Diffusion Models: Tutorial and Survey

Benyamin Ghojogh, Ali Ghodsi

Diffusion Models: Tutorial and Survey

Benyamin Ghojogh, Ali Ghodsi {BGHOJOGH, ALI.GHODSI}@UWATERLOO.CA
Waterloo, Ontario, Canada

Abstract

Diffusion models are a family of generative models which work based on a Markovian process. In their forward process, they gradually add noise to data until it becomes a complete noise. In the backward process, the data are gradually generated out of noise. In this tutorial paper, the Denoising Diffusion Probabilistic Model (DDPM) is fully explained. Detailed simplification of the variational lower bound of its likelihood, parameters of the distributions, and the loss function of the diffusion model are discussed. Some modifications to the original DDPM, including non-fixed covariance matrix, reducing the gradient noise, improving the noise schedule, and non-standard Gaussian noise distribution, and conditional diffusion model are introduced. Finally, continuous noise schedule by Stochastic Differential Equation (SDE), where the noise schedule is in a continuous domain, is explained.

1. Introduction

Diffusion models take their inspiration from the principles of non-equilibrium thermodynamics. They use a Markov chain (Markovian process) to gradually introduce noise into data and then they learn to reverse this process to recreate the desired data from the noise.

As shown in Fig. 1, a diffusion model has two processes with opposite directions. In the fixed forward diffusion process, it starts with the data and gradually adds noise to it until it becomes a complete noise. Then, in the generative reverse denoising process, it reverses that forward process and generates data out of the noise by denoising it gradually.

As Fig. 2 depicts, the forward and reverse processes can be modeled by the encoder and decoder of an autoencoder, respectively. As the forward process merely needs to make data noisy, it does not involve any learning and thus is fixed. Therefore, only the decoder needs to be learned in diffusion models. This is one of the differences of diffusion models from variational autoencoder (Kingma & Welling, 2014; Ghojogh et al., 2022) in which both encoder and decoder are learned.

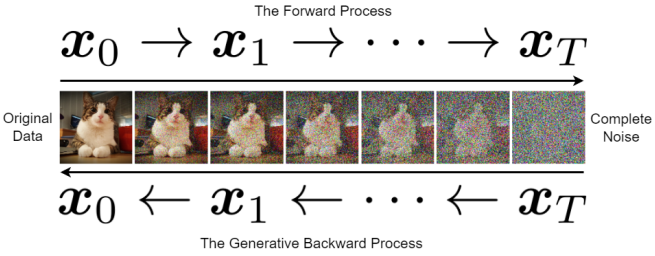

Figure 1. The forward and backward processes of the diffusion model. The credit of the used images is for Arash Vahdat.

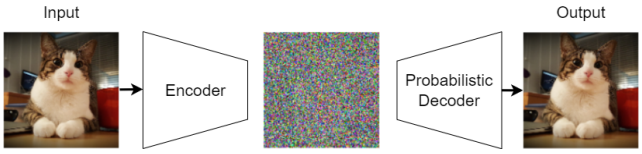

Figure 2. The diffusion model as an autoencoder.

2. Denoising Diffusion Probabilistic Model (DDPM)

The first and original diffusion model is the Denoising Diffusion Probabilistic Model (DDPM), proposed in (Ho et al., 2020). This section explains DDPM in details.

The process of adding noise to data can be seen as a sequence or chain $\{\boldsymbol{x}_0, \boldsymbol{x}_1, \dots, \boldsymbol{x}_T\}$ where $\boldsymbol{x}_0 \in \mathbb{R}^d$ and $\boldsymbol{x}_T \in \mathbb{R}^d$ are the original data and data completely corrupted with noise, respectively. Figure 3 depicts such a chain. In this notation, $\boldsymbol{x}_{t+1}$ is $\boldsymbol{x}_t$ plus some noise. As the noise is random, there can be a conditional distribution $q(\boldsymbol{x}_{t+1}|\boldsymbol{x}_t)$ to model the probability of obtaining $\boldsymbol{x}_{t+1}$ from $\boldsymbol{x}_t$. The goal of diffusion models is to learn the conditional distribution $q(\boldsymbol{x}_t|\boldsymbol{x}_{t+1})$ modeling how to generate data out of noise. This is because when $q(\boldsymbol{x}_t|\boldsymbol{x}_{t+1})$ is learned, t can be swept from T to 0 to generate a non-noisy data instance out of the complete noise.

The distribution $q(\boldsymbol{x}_t|\boldsymbol{x}_{t+1})$ is intractable and it should be approximated using a neural network. Let $p_\theta(\boldsymbol{x}_t|\boldsymbol{x}_{t+1})$ denote the neural network with weights θ which approx-

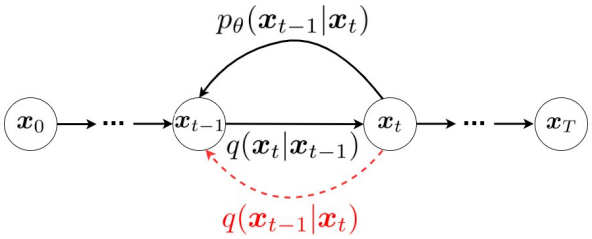

Figure 3. The conditional distributions in the forward and backward processes of the diffusion model.

imates this conditional distribution. It is noteworthy that $q(\boldsymbol{x}_t|\boldsymbol{x}_{t+1}, \boldsymbol{x}_0)$ is obviously tractable because it is conditioning on the original data $\boldsymbol{x}_0$. However, modeling $q(\boldsymbol{x}_t|\boldsymbol{x}_{t+1}, \boldsymbol{x}_0)$ is not useful because the goal of diffusion models is to find $\boldsymbol{x}_0$ and the goal cannot be used in the model. In fact, that is a chicken and egg problem. Although this distribution is not practical, but it will appear later in the derivations of this chapter.

2.1. The Forward Process

The forward process is for making data noisy gradually:

$$\boldsymbol{x}_0 \to \boldsymbol{x}_1 \to \cdots \to \boldsymbol{x}_T. \tag{1}$$

The forward process is defined by the following Gaussian conditional distribution:

$$q(\boldsymbol{x}_t|\boldsymbol{x}_{t-1}) = \mathcal{N}(\boldsymbol{x}_t; \sqrt{1-\beta_t}\boldsymbol{x}_{t-1}, \beta_t \boldsymbol{I}), \tag{2}$$

where $\sqrt{1-\beta_t}\boldsymbol{x}_{t-1}$ is the mean and $\beta_t \boldsymbol{I}$ is the covariance, $\boldsymbol{I}$ is the identity matrix, and $\beta_t \in (0, 1)$ is the noise level parameter in a pre-determined noise schedule:

$$\beta_1 < \beta_2 < \cdots < \beta_T. \tag{3}$$

The β_t parameter increases gradually in the forward process, so that the mean $\sqrt{1-\beta_t}\boldsymbol{x}_{t-1}$ deviates more and more from the previously noisy data $\boldsymbol{x}_{t-1}$ and the variance $\beta_t \boldsymbol{I}$ increases. The closer the t gets to T, the more noisy the data $\boldsymbol{x}_t$ becomes.

The forward process is considered as a Markov chain with the first Markovian property. In the Markov chain, every state is merely dependant on its previous state and not the entire previous sequence (Ghojogh et al., 2019):

$$q(\boldsymbol{x}_t|\boldsymbol{x}_{t-1}, \boldsymbol{x}_{t-2}, \dots, \boldsymbol{x}_0) = q(\boldsymbol{x}_t|\boldsymbol{x}_{t-1}). \tag{4}$$

Therefore, the joint distribution of the forward process is:

$$q(\boldsymbol{x}_{1:T}|\boldsymbol{x}_0) = \prod_{t=1}^{T} q(\boldsymbol{x}_t|\boldsymbol{x}_{t-1}), \tag{5}$$

where $\boldsymbol{x}_{1:T}$ represents all latent states from $\boldsymbol{x}_1$ to $\boldsymbol{x}_T$ and $\boldsymbol{x}_0$ is the initial state, i.e., the original data. The probabilistic graphical model (Koller & Friedman, 2009) of the diffusion model is illustrated in Fig. 4 where the states $\boldsymbol{x}_{1:T}$ are the latent (or hidden) variables for the observed data $\boldsymbol{x}_0$. This figure also compares the diffusion model with the variational autoencoder where a variable $\boldsymbol{z}$ is the latent variable for the observed data $\boldsymbol{x}$.

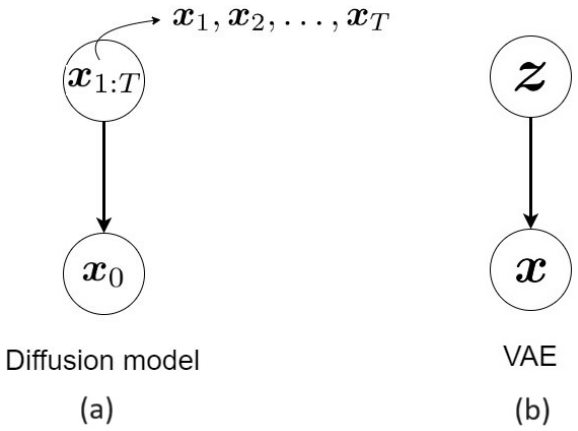

Figure 4. The probabilistic graphical models of (a) the diffusion model and (b) variational autoencoder (VAE).

2.2. The Generative Backward Process

The diffusion process, or the generative backward process, is the reverse process for generating data out of noise gradually:

$$\boldsymbol{x}_T \to \boldsymbol{x}_{T-1} \to \cdots \to \boldsymbol{x}_0. \tag{6}$$

As the forward process has Gaussian distributions in Eq. (2), it makes sense to use Gaussian distributions for the backward process, too. A generator machine learning model, with parameters or weights θ, is used to model the backward process:

$$p_\theta(\boldsymbol{x}_{t-1}|\boldsymbol{x}_t) = \mathcal{N}\big(\boldsymbol{x}_{t-1}; \boldsymbol{\mu}_\theta(\boldsymbol{x}_t; t), \boldsymbol{\Sigma}_\theta(\boldsymbol{x}_t; t)\big), \tag{7}$$

where $\boldsymbol{\mu}_\theta(\boldsymbol{x}_t; t) \in \mathbb{R}^d$ and $\boldsymbol{\Sigma}_\theta(\boldsymbol{x}_t; t) \in \mathbb{R}^{d \times d}$ are the mean and covariance of the distribution at iteration t, respectively. For simplicity, the covariance is usually set to be isotropic and diagonal:

$$\boldsymbol{\Sigma}_\theta(\boldsymbol{x}_t; t) = \sigma_t^2 \boldsymbol{I}. \tag{8}$$

The goal of diffusion model is to learn the parameters of this Gaussian distribution, which are the mean and variance.

The generative process is also considered as a Markov chain with the first Markovian property (Ghojogh et al., 2019):

$$q(\boldsymbol{x}_{t-1}|\boldsymbol{x}_t, \boldsymbol{x}_{t+1}, \dots, \boldsymbol{x}_T) = q(\boldsymbol{x}_{t-1}|\boldsymbol{x}_t). \tag{9}$$

Therefore, the joint distribution of the generative process is:

$$p_\theta(\boldsymbol{x}_{0:T}) = p(\boldsymbol{x}_T) \prod_{t=1}^{T} p_\theta(\boldsymbol{x}_{t-1}|\boldsymbol{x}_t), \tag{10}$$

where $\boldsymbol{x}_{0:T}$ represents all states from $\boldsymbol{x}_T$ to $\boldsymbol{x}_0$ and $\boldsymbol{x}_T$ is the state corresponding to the complete noise. Usually, the $p(\boldsymbol{x}_T)$ is defined to be the noise with standard Gaussian distribution:

$$p(\boldsymbol{x}_T) = \mathcal{N}(\mathbf{0}, \boldsymbol{I}). \tag{11}$$

2.3. Variational Lower Bound of the Likelihood

As discussed before, variational autoencoders consider a latent variable $\boldsymbol{z}$ as the latent factor for the observed data $\boldsymbol{x}$. Likewise, the diffusion models consider the latent variables $\{\boldsymbol{x}_1, \boldsymbol{x}_2, \dots, \boldsymbol{x}_T\}$ for the observed data $\boldsymbol{x}_0$ (see Fig. 4). The likelihood of data in the variational autoencoders and diffusion models are:

$$p_\theta(\boldsymbol{x}) = \int p_\theta(\boldsymbol{x}, \boldsymbol{z})\, d\boldsymbol{z}, \tag{12}$$

$$p_\theta(\boldsymbol{x}_0) = \int p_\theta(\boldsymbol{x}_0, \boldsymbol{x}_{1:T})\, d\boldsymbol{x}_{1:T}, \tag{13}$$

respectively. Recall that in variational inference, maximization of the likelihood was not tractable so a lower bound of the likelihood, named evidence lower bound (ELBO), was maximized instead (Ghojogh et al., 2021):

$$\mathbb{E}_{q(\boldsymbol{z}|\boldsymbol{x})}\big[\log p_\theta(\boldsymbol{x}|\boldsymbol{z})\big] - \mathrm{KL}\big(q(\boldsymbol{z}|\boldsymbol{x}) \| p_\theta(\boldsymbol{z})\big) \leq p_\theta(\boldsymbol{x}), \tag{14}$$

where $\mathbb{E}[.]$ denotes expectation and $\mathrm{KL}(.\|.)$ denotes the Kullback-Leibler (KL) divergence (Kullback & Leibler, 1951). See (Ghojogh et al., 2021; 2023) for the derivation of the ELBO in Eq. (14).

Similarly, the diffusion models maximize the variational lower bound of the likelihood of data:

$$\begin{aligned}\mathcal{L} :=\ & \mathbb{E}_{q(\boldsymbol{x}_{1:T}|\boldsymbol{x})}\big[\log p_\theta(\boldsymbol{x}_0|\boldsymbol{x}_{1:T})\big] \\ & - \mathrm{KL}\big(q(\boldsymbol{x}_{1:T}|\boldsymbol{x}_0) \| p_\theta(\boldsymbol{x}_{1:T})\big) \leq p_\theta(\boldsymbol{x}_0).\end{aligned} \tag{15}$$

The variational lower bound of likelihood in diffusion models can be simplified as follows:

$$\begin{aligned}\mathcal{L} &= \mathbb{E}_{q(\boldsymbol{x}_{1:T}|\boldsymbol{x}_0)}\big[\log p_\theta(\boldsymbol{x}_0|\boldsymbol{x}_{1:T})\big] \\ &\quad - \mathrm{KL}\big(q(\boldsymbol{x}_{1:T}|\boldsymbol{x}_0) \| p_\theta(\boldsymbol{x}_{1:T})\big) \\ &\overset{(a)}{=} \mathbb{E}_{q(\boldsymbol{x}_{1:T}|\boldsymbol{x}_0)}\big[\log p_\theta(\boldsymbol{x}_0|\boldsymbol{x}_{1:T})\big] \\ &\quad - \int q(\boldsymbol{x}_{1:T}|\boldsymbol{x}_0) \log \frac{q(\boldsymbol{x}_{1:T}|\boldsymbol{x}_0)}{p_\theta(\boldsymbol{x}_{1:T})} d\boldsymbol{x}_{1:T} \\ &\overset{(b)}{=} \mathbb{E}_{q(\boldsymbol{x}_{1:T}|\boldsymbol{x}_0)}\big[\log p_\theta(\boldsymbol{x}_0|\boldsymbol{x}_{1:T})\big] \\ &\quad - \mathbb{E}_{q(\boldsymbol{x}_{1:T}|\boldsymbol{x}_0)}\Big[\log \frac{q(\boldsymbol{x}_{1:T}|\boldsymbol{x}_0)}{p_\theta(\boldsymbol{x}_{1:T})}\Big] \\ &\overset{(c)}{=} \mathbb{E}_{q(\boldsymbol{x}_{1:T}|\boldsymbol{x}_0)}\Big[\log p_\theta(\boldsymbol{x}_0|\boldsymbol{x}_{1:T}) - \log \frac{q(\boldsymbol{x}_{1:T}|\boldsymbol{x}_0)}{p_\theta(\boldsymbol{x}_{1:T})}\Big] \\ &\overset{(d)}{=} \mathbb{E}_{q(\boldsymbol{x}_{1:T}|\boldsymbol{x}_0)}\Big[\log p_\theta(\boldsymbol{x}_0|\boldsymbol{x}_{1:T}) + \log \frac{p_\theta(\boldsymbol{x}_{1:T})}{q(\boldsymbol{x}_{1:T}|\boldsymbol{x}_0)}\Big] \\ &\overset{(e)}{=} \mathbb{E}_{q(\boldsymbol{x}_{1:T}|\boldsymbol{x}_0)}\Big[\log \frac{p_\theta(\boldsymbol{x}_0|\boldsymbol{x}_{1:T})\, p_\theta(\boldsymbol{x}_{1:T})}{q(\boldsymbol{x}_{1:T}|\boldsymbol{x}_0)}\Big] \\ &\overset{(f)}{=} \mathbb{E}_{q(\boldsymbol{x}_{1:T}|\boldsymbol{x}_0)}\Big[\log \frac{p_\theta(\boldsymbol{x}_{0:T})}{q(\boldsymbol{x}_{1:T}|\boldsymbol{x}_0)}\Big],\end{aligned} \tag{16}$$

where (a) is because of the definition of the KL divergence, (b) is because of the definition of the expectation in the second term of expression, (c) is because expectation is a linear operator, (d) and (e) are because of the property of logarithm, and (f) is because $p_\theta(\boldsymbol{x}_0|\boldsymbol{x}_{1:T})\, p_\theta(\boldsymbol{x}_{1:T}) = p_\theta(\boldsymbol{x}_0, \boldsymbol{x}_{1:T}) = p_\theta(\boldsymbol{x}_{0:T})$.

The Eq. (16) can be simplified further:

$$\begin{aligned}\mathcal{L} &= \mathbb{E}_{q(\boldsymbol{x}_{1:T}|\boldsymbol{x}_0)}\Big[\log \frac{p_\theta(\boldsymbol{x}_{0:T})}{q(\boldsymbol{x}_{1:T}|\boldsymbol{x}_0)}\Big] \\ &\overset{(10)}{=} \mathbb{E}_{q(\boldsymbol{x}_{1:T}|\boldsymbol{x}_0)}\Big[\log \frac{p(\boldsymbol{x}_T) \prod_{t=1}^{T} p_\theta(\boldsymbol{x}_{t-1}|\boldsymbol{x}_t)}{q(\boldsymbol{x}_{1:T}|\boldsymbol{x}_0)}\Big] \\ &\overset{(a)}{=} \mathbb{E}_{q(\boldsymbol{x}_{1:T}|\boldsymbol{x}_0)}\big[\log p(\boldsymbol{x}_T)\big] \\ &\quad + \sum_{t=1}^{T} \mathbb{E}_{q(\boldsymbol{x}_{1:T}|\boldsymbol{x}_0)}\big[\log p_\theta(\boldsymbol{x}_{t-1}|\boldsymbol{x}_t)\big] \\ &\quad - \mathbb{E}_{q(\boldsymbol{x}_{1:T}|\boldsymbol{x}_0)}\big[\log q(\boldsymbol{x}_{1:T}|\boldsymbol{x}_0)\big] \\ &\overset{(5)}{=} \mathbb{E}_{q(\boldsymbol{x}_{1:T}|\boldsymbol{x}_0)}\big[\log p(\boldsymbol{x}_T)\big] \\ &\quad + \sum_{t=1}^{T} \mathbb{E}_{q(\boldsymbol{x}_{1:T}|\boldsymbol{x}_0)}\big[\log p_\theta(\boldsymbol{x}_{t-1}|\boldsymbol{x}_t)\big] \\ &\quad - \mathbb{E}_{q(\boldsymbol{x}_{1:T}|\boldsymbol{x}_0)}\Big[\log \prod_{t=1}^{T} q(\boldsymbol{x}_t|\boldsymbol{x}_{t-1})\Big]\end{aligned}$$

$$
\begin{aligned}
&\overset{(b)}{=} \mathbb{E}_{q(\boldsymbol{x}_{1:T}|\boldsymbol{x}_0)}\big[\log p(\boldsymbol{x}_T)\big] \\
&\quad + \sum_{t=1}^{T} \mathbb{E}_{q(\boldsymbol{x}_{1:T}|\boldsymbol{x}_0)}\big[\log p_\theta(\boldsymbol{x}_{t-1}|\boldsymbol{x}_t)\big] \\
&\quad - \sum_{t=1}^{T} \mathbb{E}_{q(\boldsymbol{x}_{1:T}|\boldsymbol{x}_0)}\big[\log q(\boldsymbol{x}_t|\boldsymbol{x}_{t-1})\big] \\
&\overset{(c)}{=} \mathbb{E}_{q(\boldsymbol{x}_{1:T}|\boldsymbol{x}_0)}\big[\log p(\boldsymbol{x}_T)\big] \\
&\quad + \sum_{t=1}^{T} \mathbb{E}_{q(\boldsymbol{x}_{1:T}|\boldsymbol{x}_0)}\big[\log \frac{p_\theta(\boldsymbol{x}_{t-1}|\boldsymbol{x}_t)}{q(\boldsymbol{x}_t|\boldsymbol{x}_{t-1})}\big] \\
&\overset{(d)}{=} \mathbb{E}_{q(\boldsymbol{x}_{1:T}|\boldsymbol{x}_0)}\bigg[\log p(\boldsymbol{x}_T) + \sum_{t=1}^{T} \log \frac{p_\theta(\boldsymbol{x}_{t-1}|\boldsymbol{x}_t)}{q(\boldsymbol{x}_t|\boldsymbol{x}_{t-1})}\bigg],
\end{aligned} \tag{17}
$$

where (a) is because of the properties of logarithm and linearity of the expectation, (b) is because logarithm of multiplication becomes summation of logarithms, (c) is for the properties of logarithm, and (d) is because expectation is a linear operator.

According to the definition of the KL divergence, the Eq. (17), i.e., the variational lower bound, can be stated as follows:

$$
\begin{aligned}
\mathcal{L} = \mathbb{E}_{q(\boldsymbol{x}_{1:T}|\boldsymbol{x}_0)}\big[\log p(\boldsymbol{x}_T)\big] \\
- \mathrm{KL}\big(q(\boldsymbol{x}_t|\boldsymbol{x}_{t-1}) \| p_\theta(\boldsymbol{x}_{t-1}|\boldsymbol{x}_t)\big).
\end{aligned}
$$

This is the variational lower bound which needs to be maximized; therefore, the KL divergence is minimized. Here, the KL divergence is between the forward process $q(\boldsymbol{x}_t|\boldsymbol{x}_{t-1})$ and the generative process $p_\theta(\boldsymbol{x}_{t-1}|\boldsymbol{x}_t)$. Minimization of this KL divergence makes sense because the distribution of the generative process should mimic the distribution of the forward process.

As the sequence of states in the diffusion process is a Markov chain, the following holds:

$$
q(\boldsymbol{x}_t|\boldsymbol{x}_{t-1}) = q(\boldsymbol{x}_t|\boldsymbol{x}_{t-1}, \boldsymbol{x}_0). \tag{18}
$$

According to the Bayes' rule, there is:

$$
q(\boldsymbol{x}_t|\boldsymbol{x}_{t-1}, \boldsymbol{x}_0) = \frac{q(\boldsymbol{x}_{t-1}|\boldsymbol{x}_t, \boldsymbol{x}_0)\, q(\boldsymbol{x}_t|\boldsymbol{x}_0)}{q(\boldsymbol{x}_{t-1}|\boldsymbol{x}_0)}, \tag{19}
$$

where $q(\boldsymbol{x}_{t-1}|\boldsymbol{x}_t, \boldsymbol{x}_0)$ is tractable in contrast to $q(\boldsymbol{x}_{t-1}|\boldsymbol{x}_t)$ because it is conditioning on the original data. In fact, Eqs. (18) and (19) are used to make $q(\boldsymbol{x}_{t-1}|\boldsymbol{x}_t)$ tractable. The tractable distribution $q(\boldsymbol{x}_{t-1}|\boldsymbol{x}_t, \boldsymbol{x}_0)$ is called the *ground truth denoising distribution* because it sees the original data $\boldsymbol{x}_0$.

Plugging Eqs. (18) and (19) in Eq. (17) results in:

$$
\begin{aligned}
\mathcal{L} &= \mathbb{E}_{q(\boldsymbol{x}_{1:T}|\boldsymbol{x}_0)}\bigg[\log p(\boldsymbol{x}_T) \\
&\quad + \sum_{t=1}^{T} \log \frac{p_\theta(\boldsymbol{x}_{t-1}|\boldsymbol{x}_t)\, q(\boldsymbol{x}_{t-1}|\boldsymbol{x}_0)}{q(\boldsymbol{x}_{t-1}|\boldsymbol{x}_t, \boldsymbol{x}_0)\, q(\boldsymbol{x}_t|\boldsymbol{x}_0)}\bigg] \\
&\overset{(a)}{=} \mathbb{E}_{q(\boldsymbol{x}_{1:T}|\boldsymbol{x}_0)}\bigg[\log p(\boldsymbol{x}_T) + \sum_{t=2}^{T} \log \frac{p_\theta(\boldsymbol{x}_{t-1}|\boldsymbol{x}_t)}{q(\boldsymbol{x}_{t-1}|\boldsymbol{x}_t, \boldsymbol{x}_0)} \\
&\quad + \sum_{t=2}^{T} \log \frac{q(\boldsymbol{x}_{t-1}|\boldsymbol{x}_0)}{q(\boldsymbol{x}_t|\boldsymbol{x}_0)} + \log \frac{p_\theta(\boldsymbol{x}_0|\boldsymbol{x}_1)\, q(\boldsymbol{x}_0|\boldsymbol{x}_0)}{q(\boldsymbol{x}_0|\boldsymbol{x}_1, \boldsymbol{x}_0)\, q(\boldsymbol{x}_1|\boldsymbol{x}_0)}\bigg] \\
&\overset{(b)}{=} \mathbb{E}_{q(\boldsymbol{x}_{1:T}|\boldsymbol{x}_0)}\bigg[\log p(\boldsymbol{x}_T) + \sum_{t=2}^{T} \log \frac{p_\theta(\boldsymbol{x}_{t-1}|\boldsymbol{x}_t)}{q(\boldsymbol{x}_{t-1}|\boldsymbol{x}_t, \boldsymbol{x}_0)} \\
&\quad + \sum_{t=2}^{T} \log \frac{q(\boldsymbol{x}_{t-1}|\boldsymbol{x}_0)}{q(\boldsymbol{x}_t|\boldsymbol{x}_0)} + \log \frac{p_\theta(\boldsymbol{x}_0|\boldsymbol{x}_1)}{q(\boldsymbol{x}_1|\boldsymbol{x}_0)}\bigg],
\end{aligned} \tag{20}
$$

where (a) breaks down the logarithm and takes out the $t = 1$ from the summation, and (b) is because $q(\boldsymbol{x}_0|\boldsymbol{x}_0) = q(\boldsymbol{x}_0|\boldsymbol{x}_1, \boldsymbol{x}_0) = 1$. The third term in the expectation is a telescoping summation and can be simplified as follows:

$$
\begin{aligned}
&\sum_{t=2}^{T} \log \frac{q(\boldsymbol{x}_{t-1}|\boldsymbol{x}_0)}{q(\boldsymbol{x}_t|\boldsymbol{x}_0)} = \\
&\log q(\boldsymbol{x}_{T-1}|\boldsymbol{x}_0) + \cdots + \log q(\boldsymbol{x}_2|\boldsymbol{x}_0) + \log q(\boldsymbol{x}_1|\boldsymbol{x}_0) \\
&- \log q(\boldsymbol{x}_T|\boldsymbol{x}_0) - \log q(\boldsymbol{x}_{T-1}|\boldsymbol{x}_0) - \cdots - \log q(\boldsymbol{x}_2|\boldsymbol{x}_0) \\
&= -\log q(\boldsymbol{x}_T|\boldsymbol{x}_0) + \log q(\boldsymbol{x}_1|\boldsymbol{x}_0) = \log \frac{q(\boldsymbol{x}_1|\boldsymbol{x}_0)}{q(\boldsymbol{x}_T|\boldsymbol{x}_0)}.
\end{aligned}
$$

Therefore, the variational lower bound in Eq. (20) becomes:

$$
\begin{aligned}
\mathcal{L} &= \mathbb{E}_{q(\boldsymbol{x}_{1:T}|\boldsymbol{x}_0)}\bigg[\log p(\boldsymbol{x}_T) + \sum_{t=2}^{T} \log \frac{p_\theta(\boldsymbol{x}_{t-1}|\boldsymbol{x}_t)}{q(\boldsymbol{x}_{t-1}|\boldsymbol{x}_t, \boldsymbol{x}_0)} \\
&+ \log \frac{q(\boldsymbol{x}_1|\boldsymbol{x}_0)}{q(\boldsymbol{x}_T|\boldsymbol{x}_0)} + \log \frac{p_\theta(\boldsymbol{x}_0|\boldsymbol{x}_1)}{q(\boldsymbol{x}_1|\boldsymbol{x}_0)}\bigg] \\
&\overset{(a)}{=} \mathbb{E}_{q(\boldsymbol{x}_{1:T}|\boldsymbol{x}_0)}\bigg[\log \frac{p(\boldsymbol{x}_T)}{q(\boldsymbol{x}_T|\boldsymbol{x}_0)} + \sum_{t=2}^{T} \log \frac{p_\theta(\boldsymbol{x}_{t-1}|\boldsymbol{x}_t)}{q(\boldsymbol{x}_{t-1}|\boldsymbol{x}_t, \boldsymbol{x}_0)} \\
&+ \log p_\theta(\boldsymbol{x}_0|\boldsymbol{x}_1)\bigg]
\end{aligned}
$$

$$
\begin{aligned}
&\overset{(b)}{=} \mathbb{E}_{q(\boldsymbol{x}_{1:T}|\boldsymbol{x}_0)}\bigg[\log\frac{p(\boldsymbol{x}_T)}{q(\boldsymbol{x}_T|\boldsymbol{x}_0)}\bigg] \\
&+ \sum_{t=2}^{T} \mathbb{E}_{q(\boldsymbol{x}_{1:T}|\boldsymbol{x}_0)}\bigg[\log\frac{p_\theta(\boldsymbol{x}_{t-1}|\boldsymbol{x}_t)}{q(\boldsymbol{x}_{t-1}|\boldsymbol{x}_t,\boldsymbol{x}_0)}\bigg] \\
&+ \mathbb{E}_{q(\boldsymbol{x}_{1:T}|\boldsymbol{x}_0)}\big[\log p_\theta(\boldsymbol{x}_0|\boldsymbol{x}_1)\big] \\
&\overset{(c)}{=} -\mathrm{KL}\big(q(\boldsymbol{x}_T|\boldsymbol{x}_0)\|p(\boldsymbol{x}_T)\big) \\
&- \sum_{t=2}^{T} \mathrm{KL}\big(q(\boldsymbol{x}_{t-1}|\boldsymbol{x}_t,\boldsymbol{x}_0)\|p_\theta(\boldsymbol{x}_{t-1}|\boldsymbol{x}_t)\big) \\
&+ \mathbb{E}_{q(\boldsymbol{x}_{1:T}|\boldsymbol{x}_0)}\big[\log p_\theta(\boldsymbol{x}_0|\boldsymbol{x}_1)\big],
\end{aligned}
$$

where (a) is because of the properties of logarithm, (b) is because expectation is a linear operator, and (c) is because of the definition of KL divergence.

The variational lower bound of the likelihood should be maximized with respect to the parameters θ. The first KL divergence is constant with respect to θ so it can be dropped. Maximization of the lower bound is equivalent to minimization of it multiplied by negative one:

$$
\begin{aligned}
\underset{\theta}{\text{minimize}} \quad & \sum_{t=2}^{T} \mathrm{KL}\big(q(\boldsymbol{x}_{t-1}|\boldsymbol{x}_t,\boldsymbol{x}_0)\|p_\theta(\boldsymbol{x}_{t-1}|\boldsymbol{x}_t)\big) \\
& - \mathbb{E}_{q(\boldsymbol{x}_{1:T}|\boldsymbol{x}_0)}\big[\log p_\theta(\boldsymbol{x}_0|\boldsymbol{x}_1)\big].
\end{aligned} \tag{21}
$$

Therefore, it can be the loss function of a neural network with weights θ. This loss function makes sense because it learns the backward distribution $p_\theta(\boldsymbol{x}_{t-1}|\boldsymbol{x}_t)$ to become close to the ground truth denoising distribution $q(\boldsymbol{x}_{t-1}|\boldsymbol{x}_t,\boldsymbol{x}_0)$ by minimizing their KL divergence. That is why the focus of the loss function is on the KL divergence and the second term in Eq. (21) can be ignored. Remember this loss function and we will get back to it later in the chapter.

2.4. Parameters of the Distributions

2.4.1. Sampling in Forward Process at Arbitrary Iterations

Recall the Gaussian distributions of the forward process in Eq. (2). That conditional distribution was for sampling $\boldsymbol{x}_t$ given $\boldsymbol{x}_{t-1}$. However, it is also possible to sample $\boldsymbol{x}_t$ at any arbitrary iteration t in a closed form. Consider the following definitions:

$$
\alpha_t := 1 - \beta_t, \tag{22}
$$

$$
\bar{\alpha}_t = \prod_{i=1}^{t} \alpha_i. \tag{23}
$$

Therefore, Eq. (2) can be restated as:

$$
q(\boldsymbol{x}_t|\boldsymbol{x}_{t-1}) = \mathcal{N}\big(\boldsymbol{x}_t; \sqrt{\alpha_t}\boldsymbol{x}_{t-1}, (1-\alpha_t)\boldsymbol{I}\big). \tag{24}
$$

Here, the reparameterization technique (Kingma & Welling, 2014) can be used. In this technique, for sampling from a non-standard Gaussian distribution $\boldsymbol{z} \sim \mathcal{N}(\boldsymbol{\mu}, \sigma^2\boldsymbol{I})$, one can sample from a standard distribution instead, i.e., $\boldsymbol{\epsilon} \sim \mathcal{N}(\boldsymbol{0}, \boldsymbol{I})$ and obtain the sample $\boldsymbol{z}$ as $\boldsymbol{z} = \boldsymbol{\mu} + \sigma\boldsymbol{\epsilon}$. Here, using the reparameterization technique, the $\boldsymbol{\epsilon} \sim \mathcal{N}(\boldsymbol{0}, \boldsymbol{I})$ is sampled and then $\{\boldsymbol{x}_t, \boldsymbol{x}_{t-1}, \dots, \boldsymbol{x}_1\}$ are each obtained as:

$$
\boldsymbol{x}_t = \sqrt{\alpha_t}\boldsymbol{x}_{t-1} + \sqrt{1-\alpha_t}\boldsymbol{\epsilon}, \tag{25}
$$

$$
\boldsymbol{x}_{t-1} = \sqrt{\alpha_{t-1}}\boldsymbol{x}_{t-2} + \sqrt{1-\alpha_{t-1}}\boldsymbol{\epsilon}, \tag{26}
$$

$$
\boldsymbol{x}_{t-2} = \sqrt{\alpha_{t-2}}\boldsymbol{x}_{t-3} + \sqrt{1-\alpha_{t-2}}\boldsymbol{\epsilon}, \tag{27}
$$

$$
\vdots
$$

Plugging $\boldsymbol{x}_{t-1}$, Eq. (26), in $\boldsymbol{x}_t$, Eq. (25), gives:

$$
\begin{aligned}
\boldsymbol{x}_t &= \sqrt{\alpha_t}\big(\sqrt{\alpha_{t-1}}\boldsymbol{x}_{t-2} + \sqrt{1-\alpha_{t-1}}\boldsymbol{\epsilon}\big) + \sqrt{1-\alpha_t}\boldsymbol{\epsilon} \\
&= \sqrt{\alpha_t\alpha_{t-1}}\boldsymbol{x}_{t-2} + \sqrt{\alpha_t(1-\alpha_{t-1})}\boldsymbol{\epsilon} + \sqrt{1-\alpha_t}\boldsymbol{\epsilon}.
\end{aligned} \tag{28}
$$

Note that the $\boldsymbol{\epsilon}$ in the second and third terms are two different samples from the standard Gaussian distributions. Therefore, the second and third terms are two independent random Gaussian distributions:

$$
\begin{aligned}
&\sqrt{\alpha_t(1-\alpha_{t-1})}\boldsymbol{\epsilon} \sim \mathcal{N}\big(\boldsymbol{0}, \underbrace{\alpha_t(1-\alpha_{t-1})}_{=\alpha_t-\alpha_t\alpha_{t-1}}\big), \\
&\sqrt{1-\alpha_t}\boldsymbol{\epsilon} \sim \mathcal{N}(\boldsymbol{0}, 1-\alpha_t),
\end{aligned}
$$

so their summation becomes a Gaussian distribution whose variance is summation of the two variances:

$$
\sqrt{\alpha_t(1-\alpha_{t-1})}\boldsymbol{\epsilon} + \sqrt{1-\alpha_t}\boldsymbol{\epsilon} \sim \mathcal{N}(\boldsymbol{0}, 1-\alpha_t\alpha_{t-1}).
$$

As a result, Eq. (28) can be stated as:

$$
\boldsymbol{x}_t = \sqrt{\alpha_t\alpha_{t-1}}\boldsymbol{x}_{t-2} + \sqrt{1-\alpha_t\alpha_{t-1}}\boldsymbol{\epsilon}
$$

Plugging $\boldsymbol{x}_{t-2}$, Eq. (27), in this expression gives:

$$
\begin{aligned}
\boldsymbol{x}_t &= \sqrt{\alpha_t\alpha_{t-1}}\Big(\sqrt{\alpha_{t-2}}\boldsymbol{x}_{t-3} + \sqrt{1-\alpha_{t-2}}\boldsymbol{\epsilon}\Big) \\
&+ \sqrt{1-\alpha_t\alpha_{t-1}}\boldsymbol{\epsilon} \\
&= \sqrt{\alpha_t\alpha_{t-1}\alpha_{t-2}}\boldsymbol{x}_{t-3} + \sqrt{\alpha_t\alpha_{t-1} - \alpha_t\alpha_{t-1}\alpha_{t-2}}\boldsymbol{\epsilon} \\
&+ \sqrt{1-\alpha_t\alpha_{t-1}}\boldsymbol{\epsilon}.
\end{aligned} \tag{29}
$$

Again, the second and third terms are two independent random Gaussian distributions:

$$
\begin{aligned}
&\sqrt{\alpha_t\alpha_{t-1} - \alpha_t\alpha_{t-1}\alpha_{t-2}}\boldsymbol{\epsilon} \\
&\qquad\qquad \sim \mathcal{N}(\boldsymbol{0}, \alpha_t\alpha_{t-1} - \alpha_t\alpha_{t-1}\alpha_{t-2}), \\
&\sqrt{1-\alpha_t\alpha_{t-1}}\boldsymbol{\epsilon} \sim \mathcal{N}(\boldsymbol{0}, 1-\alpha_t\alpha_{t-1}),
\end{aligned}
$$

so their summation becomes a Gaussian distribution whose variance is summation of the two variances:

$$\sqrt{\alpha_t\alpha_{t-1} - \alpha_t\alpha_{t-1}\alpha_{t-2}}\boldsymbol{\epsilon} + \sqrt{1-\alpha_t\alpha_{t-1}}\boldsymbol{\epsilon} \sim \mathcal{N}\big(\mathbf{0}, 1-\alpha_t\alpha_{t-1}\alpha_{t-2}\big).$$

As a result, Eq. (29) can be stated as:

$$\boldsymbol{x}_t = \sqrt{\alpha_t\alpha_{t-1}\alpha_{t-2}}\boldsymbol{x}_{t-3} + \sqrt{1-\alpha_t\alpha_{t-1}\alpha_{t-2}}\boldsymbol{\epsilon}.$$

By induction, continuing this recursion until $\boldsymbol{x}_0$ gives $\boldsymbol{x}_t$ in terms of the original data $\boldsymbol{x}_0$ and the noise $\boldsymbol{\epsilon}$:

$$\begin{aligned}\boldsymbol{x}_t &= \sqrt{\prod_{i=1}^{t}\alpha_i}\,\boldsymbol{x}_0 + \sqrt{1-\prod_{i=1}^{t}\alpha_i}\,\boldsymbol{\epsilon} \\ &\overset{(23)}{=} \sqrt{\bar{\alpha}_t}\boldsymbol{x}_0 + \sqrt{1-\bar{\alpha}_t}\boldsymbol{\epsilon}, \end{aligned} \tag{30}$$

which states $\boldsymbol{x}_t$ in terms of the original data $\boldsymbol{x}_0$ and noise $\boldsymbol{\epsilon}$. As $\boldsymbol{x}_0$ is not random and $\boldsymbol{\epsilon}$ has standard Gaussian distribution, the distribution of $\boldsymbol{x}_t$ in this equation has mean $\sqrt{\bar{\alpha}_t}\boldsymbol{x}_0$ and covariance $(1-\bar{\alpha}_t)\boldsymbol{I}$. Therefore:

$$q(\boldsymbol{x}_t|\boldsymbol{x}_0) = \mathcal{N}\big(\boldsymbol{x}_t; \sqrt{\bar{\alpha}_t}\boldsymbol{x}_0, (1-\bar{\alpha}_t)\boldsymbol{I}\big). \tag{31}$$

By this conditional distribution, the noisy data $\boldsymbol{x}_t$ at any iteration t can be sampled based on the original data $\boldsymbol{x}_0$.

2.4.2. The Ground Truth Denoising Distribution

A neural network is required to learn the parameters, i.e., the mean and covariance, of the backward process in Eq. (7). Similar to Eq. (7), the ground truth denoising distribution can be a Gaussian distribution:

$$q(\boldsymbol{x}_{t-1}|\boldsymbol{x}_t, \boldsymbol{x}_0) = \mathcal{N}\big(\boldsymbol{x}_{t-1}; \widetilde{\boldsymbol{\mu}}(\boldsymbol{x}_t, \boldsymbol{x}_0), \widetilde{\boldsymbol{\Sigma}}(\boldsymbol{x}_t, \boldsymbol{x}_0)\big), \tag{32}$$

where $\widetilde{\boldsymbol{\mu}}(\boldsymbol{x}_t, \boldsymbol{x}_0) \in \mathbb{R}^d$ and $\widetilde{\boldsymbol{\Sigma}}(\boldsymbol{x}_t, \boldsymbol{x}_0) \in \mathbb{R}^{d\times d}$ are the mean and covariance of the distribution, respectively. For simplicity, similar to Eq. (8), the covariance is usually set to be isotropic and diagonal, i.e., $\widetilde{\boldsymbol{\Sigma}}(\boldsymbol{x}_t, \boldsymbol{x}_0) = \widetilde{\sigma}_t^2\boldsymbol{I}$.

By Bayes' rule, we have:

$$q(\boldsymbol{x}_{t-1}|\boldsymbol{x}_t, \boldsymbol{x}_0) = \frac{q(\boldsymbol{x}_t|\boldsymbol{x}_{t-1}, \boldsymbol{x}_0)\, q(\boldsymbol{x}_{t-1}|\boldsymbol{x}_0)}{q(\boldsymbol{x}_t|\boldsymbol{x}_0)}. \tag{33}$$

According to Eqs. (18), (22), (24), and (31), the probabilities in this Bayes' rule are:

$$\begin{aligned} q(\boldsymbol{x}_t|\boldsymbol{x}_{t-1}, \boldsymbol{x}_0) &\overset{(18)}{=} q(\boldsymbol{x}_t|\boldsymbol{x}_{t-1}) \\ &\overset{(22),(24)}{=} \mathcal{N}(\boldsymbol{x}_t; \sqrt{\alpha_t}\boldsymbol{x}_{t-1}, \beta_t\boldsymbol{I}), \\ &\propto \exp\Big(-\frac{1}{2}\Big(\frac{(\boldsymbol{x}_t - \sqrt{\alpha_t}\boldsymbol{x}_{t-1})^2}{\beta_t}\Big)\Big), \\ q(\boldsymbol{x}_{t-1}|\boldsymbol{x}_0) &\overset{(31)}{=} \mathcal{N}\big(\boldsymbol{x}_{t-1}; \sqrt{\bar{\alpha}_{t-1}}\boldsymbol{x}_0, (1-\bar{\alpha}_{t-1})\boldsymbol{I}\big), \\ &\propto \exp\Big(-\frac{1}{2}\Big(\frac{(\boldsymbol{x}_{t-1} - \sqrt{\bar{\alpha}_{t-1}}\boldsymbol{x}_0)^2}{1-\bar{\alpha}_{t-1}}\Big)\Big), \\ q(\boldsymbol{x}_t|\boldsymbol{x}_0) &\overset{(31)}{=} \mathcal{N}\big(\boldsymbol{x}_t; \sqrt{\bar{\alpha}_t}\boldsymbol{x}_0, (1-\bar{\alpha}_t)\boldsymbol{I}\big) \\ &\propto \exp\Big(-\frac{1}{2}\Big(\frac{(\boldsymbol{x}_t - \sqrt{\bar{\alpha}_t}\boldsymbol{x}_0)^2}{1-\bar{\alpha}_t}\Big)\Big). \end{aligned}$$

Therefore:

$$\begin{aligned} &q(\boldsymbol{x}_{t-1}|\boldsymbol{x}_t, \boldsymbol{x}_0) \\ &\overset{(33)}{\propto} \exp\Big(-\frac{1}{2}\Big(\frac{(\boldsymbol{x}_t - \sqrt{\alpha_t}\boldsymbol{x}_{t-1})^2}{\beta_t} + \frac{(\boldsymbol{x}_{t-1} - \sqrt{\bar{\alpha}_{t-1}}\boldsymbol{x}_0)^2}{1-\bar{\alpha}_{t-1}} \\ &\qquad - \frac{(\boldsymbol{x}_t - \sqrt{\bar{\alpha}_t}\boldsymbol{x}_0)^2}{1-\bar{\alpha}_t}\Big)\Big) \\ &\overset{(a)}{=} \exp\Big(-\frac{1}{2}\Big(\frac{\boldsymbol{x}_t^2 - 2\sqrt{\alpha_t}\boldsymbol{x}_t\boldsymbol{x}_{t-1} + \alpha_t\boldsymbol{x}_{t-1}^2}{\beta_t} \\ &\qquad + \frac{\boldsymbol{x}_{t-1}^2 - 2\sqrt{\bar{\alpha}_{t-1}}\boldsymbol{x}_0\boldsymbol{x}_{t-1} + \bar{\alpha}_{t-1}\boldsymbol{x}_0^2}{1-\bar{\alpha}_{t-1}} \\ &\qquad - \frac{(\boldsymbol{x}_t - \sqrt{\bar{\alpha}_t}\boldsymbol{x}_0)^2}{1-\bar{\alpha}_t}\Big)\Big) \\ &\overset{(b)}{=} \exp\Big(-\frac{1}{2}\Big((\frac{\alpha_t}{\beta_t} + \frac{1}{1-\bar{\alpha}_{t-1}})\boldsymbol{x}_{t-1}^2 \\ &\qquad - 2\big(\frac{\sqrt{\alpha_t}}{\beta_t}\boldsymbol{x}_t + \frac{\sqrt{\bar{\alpha}_{t-1}}}{1-\bar{\alpha}_{t-1}}\boldsymbol{x}_0\big)\boldsymbol{x}_{t-1} + C(\boldsymbol{x}_t, \boldsymbol{x}_0)\Big)\Big) \\ &\overset{(c)}{\propto} \exp\Big(-\frac{1}{2}\frac{\Big(\boldsymbol{x}_{t-1} - \frac{\frac{\sqrt{\alpha_t}}{\beta_t}\boldsymbol{x}_t + \frac{\sqrt{\bar{\alpha}_{t-1}}}{1-\bar{\alpha}_{t-1}}\boldsymbol{x}_0}{\frac{\alpha_t}{\beta_t} + \frac{1}{1-\bar{\alpha}_{t-1}}}\Big)^2}{1/(\frac{\alpha_t}{\beta_t} + \frac{1}{1-\bar{\alpha}_{t-1}})}\Big), \end{aligned} \tag{34}$$

where (a) and (c) are because of the binomial theorem, (b) is because of factoring out the terms $\boldsymbol{x}_{t-1}^2$ and $\boldsymbol{x}_{t-1}$, and $C(\boldsymbol{x}_t, \boldsymbol{x}_0)$ is a function not including $\boldsymbol{x}_{t-1}$ and thus it can be dropped and ignored for optimization.

2.4.3. Mean of the Ground Truth Denoising Distribution

According to Eq. (34), the mean of the ground truth denoising distribution, in Eq. (32), is as follows:

$$\widetilde{\boldsymbol{\mu}}(\boldsymbol{x}_t, \boldsymbol{x}_0) = \frac{\frac{\sqrt{\alpha_t}}{\beta_t}\boldsymbol{x}_t + \frac{\sqrt{\bar{\alpha}_{t-1}}}{1-\bar{\alpha}_{t-1}}\boldsymbol{x}_0}{\frac{\alpha_t}{\beta_t} + \frac{1}{1-\bar{\alpha}_{t-1}}} = \frac{\frac{\sqrt{\alpha_t}}{\beta_t}\boldsymbol{x}_t + \frac{\sqrt{\bar{\alpha}_{t-1}}}{1-\bar{\alpha}_{t-1}}\boldsymbol{x}_0}{\frac{\alpha_t - \alpha_t\bar{\alpha}_{t-1} + \beta_t}{\beta_t(1-\bar{\alpha}_{t-1})}}$$
$$\overset{(a)}{=} \frac{\frac{\sqrt{\alpha_t}}{\beta_t}\boldsymbol{x}_t + \frac{\sqrt{\bar{\alpha}_{t-1}}}{1-\bar{\alpha}_{t-1}}\boldsymbol{x}_0}{\frac{1-\bar{\alpha}_t}{\beta_t(1-\bar{\alpha}_{t-1})}}$$
$$= \Big(\frac{\sqrt{\alpha_t}}{\beta_t}\boldsymbol{x}_t + \frac{\sqrt{\bar{\alpha}_{t-1}}}{1-\bar{\alpha}_{t-1}}\boldsymbol{x}_0\Big)\frac{1-\bar{\alpha}_{t-1}}{1-\bar{\alpha}_t}\beta_t$$
$$= \frac{\sqrt{\alpha_t}(1-\bar{\alpha}_{t-1})}{1-\bar{\alpha}_t}\boldsymbol{x}_t + \frac{\sqrt{\bar{\alpha}_{t-1}}\beta_t}{1-\bar{\alpha}_t}\boldsymbol{x}_0, \tag{35}$$

where (a) is because $\alpha_t + \beta_t = 1$ and $\alpha_t\bar{\alpha}_{t-1} = \bar{\alpha}_t$ by Eqs. (22) and (23). According to Eq. (30), we have:

$$\boldsymbol{x}_t = \sqrt{\bar{\alpha}_t}\boldsymbol{x}_0 + \sqrt{1-\bar{\alpha}_t}\boldsymbol{\epsilon}$$
$$\implies \boldsymbol{x}_0 = \frac{1}{\sqrt{\bar{\alpha}_t}}(\boldsymbol{x}_t - \sqrt{1-\bar{\alpha}_t}\boldsymbol{\epsilon}). \tag{36}$$

Substituting Eq. (36) in Eq. (35) gives:

$$\widetilde{\boldsymbol{\mu}}(\boldsymbol{x}_t, \boldsymbol{x}_0) = \frac{\sqrt{\alpha_t}(1-\bar{\alpha}_{t-1})}{1-\bar{\alpha}_t}\boldsymbol{x}_t$$
$$+ \frac{\sqrt{\bar{\alpha}_{t-1}}\beta_t}{1-\bar{\alpha}_t}\frac{1}{\sqrt{\bar{\alpha}_t}}(\boldsymbol{x}_t - \sqrt{1-\bar{\alpha}_t}\boldsymbol{\epsilon})$$
$$\overset{(a)}{=} \frac{1}{\sqrt{\alpha_t}}\Big(\boldsymbol{x}_t - \frac{1-\alpha_t}{\sqrt{1-\bar{\alpha}_t}}\boldsymbol{\epsilon}\Big) \overset{(22)}{=} \frac{1}{\sqrt{\alpha_t}}\Big(\boldsymbol{x}_t - \frac{\beta_t}{\sqrt{1-\bar{\alpha}_t}}\boldsymbol{\epsilon}\Big), \tag{37}$$

where (a) is because of simplification of the expression while noticing Eqs. (22) and (23). Eq. (37) is the mean of the tractable backward distribution $q(\boldsymbol{x}_{t-1}|\boldsymbol{x}_t, \boldsymbol{x}_0)$ as in Eq. (32).

Let $\boldsymbol{\mu}_\theta(\boldsymbol{x}_t, t)$ denote the mean of the distribution of the backward process, i.e., $p_\theta(\boldsymbol{x}_{t-1}|\boldsymbol{x}_t)$. This mean also has a similar format as the mean $\widetilde{\boldsymbol{\mu}}(\boldsymbol{x}_t, \boldsymbol{x}_0)$; therefore:

$$\boldsymbol{\mu}_\theta(\boldsymbol{x}_t, t) = \frac{1}{\sqrt{\alpha_t}}\Big(\boldsymbol{x}_t - \frac{\beta_t}{\sqrt{1-\bar{\alpha}_t}}\boldsymbol{\epsilon}_\theta(\boldsymbol{x}_t, t)\Big), \tag{38}$$

where $\boldsymbol{\epsilon}_\theta(\boldsymbol{x}_t, t)$ is a function approximator (i.e., neural network) intended to predict $\boldsymbol{\epsilon}$ from $\boldsymbol{x}_t$. In summary, the backward process $p_\theta(\boldsymbol{x}_{t-1}|\boldsymbol{x}_t)$ has a Gaussian distribution as in the Eq. (7), whose mean and covariance are Eqs. (38) and (8), respectively.

2.5. The Loss Function of the Diffusion Model

Recall the KL-divergence in the loss function of Eq. (21):

$$\mathrm{KL}\big(q(\boldsymbol{x}_{t-1}|\boldsymbol{x}_t, \boldsymbol{x}_0)\|p_\theta(\boldsymbol{x}_{t-1}|\boldsymbol{x}_t)\big),$$

where $q(\boldsymbol{x}) = \mathcal{N}(\boldsymbol{x}; \widetilde{\boldsymbol{\mu}}, \widetilde{\boldsymbol{\Sigma}})$ and $p(\boldsymbol{x}) = \mathcal{N}(\boldsymbol{x}; \boldsymbol{\mu}_\theta, \boldsymbol{\Sigma}_\theta)$. There is a closed form expression for the KL divergence of two Gaussian distributions:

$$\mathrm{KL}(q\|p) = \frac{1}{2}\Big(\mathbf{tr}(\boldsymbol{\Sigma}_\theta^{-1}\widetilde{\boldsymbol{\Sigma}}) + (\boldsymbol{\mu}_\theta - \widetilde{\boldsymbol{\mu}})^\top\boldsymbol{\Sigma}_\theta^{-1}(\boldsymbol{\mu}_\theta - \widetilde{\boldsymbol{\mu}}) - d + \ln\big(\frac{\mathbf{det}(\boldsymbol{\Sigma}_\theta)}{\mathbf{det}(\widetilde{\boldsymbol{\Sigma}})}\big)\Big), \tag{39}$$

where d is the dimensionality of $\boldsymbol{x}$, and $\mathbf{tr}(.)$ and $\mathbf{det}(.)$ denote trace and determinant of matrix, respectively. For simplicity, assume the covariances of the two distributions are diagonal and equal, i.e., $\boldsymbol{\Sigma}_\theta = \boldsymbol{\Sigma}_q = \lambda\boldsymbol{I}$ (see Eq. (8)). Then, Eq. (39) is simplified to:

$$\mathrm{KL}(q\|p) = \frac{1}{2}(\boldsymbol{\mu}_\theta - \widetilde{\boldsymbol{\mu}})^\top\boldsymbol{\Sigma}_\theta^{-1}(\boldsymbol{\mu}_\theta - \widetilde{\boldsymbol{\mu}}),$$

which is the squared Mahalanobis distance between the means of distributions. This makes sense because the two distributions p and q are normal distributions and their covariances – related to the second moment – are equal. So, for making them the same, their first moments, i.e., their means, should become the same.

As a result, the loss function can be written as the squared Mahalanobis distance between the means of distributions:

$$\ell_t := \mathbb{E}_{\boldsymbol{x}_0,\boldsymbol{\epsilon}}\Big[\frac{1}{2\|\boldsymbol{\Sigma}_\theta(\boldsymbol{x}_t, t)\|_2^2}\|\widetilde{\boldsymbol{\mu}}(\boldsymbol{x}_t, \boldsymbol{x}_0) - \boldsymbol{\mu}_\theta(\boldsymbol{x}_t, t)\|_2^2\Big]$$
$$\overset{(a)}{=} \mathbb{E}_{\boldsymbol{x}_0,\boldsymbol{\epsilon}}\Big[\frac{1}{2\|\boldsymbol{\Sigma}_\theta(\boldsymbol{x}_t, t)\|_2^2}\Big\|\frac{1}{\sqrt{\alpha_t}}\Big(\boldsymbol{x}_t - \frac{1-\alpha_t}{\sqrt{1-\bar{\alpha}_t}}\boldsymbol{\epsilon}\Big)$$
$$- \frac{1}{\sqrt{\alpha_t}}\Big(\boldsymbol{x}_t - \frac{1-\alpha_t}{\sqrt{1-\bar{\alpha}_t}}\boldsymbol{\epsilon}_\theta\Big)\Big\|_2^2\Big]$$
$$= \mathbb{E}_{\boldsymbol{x}_0,\boldsymbol{\epsilon}}\Big[\frac{(1-\alpha_t)^2}{2\alpha_t(1-\bar{\alpha}_t)\|\boldsymbol{\Sigma}_\theta(\boldsymbol{x}_t, t)\|_2^2}\|\boldsymbol{\epsilon} - \boldsymbol{\epsilon}_\theta(\boldsymbol{x}_t, t)\|_2^2\Big]$$
$$\overset{(30)}{=} \mathbb{E}_{\boldsymbol{x}_0,\boldsymbol{\epsilon}}\Big[\frac{(1-\alpha_t)^2}{2\alpha_t(1-\bar{\alpha}_t)\|\boldsymbol{\Sigma}_\theta(\boldsymbol{x}_t, t)\|_2^2}\times$$
$$\big\|\boldsymbol{\epsilon} - \boldsymbol{\epsilon}_\theta(\sqrt{\bar{\alpha}_t}\boldsymbol{x}_0 + \sqrt{1-\bar{\alpha}_t}\boldsymbol{\epsilon}, t)\big\|_2^2\Big], \tag{40}$$

where (a) is because of Eqs. (37) and (38) while noticing Eq. (22).

Ho *et al.* (Ho et al., 2020) discovered empirically that the diffusion model produces better-quality images when using a simplified loss function, omitting the weighting term:

$$\ell = \mathbb{E}_{t\sim[1,T],\boldsymbol{x}_0,\boldsymbol{\epsilon}}\Big[\big\|\boldsymbol{\epsilon} - \boldsymbol{\epsilon}_\theta(\sqrt{\bar{\alpha}_t}\boldsymbol{x}_0 + \sqrt{1-\bar{\alpha}_t}\boldsymbol{\epsilon}, t)\big\|_2^2\Big], \tag{41}$$

where $t \sim [1, T]$ means sampling t from the set $\{1, 2, \ldots, T\}$ uniformly. Recall that $\bar{\alpha}_t$ is calculated by Eq. (22) based on the selected noise schedule β_t in Eq. (3). Note that the expectation can be approximated by average according to the Monte Carlo approximation (Ghojogh et al., 2020).

while *not converged* **do**
  $\boldsymbol{x}_0 \sim q_0(\boldsymbol{x}_0)$
  $t \sim \text{Uniform}\{1, \dots, T\}$
  $\boldsymbol{\epsilon} \sim \mathcal{N}(\boldsymbol{0}, \boldsymbol{I})$
  Do backpropagation by the gradient of the loss function (41)

Algorithm 1: Training algorithm in DDPM

$\boldsymbol{x}_T \sim \mathcal{N}(\boldsymbol{0}, \boldsymbol{I})$
for t *from* T *to* 1 **do**
  $\boldsymbol{\epsilon} \sim \mathcal{N}(\boldsymbol{0}, \boldsymbol{I})$
  $\boldsymbol{x}_{t-1} = \frac{1}{\sqrt{\alpha_t}}\Big(\boldsymbol{x}_t - \frac{1-\alpha_t}{\sqrt{1-\bar{\alpha}_t}}\boldsymbol{\epsilon}_\theta(\boldsymbol{x}_t, t)\Big) + \sigma_t \boldsymbol{\epsilon}$
Return $\boldsymbol{x}_0$

Algorithm 2: Inference algorithm for generating data in DDPM

2.6. Training Phase of DDPM

The training algorithm of the DDPM is in Algorithm 1. In every iteration, an image $\boldsymbol{x}_0$ is sampled from the training dataset. Then, the time step t is sampled uniformly from $\{1, \dots, T\}$. Then, for the sampled time step t, the noise $\boldsymbol{\epsilon}$ is sampled from the standard Gaussian distribution. Finally, a backpropagation step is performed by the gradient of the loss function in Eq. (41). This makes the neural network learn the noise. In other words, it learns how much noise it should have in the backward process to reverse the forward process. This procedure is repeated until convergence of the weights θ of neural network. One of the advantages of DDPM compared to GAN (Generative Adversarial Network) (Goodfellow et al., 2014), is that this is a simple algorithm to train and it is stable. This is while training GAN is hard and tricky.

2.7. Inference (Sampling) Phase of DDPM

In the inference phase, the network has been trained, meaning that the distribution p_θ is learned. This distribution can be used to start with a complete noise and generate (or sample) data out of it in a backward process.

The inference phase starts with the complete noise, i.e., $\boldsymbol{x}_T \sim \mathcal{N}(\boldsymbol{0}, \boldsymbol{I})$. Let $\sigma_t \boldsymbol{I}$ denote the covariance of $\boldsymbol{x}_t$. By use of the reparameterization technique (Kingma & Welling, 2014) and according to Eq. (38) while noticing Eq. (22), we have:

$$\boldsymbol{x}_{t-1} = \frac{1}{\sqrt{\alpha_t}}\Big(\boldsymbol{x}_t - \frac{1-\alpha_t}{\sqrt{1-\bar{\alpha}_t}}\boldsymbol{\epsilon}_\theta(\boldsymbol{x}_t, t)\Big) + \sigma_t \boldsymbol{\epsilon}, \tag{42}$$

where $\boldsymbol{\epsilon} \sim \mathcal{N}(\boldsymbol{0}, \boldsymbol{I})$ and α_t and $\bar{\alpha}_t$ are calculated by Eqs. (22) and (23) based on the selected noise schedule β_t in Eq. (3). Note that $\boldsymbol{\epsilon}_\theta(\boldsymbol{x}_t, t)$ is outputted by the trained network. Iterating over t from T to 1 gives the generated data $\boldsymbol{x}_0$. The inference phase of DDPM is in Algorithm 2.

3. Modifications to the Original DDPM

3.1. Non-fixed Covariance Matrix

Recall Eq. (8) in which $\boldsymbol{\Sigma}_\theta$ is assumed to be fixed and diagonal, i.e., $\boldsymbol{\Sigma}_\theta(\boldsymbol{x}_t; t) = \sigma_t^2 \boldsymbol{I}$, in the original DDPM (Ho et al., 2020). We define:

$$\widetilde{\beta}_t := \frac{1-\bar{\alpha}_{t-1}}{1-\bar{\alpha}_t}\beta_t. \tag{43}$$

Nichol *et al.* (Nichol & Dhariwal, 2021) found out empirically that the covariance $\boldsymbol{\Sigma}_\theta(\boldsymbol{x}_t; t)$ used in Eq. (40) is better to be learned as a linear combination of β_t and $\widetilde{\beta}_t$ in the log domain:

$$\boldsymbol{\Sigma}_\theta(\boldsymbol{x}_t; t) = \textbf{diag}\Big(\exp\big(\boldsymbol{v}\log\beta_t + (\boldsymbol{1}-\boldsymbol{v})\log\widetilde{\beta}_t\big)\Big), \tag{44}$$

where **diag**$(.)$ makes a diagonal matrix with its input as the diagonal, exp is the exponential operator, $\boldsymbol{1} \in \mathbb{R}^d$ is the vector of ones, and $\boldsymbol{v} \in \mathbb{R}^d$ is a vector outputted by the network of DDPM which is learned by backpropagation. The loss function of the network is changed from Eq. (41) to:

$$\ell + \lambda \sum_{t=0}^{T} \ell_t, \tag{45}$$

where ℓ_t and ℓ are defined in Eqs. (40) and (41), respectively, and $\lambda = 0.001$ is the regularization hyperparameter. Note that $\boldsymbol{\Sigma}_\theta(\boldsymbol{x}_t; t)$, defined by Eq. (44), is used in ℓ_t so its $\boldsymbol{v}$ is learned by backpropagation.

3.2. Reducing the Gradient Noise

One might wonder why a regularized loss function, as in Eq. (45), is required while ℓ is the simplification of summation of ℓ_t's; meaning that the information of the first term is contained in the second term of Eq. (45). By use of gradient noise scale (McCandlish et al., 2018), Nichol *et al.* (Nichol & Dhariwal, 2021) empirically observed that the gradient noise of ℓ_t is much more than that of ℓ. They observed that this gradient noise is caused by the uniform sampling of t in Algorithm 1. By replacing the uniform sampling with importance sampling (Ghojogh et al., 2020), which records a history of previous values, they reduced this gradient noise. When using importance sampling, they could use merely $\sum_{t=0}^{T} \ell_t$ as the loss function.

3.3. Improving the Noise Schedule

Recall the noise schedule stated in Eq. (3). While linear noise schedule used in the original DDPM (Ho et al.,

2020) works well on high-resolution images, Nichol *et al.* (Nichol & Dhariwal, 2021) observed that it does not work properly on the small 32×32 or 64×64 images. This is because the linear noise schedule makes the $\boldsymbol{x}_T$ too noisy at the end of the forward process. They proposed a cosine noise schedule which works well also on the small images (Nichol & Dhariwal, 2021):

$$\bar{\alpha}_t = \frac{f(t)}{f(0)}, \tag{46}$$

$$f(t) := \cos\left(\frac{t/T + s}{1+s} \times \frac{\pi}{2}\right)^2, \tag{47}$$

where $s = 0.008$ is the offset. Note that according to Eqs. (22) and (23), we have:

$$\beta_t = 1 - \alpha_t = 1 - \frac{\bar{\alpha}_t}{\bar{\alpha}_{t-1}}. \tag{48}$$

Using Eq. (46) in Eq. (48) gives β_t at time t in the schedule.

3.4. Non-Standard Gaussian Noise Distribution

The original DDPM (Ho et al., 2020) considers standard Gaussian distribution for the noise $\boldsymbol{\epsilon}$. However, the corresponding data distribution may be more complicated than the standard Gaussian distribution making this noise distribution not necessarily suitable for the process. PriorGrad (Lee et al., 2022) takes care of this concern and modifies DDPM to use a data-dependent prior distribution. Suppose the mean $\boldsymbol{\mu}$ and the covariance $\boldsymbol{\Sigma}$ are obtained from some data-dependent prior; for example, they can be the mean and variance of the training data. In PriorGrad, the noise is sampled from the Gaussian distribution with covariance $\boldsymbol{\Sigma}$, i.e., $\boldsymbol{\epsilon} \sim \mathcal{N}(\mathbf{0}, \boldsymbol{\Sigma})$. It changes Eq. (30) to:

$$\boldsymbol{x}_t = \sqrt{\bar{\alpha}_t}(\boldsymbol{x}_0 - \boldsymbol{\mu}) + \sqrt{1-\bar{\alpha}_t}\boldsymbol{\epsilon}, \tag{49}$$

in which the mean is deducted from the data $\boldsymbol{x}_0$. This algorithm also changes the loss function from Eq. (41) to:

$$\ell = \mathbb{E}_{t\sim[1,T],\boldsymbol{x}_0,\boldsymbol{\epsilon}}\left[\left\|\boldsymbol{\epsilon} - \boldsymbol{\epsilon}_\theta(\boldsymbol{x}_t, t)\right\|^2_{\boldsymbol{\Sigma}^{-1}}\right], \tag{50}$$

where $\boldsymbol{x}_t$ is obtained from Eq. (49) and $\|\boldsymbol{x}\|^2_{\boldsymbol{\Sigma}^{-1}} := \boldsymbol{x}^\top \boldsymbol{\Sigma}^{-1} \boldsymbol{x}$. In the inference phase, it simply adds back the subtracted mean $\boldsymbol{\mu}$ to the generated data $\boldsymbol{x}_0$. The main ideas of PriorGrad (Lee et al., 2022) are two-fold: (1) subtracting the mean of data in the training and adding it back in sampling, and (2) using the inverse of covariance matrix in the loss function. Therefore, it can use a non-standard Gaussian distribution in the diffusion model.

3.5. Some Other Improvements

An additional improvements over DDPM is the Denoising Diffusion Implicit Model (DDIM) (Song et al., 2021a) which generalizes DDPM from Markov chains to non-Markovian chains to make inference (sampling) faster. Another improvement over DDPM is the Discrete Denoising Diffusion Probabilistic Model (D3PM) (Austin et al., 2021) which extends DDPM from generating continuous-state data to discrete-state data. In other words, D3PM works on categorical (discrete) data rather than the conventional float data as in DDPM.

$\boldsymbol{x}_T \sim \mathcal{N}(\mathbf{0}, \boldsymbol{I})$
for t *from* T *to* 1 **do**
  $\boldsymbol{\epsilon} \sim \mathcal{N}(\mathbf{0}, \boldsymbol{I})$
  $\boldsymbol{x}_{t-1} = \frac{1}{\sqrt{\alpha_t}}\left(\boldsymbol{x}_t - \frac{1-\alpha_t}{\sqrt{1-\bar{\alpha}_t}}\boldsymbol{\epsilon}_\theta(\boldsymbol{x}_t, t)\right) + \sigma_t \boldsymbol{\epsilon}$
  $\boldsymbol{y}_{t-1} = \sqrt{\bar{\alpha}_{t-1}}\boldsymbol{y} + \sqrt{1-\bar{\alpha}_{t-1}}\boldsymbol{\epsilon}$
  $\boldsymbol{x}_{t-1} \leftarrow \boldsymbol{x}_{t-1} - \phi_n(\boldsymbol{x}_{t-1}) + \phi_n(\boldsymbol{y}_{t-1})$
Return $\boldsymbol{x}_0$

Algorithm 3: Inference algorithm for generating data in ILVR (conditional diffusion model)

4. Conditional Diffusion Model

DDPM (Ho et al., 2020) is unconditional; meaning that it generates any data, from the data distribution, in the inference (sampling) phase. Iterative Latent Variable Refinement (ILVR) (Choi et al., 2021) is a conditional diffusion model which can generate data from a specific category or class of data chosen by the user. In this algorithm, the algorithm takes a reference data instance (e.g., a reference image) and generates a data instance similar to, or in the category of, the reference. The training phase of ILVR is the same as training in DDPM. In fact, this algorithm modifies the inference (sampling) phase of DDPM to become a conditional generator. The inference algorithm of ILVR is Algorithm 3.

Let $\phi_n(.)$ be a low-pass filter which downsamples data by a factor of n and then upsamples it again by the factor n. This only preserves the low frequency of data and omits its high frequency. For example, in images, it makes the image blurred. Obviously, if $\boldsymbol{x} \in \mathbb{R}^d$, then $\phi_n(\boldsymbol{x}) \in \mathbb{R}^d$. Suppose the unconditional sample $\boldsymbol{x}_{t-1} \in \mathbb{R}^d$ has been calculated by Eq. (42). Let the reference data instance be denoted by $\boldsymbol{y} \in \mathbb{R}^d$. Similar to Eq. (30), $\boldsymbol{y}_{t-1}$ can be obtained from $\boldsymbol{y}$ and the noise $\boldsymbol{\epsilon}$ as:

$$\boldsymbol{y}_{t-1} = \sqrt{\bar{\alpha}_{t-1}}\boldsymbol{y} + \sqrt{1-\bar{\alpha}_{t-1}}\boldsymbol{\epsilon}. \tag{51}$$

The unconditional sample $\boldsymbol{x}_{t-1}$ can then be modified to be similar to the noisy reference $\boldsymbol{y}_{t-1}$ at time slot $(t-1)$. For this, the low frequency of $\boldsymbol{x}_{t-1}$, i.e., $\phi_n(\boldsymbol{x}_{t-1}) \in \mathbb{R}^d$, is subtracted from it and, instead, the low frequency of $\boldsymbol{y}_{t-1}$ is added back to it. This replaces the low frequency of $\boldsymbol{x}_{t-1}$ with the low frequency of the noisy reference at time slot

$(t-1)$. Performing this iteratively from $t=T$ to $t=1$ gives the sample similar to the reference sample.

It is noteworthy that n is a positive integer selected by the user. The smaller the n, the more similar the sample will be to the reference. This is because a small n makes the bandwidth of the low pass filter larger, replacing most of the frequencies of the sample with the reference data.

5. Continuous Noise Schedule by Stochastic Differential Equation

The noise schedule in DDPM (Ho et al., 2020) is in the discrete domain; meaning that there are T discrete steps from the original data to the complete noise and backwards. However, it is possible to have noise schedule of diffusion models in a continuous domain; meaning that $t \in [0, T]$ can be continuous rather than being discrete $t \in \{1, 2, \dots, T\}$. The continuous noise schedule converts the Markov chain of diffusion model to a Stochastic Differential Equation (SDE) (Song et al., 2021b).

Consider Eq. (25):

$$\boldsymbol{x}_t = \sqrt{\alpha_t}\boldsymbol{x}_{t-1} + \sqrt{1-\alpha_t}\boldsymbol{\epsilon} \overset{(22)}{=} \sqrt{1-\beta_t}\boldsymbol{x}_{t-1} + \sqrt{\beta_t}\boldsymbol{\epsilon},$$

in DDPM in which $t \in \{1, 2, \dots, T\}$. If $T \to \infty$, the number of steps in the noise schedule becomes huge making the flow of time almost continuous. By $T \to \infty$, Eq. (25) converges to the following SDE (Song et al., 2021b):

$$d\boldsymbol{x} = -\frac{1}{2}\beta(t)\,\boldsymbol{x}\,dt + \sqrt{\beta(t)}\,d\boldsymbol{w}, \tag{52}$$

where $\boldsymbol{w} \in \mathbb{R}^d$ is the standard Wiener process (Brownian motion), $\boldsymbol{x} \in \mathbb{R}^d$ is the data variable, and $d\boldsymbol{x}$ and $d\boldsymbol{w}$ are the differentials of $\boldsymbol{x}$ and $\boldsymbol{w}$. As a result, DDPM (Ho et al., 2020) is a special case of diffusion model with SDE (Song et al., 2021b).

In a more general case, the forward process in the diffusion model with SDE is the following SDE (Song et al., 2021b):

$$d\boldsymbol{x} = f(\boldsymbol{x}, t)\,dt + g(t)\,d\boldsymbol{w}, \tag{53}$$

where $\boldsymbol{w} \in \mathbb{R}^d$ is the standard Wiener process (Brownian motion), $f(., t) : \mathbb{R}^d \to \mathbb{R}^d$ is a function named the drift coefficient of $\boldsymbol{x}(t)$, and $g(.) : \mathbb{R} \to \mathbb{R}$ is a function called the diffusion coefficient of $\boldsymbol{x}(t)$. The backward process of the diffusion model with SDE is the following reverse-time SDE (Song et al., 2021b):

$$d\boldsymbol{x} = \Big(f(\boldsymbol{x}, t) - g(t)^2\nabla_{\boldsymbol{x}} \log p_t(\boldsymbol{x})\Big)dt + g(t)\,d\bar{\boldsymbol{w}}, \tag{54}$$

where $\bar{\boldsymbol{w}}$ is the standard Wiener process when time flows backwards from T to 0 and dt is the infinitesimal negative time. The operator $\nabla_{\boldsymbol{x}}$ is the derivative with respect to $\boldsymbol{x}$, the $p_t(\boldsymbol{x})$ is the probability distribution of $\boldsymbol{x}$ at time t, and the $\nabla_{\boldsymbol{x}} \log p_t(\boldsymbol{x})$ is the time-dependant gradient field, called score briefly. As long as the score gets estimated, this reverse-time SDE can be solved and therefore the sample can be generated out of noise in the backward process of diffusion model.

According to (Song et al., 2021b), the loss function of the original DDPM, for maximizing the ELBO, can be stated in the form of a weighted sum of denoising score matching (Vincent, 2011):

$$\ell = \sum_{t=1}^{T}(1-\alpha_t)\mathbb{E}_{p_{\text{data}}(\boldsymbol{x})}\mathbb{E}_{q(\boldsymbol{x}_t|\boldsymbol{x}_0)}\Big[\big\|\boldsymbol{s}_\theta(\boldsymbol{x}_t, t) - \nabla_{\boldsymbol{x}_t}\log\big(q(\boldsymbol{x}_t|\boldsymbol{x}_0)\big)\big\|\Big], \tag{55}$$

where $\boldsymbol{s}_\theta(\boldsymbol{x}_t, t)$ is the output of the neural network with weights θ. Likewise, the loss function in the diffusion model with SDE is (Song et al., 2021b):

$$\ell = \mathbb{E}_t\Bigg[\lambda(t)\,\mathbb{E}_{\boldsymbol{x}(0)}\mathbb{E}_{\boldsymbol{x}(t)|\boldsymbol{x}(0)}\Big[\big\|\boldsymbol{s}_\theta(\boldsymbol{x}_t, t) - \nabla_{\boldsymbol{x}(t)}\log\Big(q\big(\boldsymbol{x}(t)|\boldsymbol{x}(0)\big)\Big)\big\|\Big]\Bigg], \tag{56}$$

where $\lambda : [0, T] \to \mathbb{R}_{>0}$ is a positive weighting function, t is sampled from the uniform distribution in range $[0, T]$, the $\boldsymbol{x}(0)$ is a sampled data instance from the training dataset, and $\boldsymbol{x}(t) \sim q\big(\boldsymbol{x}(t)|\boldsymbol{x}(0)\big)$. This loss function is minimized by backpropagation until the output of network, i.e., $\boldsymbol{s}_\theta(\boldsymbol{x}_t, t)$, becomes a good estimator of the score $\nabla_{\boldsymbol{x}(t)}\log\big(q\big(\boldsymbol{x}(t)|\boldsymbol{x}(0)\big)\big)$.

After training, the neural network becomes a function approximator for the score. Therefore, the reverse-time SDE, Eq. (54), can be solved by numerical SDE solvers to generate data in the backward process of diffusion model. Various numerical methods exist for solving SDEs, some of which are Euler-Maruyama and stochastic Runge-Kutta (Kloeden & Platen, 1992). Any SDE solver can be used to solve the reverse-time SDE of the diffusion model to generate data.

It is noteworthy that continuous noise schedule by SDE has been also extended for discrete-state (categorical) data (Campbell et al., 2022).

6. Conclusion

This was a tutorial paper about diffusion models – a family of generative models in machine learning. The original DDPM, its forward and backward processes, its variational lower bound, and parameters of distributions were covered. Some modifications and improvements to DDPm were also introduced. Then, conditional diffusion model for generating data similar to a reference data instance was explained.

Finally, the diffusion model using SDE was covered for having a continuous noise schedule.

Acknowledgement

Some of the materials in this tutorial paper have been covered by Prof. Ali Ghodsi's videos (Data Science Courses) and Tushar Kumar's videos (Explaining AI) on YouTube.

References

Austin, Jacob, Johnson, Daniel D, Ho, Jonathan, Tarlow, Daniel, and van den Berg, Rianne. Structured denoising diffusion models in discrete state-spaces. In *Advances in Neural Information Processing Systems*, 2021.

Campbell, Andrew, Benton, Joe, De Bortoli, Valentin, Rainforth, Tom, Deligiannidis, George, and Doucet, Arnaud. A continuous time framework for discrete denoising models. In *Advances in Neural Information Processing Systems*, 2022.

Choi, Jooyoung, Kim, Sungwon, Jeong, Yonghyun, Gwon, Youngjune, and Yoon, Sungroh. ILVR: Conditioning method for denoising diffusion probabilistic models. In *Proceedings of the IEEE/CVF International Conference on Computer Vision*, pp. 14367–14376, 2021.

Ghojogh, Benyamin, Karray, Fakhri, and Crowley, Mark. Hidden Markov model: Tutorial. *Engineering Archive*, 2019.

Ghojogh, Benyamin, Nekoei, Hadi, Ghojogh, Aydin, Karray, Fakhri, and Crowley, Mark. Sampling algorithms, from survey sampling to Monte Carlo methods: Tutorial and literature review. *arXiv preprint arXiv:2011.00901*, 2020.

Ghojogh, Benyamin, Ghodsi, Ali, Karray, Fakhri, and Crowley, Mark. Factor analysis, probabilistic principal component analysis, variational inference, and variational autoencoder: Tutorial and survey. *arXiv preprint arXiv:2101.00734*, 2021.

Ghojogh, Benyamin, Crowley, Mark, Karray, Fakhri, and Ghodsi, Ali. Variational autoencoders. In *Elements of Dimensionality Reduction and Manifold Learning*, pp. 563–576. Springer, 2022.

Ghojogh, Benyamin, Crowley, Mark, Karray, Fakhri, and Ghodsi, Ali. Factor analysis and probabilistic principal component analysis. In *Elements of Dimensionality Reduction and Manifold Learning*, pp. 355–374. Springer, 2023.

Goodfellow, Ian, Pouget-Abadie, Jean, Mirza, Mehdi, Xu, Bing, Warde-Farley, David, Ozair, Sherjil, Courville, Aaron, and Bengio, Yoshua. Generative adversarial nets. *Advances in neural information processing systems*, 27, 2014.

Ho, Jonathan, Jain, Ajay, and Abbeel, Pieter. Denoising diffusion probabilistic models. *Advances in neural information processing systems*, 33:6840–6851, 2020.

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

Kloeden, Peter E and Platen, Eckhard. *Stochastic differential equations*. Springer, 1992.

Koller, Daphne and Friedman, Nir. *Probabilistic graphical models: principles and techniques*. MIT press, 2009.

Kullback, Solomon and Leibler, Richard A. On information and sufficiency. *The annals of mathematical statistics*, 22(1):79–86, 1951.

Lee, Sang-gil, Kim, Heeseung, Shin, Chaehun, Tan, Xu, Liu, Chang, Meng, Qi, Qin, Tao, Chen, Wei, Yoon, Sungroh, and Liu, Tie-Yan. PriorGrad: Improving conditional denoising diffusion models with data-dependent adaptive prior. In *International Conference on Learning Representations*, 2022.

McCandlish, Sam, Kaplan, Jared, Amodei, Dario, and Team, OpenAI Dota. An empirical model of large-batch training. *arXiv preprint arXiv:1812.06162*, 2018.

Nichol, Alexander Quinn and Dhariwal, Prafulla. Improved denoising diffusion probabilistic models. In *International Conference on Machine Learning*, pp. 8162–8171. PMLR, 2021.

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

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

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