, \mathcal{J} \in K^n$ are two different sequences of n tokens and $\mathbf{x}_0 \in \mathbb{R}^{[n,m]}$ is a sequence of m-dimensional vector embeddings for $\mathcal{I}$. $f(\mathbf{x}, \mathcal{I})$ is the dot similarity function. $C_\mathcal{I}$ collects all vectors in the embedding space that will be decoded to generate $\mathcal{I}$ and excludes vectors associated with any other token sequences $\mathcal{J}$.

Given a noisy point $\mathbf{x}_{t_0}$ locating at the boundary between $C_\mathcal{I}$ and $C_\mathcal{J}$, we can get $|f(\mathbf{x}_{t_0}, \mathcal{I}) - f(\mathbf{x}_{t_0}, \mathcal{J})| = 0$ based on previous definition. Replacing $\mathbf{x}_{t_0}$ with eqs. (5) and (6), there is:

$$f(\mathbf{u}_{t_0}\mathbf{x}_0 + \mathbf{v}_{t_0}\boldsymbol{\epsilon}, \mathcal{I}) = f(\mathbf{u}_{t_0}\mathbf{x}_0 + \mathbf{v}_{t_0}\boldsymbol{\epsilon}, \mathcal{J}). \tag{8}$$

In language modeling and categorical images, $f(\cdot)$ is a linear projection function that:

$$\mathbf{u}_{t_0}(f(\mathbf{x}_0, \mathcal{I}) - f(\mathbf{x}_0, \mathcal{J})) = \mathbf{v}_{t_0}(f(\boldsymbol{\epsilon}, \mathcal{J}) - f(\boldsymbol{\epsilon}, \mathcal{I})). \tag{9}$$

Further simplification of this equation can not be universally applied to all arbitrary forms of $\mathbf{u}_{t_0}$ and $\mathbf{v}_{t_0}$. Therefore, we calculate separately for several commonly occurring special cases.

Diffusion Process For variance preserving, there is $\mathbf{u}_t^2 + \mathbf{v}_t^2 = 1$ and we have:

$$\mathbf{u}_{t_0} = 1 \Bigg/ \sqrt{1 + \left(\frac{f(\mathbf{x}_0, \mathcal{I}) - f(\mathbf{x}_0, \mathcal{J})}{f(\boldsymbol{\epsilon}, \mathcal{J}) - f(\boldsymbol{\epsilon}, \mathcal{I})}\right)^2} \text{ and } \mathbf{v}_{t_0} = 1 \Bigg/ \sqrt{1 + \left(\frac{f(\boldsymbol{\epsilon}, \mathcal{J}) - f(\boldsymbol{\epsilon}, \mathcal{I})}{f(\mathbf{x}_0, \mathcal{I}) - f(\mathbf{x}_0, \mathcal{J})}\right)^2}. \tag{10}$$

For variance exploding, there are $\mathbf{u}_t = 1$ and $\mathbf{v}_t = \sigma_t$. We can obtain:

$$\mathbf{u}_{t_0} = 1 \text{ and } \mathbf{v}_{t_0} = \left(f(\boldsymbol{\epsilon}, \mathcal{J}) - f(\boldsymbol{\epsilon}, \mathcal{I})\right) / \left(f(\mathbf{x}_0, \mathcal{I}) - f(\mathbf{x}_0, \mathcal{J})\right). \tag{11}$$

Flow Matching For optimal transport, there is $\mathbf{u}_t + \mathbf{v}_t = 1$ and similarly we get:

$$\mathbf{u}_{t_0} = 1 \Bigg/ \left(1 + \frac{f(\mathbf{x}_0, \mathcal{I}) - f(\mathbf{x}_0, \mathcal{J})}{f(\boldsymbol{\epsilon}, \mathcal{J}) - f(\boldsymbol{\epsilon}, \mathcal{I})}\right) \text{ and } \mathbf{v}_{t_0} = 1 \Bigg/ \left(1 + \frac{f(\boldsymbol{\epsilon}, \mathcal{J}) - f(\boldsymbol{\epsilon}, \mathcal{I})}{f(\mathbf{x}_0, \mathcal{I}) - f(\mathbf{x}_0, \mathcal{J})}\right). \tag{12}$$

As a result, t_0 can be directly derived by inverting the coefficient function $\mathbf{u}_t$ or $\mathbf{v}_t$, which depends on the choice of noise scheduling strategies. Since their differences do not affect our results, we omit the detailed calculation (appendix E) and denote this process with a function $G(\cdot)$:

$$t_0 = G(\mathbf{x}_0, \boldsymbol{\epsilon}), \text{ where } \mathbf{u}(\mathbf{x}_0, G(\mathbf{x}_0, \boldsymbol{\epsilon})) = \mathbf{u}_{t_0} \text{ and } \mathbf{v}(\mathbf{x}_0, G(\mathbf{x}_0, \boldsymbol{\epsilon})) = \mathbf{v}_{t_0}. \tag{13}$$

It's worth noting that t_0 is not a scalar but a vector, where the dimension is the number of elements in $\mathbf{x}_0$. If $\mathbf{x}_0$ is a sequence of n tokens, $t_0 \in [1, T]^n$. If $\mathbf{x}_0$ is a RGB image with 3-channel $\times$ h-height $\times$ w-width of pixels, $t_0 \in [1, T]^{3\times h\times w}$. Furthermore, the corresponding noisy sample $\mathbf{x}_{t_0}$ is derived as:

$$\mathbf{x}_{t_0} = \mathbf{u}(\mathbf{x}_0, G(\mathbf{x}_0, \boldsymbol{\epsilon}))\mathbf{x}_0 + \mathbf{v}(\mathbf{x}_0, G(\mathbf{x}_0, \boldsymbol{\epsilon}))\boldsymbol{\epsilon} = \psi_{G(\mathbf{x}_0,\boldsymbol{\epsilon})}(\boldsymbol{\epsilon}), \tag{14}$$

which is a time-independent function of the Gaussian noise $\boldsymbol{\epsilon}$. It's worth mentioning that both $p(t_0|\mathbf{x}_0)$ and $p(\mathbf{x}_{t_0}|\mathbf{x}_0)$ are intractable, since $G(\mathbf{x}_0, \boldsymbol{\epsilon})$ and $\psi_{G(\mathbf{x}_0,\boldsymbol{\epsilon})}(\boldsymbol{\epsilon})$ are not invertible to $\boldsymbol{\epsilon}$. Different $\boldsymbol{\epsilon}$s can be mapped to a same t_0 or $\mathbf{x}_{t_0}$. Fortunately, there is an one-to-one mapping between $\boldsymbol{\epsilon}$ and the $[\mathbf{x}_{t_0}; t_0]$ pair. We denote the boundary flow function and the corresponding inversion as

$$\Psi(\boldsymbol{\epsilon}) = [\psi_{G(\mathbf{x}_0,\boldsymbol{\epsilon})}(\boldsymbol{\epsilon}); G(\mathbf{x}_0, \boldsymbol{\epsilon})], \qquad \Psi^{-1}([\mathbf{x}_{t_0}; t_0]) = (\mathbf{x}_{t_0} - \mathbf{u}(\mathbf{x}_0, t_0)\mathbf{x}_0)/\mathbf{v}(\mathbf{x}_0, t_0), \tag{15}$$

and the joint boundary distribution is calculated as

$$p(\mathbf{x}_{t_0}, t_0|\mathbf{x}_0) = [\Psi]_* \pi([\mathbf{x}_{t_0}; t_0]). \tag{16}$$

The support set of $\mathbf{x}_{t_0}$ is restricted to the boundary contour, while other regions in the space are assigned a probability of 0. To obtain the complete boundary, it is necessary to iterate over all possible choices of $\mathcal{J}$ and perform pairwise comparisons with $\mathcal{I}$. The complexity is $O(n \times K)$, where n elements in $\mathbf{x}_0$ is independently iterated. In practical implementation, obtaining the tightest boundary only requires one step of parallel calculation and an extra $\min(\cdot)$ function over all t_0 candidates.

Confidence Factor The discrete area defined by eq. (7) represents an ideal scenario in which the confidence of the boundary is insufficiently reliable for practical application. Due to the intractability of obtaining the probability density function across the entire discrete area and calculating its confidence interval, we employ an empirical strategy. This approach involves utilizing a confidence factor, denoted as r, ranging from 0 to 1, which is multiplied by t_0 to strike a balance between confidence and discreteness. Therefore, $r = 0$ implies the exclusion of discrete priors, causing the discrete area to collapse into a single point, which is the original diffusion process. As the value of r increases, the modeling of discrete boundaries improves at the expense of reliability. Empirically, when the model is conditioned with good guidance, setting a larger value for r allows us to obtain better discrete priors. However, in the case of unconditional modeling, maintaining reliability becomes more crucial to prevent oscillations and even collapses during training.

3.2 Rescale the Forward Trajectory

In this section, we introduce how to formulate the forward trajectory conditioned on discrete boundaries and derive the rescaled noisy sampling distribution. We start with the boundary-independent forward process $p_t(\mathbf{x}|\mathbf{x}_0)$. Let $\mathbf{x}_t$ denote a noisy point at time t sampled from $p_t(\mathbf{x}|\mathbf{x}_0)$, there is $\boldsymbol{\epsilon}_t = (\mathbf{x}_t - \mathbf{u}(\mathbf{x}_0, t)\mathbf{x}_0) / \mathbf{v}(\mathbf{x}_0, t)$ given eq. (5). Equations (13) and (14) provide the corresponding $[\mathbf{x}_{t_0}; t_0]$ pair on the same trajectory, which is deterministically calculated with no randomness:

$$[\mathbf{x}_{t_0}; t_0] = \Psi(\boldsymbol{\epsilon}_t), \text{ where } \boldsymbol{\epsilon}_t = (\mathbf{x}_t - \mathbf{u}(\mathbf{x}_0, t)\mathbf{x}_0) / \mathbf{v}(\mathbf{x}_0, t). \tag{17}$$

To model the transition probability $p_t(\mathbf{x}_{t_0}, t_0|\mathbf{x}_t, \mathbf{x}_0)$, we utilize the Dirac delta function $\delta(\mathbf{x}) \simeq \lim_{\sigma\to 0}\mathcal{N}(\mathbf{0}, \sigma^2\mathbf{I})$, which can be loosely thought of as aggregating all probability densities toward the origin, assigning an infinite density at the origin and zero densities elsewhere. Therefore, we have $p_t(\mathbf{x}_{t_0}, t_0|\mathbf{x}_t, \mathbf{x}_0) = \delta\left([\mathbf{x}_{t_0}; t_0] - \Psi(\boldsymbol{\epsilon}_t)\right)$. Then the forward process, conditioned on the discrete boundary, is simply derived via Bayes' rule:

$$p_t(\mathbf{x}_t|\mathbf{x}_{t_0}, t_0, \mathbf{x}_0) = p_t(\mathbf{x}_{t_0}, t_0|\mathbf{x}_t, \mathbf{x}_0)\frac{p_t(\mathbf{x}_t|\mathbf{x}_0)}{p(\mathbf{x}_{t_0}, t_0|\mathbf{x}_0)} = \begin{cases} 0, & [\mathbf{x}_{t_0}; t_0] \neq \Psi(\boldsymbol{\epsilon}_t) \\ +\infty \times \dfrac{p_t(\mathbf{x}_t|\mathbf{x}_0)}{p(\mathbf{x}_{t_0}, t_0|\mathbf{x}_0)}, & \text{otherwise} \end{cases}. \tag{18}$$

Since $p_t(\mathbf{x}_t|\mathbf{x}_0) > 0$ and $p(\mathbf{x}_{t_0}, t_0|\mathbf{x}_0) > 0$, $p_t(\mathbf{x}_t|\mathbf{x}_{t_0}, t_0, \mathbf{x}_0)$ is also a delta function that

$$p_t(\mathbf{x}_t|\mathbf{x}_{t_0}, t_0, \mathbf{x}_0) = \delta\left(\mathbf{x}_t - \mathbf{u}(\mathbf{x}_0, t)\mathbf{x}_0 - \mathbf{v}(\mathbf{x}_0, t)\Psi^{-1}([\mathbf{x}_{t_0}; t_0])\right). \tag{19}$$

Based on the translation property of the Dirac delta function, i.e. $\int f(x)\delta(x-a)\mathrm{d}x = f(a)$, the original forward process $p_t(\mathbf{x}_t|\mathbf{x}_0) = [\psi_t \circ \Psi^{-1} \circ \Psi]_*\pi(\mathbf{x}_t) = [\psi_t]_*\pi(\mathbf{x}_t)$ naturally ignores the influence of discrete boundaries, even if the boundary information is explicitly added as a condition.

To enable the discrete priors, we propose a simple and intuitive approach: rescale the forward trajectory. As shown in Figure 2B, the original forward process flows from $\mathbf{x}_0$ to a random noise $\boldsymbol{\epsilon}$, and we reset the starting point to $\mathbf{x}_{t_0}$. Accordingly, the intermediate noisy points $\mathbf{x}_t, t \in [1, T]$ will be proportionally mapped on this new path, which is

$$\begin{aligned} \tilde{\mathbf{x}}_t = \mathbf{x}_\tau, \quad \tau = \mathcal{T}(t, t_0) &= r \times t_0 + t \times (T - r \times t_0)/T \\ &= \mathbf{u}(\mathbf{x}_0, \mathcal{T}(t, t_0))\mathbf{x}_0 + \mathbf{v}(\mathbf{x}_0, \mathcal{T}(t, t_0))\Psi^{-1}\left([\mathbf{x}_{t_0}; t_0]\right). \end{aligned} \tag{20}$$

Similar to eq. (19), the rescaled conditional forward process is a Dirac delta function:

$$\tilde{p}_t(\tilde{\mathbf{x}}_t|\mathbf{x}_{t_0}, t_0, \mathbf{x}_0) = \delta\left(\tilde{\mathbf{x}}_t - \mathbf{u}(\mathbf{x}_0, \mathcal{T}(t, t_0))\mathbf{x}_0 - \mathbf{v}(\mathbf{x}_0, \mathcal{T}(t, t_0))\Psi^{-1}\left([\mathbf{x}_{t_0}; t_0]\right)\right). \tag{21}$$

However, $\tilde{p}_t(\tilde{\mathbf{x}}_t|\mathbf{x}_0)$ faces the same problem of irreversibility as in eq. (14) and we derive it as:

$$\begin{aligned} \tilde{p}_t(\tilde{\mathbf{x}}_t|\mathbf{x}_0) &= \int \tilde{p}_t(\tilde{\mathbf{x}}_t, \tau|\mathbf{x}_0)\mathrm{d}\tau = \int \tilde{p}_t(\tilde{\mathbf{x}}_t, \tau|\mathbf{x}_{t_0}, t_0, \mathbf{x}_0)p(\mathbf{x}_{t_0}, t_0|\mathbf{x}_0)\mathrm{d}[\mathbf{x}_{t_0}; t_0]\mathrm{d}\tau \\ &= \int [\psi_\tau \circ \Psi^{-1} \circ \Psi]_*\pi([\tilde{\mathbf{x}}_t; \tau])\mathrm{d}\tau = \int [\psi_\tau]_*\pi([\tilde{\mathbf{x}}_t; \tau])\mathrm{d}\tau. \end{aligned} \tag{22}$$

Obtaining the probability density function requires gathering together the probability densities of the same location $\tilde{\mathbf{x}}_t$ with different τ, which is intractable. Fortunately, we only need to sample noiy points from this probability distribution $\tilde{\mathbf{x}}_t \sim \tilde{p}_t(\tilde{\mathbf{x}}_t|\mathbf{x}_0)$, which is easy to implement:

$$\tilde{\mathbf{x}}_t = \mathbf{u}\left(\mathbf{x}_0, \mathcal{T}(t, G(\mathbf{x}_0, \boldsymbol{\epsilon}))\right)\mathbf{x}_0 + \mathbf{v}(\mathbf{x}_0, \mathcal{T}(t, G(\mathbf{x}_0, \boldsymbol{\epsilon})))\boldsymbol{\epsilon}, \quad \boldsymbol{\epsilon} \sim \pi(\mathbf{x}). \tag{23}$$

3.3 Recover Data from Noise

https://doi.org/10.52202/079017-3852

Training Objective Theoretically, the diffusion neural networks can be trained as in eq. (2), where the rescaled vector field is derived as $\tilde{u}_t = \frac{\mathrm{d}\tilde{\mathbf{x}}_t}{\mathrm{d}t} = \frac{\mathrm{d}\tilde{\mathbf{x}}_t}{\mathrm{d}\tau}\frac{\mathrm{d}\tau}{\mathrm{d}t}$. However, since a low error estimation on $\mathbf{x}_0$ is of significant importance to our trajectory rescaling method, according to eqs. (10) to (13), we convert the objective to an upper bound of the eq. (2) (See appendix F for more details) and train a neural network $\mathbf{x}_\theta(\tilde{\mathbf{x}}_t, t)$ to predict $\mathbf{x}_0$ directly:

Algorithm 1 Training

1: **repeat**
2: $\quad\mathbf{x}_0 \sim q(\mathbf{x}_0)$, $\boldsymbol{\epsilon} \sim \pi(\mathbf{x}) = \mathcal{N}(\mathbf{0}, \mathbf{I})$
3: $\quad t \sim \text{Uniform}(\{1, \dots, T\})$
4: $\quad\tau \coloneqq \mathcal{T}\left(t, G(\mathbf{x}_0, \boldsymbol{\epsilon})\right)$ // eqs. (13) and (20)
5: $\quad\tilde{\mathbf{x}}_t \coloneqq \mathbf{u}(\mathbf{x}_0, \tau)\mathbf{x}_0 + \mathbf{v}(\mathbf{x}_0, \tau)\boldsymbol{\epsilon}$ // eq. (23)
6: $\quad$Take gradient descent step on $\nabla_\theta ||\mathbf{x}_0 - \mathbf{x}_\theta(\tilde{\mathbf{x}}_t, t)||^2$ // eq. (24)
7: **until** converged

$$\mathcal{L}_\theta = \mathbb{E}_{\mathbf{x}_0 \sim q(\mathbf{x}_0), t \sim \mathcal{U}_{(1,T)}, \tilde{\mathbf{x}}_t \sim \tilde{p}_t(\mathbf{x}|\mathbf{x}_0)} \left[\|\mathbf{x}_0 - \mathbf{x}_\theta(\tilde{\mathbf{x}}_t, t)\|^2\right]. \tag{24}$$

The training procedure is demonstrated in algorithm 1 and key steps are summarized in the line 4.

Reverse Process A direct approach that follows the flow matching is to solve the ODE of $\mathrm{d}\psi_{T-t}(\mathbf{x}) = \tilde{u}_{T-t}(\psi_{T-t}(\mathbf{x})|\mathbf{x}_0)\mathrm{d}t, \psi_T(\mathbf{x}) \sim \pi(\mathbf{x})$. This form of transformation is inefficient with $\mathbf{x}_0$-prediction during inference because we have to solve the equation of $\tau = \mathcal{T}\left(t, G\left(\mathbf{x}_\theta, \frac{\tilde{\mathbf{x}}_t - \mathbf{u}(\mathbf{x}_\theta, \tau)\mathbf{x}_\theta}{\mathbf{v}(\mathbf{x}_\theta, \tau)}\right)\right)$ to get the τ with respect to the change of $\tilde{\mathbf{x}}_t$ and $\mathbf{x}_\theta$ in real time. Therefore, we provide a deterministic reverse process as an alternative, which is a special case of DDIM [Song et al., 2021a] or the ODE with discrete timesteps. Given the time intervals $\Delta t \in [\Delta t_1, \dots \Delta t_s], \sum \Delta t = T$, we generalize the boundary conditions $[\mathbf{x}_{t_0}; t_0]$ in $\tilde{p}_t(\tilde{\mathbf{x}}_t|\mathbf{x}_{t_0}, t_0, \mathbf{x}_0)$ of eq. (21) and $\Psi^{-1}([\mathbf{x}_{t_0}; t_0])$ of eq. (15) to any arbitrary condition pairs $[\tilde{\mathbf{x}}_t; \tau]$ and obtain the reverse process:

Algorithm 2 Sampling

1: $t \coloneqq T, \tau \coloneqq T$
2: $\hat{\boldsymbol{\epsilon}} \simeq \tilde{\mathbf{x}}_t \sim \mathcal{N}(\mathbf{0}, \mathbf{I})$ // Initialing
3: **for** $\Delta t \coloneqq \Delta t_1, \dots, \Delta t_s$ **do** // $\sum_{\Delta t} = T$
4: $\quad\hat{\mathbf{x}}_0 \coloneqq \mathbf{x}_\theta(\tilde{\mathbf{x}}_t, t)$ // Pseudo Target
5: $\quad t \coloneqq t - \Delta t$ // Updating
6: $\quad\tau \coloneqq \mathcal{T}\left(t, G(\hat{\mathbf{x}}_0, \hat{\boldsymbol{\epsilon}})\right)$ // eq. (25)
7: $\quad\tilde{\mathbf{x}}_t \coloneqq \mathbf{u}(\hat{\mathbf{x}}_0, \tau)\hat{\mathbf{x}}_0 + \mathbf{v}(\hat{\mathbf{x}}_0, \tau)\hat{\boldsymbol{\epsilon}}$
8: $\quad\hat{\epsilon} \coloneqq \Psi^{-1}([\tilde{\mathbf{x}}_t; \tau])$ // Trajectory Alteration
9: **end for**
10: $\mathbf{x}_0 \coloneqq \mathbf{x}_\theta(\tilde{\mathbf{x}}_t, t)$ // $\mathbf{x}_1 \to \mathbf{x}_0$
11: **return** $\mathbf{x}_0$

$$\begin{aligned}&\tilde{p}([\tilde{\mathbf{x}}_{t-\Delta t}; \tau_\Delta]|[\tilde{\mathbf{x}}_t; \tau], \hat{\mathbf{x}}_0) = \\ &\quad\delta\left(\begin{bmatrix}\tilde{\mathbf{x}}_{t-\Delta t} \\ \tau_\Delta\end{bmatrix} - \begin{bmatrix}\mathbf{u}(\hat{\mathbf{x}}_0, \tau_\Delta)\hat{\mathbf{x}}_0 + \mathbf{v}(\hat{\mathbf{x}}_0, \tau_\Delta)\hat{\boldsymbol{\epsilon}} \\ \mathcal{T}\left(t - \Delta t, G(\hat{\mathbf{x}}_0, \hat{\boldsymbol{\epsilon}})\right)\end{bmatrix}\right),\end{aligned} \tag{25}$$

where $\hat{\mathbf{x}}_0 = \mathbf{x}_\theta(\tilde{\mathbf{x}}_t, t)$ and τ_Δ is the previous timestep of τ on the same rescaled trajectory.

Sampling from the reverse process is illustrated in algorithm 2. Similar to the sampling process of DDIM [Song et al., 2021a], it starts from the Gaussian noise, iteratively predicts the pseudo target $\hat{\mathbf{x}}_0$, and updates the reverse trajectory. However, since the τ and $\hat{\boldsymbol{\epsilon}}$ are mutually conditioned, we have to keep track of the t, τ, $\tilde{\mathbf{x}}_t$, and $\hat{\boldsymbol{\epsilon}}$ during each iteration and split the update of $\hat{\boldsymbol{\epsilon}}$ into an asynchronous step (line 8). Because reverse trajectory keeps changing due to different pseudo targets $\hat{\mathbf{x}}_0$ predicted by learned neural networks, which brings severe instability, sometimes simply fixing the initial path (removing the line 8) exhibits better performance in experiments.

4 Language Modeling

Recent diffusion language models [Li et al., 2022, Gong et al., 2023b] inherit the embedding-rounding framework that a sentence with n discrete tokens $W = [w_1, \dots, w_n]$ is embedded to a continuous space via a trainable embedding layer $\textsc{Emb}(W) = [\textsc{Emb}(w_1), \dots, \textsc{Emb}(w_n)]$. The vocabulary set is K that $\forall w_n \in K$. Besides, the token embeddings are used as the target points $\mathbf{x}_0 = [\mathbf{x}_0^1, \dots, \mathbf{x}_0^n]$, $\mathbf{x}_0^n = \textsc{Emb}(w_n)$, for continuous diffusion trajectories. Hence, generating tokens from embeddings is:

$$p(W|\mathbf{x}_0) = \sum\nolimits_{i=1}^n p(w_i|\mathbf{x}_0^i) = \sum\nolimits_{i=1}^n \frac{\exp(f(\mathbf{x}_0^i, w_i))}{\sum_{j \in K}\exp(f(\mathbf{x}_0^i, j))}, \tag{26}$$

where $f(\mathbf{x}, j) = \textsc{Emb}(j) \cdot \mathbf{x}$ is the dot production distance. It's also the function assessing the likelihood of point $\mathbf{x}$ inside the discrete area of j. The coefficient functions follow the DDPM [Ho et al., 2020], which are $\mathbf{u}(\mathbf{x}_0, t) = \sqrt{\bar{\alpha}_t}$ and $\mathbf{v}(\mathbf{x}_0, t) = \sqrt{1 - \bar{\alpha}_t}$. Besides, the objectives are

$$\mathcal{L}_\theta = \mathbb{E}_{W, t, \tilde{\mathbf{x}}_t}\left[\sum\nolimits_{i=1}^n \|\textsc{Emb}(w_i) - \mathbf{x}_\theta(\tilde{\mathbf{x}}_t^i, t)\|^2/n\right] \tag{27}$$

Table 1: Result of BLEU scores on machine translation and ROUGE scores on text summarization.

Models	IWSLT14 DE-EN BLEU (BLEU-1/2/3/4)⇑	WMT14 EN-DE BLEU (BLEU-1/2/3/4)⇑	WMT16 EN-RO BLEU (BLEU-1/2/3/4)⇑	GIGAWORD ROUGE-1/2/L⇑
Auto-Regressive Modeling				
Transformers	34.31 (67.3/41.6/27.9/19.1)	**28.01** (58.2/33.5/21.7/14.6)	34.05 (63.1/39.9/27.6/19.6)	**37.57/18.90**/34.69
Ours+Rerank	**35.02** (68.7/43.3/29.2/20.1)	27.67 (57.9/33.2/21.4/14.3)	**34.33** (63.1/40.1/27.8/19.8)	37.49/18.68/**34.82**
Diffusion Process				
D3PM	27.61 (65.4/37.7/22.8/14.2)	22.94 (54.9/28.8/16.9/10.4)	27.84 (59.8/34.9/22.1/14.5)	33.92/14.96/31.72
DiffuSeq	28.78 (- / - / - / -)	15.37 (- / - / - / -)	25.45 (- / - / - / -)	31.17/12.23/29.24
SeqDiffuSeq	30.03 (- / - / - / -)	17.14 (- / - / - / -)	26.17 (- / - / - / -)	31.90/12.36/29.22
Differmer	31.58 (68.6/41.4/26.7/17.5)	24.80 (58.7/32.0/19.7/12.5)	30.08 (64.4/39.5/26.5/18.2)	35.47/15.17/32.82
SEDD	31.87 (68.7/41.8/27.2/18.0)	24.98 (59.2/32.4/20.1/12.9)	29.38 (62.2/38.0/24.9/16.9)	34.33/15.22/32.06
Dinoiser	31.91 (67.1/40.9/26.7/17.7)	24.77 (57.2/31.0/19.0/12.0)	31.49 (62.8/38.4/25.5/17.3)	35.17/15.63/32.53
Ours	**33.42** (68.0/42.0/27.7/18.6)	**26.69** (57.7/32.3/20.4/13.4)	**33.15** (63.4/39.9/27.4/19.2)	**36.44/16.09/33.56**

and an additional rounding objective, which is commonly used in language modeling,

$$\mathcal{L}_r = -\log p_\theta(W|\mathbf{x}_0) = -\log p_\theta(W|\mathbf{x}_\theta(\tilde{\mathbf{x}}_t, t)). \tag{28}$$

The final training target is given by $\mathcal{L} = \mathcal{L}_\theta + \mathcal{L}_r$, where the $\mathbf{x}_0$ of the same token sequence W keeps changing because the embedding layer EMB is trainable, which makes the model hard to be trained. Since previous work does not model discrete areas, a large number of noisy samples inside this area will make $\mathcal{L}_r$ too small to guide the training of the embedding layer, leading to a mode collapse.

Experimental Setup Datasets used for experiments include three translation tasks (IWSLT14 DE-EN [Cettolo et al., 2012], WMT14 EN-DE, and WMT16 EN-RO[1]) and one text summarization task (GIGAWORD [Rush et al., 2015]). We mainly follow the setting of Gao et al. [2022], which is inherited from previous non-auto-regressive text generation works [Gu et al., 2018, 2019, Ghazvininejad et al., 2019], where translation datasets are distilled [Kim and Rush, 2016]. Baselines are mainly continuous diffusion language models. DiffuSeq [Gong et al., 2023b] and SeqDiffuSeq [Yuan et al., 2022] are derived from Diffusion-LM [Li et al., 2022]. Differmer [Gao et al., 2022] and Dinoiser [Ye et al., 2023] are recent empirical studies highlighting that scaling up the noise is beneficial for language modeling. We also compare with discrete diffusion language models, including D3PM [Austin et al., 2021] and SEDD [Lou et al., 2023]. Since SEDD is a pre-trained language model, we configure its framework and train it from scratch specifically for our tasks. In addition, auto-regressive transformer [Vaswani et al., 2017] is still one of the most powerful architectures for language generation.

Our boundary conditional diffusion language model is constructed from Differmer [Gao et al., 2022], where the model configuration is *transformer-iwslt-de-en* in FAIRSEQ framework [Ott et al., 2019] for IWSLT14 DE-EN and *transformer-base* for other datasets. Sentences are tokenized with Byte-Pair Encoding [Sennrich et al., 2016] and evaluated by detokenized BLEU [Papineni et al., 2002] for machine translation and ROUGE [Lin, 2004] for summarization. During training, the diffusion step is $T = 2000$ and the confidence factor $r = 1$ for translation tasks since they have strong conditions, while $r = 0.5$ for summarization. Sentences are generated deterministically with 20 steps.

Results Performances are demonstrated in Table 1. Our approach achieves the state-of-the-art compared with continuous diffusion language models and outperforms the two discrete baselines on three machine translation and one text summarization tasks. Our method shows advantages, with a 73.6% significant improvement at most on WMT14 EN-DE, over DiffuSeq [Gong et al., 2023b] and SeqDiffuSeq [Yuan et al., 2022], which are two basic methods directly applying diffusion process to language modeling. Compared with recent strong diffusion language models like Differmer [Gao et al., 2022] and Dinoiser [Ye et al., 2023], which have deployed various effective noise scheduling strategies on diffusion processes from the empirical perspective, our model is still superior with at most 3.07 advancement of BLEU score on WMT16 EN-RO. This implies the effectiveness of modeling discrete priors. In addition, we illustrate the performance of auto-regressive modeling, where we use the transformer [Vaswani et al., 2017] to rerank the generated sentence candidates (7

[1] https://github.com/shawnkx/Fully-NAT

https://doi.org/10.52202/079017-3852

Table 3: Analysis on the training objectives.

Objectives	$\mathbb{E}_{\tilde{\mathbf{x}}_t}\|\mathbf{x}_0 - \hat{\mathbf{x}}_0\|^2$	$\mathbb{E}_{\tilde{\mathbf{x}}_t}\|\tilde{u}_t(\tilde{\mathbf{x}}_t\|\mathbf{x}_0) - \tilde{u}_t(\tilde{\mathbf{x}}_t\|\hat{\mathbf{x}}_0)\|^2$	$\mathbb{E}_{\tilde{\mathbf{x}}_t}\left[p(\hat{\mathbf{x}}_0 \in C_{\mathbf{x}_0})\right]$	BLEU
$\mathcal{L}_{\mathbf{x}_0}$ (eq. 24)	8.44	1.56	51.81%	33.42
$\mathcal{L}_{\tilde{u}_t}$	8.41	1.55	52.34%	33.49

length beam $\times$ 3 sentence beams) of our model. The reranked performance can even outperform transformers on IWSLT14 DE-EN and WMT16 EN-RO.

Table 2: Ablation studies.

Models	**IWSLT14**	**WMT16**
Base (Differmer)	31.58	30.08
+ forward only	33.02	32.86
+ forward & reverse	**33.42**	33.15
Optimal Transport	32.77	**33.65**

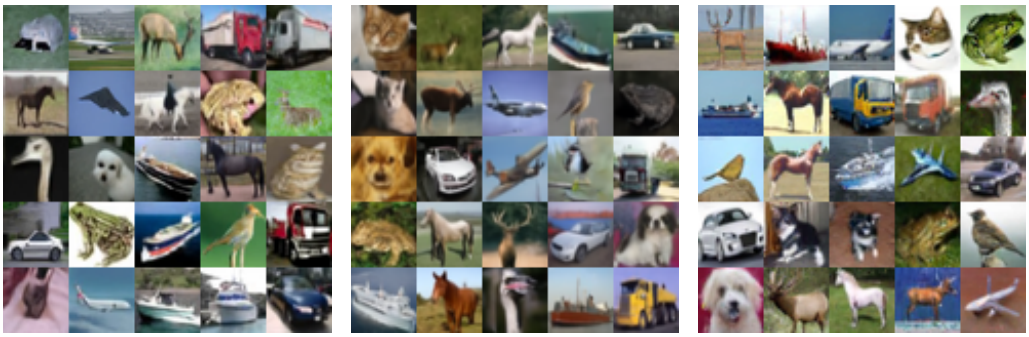
Ablation Our approach is a general framework applicable to almost all continuous diffusion models, providing them with discrete boundaries as priors. We choose Differmer [Gao et al., 2022] as the base model and follow the configurations. As proved in eq. (19), the original forward process will ignore the discrete priors although explicitly demonstrated. We conduct ablation experiments on the rescaling module. As illustrated in Table 2, our approach rescales the trajectory of both forward and reverse processes on Differmer. Only rescaling the forward trajectory is also effective but sub-optimal due to the inconsistent distribution during inference. Due to computational cost and fair comparison, our method leaves room for improvement. For example, replacing the forward trajectory with optimal transport in Flow Matching, $\mathbf{u}(\mathbf{x}_0, t) = 1 - t/T$ and $\mathbf{v}(\mathbf{x}_0, t) = t/T$, achieves better performance on WMT16.

Analysis Our training objective, eq. (24), is an upper bound of the eq. (2). We demonstrate the influence of this approximation in Table 3 on IWSLT14 DE-EN to reveal the thought of our formula. On the one hand, $\mathcal{L}_{\mathbf{x}_0}$ brings theoretical errors at a constant scale. On the other hand, $\mathcal{L}_{\mathbf{x}_0}$ mitigates some experimental errors from the neural networks. The first row $\mathcal{L}_{\mathbf{x}_0}$ is the objective we used in eq. (24) and the second row $\mathcal{L}_{\tilde{u}_t} = \mathbb{E}_{\{t,\mathbf{x}_0,\tilde{\mathbf{x}}_t\}}\left[\|\tilde{u}_t(\tilde{\mathbf{x}}_t|\mathbf{x}_\theta(\tilde{\mathbf{x}}_t, t)) - \frac{\mathrm{d}\tilde{\mathbf{x}}_t}{\mathrm{d}t}\|^2\right]$ is directly derived from the eq. (2). The first two columns represent the error expectations of $\mathbf{x}_0$ and $\tilde{u}_t$ on the test set. It is easy to observe that, with the dynamic coefficient $\frac{\mathrm{d}\tau}{\mathrm{d}t} = \frac{T - r \times G(\mathbf{x}_0, \boldsymbol{\epsilon})}{T}$ (appendix F), the value of $\mathbf{x}_0$'s error (8.44) is much larger than the $\tilde{u}_t$'s error (1.56). Therefore, $\mathcal{L}_{\mathbf{x}_0}$ is beneficial for reducing the impact of the prediction error from the neural network. The third column in Table 3 illustrates the one-step accuracy of predicting $\mathbf{x}_0$ and the fourth column is the BLEU score on the test set. Experimental results show that optimizing the upper bound has a negligible impact on the final performance (only a 0.2% drop of the BLEU score), while can improve the efficiency of the loss calculation during the training phase.

5 Discrete Image Generation

Image pixels are usually treated as real numbers in continuous space since adjacent pixel values exhibit linear continuity.They are essentially discrete and quantized data with a finite state space, such as 256 states in RGB format. We utilize two discrete image representations. One is binary coding provided by Bit Diffusion [Chen et al., 2023b] that converts a sub-pixel with 256 integers to a 8-bit binary code. It is more efficient as it stores ordinal relationships, but the representation space it constructs will be sparse. Another is pixel embedding, which is a more discrete form of representation because the relationships between pixels are thoroughly broken down and reconstructed by learning the embedding representation. Each pixel is regarded as a one-hot vector and transformed with an embedding layer EMB as used in language. Furthermore, we design an intermediate state to demonstrate the correlation between discreteness and modeling difficulty, which is initializing a fixed embedding with binary coding. The optimization target for binary coding is the MSE loss, and pixel embeddings take the same objective as in language.

Experimental Setup We use CIFAR-10 [Krizhevsky et al., 2009] for discrete image generation. The evaluation metric is FID [Heusel et al., 2017], which compares 50K generated samples with the training set. Our image generation model is constructed on Bit Diffusion [Chen et al., 2023b], where the architecture is U-Net [Ronneberger et al., 2015] with 3 stages, 256 channels and 3 residual blocks



(A) Bit Diffusion *repro* (FID 10.37) (B) DDIM (FID 4.04) (C) Ours (FID 3.86)

Figure 3: Generated images of Bit Diffusion *repro*, DDIM, and Ours on CIFAR-10.

per stage. Diffusion steps are $T = 1000$ for both the training and inference stages. The model is trained for 1.5M steps with the learning rate of $1e\text{-}4$ and batch size of 128. Since the training script and detailed hyperparameters of Bit Diffusion are not available, we have to reproduce it by ourselves and our boundary conditional diffusion model shares exactly the same configuration. Our confidence factors are $r = 0.5$ for all three settings. Other baselines include D3PM [Austin et al., 2021] and τLDR [Campbell et al., 2022] which are discrete diffusion models. SDDM [Sun et al., 2023] utilizes vector quantization from VQ-GAN [Esser et al., 2021] as a continuous space for discrete data. We also compare with DDPM [Ho et al., 2020] and DDIM [Song et al., 2021a] on continuous pixels.

Table 4: FID scores on CIFAR-10.

Models	CIFAR-10 (FID ⇓) 200K	500K	Final
Continuous Pixels			
DDPM	-	-	3.17
DDIM	-	-	4.04
Discrete Ordinal Pixels			
D3PM GAUSS	-	-	7.34
τ**LDR**-0	-	-	8.10
τ**LDR**-10	-	-	3.74
BINARY CODING (UINT8):			
Bit Diffusion	-	-	3.48
Bit Diffusion *repro*	22.12	13.23	10.37
Ours	8.17	5.03	**3.86**
FIXED EMBEDDING:			
Bit Diffusion *repro*	19.69	16.61	12.96
Ours	12.32	10.09	**9.15**
Categorical Pixels			
D3PM UNIFORM	-	-	51.27
D3PM ABSORBING	-	-	30.97
VECTOR QUANTIZATION:			
D3PM-VQ	-	-	16.47
τ**LDR**-VQ	-	-	40.06
SDDM-VQ	-	-	12.23
TRAINABLE EMBEDDING:			
Bit Diffusion *repro*	33.09	27.21	19.26
Ours	21.17	15.32	**10.99**

Results For binary coding, as shown in Table 4, our approach outperforms the reproduced Bit Diffusion and attains competitive results to state-of-the-art models. For pixel embedding where ordinal information is deconstructed and reconstituted, our method exhibits a notable improvement of 3.81 FID score over replicated Bit Diffusion. Moreover, in the case of categorical pixels, this advantage increases to 8.25, positioning our approach with trainable embedding as a new state-of-the-art solution. Additionally, as deterministic diffusion processes, our model with binary coding can slightly exceed the performance of DDIM, where the generated samples are in Figure 3.

Analysis We analyze the influence of the confidence factor r in Table 5. The factor r is selected from $[0, 0.2, 0.3, 0.5]$, where $r = 0$ is the reproduced Bit Diffusion that discards the discrete priors. As the confidence factor increases, the impact of discreteness gradually improves, simultaneously enhancing the model's performance across all three settings. Since there is no guidance for unconditional image generation, we do not use a larger factor to prevent mode collapses.

6 Related Work

Discrete Modeling Auto-regressive models have demonstrated a domination over discrete modeling, especially for text generation [Vaswani et al., 2017, Brown et al., 2020, Achiam et al., 2023]. However, the computation

Table 5: Confidence factors.

Models	r = 0	0.2	0.3	0.5
BINARY CODING	10.37	7.39	5.33	3.86
FIXED EMBEDDING	12.96	11.35	10.80	9.15
TRAINABLE EMBEDDING	19.26	15.32	11.56	10.99

https://doi.org/10.52202/079017-3852

cost increases drastically as the size of sentence length or the image resolution increases. Diffusion models [Sohl-Dickstein et al., 2015, Ho et al., 2020, Dhariwal and Nichol, 2021, Saharia et al., 2022] can generate data in parallel, but are tailored for continuous problems. To generalize diffusion models for discrete data, the most straightforward methods define discrete processes in discrete spaces [Sohl-Dickstein et al., 2015, Hoogeboom et al., 2021b, Austin et al., 2021, Campbell et al., 2022, Zhang et al., 2023, Sun et al., 2023, Lou et al., 2023], which will be bothered by large number of discrete status. Besides, a simplified version of discrete diffusion processes is recently used in language modeling [He et al., 2023, Chen et al., 2023a]. Approaches in another line argue to located discrete data in continuous spaces, which is more flexible and efficient, with the mapping functions including binary bits [Chen et al., 2023b] and embeddings [Li et al., 2022, Gong et al., 2023b,a, Yuan et al., 2022, Gulrajani and Hashimoto, 2023, Han et al., 2023]. Other generative models adapted for discrete modeling includes Variational Autoencoders [Kingma and Welling, 2014], Generative Adversarial Networks [Hjelm et al., 2018, Fedus et al., 2018], and Normalizing Flows [Lindt and Hoogeboom, 2021, Hoogeboom et al., 2021a, Tan et al., 2022].

Diffusion Models with Deterministic Trajectory Deterministic diffusion process is usually used in the inference stage to speed up sampling, where DDIM [Song et al., 2021a] derives a serial of non-Markovian diffusion processes and the deterministic one is a special case from this implicit perspective. Additionally, deterministic diffusion processes can be converted to ordinary differential equations [Song et al., 2021b], which is utilized by recent sampling acceleration approaches such as DEIS [Zhang and Chen, 2023] and DPM-Solvers [Lu et al., 2022b,a, Zheng et al., 2023]. Our approach requires a deterministic forward trajectory to eliminate the randomness between the boundary point and sampled point. Flow matching [Liu, 2022, Lipman et al., 2023, Albergo and Vanden-Eijnden, 2023, Liu et al., 2023] is a collection of generative models that employ ordinary differential equations to facilitate both forward and reverse processes. They can be regarded as generally equivalent to Diffusion models. Therefore, we extend the framework of flow matching for our method.

7 Conclusion

We studied the gap between discrete modeling and continuous spaces, focusing on the inconsistency between probability density contours learned by continuous diffusion models and discrete boundaries. We have proposed a novel and general approach to address this issue by enabling continuous diffusion models to be conditioned on discrete priors, which is achieved via discrete boundary estimation and trajectory rescaling. An important limitation is that our method is designed for continuous diffusion models, where discrete diffusion models constructed specially on the discrete state space would not encounter the problem. However, discrete diffusion models also possess their own shortcomings, and the practical applications of continuous diffusion models are more extensive. We believe that our method has the potential to advance the development of unified and general diffusion models. By bridging the gap between discrete and continuous modeling, we hope to inspire new possibilities for modeling complex systems and phenomena.

Acknowledgements

Bing Qin is the corresponding author of this work, We thank the anonymous reviewers for their insightful comments. This work was supported by the National Natural Science Foundation of China (NSFC) (U22B2059, grant 62276078), the Key R&D Program of Heilongjiang via grant 2022ZX01A32, the International Cooperation Project of PCL, PCL2022D01 and the Fundamental Research Funds for the Central Universities (Grant No.HIT.OCEF.2023018).

References

Josh Achiam, Steven Adler, Sandhini Agarwal, Lama Ahmad, Ilge Akkaya, Florencia Leoni Aleman, Diogo Almeida, Janko Altenschmidt, Sam Altman, Shyamal Anadkat, et al. Gpt-4 technical report. *arXiv preprint arXiv:2303.08774*, 2023.

Michael Samuel Albergo and Eric Vanden-Eijnden. Building normalizing flows with stochastic interpolants. In *The Eleventh International Conference on Learning Representations*, 2023. URL `https://openreview.net/forum?id=li7qeBbCR1t`.

Jacob Austin, Daniel D. Johnson, Jonathan Ho, Daniel Tarlow, and Rianne van den Berg. Structured denoising diffusion models in discrete state-spaces. In A. Beygelzimer, Y. Dauphin, P. Liang, and J. Wortman Vaughan, editors, *Advances in Neural Information Processing Systems*, 2021.

Tom Brown, Benjamin Mann, Nick Ryder, Melanie Subbiah, Jared D Kaplan, Prafulla Dhariwal, Arvind Neelakantan, Pranav Shyam, Girish Sastry, Amanda Askell, Sandhini Agarwal, Ariel Herbert-Voss, Gretchen Krueger, Tom Henighan, Rewon Child, Aditya Ramesh, Daniel Ziegler, Jeffrey Wu, Clemens Winter, Chris Hesse, Mark Chen, Eric Sigler, Mateusz Litwin, Scott Gray, Benjamin Chess, Jack Clark, Christopher Berner, Sam McCandlish, Alec Radford, Ilya Sutskever, and Dario Amodei. Language models are few-shot learners. In H. Larochelle, M. Ranzato, R. Hadsell, M.F. Balcan, and H. Lin, editors, *Advances in Neural Information Processing Systems*, volume 33, pages 1877–1901. Curran Associates, Inc., 2020.

Andrew Campbell, Joe Benton, Valentin De Bortoli, Tom Rainforth, George Deligiannidis, and Arnaud Doucet. A continuous time framework for discrete denoising models. In Alice H. Oh, Alekh Agarwal, Danielle Belgrave, and Kyunghyun Cho, editors, *Advances in Neural Information Processing Systems*, 2022.

Mauro Cettolo, Christian Girardi, and Marcello Federico. WIT3: Web inventory of transcribed and translated talks. In Mauro Cettolo, Marcello Federico, Lucia Specia, and Andy Way, editors, *Proceedings of the 16th Annual Conference of the European Association for Machine Translation*, pages 261–268, Trento, Italy, May 28–30 2012. European Association for Machine Translation.

Jiaao Chen, Aston Zhang, Mu Li, Alex Smola, and Diyi Yang. A cheaper and better diffusion language model with soft-masked noise. In Houda Bouamor, Juan Pino, and Kalika Bali, editors, *Proceedings of the 2023 Conference on Empirical Methods in Natural Language Processing*, pages 4765–4775, Singapore, December 2023a. Association for Computational Linguistics.

Ting Chen, Ruixiang Zhang, and Geoffrey Hinton. Analog bits: Generating discrete data using diffusion models with self-conditioning. In *The Eleventh International Conference on Learning Representations*, 2023b.

Prafulla Dhariwal and Alexander Nichol. Diffusion models beat gans on image synthesis. In M. Ranzato, A. Beygelzimer, Y. Dauphin, P.S. Liang, and J. Wortman Vaughan, editors, *Advances in Neural Information Processing Systems*, volume 34, pages 8780–8794. Curran Associates, Inc., 2021.

Alexey Dosovitskiy, Lucas Beyer, Alexander Kolesnikov, Dirk Weissenborn, Xiaohua Zhai, Thomas Unterthiner, Mostafa Dehghani, Matthias Minderer, Georg Heigold, Sylvain Gelly, Jakob Uszkoreit, and Neil Houlsby. An image is worth 16x16 words: Transformers for image recognition at scale. In *International Conference on Learning Representations*, 2021. URL `https://openreview.net/forum?id=YicbFdNTTy`.

Patrick Esser, Robin Rombach, and Bjorn Ommer. Taming transformers for high-resolution image synthesis. In *Proceedings of the IEEE/CVF conference on computer vision and pattern recognition*, pages 12873–12883, 2021.

William Fedus, Ian Goodfellow, and Andrew M. Dai. Maskgan: Better text generation via filling in the _. In *International Conference on Learning Representations*, 2018.

Zhujin Gao, Junliang Guo, Xu Tan, Yongxin Zhu, Fang Zhang, Jiang Bian, and Linli Xu. Difformer: Empowering diffusion model on embedding space for text generation. *arXiv preprint arXiv:2212.09412*, 2022.

Marjan Ghazvininejad, Omer Levy, Yinhan Liu, and Luke Zettlemoyer. Mask-predict: Parallel decoding of conditional masked language models. In Kentaro Inui, Jing Jiang, Vincent Ng, and Xiaojun Wan, editors, *Proceedings of the 2019 Conference on Empirical Methods in Natural Language Processing and the 9th International Joint Conference on Natural Language Processing (EMNLP-IJCNLP)*, pages 6112–6121, Hong Kong, China, November 2019. Association for Computational Linguistics.

 https://doi.org/10.52202/079017-3852

Shansan Gong, Mukai Li, Jiangtao Feng, Zhiyong Wu, and Lingpeng Kong. DiffuSeq-v2: Bridging discrete and continuous text spaces for accelerated Seq2Seq diffusion models. In Houda Bouamor, Juan Pino, and Kalika Bali, editors, *Findings of the Association for Computational Linguistics: EMNLP 2023*, pages 9868–9875, Singapore, December 2023a. Association for Computational Linguistics.

Shansan Gong, Mukai Li, Jiangtao Feng, Zhiyong Wu, and Lingpeng Kong. Diffuseq: Sequence to sequence text generation with diffusion models. In *The Eleventh International Conference on Learning Representations*, 2023b.

Jiatao Gu, James Bradbury, Caiming Xiong, Victor O.K. Li, and Richard Socher. Non-autoregressive neural machine translation. In *International Conference on Learning Representations*, 2018.

Jiatao Gu, Changhan Wang, and Junbo Zhao. Levenshtein transformer. In H. Wallach, H. Larochelle, A. Beygelzimer, F. d'Alché-Buc, E. Fox, and R. Garnett, editors, *Advances in Neural Information Processing Systems*, volume 32. Curran Associates, Inc., 2019.

Ishaan Gulrajani and Tatsunori Hashimoto. Likelihood-based diffusion language models. In *Thirty-seventh Conference on Neural Information Processing Systems*, 2023.

Xiaochuang Han, Sachin Kumar, and Yulia Tsvetkov. SSD-LM: Semi-autoregressive simplex-based diffusion language model for text generation and modular control. In Anna Rogers, Jordan Boyd-Graber, and Naoaki Okazaki, editors, *Proceedings of the 61st Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, pages 11575–11596, Toronto, Canada, July 2023. Association for Computational Linguistics.

Zhengfu He, Tianxiang Sun, Qiong Tang, Kuanning Wang, Xuanjing Huang, and Xipeng Qiu. DiffusionBERT: Improving generative masked language models with diffusion models. In Anna Rogers, Jordan Boyd-Graber, and Naoaki Okazaki, editors, *Proceedings of the 61st Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, pages 4521–4534, Toronto, Canada, July 2023. Association for Computational Linguistics.

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.

R Devon Hjelm, Athul Paul Jacob, Adam Trischler, Gerry Che, Kyunghyun Cho, and Yoshua Bengio. Boundary seeking gans. In *International Conference on Learning Representations*, 2018.

Jonathan Ho, Ajay Jain, and Pieter Abbeel. Denoising diffusion probabilistic models. In H. Larochelle, M. Ranzato, R. Hadsell, M.F. Balcan, and H. Lin, editors, *Advances in Neural Information Processing Systems*, volume 33, pages 6840–6851. Curran Associates, Inc., 2020.

Emiel Hoogeboom, Didrik Nielsen, Priyank Jaini, Patrick Forré, and Max Welling. Argmax flows and multinomial diffusion: Learning categorical distributions. In M. Ranzato, A. Beygelzimer, Y. Dauphin, P.S. Liang, and J. Wortman Vaughan, editors, *Advances in Neural Information Processing Systems*, volume 34, pages 12454–12465. Curran Associates, Inc., 2021a.

Emiel Hoogeboom, Didrik Nielsen, Priyank Jaini, Patrick Forré, and Max Welling. Argmax flows and multinomial diffusion: Learning categorical distributions. In A. Beygelzimer, Y. Dauphin, P. Liang, and J. Wortman Vaughan, editors, *Advances in Neural Information Processing Systems*, 2021b.

Yoon Kim and Alexander M. Rush. Sequence-level knowledge distillation. In Jian Su, Kevin Duh, and Xavier Carreras, editors, *Proceedings of the 2016 Conference on Empirical Methods in Natural Language Processing*, pages 1317–1327, Austin, Texas, November 2016. Association for Computational Linguistics.

Diederik P. Kingma and Max Welling. Auto-encoding variational bayes. In *International Conference on Learning Representations*, 2014.

Zhifeng Kong, Wei Ping, Jiaji Huang, Kexin Zhao, and Bryan Catanzaro. Diffwave: A versatile diffusion model for audio synthesis. In *International Conference on Learning Representations*, 2021.

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

Xiang Lisa Li, John Thickstun, Ishaan Gulrajani, Percy Liang, and Tatsunori Hashimoto. Diffusion-LM improves controllable text generation. In Alice H. Oh, Alekh Agarwal, Danielle Belgrave, and Kyunghyun Cho, editors, *Advances in Neural Information Processing Systems*, 2022.

Chin-Yew Lin. ROUGE: A package for automatic evaluation of summaries. In *Text Summarization Branches Out*, pages 74–81, Barcelona, Spain, July 2004. Association for Computational Linguistics.

Alexandra Lindt and Emiel Hoogeboom. Discrete denoising flows. In *ICML Workshop on Invertible Neural Networks, Normalizing Flows, and Explicit Likelihood Models*, 2021.

Yaron Lipman, Ricky T. Q. Chen, Heli Ben-Hamu, Maximilian Nickel, and Matthew Le. Flow matching for generative modeling. In *The Eleventh International Conference on Learning Representations*, 2023.

Qiang Liu. Rectified flow: A marginal preserving approach to optimal transport. *arXiv preprint arXiv:2209.14577*, 2022.

Xingchao Liu, Chengyue Gong, and Qiang Liu. Flow straight and fast: Learning to generate and transfer data with rectified flow. In *International conference on learning representations (ICLR)*, 2023.

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.

Cheng Lu, Yuhao Zhou, Fan Bao, Jianfei Chen, Chongxuan Li, and Jun Zhu. Dpm-solver++: Fast solver for guided sampling of diffusion probabilistic models. *arXiv preprint arXiv:2211.01095*, 2022a.

Cheng Lu, Yuhao Zhou, Fan Bao, Jianfei Chen, Chongxuan Li, and Jun Zhu. DPM-solver: A fast ODE solver for diffusion probabilistic model sampling in around 10 steps. In Alice H. Oh, Alekh Agarwal, Danielle Belgrave, and Kyunghyun Cho, editors, *Advances in Neural Information Processing Systems*, 2022b.

Ali Madani, Bryan McCann, Nikhil Naik, Nitish Shirish Keskar, Namrata Anand, Raphael R Eguchi, Po-Ssu Huang, and Richard Socher. Progen: Language modeling for protein generation. *arXiv preprint arXiv:2004.03497*, 2020.

Ali Madani, Ben Krause, Eric R Greene, Subu Subramanian, Benjamin P Mohr, James M Holton, Jose Luis Olmos, Caiming Xiong, Zachary Z Sun, Richard Socher, et al. Large language models generate functional protein sequences across diverse families. *Nature Biotechnology*, 41(8): 1099–1106, 2023.

Myle Ott, Sergey Edunov, Alexei Baevski, Angela Fan, Sam Gross, Nathan Ng, David Grangier, and Michael Auli. fairseq: A fast, extensible toolkit for sequence modeling. In *Proceedings of NAACL-HLT 2019: Demonstrations*, 2019.

Kishore Papineni, Salim Roukos, Todd Ward, and Wei-Jing Zhu. Bleu: a method for automatic evaluation of machine translation. In Pierre Isabelle, Eugene Charniak, and Dekang Lin, editors, *Proceedings of the 40th Annual Meeting of the Association for Computational Linguistics*, pages 311–318, Philadelphia, Pennsylvania, USA, July 2002. Association for Computational Linguistics.

Niki J. Parmar, Ashish Vaswani, Jakob Uszkoreit, Lukasz Kaiser, Noam Shazeer, Alexander Ku, and Dustin Tran. Image transformer. In *International Conference on Machine Learning (ICML)*, 2018. URL `http://proceedings.mlr.press/v80/parmar18a.html`.

Robin Rombach, Andreas Blattmann, Dominik Lorenz, Patrick Esser, and Björn Ommer. High-resolution image synthesis with latent diffusion models. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*, pages 10684–10695, June 2022.

 https://doi.org/10.52202/079017-3852

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*, pages 234–241. Springer, 2015.

Alexander M. Rush, Sumit Chopra, and Jason Weston. A neural attention model for abstractive sentence summarization. In Lluís Màrquez, Chris Callison-Burch, and Jian Su, editors, *Proceedings of the 2015 Conference on Empirical Methods in Natural Language Processing*, pages 379–389, Lisbon, Portugal, September 2015. Association for Computational Linguistics.

Chitwan Saharia, William Chan, Saurabh Saxena, Lala Li, Jay Whang, Emily L Denton, Kamyar Ghasemipour, Raphael Gontijo Lopes, Burcu Karagol Ayan, Tim Salimans, Jonathan Ho, David J Fleet, and Mohammad Norouzi. Photorealistic text-to-image diffusion models with deep language understanding. In S. Koyejo, S. Mohamed, A. Agarwal, D. Belgrave, K. Cho, and A. Oh, editors, *Advances in Neural Information Processing Systems*, volume 35, pages 36479–36494. Curran Associates, Inc., 2022.

Rico Sennrich, Barry Haddow, and Alexandra Birch. Neural machine translation of rare words with subword units. In Katrin Erk and Noah A. Smith, editors, *Proceedings of the 54th Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, pages 1715–1725, Berlin, Germany, August 2016. Association for Computational Linguistics.

Jascha Sohl-Dickstein, Eric Weiss, Niru Maheswaranathan, and Surya Ganguli. Deep unsupervised learning using nonequilibrium thermodynamics. In Francis Bach and David Blei, editors, *Proceedings of the 32nd International Conference on Machine Learning*, volume 37 of *Proceedings of Machine Learning Research*, pages 2256–2265, Lille, France, 07–09 Jul 2015. PMLR.

Jiaming Song, Chenlin Meng, and Stefano Ermon. Denoising diffusion implicit models. In *International Conference on Learning Representations*, 2021a.

Yang Song, Jascha Sohl-Dickstein, Diederik P Kingma, Abhishek Kumar, Stefano Ermon, and Ben Poole. Score-based generative modeling through stochastic differential equations. In *International Conference on Learning Representations*, 2021b.

Haoran Sun, Lijun Yu, Bo Dai, Dale Schuurmans, and Hanjun Dai. Score-based continuous-time discrete diffusion models. In *The Eleventh International Conference on Learning Representations*, 2023.

Ilya Sutskever, Oriol Vinyals, and Quoc V Le. Sequence to sequence learning with neural networks. *Advances in neural information processing systems*, 27, 2014.

Shawn Tan, Chin-Wei Huang, Alessandro Sordoni, and Aaron Courville. Learning to dequantise with truncated flows. In *International Conference on Learning Representations*, 2022.

Ashish Vaswani, Noam Shazeer, Niki Parmar, Jakob Uszkoreit, Llion Jones, Aidan N Gomez, Ł ukasz Kaiser, and Illia Polosukhin. Attention is all you need. In I. Guyon, U. Von Luxburg, S. Bengio, H. Wallach, R. Fergus, S. Vishwanathan, and R. Garnett, editors, *Advances in Neural Information Processing Systems*, volume 30. Curran Associates, Inc., 2017.

Jiasheng Ye, Zaixiang Zheng, Yu Bao, Lihua Qian, and Mingxuan Wang. Dinoiser: Diffused conditional sequence learning by manipulating noises. *arXiv preprint arXiv:2302.10025*, 2023.

Hongyi Yuan, Zheng Yuan, Chuanqi Tan, Fei Huang, and Songfang Huang. Seqdiffuseq: Text diffusion with encoder-decoder transformers. *ArXiv*, abs/2212.10325, 2022.

Pengze Zhang, Hubery Yin, Chen Li, and Xiaohua Xie. Formulating discrete probability flow through optimal transport. In *Thirty-seventh Conference on Neural Information Processing Systems*, 2023.

Qinsheng Zhang and Yongxin Chen. Fast sampling of diffusion models with exponential integrator. In *The Eleventh International Conference on Learning Representations*, 2023.

Kaiwen Zheng, Cheng Lu, Jianfei Chen, and Jun Zhu. Dpm-solver-v3: Improved diffusion ode solver with empirical model statistics. In *Thirty-seventh Conference on Neural Information Processing Systems*, 2023.

A Stopping Time for Forward Process

The forward diffusion process $\mathbf{X} = \{\mathbf{x}_n, n \geq 0\}$ is a markovian stochastic process with a transition probability $p(\mathbf{x}_i|\mathbf{x}_{i-1}) = \mathcal{N}\left(\mathbf{x}_i; \sqrt{\alpha_i}\mathbf{x}_{i-1}, (1-\alpha_i)\,\mathbf{I}\right)$. And a stopping time t_0 with respect to $\mathbf{X}$ is a random time such that for each $n \geq 0$, the event $\{t_0 = n\}$ is completely determined by the total information known up to time n, $\{\mathbf{x}_0, \ldots, \mathbf{x}_n\}$. Suppose the random variables $\{\mathbf{x}_n\}$ are in a one-dimensional space and the forward process starts with $\mathbf{x}_0 = 0$. Besides, let $A, \mathbf{x}_0 \in A$ be the discrete area belonging to $\mathbf{x}_0$ that for each points in area A will be regarded as $\mathbf{x}_0$ during data generation. Our expected stopping time is defined as:

$$t_0 = \min\{n \geq 0, \mathbf{x}_n \notin A\},$$

which represents the first time $\mathbf{x}_n$ leaves area A. We can write the probability of stopping time as:

$$\begin{aligned}
&P(t_0 = 0) = P(\mathbf{x}_0 \notin A) = 0 \\
&P(t_0 = 1) = P(\mathbf{x}_0 \in A, \mathbf{x}_1 \notin A) \\
&\quad = \int_{\mathbf{x}_1 \notin A} \mathcal{N}\left(\mathbf{x}_1; \sqrt{\alpha_1}\mathbf{x}_0, (1-\alpha_1)\,\mathbf{I}\right) \mathrm{d}\mathbf{x}_1 \\
&P(t_0 = 2) = P(\mathbf{x}_0 \in A, \mathbf{x}_1 \in A, \mathbf{x}_2 \notin A) \\
&\quad = P(\mathbf{x}_0 \in A, \mathbf{x}_1 \in A) \times P(\mathbf{x}_2 \notin A | \mathbf{x}_1 \in A) \\
&\quad = \int_{\mathbf{x}_2 \notin A} \Big[\int_{\mathbf{x}_1 \in A} \mathcal{N}\left(\mathbf{x}_1; \sqrt{\alpha_1}\mathbf{x}_0, (1-\alpha_1)\,\mathbf{I}\right) \times \\
&\qquad \mathcal{N}\left(\mathbf{x}_2; \sqrt{\alpha_2}\mathbf{x}_1, (1-\alpha_2)\,\mathbf{I}\right) \mathrm{d}\mathbf{x}_1 \Big] \mathrm{d}\mathbf{x}_2 \\
&\qquad\quad \cdots\cdots \\
&P(t_0 = n) = P(\mathbf{x}_0 \in A, \ldots, \mathbf{x}_{n-1} \in A, \mathbf{x}_n \notin A) \\
&= \int_{\mathbf{x}_n \notin A} \int_{\mathbf{x}_{\leq n} \in A} \prod_{i=1}^{n-1} \mathcal{N}\left(\mathbf{x}_i; \sqrt{\alpha_i}\mathbf{x}_{i-1}, (1-\alpha_i)\,\mathbf{I}\right) \mathrm{d}\mathbf{x}_{1:n}.
\end{aligned}$$

Since the diffusion process is established in continuous space, calculating the probability of the stopping time requires integrating over each intermediate state $\mathbf{x}_{1:n-1}$, rather than a simple state transfer as in the discrete space. Hence, directly obtain the stopping time is intractable. Additionally, even if we are able to get probability of the stopping time, we can only get a distribution over the time dimension, without knowing the exact time of $\mathbf{x}_n$ leaving area A. Therefore, we need to eliminate randomness from the state transition $\mathbf{x}_{i-1} \to \mathbf{x}_i$ and find a deterministic forward trajectory to estimate the stopping time.

B Properties of Dirac Delta Function

There are several useful properties of Dirac delta function:

- **Symmetry Property:** $\delta(-x) = \delta(x)$
- **Scaling Property:** $\delta(ax) = \frac{\delta(x)}{|a|}$
- **Translation Property:** $\int f(x)\delta(x-a)\mathrm{d}x = f(a)$

C Bridging Flow Matching and DDPM

In this work, we utilizes the framework of Flow Matching to model the diffusion processes, where the forward process is defined by flow functions in eq. (5). Although having different mathematical forms, it is essentially equivalent to traditional diffusion processes. Here, we provide an alternative form from the perspective of state transfer $p_t(\mathbf{x}_t|\mathbf{x}_{t-1})$.

https://doi.org/10.52202/079017-3852

C.1 Deterministic Forward Process

Equation (5) gives the definition $p_t(\mathbf{x}_t|\mathbf{x}_0) = [\psi_t]_*\pi(\mathbf{x})$, where $\psi_t(\mathbf{x}) = \mathbf{u}_t\mathbf{x}_0 + \mathbf{v}_t\mathbf{x}$. Here we provide the equivalent derivation of $p_t(\mathbf{x}_t|\mathbf{x}_0)$ from the perspective of diffusion processes:

$$\begin{aligned} p_t(\mathbf{x}_t|\mathbf{x}_0) &= \int p_t(\mathbf{x}_{1:t}|\mathbf{x}_0)\mathrm{d}\mathbf{x}_{1:t-1} \\ &= \int p(\mathbf{x}_1|\mathbf{x}_0)\prod_{s=2}^{t} p_s(\mathbf{x}_s|\mathbf{x}_{s-1},\mathbf{x}_0)\mathrm{d}\mathbf{x}_{1:t-1}, \end{aligned} \tag{29}$$

where $p(\mathbf{x}_1|\mathbf{x}_0) = \mathcal{N}(\mathbf{u}_1\mathbf{x}_0, \mathbf{v}_1^2\mathbf{I})$ is the first step of the forward process at which the global noise is introduced into the forward trajectory. The state transfer probability of forward process $p_s(\mathbf{x}_s|\mathbf{x}_{s-1},\mathbf{x}_0) = \delta(\mathbf{x}_s - \mathbf{u}_s\mathbf{x}_0 - \mathbf{v}_s\psi_{s-1}^{-1}(\mathbf{x}_{s-1}))$ is a Dirac delta function. Therefore,

$$\begin{aligned} p_t(\mathbf{x}_t|\mathbf{x}_0) = &\int \prod_{s=3}^{t} p_s(\mathbf{x}_s|\mathbf{x}_{s-1},\mathbf{x}_0)\mathbf{d}\mathbf{x}_{2:t-1} \\ &\times \underbrace{\int p_2(\mathbf{x}_2|\mathbf{x}_1,\mathbf{x}_0)p(\mathbf{x}_1|\mathbf{x}_0)\mathrm{d}\mathbf{x}_1}_{Q_1}, \end{aligned} \tag{30}$$

where we denote the integral of $\mathbf{x}_1$ as Q_1. Based on

$$\begin{aligned} Q_0 &= q(\mathbf{x}_1|\mathbf{x}_0) = \mathcal{N}(\mathbf{u}_1\mathbf{x}_0, \mathbf{v}_1^2\mathbf{I}) \\ q_2(\mathbf{x}_2|\mathbf{x}_1,\mathbf{x}_0) &= \delta\left(\mathbf{x}_2 - \mathbf{u}_2\mathbf{x}_0 - \mathbf{v}_2\psi_1^{-1}(\mathbf{x}_1)\right) \\ &= \delta\left[\mathbf{x}_2 - \frac{\mathbf{v}_2}{\mathbf{v}_1}\mathbf{x}_1 - \left(\mathbf{u}_2 - \frac{\mathbf{v}_2\mathbf{u}_1}{\mathbf{v}_1}\right)\mathbf{x}_0\right] \\ &= \delta\left[\mathbf{x}_1 - \frac{\mathbf{v}_1}{\mathbf{v}_2}\mathbf{x}_2 - \left(\mathbf{u}_1 - \frac{\mathbf{v}_1\mathbf{u}_2}{\mathbf{v}_2}\right)\mathbf{x}_0\right] \end{aligned} \tag{31}$$

(Symmetry Property of Dirac Delta Function)

and the **Translation Property** of the Dirac delta function, we can calculate Q_1 as:

$$\begin{aligned} Q_1 &= \int \underbrace{p_2(\mathbf{x}_2|\mathbf{x}_1,\mathbf{x}_0)}_{\delta(x-a)}\underbrace{p(\mathbf{x}_1|\mathbf{x}_0)}_{f(x)}\mathrm{d}\mathbf{x}_1, \\ &\text{where } \begin{cases} x : \mathbf{x}_1 \\ a : \dfrac{\mathbf{v}_1}{\mathbf{v}_2}\mathbf{x}_2 + \left(\mathbf{u}_1 - \dfrac{\mathbf{v}_1\mathbf{u}_2}{\mathbf{v}_2}\right)\mathbf{x}_0 \end{cases} \\ &\Longrightarrow Q_1 = \mathcal{N}(\mathbf{u}_2\mathbf{x}_0, \mathbf{v}_2^2\mathbf{I}.) \end{aligned} \tag{32}$$

Then we can continue the deviation of $p_t(\mathbf{x}_t|\mathbf{x}_0)$ as:

$$\begin{aligned} p_t(\mathbf{x}_t|\mathbf{x}_0) &= \int Q_0\prod_{s=2}^{t} p_s(\mathbf{x}_s|\mathbf{x}_{s-1},\mathbf{x}_0)\mathrm{d}\mathbf{x}_{1:t-1} \\ &= \int Q_1\prod_{s=3}^{t} p_s(\mathbf{x}_s|\mathbf{x}_{s-1},\mathbf{x}_0)\mathrm{d}\mathbf{x}_{2:t-1} \\ &= \cdots\cdots \\ &= \int p_t(\mathbf{x}_t|\mathbf{x}_{t-1})Q_{t-2}\mathrm{d}\mathbf{x}_{t-1} \\ &= Q_{t-1} = \mathcal{N}(\mathbf{u}_t\mathbf{x}_0, \mathbf{v}_t^2\mathbf{I}) \end{aligned} \tag{33}$$

Therefore, the probability distribution of $\mathbf{x}_t$ conditioned on $\mathbf{x}_0$ follows a Gaussian distribution $\mathcal{N}(\mathbf{u}_t\mathbf{x}_0, \mathbf{v}_t^2\mathbf{I})$, which is the same as in original DDPMs when the coefficient functions are defined as $\mathbf{u}_t = \sqrt{\bar{\alpha}_t}$ and $\mathbf{v}_t = \sqrt{1-\bar{\alpha}_t}$. This provides an important benefit that the Flow Matching and diffusion models share the same training procedure.

C.2 Deterministic Reverse Process

The reverse tranfer probability follows Bayes' rule:

$$\begin{aligned} p(\mathbf{x}_{t-1}|\mathbf{x}_t, \mathbf{x}_0) &= p_t(\mathbf{x}_t|\mathbf{x}_{t-1}, \mathbf{x}_0)\frac{p_{t-1}(\mathbf{x}_{t-1}|\mathbf{x}_0)}{p_t(\mathbf{x}_t|\mathbf{x}_0)} \\ &= \frac{p_{t-1}(\mathbf{x}_{t-1}|\mathbf{x}_0)}{p_t(\mathbf{x}_t|\mathbf{x}_0)} \times \delta\left[\mathbf{x}_t - \frac{\mathbf{v}_t}{\mathbf{v}_{t-1}}\mathbf{x}_{t-1} - \left(\mathbf{u}_t - \frac{\mathbf{v}_t\mathbf{u}_{t-1}}{\mathbf{v}_{t-1}}\right)\mathbf{x}_0\right]. \end{aligned} \tag{34}$$

Since Dirac delta function has another form of

$$\delta(x) = \begin{cases} +\infty, x = 0 \\ 0, x \neq 0 \end{cases}, \tag{35}$$

and $p_t(\mathbf{x}_t|\mathbf{x}_0) > 0$, $p_{t-1}(\mathbf{x}_{t-1}|\mathbf{x}_t) > 0$, we have

$$\begin{aligned} p(\mathbf{x}_{t-1}|\mathbf{x}_t, \mathbf{x}_0) &= p_t(\mathbf{x}_t|\mathbf{x}_{t-1}, \mathbf{x}_0)\frac{p_{t-1}(\mathbf{x}_{t-1}|\mathbf{x}_0)}{p_t(\mathbf{x}_t|\mathbf{x}_0)} \\ &= \begin{cases} +\infty \times \overbrace{\frac{p_{t-1}(\mathbf{x}_{t-1}|\mathbf{x}_0)}{p_t(\mathbf{x}_t|\mathbf{x}_0)}}^{>0}, & \mathbf{x}_t = \left[\frac{\mathbf{v}_t}{\mathbf{v}_{t-1}}\mathbf{x}_{t-1} + \left(\mathbf{u}_t - \frac{\mathbf{v}_t\mathbf{u}_{t-1}}{\mathbf{v}_{t-1}}\right)\mathbf{x}_0\right] \\ 0, & \mathbf{x}_t \neq \left[\frac{\mathbf{v}_t}{\mathbf{v}_{t-1}}\mathbf{x}_{t-1} + \left(\mathbf{u}_t - \frac{\mathbf{v}_t\mathbf{u}_{t-1}}{\mathbf{v}_{t-1}}\right)\mathbf{x}_0\right] \end{cases} \\ &\simeq \begin{cases} +\infty, \ \mathbf{x}_{t-1} = \left[\frac{\mathbf{v}_{t-1}}{\mathbf{v}_t}\mathbf{x}_t + \left(\mathbf{u}_{t-1} - \frac{\mathbf{u}_t\mathbf{v}_{t-1}}{\mathbf{v}_t}\right)\mathbf{x}_0\right] \\ 0, \ \mathbf{x}_{t-1} \neq \left[\frac{\mathbf{v}_{t-1}}{\mathbf{v}_t}\mathbf{x}_t + \left(\mathbf{u}_{t-1} - \frac{\mathbf{u}_t\mathbf{v}_{t-1}}{\mathbf{v}_t}\right)\mathbf{x}_0\right] \end{cases} \\ &= \delta\left[\mathbf{x}_{t-1} - \frac{\mathbf{v}_{t-1}}{\mathbf{v}_t}\mathbf{x}_t - \left(\mathbf{u}_{t-1} - \frac{\mathbf{u}_t\mathbf{v}_{t-1}}{\mathbf{v}_t}\right)\mathbf{x}_0\right] \\ &= \lim_{\sigma\to 0}\mathcal{N}\left(\frac{\mathbf{v}_{t-1}}{\mathbf{v}_t}\mathbf{x}_t + \left(\mathbf{u}_{t-1} - \frac{\mathbf{u}_t\mathbf{v}_{t-1}}{\mathbf{v}_t}\right)\mathbf{x}_0, \sigma^2\mathbf{I}\right). \end{aligned} \tag{36}$$

C.3 Deterministic Optimization Objective

We first include the derivation of the variational bound for diffusion models provided by Sohl-Dickstein et al. [2015]. The probability the generative model assigns to the data is:

$$\begin{aligned} p(\mathbf{x}_0) &= \int p(\mathbf{x}_{0:T})\mathrm{d}\mathbf{x}_{1:T} \\ &= \int p(\mathbf{x}_{0:T})\frac{p_T(\mathbf{x}_{1:T}|\mathbf{x}_0)}{p_T(\mathbf{x}_{1:T}|\mathbf{x}_0)}\mathrm{d}\mathbf{x}_{1:T} \\ &= \int p_T(\mathbf{x}_{1:T}|\mathbf{x}_0)\frac{p(\mathbf{x}_{0:T})}{p_T(\mathbf{x}_{1:T}|\mathbf{x}_0)}\mathrm{d}\mathbf{x}_{1:T} \\ &= \int p_T(\mathbf{x}_{1:T}|\mathbf{x}_0)p(\mathbf{x}_T)\prod_{t=1}^{T}\frac{p(\mathbf{x}_{t-1}|\mathbf{x}_t)}{p_t(\mathbf{x}_t|\mathbf{x}_{t-1})}\mathrm{d}\mathbf{x}_{1:T}. \end{aligned} \tag{37}$$

 https://doi.org/10.52202/079017-3852

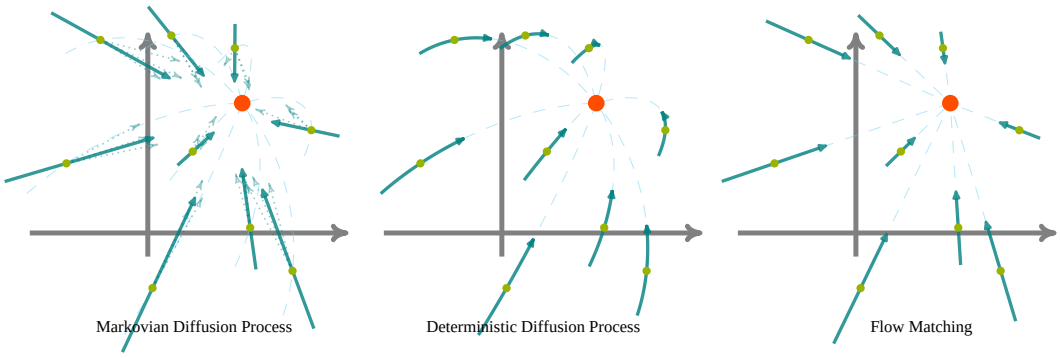


Figure 4: We demonstrate the trajectory differences among Markovian Diffusion Process, Deterministic Diffusion and Flow Matching.

Training amounts to minimizing the negative log likelihood:

$$
\begin{aligned}
\mathcal{L} &= -\int p(\mathbf{x}_0)\log p(\mathbf{x}_0)\mathrm{d}\mathbf{x}_0 \\
&= -\int p(\mathbf{x}_0)\log\left[\int p_T(\mathbf{x}_{1:T}|\mathbf{x}_0)p(\mathbf{x}_T)\prod_{t=1}^{T}\frac{p(\mathbf{x}_{t-1}|\mathbf{x}_t)}{p_t(\mathbf{x}_t|\mathbf{x}_{t-1})}\mathrm{d}\mathbf{x}_{1:T}\right]\mathrm{d}\mathbf{x}_0 \\
&\leq -\int p_T(\mathbf{x}_{0:T})\log\left[p(\mathbf{x}_T)\prod_{t=1}^{T}\frac{p(\mathbf{x}_{t-1}|\mathbf{x}_t)}{p_t(\mathbf{x}_t|\mathbf{x}_{t-1})}\right]\mathrm{d}\mathbf{x}_{0:T} \\
&= \mathbb{E}_{p_T(\mathbf{x}_{0:T})}\left[-\log p(\mathbf{x}_T)+\sum_{t=1}^{T}\log\frac{p_t(\mathbf{x}_t|\mathbf{x}_{t-1})}{p(\mathbf{x}_{t-1}|\mathbf{x}_t)}\right] \\
&= \mathbb{E}_{p_T}\left[\log\frac{p_T(\mathbf{x}_T|\mathbf{x}_0)}{p(\mathbf{x}_T)}-\log p(\mathbf{x}_0|\mathbf{x}_1)+\sum_{t=2}^{T}\log\frac{p(\mathbf{x}_{t-1}|\mathbf{x}_t,\mathbf{x}_0)}{p(\mathbf{x}_{t-1}|\mathbf{x}_t)}\right] \\
&= \mathbb{E}_{p_T}\left[\underbrace{D_{\mathrm{KL}}(p_T(\mathbf{x}_T|\mathbf{x}_0)||p(\mathbf{x}_T))}_{\mathcal{L}_T}\underbrace{-\log p(\mathbf{x}_0|\mathbf{x}_1)}_{\mathcal{L}_0}+\sum_{t=2}^{T}\underbrace{D_{\mathrm{KL}}(p(\mathbf{x}_{t-1}|\mathbf{x}_t,\mathbf{x}_0)||p(\mathbf{x}_{t-1}|\mathbf{x}_t))}_{\mathcal{L}_{t-1}}\right]
\end{aligned}
$$

where $\mathcal{L}_T$ is usually ignored as a constant and $p(\mathbf{x}_{t-1}|\mathbf{x}_t)$ is parameterized with a neural network $p_\theta(\mathbf{x}_{t-1}|\mathbf{x}_t)$ to approximate the conditioned probability distributions in the reverse process. Since $p(\mathbf{x}_{t-1}|\mathbf{x}_t,\mathbf{x}_0) = \lim_{\sigma\to 0}\mathcal{N}\left(\frac{\mathbf{v}_{t-1}}{\mathbf{v}_t}\mathbf{x}_t+\left(\mathbf{u}_{t-1}-\frac{\mathbf{u}_t\mathbf{v}_{t-1}}{\mathbf{v}_t}\mathbf{x}_0\right),\sigma^2\mathbf{I}\right)$, the parameterized $p_\theta(\mathbf{x}_{t-1}|\mathbf{x}_t)$ can take the same form $\mathcal{N}(\boldsymbol{\mu}_\theta(\mathbf{x}_t,t),\sigma_t^2\mathbf{I})$ because the Dirac delta function is a special case of Gaussian distribution and the KL divergence of two Gaussians can be simplified. Finally, the training objective for the deterministic diffusion process is divided as:

$$
\mathcal{L} = \begin{cases}
\mathcal{L}_T : \text{a constant} \\
\mathcal{L}_0 : -\log\delta\left(\mathbf{x}_0-\mathbf{x}_\theta(\mathbf{x}_1,1)\right) \\
\mathcal{L}_{t-1} : c\|\mathbf{x}_0-\mathbf{x}_\theta(\mathbf{x}_t,t)\|^2+\lim_{\sigma\to 0}\log\frac{\sigma_t}{\sigma} \\
\quad c=\frac{1}{2\sigma_t^2}\left(\mathbf{u}_{t-1}-\frac{\mathbf{u}_t\mathbf{v}_{t-1}}{\mathbf{v}_{t-1}}\right)^2,
\end{cases} \tag{38}
$$

where the simplified version $\|\mathbf{x}_0-\mathbf{x}_\theta(\mathbf{x}_t,t)\|^2$ is the same as DDPMs but with different coefficients.

D Different Diffusion Trajectories

We illustrate the trajectories of different diffusion processes in Figure 4. The forward and reverse generation for the Markovian diffusion process is:

$$\begin{cases} \mathbf{x}_t = \sqrt{\bar{\alpha}_t}\mathbf{x}_0 + \sqrt{1-\bar{\alpha}_t}\boldsymbol{\epsilon}_t \\ \mathbf{x}_{t-1} = \dfrac{\sqrt{\bar{\alpha}_{t-1}}(1-\alpha_t)}{1-\bar{\alpha}_t}\mathbf{x}_0 + \dfrac{\sqrt{\alpha_t}(1-\bar{\alpha}_{t-1})}{1-\bar{\alpha}_t}\mathbf{x}_t \\ \qquad\qquad + \dfrac{\sqrt{(1-\bar{\alpha}_{t-1})(1-\alpha_t)}}{\sqrt{1-\bar{\alpha}_t}}\boldsymbol{\epsilon}_{t-1}. \end{cases} \tag{39}$$

The deterministic diffusion process:

$$\begin{cases} \mathbf{x}_t = \sqrt{\bar{\alpha}_t}\mathbf{x}_0 + \sqrt{1-\bar{\alpha}_t}\boldsymbol{\epsilon} \\ \mathbf{x}_{t-1} = \left(\sqrt{\bar{\alpha}_{t-1}} - \dfrac{\sqrt{\bar{\alpha}_t(1-\bar{\alpha}_{t-1})}}{\sqrt{1-\bar{\alpha}_t}}\right)\mathbf{x}_0 \\ \qquad\qquad + \dfrac{\sqrt{1-\bar{\alpha}_{t-1}}}{\sqrt{1-\bar{\alpha}_t}}\mathbf{x}_t. \end{cases} \tag{40}$$

The deterministic flow matching with optimal transport:

$$\begin{cases} \mathbf{x}_t = (1-\dfrac{t}{T})\mathbf{x}_0 + \dfrac{t}{T}\boldsymbol{\epsilon} \\ \mathbf{x}_{t-1} = \dfrac{1}{t}\mathbf{x}_0 + \dfrac{t-1}{t}\mathbf{x}_t. \end{cases} \tag{41}$$

E Details of the Function G

Equation (13) defines the function $G(\mathbf{x}, \boldsymbol{\epsilon})$ as the inversion of coefficient function.

Flow Matching The coefficient is $\mathbf{u}_t = 1 - t/T$, where $t = T \times (1 - \mathbf{u}_t)$. Therefore,

$$G(\mathbf{x}_0, \boldsymbol{\epsilon}) = t_0 = T \times (1 - \mathbf{u}_{t_0}) = T \Big/ \left(1 + \frac{f(\boldsymbol{\epsilon}, \mathcal{J}) - f(\boldsymbol{\epsilon}, \mathcal{I})}{f(\mathbf{x}_0, \mathcal{I}) - f(\mathbf{x}_0, \mathcal{J})}\right) \tag{42}$$

Diffusion The coefficient for Variance Exploding is $\mathbf{v}_T = \sigma_0 \left(\frac{\sigma_T}{\sigma_0}\right)^{\frac{t}{T}}$, where $t = T \times \frac{\log \mathbf{v}_t - \log \sigma_0}{\log \sigma_T - \log \sigma_0}$.

$$G(\mathbf{x}_0, \boldsymbol{\epsilon}) = t_0 = \mathbf{v}_{t_0} = T \times \frac{\log \mathbf{v}_t - \log \sigma_0}{\log \sigma_T - \log \sigma_0} = T \times \frac{\log \frac{f(\boldsymbol{\epsilon}, \mathcal{J}) - f(\boldsymbol{\epsilon}, \mathcal{I})}{f(\mathbf{x}_0, \mathcal{I}) - f(\mathbf{x}_0, \mathcal{J})} - \log \sigma_0}{\log \sigma_T - \log \sigma_0}. \tag{43}$$

For Variance Preserving, the function $G(\mathbf{x}_0, \boldsymbol{\epsilon})$ is more difficult to calculate since $\mathbf{u}_t = \sqrt{\bar{\alpha}_t}$, where $\bar{\alpha} = \prod_{i=1}^{t} \alpha_i$, $\alpha_t = 1 - \beta_t$, and β_t is also influenced by noise schedulers. This makes $G(\mathbf{x}_0, \boldsymbol{\epsilon})$ hard to calculate. Fortunately, we can bypass this function and provide the corresponding pseudo code.

F Details of the Training Objective

The rescaled vector field is calculated as:

$$\begin{aligned} \tilde{u}_t &= \frac{\mathrm{d}\tilde{\mathbf{x}}_t}{\mathrm{d}t} = \frac{\mathrm{d}\tilde{\mathbf{x}}_t}{\mathrm{d}\tau}\frac{\mathrm{d}\tau}{\mathrm{d}t} \\ &= \left[\mathbf{u}'\left(\mathbf{x}_0, \tau\right)\mathbf{x}_0 + \mathbf{v}'(\mathbf{x}_0, \tau)\boldsymbol{\epsilon}\right]\frac{T - r \times G(\mathbf{x}_0, \boldsymbol{\epsilon})}{T} \\ &= u_\tau \times \frac{T - r \times G(\mathbf{x}_0, \boldsymbol{\epsilon})}{T}. \end{aligned} \tag{44}$$

https://doi.org/10.52202/079017-3852

Considering the expectation form of $\tilde{u}_t$, there is:

$$\begin{aligned}
\mathbb{E}_{\tilde{\mathbf{x}}_t}\left[\tilde{u}_t(\tilde{\mathbf{x}}_t|\mathbf{x}_0)\right] &= \sum p(\tilde{\mathbf{x}}_t|\mathbf{x}_0)\tilde{u}_t(\tilde{\mathbf{x}}_t|\mathbf{x}_0) \\
&= \sum p(\tilde{\mathbf{x}}_t|\mathbf{x}_0)\left[\mathbf{u}'\left(\mathbf{x}_0,\tau\right)\mathbf{x}_0 + \mathbf{v}'(\mathbf{x}_0,\tau)\boldsymbol{\epsilon}\right]\underbrace{\frac{T - r\times G(\mathbf{x}_0,\boldsymbol{\epsilon})}{T}}_{0\leq\text{coefficient}\leq 1} \\
&\leq \sum p(\tilde{\mathbf{x}}_t|\mathbf{x}_0)\left[\mathbf{u}'\left(\mathbf{x}_0,\tau\right)\mathbf{x}_0 + \mathbf{v}'(\mathbf{x}_0,\tau)\boldsymbol{\epsilon}\right] \\
&= \mathbf{u}'\left(\mathbf{x}_0,\tau\right)\left[\sum p(\tilde{\mathbf{x}}_t|\mathbf{x}_0)\mathbf{x}_0\right] + \mathbf{v}'(\mathbf{x}_0,\tau)\boldsymbol{\epsilon} \\
&= \tilde{u}_t(\tilde{\mathbf{x}}_0|\mathbb{E}_{\tilde{\mathbf{x}}_t}[\mathbf{x}_0]).
\end{aligned} \tag{45}$$

Therefore, the training objective $\mathbb{E}\|\tilde{u}_t - \tilde{u}_\theta\|^2 \leq c\,\mathbb{E}\|\mathbf{x}_0 - \mathbf{x}_\theta\|^2$, where c is the coefficient.

G Code Implementations

Our framework is a module constructed on current diffusion models. We demonstrate our kernel part *rescale diffusion trajectory* with pseudo python code as below:

```
def rescale_diffusion_trajectory(x_0, epsilon, embedding,
        labels, alphas_cumprod, timesteps, mode):
    #embedding: embedding matrix, f(x,i)=(embedding * x)[i]
    #labels: I
    #alphas_cumprod: list of all u_t
    #timesteps: t
    #mode: noising or denoising

    #1. get f(x,i):
    self_dot = torch.sum(embedding * embedding, dim=-1)
    f_x_i = self_dot[labels][..., None]
    labels = labels[..., None]

    #2. get f(x,j) and f(eps,j):
    embedding = embedding.permute(1, 0)
    f_x_j = torch.matmul(x_0, embedding)
    f_eps_j = torch.matmul(epsilon, embedding)

    #3. get f(x,i) - f(x,j): (usually >=0; smaller -> closer)
    #filter out f(x,i)-f(x,i) with a large positive number 100
    fxi_minus_fxj = (f_x_i - f_x_j).scatter(-1, labels, 100)

    #4. get f(eps,i) and f(eps,j) - f(eps,i): (larger -> more noise)
    f_eps_i = torch.gather(f_eps_j, -1, labels)
    #filter out f(eps,i)-f(eps,i) with a large negative number -100
    fepsj_minus_fepsi = (f_eps_j - f_eps_i).scatter(-1, labels, -100)

    #5. get fraction and u_t_0
    #mask results outside the support set
    info_mask = (fepsj_minus_fepsi < 0) | (fxi_minus_fxj < 0)
    fraction = fix_minus_fjx / fjeps_minus_fieps
    fraction[info_mask] = 100
    min_frac, _ = fraction.min(dim=-1) # minimum
    #Diffusion Variance Preserving eq. (9)
    u_t_0 = torch.sqrt(1 / (1 + min_frac ** 2))[..., None]

    #6. rescale timesteps
    sqrt_alphas_cumprod = torch.sqrt(alphas_cumprod)
```

```
    ###!!!important trick!!!###
    #We do not need to calculate the function G(x_0,t) (eq. (12)).
    #Timesteps of diffusion processes are discrete and
    #  we just iterate over and compare with all coefficient functions.
    #Besides, function G is easy to calculate for Flow Matching.
    index = torch.sum(u_t_0 < sqrt_alphas_cumprod, dim=-1)

    #T is the maximum timestep, for example T=2000.
    #confactor is the confidency factor
    #tau is the rescaled timestep
    #delta_tau is the rescaled decoding velocity
    if mode == 'noising':
        tau = (timesteps + index - \
            (((timesteps + 1) / T) * index)).long().clamp(0, T)
        tau = (confactor * tau.float() + \
            (1.0 - confactor) * timesteps.float()).long().clamp(0, T)
        return tau
    elif mode == 'denoising':
        delta_tau = (T - index) / T
        delta_tau = (confactor * delta_tau + \
            (1 - confactor) * 1.0).clamp(0, 1)
        return delta_tau
```


https://doi.org/10.52202/079017-3852

Table 6: FID of difference sampling strategies.

	Gaussian	Deterministic
BINARY CODING	13.39	**3.86**
FIXED EMBEDDING	12.21	**9.15**
TRAINABLE EMBEDDING	22.24	**10.99**

H Analysis

Gaussian Sampling Our framework is compatible with the Gaussian sampling in DDPM, where random noises can be added into each iteration step. Algorithm 3 demonstrates the Gaussian sampling procedure. Compared with algorithm 2, a Gaussian noise $\mathbf{z} \sim \mathcal{N}(\mathbf{0}, \sigma_t^2\mathbf{I})$ with a decreasing variance σ_t is injected to the estimated next state $\tilde{\mathbf{x}}_t$. This noise $\mathbf{z}$ will be mapped as changing the initial sampling $\tilde{\mathbf{x}}_T$ through the trajectory alteration step. We illustrate the deterministic and Gaussian sampling for our model on CIFAR-10 in Table 6, where the deterministic sampling can achieve a much better performance of FID. We assume this is because our coefficient functions $\mathbf{u}(\mathbf{x}_0, t)$ and $\mathbf{v}(\mathbf{x}_0, t)$ are dynamically calculated to rescale the deterministic trajectory in the training stage. In the inference stage, $\mathbf{x}_0$ is replaced by $\mathbf{x}_\theta(\mathbf{x}_t, t)$, where errors will accumulate if the predicted pseudo target changes frequently. Moreover, Gaussian sampling will further introduce random noises at each reverse step, making our rescaled timestep τ far away from the training situation. Therefore, errors in the calculations of trajectory scaling will explode over iterations.

Algorithm 3 Gaussian Sampling

$t \coloneqq T, \tau \coloneqq T$
$\tilde{\mathbf{x}}_t \sim \mathcal{N}(\mathbf{0}, \mathbf{I})$ // Initialing
for $\Delta t \coloneqq \Delta t_1, \ldots, \Delta t_s$ **do** // $\sum_{\Delta t} = T$
  $\mathbf{z} \sim \mathcal{N}(\mathbf{0}, \sigma_t^2\mathbf{I})$ // Gaussian Noise
  $\hat{\mathbf{x}}_0 \coloneqq \mathbf{x}_\theta(\tilde{\mathbf{x}}_t, t)$ // Pseudo Target
  $\hat{\epsilon} \coloneqq \Psi^{-1}([\tilde{\mathbf{x}}_t; \tau])$ // Trajectory Alteration
  $\tau_\Delta \coloneqq \mathcal{T}\left(t - \Delta t, G(\hat{\mathbf{x}}_0, \hat{\boldsymbol{\epsilon}})\right)$ // eq. (25)
  $\tilde{\mathbf{x}}_t \coloneqq \mathbf{u}(\hat{\mathbf{x}}_0, \tau_\Delta)\hat{\mathbf{x}}_0 + \mathbf{v}(\hat{\mathbf{x}}_0, \tau_\Delta)\hat{\boldsymbol{\epsilon}} + \mathbf{z}$
  $t \coloneqq t - \Delta t, \tau \coloneqq \tau_\Delta$ // Updating
end for
$\mathbf{x}_0 \coloneqq \mathbf{x}_\theta(\tilde{\mathbf{x}}_t, t)$ // $\mathbf{x}_1 \to \mathbf{x}_0$
return $\mathbf{x}_0$

I Limitations

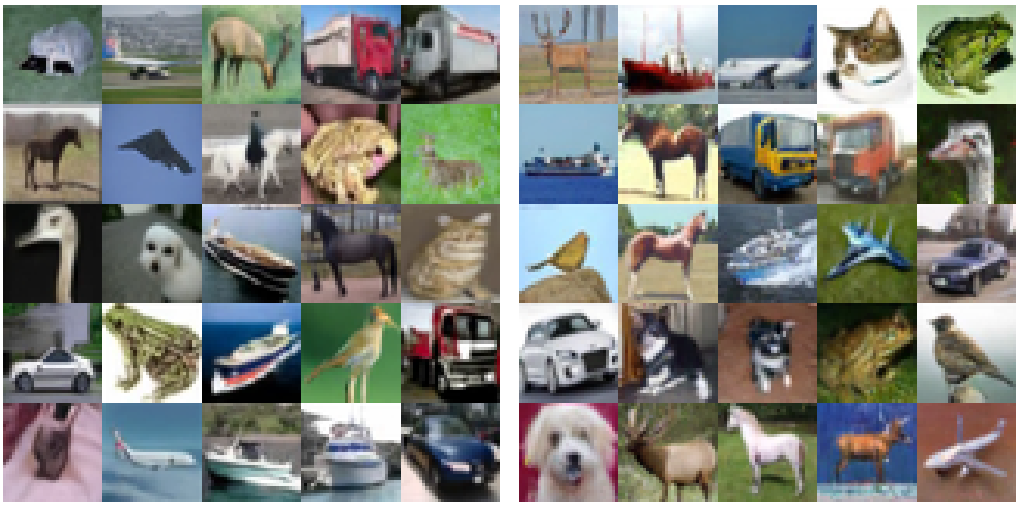
Our framework is proposed to migrate the powerful continuous diffusion models to discrete problems. There is another technical route that directly designs the diffusion process on the discrete state space and our method is not useful for this scenario. However, we believe the continuous diffusion models can be a general framework for generative modeling and our effort can advance this target.

We prefer $\mathbf{x}_0$ as the training target because we highly depend on the reliability of the predicted $\hat{\mathbf{x}}_0$ during inference. Although it is possible to use other targets, the modeling effect will decrease in practical use, which limits the flexibility of diffusion modeling. For example, predicting the $\hat{\boldsymbol{\epsilon}}$ and recovering $\hat{\mathbf{x}}_0$ with eq. (23) is inefficient, because a small error in predicting $\hat{\boldsymbol{\epsilon}}$ will be amplified by eq. (23) and lead to the collapse of $G(\hat{\mathbf{x}}_0, \hat{\boldsymbol{\epsilon}})$.

Our approach requires extra computational cost. But they are acceptable since our rescaling process is a series of parallel matrix computations. Considering that our approach is compatible with the Self-Conditioning [Chen et al., 2023b], our overhead is negligible when it is used.

J Other Experimental Details

For language modeling, we utilize the model configuration *transformer-iwslt-de-en* in FAIRSEQ framework [Ott et al., 2019] for IWSLT14 DE-EN, which has 6 transformer layers, 4 attention heads, 512 hidden dimensions, and 1024 feed forward layer dimensions. For other datasets, the configuration is *transformer-base*, which has 6 transformer layers, 8 attention heads, 512 hidden dimensions, and 2048 feed forward layer dimensions. The embedding dimension is 128. The beam size is 1 length prediction beam $\times$ 5 generation beam, since the length prediction is unstable for diffusion language models. For reranking, we take 7 length prediction beam $\times$ 3 generation beam as Differmer to let the transformer choose the best one.



(A) Bit Diffusion *repro* (FID 10.37) (B) Ours (FID 3.86)

Figure 5: Generated BINARY CODING images of reproduced Bit Diffusion and Ours on CIFAR-10.

For image generation, we set the scaling factor $r = 0.5$ for training. Besides, we find that a smaller factor for inference is sometime useful. We set $r = 0.45$ on binary coding and $r = 0.2$ on fixed embedding during inference. When the pixel embedding is learnable, the scaling factor is $r = 0.5$, which is the same as training.

Our experiments are performed with Nvidia 80G A100. Each language result requires about 2 days on one single A100. Each image result requires about a week on one single A100.

K Impact Statements

This paper presents work whose goal is to advance the field of Deep Learning. The datasets we used has been widely deployed for many years and has basically no negative impact. Our approach is a framework that migrates existing diffusion models to discrete problems, which does not provide a large pre-trained model that can be used to generate fake contents.

L Case Study

Generated sentences on IWSLT14 DE-EN and GIGAWORD are illustrated in Table 7 and Table 8. Generated images on CIFAR-10 are depicted in Figure 5, 6, and 7.

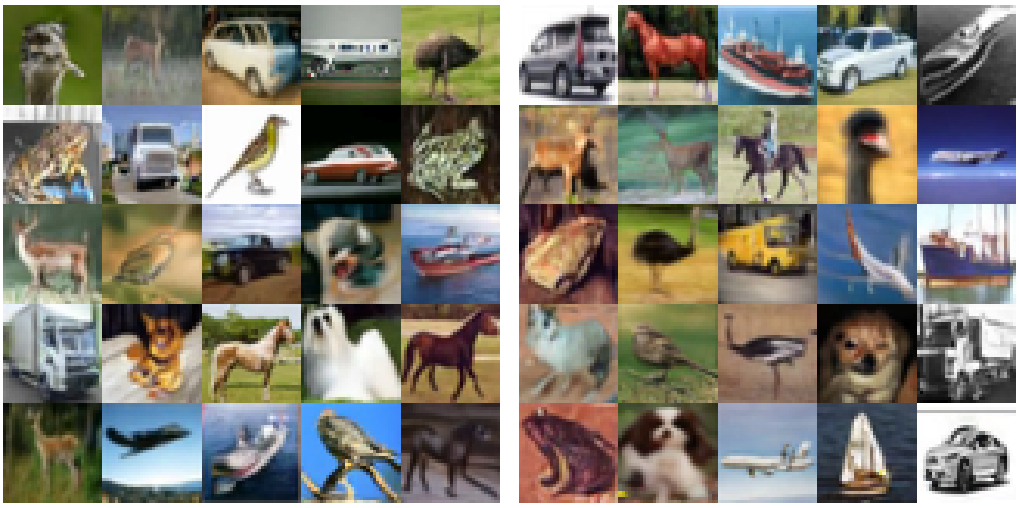
https://doi.org/10.52202/079017-3852

Table 7: Cases of translation on IWSLT14 DE-EN.

Source: GERMAN	Target: ENGLISH		
	Differmer	Ours	Golden
ich möchte ihnen erzählen , wie wir das herausgefunden haben .	i want to tell you about this .	i want to tell you how we ' ve figured that out .	i want to tell you how we found that out .
da gingen ganz schön viele verrückte dinge vor sich .	lots of crazy things .	there were quite a lot of crazy things going on .	there was a whole lot of crazy going on in there .
man macht etwas , das eigentlich ein wenig anders ist .	you do something a little different .	you ' re doing something that ' s actually a little bit different .	you do something that ' s actually a little different .
und die welt in der wir lebten sah so aus .	and the world we lived like this .	and the world we lived in looked like this .	and the world we used to live in looked like this .
man erwartet eine zusätzliche milliarde spieler im nächsten jahrzehnt .	you ' ll expect an next billion players .	you expect an extra billion players in the next decade .	they expect one billion more gamers in the next decade .
b hat diese vorteile und risiken . was wollen sie tun ?	b has risks . what do you want to do ?	b has these benefits and risks . what do you want to do ?	b has these benefits , and these risks . what do you want to do ?
wir haben also so eine situation , wo , je weiter unsere wissenschaft fortschreitet , wir uns um so mehr eingestehen müssen , dass diese kategorien , die wir für stabile anatomische kategorien gehalten hatten , welche sehr einfache zuordnungen herstellten um dauerhafte identitätskategorien zu schaffen , viel unschärfer sind , als wir angenommen haben .	so we have this situation where the continuing our science continues , we need to admit the more that these categories that we thought were stable anatomical categories , which made a very simple collaborations to create permanent identity ories are much unsharers than we ' ve assumed .	so we have a situation where , as the further our science goes on , we have to admit in terms , the more that these categories that we thought of be a stable anatomical categories , which made a very simple assaments to create permanent identity categories , are much more blanky than we ' ve accepted .	so what we have is a sort of situation where the farther our science goes , the more we have to admit to ourselves that these categories that we thought of as stable anatomical categories that mapped very simply to stable identity categories are a lot more fuzzy than we thought .

Table 8: Cases of summarization on GIGAWORD.

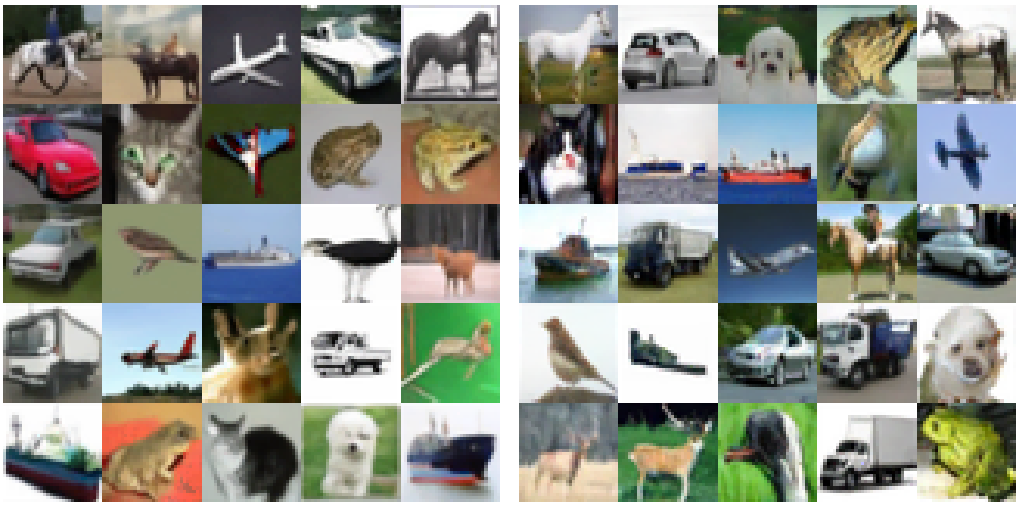
Source	Target		
	Differmer	Ours	Golden
the asian swimming record tumbled again at the seven-day olympic test event here on friday .	asian swimming record falls again	asian swimming tumble again at olympic test event	asian swimming record tumbles again at china 's olympic trials
a truck carrying illegal north african immigrants flipped over in northeastern spain , killing ## and injuring six others , police said monday .	truck carrying illegal immigrants crashes in spain killing ##	## illegal immigrants killed in truck accident in northeastern spain	## immigrants killed in road accident in spain
new zealand share prices closed #.## percent lower wednesday after investors took their lead from further weakness in overseas markets , dealers said .	new zealand shares fall #.## percent	new zealand shares close #.## percent lower	new zealand shares close down #.## percent
the sudanese opposition said here thursday it had killed more than ### government soldiers in an ambush in the east of the country .	sudanese opposition claims over ### soldiers killed	sudanese opposition claims ### soldiers killed in ambush	sudanese opposition says ### government troops killed in ambush
these sports stories for release tuesday , september ## , #### , are moving today to clients of the new york times news service .	thursday 's sports budget	cox news service sports budget	cox news service tuesday sports budget
bangladesh and india signed a deal here thursday giving green signal to resumption of passenger train service between the two neighboring countries after ## years .	bangladesh india sign agreement on train service	bangladesh india sign agreement to resume train service	bangladesh india sign agreement for resumption of train service after ## years



(A) Bit Diffusion *repro* (FID 12.96) (B) Ours (FID 9.15)

Figure 6: Generated FIXED EMBEDDING images of reproduced Bit Diffusion and Ours on CIFAR-10.

https://doi.org/10.52202/079017-3852



(A) Bit Diffusion *repro* (FID 19.26) (B) Ours (FID 10.99)

Figure 7: Generated TRAINABLE EMBEDDING images of reproduced Bit Diffusion and Ours on CIFAR-10.

NeurIPS Paper Checklist

1. **Claims**

   Question: Do the main claims made in the abstract and introduction accurately reflect the paper's contributions and scope?

   Answer: [Yes]

   Justification: Contributions and scope are in abstract and introduction.

   Guidelines:

   - The answer NA means that the abstract and introduction do not include the claims made in the paper.
   - The abstract and/or introduction should clearly state the claims made, including the contributions made in the paper and important assumptions and limitations. A No or NA answer to this question will not be perceived well by the reviewers.
   - The claims made should match theoretical and experimental results, and reflect how much the results can be expected to generalize to other settings.
   - It is fine to include aspirational goals as motivation as long as it is clear that these goals are not attained by the paper.

2. **Limitations**

   Question: Does the paper discuss the limitations of the work performed by the authors?

   Answer: [Yes]

   Justification: Discussed in (7) Conclusion and (H) Limitations sections.

   Guidelines:

   - The answer NA means that the paper has no limitation while the answer No means that the paper has limitations, but those are not discussed in the paper.
   - The authors are encouraged to create a separate "Limitations" section in their paper.
   - The paper should point out any strong assumptions and how robust the results are to violations of these assumptions (e.g., independence assumptions, noiseless settings, model well-specification, asymptotic approximations only holding locally). The authors should reflect on how these assumptions might be violated in practice and what the implications would be.
   - The authors should reflect on the scope of the claims made, e.g., if the approach was only tested on a few datasets or with a few runs. In general, empirical results often depend on implicit assumptions, which should be articulated.
   - The authors should reflect on the factors that influence the performance of the approach. For example, a facial recognition algorithm may perform poorly when image resolution is low or images are taken in low lighting. Or a speech-to-text system might not be used reliably to provide closed captions for online lectures because it fails to handle technical jargon.
   - The authors should discuss the computational efficiency of the proposed algorithms and how they scale with dataset size.
   - If applicable, the authors should discuss possible limitations of their approach to address problems of privacy and fairness.
   - While the authors might fear that complete honesty about limitations might be used by reviewers as grounds for rejection, a worse outcome might be that reviewers discover limitations that aren't acknowledged in the paper. The authors should use their best judgment and recognize that individual actions in favor of transparency play an important role in developing norms that preserve the integrity of the community. Reviewers will be specifically instructed to not penalize honesty concerning limitations.

3. **Theory Assumptions and Proofs**

   Question: For each theoretical result, does the paper provide the full set of assumptions and a complete (and correct) proof?

   Answer: [Yes]

https://doi.org/10.52202/079017-3852

Justification: From sections (A) to (E) in appendices.

Guidelines:

- The answer NA means that the paper does not include theoretical results.
- All the theorems, formulas, and proofs in the paper should be numbered and cross-referenced.
- All assumptions should be clearly stated or referenced in the statement of any theorems.
- The proofs can either appear in the main paper or the supplemental material, but if they appear in the supplemental material, the authors are encouraged to provide a short proof sketch to provide intuition.
- Inversely, any informal proof provided in the core of the paper should be complemented by formal proofs provided in appendix or supplemental material.
- Theorems and Lemmas that the proof relies upon should be properly referenced.

4. **Experimental Result Reproducibility**

   Question: Does the paper fully disclose all the information needed to reproduce the main experimental results of the paper to the extent that it affects the main claims and/or conclusions of the paper (regardless of whether the code and data are provided or not)?

   Answer: [Yes]

   Justification: We use algorithms 1 and 2 to demonstrate how to reproduce our algorithm. We provide a paragraph of Experimental Setup in (4) Language Modeling and (5) Discrete Image Generation sections. Other details are in section (I) and pseudo code of our kernel process is in (F).

   Guidelines:

   - The answer NA means that the paper does not include experiments.
   - If the paper includes experiments, a No answer to this question will not be perceived well by the reviewers: Making the paper reproducible is important, regardless of whether the code and data are provided or not.
   - If the contribution is a dataset and/or model, the authors should describe the steps taken to make their results reproducible or verifiable.
   - Depending on the contribution, reproducibility can be accomplished in various ways. For example, if the contribution is a novel architecture, describing the architecture fully might suffice, or if the contribution is a specific model and empirical evaluation, it may be necessary to either make it possible for others to replicate the model with the same dataset, or provide access to the model. In general. releasing code and data is often one good way to accomplish this, but reproducibility can also be provided via detailed instructions for how to replicate the results, access to a hosted model (e.g., in the case of a large language model), releasing of a model checkpoint, or other means that are appropriate to the research performed.
   - While NeurIPS does not require releasing code, the conference does require all submissions to provide some reasonable avenue for reproducibility, which may depend on the nature of the contribution. For example
     (a) If the contribution is primarily a new algorithm, the paper should make it clear how to reproduce that algorithm.
     (b) If the contribution is primarily a new model architecture, the paper should describe the architecture clearly and fully.
     (c) If the contribution is a new model (e.g., a large language model), then there should either be a way to access this model for reproducing the results or a way to reproduce the model (e.g., with an open-source dataset or instructions for how to construct the dataset).
     (d) We recognize that reproducibility may be tricky in some cases, in which case authors are welcome to describe the particular way they provide for reproducibility. In the case of closed-source models, it may be that access to the model is limited in some way (e.g., to registered users), but it should be possible for other researchers to have some path to reproducing or verifying the results.

5. **Open access to data and code**

Question: Does the paper provide open access to the data and code, with sufficient instructions to faithfully reproduce the main experimental results, as described in supplemental material?

Answer: [Yes]

Justification: The pseudo code of our kernel process is demonstrated in (F) and we will public our code on github.com.

Guidelines:

- The answer NA means that paper does not include experiments requiring code.
- Please see the NeurIPS code and data submission guidelines (`https://nips.cc/public/guides/CodeSubmissionPolicy`) for more details.
- While we encourage the release of code and data, we understand that this might not be possible, so "No" is an acceptable answer. Papers cannot be rejected simply for not including code, unless this is central to the contribution (e.g., for a new open-source benchmark).
- The instructions should contain the exact command and environment needed to run to reproduce the results. See the NeurIPS code and data submission guidelines (`https://nips.cc/public/guides/CodeSubmissionPolicy`) for more details.
- The authors should provide instructions on data access and preparation, including how to access the raw data, preprocessed data, intermediate data, and generated data, etc.
- The authors should provide scripts to reproduce all experimental results for the new proposed method and baselines. If only a subset of experiments are reproducible, they should state which ones are omitted from the script and why.
- At submission time, to preserve anonymity, the authors should release anonymized versions (if applicable).
- Providing as much information as possible in supplemental material (appended to the paper) is recommended, but including URLs to data and code is permitted.

6. **Experimental Setting/Details**

   Question: Does the paper specify all the training and test details (e.g., data splits, hyperparameters, how they were chosen, type of optimizer, etc.) necessary to understand the results?

   Answer: [Yes]

   Justification: We provide a paragraph of Experimental Setup in (4) Language Modeling and (5) Discrete Image Generation sections. Other details are in section (I). We provide ablation studies on the hyperparameters.

   Guidelines:

   - The answer NA means that the paper does not include experiments.
   - The experimental setting should be presented in the core of the paper to a level of detail that is necessary to appreciate the results and make sense of them.
   - The full details can be provided either with the code, in appendix, or as supplemental material.

7. **Experiment Statistical Significance**

   Question: Does the paper report error bars suitably and correctly defined or other appropriate information about the statistical significance of the experiments?

   Answer: [No]

   Justification: Error bars are not reported because it would be too computationally expensive. Each result in the experiment table needs to be run on an 80G A100 for at least 2 days. The huge overhead required to obtain a statistically significant error bar makes it impossible for us to achieve it.

   Guidelines:

   - The answer NA means that the paper does not include experiments.
   - The authors should answer "Yes" if the results are accompanied by error bars, confidence intervals, or statistical significance tests, at least for the experiments that support the main claims of the paper.

https://doi.org/10.52202/079017-3852

- The factors of variability that the error bars are capturing should be clearly stated (for example, train/test split, initialization, random drawing of some parameter, or overall run with given experimental conditions).
- The method for calculating the error bars should be explained (closed form formula, call to a library function, bootstrap, etc.)
- The assumptions made should be given (e.g., Normally distributed errors).
- It should be clear whether the error bar is the standard deviation or the standard error of the mean.
- It is OK to report 1-sigma error bars, but one should state it. The authors should preferably report a 2-sigma error bar than state that they have a 96% CI, if the hypothesis of Normality of errors is not verified.
- For asymmetric distributions, the authors should be careful not to show in tables or figures symmetric error bars that would yield results that are out of range (e.g. negative error rates).
- If error bars are reported in tables or plots, The authors should explain in the text how they were calculated and reference the corresponding figures or tables in the text.

8. **Experiments Compute Resources**

   Question: For each experiment, does the paper provide sufficient information on the computer resources (type of compute workers, memory, time of execution) needed to reproduce the experiments?

   Answer: [Yes]

   Justification: Details are in section (I).

   Guidelines:

   - The answer NA means that the paper does not include experiments.
   - The paper should indicate the type of compute workers CPU or GPU, internal cluster, or cloud provider, including relevant memory and storage.
   - The paper should provide the amount of compute required for each of the individual experimental runs as well as estimate the total compute.
   - The paper should disclose whether the full research project required more compute than the experiments reported in the paper (e.g., preliminary or failed experiments that didn't make it into the paper).

9. **Code Of Ethics**

   Question: Does the research conducted in the paper conform, in every respect, with the NeurIPS Code of Ethics `https://neurips.cc/public/EthicsGuidelines`?

   Answer: [Yes]

   Justification:

   Guidelines:

   - The answer NA means that the authors have not reviewed the NeurIPS Code of Ethics.
   - If the authors answer No, they should explain the special circumstances that require a deviation from the Code of Ethics.
   - The authors should make sure to preserve anonymity (e.g., if there is a special consideration due to laws or regulations in their jurisdiction).

10. **Broader Impacts**

    Question: Does the paper discuss both potential positive societal impacts and negative societal impacts of the work performed?

    Answer: [Yes]

    Justification: Discussed in section (J).

    Guidelines:

    - The answer NA means that there is no societal impact of the work performed.
    - If the authors answer NA or No, they should explain why their work has no societal impact or why the paper does not address societal impact.

- Examples of negative societal impacts include potential malicious or unintended uses (e.g., disinformation, generating fake profiles, surveillance), fairness considerations (e.g., deployment of technologies that could make decisions that unfairly impact specific groups), privacy considerations, and security considerations.
- The conference expects that many papers will be foundational research and not tied to particular applications, let alone deployments. However, if there is a direct path to any negative applications, the authors should point it out. For example, it is legitimate to point out that an improvement in the quality of generative models could be used to generate deepfakes for disinformation. On the other hand, it is not needed to point out that a generic algorithm for optimizing neural networks could enable people to train models that generate Deepfakes faster.
- The authors should consider possible harms that could arise when the technology is being used as intended and functioning correctly, harms that could arise when the technology is being used as intended but gives incorrect results, and harms following from (intentional or unintentional) misuse of the technology.
- If there are negative societal impacts, the authors could also discuss possible mitigation strategies (e.g., gated release of models, providing defenses in addition to attacks, mechanisms for monitoring misuse, mechanisms to monitor how a system learns from feedback over time, improving the efficiency and accessibility of ML).

11. **Safeguards**

    Question: Does the paper describe safeguards that have been put in place for responsible release of data or models that have a high risk for misuse (e.g., pretrained language models, image generators, or scraped datasets)?

    Answer: [NA]

    Justification: Our approach is a framework involves algorithms but not pre-trained models.

    Guidelines:

    - The answer NA means that the paper poses no such risks.
    - Released models that have a high risk for misuse or dual-use should be released with necessary safeguards to allow for controlled use of the model, for example by requiring that users adhere to usage guidelines or restrictions to access the model or implementing safety filters.
    - Datasets that have been scraped from the Internet could pose safety risks. The authors should describe how they avoided releasing unsafe images.
    - We recognize that providing effective safeguards is challenging, and many papers do not require this, but we encourage authors to take this into account and make a best faith effort.

12. **Licenses for existing assets**

    Question: Are the creators or original owners of assets (e.g., code, data, models), used in the paper, properly credited and are the license and terms of use explicitly mentioned and properly respected?

    Answer: [Yes]

    Justification: We provide the link or citation of each asset, where licenses are in the link.

    Guidelines:

    - The answer NA means that the paper does not use existing assets.
    - The authors should cite the original paper that produced the code package or dataset.
    - The authors should state which version of the asset is used and, if possible, include a URL.
    - The name of the license (e.g., CC-BY 4.0) should be included for each asset.
    - For scraped data from a particular source (e.g., website), the copyright and terms of service of that source should be provided.
    - If assets are released, the license, copyright information, and terms of use in the package should be provided. For popular datasets, `paperswithcode.com/datasets` has curated licenses for some datasets. Their licensing guide can help determine the license of a dataset.

https://doi.org/10.52202/079017-3852

- For existing datasets that are re-packaged, both the original license and the license of the derived asset (if it has changed) should be provided.
- If this information is not available online, the authors are encouraged to reach out to the asset's creators.

13. **New Assets**

    Question: Are new assets introduced in the paper well documented and is the documentation provided alongside the assets?

    Answer: [NA]

    Justification: No new assets introduced.

    Guidelines:

    - The answer NA means that the paper does not release new assets.
    - Researchers should communicate the details of the dataset/code/model as part of their submissions via structured templates. This includes details about training, license, limitations, etc.
    - The paper should discuss whether and how consent was obtained from people whose asset is used.
    - At submission time, remember to anonymize your assets (if applicable). You can either create an anonymized URL or include an anonymized zip file.

14. **Crowdsourcing and Research with Human Subjects**

    Question: For crowdsourcing experiments and research with human subjects, does the paper include the full text of instructions given to participants and screenshots, if applicable, as well as details about compensation (if any)?

    Answer: [NA]

    Justification: No crowdsourcing.

    Guidelines:

    - The answer NA means that the paper does not involve crowdsourcing nor research with human subjects.
    - Including this information in the supplemental material is fine, but if the main contribution of the paper involves human subjects, then as much detail as possible should be included in the main paper.
    - According to the NeurIPS Code of Ethics, workers involved in data collection, curation, or other labor should be paid at least the minimum wage in the country of the data collector.

15. **Institutional Review Board (IRB) Approvals or Equivalent for Research with Human Subjects**

    Question: Does the paper describe potential risks incurred by study participants, whether such risks were disclosed to the subjects, and whether Institutional Review Board (IRB) approvals (or an equivalent approval/review based on the requirements of your country or institution) were obtained?

    Answer: [NA]

    Justification: No crowdsourcing.

    Guidelines:

    - The answer NA means that the paper does not involve crowdsourcing nor research with human subjects.
    - Depending on the country in which research is conducted, IRB approval (or equivalent) may be required for any human subjects research. If you obtained IRB approval, you should clearly state this in the paper.
    - We recognize that the procedures for this may vary significantly between institutions and locations, and we expect authors to adhere to the NeurIPS Code of Ethics and the guidelines for their institution.
    - For initial submissions, do not include any information that would break anonymity (if applicable), such as the institution conducting the review.