Learning to Jump: Thinning and Thickening Latent Counts for Generative Modeling

Tianqi Chen [1] **Mingyuan Zhou** [1]

Abstract

Learning to denoise has emerged as a prominent paradigm to design state-of-the-art deep generative models for natural images. How to use it to model the distributions of both continuous real-valued data and categorical data has been well studied in recently proposed diffusion models. However, it is found in this paper to have limited ability in modeling some other types of data, such as count and non-negative continuous data, that are often highly sparse, skewed, heavy-tailed, and/or overdispersed. To this end, we propose learning to jump as a general recipe for generative modeling of various types of data. Using a forward count thinning process to construct learning objectives to train a deep neural network, it employs a reverse count thickening process to iteratively refine its generation through that network. We demonstrate when learning to jump is expected to perform comparably to learning to denoise, and when it is expected to perform better. For example, learning to jump is recommended when the training data is non-negative and exhibits strong sparsity, skewness, heavy-tailedness, and/or heterogeneity.

1. Introduction

Learning how to generate realistic artificial data from random noise is a foundational problem in statistics and machine learning. A common practice to address this problem is to employ deep generative models (DGMs), which include variational auto-encoders (VAEs) (Kingma & Welling, 2013; Rezende et al., 2014), autoregressive models (van Den Oord et al., 2016; van den Oord et al., 2017; Ramesh et al., 2021; 2022), and generative adversarial networks (GANs) (Goodfellow et al., 2014) as representative examples. These DGMs usually generate a random data sample by forward propagating a random noise through a decoder, empowered by deep neural networks, or producing the elements of a sequence in an autoregressive manner, via the use of a recurrent neural network (Hochreiter & Schmidhuber, 1997; Graves & Schmidhuber, 2008) or a Transformer (Vaswani et al., 2017; Radford et al., 2018).

Different from previous generative modeling frameworks, learning to denoise, which generates a sample through iterative refinement, has recently emerged as a prominent paradigm in designing DGMs (Sohl-Dickstein et al., 2015; Song & Ermon, 2019; Ho et al., 2020; Song et al., 2021b). In this paradigm, one first corrupts the clean data with noise at various signal-to-noise ratios (SNRs), and then learns how to iteratively denoise the noisy data, using the same deep neural network that is made aware of the corresponding SNR (Kingma et al., 2021). In general, DGMs built under this learning-to-denoise framework are shown to convincingly outperform previous ones in training stability, mode coverage, and generation quality (Dhariwal & Nichol, 2021; Rombach et al., 2022; Saharia et al., 2022).

Commonly formulated as either denoising diffusion probabilistic models (DDPMs) (Sohl-Dickstein et al., 2015; Ho et al., 2020) or score-based generative models (Hyvärinen, 2005; Vincent, 2011; Song & Ermon, 2019; 2020), learning-to-denoise-based DGMs have been successfully used to model high-dimensional distributions of both continuous real-valued data (Dhariwal & Nichol, 2021; Ho et al., 2022; Ramesh et al., 2022; Rombach et al., 2022; Saharia et al., 2022) and categorical data (Hoogeboom et al., 2021; Austin et al., 2021; Gu et al., 2022; Hu et al., 2022). Despite being relatively new, they have already been deployed into a diverse set of applications, including personalized image editing (Ruiz et al., 2022), audio synthesis (Chen et al., 2021; Kong et al., 2021; Yang et al., 2023), text generation (Li et al., 2022), uncertainty quantification in classification and regression (Han et al., 2022), learning expressive policies in reinforcement learning (Wang et al., 2023), and generation of chemical and biological compounds (Shi et al., 2021; Luo et al., 2022; Jing et al., 2022), to name a few.

[*]Equal contribution [1]McCombs School of Business, The University of Texas at Austin. Correspondence to: Tianqi Chen <tqch@utexas.edu>, Mingyuan Zhou <mingyuan.zhou@mccombs.utexas.edu>.

Proceedings of the 40^{th} International Conference on Machine Learning, Honolulu, Hawaii, USA. PMLR 202, 2023.

While learning to denoise so far has been successfully applied to both continuous real-valued and categorical data, non-trivial modifications to this framework are in general required on a case-by-case basis to accommodate every new type of data, such as count and sparse non-negative data. To be more specific, let's consider the case of modeling sparse data, which is prevalent in numerous real-world applications. Examples include users' ratings of movies, consumers' purchases of products, term-frequency vectors in documents, next-generation RNA-sequencing data, and graph adjacency matrices, among others. Gaussian-based denoising diffusion models, however, are known to be inherently restrictive in modeling exact sparsity, as will also be illustrated in our experiments.

Distinct from learning to denoise, we propose learning to jump, as depicted in Figure 1, that exploits the thinning and thickening of latent counts under the Poisson distribution to build a general framework for deep generative modeling. We refer to the DGMs built under this new framework as JUMP models, which are suitable for any type of data that takes non-negative values, such as count, binary, and sparse non-negative continuous data. While existing DGMs often start their generation from random noise and move to distinct states when the generation ends, the proposed thinning and thickening-based JUMP models start the data generation process from exact zeros and can stay at exact zeros when the generation ends.

JUMP models are implemented under a Bayesian framework equipped with a multi-stochastic-layer generative network, each layer of which shares the same deep neural network-based generator. We first show any univariate non-negative observation can be recovered from a Poisson-distributed count according to the strong law of large numbers, and hence any univariate non-negative observation can be converted into a latent count with a controllable accuracy to recover its original value. We show in this latent count space, one can first thin the latent counts for model training and then iteratively thicken the latent counts with discrete non-negative jumps for data generation.

More specifically, we define the decoder of a JUMP model via a Poisson distribution-based Markov chain, with the inference network defined via a forward Poisson thinning process. The Poisson Markov chain performs iterative refinement through a deep neural network. At each refinement step, the input consists of the cumulative count of all previous steps and the time embedding of the current step, while the output is a shifted Poisson-distributed random count. To train this deep neural network, we draw a Poisson-distributed count based on the observation and send it through a forward Poisson thinning process that gradually thins each count towards zero. Therefore, the number of non-zero locations during training (generation) is monotonically decreasing (increasing) as the number of diffusion (reverse-diffusion) steps increases.

The major contributions of the paper include: 1) Introducing learning to jump as a novel framework to build DGMs; 2) Revealing that learning to denoise has limited ability in modeling non-negative data that exhibit one or multiple features from the following list: sparsity, skewness, heavy-tailedness, and overdispersion; 3) Demonstrating the ability of learning-to-jump-based DGMs in modeling complex data.

2. Learning to Jump

The proposed jump diffusion probabilistic models, as depicted in Figure 1, can be roughly described as follows. First, we re-scale the data observations by a positive constant and then encode the re-scaled observations into Poisson-distributed latent counts. The applied encoding can be lossless and admits a simple approximate inverse. Second, we decrease the latent counts towards zeros through a series of binomial distribution-based thinning operations. Third, we reverse the thinning process via a count-thickening jump process, which increases the latent counts by a series of Poisson-distributed discrete jumps. We also note that if the data itself is already count-valued ($i.e.$, non-negative integers), then the initial operation of re-scaling and randomization via Poisson is not necessary. We defer the detailed discussion of that specific case into Appendix C.

2.1. Poisson-based Data Randomization

Let us denote $\mathbb{N}_0 := \{0, 1, 2, \ldots\}$ as the set of non-negative integers and express the distribution of the observed non-negative data as

$$x_0 \sim \mathcal{P}_0 \,.$$

We first encode the observation x_0 into a count variable $z_0 \in \mathbb{N}_0$ via the Poisson distribution, whose rate is defined by the re-scaled observation λx_0 where $\lambda > 0$:

$$z_0 \sim \text{Pois}(\lambda x_0) \,.$$

The resulting latent count distribution can be viewed as a mixed Poisson distribution (Karlis & Xekalaki, 2005) of which the mixing distribution is the data distribution and is identifiable[1].

For $x_0 > 0$, the standard-deviation-to-mean ratio monotonically decreases towards zero as λ increases to infinity:

$$\frac{\text{std}(z_0)}{\mathbb{E}(z_0)} = \frac{1}{(\sqrt{\lambda x_0})} \to 0 \,.$$

Moreover, when $\mathcal{P}_0$ is a Dirac distribution $\delta(x_0)$ and λ is

[1]The term "identifiable" means being able to identify the mixing distribution of a mixture.

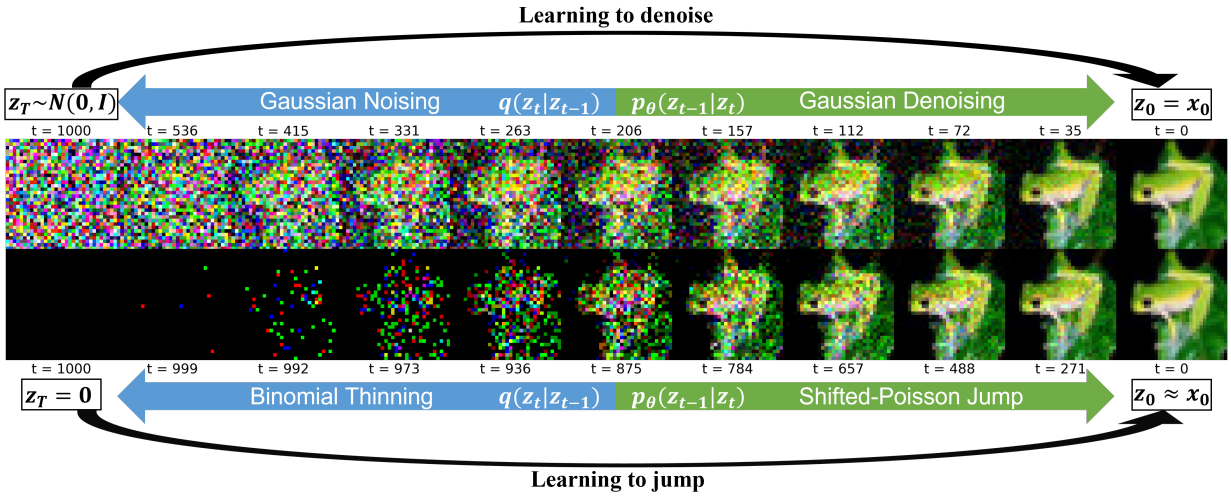

Figure 1. Illustrative comparison of learning to denoise, which adds noise for model training and denoise for data generation, and learning to jump, which thins the counts for model training and thickens the counts with Poisson jumps for data generation.

sufficiently large, approximately we have

$$\frac{z_0}{\lambda} \sim \mathcal{N}(x_0, \sigma^2),\ \sigma := \sqrt{\frac{x_0}{\lambda}}.$$

When λ approaches infinity, according to the strong law of large numbers, we have

$$\lim_{\lambda \to \infty} \frac{z_0}{\lambda} = x_0$$

with probability one. Therefore, as λ increases, $\frac{z_0}{\lambda}$ tends to provide a more and more accurate approximation to x_0. Next, we will show that the same is true for general data distributions under certain regularity conditions.

Theorem 1. *Suppose* $z_0 \sim \text{Pois}(\lambda x_0)$, $x_0 \sim \mathcal{P}_0$, *and the moment generating function* $\mathbb{E}_{x_0}[e^{tx_0}]$ *exists for* $|t| < h$, *where* $h > 0$, *then random variable*

$$\hat{x}_0 = z_0/\lambda$$

converges in distribution to $\mathcal{P}_0$ *as* λ *goes to infinity.*

Proof. The moment-generating function of $\hat{x}_0 = z_0/\lambda$ can be expressed as

$$\begin{aligned}\mathbb{E}[e^{t\frac{z_0}{\lambda}}] &= \mathbb{E}_{x_0}\mathbb{E}_{z_0 \sim \text{Pois}(\lambda x_0)}[e^{t\frac{z_0}{\lambda}}] \\ &= \mathbb{E}_{x_0}[\exp(\lambda x_0(e^{\frac{t}{\lambda}} - 1))]. \end{aligned} \quad (1)$$

Since $\lim_{\lambda \to \infty} \lambda(e^{\frac{t}{\lambda}} - 1) = t$, we have

$$\lim_{\lambda \to \infty} \mathbb{E}[e^{t\frac{z_0}{\lambda}}] = \mathbb{E}_{x_0}[e^{tx_0}],$$

where the right-hand side is the moment generating function of $x_0 \sim \mathcal{P}_0$.

□

Therefore, we will focus on modeling the distribution of latent count $z_0 \in \mathbb{N}_0$, from which we can recover x_0 as

$$\hat{x}_0 = g\left(\frac{z_0}{\lambda}\right),$$

where $g(y)$ is a function that maps its input y into the domain of x_0 and the scaling parameter λ controls the mapping accuracy. For example, if $x_0 \in \mathbb{R}_{\geq 0}$, then $g(y) = y$; if $x_0 \in \mathbb{N}_0$ is a count variable, then $g(y) = \text{round}(y)$, which rounds its input y to the nearest integer; if $x_0 \in \{0, 1\}$ is a binary variable, then $g(y) = \mathbf{1}(y > 0)$; if x_0 is a categorical variable represented as a one-hot vector, then $g(y)$ outputs a one-hot vector whose non-zero location is at the dimension that y takes its maximum value; and if x_0 is a simplex-constrained probability vector, then $g(y) = y/\|y\|_1$.

In practical scenarios, the scaling parameter λ does not necessarily need to be extremely large in order to achieve a good approximation. For instance, considering a distribution with a mean of 0.5, using a scaling parameter of 10 or 100 would result in a signal-to-noise ratio of 5 or 50, as measured by the following expression:

$$\mathbb{E}_{x_0}\left[\frac{(\mathbb{E}[z_0 \mid x_0])^2}{\text{Var}(z_0 \mid x_0)}\right] = \lambda \mathbb{E}[x_0].$$

2.2. Thinning Can be Reversed with Shifted Poisson

A well-known property of the Poisson distribution is that using a binomial distribution, one can thin a Poisson random variable into another Poisson random variable with a lower rate (*e.g.*, see page 163 of Casella & Berger (2021)). Specifically, thinning

$$p(x) = \text{Pois}(x; \lambda)$$

via

$$p(y \mid x) = \text{Binomial}(y; x, \pi),$$

where $\pi \in [0, 1]$, leads to

$$p(y) = \mathbb{E}_{x\sim p(x)}[p(y \mid x)] = \text{Pois}(y; \pi\lambda).$$

Denote $x \sim \text{Pois}_m(\lambda)$ as a shifted Poisson distribution with probability mass function

$$\Pr(x = k) = \frac{\lambda^{k-m}e^{-\lambda}}{(k-m)!}, \;\; k \in \{m, m+1, \ldots\}.$$

To reverse from $p(y)$ to $p(x)$, we show what is needed is a shifted Poisson distribution as

$$p(x \mid y) = \text{Pois}_y(x; (1-\pi)\lambda),$$

which is the same as adding a random discrete jump of $\text{Pois}((1-\pi)\lambda)$ into y to generate x. In other words, we have the following Lemma:

Lemma 1. *The Poisson-binomial bivariate count distribution and shifted-Poisson Poisson bivariate count distribution shown below are equivalent to each other:*

$$\begin{aligned} p(x, y \mid \lambda, \pi) &= \text{Binomial}(y; x, \pi)\text{Pois}(x; \lambda) \\ &= \text{Shifted-Pois}_y(x; (1-\pi)\lambda)\text{Pois}(y; \pi\lambda). \end{aligned} \tag{2}$$

The proof is straightforward by computing the probability mass functions of these two bivariate count distributions and showing they are equivalent to each other. We also note that Lemma 1 presented here can be considered as a specific instance of Lemma 4.1 in Zhou et al. (2012). The aforementioned lemma shows the equivalence between two scenarios: 1) drawing a total Poisson-distributed random count and allocating it among K distinct categories via a multinomial distribution, and 2) independently drawing K Poisson-distributed random counts. In the context of learning to jump, Lemma 1 serves as a restatement of that lemma, tailored to the framework and constraints of this specific problem, with the particular configuration of $K = 2$ representing the number of categories at each jump step.

2.3. Count-thinning-based Encoder

Starting by drawing a latent count $z_0 \sim q(z_0 \mid x_0) = \text{Pois}(z_0; \lambda x_0)$ and then gradually thinning it towards zero, we define a forward thinning process of T time steps as

$$\begin{aligned} q(z_{1:T} \mid z_0) &= \textstyle\prod_{t=1}^{T} q(z_t \mid z_{t-1}) \\ &= \textstyle\prod_{t=1}^{T} \text{Binomial}\left(z_t; z_{t-1}, \frac{\alpha_t}{\alpha_{t-1}}\right), \end{aligned} \tag{3}$$

where $\{\alpha_t\}_{0,T}$ is the set of thinning coefficients satisfying

$$1 = \alpha_0 > \alpha_1 > \ldots > \alpha_T \to 0.$$

By construction, we have

$$q(z_t \mid z_0) = \text{Binomial}\left(z_t; x_0, \alpha_t\right).$$

As $\alpha_T \to 0$, we have

$$q(z_T \mid z_0) = \text{Binomial}\left(z_T; x_0, \alpha_T\right) \to \delta_0(z_T),$$

which becomes a point mass at $z_T = 0$. In other words, the Poisson thinning process possesses an absorbing state precisely at zero.

A key property of the forward thinning process, as suggested by Lemma 1, is that the marginal distribution of the latent count at any t remains to follow a Poisson distribution as

$$\begin{aligned} q(z_t \mid x_0) &= \mathbb{E}_{z_0\sim\text{Pois}(\lambda x_0)}[q(z_t \mid z_0)] \\ &= \text{Pois}(z_t; \lambda\alpha_t x_0). \end{aligned} \tag{4}$$

With (2) and (4), we can show another key property of the thinning process: The thinning from z_{t-1} to z_t can be reversed by a conditional posterior following the shifted-Poisson distribution as

$$\begin{aligned} &q(z_{t-1} \mid z_t, x_0) \\ &= \frac{\text{Binomial}\left(z_t; z_{t-1}, \frac{\alpha_t}{\alpha_{t-1}}\right)\text{Pois}(z_{t-1}; \lambda\alpha_{t-1}x_0)}{\text{Pois}(z_t; \lambda\alpha_t x_0)} \\ &= \text{Pois}_{z_t}(z_{t-1}; \lambda(\alpha_{t-1} - \alpha_t)x_0), \end{aligned} \tag{5}$$

which is key to deriving a tractable variational lower bound.

2.4. Count-thickening-based Decoder

Let us denote

$$f_\theta(z_t, t) \geq 0$$

as a non-negative nonlinear function whose input consists of the count at time step t and the time embedding for t. We start the count-thickening process at $z_t = 0$, and at time step $t-1$, we add a jump of

$$\text{Pois}(\lambda(\alpha_{t-1} - \alpha_t)f_\theta(z_t, t))$$

into the previous accumulative count z_t to arrive at the updated accumulative count z_{t-1} at time $t-1$. More specifically, starting at $z_T \sim \text{Pois}(0)$, which means $z_T = 0$ almost surely, we define a decoder with T count-thickening steps, each of which corresponds to a shifted Poisson distribution, expressed as

$$\begin{aligned} p(z_{0:T-1} \mid z_T = 0) &= \prod_{t=1}^{T} p_\theta(z_{t-1} \mid z_t) \\ &= \prod_{t=1}^{T} q(z_{t-1} \mid z_t, \hat{x}_0 = f_\theta(z_t, t)) \\ &= \prod_{t=1}^{T} \text{Pois}_{z_t}(z_{t-1}; \lambda(\alpha_{t-1} - \alpha_t)f_\theta(z_t, t)). \end{aligned} \tag{6}$$

2.5. Auto-encoding Variational Inference

Below we provide the key steps of variational inference. While it is intractable to compute the marginal distribution $p(x_0) = \mathbb{E}_{p_\theta(z_{0:T})}[p(x_0 \mid z_{0:T})]$, similar to the optimization in DDPMs (Sohl-Dickstein et al., 2015; Ho et al., 2020), we can minimize a negative evidence lower bound (ELBO) as

$$\begin{aligned} L &= -\mathbb{E}_{x_0\sim\mathcal{P}_0}\mathbb{E}_{q(z_{0:T}|x_0)}\left[\ln\frac{p_\theta(z_{0:T},x_0)}{q(z_{0:T}\mid x_0)}\right] \\ &= \mathbb{E}_{x_0\sim\mathcal{P}_0}\left[L_{-1}+\sum_{t=1}^{T}L_{t-1}+L_T\right], \end{aligned} \tag{7}$$

where

$$L_{t-1} = \mathbb{E}_{q(z_t|x_0)}[\mathrm{KL}(q(z_{t-1}\mid z_t,x_0)\parallel p_\theta(z_{t-1}\mid z_t))] \tag{8}$$

for $1 \le t \le T$ and

$$L_{-1} = \mathbb{E}_{q(z_0|x_0)}[-\ln p_\theta(x_0\mid z_0)] \tag{9}$$

$$L_T = \mathbb{E}_{q(z_0|x_0)}[\mathrm{KL}(q(z_T\mid z_0)\parallel p(z_T))]. \tag{10}$$

As $L_T \approx 0$, it can be ignored. In practice, we do not consider L_{-1} since $\mathbb{E}_{x_0}[L_{-1}]$ minimizes when $p_\theta(x_0 \mid z_0) = q(x_0 \mid z_0)$, which can be well approximated by a deterministic mapping $g(\frac{z_0}{\lambda})$ when the scaling parameter λ is sufficiently large.

The Kullback–Leibler (KL) divergence term in (8) has an analytic expression as

$$\begin{aligned} &\mathrm{KL}\left(q(z_{t-1}\mid z_t,x_0)\parallel p(z_{t-1}\mid z_t)\right) \\ &= \lambda(\alpha_{t-1}-\alpha_t) \\ &\qquad \times [x_0(\ln x_0 - \ln f_\theta(z_t,t)) - (x_0 - f_\theta(z_t,t))] \\ &= \lambda(\alpha_{t-1}-\alpha_t)D_\varphi(x_0, f_\theta(z_t,t)), \end{aligned}$$

where $D_\varphi(p,q)$ is the relative entropy, which is a Bregman divergence (Banerjee et al., 2005) induced by the differentiable, strictly convex function $\varphi(x) = x\ln(x)$ such that

$$\begin{aligned} D_\varphi(p,q) &= \varphi(p) - \varphi(q) - (p-q)^T\nabla_q\varphi(q) \\ &= p(\ln p - \ln q) - (p-q). \end{aligned} \tag{11}$$

We divide L_{t-1} by $\lambda(\alpha_{t-1}-\alpha_t)$ to define a re-weighted negative ELBO as

$$\begin{aligned} \tilde{L} &= \mathbb{E}_{x_0\sim\mathcal{P}_0}\left[\sum_{t=1}^{T}\tilde{L}(x_0,t)\right] \\ \tilde{L}(x_0,t) &= \mathbb{E}_{z_0\sim q(z_0|x_0)}\mathbb{E}_{q(z_t|z_0)}[D_\varphi(x_0, f_\theta(z_t,t))]. \end{aligned} \tag{12}$$

As $D_\varphi(p,q) \ge 0$, we have $\tilde{L} \ge 0$ and hence we can monitor how close $\tilde{L}$ is to zero to assess convergence. Since $\mathbb{E}_{x_0\sim\mathcal{P}_0}[\tilde{L}(x_0,t)]$ can also be written as

$$\mathbb{E}_{z_t\sim q(z_t)}\mathbb{E}_{x_0\sim q(x_0|z_t)}[D_\varphi(x_0, f_\theta(z_t,t))],$$

Algorithm 1 Training

Require: Dataset $\mathcal{D}$, Mini-batch size B, Scaling parameter λ, timesteps T, thinning coefficients $\{\alpha_t\}_{t=1}^T$, and decoder deep network f_θ

repeat
  Draw a mini-batch $X_0 = \{x_0^{(i)}\}_{i=1}^B$ from $\mathcal{D}$
  for $i = 1$ to B **do**
    $t_i \sim \mathrm{Uniform}(\{1,\dots,T\})$
    $z_t^{(i)} \sim \mathrm{Poisson}(\lambda\alpha_{t_i}x_0^{(i)})$
    Compute loss $\mathcal{L}_i = D_\varphi(x_0^{(i)}, f(z_t^{(i)}, t_i))$
  end for
  Perform SGD with $\frac{1}{B}\nabla_\theta\sum_{i=1}^B\mathcal{L}_i$
until converge

using the property of the Bregman divergence (Banerjee et al., 2005), we also know that the loss is minimized at θ^* when $f_{\theta^*}(z_t,t) = \mathbb{E}[x_0 \mid z_t, t]$ for all $z_t \sim q(z_t)$.

Our algorithm is straightforward to implement:

1) Sample $t \in \{1,\dots,T\}$ uniformly at random and draw

$$z_t \sim \mathrm{Pois}(\lambda\alpha_t x_0),\ x_0 \sim \mathcal{P}_0(x_0);$$

2) Optimize θ with gradient

$$\nabla_\theta D_\varphi(x_0, f_\theta(z_t,t)) = \nabla_\theta[f_\theta(z_t,t) - x_0\ln f_\theta(z_t,t)].$$

Note that (12) can be readily extended to continuous time as

$$\int_0^1 \mathbb{E}_{x_0\sim\mathcal{P}_0}\mathbb{E}_{q(z_t|x_0)}[D_\varphi(x_0, f_\theta(z_t,t))]dt,$$

where $q(z_t \mid x_0) = \mathrm{Pois}(z_t;\lambda\alpha_t x_0)$ and $\alpha_t = \alpha(t)$ is a monotonically-decreasing function defined on $[0,1]$ such that $\alpha(0) = 1$, $\alpha(1) \approx 0$, and $1 > \alpha_s > \alpha_t > 0$ for $0 < s < t < 1$.

For data generation, we let $z_T = 0$, perform ancestral sampling via (6) to draw z_0, and then let either $\hat{x}_0 = g(z_0/\lambda)$ or $\hat{x}_0 = f_\theta(z_1, 1)$ to produce a random generation. We summarize the training and sampling algorithms in Algorithms 1 and 2, respectively.

3. Related Work

The proposed learning to jump provides a new framework to construct DGMs. Below we discuss several representative DGMs and how JUMP models differ from them.

VAEs and GANs. Both VAEs and GANs utilize deep neural networks in their data-generating process. They typically forward propagate a random noise once through an encoder, parameterized by a deep neural network, to generate a random data sample. This is different from learning to jump,

Algorithm 2 Sampling

Require: Scaling parameter λ, timesteps T, thinning coefficients $\{\alpha_t\}_{t=1}^T$, decoder deep network f_θ and constraint function g

Initialize a mini-batch $z_T = \mathbf{0}$
$z_t \leftarrow z_T$
for $t = T$ to 1 **do**
  $\hat{x}_0 \leftarrow f_\theta(z_t, t)$
  $z_{t-1} \sim z_t + \text{Poisson}(\lambda(\alpha_{t-1} - \alpha_t)\hat{x}_0)$
end for
$x_0 \leftarrow g(z_0/\lambda)$, or $x_0 = g(\hat{x}_0)$
return x_0

which needs to iterate its generation through the same deep neural network multiple times before producing a single realistic sample in the original data space. VAEs (Kingma & Welling, 2013; Rezende et al., 2014) are learned by maximizing the lower bound of an intractable log marginal likelihood, whereas GANs (Goodfellow et al., 2014) are learned under a min-max adversarial game between a discriminator and a generator. When applied to image generation, VAEs are known to generate blurry images while GANs are known to be susceptible to training instability and dropping data density modes.

A wide variety of techniques have been developed over the years to improve their performance. For VAEs, a considerable amount of effort has been spent on constructing more expressive tractable variational posteriors (Ranganath et al., 2016; Huszár, 2017; Yin & Zhou, 2018; Zhang et al., 2018; Molchanov et al., 2019; Titsias & Ruiz, 2019) and improving the decoder architecture to generate more photo-realistic images (Razavi et al., 2019; Maaløe et al., 2019; Vahdat & Kautz, 2020). Whereas for GANs, steady progress has been made on improving training stability (Arjovsky et al., 2017; Gulrajani et al., 2017; Miyato et al., 2018; Mescheder et al., 2018), generation fidelity (Radford et al., 2015; Brock et al., 2019; Karras et al., 2019; Sauer et al., 2022), and mode coverage and data efficiency (Zhao et al., 2020; Karras et al., 2020; Yang et al., 2021; Wang et al., 2022).

Learning to denoise. Both score-based generative models (Song & Ermon, 2019) and DDPMs (Ho et al., 2020) can be considered as representative DGMs developed under the learning-to-denoise framework, which generates a random sample by iteratively refining its generation through a deep neural network. From the Bayesian perspective, the method of learning to denoise can be implemented under an auto-encoding variational inference framework. Defining data generation via a multi-stochastic-layer generative network and introducing a fixed hierarchical variational encoder for inference, one may construct an ELBO of the log marginal likelihood to derive both the training and inference algorithms (Sohl-Dickstein et al., 2015; Ho et al., 2020). Specifically, as in DDPMs, one may take a Markov diffusion chain as the encoder, which gradually corrupts the data towards pure Gaussian noise by repeatedly mixing it with Gaussian noise at various scales. This provides the training data to supervise the learning of a reverse Markov diffusion chain. After being trained, this reverse chain iterates through the same deep neural network, whose inputs include the time embedding of the current diffusion step, to gradually refine a Gaussian noise into a noise-free data generation.

We note that DDPMs can also be formulated as either performing denoising score matching in a discrete-time setting or solving stochastic differential equations in a continuous-time setting (Song et al., 2021b). While the development of the JUMP models under the learning-to-jump framework mimics that of DDPMs under the learning-to-denoise framework, the JUMP models can be viewed as neither score matching nor stochastic differential equations. This is because the generations of JUMP models take count values that are not continuous.

Learning to denoise has also been generalized to model categorical data, where the noise corruption corresponds to randomly transiting a categorical observation to some other category under a pre-defined transition probability matrix (Hoogeboom et al., 2021; Austin et al., 2021). Inspired by the success of masked language models in natural language processing (Devlin et al., 2018), one may further augment the existing categories with a mask category whose self-transition probability is one (Austin et al., 2021). In other words, the mask category is an absorbing category. Consequently, unmasking becomes an essential part of the denoising process for data generation. When combined with an appropriate pretrained encoder-decoder with a tokenized discrete latent space, such as that provided by VQ-VAE (van den Oord et al., 2017) or VQ-GAN (Esser et al., 2021), learning to unmask and/or denoise in this discrete space has led to strong performance in both unconditional and text-conditioned image generation (Gu et al., 2022; Hu et al., 2022; Chang et al., 2022; 2023). Related to learning to unmask which has the mask category as its unique absorbing state, the proposed learning to jump also has a unique absorbing state, which is 0. A distinction is that the mask category is a nonexistent fictitious category, while 0 is a possible true data value in learning to jump.

Learning to reverse other types of corruptions. Several recent works have all tried to come up with a method that mimics learning to denoise but changes the data corruption from adding Gaussian noise to another type of corruption. Representative examples include learning to de-blur (Hoogeboom & Salimans, 2023), learning to reverse heat dissipation (Rissanen et al., 2023), and learning to reverse arbitrary and even noiseless/cold image transforms (Bansal et al., 2022; Daras et al., 2022). These variations of learning

to denoise ultimately all boil down to minimizing an L_2 loss between the clean data and the predicted reconstruction of the corrupted version of the data. From this perspective, learning to jump differs from all of them in having a relative entropy-based loss, as shown in (12), that is different from an L_2 loss. Another notable difference is that the starting point of the reverse chain in learning to jump is a point mass at 0, rather than some random noise from a fixed prior distribution.

4. Experiments

To demonstrate the power and versatility of the proposed learning-to-jump framework for generative modeling, we evaluate JUMP models on a diverse set of non-negative data, ranging from univariate non-negative data of various types, document term-frequency count vectors and TF-IDF vectors obtained from two representative text corpora, to natural images whose pixel values lie between 0 and 255. For data such as natural images that typically exhibit no strong sparsity, skewness, heavy-tailedness, or overdispersion, learning-to-denoise-based DGMs, such as DDPMs, have already been proven to perform well. In this case, we don't expect JUMP models to provide unique advantages in faithfully regenerating the original data distribution. However, when the data is highly sparse, skewed, heavy-tailed, and/or overdispersed, we expect the proposed JUMP models to behave differently, potentially providing distinct advantages in data regeneration, as confirmed by a rich set of experiments shown below. Our code is available at https://github.com/tqch/poisson-jump.

4.1. Univariate Non-negative Data

We consider three types of univariate non-negative data x, including $x \in \mathbb{N}_0$, $x \in [0, \infty)$, and $x \in [0, 1]$. For $x \in \mathbb{N}_0$, we synthesize three count datasets, which are characteristic of sparsity, skewness, and overdispersion, from a bi-modal Poisson mixture (PoisMix)

$$0.9 \cdot \text{Pois}(1) + 0.1 \cdot \text{Pois}(100),$$

a bi-modal negative binomial (Nbinom) mixture (NbinomMix)

$$0.75 \cdot \text{Nbinom}(1, 0.9) + 0.25 \cdot \text{Nbinom}(10, 0.1),$$

and a beta-negative-binomial (BNB) distribution with probability mass

$$P(k) = \int_0^1 \text{Nbinom}(k; 1, p)\text{Beta}(p; 1.5, 1.5)dp,\ k \in \mathbb{N}_0\,.$$

We also experiment on continuous non-negative data from either skewed or heavy-tailed distributions, including $\text{Gamma}(0.5, 0.05)$, a half-Cauchy with probability density

$$p(x) \propto (1 + x^2)^{-1},\ x \geq 0,$$

Table 1. Wasserstein-1 distances between the empirical discrete distribution of the generated random samples and **Top:** the true training distribution, **Middle & Bottom:** the empirical distribution of the training set. The unit in the Bottom table is 10^{-2}.

	Discrete $x \in \mathbb{N}_0$		
	PoisMix	NbinomMix	BNB
DDPM (Ho et al., 2020)	1.48 ± 0.37	2.17 ± 0.77	4.49 ± 4.41
D3PM Unif (Austin et al., 2021)	23.53 ± 0.48	22.31 ± 0.09	2.53 ± 0.02
D3PM Gauss (Austin et al., 2021)	17.29 ± 1.41	19.57 ± 0.67	2.59 ± 0.02
JUMP (ours)	$\mathbf{0.85 \pm 0.41}$	$\mathbf{1.84 \pm 0.44}$	$\mathbf{1.11 \pm 0.35}$

	Continuous $x \in [0, \infty)$		
	Gamma	Half-Cauchy	Half-t
DDPM (Ho et al., 2020)	0.85 ± 0.26	17.05 ± 18.58	0.51 ± 0.76
JUMP (ours)	$\mathbf{0.60 \pm 0.27}$	$\mathbf{3.56 \pm 0.76}$	$\mathbf{0.11 \pm 0.02}$

	Continuous $x \in [0, 1]$		
	Uniform	Beta$(0.5, 0.5)$	Beta$(2, 2)$
DDPM (Ho et al., 2020)	1.40 ± 0.26	$\mathbf{1.24 \pm 0.44}$	1.49 ± 0.35
JUMP (ours)	$\mathbf{1.39 \pm 0.42}$	1.57 ± 0.17	$\mathbf{1.30 \pm 0.34}$

and a half-t distribution with probability density

$$p(x) \propto \left(1 + \frac{x^2}{2}\right)^{-\frac{3}{2}},\ x \geq 0.$$

We further consider $x \in [0, 1]$ drawn from $\text{Uniform}(0, 1)$, $\text{Beta}(0.5, 0.5)$, or $\text{Beta}(2, 2)$, which correspond to flat, convex, and concave shaped probability density functions, respectively, that are all symmetric around the mean $x = 0.5$.

We draw 100,000 random samples from each distribution to form the training data. All the evaluated models in the experiment use the same 3-layer MLP architecture with 128 hidden units and leaky-ReLU activation. For D3PMs (Austin et al., 2021), we truncate the distributions at their 0.999 quantiles to ensure the number of states is finite. For the proposed JUMP, we set λ to 10 for all the univariate datasets except for Uniform, $\text{Beta}(0.5, 0.5)$ and $\text{Beta}(2, 2)$, where we use $\lambda = 100$.

We report the mean and standard deviation of the Wasserstein-1 distances between the true/empirical distribution of the training data and the empirical distribution of 100,000 generated samples over 5 independent runs. We compute Wasserstein-1 (Peyré & Cuturi, 2019) as

$$W_1(p, q) \triangleq \int_{\mathbb{R}\times\mathbb{R}} |x - y| d\pi(x, y) = \int_{\mathbb{R}} |P - Q|(x)dx,$$

where p, q are arbitrary univariate distributions, $\pi(x, y)$ is their coupling, and P, Q are cumulative distribution functions of p, q respectively. When p, q are two empirical distributions with the same size n, it reduces to compute $\|\text{sort}(X) - \text{sort}(Y)\|_1/n$, where X, Y are n-dimensional data vectors of p, q.

We summarize the results in Table 1 and Figure 2. The results on $\text{Gamma}(0.5, 0.05)$ and the 3 datasets supported

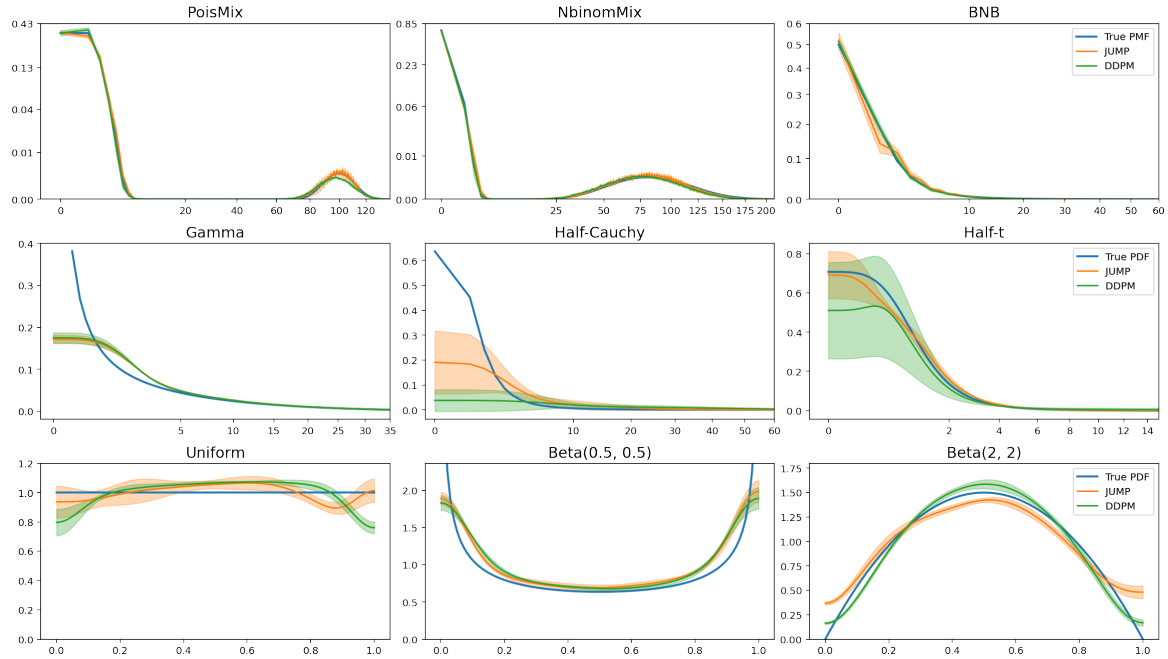

Figure 2. Visual comparison of the true and regenerated probability distributions by both DDPM and JUMP, across 9 univariate synthetic datasets described in Section 4.1.

on $[0, 1]$ all suggest that when the data is well behaved in the sense that there is no strong sparsity, skewness, heavy-tailedness, or overdispersion, the proposed JUMP has comparable performance to DDPM in terms of the Wasserstein-1 metric. However, the results on the other 5 training datasets, which all clearly exhibit sparsity, skewness, heavy-tailedness, and/or overdispersion, show that the proposed JUMP convincingly outperforms DDPM (and also D3PM on discrete data).

4.2. Sparse and Heterogeneous Data

Bag-of-words (BOW) and term frequency-inverse document frequency (TF-IDF) are two common types of document representations. Both are known to be highly sparse and heterogeneous. We prepare two datasets for the document generation task: 20 Newsgroups[2] and NeurIPS[3].

The 20 Newsgroups dataset comprises 18,846 news posts on 20 topics. The NeurIPS dataset is a collection of 7241 papers published in NeurIPS from 1987 to 2016. After standard preprocessing ($e.g.$, removing stop-words), we represent each document in 20 Newsgroup and NeurIPS with a count / TF-IDF vector of 8934 and 12038 dimensions, respectively. We set λ as 10 for both datasets. For the evaluation of BOW, we consider two summary statistics of the documents, including the sparsity and the length between the true data and generated samples, while for TF-IDF we consider the sparsity and the ℓ_1-norm.

Table 2. **Top:** Wasserstein-1 distances of summary statistics (sparsity and length) between true samples and generated samples in BOW representations. **Bottom:** Wasserstein-1 distances of summary statistics (sparsity and ℓ_1-norm) between true data and generated samples in TF-IDF representations.

	BOW			
	20 Newsgroup		NeurIPS	
	Sparsity (‰)	Length	Sparsity (%)	Length
DDPM (Ho et al., 2020)	5.49 ± 0.04	81.80 ± 10.2	3.46 ± 0.38	191.86 ± 98.4
JUMP (ours)	$\mathbf{2.65 \pm 0.46}$	$\mathbf{35.43 \pm 4.16}$	$\mathbf{2.40 \pm 0.29}$	$\mathbf{95.56 \pm 29.36}$

	TF-IDF			
	20 Newsgroup		NeurIPS	
	Sparsity (‰)	ℓ_1-norm	Sparsity (%)	ℓ_1-norm
DDPM (Ho et al., 2020)	693.21 ± 5.45	0.65 ± 0.02	66.59 ± 1.74	$\mathbf{0.20 \pm 0.02}$
JUMP (ours)	$\mathbf{3.18 \pm 0.06}$	$\mathbf{0.41 \pm 0.02}$	$\mathbf{6.23 \pm 0.00}$	0.26 ± 0.00

We report the Wasserstein-1 distances of the summary statistic distributions in Table 2. The results show that JUMP has done extremely well in recovering the inherent sparsity of document-type data. In contrast, DDPM has almost completely failed in capturing the sparsity patterns of the TF-IDF documents. Overall, JUMP consistently outperforms DDPM in all scenarios and metrics except for the ℓ_1-norm of the NeurIPS dataset using TF-IDF representation, where the performance is comparable to that of DDPM.

[2] http://qwone.com/~jason/20Newsgroups/

[3] https://www.kaggle.com/datasets/benhamner/nips-papers

4.3. Natural Images

Images are by nature high-dimensional data with 256 ordinal pixel values. Typically, the pixel values in images are neither sparse nor heavy-tailed, and hence as analyzed in Section 4.1, we do not expect JUMP to outperform DDPM. Our experiments show that while JUMP currently underperforms DDPM in modeling natural images, using the same UNet architecture and diffusion schedule that have been well-tuned to suit DDPM, it can nevertheless generate realistic-looking natural images and achieve comparable evaluation results on standard metrics, including Fréchet Inception distance (FID) (Heusel et al., 2017) and Inception score (IS) (Salimans et al., 2016), to diffusion models that operate on non-integer latent states. We present uncurated randomly-generated images in Figure 3 and report the FID and IS metrics in Table 3.

Figure 3. Uncurated randomly-generated samples by JUMP trained on CIFAR-10.

Table 3. Comparison of different models on CIFAR-10.

Latent space	Model	FID (↓)	IS (↑)
Real	DDPM (Ho et al., 2020)	3.17	9.46
	Bit Diffusion (Chen et al., 2023)	3.48	-
Categorical	D3PM Gauss+Logistic (Austin et al., 2021)	7.34	8.56
	τLDR-10 (Campbell et al., 2022)	3.74	9.49
Integer	JUMP (ours)	4.80	9.04

5. Limitations and Future Work

A notable limitation of the learning-to-denoise framework is that it often needs to iterate through the same denoising deep neural network hundreds or even thousands of times to refine its generation, which increases the computational cost of data generation by orders of magnitude compared to VAEs and GANs of similar sizes. The proposed learning-to-jump framework has the same limitation, as it also requires iterating T times through $f_\theta(z_t, t)$ to generate a single output, where T often needs to be sufficiently large, $e.g$, $T = 1000$, especially for high-dimensional data whose different dimensions exhibit complex dependencies. For learning to denoise, a variety of methods have been proposed to accelerate the generation, but often at the expense of somewhat compromised generation quality when T is limited to a small number (Song et al., 2021a; Luhman & Luhman, 2021; Kong & Ping, 2021; Xiao et al., 2022; Salimans & Ho, 2022; Zheng et al., 2023; Lu et al., 2022). How these acceleration techniques developed for learning to denoise can be extended for the proposed learning-to-jump framework is a research topic worth further investigation.

Another limitation, as shown by the results in Figure 2 and Table 1 and the image generation results in Table 3, is that for "normal" data that are not sparse, skewed, heavy-tailed, or heterogeneous, JUMP often trails behind DDPM, under the same UNet architecture and diffusion schedule that have been tailored for DDPM. How to further improve JUMP for "normal" data, such as by developing a customized model architecture and optimizing the diffusion schedule, is worth further investigation.

6. Conclusion

Iterative-refinement-based deep generative models (DGMs) are typically developed under a learning-to-denoise framework, which includes diffusion and score-based generative models as representative examples. They have shown impressive performance in capturing the high-dimensional distribution of natural images, but may not perform that well when the data is characterized by sparsity, skewness, heavy-tailedness, overdispersion, and/or heterogeneity. To this end, we propose learning to jump as a novel generative-modeling framework, which is well suited to model sparse, skewed, heavy-tailed, overdispersed, and/or heterogeneous data, and hence generalizes the applicability of DGMs into broader settings. Experimental results on a diverse set of data of various types demonstrate the unique behaviors and modeling potentials of the learning-to-jump-based DGMs. In particular, for high-dimensional data, we recommend using learning-to-jump in lieu of learning-to-denoise when the training data are highly sparse and heterogeneous.

Acknowledgments

The authors acknowledge the support of NSF-IIS 2212418, the Fall 2022 McCombs REG award, the NSF AI Institute for Foundations of Machine Learning (IFML), and Texas Advanced Computing Center (TACC).

References

Arjovsky, M., Chintala, S., and Bottou, L. Wasserstein generative adversarial networks. In *Proceedings of the 34th International Conference on Machine Learning-Volume 70*, pp. 214–223, 2017.

Austin, J., Johnson, D. D., Ho, J., Tarlow, D., and van den Berg, R. Structured denoising diffusion models in discrete state-spaces. *Advances in Neural Information Processing Systems*, 34:17981–17993, 2021.

Ba, J. L., Kiros, J. R., and Hinton, G. E. Layer normalization. *arXiv preprint arXiv:1607.06450*, 2016.

Banerjee, A., Merugu, S., Dhillon, I. S., Ghosh, J., and Lafferty, J. Clustering with bregman divergences. *Journal of machine learning research*, 6(10), 2005.

Bansal, A., Borgnia, E., Chu, H.-M., Li, J. S., Kazemi, H., Huang, F., Goldblum, M., Geiping, J., and Goldstein, T. Cold diffusion: Inverting arbitrary image transforms without noise. *arXiv preprint arXiv:2208.09392*, 2022.

Brock, A., Donahue, J., and Simonyan, K. Large scale GAN training for high fidelity natural image synthesis. In *International Conference on Learning Representations*, 2019. URL https://openreview.net/forum?id=B1xsqj09Fm.

Campbell, A., Benton, J., Bortoli, V. D., Rainforth, T., Deligiannidis, G., and Doucet, A. A continuous time framework for discrete denoising models. In Oh, A. H., Agarwal, A., Belgrave, D., and Cho, K. (eds.), *Advances in Neural Information Processing Systems*, 2022. URL https://openreview.net/forum?id=DmT862YAieY.

Casella, G. and Berger, R. L. *Statistical inference*. Cengage Learning, 2021.

Chang, H., Zhang, H., Jiang, L., Liu, C., and Freeman, W. T. MaskGIT: Masked generative image transformer. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pp. 11315–11325, 2022.

Chang, H., Zhang, H., Barber, J., Maschinot, A., Lezama, J., Jiang, L., Yang, M.-H., Murphy, K., Freeman, W. T., Rubinstein, M., et al. Muse: Text-to-image generation via masked generative transformers. *arXiv preprint arXiv:2301.00704*, 2023.

Chen, N., Zhang, Y., Zen, H., Weiss, R. J., Norouzi, M., and Chan, W. WaveGrad: Estimating gradients for waveform generation. In *International Conference on Learning Representations*, 2021. URL https://openreview.net/forum?id=NsMLjcFaO8O.

Chen, T., ZHANG, R., and Hinton, G. Analog bits: Generating discrete data using diffusion models with self-conditioning. In *The Eleventh International Conference on Learning Representations*, 2023. URL https://openreview.net/forum?id=3itjR9QxFw.

Daras, G., Delbracio, M., Talebi, H., Dimakis, A. G., and Milanfar, P. Soft diffusion: Score matching for general corruptions. *arXiv preprint arXiv:2209.05442*, 2022.

Devlin, J., Chang, M.-W., Lee, K., and Toutanova, K. Bert: Pre-training of deep bidirectional transformers for language understanding. *arXiv preprint arXiv:1810.04805*, 2018.

Dhariwal, P. and Nichol, A. Q. Diffusion models beat GANs on image synthesis. In Beygelzimer, A., Dauphin, Y., Liang, P., and Vaughan, J. W. (eds.), *Advances in Neural Information Processing Systems*, 2021. URL https://openreview.net/forum?id=AAWuCvzaVt.

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

Goodfellow, I., Pouget-Abadie, J., Mirza, M., Xu, B., Warde-Farley, D., Ozair, S., Courville, A., and Bengio, Y. Generative adversarial nets. In *Advances in Neural Information Processing Systems*, pp. 2672–2680, 2014.

Graves, A. and Schmidhuber, J. Offline handwriting recognition with multidimensional recurrent neural networks. *Advances in neural information processing systems*, 21, 2008.

Gu, S., Chen, D., Bao, J., Wen, F., Zhang, B., Chen, D., Yuan, L., and Guo, B. Vector quantized diffusion model for text-to-image synthesis. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pp. 10696–10706, 2022.

Gulrajani, I., Ahmed, F., Arjovsky, M., Dumoulin, V., and Courville, A. C. Improved training of Wasserstein GANs. In *Advances in Neural Information Processing Systems*, pp. 5767–5777, 2017.

Han, X., Zheng, H., and Zhou, M. CARD: Classification and regression diffusion models. In *Advances in Neural Information Processing Systems*, 2022.

Heusel, M., Ramsauer, H., Unterthiner, T., Nessler, B., and Hochreiter, S. GANs trained by a two time-scale update rule converge to a local Nash equilibrium. In *Advances in Neural Information Processing Systems*, pp. 6626–6637, 2017.

Ho, J., Jain, A., and Abbeel, P. Denoising diffusion probabilistic models. In *Advances in Neural Information Processing Systems*, 2020.

Ho, J., Saharia, C., Chan, W., Fleet, D. J., Norouzi, M., and Salimans, T. Cascaded diffusion models for high fidelity image generation. *J. Mach. Learn. Res.*, 23:47–1, 2022.

Hochreiter, S. and Schmidhuber, J. Long short-term memory. *Neural computation*, 9(8):1735–1780, 1997.

Hoogeboom, E. and Salimans, T. Blurring diffusion models. In *The Eleventh International Conference on Learning Representations*, 2023. URL https://openreview.net/forum?id=OjDkC57x5sz.

Hoogeboom, E., Nielsen, D., Jaini, P., Forré, P., and Welling, M. Argmax flows and multinomial diffusion: Learning categorical distributions. *Advances in Neural Information Processing Systems*, 34:12454–12465, 2021.

Hu, M., Wang, Y., Cham, T.-J., Yang, J., and Suganthan, P. N. Global context with discrete diffusion in vector quantised modelling for image generation. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pp. 11502–11511, 2022.

Huszár, F. Variational inference using implicit distributions. *arXiv preprint arXiv:1702.08235*, 2017.

Hyvärinen, A. Estimation of non-normalized statistical models by score matching. *Journal of Machine Learning Research*, 6(24):695–709, 2005. URL http://jmlr.org/papers/v6/hyvarinen05a.html.

Jing, B., Corso, G., Chang, J., Barzilay, R., and Jaakkola, T. S. Torsional diffusion for molecular conformer generation. In Oh, A. H., Agarwal, A., Belgrave, D., and Cho, K. (eds.), *Advances in Neural Information Processing Systems*, 2022. URL https://openreview.net/forum?id=w6fj2r62r_H.

Karlis, D. and Xekalaki, E. Mixed poisson distributions. *International Statistical Review/Revue Internationale de Statistique*, pp. 35–58, 2005.

Karras, T., Laine, S., and Aila, T. A style-based generator architecture for generative adversarial networks. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pp. 4401–4410, 2019.

Karras, T., Aittala, M., Hellsten, J., Laine, S., Lehtinen, J., and Aila, T. Training generative adversarial networks with limited data. *Advances in Neural Information Processing Systems*, 33:12104–12114, 2020.

Kingma, D., Salimans, T., Poole, B., and Ho, J. Variational diffusion models. *Advances in neural information processing systems*, 34:21696–21707, 2021.

Kingma, D. P. and Ba, J. Adam: A method for stochastic optimization. In Bengio, Y. and LeCun, Y. (eds.), *3rd International Conference on Learning Representations, ICLR 2015, San Diego, CA, USA, May 7-9, 2015, Conference Track Proceedings*, 2015. URL http://arxiv.org/abs/1412.6980.

Kingma, D. P. and Welling, M. Auto-encoding variational Bayes. *arXiv preprint arXiv:1312.6114*, 2013.

Kong, Z. and Ping, W. On fast sampling of diffusion probabilistic models. *arXiv preprint arXiv:2106.00132*, 2021.

Kong, Z., Ping, W., Huang, J., Zhao, K., and Catanzaro, B. DiffWave: A versatile diffusion model for audio synthesis. In *International Conference on Learning Representations*, 2021. URL https://openreview.net/forum?id=a-xFK8Ymz5J.

Li, X. L., Thickstun, J., Gulrajani, I., Liang, P., and Hashimoto, T. Diffusion-LM improves controllable text generation. In Oh, A. H., Agarwal, A., Belgrave, D., and Cho, K. (eds.), *Advances in Neural Information Processing Systems*, 2022. URL https://openreview.net/forum?id=3s9IrEsjLyk.

Loshchilov, I. and Hutter, F. Decoupled weight decay regularization. In *International Conference on Learning Representations*, 2019. URL https://openreview.net/forum?id=Bkg6RiCqY7.

Lu, C., Zhou, Y., Bao, F., Chen, J., Li, C., and Zhu, J. DPM-solver: A fast ODE solver for diffusion probabilistic model sampling in around 10 steps. In Oh, A. H., Agarwal, A., Belgrave, D., and Cho, K. (eds.), *Advances in Neural Information Processing Systems*, 2022. URL https://openreview.net/forum?id=2uAaGwlP_V.

Luhman, E. and Luhman, T. Knowledge distillation in iterative generative models for improved sampling speed. *arXiv preprint arXiv:2101.02388*, 2021.

Luo, S., Su, Y., Peng, X., Wang, S., Peng, J., and Ma, J. Antigen-specific antibody design and optimization with diffusion-based generative models for protein structures. In Oh, A. H., Agarwal, A., Belgrave, D., and Cho, K. (eds.), *Advances in Neural Information Processing Systems*, 2022. URL https://openreview.net/forum?id=jSorGn2Tjg.

Maaløe, L., Fraccaro, M., Liévin, V., and Winther, O. BIVA: A very deep hierarchy of latent variables for generative modeling. *Advances in neural information processing systems*, 32, 2019.

Mescheder, L., Geiger, A., and Nowozin, S. Which training methods for GANs do actually converge? In *International conference on machine learning*, pp. 3481–3490. PMLR, 2018.

Miyato, T., Kataoka, T., Koyama, M., and Yoshida, Y. Spectral normalization for generative adversarial networks. In *International Conference on Learning Representations*, 2018.

Molchanov, D., Kharitonov, V., Sobolev, A., and Vetrov, D. Doubly semi-implicit variational inference. In *The 22nd International Conference on Artificial Intelligence and Statistics*, pp. 2593–2602. PMLR, 2019.

Nichol, A. Q. and Dhariwal, P. Improved denoising diffusion probabilistic models. In *International Conference on Machine Learning*, pp. 8162–8171. PMLR, 2021.

Peyré, G. and Cuturi, M. Computational optimal transport. *Foundations and Trends in Machine Learning*, 11(5-6): 355–607, 2019.

Radford, A., Metz, L., and Chintala, S. Unsupervised representation learning with deep convolutional generative adversarial networks. *arXiv preprint arXiv:1511.06434*, 2015.

Radford, A., Narasimhan, K., Salimans, T., Sutskever, I., et al. Improving language understanding by generative pre-training. 2018.

Ramesh, A., Pavlov, M., Goh, G., Gray, S., Voss, C., Radford, A., Chen, M., and Sutskever, I. Zero-shot text-to-image generation. In *International Conference on Machine Learning*, pp. 8821–8831. PMLR, 2021.

Ramesh, A., Dhariwal, P., Nichol, A., Chu, C., and Chen, M. Hierarchical text-conditional image generation with clip latents. *arXiv preprint arXiv:2204.06125*, 2022.

Ranganath, R., Tran, D., and Blei, D. Hierarchical variational models. In *International conference on machine learning*, pp. 324–333. PMLR, 2016.

Razavi, A., Van den Oord, A., and Vinyals, O. Generating diverse high-fidelity images with VQ-VAE-2. *Advances in neural information processing systems*, 32, 2019.

Rezende, D. J., Mohamed, S., and Wierstra, D. Stochastic backpropagation and approximate inference in deep generative models. In *International conference on machine learning*, pp. 1278–1286. PMLR, 2014.

Rissanen, S., Heinonen, M., and Solin, A. Generative modelling with inverse heat dissipation. In *The Eleventh International Conference on Learning Representations*, 2023. URL https://openreview.net/forum?id=4PJUBT9f2Ol.

Rombach, R., Blattmann, A., Lorenz, D., Esser, P., and Ommer, B. High-resolution image synthesis with latent diffusion models. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pp. 10684–10695, 2022.

Ruiz, N., Li, Y., Jampani, V., Pritch, Y., Rubinstein, M., and Aberman, K. DreamBooth: Fine tuning text-to-image diffusion models for subject-driven generation. 2022.

Saharia, C., Chan, W., Saxena, S., Li, L., Whang, J., Denton, E. L., Ghasemipour, K., Gontijo Lopes, R., Karagol Ayan, B., Salimans, T., et al. Photorealistic text-to-image diffusion models with deep language understanding. *Advances in Neural Information Processing Systems*, 35: 36479–36494, 2022.

Salimans, T. and Ho, J. Progressive distillation for fast sampling of diffusion models. In *International Conference on Learning Representations*, 2022. URL https://openreview.net/forum?id=TIdIXIpzhoI.

Salimans, T., Goodfellow, I., Zaremba, W., Cheung, V., Radford, A., and Chen, X. Improved techniques for training GANs. In *Advances in Neural Information Processing Systems*, pp. 2234–2242, 2016.

Sauer, A., Schwarz, K., and Geiger, A. StyleGAN-XL: Scaling StyleGAN to large diverse datasets. In *ACM SIGGRAPH 2022 Conference Proceedings*, pp. 1–10, 2022.

Shi, C., Luo, S., Xu, M., and Tang, J. Learning gradient fields for molecular conformation generation. In *International Conference on Machine Learning*, pp. 9558–9568. PMLR, 2021.

Sohl-Dickstein, J., Weiss, E., Maheswaranathan, N., and Ganguli, S. Deep unsupervised learning using nonequilibrium thermodynamics. In *International Conference on Machine Learning*, pp. 2256–2265. PMLR, 2015.

Song, J., Meng, C., and Ermon, S. Denoising diffusion implicit models. In *International Conference on Learning Representations*, 2021a. URL https://openreview.net/forum?id=St1giarCHLP.

Song, Y. and Ermon, S. Generative modeling by estimating gradients of the data distribution. In *Advances in Neural Information Processing Systems*, pp. 11918–11930, 2019.

Song, Y. and Ermon, S. Improved techniques for training score-based generative models. *Advances in neural information processing systems*, 33:12438–12448, 2020.

Song, Y., Sohl-Dickstein, J., Kingma, D. P., Kumar, A., Ermon, S., and Poole, B. Score-based generative modeling through stochastic differential equations. In *International Conference on Learning Representations*,

2021b. URL https://openreview.net/forum?id=PxTIG12RRHS.

Titsias, M. K. and Ruiz, F. Unbiased implicit variational inference. In *The 22nd International Conference on Artificial Intelligence and Statistics*, pp. 167–176. PMLR, 2019.

Vahdat, A. and Kautz, J. NVAE: A deep hierarchical variational autoencoder. In *Advances in neural information processing systems*, 2020.

van Den Oord, A., Kalchbrenner, N., and Kavukcuoglu, K. Pixel recurrent neural networks. In *International conference on machine learning*, pp. 1747–1756. PMLR, 2016.

van den Oord, A., Vinyals, O., and Kavukcuoglu, K. Neural discrete representation learning. *Advances in neural information processing systems*, 30, 2017.

Vaswani, A., Shazeer, N., Parmar, N., Uszkoreit, J., Jones, L., Gomez, A. N., Kaiser, Ł., and Polosukhin, I. Attention is all you need. In *Advances in neural information processing systems*, pp. 5998–6008, 2017.

Vincent, P. A connection between score matching and denoising autoencoders. *Neural computation*, 23(7):1661–1674, 2011.

Wang, Z., Zheng, H., He, P., Chen, W., and Zhou, M. Diffusion-GAN: Training GANs with diffusion. *arXiv preprint arXiv:2206.02262*, 2022.

Wang, Z., Hunt, J. J., and Zhou, M. Diffusion policies as an expressive policy class for offline reinforcement learning. In *The Eleventh International Conference on Learning Representations*, 2023. URL https://openreview.net/forum?id=AHvFDPi-FA.

Xiao, Z., Kreis, K., and Vahdat, A. Tackling the generative learning trilemma with denoising diffusion GANs. In *International Conference on Learning Representations*, 2022. URL https://openreview.net/forum?id=JprM0p-q0Co.

Yang, C., Shen, Y., Xu, Y., and Zhou, B. Data-efficient instance generation from instance discrimination. *Advances in Neural Information Processing Systems*, 34: 9378–9390, 2021.

Yang, D., Yu, J., Wang, H., Wang, W., Weng, C., Zou, Y., and Yu, D. Diffsound: Discrete diffusion model for text-to-sound generation. *IEEE/ACM Transactions on Audio, Speech, and Language Processing*, 31:1720–1733, 2023. doi: 10.1109/TASLP.2023.3268730.

Yin, M. and Zhou, M. Semi-implicit variational inference. In *International Conference on Machine Learning*, pp. 5660–5669, 2018.

Zhang, C., Bütepage, J., Kjellström, H., and Mandt, S. Advances in variational inference. *IEEE transactions on pattern analysis and machine intelligence*, 41(8):2008–2026, 2018.

Zhao, S., Liu, Z., Lin, J., Zhu, J.-Y., and Han, S. Differentiable augmentation for data-efficient gan training. *Advances in Neural Information Processing Systems*, 33: 7559–7570, 2020.

Zheng, H., He, P., Chen, W., and Zhou, M. Truncated diffusion probabilistic models and diffusion-based adversarial auto-encoders. In *The Eleventh International Conference on Learning Representations*, 2023. URL https://openreview.net/forum?id=HDxgaKk956l.

Zhou, M., Hannah, L., Dunson, D., and Carin, L. Beta-negative binomial process and Poisson factor analysis. In *Artificial Intelligence and Statistics*, pp. 1462–1471. PMLR, 2012.

Learning to Jump: Appendix

A. Hyperparameter Settings

A.1. Diffusion Schedule

We have not conducted an extensive tuning of the diffusion schedule. As a default choice, we utilize the beta-linear schedule introduced in the work of Ho et al. (2020). We set β_1 to 0.001 by default. The value of β_T is selected such that the log-SNR (Signal-to-Noise Ratio) will be approximately -12 on average at the end of the forward chain, ensuring that the loss of the last time step L_T is approximately 0.

A.2. Scaling Parameter

Intuitively speaking, the scaling parameter λ controls how close the initial latent count distribution and original data distribution are. Specifically, the larger λ is, the more precise the transform $f : z_0 \mapsto z_0/\lambda$ will be to recover the x_0 in distribution. Based on our practical observations, we have found that starting with values of 10 or 100 for the noise schedule parameter is often suitable for most datasets, excluding image data. These values have shown promising performance as initial choices in our experiments. However, it is important to note that for image datasets, larger values often yield better results, and further experimentation and tuning are recommended to determine the optimal scaling parameter for a specific image dataset. Indeed, we find out that the Fréchet Inception Distance (FID) (Heusel et al., 2017) is highly sensitive to noise, even when the noise becomes imperceptible to humans. Figure 4 illustrates the relationship between the scaling parameter λ and the corresponding FID between the data distribution of CIFAR-10 and the reconstructed distribution obtained from the initial latent counts, $i.e.$, Poisson-randomized data.

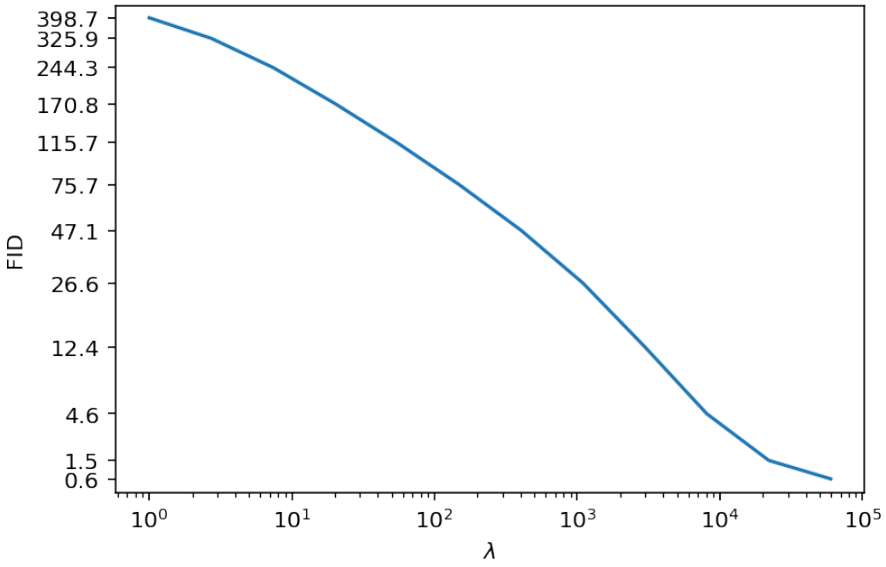

Figure 4. Relationship between FID of the initial latent count distribution and the scaling parameter λ on CIFAR-10

A.3. Training

For all the univariate and document-type datasets, our models are trained for 600 epochs using the Adam optimizer (Kingma & Ba, 2015). We use a fixed learning rate of 0.001 and the default values for the parameters $\beta_1 = 0.9$ and $\beta_2 = 0.999$.

In the case of CIFAR-10 image generation, we utilize the AdamW optimizer (Loshchilov & Hutter, 2019) with a learning rate of 0.0002 and a weight decay of 0.001. The JUMP model is trained for 3600 epochs. Unlike DDPM, we do not employ any learning rate warmup schedule in our training process.

B. Model Architectures

In the case of univariate and document datasets, we employ a Multi-Layer Perceptron (MLP) model architecture composed of three blocks. Each block consists of two fully-connected layers and one time embedding projection layer. Intermediate layers use Leaky-ReLU activation along with layer normalization (Ba et al., 2016). For CIFAR-10 image generation, we use a UNet model architecture similar to the one used by Nichol & Dhariwal (2021). Our UNet model has three stages through downsampling and upsampling, which correspond to spatial dimensions of 32×32, 16×16, and 8×8. Each stage of the model consists of 3 residual blocks with 128 hidden channels followed by a self-attention layer except for the first stage. In addition, we use a dropout rate of 0.2 for extra regularization.

C. Binomial JUMP for Count Data

We also consider a variant of the proposed JUMP model when the original data are already counts and hence Poisson randomization may not be necessary. Specifically, we remove the Poisson randomization step used by default in Poisson JUMP, resulting in a JUMP variant where the conditional posteriors of the reverse process follow the shifted-binomial distributions. We refer to this variant as binomial JUMP. The definition of the forward process of binomial JUMP can be expressed as

$$q(z_{1:T} \mid x_0) = \prod_{t=1}^{T} q(z_t \mid z_{t-1}), \tag{13}$$

$$q(z_t \mid z_{t-1}) = \text{Binomial}\left(z_t; z_{t-1}, \frac{\alpha_t}{\alpha_{t-1}}\right), \tag{14}$$

where $z_0 = x_0$, $1 = \alpha_0 > \alpha_1 > \ldots > \alpha_T$. It immediately follows that the marginal distribution of z_t given x_0 is binomial:

$$q(z_t \mid x_0) = \text{Binomial}(z_t; x_0, \alpha_t).$$

If we know z_t and x_0, then we have $z_t \leq z_{t-1} \leq x_0$ almost surely. Thus, with Bayes' rule, we have

$$q(z_{t-1} = m \mid z_t, x_0) \propto \text{Binomial}(m; x_0, \alpha_{t-1})\text{Binomial}\left(z_t; m, \frac{\alpha_t}{\alpha_{t-1}}\right) \tag{15}$$

$$\propto \frac{x_0!}{m!(x_0-m)!}\alpha_{t-1}^{m}(1-\alpha_{t-1})^{x_0-m}\frac{m!}{z_t!(m-z_t)!}\left(\frac{\alpha_t}{\alpha_{t-1}}\right)^{z_t}\left(1-\frac{\alpha_t}{\alpha_{t-1}}\right)^{m-z_t} \tag{16}$$

$$\propto \frac{1}{(x_0-m)!(m-z_t)!}\left(\frac{\alpha_{t-1}-\alpha_t}{1-\alpha_{t-1}}\right)^{m} \tag{17}$$

$$\propto \frac{(x_0-z_t)!}{(x_0-m)!(m-z_t)!}\left(\frac{\alpha_{t-1}-\alpha_t}{1-\alpha_t}\right)^{m-z_t}\left(1-\frac{\alpha_{t-1}-\alpha_t}{1-\alpha_t}\right)^{x_0-m} \tag{18}$$

$$= \text{Binomial}\left(m-z_t; x_0-z_t, \frac{\alpha_{t-1}-\alpha_t}{1-\alpha_t}\right), \tag{19}$$

where $z_t \leq m \leq x_0$. Therefore, the conditional posterior is a shifted-Binomial distribution as

$$q(z_{t-1} \mid z_t, x_0) = \text{Shifted-Binomial}_{z_t}\left(x_0 - z_t, p_t\right), \;\; p_t = \frac{\alpha_{t-1}-\alpha_t}{1-\alpha_t}. \tag{20}$$

However, in this case, we cannot simply let $p_\theta(z_{t-1} \mid z_t) = q(z_{t-1} \mid z_t, \hat{x}_0 = f_\theta(z_t, t))$. This is because, in order for the KL divergence from the approximated conditional posterior to the true one to be well defined, we will need to ensure that both $f_\theta(z_t, t)$ is a count and $f_\theta(z_t, t) \geq x_0$, which are difficult to realize in practice using a non-linear function defined by the deep neural network-based f_θ. To address this issue, noticing that a binomial distribution $x \sim \text{Binomial}(n, p)$ can often be well approximated by a Poisson distribution $x \sim \text{Pois}(np)$ when n is large and np is small, we propose to approximate the shifted-binomial distribution in (20) with a shifted-Poisson distribution as

$$\hat{q}(z_{t-1} \mid z_t, x_0) = \text{Shifted-Pois}_{z_t}\left(z_{t-1}; p_t(x_0 - z_t)\right),$$

and define a Markovian reverse process as $p_\theta(z_{0:T-1} \mid z_T = 0) = \prod_{t=1}^{T} p_\theta(z_{t-1} \mid z_t)$, where

$$p_\theta(z_{t-1} \mid z_t) = \sum_{n_t} \text{Shifted-Binomial}_{z_t}\left(z_{t-1}; n_t, p_t\right) \text{Pois}(n_t; \max(f_\theta(z_t, t) - z_t, 0)) \tag{21}$$

$$= \text{Shifted-Pois}_{z_t}\left(z_{t-1}; p_t \max(f_\theta(z_t, t) - z_t, 0)\right). \tag{22}$$

Similar to the derivation of Equation (7), the approximated negative ELBO loss of binomial jump can be expressed as follows:

$$L = -\mathbb{E}_{x_0}\mathbb{E}_{q(z_{1:T}|x_0)}\left[\ln \frac{p_\theta(z_{1:T}, x_0)}{q(z_{1:T} \mid x_0)}\right] = \mathbb{E}_{x_0}\left[L_0 + \sum_{t=2}^{T} L_{t-1} + L_T\right] \tag{23}$$

where

$$L_0 = \mathbb{E}_{q(z_1|x_0)}\left[-\ln p_\theta(x_0 \mid z_1)\right] \tag{24}$$

$$L_{t-1} = \mathbb{E}_{q(z_t|x_0)}\left[D_\varphi\left(p_t\left(x_0 - z_t\right), p_t \max(f_\theta(z_t, t) - z_t, 0)\right)\right], \quad \text{for } t = 2, \ldots, T \tag{25}$$

$$L_T = \text{KL}(q(z_T \mid x_0) \parallel p(z_T)) \tag{26}$$

where $D_\varphi(\cdot, \cdot)$ denotes the relative entropy defined in (11). More specifically, ignoring p_t in (25), we have

$$D_\varphi\left(x_0 - z_t, \max(f_\theta(z_t, t) - z_t, 0)\right) = (x_0 - z_t) \ln \frac{x_0 - z_t}{\max(f_\theta(z_t, t) - z_t, 0)} - \left[(x_0 - z_t) - \max(f_\theta(z_t, t) - z_t, 0)\right].$$

We observe that the relative entropy-based loss function of the binomial JUMP, as illustrated above, shares a close connection with the loss function of the Poisson JUMP, as shown in (12), with a clear distinction: In the binomial JUMP, we have $z_t \sim \text{Binomial}(x_0, \alpha_t)$, resulting in $z_t \leq x_0$, whereas in the Poisson JUMP, we have $z_t \sim \text{Pois}(\lambda \alpha_t x_0)$, allowing z_t to potentially exceed x_0.