國　立　中　央　大　學

數學研究所

碩士論文

基於擴散模型的生成式 AI

A Mathematical Study of Generative AI Models

研 究 生：許展榮

指導教授：胡偉帆　教授

中 華 民 國 114 年 6 月

基於擴散模型的生成式 AI

摘要

隨著深度學習在圖像生成領域的快速發展，擴散模型（Diffusion Models）已成為一種兼具生成品質與靈活性的前沿方法。本論文系統性地回顧了三大類擴散生成模型：一是以離散馬可夫鏈為基礎的 Denoising Diffusion Probabilistic Models（DDPM）；二是透過隨機微分方程（SDE）與分數匹配（score matching）建立的 Score-Based Diffusion Models；三是以「一致性訓練」（Consistency Training）為核心，旨在以單步或少量步驟完成高品質生成的 Consistency Models。

在實驗部分，本研究採用 MNIST、CIFAR-10 與 Cat Faces 三個資料集，並以 Fréchet Inception Distance（FID）評估生成影像之真實度，同時計算反向取樣過程的函數評估次數（NFE）以衡量效率。結果顯示：DDPM 在相同步數下達到最低 FID. Consistency Models 雖能將取樣速度提升數倍，但在採用 LPIPS 感知損失時，其生成品質方能逼近傳統擴散模型。

關鍵字：生成式模型、擴散模型、一致性模型

A Mathematical Study of Generative AI Models

Abstract

Diffusion models have recently emerged as a powerful class of generative techniques, offering state-of-the-art sample quality in image generation tasks. This thesis provides a comprehensive review of three representative diffusion-based frameworks: (1) Denoising Diffusion Probabilistic Models (DDPM), which employ discrete Markov chains to model forward noising and reverse denoising; (2) Score-Based Diffusion Models, which generalize this approach via continuous-time stochastic differential equations (SDEs) and score matching; and (3) Consistency Models, which aim to collapse iterative sampling into a single or few learned steps through consistency training.

We conduct experiments on MNIST, CIFAR-10, and a Cat Faces dataset, evaluating generation fidelity using the Fréchet Inception Distance (FID) and measuring by the Number of Function Evaluations (NFE). Our findings reveal that DDPM consistently achieves the lowest FID under equal step counts. Consistency Models dramatically accelerate sampling—reducing NFE by orders of magnitude—but only attain comparable fidelity when trained with a perceptual LPIPS loss.

Keywords: Generative model, Diffusion model, Consistency model

Contents

List of Figures

List of Tables

page

Chapter 1

Introduction

The diffusion models, a type of generative model, are more and more popular in recent years. It predicts the noise in the image and then denoises it to generate a new image. It is a powerful model that can generate high-quality images. It used by many applications, such as image generation, image inpainting, and image super-resolution. Although the diffusion models are powerful, their concept is not hard to understand.

This thesis focuses on three representative branches of diffusion-based generative modeling. First, Denoising Diffusion Probabilistic Models (DDPM) [1] view the forward and reverse processes as discrete Markov chains, and they will recover clean data from noisy inputs. Second, Score-Based Diffusion Models [2] generalize this framework via stochastic differential equations (SDEs) and score matching, enabling continuous-time perturbations and flexible noise schedules. Third, Consistency Models [3] seek the efficient sampling method of diffusion models, learned function that directly maps noisy inputs back to clean data.

The remainder of the thesis is organized as follows:

- Chapter 2 reviews the mathematical foundations of diffusion models, including DDPM, score-based methods, and consistency training.
- Chapter 3 and Chapter 4 details the experimental setup on MNIST, CIFAR-10, and Cat Faces datasets, specifying architectures, loss functions, training hyperparameters, and results.
- Chapter 5 concludes with a summary of key findings, discusses limitations of current approaches.

Chapter 2

Diffusion Model

2.1 Diffusion Models

In the diffusion model, we have two parts: the forward process and the backward process. In the forward process, we gradually add noise to the image until it becomes sufficiently noisy. In the backward process, we progressively remove noise from the image until we get the final image. The neural network is trained to predict the noise added at each step, which enables effective denoising.

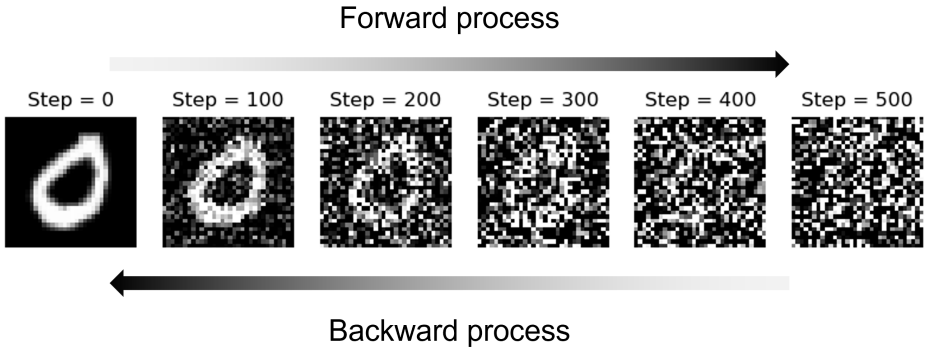


Figure 2.1: Diffusion model example

2.2 Denising Diffusion Probabilistic Models (DDPM)

In DDPM [1], we view the diffusion model as a Markov chain. Let $\boldsymbol{x}_0$ be the original image, and $\boldsymbol{x}_T$ be the final image after T steps of noise addition. Thus, we obtain a sequence $\boldsymbol{x}_0, \boldsymbol{x}_1, \ldots, \boldsymbol{x}_t, \ldots, \boldsymbol{x}_T$. At each timestep t, we define the state $\boldsymbol{x}_t$ in terms of the previous state $\boldsymbol{x}_{t-1}$ as follows:

$$\boldsymbol{x}_t := \sqrt{1-\beta_t}\boldsymbol{x}_{t-1} + \sqrt{\beta_t}\boldsymbol{z}_{t-1}, \tag{2.1}$$

where β_t, is a hyperparameter known as the variance schedule, which controls the magnitude of noise added at each step, $\boldsymbol{z}_{t-1}$ represents standard Gaussian noise. The noise term $\sqrt{\beta_t}\boldsymbol{z}_{t-1}$ determines the amount of noise added to the image at each timestep. In this way, the forward process can be expressed in terms of a conditional probability distribution:

$$q(\boldsymbol{x}_t|\boldsymbol{x}_{t-1}) := \mathcal{N}(\boldsymbol{x}_t; \sqrt{1-\beta_t}\boldsymbol{x}_{t-1}, \beta_t I), \tag{2.2}$$

where $\mathcal{N}(\cdot;\cdot,\cdot)$ denotes the isotropic Gaussian distribution of $\boldsymbol{x}_t$ with mean $\sqrt{1-\beta_t}\boldsymbol{x}_{t-1}$, covariance matrix $\beta_t I$ and I is the identity matrix. Additionally, the forward process establishes a relationship between the initial image $\boldsymbol{x}_0$ and the intermediate states $\boldsymbol{x}_t$ at any given timestep. Since all noise terms in the equation follow a Gaussian distribution, we use the properties of Gaussian distributions: the sum of two Gaussian-distributed variables is also Gaussian. This is described by the conditional distribution

$$q(\boldsymbol{x}_t|\boldsymbol{x}_0) = \mathcal{N}(\boldsymbol{x}_t; \sqrt{\overline{\alpha}_t}\boldsymbol{x}_0, (1-\overline{\alpha}_t)I), \tag{2.3}$$

where $\alpha_t = 1 - \beta_t$ is another hyperparameter that defines the noise variance at each step, and $\overline{\alpha}_t$ is recursively defined as $\overline{\alpha}_t := \prod_{s=1}^{t} \alpha_s$. Equivalently, one may express as $\boldsymbol{x}_t$:

$$\boldsymbol{x}_t = \sqrt{\overline{\alpha}_t}\boldsymbol{x}_0 + \sqrt{1-\overline{\alpha}_t}\boldsymbol{z}_{t,1}, \tag{2.4}$$

where $\boldsymbol{z}_{t,1}$ is a standard Gaussian random variable. This equation shows how the original image $\boldsymbol{x}_0$ influences the intermediate states $\boldsymbol{x}_t$ through the noise scaling factor α_t.

On the other hand, the backward process, which is aimed at denoising the image and recovering $\boldsymbol{x}_0$, can be derived from the forward process by applying Bayes' theorem. Specifically, the backward process from $\boldsymbol{x}_t$ to $\boldsymbol{x}_{t-1}$ given $\boldsymbol{x}_0$ is expressed as:

$$q(\boldsymbol{x}_{t-1}|\boldsymbol{x}_t, \boldsymbol{x}_0) = \frac{q(\boldsymbol{x}_t|\boldsymbol{x}_{t-1})q(\boldsymbol{x}_{t-1})}{q(\boldsymbol{x}_t)}. \tag{2.5}$$

We can compute the backward process then Eq. (2.7) becomes

$$q(\boldsymbol{x}_{t-1}|\boldsymbol{x}_t, \boldsymbol{x}_0) = \mathcal{N}(\boldsymbol{x}_{t-1}; \tilde{\mu}_t(\boldsymbol{x}_t), \tilde{\beta}_t\mathbf{I}), \tag{2.6}$$

where $\tilde{\boldsymbol{\mu}}_t(\boldsymbol{x}_t, \boldsymbol{x}_0) := \frac{\sqrt{\overline{\alpha}_{t-1}}}{1-\overline{\alpha}_{t-1}}\boldsymbol{x}_0 - \frac{\sqrt{\alpha_t}}{1-\alpha_t}\boldsymbol{x}_t$ and $\tilde{\beta}_t := \frac{1-\overline{\alpha}_{t-1}}{1-\overline{\alpha}_t}\beta_t$. Then, substitute $\boldsymbol{x}_t$ with $\sqrt{\overline{\alpha}_t}\boldsymbol{x}_0 + \sqrt{1-\overline{\alpha}_t}\boldsymbol{z}_{t,1}$ from Eq. (2.4) gives $\tilde{\boldsymbol{\mu}}_t(\boldsymbol{x}_t, \boldsymbol{x}_0) = \frac{1}{\sqrt{\alpha_t}}(\boldsymbol{x}_t - \frac{1-\alpha_t}{\sqrt{1-\overline{\alpha}_t}}\boldsymbol{z}_{t,1})$. In other words, the previous state $\boldsymbol{x}_{t-1}$ can be expressed in terms of the current state $\boldsymbol{x}_t$ as follows:

$$\boldsymbol{x}_{t-1} = \frac{1}{\sqrt{\alpha_t}}(\boldsymbol{x}_t - \frac{1-\alpha_t}{\sqrt{1-\overline{\alpha}_t}}\boldsymbol{z}_{t,1}) + \frac{1-\overline{\alpha}_{t-1}}{1-\overline{\alpha}_t}\beta_t\boldsymbol{z}, \tag{2.7}$$

where $\boldsymbol{z}$ is a standard Gaussian random variable. This shows that $\boldsymbol{x}_t$ can be expressed in terms of $\boldsymbol{x}_0$ and noise.

2.3 Score-Based Diffusion Models

In the score-based diffusion models [2], we can view the diffusion model as a stochastic diffusion equation (SDE). This perspective provides a continuous-time framework to model the gradual corruption and recovery of data using stochastic processes.

The forward process is described as a SDE in general form:

$$\begin{cases} d\boldsymbol{x}_t = \boldsymbol{f}(\boldsymbol{x}_t, t)dt + \boldsymbol{G}(\boldsymbol{x}_t, t)d\boldsymbol{W}_t, \\ \boldsymbol{x}(0) \sim p_0(\boldsymbol{x}), \end{cases} \tag{2.8}$$

where the random variable $\boldsymbol{x}(t)$ denoted as $\boldsymbol{x}_t$, $\boldsymbol{f}(\cdot, t) : \mathbb{R}^d \to \mathbb{R}^d$ is the drift coefficient, $\boldsymbol{G}(\cdot, t) : \mathbb{R}^d \to \mathbb{R}^{d\times d}$ is the diffusion coefficient, and $\boldsymbol{W}_t \in \mathbb{R}^{d\times d}$ is a standard Brownian motion. The standard Brownian motion $\boldsymbol{W}_t \in \mathbb{R}^{d\times d}$ is defined as a continuous-time stochastic process with independent increments, where each increment $\Delta\boldsymbol{W}_t = \boldsymbol{W}_{t+\Delta t} - \boldsymbol{W}_t$ follows a isotropic Gaussian distribution with mean zero and variance of time increment Δt. Those equations describe the dynamics of a system with random perturbations, where the drift term determines deterministic behavior and the diffusion term controls stochastic noise. Often, we use a simpler version where the noise does not depend on $\boldsymbol{x}_t$. In this case, let $\boldsymbol{G}(\cdot, t) = g(t)$:

$$\begin{cases} d\boldsymbol{x}_t = \boldsymbol{f}(\boldsymbol{x}_t, t)dt + g(t)d\boldsymbol{W}_t, \\ \boldsymbol{x}(0) \sim p_0(\boldsymbol{x}). \end{cases} \tag{2.9}$$

Let $\boldsymbol{x}_t \sim p_t(\boldsymbol{x})$. By the Fokker-Planck equation, we know that the probability distribution function $p_t(x)$ needs to satisfy the following equation:

$$\begin{cases} \frac{\partial}{\partial t}p_t(\boldsymbol{x}_t) = -\sum_{i=1}^{d} \frac{\partial}{\partial x_i}(\boldsymbol{f}_i(\boldsymbol{x}_t, t)p_t(\boldsymbol{x}_t)) + \frac{1}{2}\sum_{i=1}^{d}\sum_{j=1}^{d} \frac{\partial^2}{\partial x_i \partial x_j}\big[(g(t)^2 p_t(\boldsymbol{x}_t)\big], \\ p_0 \text{ is the initial distribution of } \boldsymbol{x}_0. \end{cases} \tag{2.10}$$

To generate samples from the original data, we use a backward process that reverses the forward process. Let $\boldsymbol{y}_t = \boldsymbol{x}_{T-t}$ be the reversed version of $\boldsymbol{x}_t$. Then, the backward SDE looks like this:

$$\begin{cases} d\boldsymbol{y}_t = \tilde{\boldsymbol{f}}(\boldsymbol{y}_t, t)dt + \tilde{g}(t)d\boldsymbol{W}_t, \\ \boldsymbol{y}(0) \sim p_T(\boldsymbol{y}). \end{cases} \tag{2.11}$$

By the Fokker-Planck equation and changing the time variable. Then we have corre-

sponding backward process for Eq. (2.10):

$$\begin{cases} d\boldsymbol{x}_t = \big[\boldsymbol{f}(\boldsymbol{x}_t, t) - g(t)^2 \nabla_{\boldsymbol{x}} \log p_t(\boldsymbol{x}_t)\big] dt + g(t) d\boldsymbol{W}_t, \\ \boldsymbol{x}(T) \sim p_T(\boldsymbol{x}), \end{cases} \tag{2.12}$$

where $\nabla_{\boldsymbol{x}} \log p_t(\boldsymbol{x}_t)$ is the score function, which points in the direction of data density. The back process indicates that, given a noise sample $\boldsymbol{x}_T \sim p_T(\boldsymbol{x}_T)$, which results from diffusing the original data distribution $p_0(\boldsymbol{x}_0)$, it is possible to recover the original data distribution $p_0(\boldsymbol{x}_0)$. Moreover, We can also write the backward process as an ordinary differential equation (ODE):

$$\begin{cases} d\boldsymbol{x}_t = \big[\boldsymbol{f}(\boldsymbol{x}_t, t) - \frac{1}{2} g(t)^2 \nabla_{\boldsymbol{x}} \log p_t(\boldsymbol{x}_t)\big] dt, \\ \boldsymbol{x}(T) \sim p_T(\boldsymbol{x}). \end{cases} \tag{2.13}$$

This form allows us to generate data in a deterministic way by solving an ODE backward in time. In summary, both formulations of the backward process, the SDE and ODE, aim to map noisy data distribution p_T back to the clean data p_0, each achieving this in different ways.

2.4 Denoising Score Matching with Langevin Dynamics (SMLD)

In SMLD [4], we assume that the data distribution is gradually noised by adding Gaussian noise with increasing variance. We add this noise step by step, using isotropic multivariate Gaussian distributions. We can define $\{\sigma_i\}_{i=1}^{L}$ as the noise scales with $\frac{\sigma_1}{\sigma_2} = \frac{\sigma_2}{\sigma_3} = \cdots = \frac{\sigma_{L-1}}{\sigma_L}$. Let $\boldsymbol{x}(0) \sim p(\boldsymbol{x})$ be the original data distribution, we have the noisy distribution $p_{\sigma_i}(\boldsymbol{x})$ as the convolution of $p(\boldsymbol{x})$ and the noise $p_{\sigma_i}(\tilde{\boldsymbol{x}}|\boldsymbol{x})$:

$$p_{\sigma_i}(\tilde{\boldsymbol{x}}) = \int p_{\sigma_i}(\tilde{\boldsymbol{x}}|\boldsymbol{x}) p(\boldsymbol{x}) dx, \tag{2.14}$$

where $p_{\sigma_i}(\tilde{\boldsymbol{x}}|\boldsymbol{x})$ is isotropic multivariate Gaussian distribution. Because the noise is an isotropic multivariate Gaussian distribution, we can derive the score function is $\nabla_{\boldsymbol{x}} \log p_{\sigma_i}(\boldsymbol{x}_t) = -\frac{\boldsymbol{x}_t - \boldsymbol{x}_0}{\sigma_i^2}$. This formula tells us how to move the noisy data $\tilde{\boldsymbol{x}}$ back toward the clean data $\boldsymbol{x}_0$ by following the gradient of the log-probability.

2.5 Consistency models

In DDPM and score-based diffusion models, image generation is often slow because it requires many iterative steps to sample from the model. To address this, consistency models propose learning a function that directly maps a noisy image to the original image without step-by-step sampling. In the consistency models [3], there are two main methods, consistency distillation and consistency training. In this section, we focus on the consistency training.

In the consistency training, the forward process also is modeled as an SDE like score-based diffusion model, and its corresponding backward process is an ODE. This means that for every original image, there exists a unique diffusion trajectory defined by the forward SDE and the backward ODE.

We can think of the entire diffusion process of an image as a continuous trajectory. The goal is to learn a function $\boldsymbol{f}(\cdot, t) : \mathbb{R}^d \to \mathbb{R}^d$ that satisfies self-consistency, meaning $\boldsymbol{f}(\boldsymbol{x}_t, t) = \boldsymbol{f}(\boldsymbol{x}_{t'}, t')$, so that all points along the same trajectory map back to the same original image $\boldsymbol{x}_0$. In particular, this implies $\boldsymbol{f}(\boldsymbol{x}_0, 0) = \boldsymbol{x}_0$. We aim to train a function $\boldsymbol{f}_\theta(\boldsymbol{x}_t, t)$ to predicts $\boldsymbol{f}(\boldsymbol{x}_t, t)$. A simple construction of such a function is:

$$\boldsymbol{f}_\theta(\boldsymbol{x}_t, t) = \begin{cases} \boldsymbol{x}_0 & t = 0, \\ \boldsymbol{F}_\theta(\boldsymbol{x}_t, t) & 0 < t \leq T, \end{cases} \tag{2.15}$$

where $\boldsymbol{F}_\theta(\boldsymbol{x}_t, t)$ is a neural network. Moreover, if we make $\boldsymbol{f}_\theta(\boldsymbol{x}_t, t)$ be a differentiable function, then

$$\boldsymbol{f}_\theta(\boldsymbol{x}_t, t) = C_{\text{skip}}(t)\boldsymbol{x}_t + C_{\text{out}}(t)\boldsymbol{F}_\theta(\boldsymbol{x}_t, t), \tag{2.16}$$

where $C_{\text{skip}}(t)$ and $C_{\text{out}}(t)$ are two differentiable functions such that $C_{\text{skip}}(0) = 1$ and $C_{\text{out}}(0) = 0$. The differentiable function f_θ describe the original image conbines a noisy image $\boldsymbol{x}_t$ and a denoiser $\boldsymbol{F}_\theta(\boldsymbol{x}_t, t)$. This form describes the reconstruction of the original image as a combination of the noisy image $\boldsymbol{x}_t$ and a learned denoising term $F_\theta(\boldsymbol{x}_t, t)$. During training, we compare adjacent points $\boldsymbol{x}_t$ and $\boldsymbol{x}_{t-1}$ in a trajectory to guide the optimization of F_θ. In this way, image generation can be done directly using the function f_θ without explicitly running the full backward diffusion process.

Chapter 3

Loss Function of Diffusion Models

In DDPM and score-based diffusion models, we aim to train a neural network to learn how noise is added to the images in the forward process. Since the amount of noise added at each step depends on the specific time step and the image at that moment, the input to the neural network is a time-dependent noisy image vector $\boldsymbol{x}_t$, along with a scalar time index t. The neural network, parameterized by θ, is denoted as $f_\theta(\boldsymbol{x}_t, t)$ and is designed to predict either the original data or the added noise, depending on the training objective. This model plays a central role in estimating the clean data distribution from the noisy input during the reverse process.

In DDPM, the forward process and backward process follow Eq. (2.4) and Eq. (2.7) respectively. The forward process uses a schedule of variance parameters $\beta_t \in [0.0001, 0.02]$, chosen uniformly, and splits the time T into 1000 steps. This long diffusion chain ensures the transition from data to nearly pure noise. The loss function for DDPM is defined as:

$$\text{Loss}_{\text{DDPM}}(\theta) = \mathbb{E}_{p_t(\boldsymbol{x})}\left[||f_\theta(\boldsymbol{x}_t, t) - \boldsymbol{z}_t||_2^2\right], \tag{3.1}$$

where $\boldsymbol{z}_t$ denotes the noise added at time step t and $||\cdot||_2$ is L_2-norm. It is the expected value of the squared difference between the neural network output and the noise. In practice, the Monte Carlo method is used to approximate the expected value by computing the mean of all sampled points.

In score-based diffusion models, we model two types of forward SDE:

- Forward Variance Preserving (VP) SDE:

$$d\boldsymbol{x}_t = -\frac{1}{2}\beta(t)\boldsymbol{x}_t dt + \sqrt{\beta(t)}d\boldsymbol{W}_t, \tag{3.2}$$

- Forward Variance Exploding (VE) SDE:

$$d\boldsymbol{x}_t = \sqrt{\frac{d[\sigma^2(t)]}{dt}} d\boldsymbol{W}_t, \tag{3.3}$$

where $\beta(t)$ and $\sigma(t)$ are time-dependent functions. The corresponding backward process in form of SDEs are:

- Backward VP SDE:

$$d\boldsymbol{x}_t = \left(-\frac{1}{2}\beta(t)\boldsymbol{x}_t - \beta(t)\nabla_{\boldsymbol{x}} \log p_t(\boldsymbol{x}_t)\right) dt + \sqrt{\beta(t)} d\boldsymbol{W}_t, \tag{3.4}$$

- Backward VE SDE:

$$d\boldsymbol{x}_t = \left(-\frac{d[\sigma^2(t)]}{dt}\nabla_{\boldsymbol{x}} \log p_t(\boldsymbol{x}_t)\right) dt + \sqrt{\frac{d[\sigma^2(t)]}{dt}} d\boldsymbol{W}_t, \tag{3.5}$$

where $\nabla_{\boldsymbol{x}} \log p_t(\boldsymbol{x}_t)$ is the score function. These models can be solved either numerically or analytically. In particular, the exact solutions for the two forward SDEs are known:

- VP SDE solution:

$$\boldsymbol{x}_t \sim \mathcal{N}\left(\exp\left(-\frac{1}{2}\int_0^t \beta(s)ds\right)\boldsymbol{x}_0, \left[1 - \exp\left(-\int_0^t \beta(s)ds\right)\right]\mathbb{I}\right), \tag{3.6}$$

- VE SDE solution:

$$\boldsymbol{x}_t \sim \mathcal{N}\left(\boldsymbol{x}_0, \{\sigma^2(0)[(\frac{\sigma(T)}{\sigma(0)})^{\frac{2t}{T}} - 1]\}\mathbb{I}\right). \tag{3.7}$$

These results indicate that, in the VP SDE, the noisy sample $\boldsymbol{x}_t$ is a combination of a scaled version of the original image $\boldsymbol{x}_0$ and a bounded amount of noise. In contrast, the VE setting represents $\boldsymbol{x}_t$ as the original image corrupted with noise whose variance grows over time. In practice, instead of simulating the SDE forward process step-by-step, we directly sample noisy images using these closed-form solutions, which is computationally efficient.

As the number of DDPM time steps $N \to \infty$, discrete forward process Eq. (2.1) converges to the continuous VP SDE Eq. (3.2), revealing a deep connection between DDPMs and score-based diffusion models. For matching the time schedule of DDPM, To compare the two models, we use a linearly increasing beta schedule in score-based diffusion models, such as $\beta(t) = 0.1 + \frac{(20-0.1)}{T}t$ where $T = 1$. In SMLD, where Eq. (3.3) corresponds to the VE SDE, we can discretize time $[0, T]$ into N intervals with $\Delta t = \frac{T}{N}$

and use the Euler-Maruyama method. We have the following discretized SDE:

$$\boldsymbol{x}_{t+1} = \boldsymbol{x}_t + \sqrt{\sigma^2(t+1) - \sigma^2(t)}\boldsymbol{z}_t, \tag{3.8}$$

where $\boldsymbol{z}_t \sim \mathcal{N}(0, I)$. For convenience, we typically choose $\sigma(0) = 0.01$, $\sigma(T) = 50$ and $T = 1$.

In the backward process of score-based diffusion model, we need to learn the score function $\nabla_{\boldsymbol{x}} \log p_t(\boldsymbol{x}_t)$ and define the Explicit Score Matching (ESM)[5]:

$$J_{\mathrm{ESM},p_t}(\theta) = \mathbb{E}_{p_t(\boldsymbol{x})}\Big[||f_\theta(\boldsymbol{x}_t, t) - \nabla_{\boldsymbol{x}} \log p_t(\boldsymbol{x}_t)||_2^2\Big], \tag{3.9}$$

where $p_t(\boldsymbol{x})$ is the probability distribution function of $\boldsymbol{x}_t$ at time t and representing the expected value of the squared difference between the neural network output and the score function. It is straightforward learning the score function by minimizing the ESM loss function. However, we typically do not know the score function of the unknown dataset $\nabla_{\boldsymbol{x}} \log p_t(\boldsymbol{x}_t)$.

Now, we define another loss function to predict the noise called Denoising Score Matching (DSM):

$$J_{\mathrm{DSM},p_t}(\theta) = \mathbb{E}_{p_{t0}(\boldsymbol{x})}\Big[||f_\theta(\boldsymbol{x}_t, t) - \nabla_{\boldsymbol{x}} \log p_t(\boldsymbol{x}_t|\boldsymbol{x}_0)||_2^2\Big], \tag{3.10}$$

where p_{t0} is the joint probability function of $\boldsymbol{x}_0$, $\boldsymbol{x}_t$. It predicts the noise through the joint probability distribution of the noisy data $\boldsymbol{x}_t$ at time t and the original data $\boldsymbol{x}_0$ at time 0. These two objectives are theoretically connected [6]:

$$J_{\mathrm{ESM},p_t}(\theta) = J_{\mathrm{DSM},p_t}(\theta) + C, \tag{3.11}$$

where C is a constant. This relation confirms that learning to denoise is equivalent to learning the score function. As such, the loss function commonly used in training score-based diffusion models is:

$$\mathrm{Loss}_{\mathrm{SBD}}(\theta) = \mathbb{E}_{p_t(\boldsymbol{x})}\left[||f_\theta(\boldsymbol{x}_t, t) - \boldsymbol{z}_t||_2^2\right]. \tag{3.12}$$

In consistency models, we define a non-uniform time schedule using a hyperparameter $\rho = 7$, minimun time $\epsilon = 0.002$, and total time $T = 80$. The time t at step i follows the function $t(i) = \epsilon^{\frac{1}{\rho}} + \frac{i-1}{N-1}(T^{\frac{1}{\rho}} + \epsilon^{\frac{1}{\rho}})^\rho$ for $\rho = 7$. As time increases, more noise is added to the input. The Loss function for consistency models is defined as:

$$\mathrm{Loss}_{\mathrm{CT}}(\theta) = \mathbb{E}_{p_t(\boldsymbol{x})}\left[||f_\theta(\boldsymbol{x}_t, t) - f_{\theta^-}(\boldsymbol{x}_{t+1}, t+1)||_2^2\right], \tag{3.13}$$

or

$$\text{Loss}_{\text{CT}}(\theta) = \text{LPIPS}\big(f_\theta(\boldsymbol{x}_t, t), f_{\theta^-}(\boldsymbol{x}_{t+1}, t+1))\big), \tag{3.14}$$

where $f_\theta(\cdot,\cdot)$, $f_{\theta^-}(\cdot,\cdot)$ are the neural network with parameter θ, θ^-, respectively, and LPIPS (Learned Perceptual Image Patch Similarity) [7] is defined as:

$$d(x, x_0) = \sum_l \frac{1}{H_l W_l} \sum_{h,w} \|\omega_l \odot (\hat{y}^l_{hw} - \hat{y}^l_{0hw})\|_2^2, \tag{3.15}$$

where $\hat{y}^l, \hat{y}^l_0 \in \mathbb{R}^{H_l \times W_l \times C_l}$ ($\hat{y}^l_{hw}, \hat{y}^l_{0hw} \in \mathbb{R}^{C_l}$) for layer l, $\omega^l \in \mathbb{R}^{C_l}$ is a vector for activations channel-wise and compute the ℓ_2 distance. This method measures image similarity by computing perceptual loss using features from a trained neural network. Note that θ^- is updated by exponential moving average, $\theta^- = (1-\mu)\theta^- + \mu\theta$, where $0 \leq \mu < 1$ is a decay rate, instead of by gradient descent. That is, only θ is being trained during optimization step. Although the consistency model is faster in sampling, image quality is not always as good as DDPM and score-based diffusion model. We can sampe by multi-step sampling to get better performance.

Chapter 4

Experiments

This chapter presents the image generation results obtained using DDPM, VP SDE, VE SDE, and consistency training across three datasets: MNIST, CIFAR-10, and Cat Faces [8].

- The MNIST dataset contains 60,000 grayscale handwritten digit images with a resolution of 28×28 pixels.
- The CIFAR-10 dataset consists of 50,000 color images, including categories such as cars, dogs, etc., with a resolution of 32×32 pixels.
- The Cat Faces dataset includes approximately 30,000 color images of cat faces, each with a resolution of 64×64 pixels.

To evaluate the performance of image generation, we use the Inception score (IS) [9] and Fréchet Inception Distance (FID) [10] as the primary metric. IS measures the quality and diversity of generated images. It is calculated as follows:

$$\mathrm{IS}(\boldsymbol{x}) = \exp\left(\mathbb{E}_{\boldsymbol{x}}\left[\sum_{y} p(y|\boldsymbol{x}) \ln \frac{p(y|\boldsymbol{x})}{p(y)})\right]\right), \tag{4.1}$$

where $p(y|\boldsymbol{x})$ is the conditional probability being the given object and $p(y)$ is the marginal probability that the given image is real, A higher IS indicates better quality and diversity of generated images. FID measures the difference between the distributions of generated images and real images in the feature space. Let $\boldsymbol{x}$ and $\boldsymbol{y}$ be the feature vectors of the generated images and real images, respectively. It is calculated as follows:

$$\mathrm{FID}(\boldsymbol{x}, \boldsymbol{y}) = ||\mu_{\boldsymbol{x}} - \mu_{\boldsymbol{y}}||_2^2 + \mathrm{Tr}(\Sigma_{\boldsymbol{x}} + \Sigma_{\boldsymbol{y}} - 2(\Sigma_{\boldsymbol{x}}\Sigma_{\boldsymbol{y}})^{1/2}), \tag{4.2}$$

where $\mu_{\boldsymbol{x}} \in \mathbb{R}^d$ and $\mu_{\boldsymbol{y}} \in \mathbb{R}^d$ are the mean of $\boldsymbol{x}$ and $\boldsymbol{y}$, respectively, and $\Sigma_{\boldsymbol{x}} \in \mathbb{R}^{d \times d}$ and $\Sigma_{\boldsymbol{y}} \in \mathbb{R}^{d \times d}$ are their corresponding covariances matrices. FID is a metric for comparing

two probability distributions, where a lower score indicates better performance and higher similarity between generated and real images. We display 64 sample images generated by each model and calculate the FID score using 10,000 generated images per method. Additionally, we report the Number of Function Evaluations (NFE), which corresponds to the number of steps used during the reverse sampling process, and measure measure the average CPU runtime per sample, calculated as the total sampling time divided by the number of generated samples. We use the U-net construct [11] and Adam optimizer running on an NVIDIA GeForce RTX 4090.

4.1 MNIST

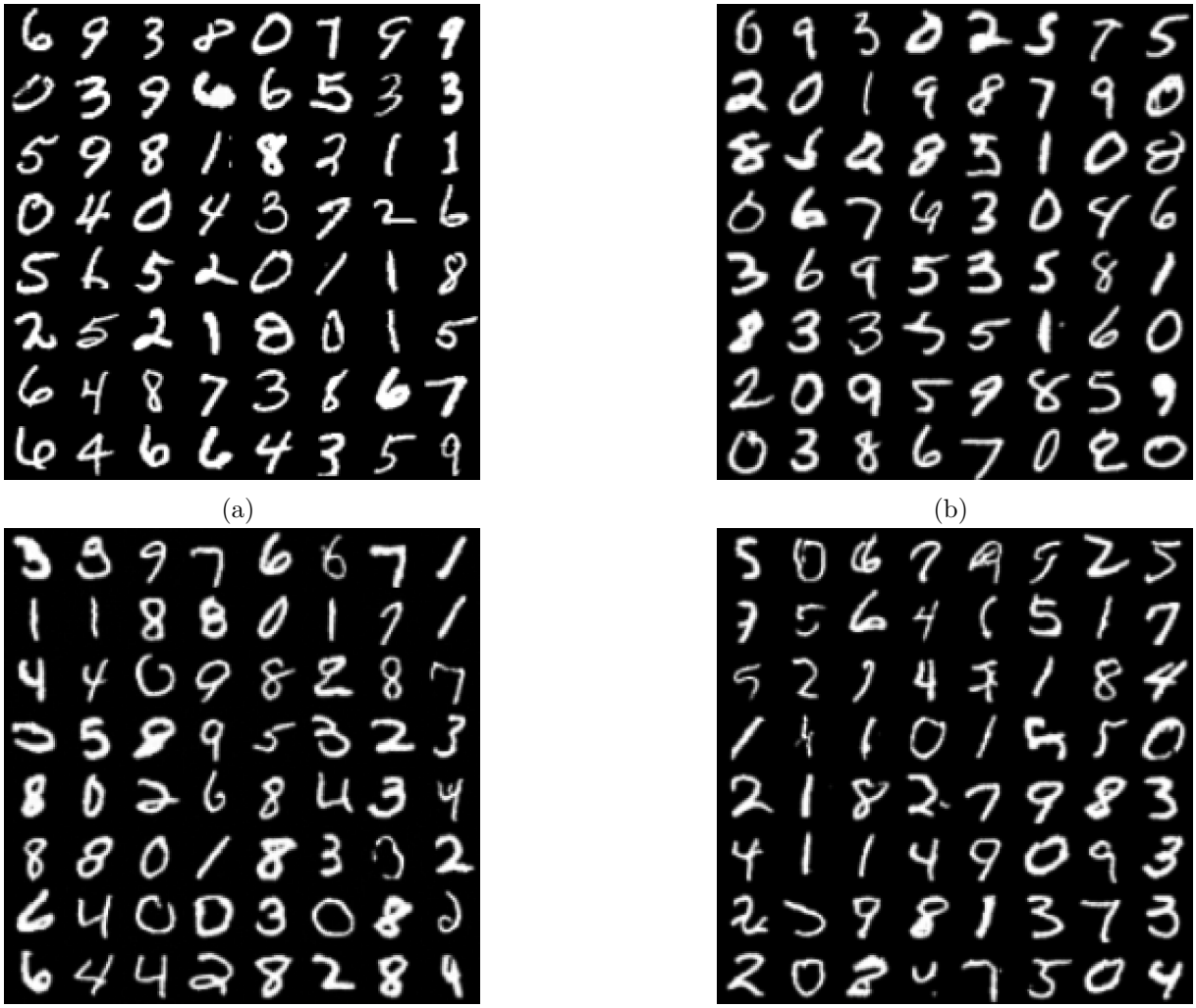
Fig. 4.1 shows samples generated on the MNIST dataset by DDPM, VP SDE, and VE SDE. All three models were trained for 100 epochs with a batch size of 128, using $N = 1000$ forward and backward diffusion steps.

Fig. 4.2 shows samples produced by consistency training with the ℓ_2 norm and LPIPS losses. These models were also trained for 100 epochs at batch size 128, but with $N = 80$ discretization steps and a decay rate $\mu = 0.9$. We compare two sampling strategies: a single-step method at the final time point, and a two-step method using a uniform time schedule. In panels (b) and (c), the ℓ_2 norm outputs appear noticeably blurrier than those using LPIPS; in panels (d) and (e), two-step sampling yields sharper images than the single-step approach.

Table 4.1 summarizes FID scores on MNIST for all methods. DDPM achieves the lowest FID, outperforming VP SDE and VE SDE under the same number of steps. Consistency training with the ℓ_2 norm yields higher FID than the diffusion models, whereas consistency training with LPIPS attains FID values comparable to DDPM, VP SDE, and VE SDE. All methods yield similar IS on the MNIST dataset, likely because the images are too simple and lack sufficient diversity. CT achieves faster sampling times than other diffusion models when using a reduced number of sampling steps.

Table 4.1: FID scores on the MNIST dataset for each method.

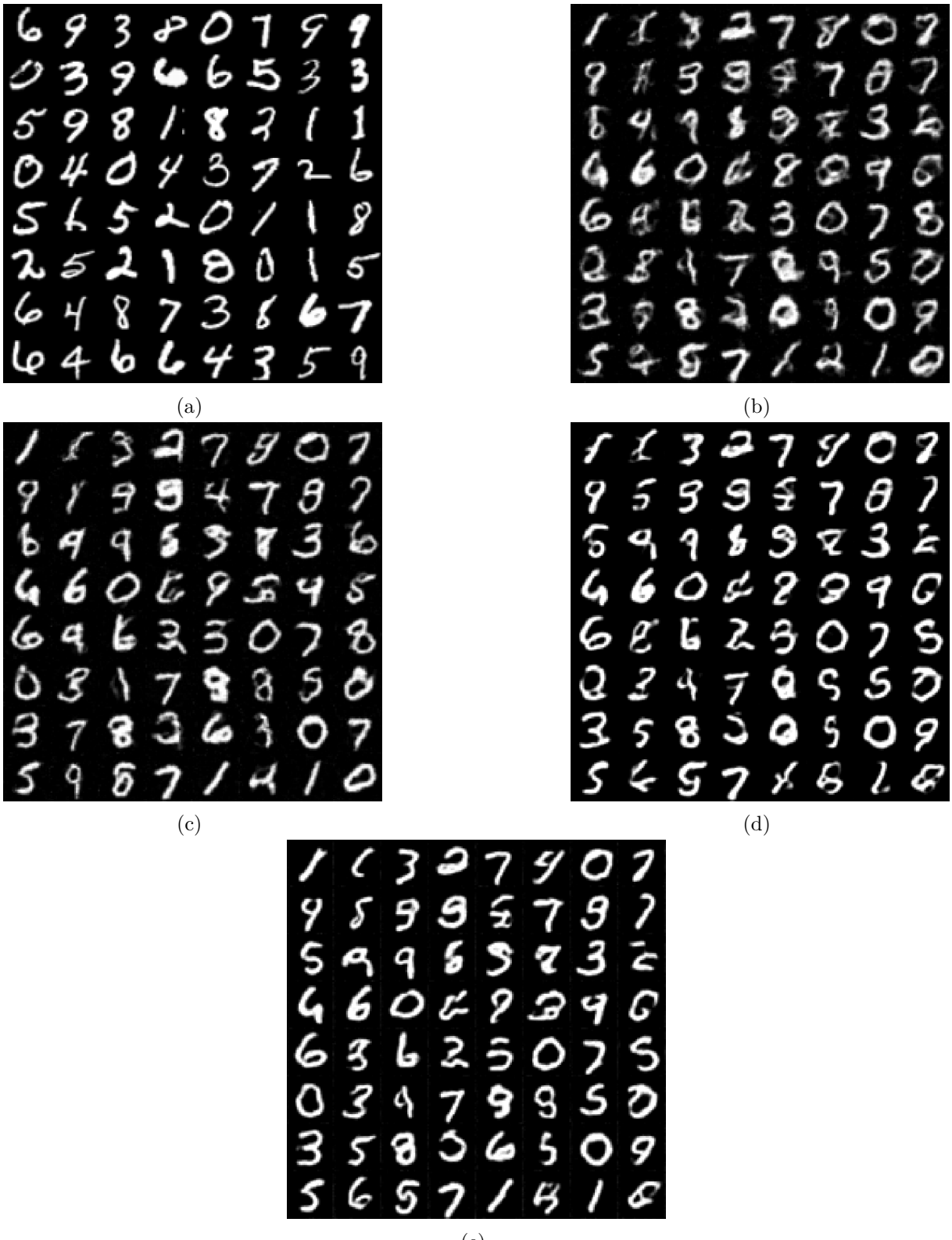
Method	DDPM	VP SDE	VE SDE	CT (ℓ_2 norm)		CT (LPIPS)	
NFE	1000	1000	1000	1	2	1	2
CPU time(sec)	0.41	0.41	0.41	0.01	0.01	0.01	0.01
IS	2.24	2.26	2.29	2.27	2.33	2.29	2.23
FID	34.16	36.53	35.40	85.70	62.79	37.59	35.41



(a) (b)

(c) (d)

Figure 4.1: (a) Original MNIST images. (b-d) Samples generated on MNIST by DDPM, VP SDE, and VE SDE. Note that some generated images do not resemble typical handwritten digits.



(a) (b) (c) (d) (e)

Figure 4.2: (a) Original MNIST images. (b) and (c) Samples generated by consistency training using the ℓ_2 norm with single-step and two-step sampling, respectively. (d) and (e) Samples generated by consistency training using LPIPS with single-step and two-step sampling, respectively.

4.2 CIFAR-10

Fig. 4.3 shows samples generated on the CIFAR-10 dataset by DDPM, VP SDE, and VE SDE. All three models were trained for 200 epochs with a batch size of 128, using $N = 1000$ forward and reverse diffusion steps. Panels (b) and (c) demonstrate that DDPM produces sharp, realistic textures, whereas panels (d) and (e) reveal that VP SDE and VE SDE outputs appear more blurred and exhibit color casts.

(a) (b)

(c) (d)

Figure 4.3: (a) Original CIFAR-10 images. (b-d) Samples generated by DDPM, VP SDE, and VE SDE, respectively.

Fig. 4.4 and Fig. 4.5 compare consistency training with the ℓ_2 norm and LPIPS losses under two sampling strategies. In both cases, models were trained for 200 epochs at batch size 128.

- In Fig. 4.4, a fixed configuration is used with $N = 120$ steps, $\mu = 0.9$.

- In Fig. 4.5, the number of steps N and EMA decay rate μ vary over training iterations according to schedule functions: $N(k) = \left\lceil \sqrt{\frac{k}{K}((s_1+1)^2 - s_0) + s_0^2} - 1 \right\rceil + 1$, $\mu(k) = \exp(\frac{s_0 \log \mu_0}{N(k)})$,

where k is the current training iteration, K is the total number of iterations, $s_0 = 2$ is the initial discretization steps, $s_1 = 150$ is the target discretization steps, and $\mu_0 = 0.9$ is initial EMA decay rate. Across both figures, ℓ_2 norm outputs lack discernible detail, while LPIPS samples preserve finer structure. Two-step sampling does not yield a clearer result compared to single-step sampling.

Table 4.2 reports FID scores for the diffusion models: DDPM achieves the lowest FID, outperforming VP SDE and VE SDE. Table 4.3 lists FID results for consistency training: LPIPS consistently outperforms the ℓ_2 norm, and models using the schedule function achieve better fidelity than those with a fixed N and μ. In both tables, DDPM, VP SDE, and VE SDE attain higher IS than consistency training, indicating that the FID and IS are consistent with each other in this dataset. In terms of CPU runtime, consistency training samples far more quickly than the diffusion models, owing to its reduced number of sampling steps.

Table 4.2: FID score on CIFAR-10 dataset by DDPM, VP SDE, and VE SDE.

Method	DDPM	VP SDE	VE SDE
NFE	1000	1000	1000
CPU time(sec)	1.36	1.37	1.36
IS	5.71	4.34	5.87
FID	27.45	62.86	35.45

Table 4.3: FID score on CIFAR-10 dataset by consistency training.

Method	CT (ℓ_2 norm)				CT (LPIPS)			
(N, μ)	(120,0.9)		schedule function		(120,0.9)		schedule function	
NFE	1	2	1	2	1	2	1	2
CPU time	0.01	0.01	0.01	0.01	0.01	0.01	0.01	0.01
IS	1.36	1.36	1.85	1.62	3.50	3.50	3.99	3.98
FID	273.26	271.37	264.40	233.18	66.53	66.24	53.10	51.40

(a) (b) (c) (d) (e)

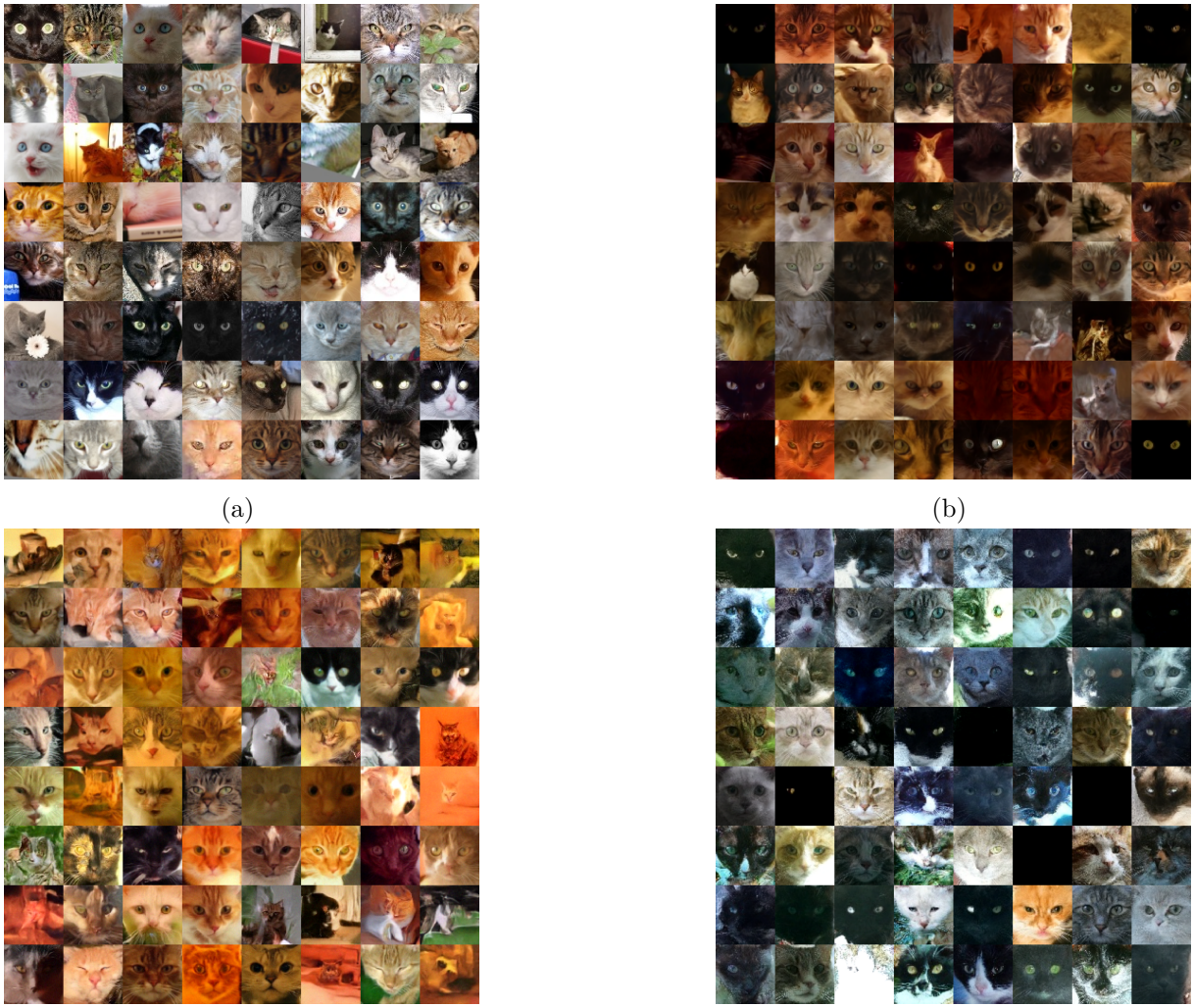
Figure 4.4: (a) Original CIFAR-10 images. (b-e) Samples generated by consistency training with $N = 120$ steps and $\mu = 0.9$. (b) and (c) Samples using ℓ_2 norm with single-step and two-step sampling, respectively. (d) and (e) Samples using LPIPS with single-step and two-step sampling, respectively.

(a) (b) (c) (d) (e)

Figure 4.5: (a) Original CIFAR-10 images. (b-e) Samples generated by consistency training with N steps, μ following schedule functions. (b) and (c) Samples using ℓ_2 norm with single-step and two-step sampling, respectively. (d) and (e) Samples using LPIPS with single-step and two-step sampling, respectively.

4.3 Cat Faces

Fig. 4.6 shows samples generated on the Cat Faces dataset by DDPM, VP SDE, and VE SDE. All three models were trained for 200 epochs with a batch size of 16, using $N = 1000$ diffusion steps in both the forward and backward processes. Panels (b)-(d) indicate that DDPM yields the most realistic textures, while VP SDE and VE SDE produce blurrier outputs with minor color shifts.



(a) (b) (c) (d)

Figure 4.6: (a) Original Cat Faces images. (b-d) Samples generated on the Cat Faces dataset by DDPM, VP SDE, and VE SDE, respectively. Some generated images exhibit minor artifacts and color casts.

Fig. 4.7 and Fig. 4.8 compare consistency training under two scheduling schemes, using the ℓ_2 norm and LPIPS losses. Both sets of models were trained for 100 epochs with batch size 16:

- $N = 120$ steps, $\mu = 0.95$ in Fig. 4.7.

- N steps, μ following schedule function in Fig. 4.8.

In both figures, outputs using the ℓ_2 norm (panels (b) and (c)) appear overly smooth and lack detail, whereas LPIPS-enhanced samples (panels (d) and (e)) preserve finer facial features but occasionally look artificially sharp. Two-step sampling does not produce noticeably clearer images compared to single-step sampling in either setting.

Table 4.4 reports FID scores for the diffusion models: DDPM achieves the lowest FID, outperforming VP SDE and VE SDE. Table 4.5 presents FID results for consistency training: the LPIPS loss consistently outperforms the ℓ_2 norm, and models trained with schedule function attain better fidelity than those using a fixed N and μ. CT achieves faster sampling times than other diffusion models when using a reduced number of sampling steps. In both tables, DDPM, VP SDE, and VE SDE attain higher IS than consistency training, indicating that the FID and IS are consistent with each other in this dataset. In terms of CPU runtime, consistency training samples far more quickly than the diffusion models, owing to its reduced number of sampling steps.

Table 4.4: FID score on Cat Faces dataset by DDPM, VP SDE, VE SDE.

Method	DDPM	VP SDE	VE SDE
NFE	1000	1000	1000
CPU time(sec)	2.98	2.98	2.98
IS	3.65	3.40	4.02
FID	30.10	38.54	43.87

Table 4.5: FID score on Cat Faces dataset by consistency training.

Method	CT (ℓ_2 norm)				CT (LPIPS)			
(N, μ)	(120,0.95)		schedule function		(120,0.95)		schedule function	
NFE	1	2	1	2	1	2	1	2
CPU time	0.01	0.02	0.01	0.01	0.01	0.02	0.01	0.02
IS	1.54	1.61	1.64	1.59	2.75	2.68	2.68	2.69
FID	270.28	264.31	286.13	284.63	33.29	31.57	37.84	38.11

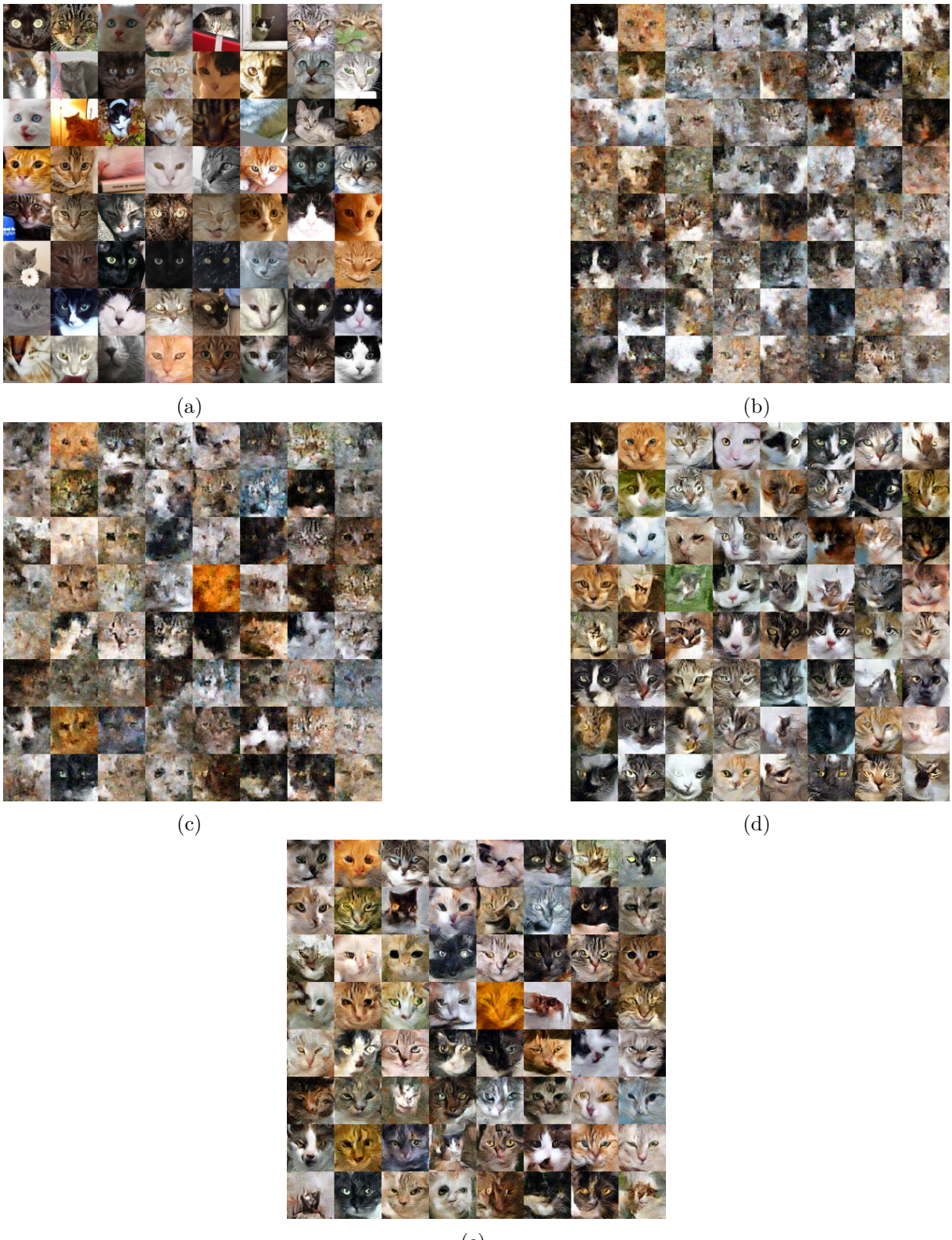


Figure 4.7: (a) Original Cat Faces images. (b-e) Samples generated by consistency training with $N = 120$ steps and $\mu = 0.95$. (b) and (c) Samples using ℓ_2 norm with single-step and two-step sampling, respectively. (d) and (e) Samples using LPIPS with single-step and two-step sampling, respectively.

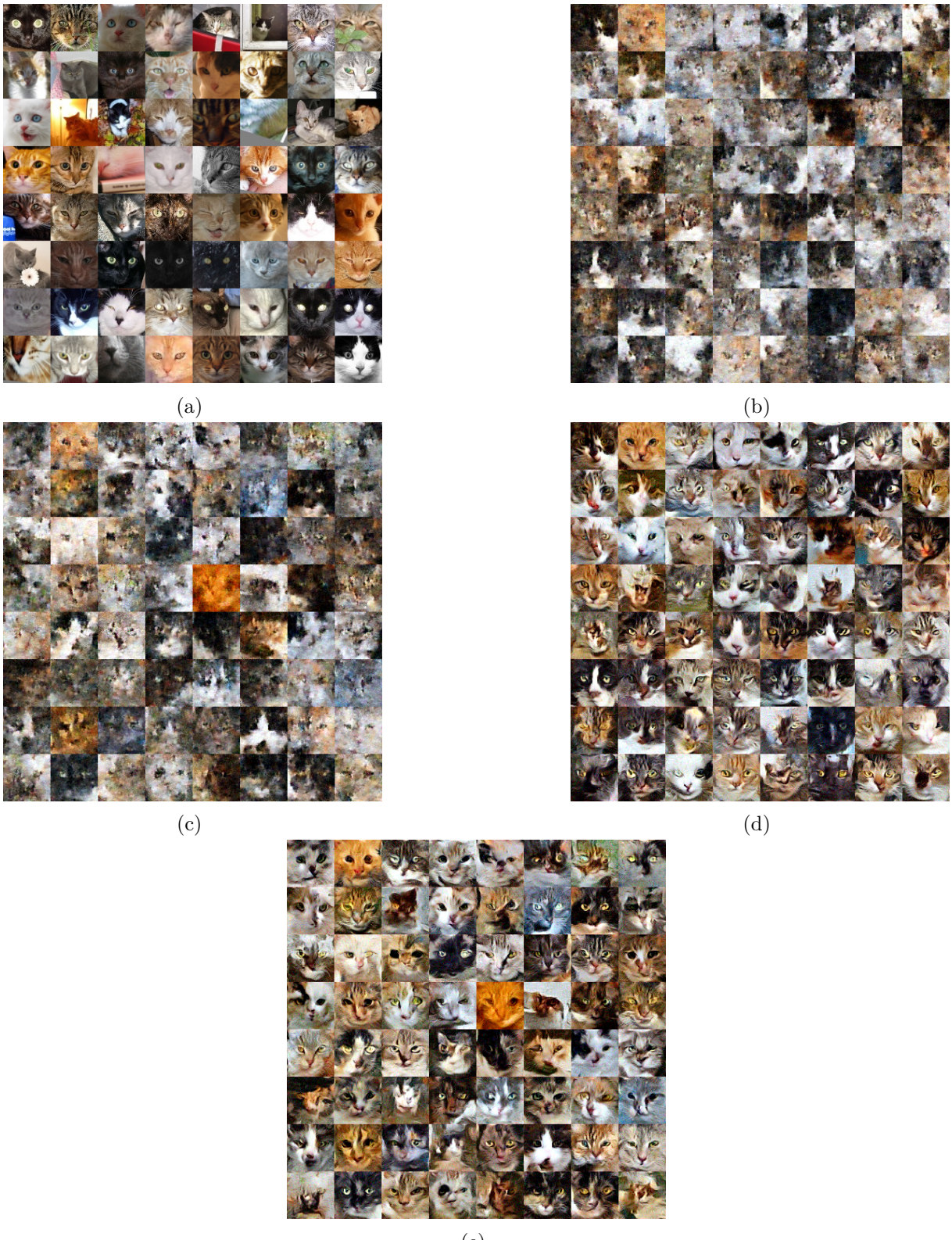


Figure 4.8: (a) Original Cat Faces images. (b-e) Samples generated by consistency training with N steps, μ following schedule functions. (b) and (c) Samples using ℓ_2 norm with single-step and two-step sampling, respectively. (d) and (e) Samples using LPIPS with single-step and two-step sampling, respectively.

Chapter 5

Conclusion

In this thesis, we investigated three classes of image generative models: Denoising Diffusion Probabilistic Models (DDPM), score-based generative models, and consistency models. While the forward processes of DDPM and SMLD can be interpreted as discrete versions of score-based generative models governed by different SDEs, their primary distinction lies in the design of the reverse process.

Experimental results indicate that DDPM generally produces higher-quality samples than score-based models when using the same number of sampling steps. On MNIST and CIFAR-10, however, SMLD yields better samples than DDPM, whereas on the Cat Faces dataset, DDPM performs better.

Consistency models, a recent class of generative models, offer a substantial advantage in sampling efficiency. In our experiments, consistency models required approximately one minute to generate 10,000 samples, compared to around two hours for DDPM and score-based models. Despite the improved efficiency, the quality of samples generated by consistency models on MNIST and Cat Faces is comparable to those from DDPM and SMLD. In our experiments, we observed that uniform two-step sampling does not consistently outperform one-step sampling. In fact, increasing the number of sampling steps did not consistently improve the results, and the choice of the second step remains an important factor deserving further study. LPIPS was found to be an effective metric for evaluating perceptual similarity in this context.

Our results do not fully match those reported in the literature, which is likely due to architectural differences. Although all models were based on a U-Net backbone, variations in layer types (e.g., attention or pooling layers) and network width may account for the discrepancy.

Chapter 6

Bibliography

[1] J. Ho, A. Jain, and P. Abbeel, "Denoising diffusion probabilistic models," in *Advances in Neural Information Processing Systems*, H. Larochelle, M. Ranzato, R. Hadsell, M. Balcan, and H. Lin, Eds., vol. 33, Curran Associates, Inc., 2020, pp. 6840–6851.

[2] Y. Song, J. Sohl-Dickstein, D. P. Kingma, A. Kumar, S. Ermon, and B. Poole, "Score-based generative modeling through stochastic differential equations," in *International Conference on Learning Representations*, 2021.

[3] Y. Song, P. Dhariwal, M. Chen, and I. Sutskever, "Consistency models," in *International Conference on Machine Learning*, PMLR, 2023, pp. 32 211–32 252.

[4] Y. Song and S. Ermon, "Generative modeling by estimating gradients of the data distribution," *Advances in neural information processing systems*, vol. 32, 2019.

[5] A. Hyvärinen, "Estimation of non-normalized statistical models by score matching," *J. Mach. Learn. Res.*, vol. 6, pp. 695–709, Dec. 2005, ISSN: 1532-4435.

[6] P. Vincent, "A connection between score matching and denoising autoencoders," *Neural Computation*, vol. 23, no. 7, pp. 1661–1674, 2011. DOI: 10.1162/NECO_a_00142.

[7] R. Zhang, P. Isola, A. A. Efros, E. Shechtman, and O. Wang, "The unreasonable effectiveness of deep features as a perceptual metric," in *2018 IEEE/CVF Conference on Computer Vision and Pattern Recognition*, 2018, pp. 586–595. DOI: 10.1109/CVPR.2018.00068.

[8] F. Ferlito, *Cat-faces-dataset*, 2019. [Online]. Available: https://github.com/fferlito/Cat-faces-dataset.

[9] T. Salimans et al., "Improved techniques for training gans," in *Advances in Neural Information Processing Systems*, D. Lee, M. Sugiyama, U. Luxburg, I. Guyon, and R. Garnett, Eds., vol. 29, Curran Associates, Inc., 2016. [Online]. Available: https://proceedings.neurips.cc/paper_files/paper/2016/file/8a3363abe792db2d8761d6403605aeb7-Paper.pdf.

[10] M. Heusel, H. Ramsauer, T. Unterthiner, B. Nessler, and S. Hochreiter, "Gans trained by a two time-scale update rule converge to a local nash equilibrium," *Advances in neural information processing systems*, vol. 30, 2017.

[11] O. Ronneberger, P. Fischer, and T. Brox, "U-net: Convolutional networks for biomedical image segmentation," in *Medical image computing and computer-assisted intervention–MICCAI 2015: 18th international conference, Munich, Germany, October 5-9, 2015, proceedings, part III 18*, Springer, 2015, pp. 234–241.