# Review of Denoising Diffusion Probabilistic Models

Minji Kim Hyeon Lee
University of North Carolina, Chapel Hill

February 19, 2024

## 1 Introduction

What do generative models do? In the machine learning literature, "generation" refers to sampling $X$ from the ground truth probability distribution of the target data $\mathbb{P}_{\text{data}}$,

$$X \sim \mathbb{P}_{\text{data}},$$

obtaining a data point that was not observed or used during the learning process. In practice, $X$ usually represents complicated data such as images, texts, audios, or even movies. An important class of generative models is *probabilistic generative models*; they aim to learn the data distribution itself, finding a parameterized function

$$f_{\boldsymbol{\theta}}(Z) \sim \mathbb{P}_{\text{data}},$$

where $Z$ is a random variable from a noise space. In other words, it aims to learn a mapping from $Z$ to a random variable of the data distribution.

There exist other types of generative models depending on the tasks. For example, *non-probabilistic generative models* such as Generative Adversarial Networks (GANs) (Goodfellow et al. (2014)) can be used when one requires samples from the population distribution but the distribution itself is not of interest. On the other hand, one may wish to request the model to generate images based on certain descriptions; these procedure is called *conditional generation*. The last example is *sequential generation*, which is useful for generating long text data or generating musics. In this report, we restrict our attention to the ground problem to generate samples just from the data distribution $\mathbb{P}_{\text{data}}$.

A Diffusion model is a type of probabilistic generative models which has garnered considerable attention from researchers due to its capacity for generating high-quality samples comparable to those produced by Generative Adversarial Networks (GANs), while also generating a diversity of samples akin to Variation Autoencoders (VAE) (Kingma and Welling (2014)) and Normalizing Flows (Rezende and Mohamed (2015)). Denoising diffusion models were initially introduced by Sohl-Dickstein et al. (2015), and early related work focusing on score-matching was developed by

Song and Ermon (2019). Ho et al. (2020) significantly advanced this field by successfully producing high-quality training results in generative imaging.

In this report, we aim to deliver what are the diffusion models and how variational inference is employed in its learning procedure. To understand diffusion models' strengths and weaknesses, we also review core ideas of the other three popular generative models mentioned above. Figure 1 summarizes their relative strengths and weaknesses, which will become clear as we learn their model structures in the later sections. Most of the contents in this report is derived from these papers and Prince (2023).

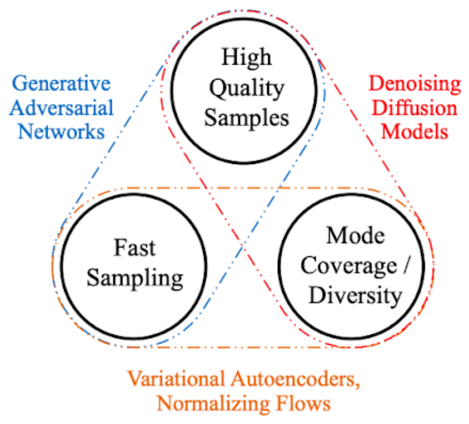
High Quality Samples
Generative Adversarial Networks
Denoising Diffusion Models
Fast Sampling
Mode Coverage / Diversity
Variational Autoencoders, Normalizing Flows

Figure 1: Strengths and Weaknesses of Generative Models, (Image from NVIDIA, link)

This report is organized as follows. We first discuss performance measures illustrated in Figure 1, and briefly review the core ideas of GAN, VAE, and normalizing flows. Section 3 provides a comprehensive overview and introduce the structure of the diffusion probabilistic model. Section 4 summarizes statistical techniques used for efficient optimization, while Section 5 delves into the reparameterization strategy that facilitates effective training and sampling algorithms. Additionally, we include special notes on two key aspects: First, we examine what information about the data distribution the model learns through the denoising process, revealing that it primarily learns the score function of the input data distribution. Second, we discuss how diffusion models approximate the solution to a specific stochastic differential equation.

## 2 Related Works

### 2.1 Performance measure for generative models

Goodness of generative models is subjective and there are different criteria depending on the specific tasks. However, four performance measures listed below are most fundamental and applicable to most of the generative models.

- **High Quality Samples**: The generated samples should be indistinguishable from the data that the model learned.

- **Fast Sampling** Once the model was trained, generating new samples from the model should be computationally efficient.
- **Mode Coverage/Diversity** The model should generate new samples from the entire data space, rather than drawing samples from a relatively small region of the data space.
- **Probabilistic Model** It is a strength for models to learn probability of data points, as the probability can be used for further analysis such as anomaly detection.

As we will see in the later sections, GAN is a non-probabilistic model and all other three models are probabilistic models. As the opposite of the term mode coverage, the phenomenon where a trained model generates samples from a small part of the data space is called *mode collapse*; GAN is well-known for its susceptibility to model collapse, because it aims to generate samples that are not distinguishable from the training data set without accounting for how the training data points are likely to occur within the model. In contrast, VAE and normalizing flows are fast and evenly generate samples from the data space, but they are limited to generate high quality samples due to their simplicity of the model and the latent space, respectively. A diffusion model lies in between them in the sense that it is capable of generating high quality samples from the entire data space, at the cost of higher computational expense.

## 2.2 Generative Adversarial Network

A *Generative Adversarial Network (GAN)* comprises two machines, a *generator* and a *discriminator*. A generator $\mathbf{g}(\boldsymbol{z}, \boldsymbol{\theta})$ takes a random noise $\boldsymbol{z}$ from a base distribution and generates a fake data $\boldsymbol{x} = \mathbf{g}(\boldsymbol{z}, \boldsymbol{\theta})$. A discriminator $\mathbf{f}(\boldsymbol{x}, \boldsymbol{\phi})$ takes either a real or a fake data $\boldsymbol{x}$ and returns the logit of the probability that the data comes from the real observations. Two machines go into a 2-player game: a discriminator tries to best classify inputs into real and fake observations, and a generator tries to forge new observations so the discriminator cannot distinguish them from the real data. Suppose we adopt the Bernoulli log-likelihood as the loss, that is,

$$l\left(\widehat{p}, y\right) = -(1-y)\log(1-\widehat{p}) - y\log(\widehat{p}) \tag{1}$$

where $y = 0$ if the data was real and $y = 1$ if the data was fake.

In each iteration of training, the discriminator learns the parameter $\boldsymbol{\phi}$ from the set of real observations $\{\boldsymbol{x}_1, \cdots, \boldsymbol{x}_n\}$ and the set of fake observations $\{\boldsymbol{x}_1^*, \cdots, \boldsymbol{x}_m^*\}$ by minimizing the average loss as follows.

$$\widehat{\boldsymbol{\phi}} := \arg\min_{\boldsymbol{\phi}} \left\{ \sum_{j=1}^{m} -\log\left[1 - \operatorname{sign}\left(\mathbf{f}(\boldsymbol{x}_j^*, \boldsymbol{\phi})\right)\right] - \sum_{i=1}^{n} \log\left[\operatorname{sign}\left(\mathbf{f}(\boldsymbol{x}_i, \boldsymbol{\phi})\right)\right] \right\} \tag{2}$$

The generator learns the parameter $\boldsymbol{\theta}$ by maximizing the minimum loss that a discriminator can yield:

$$\widehat{\boldsymbol{\theta}} := \arg\max_{\boldsymbol{\theta}} \left\{ \min_{\boldsymbol{\phi}} \left\{ \sum_{j=1}^{m} -\log\Big[1 - \operatorname{sign}\left(\mathbf{f}(\mathbf{g}(\boldsymbol{z}_j, \boldsymbol{\theta}), \boldsymbol{\phi})\right)\Big] - \sum_{i=1}^{n} \log\Big[\operatorname{sign}\left(\mathbf{f}(\boldsymbol{x}_i, \boldsymbol{\phi})\right)\Big] \right\} \right\}. \tag{3}$$

## 2.3 Normalizing Flow

The purpose of *normalizing flow* is to learn a function $\mathbf{f}(\cdot, \boldsymbol{\phi})$ such that the distribution of $\mathbf{f}(\boldsymbol{Z}, \boldsymbol{\phi})$ best mimics the sample distribution, where $\boldsymbol{Z}$ follows a pre-specified distribution which is usually chosen as standard multivariate normal distribution. If $\mathbf{f}(\cdot, \boldsymbol{\phi})$ is one-to-one, the density of $\boldsymbol{X} = \mathbf{f}(\boldsymbol{Z}, \boldsymbol{\phi})$ is given as

$$\mathbf{f}_{\boldsymbol{X}}(\boldsymbol{x}) = \left| \frac{\partial \mathbf{f}(\boldsymbol{z}, \boldsymbol{\phi})}{\partial \boldsymbol{z}} \right|^{-1} \mathbf{f}_{\boldsymbol{Z}}(\boldsymbol{z}), \boldsymbol{z} = \mathbf{f}^{-1}(\boldsymbol{x}, \boldsymbol{\phi}). \tag{4}$$

We estimate $\boldsymbol{\phi}$ by maximizing likelihood of observed data $\{\boldsymbol{x}_1, \cdots, \boldsymbol{x}_n\}$. For $\boldsymbol{z}_i = \mathbf{f}^{-1}(\boldsymbol{x}_i, \boldsymbol{\phi}), i = 1, \cdots, n$,

$$\begin{aligned} \widehat{\boldsymbol{\phi}} :&= \arg\max_{\boldsymbol{\phi}} \left\{ \sum_{i=1}^{n} \log\left( \left| \frac{\partial \mathbf{f}(\boldsymbol{z}_i, \boldsymbol{\phi})}{\partial \boldsymbol{z}_i} \right|^{-1} \mathbf{f}_{\boldsymbol{Z}}(\boldsymbol{z}_i) \right) \right\} \\ &= \arg\min_{\boldsymbol{\phi}} \left\{ \sum_{i=1}^{n} \log\left( \left| \frac{\partial \mathbf{f}(\boldsymbol{z}_i, \boldsymbol{\phi})}{\partial \boldsymbol{z}_i} \right| \right) \right\} \end{aligned} \tag{5}$$

An important assumption for this formulation is that $\mathbf{f}(\cdot, \boldsymbol{\phi})$ is invertible. Because it is difficult to maintain a function being invertible while allowing it to be flexible at the same time, we factor the function as the composition of sequence of simple, invertible functions.

$$\boldsymbol{x} = \mathbf{f}(\boldsymbol{z}, \boldsymbol{\phi}) = \mathbf{f}_K\left(\mathbf{f}_{K-1}\left(\cdots \mathbf{f}_2\left(\mathbf{f}_1(\boldsymbol{z}, \boldsymbol{\phi}_1), \boldsymbol{\phi}_2\right), \cdots \boldsymbol{\phi}_{k-1}\right), \boldsymbol{\phi}_K\right). \tag{6}$$

Writing the sequence in the reverse order gives a function that maps $\boldsymbol{X}$ to $\boldsymbol{Z}$. This formulation explains the name "normalizing flow", because the sequence of functions flows the sample distribution towards a normal distribution.

$$\boldsymbol{z} = \mathbf{f}^{-1}(\boldsymbol{x}, \boldsymbol{\phi}) = \mathbf{f}_1^{-1}\left(\mathbf{f}_2^{-1}\left(\cdots \mathbf{f}_{K-1}^{-1}\left(\mathbf{f}_K^{-1}(\boldsymbol{z}, \boldsymbol{\phi}_K), \boldsymbol{\phi}_{K-1}\right), \cdots \boldsymbol{\phi}_2\right), \boldsymbol{\phi}_1\right). \tag{7}$$

Normalizing flows compute exact likelihood of thus are capable of generating high-quality samples

## 2.4 Variational Autoencoder

A *Variational Autoencoder (VAE)* models the distribution of data $\boldsymbol{X}$ through a lower dimensional latent variable $\boldsymbol{Z}$ with a simple and known distribution. Suppose $p_{\boldsymbol{\phi}}(\boldsymbol{x}|\boldsymbol{z})$ models the conditional density function of $\boldsymbol{x}$ given $\boldsymbol{z}$. Then we have

$$p_{\boldsymbol{\phi}}(\boldsymbol{x}) = \int p_{\boldsymbol{\phi}}(\boldsymbol{x}|\boldsymbol{z})p(\boldsymbol{z})d\boldsymbol{z} \tag{8}$$

and the model learns the parameter $\boldsymbol{\phi}$ by maximizing log-likelihood of the sample $\{\boldsymbol{x}_1, \cdots, \boldsymbol{x}_n\}$

$$\sum_{i=1}^{n} \log \int p_{\boldsymbol{\phi}}(\boldsymbol{x}_i|\boldsymbol{z}_i)p(\boldsymbol{z}_i)d\boldsymbol{z}_i. \tag{9}$$

The distribution of $\boldsymbol{Z}$ is usually chosen to be a standard multivariate normal distribution.

Nevertheless, direct optimization of Equation (9) over $\boldsymbol{\phi}$ is not straightforward because it involves an integral with respect to a complicated function $p_{\boldsymbol{\phi}}(\boldsymbol{x}|\boldsymbol{z})$. VAE overcomes this challenge by approximating the reverse relationship $p_{\boldsymbol{\phi}}(\boldsymbol{z}|\boldsymbol{x})$ using a normal distribution $q_{\boldsymbol{\theta}}(\boldsymbol{z}|\boldsymbol{x})$ which is easy to manipulate. Let us denote by $\mathcal{N}_{\boldsymbol{x}}(\boldsymbol{\mu}, \boldsymbol{\Sigma})$ the density function of normal distribution with the corresponding mean and variance:

$$\mathcal{N}_{\boldsymbol{x}}(\boldsymbol{\mu}, \boldsymbol{\Sigma}) = |2\pi\boldsymbol{\Sigma}|^{-1/2}\, exp\left(-\frac{1}{2}\boldsymbol{(x-\mu)^{\top}\Sigma^{-1}(x-\mu)}\right).$$

Then,

$$q_{\boldsymbol{\theta}}(\boldsymbol{z}|\boldsymbol{x}) \approx p_{\boldsymbol{\phi}}(\boldsymbol{z}|\boldsymbol{x}), \quad q_{\boldsymbol{\theta}}(\boldsymbol{z}|\boldsymbol{x}) = \mathcal{N}_{\boldsymbol{z}}\Big(\boldsymbol{\mu}(\boldsymbol{x},\boldsymbol{\theta}), \boldsymbol{\Sigma}(\boldsymbol{x},\boldsymbol{\theta})\Big). \tag{10}$$

Lastly, equipped with the manageable function $q_{\boldsymbol{\theta}}$, we make one more approximation to handle the likelihood function by introducing *evidence lower bound (ELBO)* as follows.

$$\begin{aligned}
\log p_{\boldsymbol{\phi}}(\boldsymbol{x}) &= \log \int p_{\boldsymbol{\phi}}(\boldsymbol{x}|\boldsymbol{z})p(\boldsymbol{z})d\boldsymbol{z} \\
&= \log \int p_{\boldsymbol{\phi}}(\boldsymbol{x}|\boldsymbol{z})p(\boldsymbol{z})\frac{q_{\boldsymbol{\theta}}(\boldsymbol{z}|\boldsymbol{x})}{q_{\boldsymbol{\theta}}(\boldsymbol{z}|\boldsymbol{x})}d\boldsymbol{z} \\
&\geq \int q_{\boldsymbol{\theta}}(\boldsymbol{z}|\boldsymbol{x}) \log\left(\frac{p_{\boldsymbol{\phi}}(\boldsymbol{x}|\boldsymbol{z})p(\boldsymbol{z})}{q_{\boldsymbol{\theta}}(\boldsymbol{z}|\boldsymbol{x})}\right)d\boldsymbol{z} \\
&= \int q_{\boldsymbol{\theta}}(\boldsymbol{z}|\boldsymbol{x}) \log\left(p_{\boldsymbol{\phi}}(\boldsymbol{x}|\boldsymbol{z})\right)d\boldsymbol{z} - D_{KL}\Big(q_{\boldsymbol{\theta}}(\boldsymbol{z}|\boldsymbol{x})\big\|p(\boldsymbol{z})\Big) \\
&=: ELBO(\boldsymbol{x}; \boldsymbol{\phi}, \boldsymbol{\theta}).
\end{aligned} \tag{11}$$

Here the inequality in the third line is given by Jensen's inequality. ELBO can be better estimated than $\log p_{\boldsymbol{\phi}}(\boldsymbol{x})$ itself. First, the second Kullback-Leibler divergence term is instantly computable because both $q_{\boldsymbol{\theta}}(\boldsymbol{z}|\boldsymbol{x})$ and $p(\boldsymbol{z})$ are normal density functions and Kullback-Leibler divergence between two normal distributions has a closed form. The first integral term still involves an integral, but now we can easily sample a point $\boldsymbol{z}$ from the normal distribution $q_{\boldsymbol{\theta}}(\cdot|\boldsymbol{x}_i)$ for some randomly chosen $\boldsymbol{x}_i$ and estimate the integral by

$$\int q_{\boldsymbol{\theta}}(\boldsymbol{z}|\boldsymbol{x}) \log\left(p_{\boldsymbol{\phi}}(\boldsymbol{x}|\boldsymbol{z})\right)d\boldsymbol{z} \approx \log\left(p_{\boldsymbol{\phi}}(\boldsymbol{x}_i|\boldsymbol{z})\right). \tag{12}$$

Two parameters $\boldsymbol{\theta}$ and $\boldsymbol{\phi}$ play different roles in maximizing log-likelihood. Given a fixed value of $\boldsymbol{\phi}$, maximizing ELBO with respect to $\boldsymbol{\theta}$ tightens the lower bound. On the other hand, maximizing ELBO with respect to $\boldsymbol{\phi}$ for the given $\boldsymbol{\theta}$ maximizes the lower bound of log-likelihood. As a result, the estimated parameters are

$$\left(\widehat{\boldsymbol{\phi}}, \widehat{\boldsymbol{\theta}}\right) = \underset{\boldsymbol{\phi},\boldsymbol{\theta}}{\arg\max} \left\{ \sum_{i=1}^{n} \int q_{\boldsymbol{\theta}}(\boldsymbol{z}_i|\boldsymbol{x}_i) \log\left(p_{\boldsymbol{\phi}}(\boldsymbol{x}_i|\boldsymbol{z}_i)\right) d\boldsymbol{z}_i - D_{KL}\Big(q_{\boldsymbol{\theta}}(\boldsymbol{z}_i|\boldsymbol{x}_i) \Big\| p(\boldsymbol{z}_i)\Big) \right\} \tag{13}$$

Once we have learned a VAE, the model generates a new sample point by drawing $\boldsymbol{z}$ from $N(0, \boldsymbol{I})$ and then drawing $\boldsymbol{z}$ from $p_{\boldsymbol{\phi}}(\boldsymbol{x}|\boldsymbol{z})$.

The framework is named an autoencoder because the it has the *encoder* $q_{\boldsymbol{\theta}}(\boldsymbol{z}|\boldsymbol{x})$ and the *decoder* $p_{\boldsymbol{\phi}}(\boldsymbol{x}|\boldsymbol{z})$. The model is variational because it approximates the encoding probability with a family of simple and known distributions.

## 3 Diffusion model

The main framework of a diffusion model is to construct a parameterized Markov chain and to train the chain through variational inference to produce samples matching the data distribution after finite time. Transitions of this chain are learned to reverse the diffusion process, which is a Markov chain that gradually adds noise to the data in the opposite direction of sampling until signal is destroyed. This is illustrated in Figure 2, where the latent variables obtained by adding noise to data sample $\boldsymbol{x}$ are denoted as $\boldsymbol{z}_1, \ldots, \boldsymbol{z}_T$. The essential idea behind the diffusion model is to (i) **systematically and slowly destroy structure in a data distribution through an iterative forward diffusion process**, and then (ii) **learn a reverse diffusion process that restores structure in data** (Sohl-Dickstein et al. (2015)).

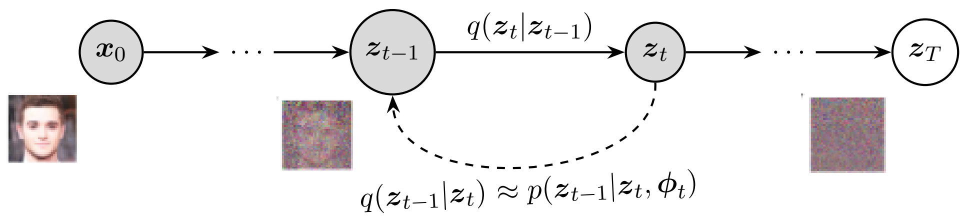
$\boldsymbol{x}_0$
$\cdots$
$\boldsymbol{z}_{t-1}$
$q(\boldsymbol{z}_t|\boldsymbol{z}_{t-1})$
$\boldsymbol{z}_t$
$\cdots$
$\boldsymbol{z}_T$
$q(\boldsymbol{z}_{t-1}|\boldsymbol{z}_t) \approx p(\boldsymbol{z}_{t-1}|\boldsymbol{z}_t, \boldsymbol{\phi}_t)$

Figure 2: The directed graphical model illustrating diffusion model.

Technically, a diffusion model works as follows. Firstly, the diffusion model sequentially adds small Gaussian noises to the sample point $\boldsymbol{x}$ to obtain a sequence of noisy data $\boldsymbol{z}_1, \cdots, \boldsymbol{z}_T$. We denote this process by $q(\boldsymbol{z}_t|\boldsymbol{z}_{t-1})$. After a large number of steps $T$, $\boldsymbol{z}_T$ becomes approximately a white noise. The goal of a diffusion model is to maximize the likelihood of the observed samples

$\boldsymbol{x}_1, \cdots, \boldsymbol{x}_I$, assuming $\boldsymbol{z}_{iT}$ come from the standard normal distribution:

$$\begin{aligned}\widehat{\boldsymbol{\phi}}_{1,\cdots,T} &:= \underset{\boldsymbol{\phi}_{1,\cdots,T}}{\arg\max} \sum_{i=1}^{I} q(\boldsymbol{x}_i) \\ &= \underset{\boldsymbol{\phi}_{1,\cdots,T}}{\arg\max} \sum_{i=1}^{I} \int q(\boldsymbol{x}_i|\boldsymbol{z}_{iT}) q(\boldsymbol{z}_{iT}) d\boldsymbol{z}_{iT}.\end{aligned} \tag{14}$$

Hypothetically, we can compute the density of the data from the reverse process $q(\boldsymbol{z}_{t-1}|\boldsymbol{z}_t)$ as follows:

$$\begin{aligned}q(\boldsymbol{x}) &= \int q(\boldsymbol{x}|\boldsymbol{z}_T) q(\boldsymbol{z}_T) d\boldsymbol{z}_T \\ &= \int q(\boldsymbol{x}|\boldsymbol{z}_1) q(\boldsymbol{z}_1|\boldsymbol{z}_2) \cdots q(\boldsymbol{z}_{T-1}|\boldsymbol{z}_T) q(\boldsymbol{z}_T) d\boldsymbol{z}_{1,\cdots,T}.\end{aligned} \tag{15}$$

However, in practice, the exact form of $q(\boldsymbol{z}_{t-1}|\boldsymbol{z}_t)$ is neither known to us nor tractable. In addition, the high-dimensional integral is not straightforward to evaluate. A diffusion model overcomes these difficulties by first approximating each $q(\boldsymbol{z}_{t-1}|\boldsymbol{z}_t)$ by a normal density function

$$\begin{aligned}q(\boldsymbol{z}_{t-1}|\boldsymbol{z}_t) &\approx p(\boldsymbol{z}_{t-1}|\boldsymbol{z}_t, \boldsymbol{\phi}_t) \\ &= \mathcal{N}_{\boldsymbol{z}_{t-1}}\left[\mathbf{f}_t(\boldsymbol{z}_t, \boldsymbol{\phi}), \sigma_t^2 \boldsymbol{I}\right]\end{aligned} \tag{16}$$

where the mean $\mathbf{f}_t(\boldsymbol{z}_t, \boldsymbol{\phi})$ is trained as a neural network from the data and the variance $\sigma_t^2$ is prespecified. The second challenge of handling the high-dimensional integral is resolved by substituting the integral by the Evidence Lower Bound (ELBO).

## 3.1 Encoder (Forward Pass, Model Setting)

A diffusion model is composed of an encoder and a decoder. Unlike the Variational Autoencoder (VAE) model, the encoder in a diffusion model is prespecified, representing the process of gradually adding noise to the data $\boldsymbol{x}$ and simulating its diffusion. To be specific, one can formulate the forward process as follows:

$$\begin{aligned}\boldsymbol{z}_1 &= \sqrt{1-\beta_1}\boldsymbol{x} + \sqrt{\beta_1}\boldsymbol{\epsilon}_1 \\ \boldsymbol{z}_t &= \sqrt{1-\beta_t}\boldsymbol{z}_{t-1} + \sqrt{\beta_t}\boldsymbol{\epsilon}_t, \quad \forall t \in 2, \ldots, T,\end{aligned} \tag{17}$$

where $\boldsymbol{\epsilon}_t$ follows a standard normal distribution. The hyperparameters $\beta_t \in [0,1]$ determine the magnitude of the added noise. Here we note the following properties.

- We can derive the following distributions:

$$\begin{aligned}q(\boldsymbol{z}_1|\boldsymbol{x}) &= \mathcal{N}(\sqrt{1-\beta_1}\boldsymbol{x}, \beta_1 \boldsymbol{I}), \\ q(\boldsymbol{z}_t|\boldsymbol{z}_{t-1}) &= \mathcal{N}(\sqrt{1-\beta_t}\boldsymbol{z}_{t-1}, \beta_t \boldsymbol{I}), \quad \forall t \in 2, \ldots, T.\end{aligned} \tag{18}$$

This is a Markov chain.

- With enough steps, i.e. large $T$, all traces of the original data are removed, and the conditional distribution $q(\boldsymbol{z}_T|\boldsymbol{x})$ and marginal distribution $q(\boldsymbol{z}_T)$ both become the standard normal distribution. Indeed, the ultimate goal of this forward pass is to gradually convert the data distribution into a well behaved distribution by repeated application of Markov diffusion kernel.

- One can directly sample $\boldsymbol{z}_t$ given $\boldsymbol{x}$ without computing intermediate variables $\boldsymbol{z}_1, \ldots, \boldsymbol{z}_{t-1}$. That is, by iteratively substituting variables, one can obtain

$$\begin{aligned} \boldsymbol{z}_t &= \sqrt{\alpha_t}\boldsymbol{x} + \sqrt{1-\alpha_t}\boldsymbol{\epsilon}, \\ q(\boldsymbol{z}_t|\boldsymbol{x}) &= \mathcal{N}(\sqrt{\alpha_t}\boldsymbol{x}, (1-\alpha_t)\boldsymbol{I}), \end{aligned} \tag{19}$$

  where $\alpha_t = \Pi_{s=1}^t(1-\beta_s)$ and $\boldsymbol{\epsilon}$ is a sample from standard normal distribution. This $q(\boldsymbol{z}_t|\boldsymbol{x})$ is known as the diffusion kernel.

In summary, for any starting point $\boldsymbol{x}$, variable $\boldsymbol{z}_t$ is normally distributed with a known mean and variance.

## 3.2 Conditional Distributions

Before we proceed with learning the reverse process of the diffusion model, we examine here the true reverse probabilities that are of our interest. We first note that the marginal distribution of $\boldsymbol{z}_t$ can be computed using the diffusion kernel,

$$q(\boldsymbol{z}_t) = \int q(\boldsymbol{z}_t|\boldsymbol{x})p(\boldsymbol{x})d\boldsymbol{x}. \tag{20}$$

However, it is difficult to know $p(\boldsymbol{x})$ in practice, and consequently $q(\boldsymbol{z}_t)$ can not be obtained in a closed form. The conditional distribution of the reverse process can then be written as

$$q(\boldsymbol{z}_{t-1}|\boldsymbol{z}_t) = \frac{q(\boldsymbol{z}_t|\boldsymbol{z}_{t-1})q(\boldsymbol{z}_{t-1})}{q(\boldsymbol{z}_t)}, \tag{21}$$

as a result of the Bayes' rule. Again, this is intractable since we cannot compute the marginal distributions in (20).

We then focus on the conditional diffusion distribution, defined as $q(\boldsymbol{z}_{t-1}|\boldsymbol{z}_t, \boldsymbol{x})$. As we do know $q(\boldsymbol{z}_{t-1}|\boldsymbol{x})$, which is just an diffusion kernel, we can utilize this to compute the conditional diffusion distribution in a closed form.

**Lemma 1.**

$$\begin{aligned} &(a) \quad \mathcal{N}(\boldsymbol{A}\boldsymbol{w}, \boldsymbol{B}) \propto \mathcal{N}\left((\boldsymbol{A}^\top\boldsymbol{B}^{-1}\boldsymbol{A})^{-1}\boldsymbol{A}^\top\boldsymbol{B}^{-1}\boldsymbol{v},\ (\boldsymbol{A}^\top\boldsymbol{B}^{-1}\boldsymbol{A})^{-1}\right), \\ &(b) \quad \mathcal{N}(\boldsymbol{a}, \boldsymbol{A}) \cdot \mathcal{N}(\boldsymbol{b}, \boldsymbol{B}) \propto \mathcal{N}\left((\boldsymbol{A}^{-1}+\boldsymbol{B}^{-1})^{-1}(\boldsymbol{A}^{-1}\boldsymbol{a}+\boldsymbol{B}^{-1}\boldsymbol{b}),\ (\boldsymbol{A}^{-1}+\boldsymbol{B}^{-1})^{-1}\right) \end{aligned} \tag{22}$$

**Theorem 2.**

$$q(\boldsymbol{z}_{t-1}|\boldsymbol{z}_t,\boldsymbol{x}) = \mathcal{N}(\frac{1-\alpha_{t-1}}{1-\alpha_t}\sqrt{1-\beta_t}\boldsymbol{z}_t + \frac{\sqrt{\alpha_{t-1}}\beta_t}{1-\alpha_t}\boldsymbol{x},\ \frac{\beta_t(1-\alpha_{t-1})}{1-\alpha_t}\boldsymbol{I}).$$

*Proof.*

$$\begin{aligned} q(\boldsymbol{z}_{t-1}|\boldsymbol{z}_t,\boldsymbol{x}) &= \frac{q(\boldsymbol{z}_t|\boldsymbol{z}_{t-1},\boldsymbol{x})q(\boldsymbol{z}_{t-1}|\boldsymbol{x})}{q(\boldsymbol{z}_t|\boldsymbol{x})} \\ &\propto q(\boldsymbol{z}_t|\boldsymbol{z}_{t-1})q(\boldsymbol{z}_{t-1}|\boldsymbol{x}) \\ &= \mathcal{N}\left(\sqrt{1-\beta_t}\boldsymbol{z}_{t-1}, \beta_t\boldsymbol{I}\right)\mathcal{N}\left(\sqrt{\alpha_{t-1}}\boldsymbol{x}, (1-\alpha_{t-1})\boldsymbol{I}\right) \\ &\propto \mathcal{N}\left(\frac{1}{\sqrt{1-\beta_t}}\boldsymbol{z}_t, \frac{\beta_t}{1-\beta_t}\boldsymbol{I}\right)\mathcal{N}\left(\sqrt{\alpha_{t-1}}\boldsymbol{x}, (1-\alpha_{t-1})\boldsymbol{I}\right) \end{aligned} \tag{23}$$

Applying Lemma 1, we obtain the result. □

We highlight here that this fact in Theorem 2 is then used to train the decoder.

## 3.3 Decoder (Backward pass, Model Learning)

When we train a diffusion model, it learns the reverse process, i.e. the backward map between $\boldsymbol{z}_t$ and $\boldsymbol{z}_{t-1}$. This network is trained to gradually remove noise, originating the term 'denoising diffusion probabilistic model'. To generate a new data example $\boldsymbol{x}$, we draw a sample from $q(\boldsymbol{z}_T)$, which is simply a standard normal distribution, and then process it through the decoder models.

As seen in Section 3.2, it is difficult to know the true reverse distribution $q(\boldsymbol{z}_{t-1}|\boldsymbol{z}_t)$ of the diffusion process. We approximate these as normal distributions and set the following model:

$$\begin{aligned} p(\boldsymbol{z}_T) &= \mathcal{N}(\boldsymbol{0},\boldsymbol{I}), \\ p(\boldsymbol{z}_{t-1}|\boldsymbol{z}_t,\boldsymbol{\phi}_t) &= \mathcal{N}(\mathbf{f}_t(\boldsymbol{z}_t,\boldsymbol{\phi}_t),\sigma_t^2\boldsymbol{I}) \\ p(\boldsymbol{x}|\boldsymbol{z}_1,\boldsymbol{\phi}_1) &= \mathcal{N}(\mathbf{f}_1(\boldsymbol{z}_1,\boldsymbol{\phi}_1),\sigma_1^2\boldsymbol{I}), \end{aligned} \tag{24}$$

where $\boldsymbol{\phi}_t, t=1,\ldots,T$ are parameters to learn and $\mathbf{f}_t(\boldsymbol{z}_t,\boldsymbol{\phi}_t)$ is a neural network model that predicts the mean of the normal distribution for the preceding latent variable $\boldsymbol{z}_{t-1}$ given $\boldsymbol{z}_t$. Here, $\{\sigma_t^2\}$ are predetermined parameters.

To train the model, we generate new examples from $p(\boldsymbol{x})$ as follows. First, sample $\boldsymbol{z}_T$ from $p(\boldsymbol{z}_T)$. Then, sample $\boldsymbol{z}_{T-1}$ from $p(\boldsymbol{z}_{T-1}|\boldsymbol{z}_T,\boldsymbol{\phi}_T)$, and repeat this until we finally generate $\boldsymbol{x}$ from $p(\boldsymbol{x}|\boldsymbol{z}_1,\boldsymbol{\phi}_1)$. Now we can approximate the probability of training dataset $\{\boldsymbol{x}_i\}$ as follows. First, the entire joint probability of $\boldsymbol{x}$ and latent variables $\boldsymbol{z}_{1,\cdots,T}$ is

$$p(\boldsymbol{x},\boldsymbol{z}_{1,\cdots,T}|\boldsymbol{\phi}_{1,\cdots,T}) = p(\boldsymbol{x}|\boldsymbol{z}_1,\boldsymbol{\phi}_1)\prod_{t=2}^{T}p(\boldsymbol{z}_{t-1}|\boldsymbol{z}_t,\boldsymbol{\phi}_t)\cdot p(\boldsymbol{z}_T). \tag{25}$$

The marginal probability of $\boldsymbol{x}$ is obtained by integrating Equation (25) over all the latent variables:

$$p(\boldsymbol{x}|\boldsymbol{\phi}_{1,\cdots,T}) = \int p(\boldsymbol{x}, \boldsymbol{z}_{1,\cdots,T}|\boldsymbol{\phi}_{1,\cdots,T}) d\boldsymbol{z}_{1,\cdots,T}. \tag{26}$$

To train the model, we maximize the log-likelihood of the training data $\{\boldsymbol{x}_i\}$ with respect to the parameters $\boldsymbol{\phi}$:

$$\widehat{\boldsymbol{\phi}}_{1,\cdots,T} = \underset{\boldsymbol{\phi}_{1,\cdots,T}}{\arg\max} \left[ \sum_{i=1}^{I} \log \left[ p(\boldsymbol{x}_i|\boldsymbol{\phi}_{1,\cdots,T}) \right] \right]. \tag{27}$$

## 4 Evidence Lower Bound (ELBO)

The exact optimization problem in (27) is intractable because of the integral involved in Equation (26). We make a detour around this difficulty using *evidence lower bound (ELBO).*

$$\begin{aligned}
\log \left[ p(\boldsymbol{x}|\boldsymbol{\phi}_{1,\cdots,T}) \right] &= \log \left[ \int p(\boldsymbol{x}, \boldsymbol{z}_1, \cdots, \boldsymbol{z}_T|\boldsymbol{\phi}_{1,\cdots,T}) d\boldsymbol{z}_{1,\cdots,T} \right] \\
&= \log \left[ \int q(\boldsymbol{z}_{1,\cdots,T}|\boldsymbol{x}) \frac{p(\boldsymbol{x}, \boldsymbol{z}_1, \cdots, \boldsymbol{z}_T|\boldsymbol{\phi}_{1,\cdots,T})}{q(\boldsymbol{z}_{1,\cdots,T}|\boldsymbol{x})} d\boldsymbol{z}_{1,\cdots,T} \right] \\
&\geq \int q(\boldsymbol{z}_{1,\cdots,T}|\boldsymbol{x}) \log \left[ \frac{p(\boldsymbol{x}, \boldsymbol{z}_1, \cdots, \boldsymbol{z}_T|\boldsymbol{\phi}_{1,\cdots,T})}{q(\boldsymbol{z}_{1,\cdots,T}|\boldsymbol{x})} \right] d\boldsymbol{z}_{1,\cdots,T} \\
&=: \mathrm{ELBO}[\boldsymbol{\phi}_{1,\cdots,T}]
\end{aligned} \tag{28}$$

Next, we simplify the whole joint probabilities of $\boldsymbol{x}$ and $\boldsymbol{z}_{1,\cdots,T}$ in Equation (28) in terms of the conditional probabilities between $\boldsymbol{x}, \boldsymbol{\phi}$, and consecutive latent variables $\boldsymbol{z}_t$ and $\boldsymbol{z}_{t-1}$.

$$\begin{aligned}
&\log \left[ \frac{p(\boldsymbol{x}, \boldsymbol{z}_{1,\cdots,T}|\boldsymbol{\phi}_{1,\cdots,T})}{q(\boldsymbol{z}_{1,\cdots,T}|\boldsymbol{x})} \right] \\
&\quad = \log \left[ \frac{p(\boldsymbol{x}|\boldsymbol{z}_1, \boldsymbol{\phi}_1) \prod_{t=2}^{T} p(\boldsymbol{z}_{t-1}|\boldsymbol{z}_t, \boldsymbol{\phi}_t) \cdot p(\boldsymbol{z}_T)}{q(\boldsymbol{z}_1|\boldsymbol{x}) \prod_{t=2}^{T} q(\boldsymbol{z}_t|\boldsymbol{z}_{t-1})} \right] \\
&\quad = \log \left[ \frac{p(\boldsymbol{x}|\boldsymbol{z}_1, \boldsymbol{\phi}_1)}{q(\boldsymbol{z}_1|\boldsymbol{x})} \right] + \log \left[ \frac{\prod_{t=2}^{T} p(\boldsymbol{z}_{t-1}|\boldsymbol{z}_t, \boldsymbol{\phi}_t)}{\prod_{t=2}^{T} q(\boldsymbol{z}_t|\boldsymbol{z}_{t-1})} \right] + \log \left[ p(\boldsymbol{z}_T) \right].
\end{aligned} \tag{29}$$

In the middle term, the numerator $p(\boldsymbol{z}_{t-1}|\boldsymbol{z}_t, \boldsymbol{\phi}_t)$ is given as the normal density with mean $\mathbf{f}[\boldsymbol{z}_t, \boldsymbol{\phi}_t]$. To make the denominator $q(\boldsymbol{z}_t|\boldsymbol{z}_{t-1})$ more accessible, we rewrite the term as

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

As a result,

$$\begin{aligned}
&\log\left[\frac{p(\boldsymbol{x},\boldsymbol{z}_{1,\cdots,T}|\boldsymbol{\phi}_{1,\cdots,T})}{q(\boldsymbol{z}_{1,\cdots,T}|\boldsymbol{x})}\right]\\
&\quad=\log\left[\frac{p(\boldsymbol{x}|\boldsymbol{z}_1,\boldsymbol{\phi}_1)}{q(\boldsymbol{z}_1|\boldsymbol{x})}\right]+\log\left[\prod_{t=2}^{T}\frac{p(\boldsymbol{z}_{t-1}|\boldsymbol{z}_t,\boldsymbol{\phi}_t)}{q(\boldsymbol{z}_{t-1}|\boldsymbol{z}_t,\boldsymbol{x})}\cdot\frac{q(\boldsymbol{z}_{t-1}|\boldsymbol{x})}{q(\boldsymbol{z}_t|\boldsymbol{x})}\right]+\log\left[p(\boldsymbol{z}_T)\right]\\
&\quad=\log\left[p(\boldsymbol{x}|\boldsymbol{z}_1,\boldsymbol{\phi}_1)\right]+\log\left[\prod_{t=2}^{T}\frac{p(\boldsymbol{z}_{t-1}|\boldsymbol{z}_t,\boldsymbol{\phi}_t)}{q(\boldsymbol{z}_{t-1}|\boldsymbol{z}_t,\boldsymbol{x})}\right]+\log\left[\frac{p(\boldsymbol{z}_T)}{q(\boldsymbol{z}_T|\boldsymbol{x})}\right]\\
&\quad=\log\left[p(\boldsymbol{x}|\boldsymbol{z}_1,\boldsymbol{\phi}_1)\right]+\sum_{t=2}^{T}\log\left[\frac{p(\boldsymbol{z}_{t-1}|\boldsymbol{z}_t,\boldsymbol{\phi}_t)}{q(\boldsymbol{z}_{t-1}|\boldsymbol{z}_t,\boldsymbol{x})}\right]+\log\left[\frac{p(\boldsymbol{z}_T)}{q(\boldsymbol{z}_T|\boldsymbol{x})}\right]\\
&\quad\approx\log\left[p(\boldsymbol{x}|\boldsymbol{z}_1,\boldsymbol{\phi}_1)\right]+\sum_{t=2}^{T}\log\left[\frac{p(\boldsymbol{z}_{t-1}|\boldsymbol{z}_t,\boldsymbol{\phi}_t)}{q(\boldsymbol{z}_{t-1}|\boldsymbol{z}_t,\boldsymbol{x})}\right]
\end{aligned}\tag{30}$$

Here the last term $\log\left[\frac{p(\boldsymbol{z}_T)}{q(\boldsymbol{z}_T|\boldsymbol{x})}\right]$ is approximated to zero, because for sufficiently large $T$, both $p(\boldsymbol{z}_T)$ and $q(\boldsymbol{z}_T|\boldsymbol{x})$ are approximately standard normal densities. We obtain the following simplified version of ELBO by substituting (30) into the definition of ELBO:

$$\begin{aligned}
&\mathrm{ELBO}[\boldsymbol{\phi}_{1,\cdots,T}]\\
&\approx\int q(\boldsymbol{z}_{1,\cdots,T}|\boldsymbol{x})\left(\log\left[p(\boldsymbol{x}|\boldsymbol{z}_1,\boldsymbol{\phi}_1)\right]+\sum_{t=2}^{T}\log\left[\frac{p(\boldsymbol{z}_{t-1}|\boldsymbol{z}_t,\boldsymbol{\phi}_t)}{q(\boldsymbol{z}_{t-1}|\boldsymbol{z}_t,\boldsymbol{x})}\right]\right)d\boldsymbol{z}_{1,\cdots,T}\\
&=\int q(\boldsymbol{z}_1|\boldsymbol{x})\log\left[p(\boldsymbol{x}|\boldsymbol{z}_1,\boldsymbol{\phi}_1)\right]d\boldsymbol{z}_1\\
&\qquad\qquad-\sum_{t=2}^{T}\int q(\boldsymbol{z}_t|\boldsymbol{x})q(\boldsymbol{z}_{t-1}|\boldsymbol{z}_t,\boldsymbol{x})\log\left[\frac{q(\boldsymbol{z}_{t-1}|\boldsymbol{z}_t,\boldsymbol{x})}{p(\boldsymbol{z}_{t-1}|\boldsymbol{z}_t,\boldsymbol{\phi}_t)}\right]d\boldsymbol{z}_{t-1}d\boldsymbol{z}_t
\end{aligned}\tag{31}$$

The first integral term will be replaced by the empirical expectation of $\log\left[p(\boldsymbol{x}|\boldsymbol{z}_1,\boldsymbol{\phi}_1)\right]$ with a single sample from the distribution $q(\boldsymbol{z}_1|\boldsymbol{x})$, that is,

$$\int q(\boldsymbol{z}_1|\boldsymbol{x})\log\left[p(\boldsymbol{x}|\boldsymbol{z}_1,\boldsymbol{\phi}_1)\right]d\boldsymbol{z}_1\approx\log\left[p(\boldsymbol{x}|\boldsymbol{z}_1^*,\boldsymbol{\phi}_1)\right]\tag{32}$$

where $\boldsymbol{z}_1^*\sim q(\boldsymbol{z}_1|\boldsymbol{x})$. The second integral term can be further simplified as follows.

$$\begin{aligned}
&\int q(\boldsymbol{z}_t|\boldsymbol{x})q(\boldsymbol{z}_{t-1}|\boldsymbol{z}_t,\boldsymbol{x})\log\left[\frac{q(\boldsymbol{z}_{t-1}|\boldsymbol{z}_t,\boldsymbol{x})}{p(\boldsymbol{z}_{t-1}|\boldsymbol{z}_t,\boldsymbol{\phi}_t)}\right]d\boldsymbol{z}_{t-1}d\boldsymbol{z}_t\\
&\quad=\int q(\boldsymbol{z}_t|\boldsymbol{x})D_{KL}\left[q(\boldsymbol{z}_{t-1}|\boldsymbol{z}_t,\boldsymbol{x})\|p(\boldsymbol{z}_{t-1}|\boldsymbol{z}_t,\boldsymbol{\phi}_t)\right]d\boldsymbol{z}_t\\
&\quad=\mathbb{E}_{q(\boldsymbol{z}_t|\boldsymbol{x})}\left[D_{KL}\left[q(\boldsymbol{z}_{t-1}|\boldsymbol{z}_t,\boldsymbol{x})\|p(\boldsymbol{z}_{t-1}|\boldsymbol{z}_t,\boldsymbol{\phi}_t)\right]\right]
\end{aligned}\tag{33}$$

Recall that $q(\boldsymbol{z}_{t-1}|\boldsymbol{z}_t,\boldsymbol{x})$ and $p(\boldsymbol{z}_{t-1}|\boldsymbol{z}_t,\boldsymbol{\phi}_t)$ are normal probability densities. Kullback-Leibler divergence between two normal distributions can be calculated in a Rao-Blackwellized fashion with closed form expressions instead of high variance Monte Carlo estimates (Ho et al. (2020)).

**Lemma 3.** *Suppose $\boldsymbol{p}(\boldsymbol{x}) = \mathcal{N}_{\boldsymbol{x}}(\boldsymbol{a}, \boldsymbol{A})$ and $\boldsymbol{q}(\boldsymbol{x}) = \mathcal{N}_{\boldsymbol{x}}(\boldsymbol{b}, \boldsymbol{B})$ are d-dimensional normal distributions. Then the KL-divergence between $\boldsymbol{p}$ and $\boldsymbol{q}$ is given as follows.*

$$D_{KL}\left[\boldsymbol{p}\|\boldsymbol{q}\right] = \frac{1}{2}\left(tr\left[\boldsymbol{B}^{-1}\boldsymbol{A}\right] - d + (\boldsymbol{a}-\boldsymbol{b})^{\top}\boldsymbol{B}^{-1}(\boldsymbol{a}-\boldsymbol{b}) + \log\left[\frac{|\boldsymbol{B}|}{|\boldsymbol{A}|}\right]\right). \tag{34}$$

*If $\boldsymbol{A} = \sigma_a \boldsymbol{I}_d$ and $\boldsymbol{B} = \sigma_b \boldsymbol{I}_d$, KL-divergence is further simplified as*

$$D_{KL}\left[\boldsymbol{p}\|\boldsymbol{q}\right] = \frac{1}{2}\left(\frac{1}{\sigma_b}\|\boldsymbol{a}-\boldsymbol{b}\|^2 + d\left(\frac{\sigma_a}{\sigma_b} - 1 + \log\left(\frac{\sigma_a}{\sigma_b}\right)\right)\right).$$

Applying Lemma 3 to our problem yields:

$$\begin{aligned} L_{t-1} &:= D_{KL}\left[q(\boldsymbol{z}_{t-1}|\boldsymbol{z}_t, \boldsymbol{x})\|p(\boldsymbol{z}_{t-1}|\boldsymbol{z}_t, \boldsymbol{\phi}_t)\right] \\ &= \frac{1}{2\sigma_t^2}\left\|\frac{1-\alpha_{t-1}}{1-\alpha_t}\sqrt{1-\beta_t}\boldsymbol{z}_t + \frac{\sqrt{\alpha_{t-1}}\beta_t}{1-\alpha_t}\boldsymbol{x} - \mathbf{f}_t[\boldsymbol{z}_t, \boldsymbol{\phi}_t]\right\|^2 + C \end{aligned} \tag{35}$$

for some constant $C$ that does not depend on $\boldsymbol{\phi}_{1,\cdots,T}$. Finally, we again replace the expectation with respect to $\boldsymbol{z}_{1,\cdots,T}$ in Equation (31) by the empirical expectation, by drawing $\boldsymbol{z}_{it}^*$ from $q(\boldsymbol{z}_{it}^*|\boldsymbol{x}_i), i = 1,\cdots,I, t = 1,\cdots,T$.

$$\begin{aligned} L[\boldsymbol{\phi}_{1,\cdots,T}] \approx \sum_{i=1}^{I}\Bigg( &-\log\left[\mathcal{N}_{\boldsymbol{x}_i}\left[\mathbf{f}_1\left[\boldsymbol{z}_{i1}^*, \boldsymbol{\phi}_1\right], \sigma_1^2\boldsymbol{I}\right]\right] \\ &+ \sum_{t=2}^{T}\frac{1}{2\sigma_t^2}\left\|\frac{1-\alpha_{t-1}}{1-\alpha_t}\sqrt{1-\beta_t}\boldsymbol{z}_{it}^* + \frac{\sqrt{\alpha_{t-1}}\beta_t}{1-\alpha_t}\boldsymbol{x}_i - \mathbf{f}_t[\boldsymbol{z}_{it}^*, \boldsymbol{\phi}_t]\right\|^2\Bigg). \end{aligned} \tag{36}$$

## 5 Reparameterization and the resulting algorithms

The loss function in (36) can be decomposed in to the following parts:

$$\begin{aligned} &\text{Reconstruction term} : \log\left[\mathcal{N}_{\boldsymbol{x}_i}\left[\mathbf{f}_1\left[\boldsymbol{z}_{i1}, \boldsymbol{\phi}_1\right], \sigma_1^2\boldsymbol{I}\right]\right] \\ &\text{(Target) mean of } q(\boldsymbol{z}_{t-1}|\boldsymbol{z}_t, \boldsymbol{x}) : \frac{1-\alpha_{t-1}}{1-\alpha_t}\sqrt{1-\beta_t}\boldsymbol{z}_{it} + \frac{\sqrt{\alpha_{t-1}}\beta_t}{1-\alpha_t}\boldsymbol{x}_i \\ &\text{Predicted } \boldsymbol{z}_{t-1} : \mathbf{f}_t[\boldsymbol{z}_{it}, \boldsymbol{\phi}_t]. \end{aligned} \tag{37}$$

Here, each $\boldsymbol{x}_i$ is the $i$th data point and $\boldsymbol{z}_{it}$ is the associated latent variable at diffusion step $t$. This loss function can be used to train a network **for each diffusion time step**. It minimizes the difference between the estimated $\mathbf{f}_t[\boldsymbol{z}_{it}, \boldsymbol{\phi}_t]$ of the hidden variable at the previous time step and the most likely value that it took given the ground truth de-noised data $\boldsymbol{x}$. However, the optimization can be further simplified by reparameterizing the target and the network so that the model aims to predict the noise, instead of the hidden variable.

## 5.1 Reparameterization

The original diffusion update was given by (19). One can rewrite it in terms of the data $\boldsymbol{x}$ such that

$$\boldsymbol{x} = \frac{1}{\sqrt{\alpha_t}}\boldsymbol{z}_t + \frac{\sqrt{1-\alpha_t}}{\sqrt{\alpha_t}}\boldsymbol{\epsilon}. \tag{38}$$

Substituting this into the target terms in (37) yields

$$\begin{aligned}
&\frac{1-\alpha_{t-1}}{1-\alpha_t}\sqrt{1-\beta_t}\boldsymbol{z}_t + \frac{\sqrt{\alpha_{t-1}}\beta_t}{1-\alpha_t}\boldsymbol{x} \\
&\quad= \frac{1-\alpha_{t-1}}{1-\alpha_t}\sqrt{1-\beta_t}\boldsymbol{z}_t + \frac{\sqrt{\alpha_{t-1}}\beta_t}{1-\alpha_t}\left(\frac{1}{\sqrt{\alpha_t}}\boldsymbol{z}_t + \frac{\sqrt{1-\alpha_t}}{\sqrt{\alpha_t}}\boldsymbol{\epsilon}\right) \\
&\quad= \left(\frac{(1-\alpha_{t-1})\sqrt{1-\beta_t}}{1-\alpha_t} + \frac{\beta_t}{(1-\alpha_t)\sqrt{1-\beta_t}}\right)\boldsymbol{z}_t - \frac{\beta_t}{\sqrt{1-\alpha_t}\sqrt{1-\beta_t}}\boldsymbol{\epsilon} \\
&\quad= \frac{1}{\sqrt{1-\beta_t}}\boldsymbol{z}_t - \frac{\beta_t}{\sqrt{1-\alpha_t}\sqrt{1-\beta_t}}\boldsymbol{\epsilon},
\end{aligned} \tag{39}$$

where we have used the fact that $\frac{\sqrt{\alpha_t}}{\sqrt{\alpha_{t-1}}} = \sqrt{1-\beta_t}$.

Then, the network should also be reparameterized. We replace the model $\widehat{\boldsymbol{z}}_{t-1} = \mathbf{f}[\boldsymbol{z}_t, \boldsymbol{\phi}_t]$ with a new model $\boldsymbol{\epsilon} = \mathbf{g}_t[\boldsymbol{z}_t, \boldsymbol{\phi}_t]$, which predicts the noise $\boldsymbol{\epsilon}$ that was mixed with $\boldsymbol{x}$ to create $\boldsymbol{z}_t$:

$$\mathbf{f}[\boldsymbol{z}_t, \boldsymbol{\phi}_t] = \frac{1}{\sqrt{1-\beta_t}}\boldsymbol{z}_t - \frac{\beta_t}{\sqrt{1-\alpha_t}\sqrt{1-\beta_t}}\mathbf{g}_t[\boldsymbol{z}_t, \boldsymbol{\phi}_t]. \tag{40}$$

Combining the reparameterization of the target and the network, the loss function given in (36) can be rewritten as

$$\begin{aligned}
&L[\boldsymbol{\phi}_{1,\cdots,T}] \\
&\quad= \sum_{i=1}^{I}\left(-\log\left[\mathcal{N}_{\boldsymbol{x}_i}\left[\mathbf{f}_1\left[\boldsymbol{z}_{i1}, \boldsymbol{\phi}_1\right], \sigma_1^2\boldsymbol{I}\right]\right] + \sum_{t=2}^{T}\frac{\beta_t^2}{(1-\alpha_t)(1-\beta_t)2\sigma_t^2}\left\|\mathbf{g}_t[\boldsymbol{z}_{it}, \boldsymbol{\phi}_t] - \boldsymbol{\epsilon}_{it}\right\|^2\right) \\
&\quad= \sum_{i=1}^{I}\left(\frac{1}{2\sigma_1^2}\left\|\boldsymbol{x}_i - \mathbf{f}_t[\boldsymbol{z}_{i1}, \boldsymbol{\phi}_1]\right\|^2 + C_i + \sum_{t=2}^{T}\frac{\beta_t^2}{(1-\alpha_t)(1-\beta_t)2\sigma_t^2}\left\|\mathbf{g}_t[\boldsymbol{z}_{it}, \boldsymbol{\phi}_t] - \boldsymbol{\epsilon}_{it}\right\|^2\right) \\
&\quad= \sum_{i=1}^{I}\sum_{t=1}^{T}\frac{\beta_t^2}{(1-\alpha_t)(1-\beta_t)2\sigma_t^2}\left\|\mathbf{g}_t[\boldsymbol{z}_{it}, \boldsymbol{\phi}_t] - \boldsymbol{\epsilon}_{it}\right\|^2 + C_i,
\end{aligned} \tag{41}$$

where we substituted in (38) and (40) for $t = 1$.

## 5.2 Algorithms

In practice, to solve the optimization problem, we can ignore the scaling factor and the constants. This leads to straightforward algorithms for both training the model and sampling as in Algorithms 1 and 2. These algorithms are simple to implement and naturally augments the dataset, as we can reuse every original data point $\boldsymbol{x}_i$ as many times as we want at each time step with different noise $\boldsymbol{\epsilon}$ (Prince (2023)).

**Algorithm 1** Training

```
1: repeat
2:     $\boldsymbol{x} \sim q(\boldsymbol{x})$
3:     $t \sim \text{Uniform}(\{1, \ldots, T\})$                ▷ sample random timestep
4:     $\boldsymbol{\epsilon} \sim \mathcal{N}(\mathbf{0}, \boldsymbol{I})$
5:     Take gradient descent step on
       $\nabla_\theta \left\| \boldsymbol{\epsilon} - \mathbf{g}_t\left(\sqrt{\alpha_t}\boldsymbol{x} + \sqrt{1-\alpha_t}\boldsymbol{\epsilon}, \boldsymbol{\phi}_t\right) \right\|^2$
6: until converged
```

**Algorithm 2** Sampling

```
1: $\boldsymbol{z}_T \sim \mathcal{N}(\mathbf{0}, \boldsymbol{I})$
2: for $t = T, \ldots, 2$ do
3:     $\boldsymbol{\epsilon} \sim \mathcal{N}(0, \boldsymbol{I})$ if $t > 1$, else $\boldsymbol{\epsilon} = 0$
4:     $\widehat{\boldsymbol{z}}_{t-1} = \frac{1}{\sqrt{1-\beta_t}}\left(\boldsymbol{z}_t - \frac{\beta_t}{\sqrt{1-\alpha_t}}\mathbf{g}_t[\boldsymbol{z}_t, \boldsymbol{\phi}_t]\right)$
5:     $\boldsymbol{z}_{t-1} = \widehat{\boldsymbol{z}}_{t-1} + \sigma_t\boldsymbol{\epsilon}$                ▷ add noise to the next input
6: end for
7: return $\boldsymbol{x} = \frac{1}{\sqrt{1-\beta_1}}\left(\boldsymbol{z}_1 - \frac{\beta_1}{\sqrt{1-\alpha_1}}\mathbf{g}_1[\boldsymbol{z}_1, \boldsymbol{\phi}_1]\right)$
```

## 5.3 Special note on the connection to the Langevin dynamics

What information about the data distribution does the model learn through the denoising process? An interesting note can be made to say that "Diffusion models approximate the solution to stochastic differential equations." To understand it, we note the following concepts.

- The score function is defined as the gradient vector of the log of the probability density function,

  $$\nabla_{\boldsymbol{x}} \log p(\boldsymbol{x}). \tag{42}$$

  The score matching (SM) technique aims to train the model to learn the Score function.

- Denoising autoencoder model (DAE) aims to learn the model $\widehat{X} = f_\theta(Z)$, where $Z$ is a noise random variable. This represents a one-step denoising process. In contrast, diffusion models employ a more gradual approach, iteratively adding noise and learning the reverse procedure.

- Interestingly, Vincent (2011) has shown that the denoising autoencoder model (DAE) shares the same objective function with the denoising score matching (DSM) technique, as well as that of the SM technique.

  - DSM aims to learn the Score function of the conditional probability distribution $q(\boldsymbol{z}|\boldsymbol{x})$. If the noise follows $\mathcal{N}(0, \sigma^2)$, then the score function of the conditional probability distribution is
$$s(\boldsymbol{z}|\boldsymbol{x}) = \frac{\boldsymbol{x} - \boldsymbol{z}}{\sigma^2}. \tag{43}$$
  - When a machine learning model is optimized to learn the score function of the above conditional probability distribution, it becomes capable of estimating the relative magnitude of injected noise in comparison to the variance when it receives noise-added data as input. Consequently, the model learns the trajectory from the corrupted information $Z$ towards the original data $X$, effectively learning the path of data restoration.
  - DAE sharing the same objective function means that through the denoising process, the DAE model ultimately learns the score function of the input data distribution.

- Langevin dynamics can produce samples from a probability density $p(\boldsymbol{x})$ using the score function $\nabla_{\boldsymbol{x}} \log p(\boldsymbol{x})$. That is, given a fixed step size $\eta$, and an initial value $\boldsymbol{x} \sim \pi(\boldsymbol{x})$ with $\pi$ being a prior distribution, the Langevin method recursively computes the following,
$$\boldsymbol{x}_t \leftarrow \boldsymbol{x}_{t-1} + \frac{\eta}{2} \nabla_{\boldsymbol{x}} \log p(\boldsymbol{x}_t) + \sqrt{\eta} \boldsymbol{\epsilon}_{t-1}, \tag{44}$$
where $\boldsymbol{\epsilon}_{t-1} \sim \mathcal{N}(0, I)$. Then, the distribution of $\boldsymbol{x}_T$ equals $p(\boldsymbol{x})$ when $\eta \to 0$ and $T \to \infty$.

- Let us revisit our sampling procedure in Algorithm 2. To sample $\boldsymbol{z}_{t-1} \sim p(\boldsymbol{z}_{t-1}|\boldsymbol{z}_t, \boldsymbol{\phi}_t)$, we compute
$$\boldsymbol{z}_{t-1} = \frac{1}{\sqrt{1-\beta_t}} \left( \boldsymbol{z}_t - \frac{\beta_t}{\sqrt{1-\alpha_t}} \mathbf{g}_t[\boldsymbol{z}_t, \boldsymbol{\phi}_t] \right) + \sigma_t \boldsymbol{\epsilon}. \tag{45}$$
As we have the fact that the denoising model ulitmately learns the score function of the input data distribution, it can be viewed as the reverse Langevin procedure of (44).

- The squared norm objective loss given in (41) resembles denoising score matching (DSM) over multiple noise scales indexed by $t$. Also, the loss (41) can be viewed as the variational bound for the process defined in (40), which is a Langevin-like process. By combining these two observations, optimizing an objective resembling DSM is equivalent to using variational inference to fit the finite-time marginal of a sampling chain resembling Langevin dynamics.

.

In this part, we briefly introduced how this $\boldsymbol{\epsilon}$-prediction parameterization resembles Langevin dynamics and simplifies the diffusion model's variational bound to an objective that resembles denoising score matching (Ho et al. (2020)). Moreover, we note that the sampling algorithm of the Diffusion Model approximates the Langevin Monte Carlo (LMC) algorithm, which is a numerical solution to the stochastic differential equation known as the Langevin equation!

## 6 Conclusion

Diffusion models represent a significant breakthrough in the generative modeling domain, offering a novel approach to data generation that balances quality and diversity. Their method of mapping data through latent variables and the reverse denoising process demonstrate the advanced capabilities of these models. This report has followed a structured exploration, beginning with the initial introduction of denoising diffusion models by Sohl-Dickstein et al. (2015). The loss function could be simplified based on the evidence lower bound (ELBO), which leads to a least-squares formulation. Additionally, this report has elucidated the link to the score-matching method as introduced in Song and Ermon (2019) and its resemblance to Langevin dynamics (Ho et al. (2020)), further highlighting the nature of diffusion models.

As this field continues to mature, the potential applications of diffusion models in various domains like image and audio generation, and even in more complex data forms, are expansive. The ongoing development and refinement of these models will undoubtedly be a pivotal area of research, contributing substantially to the advancements in machine learning and artificial intelligence.

## References

Goodfellow, I., Pouget-Abadie, J., Mirza, M., Xu, B., Warde-Farley, D., Ozair, S., Courville, A. and Bengio, Y. (2014), 'Generative adversarial nets', *Advances in neural information processing systems* **27**.

Ho, J., Jain, A. and Abbeel, P. (2020), 'Denoising diffusion probabilistic models', *Advances in neural information processing systems* **33**, 6840–6851.

Kingma, D. P. and Welling, M. (2014), Auto-encoding variational bayes, *in* 'Proceedings of the International Conference on Learning Representations (ICLR)'.

Prince, S. J. (2023), *Understanding Deep Learning.*, MIT PRESS.

Rezende, D. and Mohamed, S. (2015), Variational inference with normalizing flows, *in* 'International conference on machine learning', PMLR, pp. 1530–1538.

Sohl-Dickstein, J., Weiss, E., Maheswaranathan, N. and Ganguli, S. (2015), Deep unsupervised learning using nonequilibrium thermodynamics, *in* 'International conference on machine learning', PMLR, pp. 2256–2265.

Song, Y. and Ermon, S. (2019), *Generative Modeling by Estimating Gradients of the Data Distribution*, Curran Associates Inc., Red Hook, NY, USA.

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