Piecewise Diffusive Score-based Generative Model

Master's thesis in Master Data science and AI

Qiba Lu

DEPARTMENT OF COMPUTER SCIENCE AND ENGINEERING

CHALMERS UNIVERSITY OF TECHNOLOGY

Gothenburg, Sweden 2025

www.chalmers.se

Master's thesis 2025

Diffusion-based generative model with Jump

Qiba Lu

CHALMERS
UNIVERSITY OF TECHNOLOGY

Department of Computer Science and Engineering
Chalmers University of Technology
Gothenburg, Sweden 2025

Supervisor: Akash Sharma, Department of Mathematical Sciences
Examiner: Axel Ringh, Department of Mathematical Sciences

Master's Thesis 2025
Department of Computer Science and Engineering
Chalmers University of Technology
SE-412 96 Gothenburg
Telephone +46 31 772 1000

Cover: Floral dataset used for training the diffusion models.

Typeset in LaTeX
Printed by Chalmers Reproservice
Gothenburg, Sweden 2025

Abstract

Diffusion-based generative models have achieved remarkable success across a range of applications. However, their performance often degrades in the presence of imbalanced data. To address this limitation, we introduce a novel generative modeling framework that combines stochastic differential equation (SDE) driven by Brownian noise with Poisson random measures. We also provide the explicit form of reverse SDE, with corresponding loss function to train the neural network. Our experimental results show that the model with Poisson jump outperforms the model without jump across various toy datasets and a small subset of CIFAR-10.

Keywords: diffusion model, generative model, stochastic differential equations, random measures.

Acknowledgements

This thesis was supervised by Akash Sharma and examined by Axel Ringh.

Qiba Lu, Gothenburg, June 2025

Contents

1

Introduction

1.1 Motivation

The motivation of this thesis is the application of diffuion process in generative modeling. Generative AI has advanced rapidly since Kingma and Welling (2014) introduced Variational Autoencoders (VAEs) which are latent variable models for data synthesis. Subsequently, Generative Adversarial Networks (GANs) (Goodfellow et al., 2014) enhanced the quality and realism of generated samples across various domains. These advancements facilitated the development of more sophisticated generative models, addressing challenges in data diversity and fidelity. Moreover, GANs sparked significant interest in adversarial training, influencing applications like image and text synthesis.

Diffusion-based generative models (Sohl-Dickstein et al., 2015) have revolutionized generative modeling, delivering state-of-the-art performance in diverse applications, including video generation, image synthesis, audio generation, and image editing (Blattmann et al., 2023; Ho et al., 2022; Kong et al., 2021; Saharia et al., 2022; Liu et al., 2022; Meng et al., 2022). These models can be formulated as stochastic differential equations (SDE) which, driven by Brownian motion, progressively corrupt data into a simple and tractable distribution. Through the corresponding reverse SDE, diffusion models reconstruct the data from this distribution. Compared to GANs (Goodfellow et al., 2014), diffusion models, including latent diffusion models, consistently outperform GANs in sample diversity, quality, and robustness on various datasets (Dhariwal and Nichol, 2021; Rombach et al., 2022).

Diffusion models and GANs often struggle on class-imbalanced datasets, leading to training instability and mode collapse, with sample quality and diversity degrading significantly (Ho et al., 2020; Arjovsky et al., 2017; Radford et al., 2015; Yoon et al., 2023; Deasy et al., 2021).

Building on diffusion models, we introduce a framework that combines SDE with Poisson jump, further generalizing the current SDE-based generative model. The addition of jumps in physical and biological models addresses the limitations of purely continuous frameworks by capturing sudden discontinuities (e.g., market crashes, genetic mutations) that diffusion processes cannot represent. For example, in finance, Merton's jump-diffusion model Merton (1976) integrates Poisson-driven jumps with Brownian motion to better price options during crises; in biology, jump processes model rare but critical events like cell-state transitions Gao et al. (2022); in physics,

jumps are leveraged to describe atomic diffusion in crystals (via intermittent lattice jumps) Chudley and Elliott (1961); and in optimization improved performance is observed for certain test functions by adding jumps Kalise et al. (2023). In this thesis, we show that generative diffusion models with jumps help in dealing with the issue of mode collapse in imbalanced data-set.

1.2 Overview

This thesis investigates the theoretical and practical aspects of score-based diffusion models with jump. It examines the impact of jump on several toy datasets and a subset of the CIFAR-10 dataset. In Chapter 2, we provide a background on measure theory and key concepts from the diffusion model literature. Chapter 3 discusses the challenges diffusion models (Ho et al., 2020; Sohl-Dickstein et al., 2015; Song et al., 2021b) face with class-imbalanced datasets, demonstrated through a toy dataset, and presents the forward SDE, reverse SDE, and training objective of our proposed model. We then briefly introduce related work. In Chapter 4, we evaluate our model's performance on several toy datasets and a subset of the CIFAR-10 dataset. Finally, we summarize the findings and contributions of this thesis in the conclusion.

Our work highlights the potential of jump to improve generative modeling under challenging conditions. This research also lays the groundwork for future studies on hybrid diffusion frameworks.

1.3 Notations

Below are the notations that have been used throughout this thesis.

$\|\cdot\|$	L_2-norm of a vector
∇	Gradient operator
$\nabla\cdot$	Divergence operator
Δ	Laplacian operator
$\mathrm{Tr}(\cdot)$	Trace operator
I_d	Identity matrix with dimension d
$\mathcal{N}$	Normal distribution
Pois	Poisson distribution
$\overset{D}{=}$	Equality in distribution
$C^\infty(\mathbb{R}^d)$	Space of infinitely differentiable functions on $\mathbb{R}^d$.

2

Background

Diffusion-based generative models gained prominence following the introduction of Denoising Diffusion Probabilistic Models (DDPM) (Ho et al., 2020), while Song et al. (2021b) provided a continuous score-based diffusion model using stochastic differential equations (SDE). In Section 2.1, we present the theoretical framework of continuous score-based diffusion models, and we mention the related works to our contribution in Section 2.2. Finally, we outline the evaluation metrics used to assess model performance.

2.1 Forward and Reverse SDE

Stochastic differential equation (SDE) forms the mathematical foundation of score-based diffusion models, where the SDE describes the forward process of adding noise to data and its reverse, which uses the score function to generate new samples by progressively removing noise. Specifically, let t vary over the interval $[0, T]$. We consider a stochastic process $X(t) \in \mathbb{R}^d$ and work on a filtered probability space $(\Omega, \mathcal{F}, (\mathcal{F}_t)_{t\in[0,T]}, \mathbb{P})$, and let $W(t)$ be a d-dimensional Wiener Process adapted to $\mathcal{F}_t$. Suppose $X(t)$ satisfies the stochastic differential equation

$$\begin{cases} dX(t) = b(t, X(t))dt + \sigma(t)dW(t), & t \in [0, T], \\ X(0) \sim p(0). \end{cases} \tag{2.1}$$

where $b : [0, T] \times \mathbb{R}^d \to \mathbb{R}^d$ is the drift function, $\sigma : [0, T] \to \mathbb{R}$ is the diffusion coefficient, and $p(0)$ is the data distribution p_{data}. We design that by time T, the distribution of $X(T)$ is approximately distributed as some tractable distribution for example a multivariate normal $\mathcal{N}(0, \sigma^2 I_d)$. Under standard Lipschitz continuity and growth conditions on b, existence and uniqueness of solutions to this SDE are guaranteed (Applebaum, 2009). Given a new stochastic process $Y(t) \overset{D}{=} X(T - t)$, $t \in [0, T]$, also taking values in $\mathbb{R}^d$, we then have $Y(0) \overset{D}{=} X(T)$ and $Y(T) \overset{D}{=} X(0) \sim p(0) := p_{\text{data}}$. Since $X(T)$ is approximately distributed as $\mathcal{N}(0, I_d)$, we set $Y(0) \sim \mathcal{N}(0, I_d)$. If a stochastic process $X(t)$ satisfies a forward SDE (2.1), the stochastic process $Y(t)$ is itself a solution satisfying following SDE (Anderson, 1982):

$$dY(t) = -b(T - t, Y(t))\, dt + \sigma^2(T - t)\nabla_y \log p_{T-t}(Y(t))\, dt + \sigma(T - t)\, d\overline{W}(t). \tag{2.2}$$

$\overline{W}(t) \in \mathbb{R}^d$ is another Wiener process. The gradient of the logarithm of the time-marginal density $p_{T-t}(y)$, evaluated at $y = Y(t)$, is denoted $\nabla_y \log p_{T-t}(Y(t))$. For a fixed time $t \in [0, T]$, the stochastic process $X(t) \in \mathbb{R}^d$ is a random variable. Its distribution, denoted p_t, is referred to as the time-marginal distribution of the process at time t, and $\nabla_y \log p_{T-t}(y)$ is referred to as score function. The score function is typically intractable, so we learn a neural network $s_\theta(t, x)$ to approximate it, as in rest of the score-based diffusion models (Song et al., 2021b; Karras et al., 2022; Dockhorn et al., 2022).

A fundamental principle of diffusion-based generative models (Sohl-Dickstein et al., 2015; Ho et al., 2020) is the formulation of a forward Markov process that progressively perturbs samples drawn from an empirical data distribution, $p(0)$, transforming them into structured noise. In the continuous-time framework (Karras et al., 2022; Song et al., 2021b), this process is governed by the forward SDE (2.1), which models the gradual corruption of data. The reverse SDE (2.2) defines a stochastic process that inverts this transformation (Haussmann and Pardoux, 1986; Anderson, 1982), enabling the recovery of high-fidelity samples from noise through learned score functions. To simulate Eq. (2.2), the **Euler-Maruyama** method is considered. Discretize $[0, T]$ into N steps of size $\delta t = \frac{T}{N}$. Let $t_n = n\delta t$, $n = 0, 1, 2, ..., N$. The **Euler–Maruyama** scheme is defined by the following recursive formula:

$$\begin{aligned} Y_{n+1} = Y_n - b(T - t_n, Y_n)\,\delta t + \sigma^2(T - t_n)\nabla_x \log p_{T-t_n}(Y_n)\,\delta t \\ + \sigma(T - t_n)\,\delta\overline{W}_n. \end{aligned} \tag{2.3}$$

Similar to the reverse SDE (2.2), the $\overline{W}_n$ here is another Wiener process, and $\delta\overline{W}_n \sim \mathcal{N}(0, \delta t)$. As $\delta t \to 0$, these discrete approximations converge to the respective continuous time processes in some sense, providing a practical way to simulate both the forward "noising" direction and the reverse "denoising" direction.

2.2 Related Work

This section reviews key advancements in diffusion-based generative models. In Subsection 2.2.1, we mention the contributions of Song et al. (2021b), who introduced score-based diffusion models using SDE, and derive their relationship to the discrete-time framework of Denoising Diffusion Probabilistic Models (DDPM). Finally, we discuss recent extensions that incorporate the SDE with jump (Yoon et al., 2023; Popov et al., 2025; Deasy et al., 2021; Nachmani et al., 2021) in Subsection 2.2.2.

2.2.1 Diffusion-based Generative Models

2.2.1.1 Variance Preserving Stochastic Differential Equation

We first mention the score-based diffusion models proposed by Song et al. (2021b). In the forward SDE (2.1), $b(t, x) := -\frac{1}{2}\beta(t)x$ and $\sigma(t) := \sqrt{\beta(t)}$, and therefore forward process is given by

$$dX(t) = -\frac{1}{2}\beta(t)X(t)dt + \sqrt{\beta(t)}dW(t), \quad X(0) \sim p_{\text{data}}, \tag{2.4}$$

where $\beta : [0, T] \to \mathbb{R}_+$ is a time-scheduler, and the solution is

$$X(t) = e^{-\frac{1}{2}\int_0^t \beta(s)\,ds} X(0) + \sqrt{1 - e^{-\int_0^t \beta(s)\,ds}}\epsilon, \quad \epsilon \sim \mathcal{N}(0, I_d). \tag{2.5}$$

For simplicity, Song et al. (2021b) defined $\alpha(t) \coloneqq e^{-\int_0^t \beta(s)ds}$. This gives

$$X(t) = \sqrt{\alpha(t)}X(0) + \sqrt{1 - \alpha(t)}\epsilon, \quad \epsilon \sim \mathcal{N}(0, I_d). \tag{2.6}$$

Song et al. (2021b) refers to this type of SDE as Variance Preserving SDE (VPSDE). The corresponding reverse SDE is given by

$$dY(t) = \left[\frac{Y(t)}{2} + \nabla_y \log p_{T-t}(Y(t))\right]\beta(T-t)dt + \sqrt{\beta(T-t)}d\overline{W}(t), \tag{2.7}$$

where $\overline{W}(t)$ is another Wiener process in $\mathbb{R}^d$, and $Y(t) \overset{D}{=} X(T-t)$, $t \in [0, T]$.

2.2.1.2 Training objective

In the VPSDE case, notice that there is an intractable term $\nabla_y \log p_{T-t}(Y(t))$ in reverse SDE (2.7). This is referred as score function Song et al. (2021b). We approximate $\nabla_y \log p_{T-t}(y)$ using the neural network $s_\theta(t, x)$ trained as $s_\theta(t, x) \approx \nabla_x \log p_t(x)$, such that

$$s_\theta(T - t, y) \approx \nabla_y \log p_{T-t}(y).$$

Given the initial condition $X(0)$, VPSDE admits a closed-form distribution (Applebaum, 2009; Song et al., 2021b). This enables training the neural network $s_\theta(t, x)$ with the conditional score function $\nabla_x \log p_t(x|X(0))$, so we consider the loss function defined as

$$\mathcal{L}(\theta) \coloneqq \int_0^T c(t)\mathbb{E}_{X(t)}\left[\|s_\theta(t, X(t)) - \nabla_x \log p_t(X(t))\|^2\right] dt. \tag{2.8}$$

Given that

$$\begin{aligned}
\nabla_x \log p_t(x) &= \frac{\nabla_x p_t(x)}{p_t(x)} \\
&= \int_{\mathbb{R}^d} \nabla_x p_t(x \mid x_0)\frac{\mu_0(dx_0)}{p_t(x)} \\
&= \int_{\mathbb{R}^d} \nabla_x \log p_t(x \mid x_0)\frac{p_t(x \mid x_0)\mu_0(dx_0)}{p_t(x)} \\
&= \int_{\mathbb{R}^d} \nabla_x \log p_t(x \mid x_0)\mu_0(dx_0 \mid x) \\
&= \mathbb{E}_{X(0)|X(t)=x}\left[\nabla_x \log p_t(x \mid X(0))\right],
\end{aligned} \tag{2.9}$$

substitution into the loss function (2.8) yields

$$\mathcal{L}(\theta) = \int_0^T c(t)\mathbb{E}_{X(t)}\left\{\left\|s_\theta(t, X(t)) - \mathbb{E}_{X(0)|X(t)}[\nabla_x \log p_t(X(t)|X(0))]\right\|^2\right\} dt. \tag{2.10}$$

Since

$$\begin{aligned}
&\left\| s_\theta(t, X(t)) - \mathbb{E}_{X(0)|X(t)}[\nabla_x \log p_t(X(t)|X(0))] \right\|^2 \\
&= \|s_\theta(t, X(t))\|^2 - 2s_\theta(t, X(t)) \cdot \mathbb{E}_{X(0)|X(t)}[\nabla_x \log p_t(X(t)|X(0))] \\
&\quad + \left\| \mathbb{E}_{X(0)|X(t)}[\nabla_x \log p_t(X(t)|X(0))] \right\|^2, \qquad (2.11)
\end{aligned}$$

our loss function becomes

$$\begin{aligned}
\mathcal{L}(\theta) = \int_0^T c(t)\mathbb{E}_{X(t)}\Big\{ &\|s_\theta(t, X(t))\|^2 \\
&- 2s_\theta(t, X(t)) \cdot \mathbb{E}_{X(0)|X(t)} \left[\nabla_x \log p_t(X(t)|X(0))\right] \Big\} dt + C, \qquad (2.12)
\end{aligned}$$

where C is a constant here. Utilizing the properties of expectation, we have

$$\begin{aligned}
\mathcal{L}(\theta) &= \int_0^T c(t)\mathbb{E}_{X(t)}\Big\{ \|s_\theta(t, X(t))\|^2 \\
&\quad - 2\mathbb{E}_{X(0)|X(t)}\Big[s_\theta(t, X(t)) \cdot \nabla_x \log p_t(X(t)|X(0)) \Big] \Big\} dt \qquad (2.13) \\
&= \int_0^T c(t)\mathbb{E}_{X(t)}\Big\{ \mathbb{E}_{X(0)|X(t)}\Big[\|s_\theta(t, X(t))\|^2 \\
&\quad - 2s_\theta(t, X(t)) \cdot \nabla_x \log p_t(X(t)|X(0)) \Big] \Big\} dt. \qquad (2.14)
\end{aligned}$$

The law of total expectation yields

$$\begin{aligned}
\mathcal{L}(\theta) &= \int_0^T c(t)\mathbb{E}_{X(t),X(0)}\Big[\|s_\theta(t, X(t))\|^2 \\
&\quad - 2s_\theta(t, X(t)) \cdot \nabla_x \log p_t(X(t)|X(0)) \Big] dt \qquad (2.15) \\
&= \int_0^T c(t)\mathbb{E}_{X(t),X(0)}\left[\|s_\theta(t, X(t)) - \nabla_x \log p_t(X(t)|X(0))\|^2 \right] dt. \qquad (2.16)
\end{aligned}$$

which is recognized as Denoising Score Matching (DSM) (Vincent, 2011). Recall that Song et al. (2021b) proposed $b(t, X(t)) \coloneqq -\frac{1}{2}\beta(t)X(t)$, and $\sigma(t) \coloneqq \sqrt{\beta(t)}$, with the conditional distribution

$$X(t)|X(0) \sim \mathcal{N}\Big(\sqrt{\alpha(t)}X(0), (1 - \alpha(t))I_d \Big),$$

we then have the conditional density

$$p_t(x|X(0)) = \frac{\exp\left(-\frac{1}{2(1-\alpha(t))} \left\| x - \sqrt{\alpha(t)}X(0) \right\|^2 \right)}{\left[2\pi\left(1 - \alpha(t)\right)\right]^{d/2}}, \qquad (2.17)$$

and this gives the gradient of the density

$$\nabla_x \log p_t(x|X(0)) = -\frac{x - \sqrt{\alpha(t)}X(0)}{1-\alpha(t)}. \tag{2.18}$$

Since

$$X(t) = \sqrt{\alpha(t)}X(0) + \sqrt{1-\alpha(t)}\,\epsilon,\ \epsilon \sim \mathcal{N}(0, I_d),$$

we evaluate $\nabla_x \log p_t(x \mid X(0))$ at $x = X(t)$, obtaining

$$\nabla_x \log p_t(x \mid X(0))\Big|_{x=X(t)} = -\frac{1}{\sqrt{1-\alpha(t)}}\epsilon.$$

Following the setting in Song et al. (2021b), we defined

$$\epsilon_\theta(t, X(t)) \coloneqq -\sqrt{1-\alpha(t)}\, s_\theta(t, X(t)),$$

and let

$$c(t) \propto \frac{1}{\mathbb{E}_{X(t),X(0)}\left[\|\nabla_x \log p_t(X(t) \mid X(0))\|^2\right]} \propto 1-\alpha(t).$$

Substituting into the loss function (2.16) yields

$$\mathcal{L}(\theta) = \int_0^T (1-\alpha(t))\mathbb{E}_{X(t),X(0)}\left[\left\|-\frac{\epsilon_\theta(t, X(t)) + \epsilon}{\sqrt{1-\alpha(t)}}\right\|^2\right] dt + C \tag{2.19}$$

$$= \int_0^T \mathbb{E}_{X(t),X(0)}\left[\|\epsilon_\theta(t, X(t)) - \epsilon\|^2\right] dt. \tag{2.20}$$

2.2.1.3 Denoising Diffusion Probabilistic Models

While DDPM pioneered practical diffusion models, VPSDE provides a continuous-time perspective on the diffusion models. In this section, we show the connections between VPSDE and DDPM. Recall that Song et al. (2021a) gives the solution of the forward SDE (2.1):

$$X(t) = \sqrt{\alpha(t)}X(0) + \sqrt{1-\alpha(t)}\epsilon, \quad \epsilon \sim \mathcal{N}(0, I_d), \tag{2.21}$$

where $\alpha(t) = e^{-\int_0^t \beta(s)ds}$, and $\beta : [0, T] \to \mathbb{R}_+$ is a time-scheduler. In its discretized form, this becomes

$$X_n = \sqrt{\alpha_n}X(0) + \sqrt{1-\alpha_n}\epsilon, \quad \epsilon \sim \mathcal{N}(0, I_d), \tag{2.22}$$

where

$$\alpha_n = \prod_{i=0}^{n-1}(1-\beta_i\delta t), \quad \beta_i = \beta(i\delta t),$$

so $X_n \sim \mathcal{N}(\sqrt{\alpha_n}X(0), (1-\alpha_n)I_d)$, which is the Eq. (4) in DDPM (Ho et al., 2020). Define that

$$\epsilon_\theta(t, X(t)) \coloneqq -\sqrt{1-\alpha(t)}\, s_\theta(t, X(t)),$$

this gives

$$\nabla_y \log p_{T-t}(Y_n) \approx -\frac{\epsilon_\theta(T-t_n, Y_n)}{\sqrt{1-\alpha_{N-n}}}.$$

Substitute $b(T-t_n, Y_n) = -\frac{1}{2}\beta_{N-n}Y_n$ and $\sigma(T-t_n) = \sqrt{\beta_{N-n}}$ into Eq. (2.3), we then have

$$Y_{n+1} = Y_n + \left[\frac{1}{2}Y_n - \frac{\epsilon_\theta(T-t_n, Y_n)}{\sqrt{1-\alpha_{N-n}}}\right]\beta_{N-n}\delta t + \sqrt{\beta_{N-n}\delta t}\,\epsilon, \tag{2.23}$$

and note that Eq. (2.23) corresponds to the sampling algorithm in DDPM (Ho et al., 2020).

2.2.2 Diffusion-based generative models with Jump

Deasy et al. (2021); Nachmani et al. (2021) explored non-Brownian noise to model heavy-tailed distributions underlying data, but their approaches did not provide a reverse SDE. Yoon et al. (2023) introduces α-stable isotropic Lévy noise and employs a reverse SDE driven by Lévy processes. Specifically, they proposed the forward SDE:

$$dX(t) = b(t, X(t))dt + \sigma(t)dL_t^\alpha,$$

with L_t^α representing an isotropic α-stable Lévy process, characterized by exponent $\alpha \in (1, 2]$. The corresponding reverse SDE is:

$$dY(t) = \left(-\frac{\beta(T-t)}{\alpha}Y(t) - \alpha\beta(T-t)S_t^{(\alpha)}(Y(t))\right)dt + \beta(T-t)^{\frac{1}{2}}d\bar{L}_t^\alpha,$$

with $\bar{L}_t^\alpha$ denoting the reverse Lévy process and $S_t^{(\alpha)}(x) = \frac{\Delta^{\frac{-\alpha}{2}}\nabla p_t(x)}{p_t(x)}$ representing the fractional score function, defined via the fractional Laplacian $\Delta^{\frac{-\alpha}{2}}$. Notably, the α-stable process exhibits infinite activity, that is, it permits an infinite number of jump per unit time. This yields a reverse SDE driven by infinite-activity Lévy noise. Furthermore, the time-reversed dynamics involve an intractable data-dependent stochastic process, which the authors propose to ignore, resulting in approximate reverse dynamics. Popov et al. (2025) proposed a reparameterized reverse SDE for Yoon et al. (2023)'s model. Their approach introduces a parametric family of reverse SDE:

$$dY(t) = \left(-\frac{\beta(T-t)}{\alpha}Y(t) - (1+\zeta_{T-t})\beta(T-t)^{\frac{\alpha}{2}}S_{T-t}^{(\alpha)}(Y(t))\right)dt + \beta(T-t)^{\frac{1}{2}}\zeta_{T-t}^{1/\alpha}d\bar{L}_t^\alpha,$$

where $\zeta_t \geq 0$ is a time-dependent parameter controlling the amount of noise added at each reverse step.

Our main contribution is that we provided a continuous-time framework of generative models based on stochastic differential equations driven by Poisson random measure with finite jump, so that we have a explicit form of reverse SDE, this enables us to simulate and compute easier, with tractable sampling and score functions.

2.3 Evaluation Metric

The Euler-Maruyama scheme discretizes the reverse SDE into $N = \frac{T}{\delta t}$ steps, which is called the total number of score function evaluations (NFE). We use two metrics to evaluate the performance of the models in this thesis.

2.3.1 Maximum Mean Discrepancy

The Maximum Mean Discrepancy (MMD) measures the difference between two probability distributions. Given two sets of samples $X = \{x_1, \ldots, x_m\} \sim P$ and $Y = \{y_1, \ldots, y_n\} \sim Q$, we compute the MMD using a mixture of rational kernels. The kernel function is defined as

$$k(x, y) \coloneqq \sum_{b \in B} \frac{b^2}{b^2 + \|x - y\|^2},$$

where B is a set of bandwidth parameters. The squared MMD is then

$$\begin{aligned}\mathrm{MMD}^2(X, Y) = &\frac{1}{m^2} \sum_{i=1}^{m} \sum_{j=1}^{m} \sum_{b \in B} \frac{b^2}{b^2 + \|x_i - x_j\|^2} + \frac{1}{n^2} \sum_{i=1}^{n} \sum_{j=1}^{n} \sum_{b \in B} \frac{b^2}{b^2 + \|y_i - y_j\|^2} \\ &- \frac{2}{mn} \sum_{i=1}^{m} \sum_{j=1}^{n} \sum_{b \in B} \frac{b^2}{b^2 + \|x_i - y_j\|^2}.\end{aligned}$$

A smaller MMD indicates greater similarity between the true and generated distributions.

2.3.2 Fréchet Inception Distance

The Fréchet Inception Distance (FID) (Heusel et al., 2017) measures the similarity between two distributions by comparing features extracted from a pre-trained Inception V3 model. Given two sets of samples $X = \{x_1, \ldots, x_m\} \sim P$ and $Y = \{y_1, \ldots, y_n\} \sim Q$, features are approximated as multivariate Gaussians $\phi(X) \sim \mathcal{N}(\mu_X, \Sigma_X)$ and $\phi(Y) \sim \mathcal{N}(\mu_Y, \Sigma_Y)$, where μ_X, μ_Y are the mean feature vectors, and Σ_X, Σ_Y are the covariance matrices. The FID is

$$\mathrm{FID}(X, Y) = \|\mu_X - \mu_Y\|_2^2 + \mathrm{Tr}\left(\Sigma_X + \Sigma_Y - 2(\Sigma_X \Sigma_Y)^{\frac{1}{2}}\right),$$

where $\mathrm{Tr}(\cdot)$ is the trace operator. A lower FID indicates that the generated samples are more similar to the true data in the feature space.

3

Diffusion Models with Jump

This chapter introduces the proposed diffusion models with jump. We first discuss limitations of current diffusion models, such as mode collapse in Section 3.1. We then provide a detailed introduction to the theoretical framework in Section 3.2, including the forward SDE, reverse SDE, and the training objective.

3.1 Limitation of Traditional Diffusion Models

Class-imbalanced datasets often lead to mode collapse in generative models, where the model generates a limited set of outputs in the training data. This issue causes current diffusion models (Ho et al., 2020; Song et al., 2021b) to produce lower-quality and less varied samples (Shariatian et al., 2025; Yoon et al., 2023; Deasy et al., 2021). To illustrate this, we tested several diffusion models on three Gaussian mixture datasets, increasing the distance between modes, as shown in Figure 1. Figure 1 demonstrates the limitations of traditional diffusion models in handling class-imbalanced datasets, particularly as mode distances increase, leading to mode collapse and bias towards the dominant mode during sampling. Diffusion models with jump offer a promising approach to mitigate these limitations, enabling better exploration of the data space and improved capture of minority modes.

3.2 Stochastic Differential Equations with Jump

Consider a probability space $(\Omega, \mathcal{F}, \mathbb{P})$ equipped with a filtration $\{\mathcal{F}_t : t \in [0,T]\}$, generated by a d-dimensional Wiener process $W(t)$ and a Poisson random measure $N(dt, d\mathbf{z})$ with intensity measure $\nu(d\mathbf{z})dt$, where $\nu(d\mathbf{z}) = \lambda\phi(\mathbf{z})d\mathbf{z}$. Here, $\phi(\mathbf{z})$ is the density of a standard Gaussian distribution $\mathcal{N}(0, I_d)$, and $\lambda > 0$ represents the jump intensity. We define the SDE with jump as

$$\begin{cases} dX(t) = b(t, X(t))\, dt + \sigma(t)\, dW(t) + \int_{\mathbb{R}^d} \eta(t)\mathbf{z}\, N(dt, d\mathbf{z}), \\ X(0) \sim p(0) = p_{\text{data}}, \quad Y(0) \sim X(T), \end{cases} \tag{3.1}$$

where $X(t) \in \mathbb{R}^d$ represents the state variable with initial condition following the data distribution p_{data}, $b : [0,T] \times \mathbb{R}^d \to \mathbb{R}^d$ is the drift function, $\sigma : [0,T] \to \mathbb{R}_+$ is the diffusion coefficient, and $\eta : [0,T] \to \mathbb{R}_+$ scales the jump amplitude.

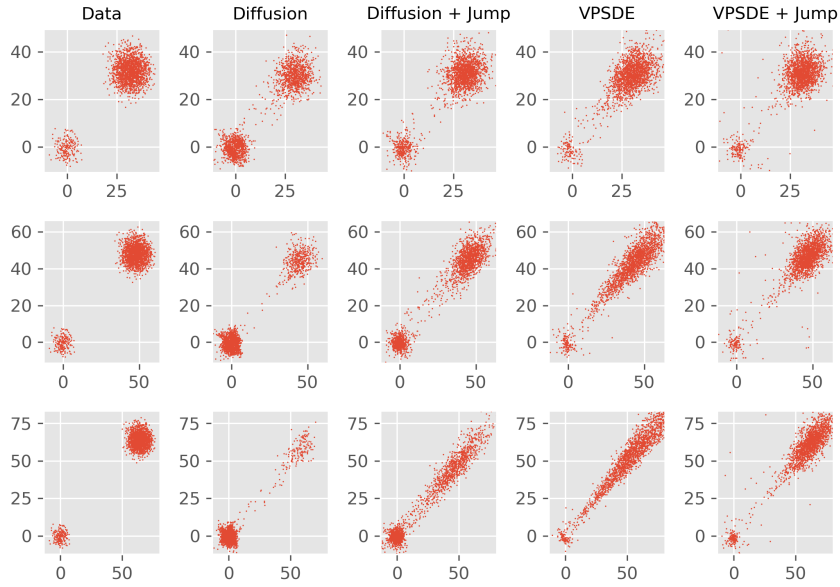

Figure 1: A Gaussian mixture dataset with two components, where the distance between component means increases progressively. The mean pairs are $[0, 0]$ and $[32, 32]$ (GM32), $[0, 0]$ and $[48, 48]$ (GM48), and $[0, 0]$ and $[64, 64]$ (GM64). The covariances are fixed at $10I_d$ and $20I_d$, respectively, with mixture weights in a $1 : 9$ ratio for the components. We give the detailed explanation of these dataset in 4.1.

3.2.1 Derivation of Reverse SDE

We now provide the formal derivation of the reverse SDE of our model. We assume that $X(t)$ has time-marginal density p_t and $\nu(d\mathbf{z})$ is finite and valid. The infinitesimal generator Applebaum (2009, pp. 399–404) is given by

$$\mathcal{A}\psi(x) = b(t,x)\cdot\nabla\psi(x) + \frac{1}{2}\sigma^2(t)\Delta\psi(x) + \int_{\mathbb{R}^d}\Big[\psi(x+\eta(t)\mathbf{z}) - \psi(x)\Big]\nu(d\mathbf{z}), \tag{3.2}$$

where ψ belongs to the domain of the generator. We have

$$\begin{aligned}\int_{\mathbb{R}^d}\mathcal{A}\psi(x)p_t(x)\,dx &= \int_{\mathbb{R}^d}\left[b(t,x)\cdot\nabla_x\psi(x)\right]p_t(x)\,dx + \frac{1}{2}\sigma^2(t)\int_{\mathbb{R}^d}\Delta\psi(x)p_t(x)\,dx \\ &\quad + \int_{\mathbb{R}^d}\int_{\mathbb{R}^d}\left[\psi(x+\eta(t)\mathbf{z}) - \psi(x)\right]\nu(d\mathbf{z})p_t(x)\,dx \qquad (3.3)\\ &= \underbrace{\sum_{i=1}^d\int_{\mathbb{R}^d} b_i(t,x)\frac{\partial\psi(x)}{\partial x_i}p_t(x)dx}_{I_1} + \frac{1}{2}\sigma^2(t)\underbrace{\sum_{i=1}^d\int_{\mathbb{R}^d}\frac{\partial^2\psi(x)}{\partial x_i^2}p_t(x)dx}_{I_2} \\ &\quad + \underbrace{\int_{\mathbb{R}^d}\int_{\mathbb{R}^d}[\psi(x+\eta(t)\mathbf{z}) - \psi(x)]\nu(d\mathbf{z})p_t(x)d(x)}_{I_3}. \qquad (3.4)\end{aligned}$$

For each element of the first term I_1, integrating by parts, we obtain

$$\int_{\mathbb{R}^d} b_i(t,x)\frac{\partial\psi(x)}{\partial x_i}p_t(x)dx_i \tag{3.5}$$

$$=[b_i(t,x)\psi(x)p_t(x)]_{x_i=-\infty}^{x_i=\infty} - \int_{-\infty}^{\infty}\psi(x)\frac{\partial}{\partial x_i}[b_i(t,x)p_t(x)]dx_i \tag{3.6}$$

$$= -\int_{-\infty}^{\infty}\psi(x)\frac{\partial}{\partial x_i}[b_i(t,x)p_t(x)]dx_i. \tag{3.7}$$

Note that $[b_i(t,x)\psi(x)p_t(x)]_{x_i=-\infty}^{x_i=\infty} = 0$ because of $\psi \in C_0^\infty(\mathbb{R}^d)$, thus

$$I_1 = -\int_{\mathbb{R}^d}\psi(x)\nabla_x\cdot[b(t,x)p_t(x)]\,dx. \tag{3.8}$$

$$I_2 = \int_{\mathbb{R}^d}\psi(x)\Delta p_t(x)\,dx, \tag{3.9}$$

and for the jump term I_3, since

$$\int_{\mathbb{R}^d}\int_{\mathbb{R}^d}[\psi(x+\eta(t)\mathbf{z})-\psi(x)]\nu(d\mathbf{z})p_t(x)dx$$

$$=\int_{\mathbb{R}^d}\int_{\mathbb{R}^d}\left[\int_0^1\nabla_x\psi(x+h\eta(t)\mathbf{z})\cdot(\eta(t)\mathbf{z})dh\right]\nu(d\mathbf{z})p_t(x)dx, \tag{3.10}$$

we set $x' = x + h\eta(t)\mathbf{z}$ and integrate by parts, obtaining

$$-\int_{\mathbb{R}^d}\psi(x')\left[\int_{\mathbb{R}^d}\int_0^1\nabla_{x'}\cdot(\mathbf{z}p_t(x'-h\eta(t)\mathbf{z})\eta(t))\,dh\,\nu(d\mathbf{z})\right]dx' \tag{3.11}$$

$$=-\int_{\mathbb{R}^d}\psi(x)\left[\eta(t)\int_{\mathbb{R}^d}\int_0^1\mathbf{z}^\top\nabla_x p_t(x-h\eta(t)\mathbf{z})\,dh\,\nu(d\mathbf{z})\right]dx. \tag{3.12}$$

We need to find the adjoint of $\mathcal{A}$, which we denote as $\mathcal{A}^*$ and it satisfies

$$\int(\mathcal{A}\psi)(x)p_t(x)d\mathbf{x} = \int\psi(x)(\mathcal{A}^*p_t)(x)dx. \tag{3.13}$$

From the derivation above, we now can conclude that

$$\int\psi(x)(\mathcal{A}^*p_t)(x)dx$$

$$=-\int_{\mathbb{R}^d}\psi(x)\nabla_x\cdot[b(t,x)p_t(x)]\,dx + \frac{\sigma^2(t)}{2}\int_{\mathbb{R}^d}\psi(x)\Delta p_t(x)\,dx$$

$$-\int_{\mathbb{R}^d}\psi(x)\left[\eta(t)\int_{\mathbb{R}^d}\int_0^1\mathbf{z}^\top\nabla_x p_t(x-h\eta(t)\mathbf{z})\,dh\,\nu(d\mathbf{z})\right]dx \tag{3.14}$$

The reverse SDE is a stochastic process $Y(t) \overset{D}{=} X(T-t)$, implying that $Y(0) \overset{D}{=} X(T)$ and $Y(T) \overset{D}{=} X(0)$, yielding

$$\frac{\partial p_{T-t}}{\partial t} = \frac{\partial p_{T-t}}{\partial(T-t)}\frac{d(T-t)}{dt} = -\frac{\partial p_{T-t}}{\partial(T-t)}, \tag{3.15}$$

so we have

$$\frac{\partial p_{T-t}}{\partial t} = \nabla_x \cdot (b(T-t,x)p_{T-t}) - \frac{\sigma^2(T-t)}{2}\Delta p_{T-t} + \eta(T-t)\int_{\mathbb{R}^d}\int_0^1 \mathbf{z}^\top \nabla_x (p_{T-t}(x - h\eta(T-t)\mathbf{z}])dh\nu(d\mathbf{z}). \tag{3.16}$$

Notice that

$$-\frac{\sigma^2(T-t)}{2}\Delta p_{T-t} = -\sigma^2(T-t)\Delta p_{T-t} + \frac{\sigma^2(T-t)}{2}\Delta p_{T-t}, \tag{3.17}$$

and

$$\begin{aligned} &\eta(T-t)\int_{\mathbb{R}^d}\int_0^1 \mathbf{z}^\top \nabla_x p_{T-t}(x - h\eta(T-t)\mathbf{z})dh\nu(d\mathbf{z}) \\ =&\frac{2\eta(T-t)p_{T-t}}{p_{T-t}}\int_{\mathbb{R}^d}\int_0^1 \mathbf{z}^\top \nabla_x p_{T-t}(x - h\eta(T-t)\mathbf{z})dh\nu(d\mathbf{z}) \\ &- \eta(T-t)\int_{\mathbb{R}^d}\int_0^1 \mathbf{z}^\top \nabla_x p_{T-t}(x - h\eta(T-t)\mathbf{z})dh\nu(d\mathbf{z}), \end{aligned} \tag{3.18}$$

we then have

$$\begin{aligned} \frac{\partial p_{T-t}}{\partial t} &= \nabla_x \cdot (b(T-t,x)p_{T-t}) - \sigma^2(T-t)\Delta p_{T-t} \\ &\quad + \frac{2\eta(T-t)p_{T-t}}{p_{T-t}}\int_{\mathbb{R}^d}\int_0^1 \mathbf{z}^\top \nabla_x p_{T-t}(x - h\eta(T-t)\mathbf{z})dh\nu(d\mathbf{z}) \\ &\quad + \frac{\sigma^2(T-t)}{2}\Delta p_{T-t} \\ &\quad - \eta(T-t)\int_{\mathbb{R}^d}\int_0^1 \mathbf{z}^\top \nabla_x p_{T-t}(x - h\eta(T-t)\mathbf{z})dh\nu(d\mathbf{z}) \end{aligned} \tag{3.19}$$

$$\begin{aligned} &= \nabla \cdot \left(\left[b(T-t,x) - \sigma^2(T-t)\nabla \log p_{T-t} \right.\right. \\ &\quad \left.\left. + \frac{2\eta(T-t)}{p_{T-t}}\int_{\mathbb{R}^d}\int_0^1 \mathbf{z}p_{T-t}(x - h\eta(T-t)\mathbf{z})dh\nu(d\mathbf{z}) \right] p_{T-t} \right) \\ &\quad + \frac{\sigma^2(T-t)}{2}\Delta p_{T-t} \\ &\quad - \eta(T-t)\int_{\mathbb{R}^d}\int_0^1 \mathbf{z}^\top \nabla_x p_{T-t}(x - h\eta(T-t)\mathbf{z})dh\nu(d\mathbf{z}). \end{aligned} \tag{3.20}$$

The above derivation helps us characterize the reverse SDE for the proposed piecewise-diffusive model (3.1):

$$\begin{aligned} dY(t) = &- \left[b(T-t,Y(t)) - \sigma^2(T-t)\nabla_y \log p_{T-t}(Y(t)) \right. \\ &\left. + \frac{2\eta(T-t)}{p_{T-t}(Y(t))}\int_{\mathbb{R}^d}\int_0^1 \mathbf{z}p_{T-t}(Y(t) - h\eta(T-t)\mathbf{z})dh\nu(d\mathbf{z}) \right] dt \\ &+ \sigma(T-t)d\overline{W}(t) + \int_{\mathbb{R}^d}\eta(T-t)\mathbf{z}N(dt,d\mathbf{z}). \end{aligned} \tag{3.21}$$

3.2.2 Training Objective

The reverse SDE (3.21) includes intractable terms in its drift term. For the score function $\nabla_y \log p_{T-t}(y)$, similar to the most diffusion-based model (Song et al., 2021b; Karras et al., 2022), we use a neural network $s_\theta(t,x)$ to approximate it. For simplicity and convenience, we use Taylor expansion to approximate the jump term. We define

$$S^\eta(t,x) \coloneqq \frac{1}{p_t(x)} \int_{\mathbb{R}^d} \int_0^1 \mathbf{z} p_t(x - h\eta(t)\mathbf{z}) dh\nu(d\mathbf{z}),$$

and, using a Taylor expansion, we obtain

$$p_t(x - h\eta(t)\mathbf{z}) \approx p_t(x) - h\eta(t)(\mathbf{z}^\top \nabla_x p_t(x)). \tag{3.22}$$

This yields

$$\begin{aligned}
S^\eta(t,x) &\approx \frac{1}{p_t(x)} \int_{\mathbb{R}^d} \int_0^1 \mathbf{z} p_t(x) dh\nu(d\mathbf{z}) \\
&\quad - \frac{1}{p_t(x)} \int_{\mathbb{R}^d} \int_0^1 h\eta(t)\mathbf{z}(\mathbf{z}^\top \nabla_x p_t(x) dh\nu(d\mathbf{z}) \qquad (3.23) \\
&= \frac{1}{p_t(x)} \int_0^1 dh \int_{\mathbb{R}^d} \mathbf{z} p_t(x) \nu(d\mathbf{z}) \\
&\quad - \frac{\eta(t)}{p_t(x)} \int_0^1 hdh \int_{\mathbb{R}^d} \mathbf{z}(\mathbf{z}^\top \nabla_x p_t(x) dh\nu(d\mathbf{z}) \qquad (3.24) \\
&= \mathbb{E}_{\mathbf{Z}}(\mathbf{Z}) - \frac{\lambda\eta(t)}{2p_t(x)} \mathbb{E}_{\mathbf{Z}}\Big[\mathbf{Z}(\mathbf{Z}^\top \nabla_x p_t(x))\Big] \qquad (3.25) \\
&= -\frac{\lambda\eta(t)}{2} \nabla_x \log p_t(x), \qquad (3.26)
\end{aligned}$$

and

$$S^\eta(T-t,y) = -\frac{\lambda\eta(T-t)}{2} \nabla_y \log p_{T-t}(y). \tag{3.27}$$

This gives an approximate reverse SDE with jump

$$\begin{aligned}
dY(t) =& \Big\{ - b(T-t, Y(t)) + [\sigma^2(T-t) + \lambda\eta^2(T-t)] \nabla_y \log p_{T-t}(Y(t)) \Big\} dt \\
&+ \sigma(T-t) d\overline{W}(t) + \int_{\mathbb{R}^d} \eta(T-t)\mathbf{z} N(dt, d\mathbf{z}).
\end{aligned} \tag{3.28}$$

We use the neural network $s_\theta(t,x)$ to further approximate $[\sigma^2(t)+\lambda\eta^2(t)]\nabla_x \log p_t(x)$ by using the following score-matching loss function:

$$\begin{aligned}
\mathcal{L}(\theta) &\coloneqq \int_0^T \mathbb{E}_{X(t)} \Big[\Big\| s_\theta(t, X(t)) - [\sigma^2(t) + \lambda\eta^2(t)] \nabla_x \log p_t(X(t)) \Big\|^2 \Big] dt \qquad (3.29) \\
&= \int_0^T \mathbb{E}_{X(t)} \Big[\Big\| s_\theta(t, X(t)) \Big\|^2 + 2[\sigma^2(t) + \lambda\eta^2(t)] \nabla_x \cdot s_\theta(t, X(t)) \Big] dt + C, \qquad (3.30)
\end{aligned}$$

where C is a constant. Directly computing the $\nabla_x \cdot s_\theta(t,x)$ is expensive. To address this, Song et al. (2019) introduced Sliced Score Matching (SSM) for approximating this term. SSM leverages the Hutchinson trace estimator (Hutchinson, 1990),

which uses random projections to estimate the trace of $s_\theta(t,x)$. Specifically, SSM approximates $\nabla_x \cdot s_\theta(t,x)$ with $\zeta^\top \nabla s_\theta(t,x)\zeta$, where $\zeta \sim \mathcal{N}(0, I_d)$. Using the same approximation in our setting, this results

$$\mathcal{L}(\theta) = \int_0^T \mathbb{E}_{X(t),\zeta}\left[\left\|s_\theta(t,X(t))\right\|^2 + 2[\sigma^2(t) + \lambda\eta^2(t)]\zeta^\top \nabla_x s_\theta(t,X(t))\zeta\right] dt. \quad (3.31)$$

With this approximation, the reverse SDE (3.28) becomes

$$\begin{aligned} dY(t) =& \Big[-b(T-t,Y(t)) + s_\theta(T-t,Y(t))\Big]dt + \sigma(T-t)d\overline{W}(t) \\ &+ \int_{\mathbb{R}^d} \eta(T-t)\mathbf{z}N(dt,d\mathbf{z}). \end{aligned} \quad (3.32)$$

3.3 Two examples of piecewise-diffusive models

We consider two specific cases of the model (3.1) throughout this thesis, and we provide the corresponding forward SDE, reverse SDE, and loss function in the following subsections

3.3.1 Diffusion with Jump

Here, we set $b(t,x) := -ax$, with constant $\sigma(t) \equiv \sigma$ and $\eta(t) \equiv \eta$. Substituting into the forward SDE (3.1), we obtain

$$dX(t) = -aX(t)\,dt + \sigma\,dW(t) + \int_{\mathbb{R}^d} \eta\mathbf{z}\,N(dt,d\mathbf{z}), \quad (3.33)$$

where $X(t) \in \mathbb{R}^d$, $a > 0$, $\sigma > 0$, and $\eta > 0$ are constants, and $W(t)$ is a d-dimensional Brownian motion. Applebaum (2009, pp. 363–377) provides the solution

$$X(t) = e^{-at}X(0) + \sigma\int_0^t e^{-a(t-s)}dW(s) + \eta\int_0^t\int_{\mathbb{R}^d} e^{-a(t-s)}\mathbf{z}\,N(ds,d\mathbf{z}). \quad (3.34)$$

As the Itô integral with a deterministic integrand follows a normal distribution, we obtain

$$\sigma\int_0^t e^{-a(t-s)}dW(s) \sim \mathcal{N}\Big(0, \sigma^2\int_0^t e^{-2a(t-s)}ds I_d\Big),$$

and due to the properties of the compound Poisson process, we have

$$\eta\int_0^t\int_{\mathbb{R}^d} e^{-a(t-s)}\mathbf{z}\,N(ds,d\mathbf{z}) = \eta\sum_{i=1}^{N(t)} e^{-a(t-\tau_i)}\mathbf{z}_i, \quad (3.35)$$

where $\{\tau_i\}$ are the jump times in $[0,t]$, $\{N(t)\}_{t\geq 0}$ is a Poisson process, and $N(t)$ is the number of jumps in $[0,t]$. For simulation purposes, we use

$$X(t) = e^{-at}X(0) + \sigma\sqrt{\frac{1-e^{-2at}}{2a}}\,\epsilon + \eta\sum_{i=1}^{N(t)} e^{-a(t-\tau_i)}\mathbf{z}_i \quad (3.36)$$

to add noise to the data, where $\epsilon \sim \mathcal{N}(0, I_d)$, and $\mathbf{z}_i \in \mathbb{R}^d$ are the jump sizes at times τ_i. Substituting $b(t,x) = -ax$, $\sigma(t) = \sigma$, and $\eta(t) = \eta$ into the reverse SDE (3.21), we obtain

$$dY(t) = \Big[aY(t) + s_\theta(T-t, Y(t))\Big]dt + \sigma d\overline{W}(t) + \int_{\mathbb{R}^d} \eta \mathbf{z} N(dt, dz). \tag{3.37}$$

The corresponding Euler-Maruyama scheme is given by

$$Y_{n+1} = \begin{cases} Y_n + [aY_n + s_\theta(T-t_n, Y_n)]\delta t \\ +\sigma\delta\overline{W}_n, \quad \text{with probability } 1-\lambda\delta t, \\ Y_n + [aY_n + s_\theta(T-t_n, Y_n)]\delta t \\ +\sigma\delta\overline{W}_n, +\eta\mathbf{z}_n \quad \text{with probability } \lambda\delta t. \end{cases} \tag{3.38}$$

Substituting into the loss function (3.31), we obtain the loss function for training the neural network $s_\theta(t, X(t))$:

$$\mathcal{L}(\theta) = \int_0^T \mathbb{E}_{X(t),\zeta}\Big[\Big\|s_\theta(t, X(t))\Big\|^2 + 2(\sigma^2 + \lambda\eta^2)\zeta^\top \nabla_x s_\theta(t, X(t))\zeta\Big]dt.$$

We refer to this model as "Diffusion + Jump" model in our experiments.

3.3.2 VPSDE with Jump

Song et al. (2021b) proposed $b(t,x) := -\frac{1}{2}\beta(t)x$ and $\sigma(t) := \sqrt{\beta(t)}$, where $\beta(t)$ is a time-scheduler. The solution of VPSDE (Song et al., 2021b) is

$$X(t) = \sqrt{\alpha(t)}X(0) + \sqrt{1-\alpha(t)}\epsilon,$$

where $\epsilon \sim \mathcal{N}(0, I_d)$. By incorporating jump into VPSDE, we set $\sigma(t) = \eta(t) = \sqrt{\beta(t)}$ and keep $b(t,x) = -\frac{1}{2}\beta(t)x$. The solution becomes

$$X(t) = \sqrt{\alpha(t)}X(0) + \sqrt{1-\alpha(t)}\epsilon + \sqrt{\alpha(t)}\sum_{i=1}^{N(t)} \sqrt{\frac{\beta(\tau_i)}{\alpha(\tau_i)}}\mathbf{z}_i, \tag{3.39}$$

where $\epsilon \sim \mathcal{N}(0, I_d)$. The settings are the same in 3.3.1, $\{\tau_i\}$ are the jump times in $[0,t]$, $\{N(t)\}_{t\geq 0}$ is a Poisson process, $N(t)$ is the number of jumps in $[0,t]$ and $\mathbf{z}_i \in \mathbb{R}^d$ are the jump sizes at times τ_i. We use a linear time-scheduler $\beta(t) := (\beta_{\max} - \beta_{\min})t + \beta_{\min}$ in our experiments, with chosen values presented in Section 4.1. Substituting into the reverse SDE (3.21) yields

$$\begin{aligned} dY(t) = &\left[\frac{1}{2}\beta(T-t)Y(t) + s_\theta(T-t, Y(t))\right]dt \\ &+ \sqrt{\beta(T-t)}d\overline{W}(t) + \int_{\mathbb{R}^d}\sqrt{\beta(T-t)}\mathbf{z}N(dt, d\mathbf{z}), \end{aligned} \tag{3.40}$$

with the corresponding Euler-Maruyama scheme

$$Y_{n+1} = \begin{cases} Y_n + [\frac{1}{2}\beta(T-t_n)Y_n + s_\theta(T-t_n, Y_n)]\delta t \\ \quad +\sqrt{\beta(T-t_n)}\delta\overline{W}_n, \quad \text{with probability } 1-\lambda\delta t, \\ Y_n + [\frac{1}{2}\beta(T-t_n)Y_n + s_\theta(T-t_n, Y_n)]\delta t \\ \quad +\sqrt{\beta(T-t_n)}\delta\overline{W}_n, +\sqrt{\beta(T-t_n)}\mathbf{z} \quad \text{with probability } \lambda\delta t. \end{cases} \tag{3.41}$$

The neural network $s_\theta(t,x)$ is trained with the following loss function:

$$\mathcal{L}(\theta) = \int_0^T \mathbb{E}_{X(t),\zeta}\left\{\left\|s_\theta(t,X(t))\right\|^2 + 2[\sigma^2(t) + \lambda\eta^2(t)]\zeta^\top \nabla_x s_\theta(t,X(t))\zeta\right\}dt \tag{3.42}$$

$$= \int_0^T \mathbb{E}_{X(t),\zeta}\left\{\left\|s_\theta(t,X(t))\right\|^2 + 2(1+\lambda)\beta(t)\zeta^\top \nabla_x s_\theta(t,X(t))\zeta\right\}dt. \tag{3.43}$$

We refer to this case as "VPSDE + Jump" in our experiments.

4

Experiments

In our experiments, we implemented and trained three neural networks using JAX on toy and real-world datasets. For toy datasets, we employed two networks: a simpler architecture, depicted in Figure 1, for datasets that are relatively easy to train, and a deeper MLP for more challenging datasets, which incorporates the time variable $t \in [0, T]$ via positional embeddings (Gehring et al., 2017), concatenated with input data at multiple stages, as shown in Figure 2. For real-world datasets, we implemented a U-Net, with an architecture that follows those used in most diffusion models (Song et al., 2021b; Ho et al., 2020). All neural networks are trained using

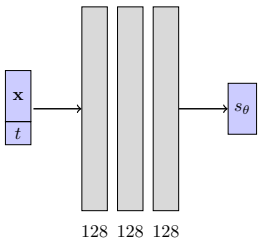

Figure 1: Simple Neural Network

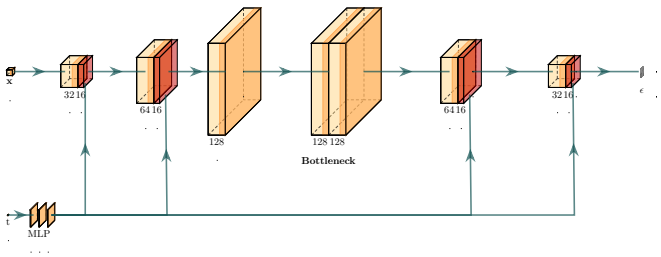

Figure 2: A deeper neural network with time embedding and projection.

the LION optimizer and SiLU activation function. Gradient norm clipping is applied with a maximum norm of 0.5 for toy datasets and 1.0 for the real-world dataset. An exponential moving average (EMA) is used during training, with a rate of 0.99 for toy datasets and 0.9999 for the real-world dataset, aligning with settings in most diffusion model papers (Yoon et al., 2023; Dockhorn et al., 2022).

4.1 Toy Dataset

For all toy datasets, we evaluate model performance using MMD. We conducted experiments with 9 toy datasets and the results are presented in the Table 4.2. The datasets are described as follows.

- **GM32, GM48, GM64**: These are Gaussian Mixture datasets, each with two components centered at $[0, 0]$ and $[d, d]$, where d is 32, 48, or 64, respectively. The covariances are fixed at $10I_d$ and $20I_d$, with mixture weights in a 1:9 ratio. The MMD results are shown in Table 4.2.

Hyperparameters	Diffusion	VPSDE	VPSDE + Jump
ξ	3×10^{-4}	3×10^{-4}	3×10^{-4}
β_{min}	-	0.02	3.4
β_{max}	-	20	3.5
λ	5	-	0.1
η	1.0	-	-
a	1.0	-	-
σ	$\sqrt{2}$	-	-
δt	0.01	0.01	0.01
T	3	3	3
Features	128	128	128

Table 4.1: Hyperparameter Settings.

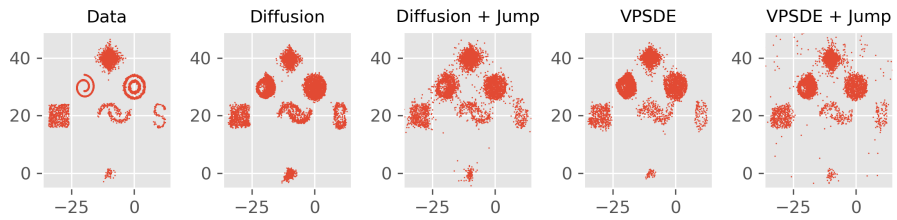

Figure 3: Results evaluated at step 300 on the Mixture dataset, using the deeper neural network trained for 10000 epochs.

- **Mixture**: A combination of 7 distributions (Laplace, Swiss Roll, Circles, Uniform, Moons, S-Curve, and Multivariate Normal), each positioned at distinct locations in the plane and scaled by a distance parameter, as shown in Figure 3.
- **GM**: A Gaussian Mixture with 9 components, with means at positions such as $[0, 2]$, $[-1, 1]$, $[1, 1]$, $[-2, 0]$, $[0, 0]$, $[2, 0]$, $[0, -2]$, $[-1, -1]$, and $[1, -1]$, scaled by a distance parameter. Each component has a small covariance matrix, as shown in Figure 4.
- **Swiss Rolls**: Comprises five Swiss Roll distributions, each a 2D projection of a 3D spiral, shifted to positions such as $[0, 2]$, $[-2, 0]$, $[0, 0]$, $[2, 0]$, and $[0, -2]$, scaled by a distance parameter, as shown in Figure 5
- **Circles**: This dataset consists of four large circles (radius 10) and four smaller

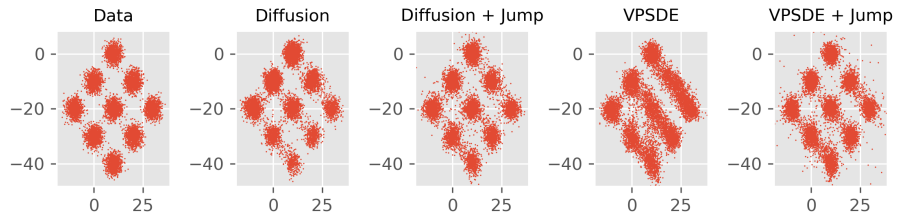

Figure 4: Results evaluated at step 300 on the GM dataset, using the simple neural network trained for 5000 epochs.

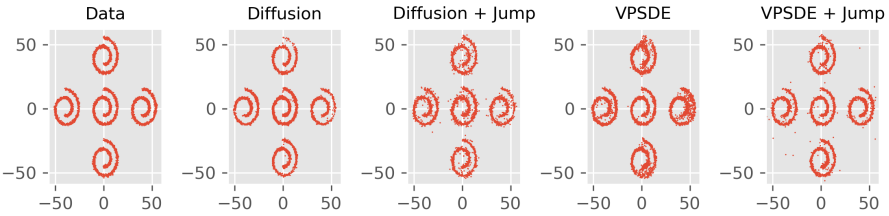

Figure 5: Results evaluated at step 300 on the Swiss Rolls dataset, using the deeper neural network trained for 10000 epochs.

circles (radius 5) nested inside them, centered at $(-20, 0)$, $(20, 0)$, $(0, 20)$, and $(0, -20)$. Gaussian noise is added to the points, as shown in Figure 6.

- **Floral**: Generated from a grayscale floral image, points are sampled from black pixels with Gaussian noise added and scaled to represent the flower's intricate, non-uniform shape, as shown in Figure 8.
- **Rose**: Similar to Floral, this dataset is generated from a grayscale rose image, as shown in Figure 9.

The neural networks are trained with a learning rate schedule combining a linear warmup phase, increasing linearly from 0 to a peak value of ξ over the first half of the training epochs, and a cosine decay phase, reducing from ξ to 0.01ξ over the remaining epochs. For more challenging datasets, we use the same parameters, which are shown in Table 4.1, except for reducing the first hidden layer's dimension from 128 to 32, resulting in hidden layer dimensions of $[32, 64, 128, 128, 128, 64, 32]$, as shown in Figure 2. Our experiment involved 9 toy datasets, with MMD results summarized in Table 4.2.

Figure 1 in Section 3.1 shows results for a simple neural network trained for 1000 epochs on the GM32, GM48, and GM64 datasets. We use 1000 epochs for two reasons: first, fewer epochs highlight the jump's effect; second, the jump model consistently outperforms non-jump models, even with more epochs, as shown in Figure 7 for the GM64 dataset after 3000 epochs.

Additionally, Figure 6 illustrates the MMD progression over time steps for the Circles dataset, showing that Diffusion with Jump achieves a lower final MMD ($0.42 \pm 0.15 \times 10^{-3}$) compared to Diffusion alone ($0.58 \pm 0.20 \times 10^{-3}$). Figures 8 and 9 present results for a deeper neural network trained on the Floral and Rose datasets. These toy datasets demonstrate that jump models effectively capture their distributions.

Conclusion from toy-dataset experiments

Table 4.2 presents the MMD scores under the hyperparameter settings shown in Table 4.1. The table reports MMD scores evaluated at step $N = \frac{T}{\delta t} = 300$ for four models: Diffusion, Jump-Diffusion, VPSDE, and Jump-VPSDE. Evaluations were conducted using a simple neural network on the GM32, GM48, GM64, GM, and Circles datasets, and a deeper neural network on the Mixture, Floral, Swiss Roll, and Rose datasets. Across all datasets, models incorporating jump processes consistently achieve lower MMD scores than diffusion-only models.

Data	Diff.	Diff. + Jump	VPSDE	VPSDE + Jump
	Simple Neural Network			
GM32	73.08 ± 1.33	$\mathbf{9.93} \pm 0.33$	13.21 ± 0.32	$\mathbf{6.79} \pm 0.17$
GM48	188.98 ± 1.50	$\mathbf{55.41} \pm 1.12$	46.49 ± 0.81	$\mathbf{18.90} \pm 0.56$
GM64	248.34 ± 1.52	$\mathbf{132.65} \pm 0.87$	78.49 ± 0.69	$\mathbf{43.27} \pm 0.71$
GM	13.15 ± 0.18	$\mathbf{3.03} \pm 0.10$	6.36 ± 0.10	$\mathbf{1.78} \pm 0.04$
Circles	1.62 ± 0.02	$\mathbf{1.43} \pm 0.02$	2.08 ± 0.03	$\mathbf{1.67} \pm 0.02$
	Deeper Neural Network			
Mixture	28.67 ± 9.94	$\mathbf{4.48} \pm 1.55$	5.25 ± 1.81	$\mathbf{3.21} \pm 1.10$
Floral	10.34 ± 3.57	$\mathbf{3.01} \pm 1.03$	3.01 ± 1.05	$\mathbf{2.97} \pm 1.03$
Swiss Rolls	2.62 ± 0.90	$\mathbf{1.28} \pm 0.44$	0.99 ± 0.35	$\mathbf{2.56} \pm 0.87$
Rose	23.66 ± 0.17	$\mathbf{4.49} \pm 0.09$	2.44 ± 0.03	$\mathbf{0.81} \pm 0.28$

Table 4.2: MMD evaluated in last step, averaged over 30 runs, with the corresponding 95 % confidence intervals, in units of $1e-3$.

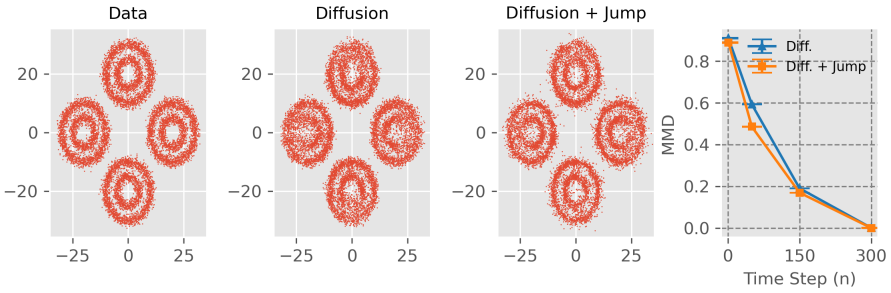

Figure 6: Results evaluated at step 300 on the Circles dataset, using the simple neural network trained for 10000 epochs. The line plot shows MMD with final achieved MMD is $\mathbf{0.42} \pm 0.15$ and 0.58 ± 0.20 (in units of $1e-3$) for jump + diffusion and diffusion, respectively.

Table 4.3 reports MMD scores for the Floral dataset. The table compares Diffusion and VPSDE models with no jump and with jump at $\lambda = 0.05, 0.1, 0.3, 0.5, 1, 3, 5$, evaluated at steps 2, 75, 150, and 300. In this table, we show that models with jump outperform their non-jump counterparts at step 300 across a wide range of λ values, indicating that the performance of jump processes is robust across different jump intensities. Additional experimental results are provided in Appendix B.

4.2 Image Generation

This experiment primarily investigates a subset of the CIFAR-10 dataset, consisting of 40 cat images and 10 dog images, by applying the Diffusion with Jump and comparing it with the Diffusion. We set the dimensions of the hidden layers of the U-Net to $[128, 256, 512, 512]$. Each operation is followed by group normalization with 8 groups and a residual connection within each block. The bottleneck, at a resolution of 4×4 with 512 channels, comprises a residual block and a multi-head self-attention layer with 8 heads. For the CIFAR-10 subset, we set λ to 5 during training, and then

λ	Step	Diffusion		VPSDE	
		Mean	95% CI	Mean	95% CI
No Jump	2	512.71	± 182.72	481.89	± 171.51
	75	451.48	± 160.71	486.17	± 173.34
	150	329.80	± 117.56	495.85	± 176.58
	300	10.34	± 3.57	3.01	± 1.05
0.05	2	511.14	± 181.90	512.32	± 182.31
	75	448.26	± 159.54	494.54	± 175.98
	150	328.05	± 116.70	453.19	± 161.15
	300	**9.81**	± 3.39	**2.73**	± 0.94
0.1	2	511.72	± 182.00	514.35	± 182.93
	75	441.76	± 157.16	493.63	± 175.48
	150	325.15	± 115.84	442.84	± 157.47
	300	**9.83**	± 3.40	**2.97**	± 1.03
0.3	2	510.76	± 181.86	518.00	± 184.39
	75	433.69	± 154.21	490.64	± 174.32
	150	320.55	± 114.21	414.08	± 147.03
	300	**8.56**	± 2.96	3.67	± 1.27
0.5	2	508.83	± 181.11	519.40	± 184.85
	75	423.67	± 150.99	482.17	± 171.52
	150	315.69	± 112.38	386.59	± 137.27
	300	**7.62**	± 2.62	5.06	± 1.76
1	2	508.13	± 180.92	526.01	± 187.20
	75	405.64	± 144.18	452.16	± 160.04
	150	303.14	± 107.77	328.69	± 116.23
	300	**5.78**	± 1.97	7.44	± 2.60
3	2	498.76	± 177.53	524.35	± 186.45
	75	350.67	± 124.71	320.45	± 113.27
	150	271.99	± 96.80	214.51	± 75.85
	300	**3.71**	± 1.26	11.19	± 3.92
5	2	490.95	± 174.66	502.91	± 178.74
	75	307.96	± 109.49	233.10	± 82.40
	150	245.35	± 87.27	159.80	± 56.62
	300	**3.01**	± 1.03	13.78	± 4.83

Table 4.3: The MMD averaged over 30 runs with 95% confidence intervals across steps and λ for the Floral dataset, in units of $1e-3$.

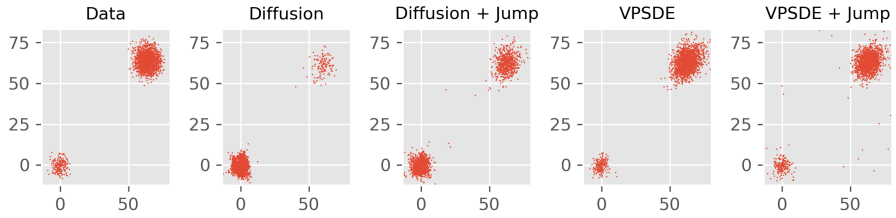

Figure 7: Results evaluated at step 300 on the GM64 dataset, using the simple neural network trained for 3000 epochs. The MMDs are 259.95±1.65, **124.35** ±1.69, 9.44 ± 0.28, and **8.78** $\pm$ 0.33 respectively.

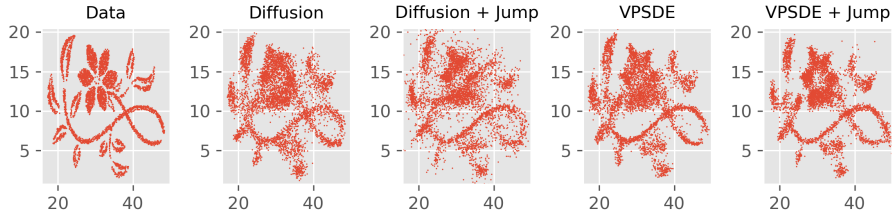

Figure 8: Results evaluated at step 300 on the Floral dataset, using the deeper neural network trained for 10000 epochs.

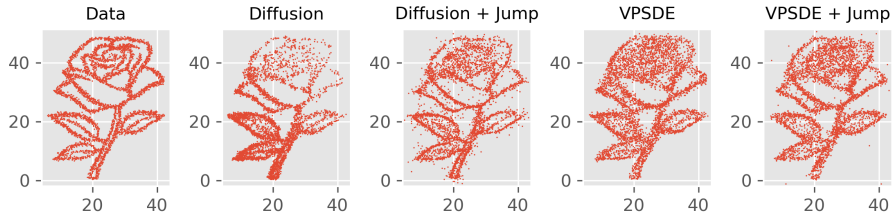

Figure 9: Results evaluated at step 300 on the Rose dataset, using the deeper neural network trained for 10000 epochs. This dataset is from the Dockhorn et al. (2022).

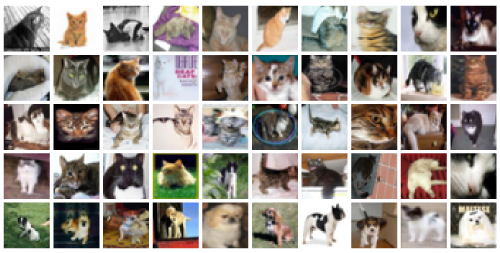

Figure 10: 50 Images from CIFAR-10 dataset.

	Diff. (200K)	Diff. (500K)	Diff. + Jump (200K)
NFE	FID	FID	FID
2	439.34	446.47	470.50
10	295.97	292.46	301.24
100	259.85	253.78	300.58
200	248.75	233.54	299.13
500	238.88	235.53	276.36
800	233.40	230.05	255.08
1000	227.41	231.91	224.61

Table 4.4: FID and time for Diffusion and Diffusion + Jump across NFE.

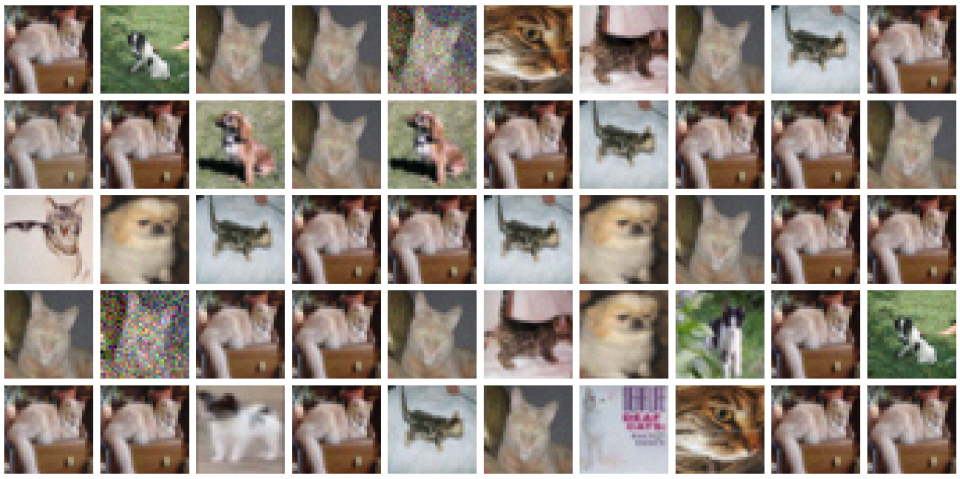

Figure 11: The images generated by Diffusion + jump with NFE = 1000, $\lambda = 1$. U-Net was trained for 200000 epochs.

set to 1 during the reverse process. We conducted two comparisons between Diffusion and Diffusion with Jump. First, we trained the U-Net for both processes using the same number of epochs. Second, training Diffusion with Jump via slice score matching is slower than training Diffusion, so we trained the U-Net for Diffusion for 500000 epochs to match the training clock time of Diffusion with Jump, revealing that Diffusion generates only five distinct images, with remaining outputs being duplicates due to mode collapse, as shown in Figure 13, whereas Diffusion with Jump produces significantly more diverse images, as shown in Figure 11. Additionally, we evaluate the quality of generated images using FID, comparing FID scores across NFE, as shown in Table 4.4. In our case, NFE is the same as the step N because we are using the Euler-Maruyama scheme to simulate the reverse SDE. The settings of hyperparameters are align with Table 4.1.

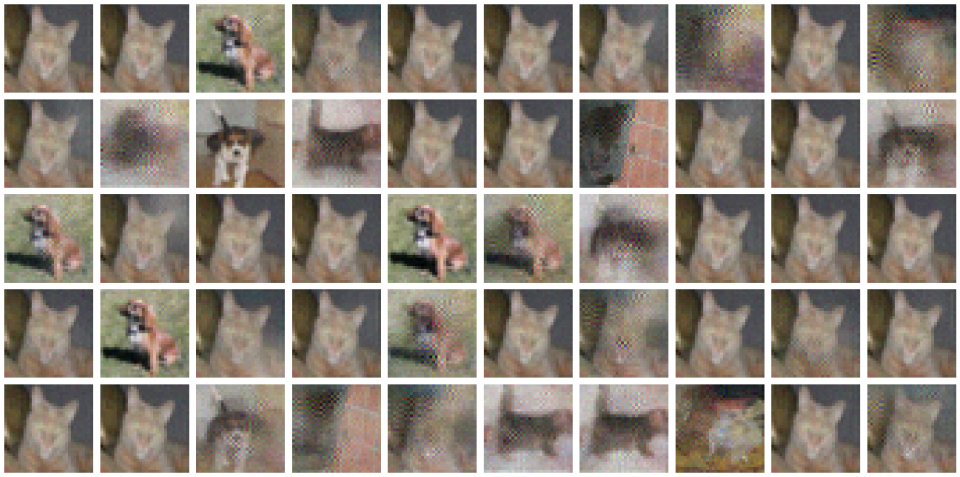

Figure 12: The images generated by Diffusion with NFE = 1000. U-Net was trained for 200000 epochs.

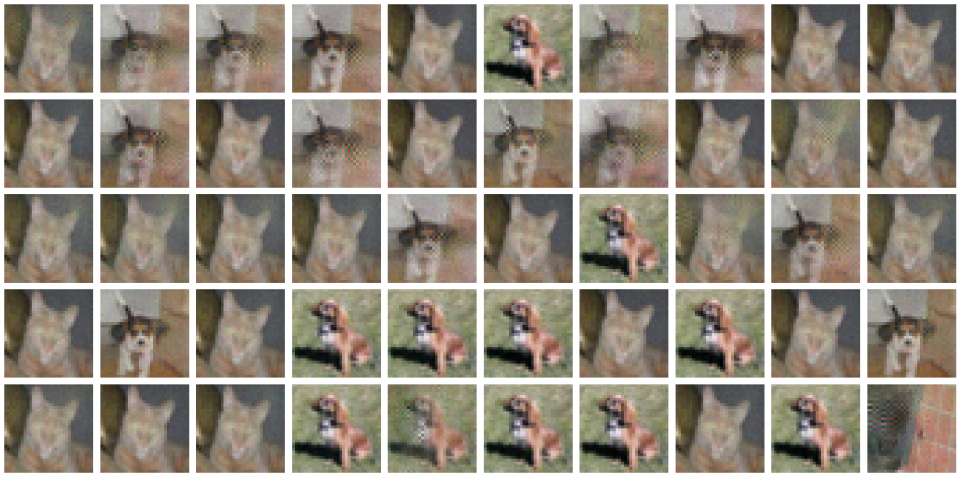

Figure 13: The images generated by Diffusion with NFE = 1000. U-Net was trained for 500000 epochs.

5

Conclusion

In this thesis, we introduced piecewise diffusive generative models based on stochastic differential equations driven by a Poisson random measure. This extends the current diffusion models by incorporating jump processes. We also derived a score-based loss function to train the score neural network. We investigate two cases throughout this thesis : Diffusion with Jump and VPSDE with Jump. Our experiments on toy datasets (see Section 4.1) suggest that jump processes can enhance performance, as evidenced by improved maximum mean discrepancy (MMD) metrics in most cases (e.g., $0.42 \pm 0.15 \times 10^{-3}$ for the Circles dataset with jump, compared to $0.58 \pm 0.20 \times 10^{-3}$ without). For the CIFAR-10 subset, a U-Net trained for 200000 epochs with jump processes generated more diverse images, mitigating model collapse observed in the diffusion models. These findings indicate that combining diffusion models with jump offer advantages over diffusion-only models.

The potential of the proposed piecewise-diffusive generative model to mitigate challenges associated with heavy-tailed distributions is promising, particularly when compared to diffusion-only models or approaches using infinite-activity α-stable Lévy noise, such as those explored by Yoon et al. (2023), which encounter intractable reverse dynamics. Given these preliminary results, we believe our approach offers potential as a foundation for future generative modeling research, warranting additional exploration to assess its broader impact.

Bibliography

Anderson, B. D. (1982). Reverse-time diffusion equation models. *Stochastic Processes and their Applications*, 12(3):313–326.

Applebaum, D. (2009). *Lévy Processes and Stochastic Calculus*. Cambridge Studies in Advanced Mathematics. Cambridge University Press, 2 edition.

Arjovsky, M., Chintala, S., and Bottou, L. (2017). Wasserstein gan. In *Proceedings of the 34th International Conference on Machine Learning*, volume 70, pages 214–223.

Athreya, K. B. and Lahiri, S. N. (2006). *Measure Theory and Probability Theory*. Springer.

Blattmann, A., Rombach, R., Esser, P., and Ommer, B. (2023). Videoldm: Video generation with latent diffusion models. *arXiv preprint arXiv:2302.03011*.

Chudley, C. and Elliott, R. (1961). Neutron scattering from a liquid on a jump diffusion model. *Proceedings of the Physical Society*, 77(2):353.

Cohn, D. L. (2013). *Measure Theory*. Birkhäuser, Boston, MA, second edition.

Deasy, J., Simidjievski, N., and Liò, P. (2021). Heavy-tailed denoising score matching. *arXiv preprint arXiv:2112.09788*.

Dhariwal, P. and Nichol, A. (2021). Diffusion models beat gans on image synthesis. In *Advances in Neural Information Processing Systems (NeurIPS)*, volume 34, pages 8780–8794.

Dockhorn, T., Vahdat, A., and Kreis, K. (2022). Score-based generative modeling with critically-damped langevin diffusion. In *International Conference on Learning Representations*.

Gao, J.-X., Wang, Z.-Y., Zhang, M. Q., Qian, M.-P., and Jiang, D.-Q. (2022). A data-driven method to learn a jump diffusion process from aggregate biological gene expression data. *Journal of Theoretical Biology*, 532:110923.

Gehring, J., Auli, M., Grangier, D., Yarats, D., and Dauphin, Y. N. (2017). Convolutional sequence to sequence learning. In Precup, D. and Teh, Y. W., editors, *Proceedings of the 34th International Conference on Machine Learning*, volume 70 of *Proceedings of Machine Learning Research*, pages 1243–1252. PMLR.

Goodfellow, I., Pouget-Abadie, J., Mirza, M., Xu, B., Warde-Farley, D., Ozair, S., Courville, A., and Bengio, Y. (2014). Generative adversarial networks. *Advances in Neural Information Processing Systems*, 27.

Haussmann, U. G. and Pardoux, E. (1986). Time reversal of diffusions. *The Annals of Probability*, 14(4):1188–1205.

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

Ho, J., Jain, A., and Abbeel, P. (2020). Denoising diffusion probabilistic models. In *Advances in Neural Information Processing Systems*, volume 33, pages 6840–6851.

Ho, J., Salimans, T., Jain, A., Sohl-Dickstein, J., and Abbeel, P. (2022). Imagen video: High definition video generation with diffusion models. *arXiv preprint arXiv:2209.01658*.

Hutchinson, M. F. (1990). A stochastic estimator of the trace of the influence matrix for laplacian smoothing splines. *Communications in Statistics - Simulation and Computation*, 19(2):433–450.

Kalise, D., Sharma, A., and Tretyakov, M. V. (2023). Consensus-based optimization via jump-diffusion stochastic differential equations. *Mathematical Models and Methods in Applied Sciences*, 33(02):289–339.

Karras, T., Aittala, M., Aila, T., and Laine, S. (2022). Elucidating the Design Space of Diffusion-Based Generative Models. *arXiv preprint arXiv:2206.00364*.

Kingma, D. P. and Welling, M. (2014). Auto-encoding variational bayes. In *2nd International Conference on Learning Representations (ICLR)*.

Kong, Z., Ping, W., Huang, J., Zhao, K., and Catanzaro, B. (2021). Diffwave: A versatile diffusion model for audio synthesis. *International Conference on Learning Representations*.

Liu, Y., Ren, T., Zeng, W., Wang, P., Yu, N., and Chen, B. (2022). Flow-guided diffusion for video frame interpolation. *arXiv preprint arXiv:2208.13744*.

Meng, C., Song, Y., Song, J., Wu, J., and Ermon, S. (2022). Sdedit: Image synthesis and editing with stochastic differential equations. *International Conference on Learning Representations*.

Merton, R. C. (1976). Option pricing when underlying stock returns are discontinuous. *Journal of financial economics*, 3(1-2):125–144.

Nachmani, E., Roman, R. S., and Wolf, L. (2021). Denoising diffusion gamma models. *arXiv preprint arXiv:2110.05948*.

Popov, V., Yermekova, A., Sadekova, T., Khrapov, A., and Kudinov, M. S. (2025). Improved sampling algorithms for lévy-itô diffusion models. In *The Thirteenth International Conference on Learning Representations*, Singapore. ICLR 2025 Poster.

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

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

Saharia, C., Ho, J., Chan, W., Salimans, T., Fleet, D. J., and Norouzi, M. (2022). Palette: Image-to-image diffusion models. *ACM Transactions on Graphics*, 41(4):54:1–54:10.

Shariatian, D., Simsekli, U., and Durmus, A. O. (2025). Denoising levy probabilistic models. In *The Thirteenth International Conference on Learning Representations*.

Sohl-Dickstein, J., Weiss, E., Maheswaranathan, N., and Ganguli, S. (2015). Deep unsupervised learning using nonequilibrium thermodynamics. *International Conference on Machine Learning*.

Song, J., Meng, C., and Ermon, S. (2021a). Denoising diffusion implicit models. In *International Conference on Learning Representations (ICLR)*.

Song, Y., Garg, S., Shi, J., and Ermon, S. (2019). Sliced score matching: A scalable approach to density and score estimation. In *Proceedings of the Thirty-Fifth Conference on Uncertainty in Artificial Intelligence, UAI 2019, Tel Aviv, Israel, July 22-25, 2019*, volume 115 of *Proceedings of Machine Learning Research*, pages 574–584. PMLR.

Song, Y., Sohl-Dickstein, J., Kingma, D. P., Kumar, A., Ermon, S., and Poole, B. (2021b). Score-based generative modeling through stochastic differential equations. In *International Conference on Learning Representations*.

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

Yoon, E., Park, K., Kim, S., and Lim, S. (2023). Score-based generative models with lévy processes. In *Thirty-seventh Conference on Neural Information Processing Systems*.

Øksendal, B. (2003). *Stochastic Differential Equations: An Introduction with Applications*. Springer-Verlag Heidelberg New York, fifth edition, corrected printing edition.

A

Preliminaries

The appendices offer necessary definitions and more experiment details. The concepts and results presented here are primarily based on Øksendal (2003); Athreya and Lahiri (2006); Cohn (2013); Applebaum (2009). Appendix A.1 provides the definitions used to derive the reverse SDE. Appendix B presents all experiment results. The code is available here.

A.1 Preliminaries in Measure Theory

This section introduces fundamental concepts from measure theory required for understanding stochastic differential equation (SDE) based diffusion models.

Definition A.1.1. Let Ω be a nonempty set. The **power set** of Ω is the set of *all* subsets of Ω, denoted by $\mathcal{P}(\Omega)$, and is defined as

$$\mathcal{P}(\Omega) \coloneqq \{A : A \subseteq \Omega\}.$$

Definition A.1.2. A collection of sets $\mathcal{F} \subset \mathcal{P}(\Omega)$ is called an **algebra** if

(a) $\Omega \in \mathcal{F}$;

(b) $A \in \mathcal{F} \implies A^c \in \mathcal{F}$;

(c) $A, B \in \mathcal{F} \implies A \cup B \in \mathcal{F}$.

Definition A.1.3. A class $\mathcal{F} \subset \mathcal{P}(\Omega)$ is called a σ-**algebra** if it is an algebra and if it satisfies

- $A_1, A_2, \cdots \in \mathcal{F} \implies \bigcup_{n=1}^{\infty} A_n \in \mathcal{F}$.

Let $\mathcal{G} \subset \mathcal{P}(\Omega)$ be a σ-**algebra**, $\mathcal{G}$ is **sub-**σ-**algebra** if $\mathcal{G} \subset \mathcal{F}$.

Definition A.1.4. Let Ω be a nonempty set and $\mathcal{F}$ be an algebra on Ω. Then, a set function μ on $\mathcal{F}$ is called a **measure** if

(a) $\forall\, A \in \mathcal{F}$: $\mu(A) \in [0, \infty]$;

(b) $\mu(\emptyset) = 0$;

(c) for any disjoint collection of sets $A_1, A_2, \cdots \in \mathcal{F}$,

$$\mu\left(\bigcup_{n=1}^{\infty} A_n\right) = \sum_{n=1}^{\infty} \mu(A_n).$$

Definition A.1.5. Let Ω be a nonempty set and let $\mathcal{F}$ be a σ-algebra on Ω. Then the pair $(\Omega, \mathcal{F})$ is called a **measurable space**. If μ is a measure on $(\Omega, \mathcal{F})$, then the triplet $(\Omega, \mathcal{F}, \mu)$ is called a **measure space**.

Definition A.1.6. A **probability measure** P on a measurable space $(\Omega, \mathcal{F})$ is a function $P\colon \mathcal{F} \to [0,1]$ such that

(a) $P(\emptyset) = 0$;

(b) $P(\Omega) = 1$;

(c) if $A_1, A_2, \cdots \in \mathcal{F}$ and $\{A_i\}_{i=1}^{\infty}$ is disjoint (i.e. $A_i \cap A_j = \emptyset$ if $i \neq j$) then

$$P\left(\bigcup_{n=1}^{\infty} A_n\right) = \sum_{n=1}^{\infty} P(A_n).$$

The triplet $(\Omega, \mathcal{F}, P)$ is called a **probability space**.

Definition A.1.7. If $(\Omega, \mathcal{F}, P)$ is a given probability space, then a function $Y\colon \Omega \to \mathbb{R}^n$ is called $\mathcal{F}$-**measurable** if

$$Y^{-1}(U) := \{\omega \in \Omega; Y(\omega) \in U\} \in \mathcal{F}$$

for all open sets $U \in \mathbb{R}^n$.

Definition A.1.8. Let $(\Omega, \mathcal{F}, P)$ be a probability space. An $\mathcal{F}$-measurable function $X\colon \Omega \to \mathbb{R}^n$ is called a **random variable**.

Definition A.1.9. Let $\Omega = \mathbb{R}^d$. The **Borel σ-algebra** on $\mathbb{R}^d$, denoted $\mathcal{B}(\mathbb{R}^d)$, is the smallest σ-algebra on $\mathbb{R}^d$ that contains all open subsets of $\mathbb{R}^d$.

Definition A.1.10. Let $(\Omega, \mathcal{F}, \mu)$ be a measure space and $0 < p \leq \infty$. The L^p-**spaces** $L^p(\Omega, \mathcal{F}, \mu)$ is defined as

$$L^p(\Omega, \mathcal{F}, \mu) \coloneqq \left\{ f : \int |f|^p \, d\mu < \infty, \quad 0 < p < \infty \right\}.$$

Let $1 < p \leq \infty$, then $L^p(\Omega, \mathcal{F}, \mu)$ is a vector space.

Definition A.1.11. A function f from vector space V to $\mathbb{R}_+$ denoted by $f(v) \coloneqq \|v\|$ is called a **norm** if

(a) $v_1, v_2 \in V \implies \|v_1 + v_2\| \leq \|v_1\| + \|v_2\|$;

(b) $a \in \mathbb{R}, v \in V \implies \|av\| = |a|\|v\|$;

(c) $\|v\| = 0 \iff v = 0$.

Definition A.1.12. A vector space V with a norm is called a **normed linear space**, denoted as $(V, \|\cdot\|)$.

Definition A.1.13. A complete normed linear space $(V, \|\cdot\|)$ is called a **Banach space**.

Definition A.1.14. Let $(V_i, \|\cdot\|_i)$, $i = 1, 2$ be two normed linear spaces over $\mathbb{R}$. A function T from V_1 to V_2 is called a **linear operator** if

$$T(\alpha_1 x + \alpha_2 y) = \alpha_1 T(x) + \alpha_2 T(y), \quad \alpha_1, \alpha_2 \in \mathbb{R}, \quad x, y \in V_1.$$

A linear operator T from $(V_1, \|\cdot\|_1)$ to $(V_2, \|\cdot\|_2)$ is called **bounded** if $\|T\| := sup\{\|Tx\|_2 : \|x\|_1 < 1\} < \infty$.

Definition A.1.15. Let B be a real Banach space. A **semigroup** is a family of bounded, linear operators $(T_t, t \geq 0)$ on B satisfies:

(a) $T_{s+t} = T_s T_t$ for all $s, t \geq 0$;

(b) $T_0 = I$;

(c) $\forall\, \psi \in B : \lim_{t \to 0^+} \|T_t \psi - \psi\| = 0$.

Let $(T_t, t \geq 0)$ be a semigroup in a Banach space B, and define

$$D_{\mathcal{A}} := \left\{ \psi \in B; \exists \phi \in B : \lim_{t \to 0^+} \left\| \frac{T_t \psi - \psi}{t} - \phi \right\| = 0 \right\},$$

where $\|\cdot\|$ is the norm on B.

Definition A.1.16. Let $\mathcal{A} : D_{\mathcal{A}} \to B$. $\mathcal{A}$ is called the **generator** of the semigroup $(T_t, t \geq 0)$, and

$$\mathcal{A}\psi := \lim_{t \to 0^+} \frac{T_t \psi - \psi}{t}.$$

A.2 Preliminaries in Diffusion Model

Building on Section 2.1, this section introduces definitions of stochastic processes, Brownian motion and SDE.

Definition A.2.1. A **stochastic process** is a parametrized collection of random variables $\{X(t) : t \in T\}$ defined on a probability space $(\Omega, \mathcal{F}, P)$.

Definition A.2.2. A stochastic process $\{W(t) : t \geq 0\}$ is called **Wiener Process** if it satisfies

(a) $W(0) = 0$ almost surely;

(b) $\forall\ t \geq 0$: $W(t) \sim \mathcal{N}(0, t)$;

(c) $W(t) - W(s) \overset{D}{=} W(t - s)$;

(d) If $0 \leq q < r \leq s < t$, then $W(t) - W(s)$ and $W(r) - W(q)$ are independent random variables.

Definition A.2.3. A stochastic process $\{N(t) : t \geq 0\}$ is called **Poisson Process** if it satisfies

(a) $N(0) = 0$ almost surely ;

(b) $\forall\ t \geq 0$: $N(t) \sim \text{Pois}(\lambda t)$, where $\lambda \in (0, \infty)$;

(c) $N(t) - N(s) \stackrel{D}{=} N(t-s)$;

(d) If $0 \leq q < r \leq s < t$, then $N(t) - N(s)$ and $N(r) - N(q)$ are independent random variables.

Definition A.2.4. Let $(\Omega, \mathcal{F}, P)$ be a probability space. Let $N = \{1, \dots, n_0\}$ be a nonempty subset of $\mathbb{N} = \{1, 2, \dots\}$, $n_0 \leq \infty$.

(a) A collection $\{\mathcal{F}_n : n \in N\}$ of sub-σ-algebras of $\mathcal{F}$ is called a **filtration** if $\mathcal{F}_n \subset \mathcal{F}_{n+1}$ for all $1 \leq n < n_0$.

(b) A collection of random variables $\{X_n : n \in N\}$ is said to be **adapted** to the filtration $\{\mathcal{F}_n : n \in N\}$ if X_n is $\mathcal{F}_n$-measurable for all $n \in N$.

B

Experiments

λ	Step	Diffusion		VPSDE	
		Mean	95% CI	Mean	95% CI
No Jump	2	606.75	± 216.40	579.65	± 206.78
	75	472.53	± 168.52	584.20	± 208.57
	150	330.37	± 117.86	593.74	± 212.02
	300	28.67	± 9.94	5.25	± 1.81
0.05	2	605.89	± 216.07	610.83	± 217.82
	75	470.77	± 167.93	589.45	± 210.27
	150	330.25	± 117.90	533.08	± 190.15
	300	**26.92**	± 9.39	**3.12**	± 1.07
0.1	2	605.79	± 216.13	611.90	± 218.30
	75	470.16	± 167.75	589.96	± 210.46
	150	331.19	± 118.18	526.62	± 187.72
	300	**25.37**	± 8.82	**3.21**	± 1.10
0.3	2	605.04	± 215.72	615.93	± 219.59
	75	469.43	± 167.27	586.27	± 208.60
	150	335.43	± 119.77	498.38	± 177.65
	300	**20.27**	± 7.05	**5.09**	± 1.75
0.5	2	603.82	± 215.54	617.82	± 220.53
	75	465.19	± 165.97	575.55	± 205.26
	150	335.56	± 119.65	476.14	± 169.60
	300	**18.32**	± 6.27	7.12	± 2.48
1	2	602.76	± 215.01	623.53	± 222.37
	75	462.49	± 165.02	547.14	± 194.63
	150	341.51	± 121.72	420.53	± 149.38
	300	**11.20**	± 3.86	14.71	± 5.18
3	2	595.16	± 212.31	622.79	± 222.03
	75	427.08	± 152.30	417.58	± 148.48
	150	336.28	± 119.85	309.74	± 110.17
	300	**4.74**	± 1.64	33.88	± 11.86
5	2	587.22	± 209.34	599.35	± 213.59
	75	394.54	± 140.72	329.20	± 117.21
	150	320.77	± 114.33	254.53	± 90.50
	300	**4.48**	± 1.55	45.59	± 16.00

Table B.1: The MMD averaged 30 runs with 95% confidence intervals across steps and λ in Mixture dataset, in units of $1e-3$.

λ	Step	Diffusion		VPSDE	
		Mean	95% CI	Mean	95% CI
No Jump	2	452.72	± 161.35	432.07	± 154.03
	75	231.53	± 82.50	436.21	± 155.64
	150	95.44	± 34.06	445.57	± 159.06
	300	0.57	± 0.20	0.72	± 0.25
0.05	2	452.57	± 161.36	462.34	± 164.85
	75	231.47	± 82.50	426.01	± 151.81
	150	95.34	± 34.05	270.93	± 96.60
	300	**0.45**	± 0.16	**0.52**	± 0.18
0.1	2	452.44	± 161.34	463.95	± 165.45
	75	230.34	± 82.20	428.40	± 152.72
	150	95.32	± 34.01	267.13	± 95.17
	300	**0.46**	± 0.16	**0.54**	± 0.19
0.3	2	452.83	± 161.45	467.33	± 166.61
	75	231.28	± 82.38	423.75	± 150.61
	150	95.15	± 33.97	247.28	± 88.01
	300	**0.49**	± 0.17	0.75	± 0.26
0.5	2	451.55	± 161.14	469.49	± 167.50
	75	230.80	± 82.34	417.84	± 148.91
	150	95.33	± 34.02	234.59	± 83.42
	300	**0.47**	± 0.16	0.92	± 0.32
1	2	451.09	± 160.78	475.29	± 169.34
	75	227.71	± 81.27	390.44	± 138.68
	150	94.07	± 33.56	197.88	± 70.20
	300	**0.46**	± 0.16	1.28	± 0.45
3	2	445.51	± 158.83	474.52	± 169.06
	75	212.67	± 75.79	263.47	± 93.55
	150	89.93	± 32.06	127.01	± 44.97
	300	**0.44**	± 0.15	1.80	± 0.63
5	2	437.32	± 155.80	451.29	± 160.72
	75	193.29	± 68.87	179.97	± 63.88
	150	85.20	± 30.35	89.97	± 31.83
	300	**0.42**	± 0.15	2.00	± 0.71

Table B.2: The MMD averaged 30 runs with 95% confidence intervals across steps and λ in Circles dataset, in units of $1e-3$.

λ	Step	Diffusion		VPSDE	
		Mean	95% CI	Mean	95% CI
No Jump	1	949.81	± 2.16	899.72	± 2.22
	50	777.26	± 2.07	893.64	± 2.64
	150	376.75	± 1.35	919.81	± 3.06
	300	23.66	± 0.17	2.44	± 0.03
0.05	1	947.62	± 2.40	954.64	± 2.22
	50	771.83	± 2.57	919.72	± 2.52
	150	373.57	± 0.90	706.01	± 2.43
	300	**22.87**	± 0.15	**2.01**	± 0.03
0.1	1	946.50	± 2.82	960.70	± 2.12
	50	767.63	± 2.00	928.64	± 2.55
	150	370.29	± 1.30	694.40	± 2.57
	300	**22.54**	± 0.20	**2.07**	± 0.04
0.3	1	948.84	± 3.07	961.48	± 2.46
	50	750.30	± 2.58	940.49	± 2.74
	150	362.37	± 1.33	645.62	± 3.32
	300	**20.67**	± 0.22	2.68	± 0.04
0.5	1	944.98	± 2.96	963.93	± 2.65
	50	737.34	± 2.63	942.40	± 3.88
	150	353.95	± 1.31	604.64	± 2.12
	300	**17.95**	± 0.20	3.55	± 0.05
1	1	944.31	± 2.19	970.54	± 2.26
	50	714.79	± 2.77	910.49	± 4.52
	150	338.97	± 1.15	512.05	± 2.25
	300	**14.20**	± 0.28	5.16	± 0.05
3	1	934.42	± 2.62	974.55	± 2.64
	50	630.41	± 2.53	674.76	± 4.55
	150	303.21	± 0.92	321.85	± 3.14
	300	**6.85**	± 0.13	7.59	± 0.08
5	1	921.08	± 2.27	939.76	± 3.36
	50	554.68	± 2.02	481.39	± 2.67
	150	279.40	± 0.71	228.96	± 1.63
	300	**4.49**	± 0.09	8.47	± 0.09

Table B.3: The MMD averaged 30 runs with 95% confidence intervals across steps and λ in Rose dataset, in units of $1e-3$.

DEPARTMENT OF COMPUTER SCIENCE AND ENGINEERING
CHALMERS UNIVERSITY OF TECHNOLOGY
Gothenburg, Sweden
www.chalmers.se

CHALMERS
UNIVERSITY OF TECHNOLOGY