Generative Models

Sunmook Choi felixchoi@korea.ac.kr

August 24, 2023

1 Variational Auto Encoder

2 Generative Adversarial Networks

In this section, we delve into a generative algorithm, called Generative Adversarial Networks (GAN) [1], which was an innovative paradigm in deep learning. The proposed structure of GAN is its *adversarial nets* framework. It contains two networks: a generator and a discriminator. Two networks are simultaneously trained in order to fool the other. The generator model can be thought of as analogous to a team of counterfeiters, trying to produce fake currency and use it without detection, while the discriminative model is analogous to the police, trying to detect the counterfeit currency. Optimization on both models drives both models to improve their methods until the counterfeits are indistinguishable from the genuine articles.

2.1 Algorithm of GAN

Given a dataset, we want to generate new data samples which seem to be obtained from the original dataset. In other words, given the data distribution p_{data}, we want to learn the generator's distribution p_g over data $\boldsymbol{X}$ such that $p_{\text{data}} = p_g$.

To learn p_g over data $\boldsymbol{X}$, we define a prior p_z on input noise (or latent) variables $\boldsymbol{Z}$, then represent a mapping to a data space as $G(\boldsymbol{Z}; \boldsymbol{\theta}_g)$, where G, called the generator, is a differentiable function (usually represented by a deep neural network) with parameters $\boldsymbol{\theta}_g$. We also define another function D, called the discriminator, with parameters $\boldsymbol{\theta}_d$ such that $D(\boldsymbol{X}; \boldsymbol{\theta}_d)$ represents the probability that the data $\boldsymbol{X}$ is came from the data rather than p_g. We train D to maximize the probability of assigning the correct label to both training examples (real, label=1) and samples from G (fake, label=0). We simultaneously train G to minimize $\log(1 - D(G(\boldsymbol{Z})))$, which is a way to fool the discriminator D.

In other words, two networks are playing the following two-player minimax game with value function $V(G, D)$:

$$\min_G \max_D V(D, G) = \mathbb{E}_{\boldsymbol{X} \sim p_{\text{data}}} \big[\log D(\boldsymbol{X}) \big] + \mathbb{E}_{\boldsymbol{Z} \sim p_z} \big[\log(1 - D(G(\boldsymbol{Z}))) \big] \tag{2.1}$$

Remark 1. For a more comprehensive understanding of this objective, we recall the binary cross entropy loss for logistic regression. For binary dataset $\{\boldsymbol{x}^n, y^n\}_{n=1}^N$ where $y^n \in \{0, 1\}$ with a hypothesis $h_{\boldsymbol{\theta}}$, the binary cross entropy loss $BCE(\boldsymbol{\theta})$ is given by

$$BCE(\boldsymbol{\theta}) = \frac{1}{N} \sum_{n=1}^{N} \Big[- y^n \log \left(h_{\boldsymbol{\theta}}(\boldsymbol{x}^n) \right) - (1 - y^n) \log \left(1 - h_{\boldsymbol{\theta}}(\boldsymbol{x}^n) \right) \Big]. \tag{2.2}$$

Let's compare it with the equation (2.1). First, when we are optimizing the discriminator D, that is, considering the hypothesis $h_{\boldsymbol{\theta}}$ to be the discriminator D, a real data $\boldsymbol{X} \sim p_{\text{data}}$ should have label 1, while a generated data $G(\boldsymbol{Z}) \sim p_g$ should have label 0. Therefore, maximizing $V(D, G)$ in the equation (2.1) with respect to D is equivalent to minimizing the binary cross entropy loss with respect to D. Now, consider the case when we are optimizing the generator G. Note that the optimization is actually

$$\min_G V(D, G) = \mathbb{E}_{\boldsymbol{Z} \sim p_z} \big[\log(1 - D(G(\boldsymbol{Z}))) \big]. \tag{2.3}$$

In practice, the value $V(D, G)$ may not provide sufficient gradient for G early in learning. Instead, we train G to minimize $-\log D(G(\boldsymbol{Z}))$. Hence, minimizing $-\log D(G(\boldsymbol{Z}))$ with respect to G is equivalent to minimizing the binary cross entropy, where the input is the latent vector and the hypothesis is set to $D(G(\cdot))$. Therefore, if we are to optimize GAN in practice, we only need to define binary cross entropy loss for its objective function.

The algorithm about training GAN is now described as below. In the algorithm, the objective is modified as I discussed in the remark above.

2.2 Theoretical Results of GAN

The generator G implicitly defines a probability distribution p_g as the distribution of the samples $G(\boldsymbol{Z})$ obtained when $\boldsymbol{Z} \sim p_z$. The theoretical results in this section show that the minimax game indeed has a global optimum for $p_g = p_{\text{data}}$, and the algorithm obtains the desired result when it converges.

2.2.1 Global Optimality of $p_g = p_{\textbf{data}}$

Consider the optimal discriminator D for any given generator G.

Proposition 2. *For G fixed, the optimal discriminator D is*

$$D_G^*(\boldsymbol{x}) = \frac{p_{data}(\boldsymbol{x})}{p_{data}(\boldsymbol{x}) + p_g(\boldsymbol{x})}. \tag{2.4}$$

Algorithm 2.1 Minibatch stochastic gradient descent training of GAN. The number of steps to apply to the discriminator, k, is a hyperparameter.

```
1: for number of epochs do
2:     for $k$ steps do
3:         Sample minibatch of $m$ noise samples $\{\boldsymbol{z}^{(1)}, \dots, \boldsymbol{z}^{(m)}\}$ from noise prior $p_{\boldsymbol{z}}$.
4:         Sample minibatch of $m$ examples $\{\boldsymbol{x}^{(1)}, \dots, \boldsymbol{x}^{(m)}\}$ from data generating distribution $p_{\text{data}}$.
5:         Update the discriminator by gradient ascent method:
               $\nabla_{\boldsymbol{\theta}_d} \frac{1}{m} \sum_{i=1}^{m} \left[\log D\left(\boldsymbol{x}^{(i)}\right) + \log\left(1 - D\left(G\left(\boldsymbol{z}^{(i)}\right)\right)\right)\right].$
6:     end for
7:     Sample minibatch of $m$ noise samples $\{\boldsymbol{z}^{(1)}, \dots, \boldsymbol{z}^{(m)}\}$ from noise prior $p_{\boldsymbol{z}}$.
8:     Update the generator by gradient descent method:
           $\nabla_{\boldsymbol{\theta}_g} \frac{1}{m} \sum_{i=1}^{m} \left[-\log\left(D\left(G\left(\boldsymbol{z}^{(i)}\right)\right)\right)\right].$
9: end for
```

Proof. The training criterion for the discriminator D, given any generator G, is to maximize the quantity $V(G, D)$

$$\begin{aligned} V(D,G) &= \int_{\boldsymbol{x}} p_{\text{data}}(\boldsymbol{x}) \log(D(\boldsymbol{x}))\, d\boldsymbol{x} + \int_{\boldsymbol{z}} p_z(\boldsymbol{z}) \log(1 - D(G(\boldsymbol{z})))\, d\boldsymbol{z} \\ &= \int_{\boldsymbol{x}} p_{\text{data}}(\boldsymbol{x}) \log(D(\boldsymbol{x})) + p_g(\boldsymbol{x}) \log(1 - D(\boldsymbol{x}))\, d\boldsymbol{x}. \end{aligned} \tag{2.5}$$

This equation is nothing but the binary cross entropy. For any $(a, b) \in \mathbb{R}^2 \setminus \{(0, 0)\}$, the function $y \mapsto a\log(y) + b\log(1-y)$ achieves its maximum in $[0, 1]$ at $\frac{a}{a+b}$. □

With the optimal discriminator D_G^* for fixed G, the equation (2.5) can be reformulated as below:

$$\begin{aligned} C(G) &= \max_D V(G, D) \\ &= \mathbb{E}_{\boldsymbol{X}\sim p_{\text{data}}}[\log D_G^*(\boldsymbol{X})] + \mathbb{E}_{\boldsymbol{X}\sim p_g}[\log(1 - D_G^*(\boldsymbol{X}))] \\ &= \mathbb{E}_{\boldsymbol{X}\sim p_{\text{data}}}\left[\log \frac{p_{\text{data}}(\boldsymbol{X})}{p_{\text{data}}(\boldsymbol{X}) + p_g(\boldsymbol{X})}\right] + \mathbb{E}_{\boldsymbol{X}\sim p_g}\left[\log \frac{p_g(\boldsymbol{X})}{p_{\text{data}}(\boldsymbol{X}) + p_g(\boldsymbol{X})}\right] \end{aligned} \tag{2.6}$$

Now we consider the global minimum of $C(G)$.

Theorem 3. *The global minimum of the virtual training criterion $C(G)$ is achieved if and only if $p_g = p_{data}$. At that point, $C(G)$ achieves the value $-\log 4$.*

Proof. For $p_g = p_{\text{data}}$, $D_G^*(\boldsymbol{X}) = \frac{1}{2}$ by Proposition 2. Hence, by inspecting the equation (2.6) at $D_G^*(\boldsymbol{X}) = \frac{1}{2}$, we find $C(G) = -\log 4$.

Recall the definitions of Kullback-Leibler divergence D_{KL} and Jensen-Shannon divergence D_{JS}:

$$\begin{aligned} D_{KL}(p \,\|\, q) &= \mathbb{E}_{x\sim p(x)}\left[-\log\frac{q(x)}{p(x)}\right] = \mathbb{E}_{x\sim p(x)}\left[\log\frac{p(x)}{q(x)}\right] \\ D_{JS}(p,q) &= \frac{1}{2}D_{KL}\left(p \,\|\, \frac{p+q}{2}\right) + \frac{1}{2}D_{KL}\left(q \,\|\, \frac{p+q}{2}\right). \end{aligned}$$

Then, the value $C(G) = V(D_G^*, G)$ can be computed as below:

$$\begin{aligned} C(G) &= \mathbb{E}_{\boldsymbol{X}\sim p_{\text{data}}}\left[\log\frac{p_{\text{data}}(\boldsymbol{X})}{p_{\text{data}}(\boldsymbol{X}) + p_g(\boldsymbol{X})}\right] + \mathbb{E}_{\boldsymbol{X}\sim p_g}\left[\log\frac{p_g(\boldsymbol{X})}{p_{\text{data}}(\boldsymbol{X}) + p_g(\boldsymbol{X})}\right] \\ &= D_{KL}\left(p_{\text{data}} \,\middle\|\, \frac{p_{\text{data}} + p_g}{2}\right) + D_{KL}\left(p_g \,\middle\|\, \frac{p_{\text{data}} + p_g}{2}\right) - 2\log 2 \\ &= D_{JS}\left(p_{\text{data}},\, p_g\right) - \log 4 \end{aligned}$$

Since the Jensen-Shannon divergence is a metric, we have shown that the global minimum of $C(G)$ is $-\log 4$ and that the only solution is $p_g = p_{\text{data}}$, that is, the generative model perfectly replicates the data generating process. $\square$

2.2.2 Convergence of Algorithm 2.1

Proposition 4. *If G and D have enough capacity, and at each step of Algorithm 2.1, the discriminator is allowed to reach its optimum given G, and p_g is updated so as to improve the criterion*

$$\mathbb{E}_{\boldsymbol{X}\sim p_{data}}\left[\log\frac{p_{data}(\boldsymbol{X})}{p_{data}(\boldsymbol{X}) + p_g(\boldsymbol{X})}\right],$$

then p_g converges to p_{data}.

3 InfoGAN

4 Diffusion Models

This section is my work on understanding the papers [2, 3]. The essential idea of diffusion models is inspired by non-equilibrium statistical physics. It is to systematically and slowly destroy structure (or add noise) in a data distribution through an iterative forward diffusion process. Then we learn a reverse diffusion process that restores structure in data, yielding a highly flexible and tractable generative model of the data. Recall that we say a model is tractable if it can be analytically evaluated and easily fit to data, and we say a model is flexible if it can be molded to fit structure in arbitrary data.

4.1 Algorithm

Our goal can be explained in two steps. The first step contains defining a forward diffusion process, which is a Markov chain that converts complex data distribution into a simple and tractable distribution by repeatedly applying Markov diffusion kernel. The second step contains learning a finite-time reversal of the diffusion process which defines our generative model distributions by denoising.

4.1.1 Diffusion Process

Let q be the distribution over data $\boldsymbol{X}_0$.

4.1.2 Reverse Process

In the diffusion process, data is corrupted step by step by the diffusion kernel $q(\boldsymbol{X}_t \mid \boldsymbol{X}_{t-1})$. The true denoising distribution will be $q(\boldsymbol{X}_{t-1} \mid \boldsymbol{X}_t)$ and it can be computed using Bayes' rule:

$$\begin{aligned} &q(\boldsymbol{X}_{t-1} \mid \boldsymbol{X}_t) \propto q(\boldsymbol{X}_{t-1})\, q(\boldsymbol{X}_t \mid \boldsymbol{X}_{t-1}) \\ &\text{where} \quad q(\boldsymbol{x}_t) = \int q(\boldsymbol{x}_t, \boldsymbol{x}_0)\, d\boldsymbol{x}_0 = \int q(\boldsymbol{x}_0)\, q(\boldsymbol{x}_t \mid \boldsymbol{x}_0)\, d\boldsymbol{x}_0. \end{aligned} \tag{4.1}$$

The integration is intractable since we do not have an access to the data distribution q. Hence, we should find a way to approximate the distribution. We first state a known fact below.

Theorem 5 (Feller, 1949)**.** *For both Gaussian and binomial diffusion, for infinitesimal step size β, the reversal of the diffusion process has the identical functional form as the forward process.*

By the theorem, we can see the functional form of the true distribution $q(\boldsymbol{X}_{t-1} \mid \boldsymbol{X}_t)$. That is, if the variance schedule β_t is small enough, $q(\boldsymbol{X}_{t-1} \mid \boldsymbol{X}_t)$ will also be a Gaussian (or binomial) distribution because the Markov diffusion kernel $q(\boldsymbol{X}_t \mid \boldsymbol{X}_{t-1}; \beta_t)$ is a Gaussian (or binomial) distribution. Therefore, during the training session, only the mean and covariance for a Gaussian diffusion kernel (or the bit flip probability for a binomial kernel) needed be estimated. We denote the approximated likelihood by $p_\theta(\boldsymbol{X}_{t-1} \mid \boldsymbol{X}_t)$ with learnable parameters θ.

Proposition 6. *The reversal of diffusion process also has the Markov property.*

Proof. Recall that the diffusion process has the Markov property. Then, for any $i = 1, \ldots, T-1$,

$$\begin{aligned} q(\boldsymbol{X}_{i:T}) &= q(\boldsymbol{X}_i)\, q(\boldsymbol{X}_{i+1} \mid \boldsymbol{X}_i)\, q(\boldsymbol{X}_{i+2} \mid \boldsymbol{X}_i, \boldsymbol{X}_{i+1}) \cdots q(\boldsymbol{X}_T \mid \boldsymbol{X}_i, \boldsymbol{X}_{i+1}, \ldots, \boldsymbol{X}_{T-1}) \\ &= q(\boldsymbol{X}_i)\, q(\boldsymbol{X}_{i+1} \mid \boldsymbol{x}_i)\, q(\boldsymbol{X}_{i+2} \mid \boldsymbol{X}_{i+1}) \cdots q(\boldsymbol{X}_T \mid \boldsymbol{X}_{T-1}) \\ &= q(\boldsymbol{X}_i) \prod_{j=i+1}^{T} q(\boldsymbol{X}_j \mid \boldsymbol{X}_{j-1}). \end{aligned}$$

Hence, we obtain the following:

$$q(\boldsymbol{X}_{t-1} \mid \boldsymbol{X}_{t:T}) = \frac{q(\boldsymbol{X}_{t-1:T})}{q(\boldsymbol{X}_{t:T})} = \frac{q(\boldsymbol{X}_{t-1}) \prod_{i=t}^{T} q(\boldsymbol{X}_i \mid \boldsymbol{X}_{i-1})}{q(\boldsymbol{X}_t) \prod_{i=t+1}^{T} q(\boldsymbol{X}_i \mid \boldsymbol{X}_{i-1})} = \frac{q(\boldsymbol{X}_{t-1}) q(\boldsymbol{X}_t \mid \boldsymbol{X}_{t-1})}{q(\boldsymbol{X}_t)} = q(\boldsymbol{X}_{t-1} \mid \boldsymbol{X}_t)$$

This shows that the reversal of diffusion process has the Markov property. □

Now, we define the full reverse of the diffusion forward trajectory. Using the Markov property proven in Proposition 6, the reverse Gaussian diffusion trajectory is defined as

$$p(\boldsymbol{X}_T) = \pi(\boldsymbol{X}_T) \quad \text{and} \quad p_\theta(\boldsymbol{X}_{0:T}) = p(\boldsymbol{X}_T) \prod_{t=1}^{T} p_\theta(\boldsymbol{X}_{t-1} \,|\, \boldsymbol{X}_t) \tag{4.2}$$

where $p(\boldsymbol{X}_T) = \mathcal{N}(\boldsymbol{X}_T \mid \boldsymbol{0}, \boldsymbol{I})$ and $p_\theta(\boldsymbol{X}_{t-1} \mid \boldsymbol{X}_t) := \mathcal{N}\Big(\boldsymbol{X}_{t-1} \mid \boldsymbol{\mu}_\theta(\boldsymbol{X}_t, t), \boldsymbol{\Sigma}_\theta(\boldsymbol{X}_t, t)\Big)$.

Remark 7 (Denoising Diffusion Probabilistic Model (DDPM))**.** The paper [3] constructs the reverse transition for the Gaussian diffusion kernel $p_\theta(\boldsymbol{X}_{t-1} \,|\, \boldsymbol{X}_t)$ as below. For $\alpha_t = 1 - \beta_t$ and $\bar{\alpha}_t = \prod_{s=1}^{t} \alpha_s$, The covariance parameter is set to be an isotropic covariance $\boldsymbol{\Sigma}_\theta(\boldsymbol{x}_t, t) = \sigma_t^2 \boldsymbol{I}$ to be untrained time dependent constants. The paper explains that both $\sigma_t^2 = \beta_t$ and $\sigma_t^2 = \frac{1-\bar{\alpha}_{t-1}}{1-\bar{\alpha}_t}\beta_t$ had similar results. The mean parameter is set to be

$$\boldsymbol{\mu}_\theta(\boldsymbol{x}_t, t) = \frac{1}{\sqrt{\alpha_t}} \left(\boldsymbol{x}_t - \frac{\beta_t}{\sqrt{1-\bar{\alpha}_t}} \boldsymbol{\epsilon}_\theta(\boldsymbol{x}_t, t) \right) \tag{4.3}$$

where $\boldsymbol{\epsilon}_\theta$ is a function approximator intended to predict $\boldsymbol{\epsilon}$ from $\boldsymbol{x}_t$. To see why the specific parameterization of $\boldsymbol{\mu}_\theta$ is chosen, we should take a look at the analysis of the loss function. This will be described in the following section.

4.1.3 Model probability

The probability $p_\theta(\boldsymbol{x}_0)$ that the generative model assigns to the data is

$$p_\theta(\boldsymbol{x}_0) = \int p_\theta(\boldsymbol{x}_{0:T})\, d\boldsymbol{x}_{1:T}. \tag{4.4}$$

It is intractable since it marginalizes the joint probability over all possible trajectories. Instead, by a simple modification, we obtain the following:

$$\begin{aligned}
p_\theta(\boldsymbol{x}_0) &= \int p_\theta(\boldsymbol{x}_{0:T})\, d\boldsymbol{x}_{1:T} && (4.5)\\
&= \int q(\boldsymbol{x}_{1:T} \mid \boldsymbol{x}_0)\, \frac{p_\theta(\boldsymbol{x}_{0:T})}{q(\boldsymbol{x}_{1:T} \mid \boldsymbol{x}_0)} d\boldsymbol{x}_{1:T} && (4.6)\\
&= \int q(\boldsymbol{x}_{1:T} \mid \boldsymbol{x}_0)\, p(\boldsymbol{x}_T) \prod_{t=1}^{T} \frac{p_\theta(\boldsymbol{x}_{t-1} \mid \boldsymbol{x}_t)}{q(\boldsymbol{x}_t \mid \boldsymbol{x}_{t-1})} d\boldsymbol{x}_{1:T} && (4.7)\\
&= \mathbb{E}_{\boldsymbol{x}_{1:T} \sim q(\,\cdot\,\mid \boldsymbol{x}_0)} \left[p(\boldsymbol{x}_T) \prod_{t=1}^{T} \frac{p_\theta(\boldsymbol{x}_{t-1} \mid \boldsymbol{x}_t)}{q(\boldsymbol{x}_t \mid \boldsymbol{x}_{t-1})} \,\middle|\, \boldsymbol{x}_0 \right] && (4.8)
\end{aligned}$$

The probability $p_\theta(\boldsymbol{x}_0)$ can be estimated rapidly by averaging over samples from the forward trajectory $q(\boldsymbol{x}_{1:T} \mid \boldsymbol{x}_0)$ (Monte Carlo estimation).

4.1.4 Training

Training amounts to maximizing the model log likelihood $\ell(\theta) = \int q(\boldsymbol{x}_0) \log p_\theta(\boldsymbol{x}_0)\, d\boldsymbol{x}_0$, which has a lower bound provided by Jensen's inequality.

$$\begin{aligned}
\ell(\theta) &= \int q(\boldsymbol{x}_0) \log p_\theta(\boldsymbol{x}_0)\, d\boldsymbol{x}_0 = \mathbb{E}_{\boldsymbol{x}_0 \sim q(\cdot)} \big[\log p_\theta(\boldsymbol{x}_0) \big]\\
&= \mathbb{E}_{\boldsymbol{x}_0 \sim q(\cdot)} \left[\log \left(\mathbb{E}_{\boldsymbol{x}_{1:T} \sim q(\,\cdot\,\mid \boldsymbol{x}_0)} \left[p(\boldsymbol{x}_T) \prod_{t=1}^{T} \frac{p_\theta(\boldsymbol{x}_{t-1} \mid \boldsymbol{x}_t)}{q(\boldsymbol{x}_t \mid \boldsymbol{x}_{t-1})} \,\middle|\, \boldsymbol{x}_0 \right] \right) \right]\\
&\geq \mathbb{E}_{\boldsymbol{x}_0 \sim q(\cdot)} \left[\mathbb{E}_{\boldsymbol{x}_{1:T} \sim q(\,\cdot\,\mid \boldsymbol{x}_0)} \left[\log \left(p(\boldsymbol{x}_T) \prod_{t=1}^{T} \frac{p_\theta(\boldsymbol{x}_{t-1} \mid \boldsymbol{x}_t)}{q(\boldsymbol{x}_t \mid \boldsymbol{x}_{t-1})} \right) \,\middle|\, \boldsymbol{x}_0 \right] \right] \qquad (\because \text{ Jensen's inequality})\\
&= \mathbb{E}_{\boldsymbol{x}_{0:T} \sim q} \left[\log p(\boldsymbol{x}_T) + \sum_{t=1}^{T} \log \frac{p_\theta(\boldsymbol{x}_{t-1} \mid \boldsymbol{x}_t)}{q(\boldsymbol{x}_t \mid \boldsymbol{x}_{t-1})} \right]
\end{aligned}$$

Recall that Jensen's inequality is that, for any concave function g, we have $\mathbb{E}[g(X)] \leq g(\mathbb{E}[X])$. Our goal is to maximize the lower bound of $\ell(\theta)$, or equivalently, to minimize the upper bound L of the negative log likelihood $\mathbb{E}[-\log p_\theta(\boldsymbol{x}_0)]$:

$$L := \mathbb{E}_{\boldsymbol{x}_{0:T} \sim q} \left[-\log p(\boldsymbol{x}_T) - \sum_{t=1}^{T} \log \frac{p_\theta(\boldsymbol{x}_{t-1} \mid \boldsymbol{x}_t)}{q(\boldsymbol{x}_t \mid \boldsymbol{x}_{t-1})} \right] \tag{4.9}$$

Hence, efficient training is possible by optimizing random terms of L with gradient descent algorithms. However, it has a problem of high variance due to Monte Carlo estimates, because each long sampled trajectory $\boldsymbol{x}_{0:T}$ would be erratic. Further improvements come from variance reduction by rewriting the upper bound as in the following theorem.

Theorem 8. *The objective L in Eq. 4.9 is equal to the following:*

$$L = \mathbb{E}_{\boldsymbol{x}_{0:T} \sim q} \left[D_{KL}\big(q(\boldsymbol{x}_T \mid \boldsymbol{x}_0) \,\|\, p(\boldsymbol{x}_T)\big) + \sum_{t=2}^{T} D_{KL}\Big(q(\boldsymbol{x}_{t-1} \mid \boldsymbol{x}_t, \boldsymbol{x}_0) \,\|\, p_\theta(\boldsymbol{x}_{t-1} \mid \boldsymbol{x}_t)\Big) - \log p_\theta(\boldsymbol{x}_0 \mid \boldsymbol{x}_1) \right] \tag{4.10}$$

Proof. The equality can be shown as follows:

$$L = \mathbb{E}_{\boldsymbol{x}_{0:T}\sim q}\left[-\log p(\boldsymbol{x}_T) - \sum_{t=1}^{T}\log\frac{p_\theta(\boldsymbol{x}_{t-1}\mid\boldsymbol{x}_t)}{q(\boldsymbol{x}_t\mid\boldsymbol{x}_{t-1})}\right] \tag{4.11}$$

$$= \mathbb{E}_{\boldsymbol{x}_{0:T}\sim q}\left[-\log p(\boldsymbol{x}_T) - \sum_{t=2}^{T}\log\frac{p_\theta(\boldsymbol{x}_{t-1}\mid\boldsymbol{x}_t)}{q(\boldsymbol{x}_t\mid\boldsymbol{x}_{t-1})} - \log\frac{p_\theta(\boldsymbol{x}_0\mid\boldsymbol{x}_1)}{q(\boldsymbol{x}_1\mid\boldsymbol{x}_0)}\right] \tag{4.12}$$

$$= \mathbb{E}_{\boldsymbol{x}_{0:T}\sim q}\left[-\log p(\boldsymbol{x}_T) - \sum_{t=2}^{T}\log\frac{p_\theta(\boldsymbol{x}_{t-1}\mid\boldsymbol{x}_t)}{q(\boldsymbol{x}_{t-1}\mid\boldsymbol{x}_t,\boldsymbol{x}_0)}\cdot\frac{q(\boldsymbol{x}_{t-1}\mid\boldsymbol{x}_0)}{q(\boldsymbol{x}_t\mid\boldsymbol{x}_0)} - \log\frac{p_\theta(\boldsymbol{x}_0\mid\boldsymbol{x}_1)}{q(\boldsymbol{x}_1\mid\boldsymbol{x}_0)}\right] \tag{4.13}$$

$$= \mathbb{E}_{\boldsymbol{x}_{0:T}\sim q}\left[-\log\frac{p(\boldsymbol{x}_T)}{q(\boldsymbol{x}_T\mid\boldsymbol{x}_0)} - \sum_{t=2}^{T}\log\frac{p_\theta(\boldsymbol{x}_{t-1}\mid\boldsymbol{x}_t)}{q(\boldsymbol{x}_{t-1}\mid\boldsymbol{x}_t,\boldsymbol{x}_0)} - \log p_\theta(\boldsymbol{x}_0\mid\boldsymbol{x}_1)\right] \tag{4.14}$$

$$= \mathbb{E}_{\boldsymbol{x}_0\sim q}\left[\mathbb{E}_{\boldsymbol{x}_{1:T}|\boldsymbol{x}_0}\left[-\log\frac{p(\boldsymbol{x}_T)}{q(\boldsymbol{x}_T\mid\boldsymbol{x}_0)}\middle|\boldsymbol{x}_0\right] - \sum_{t=2}^{T}\mathbb{E}_{\boldsymbol{x}_{1:T}|\boldsymbol{x}_0}\left[\log\frac{p_\theta(\boldsymbol{x}_{t-1}\mid\boldsymbol{x}_t)}{q(\boldsymbol{x}_{t-1}\mid\boldsymbol{x}_t,\boldsymbol{x}_0)}\middle|\boldsymbol{x}_0\right] - \mathbb{E}_{\boldsymbol{x}_{1:T}|\boldsymbol{x}_0}\big[\log p_\theta(\boldsymbol{x}_0\mid\boldsymbol{x}_1)\mid\boldsymbol{x}_0\big]\right] \tag{4.15}$$

$$= \mathbb{E}_{\boldsymbol{x}_0\sim q}\left[\mathbb{E}_{\boldsymbol{x}_T|\boldsymbol{x}_0}\left[-\log\frac{p(\boldsymbol{x}_T)}{q(\boldsymbol{x}_T\mid\boldsymbol{x}_0)}\middle|\boldsymbol{x}_0\right] - \sum_{t=2}^{T}\mathbb{E}_{\boldsymbol{x}_{t-1:t}|\boldsymbol{x}_0}\left[\log\frac{p_\theta(\boldsymbol{x}_{t-1}\mid\boldsymbol{x}_t)}{q(\boldsymbol{x}_{t-1}\mid\boldsymbol{x}_t,\boldsymbol{x}_0)}\middle|\boldsymbol{x}_0\right] - \mathbb{E}_{\boldsymbol{x}_1|\boldsymbol{x}_0}\big[\log p_\theta(\boldsymbol{x}_0\mid\boldsymbol{x}_1)\mid\boldsymbol{x}_0\big]\right] \tag{4.16}$$

$$= \mathbb{E}_{\boldsymbol{x}_0\sim q}\Bigg[\mathbb{E}_{\boldsymbol{x}_T|\boldsymbol{x}_0}\left[-\log\frac{p(\boldsymbol{x}_T)}{q(\boldsymbol{x}_T\mid\boldsymbol{x}_0)}\middle|\boldsymbol{x}_0\right] - \sum_{t=2}^{T}\mathbb{E}_{\boldsymbol{x}_t|\boldsymbol{x}_0}\left[\mathbb{E}_{\boldsymbol{x}_{t-1}|\boldsymbol{x}_t,\boldsymbol{x}_0}\left[\log\frac{p_\theta(\boldsymbol{x}_{t-1}\mid\boldsymbol{x}_t)}{q(\boldsymbol{x}_{t-1}\mid\boldsymbol{x}_t,\boldsymbol{x}_0)}\middle|\boldsymbol{x}_t,\boldsymbol{x}_0\right]\middle|\boldsymbol{x}_0\right] - \mathbb{E}_{\boldsymbol{x}_1|\boldsymbol{x}_0}\big[\log p_\theta(\boldsymbol{x}_0 \tag{4.17}$$

$$= \mathbb{E}_{\boldsymbol{x}_0\sim q}\Bigg[D_{KL}\big(q(\boldsymbol{x}_T\mid\boldsymbol{x}_0)\|p(\boldsymbol{x}_T)\big) + \sum_{t=2}^{T}\mathbb{E}_{\boldsymbol{x}_t|\boldsymbol{x}_0}\Big[D_{KL}\big(q(\boldsymbol{x}_{t-1}\mid\boldsymbol{x}_t,\boldsymbol{x}_0)\|p_\theta(\boldsymbol{x}_{t-1}\mid\boldsymbol{x}_t)\big)\Big|\boldsymbol{x}_0\Big] + \mathbb{E}_{\boldsymbol{x}_1|\boldsymbol{x}_0}\big[-\log p_\theta(\boldsymbol{x}_0\mid\boldsymbol{x}_1)\mid \tag{4.18}$$

$$= \mathbb{E}_{\boldsymbol{x}_{0:T}\sim q}\left[D_{KL}\big(q(\boldsymbol{x}_T\mid\boldsymbol{x}_0)\|p(\boldsymbol{x}_T)\big) + \sum_{t=2}^{T}D_{KL}\big(q(\boldsymbol{x}_{t-1}\mid\boldsymbol{x}_t,\boldsymbol{x}_0)\,\|\,p_\theta(\boldsymbol{x}_{t-1}\mid\boldsymbol{x}_t)\big) - \log p_\theta(\boldsymbol{x}_0\mid\boldsymbol{x}_1)\right] \tag{4.19}$$

□

Forward process and L_T

Reverse process and $L_{1:T-1}$

Data scaling, reverse process decoder, and L_0

Simplified Objective

4.1.5 Summary of Algorithms

In summary, two algorithms are described. Algorithm 4.1.5 describes the training phase of DDPM, and Algorithm 4.1.5 describes how to generate real-like samples after training.

Algorithm 4.1 Training

```
1: repeat
2:     $\boldsymbol{x}_0 \sim q$
3:     $t \sim Unif(\{1, \ldots, T\})$
4:     $\boldsymbol{\epsilon} \sim \mathcal{N}(\mathbf{0}, \boldsymbol{I})$
5:     Take a gradient descent step on $\nabla_\theta \|\boldsymbol{\epsilon} - \boldsymbol{\epsilon}_\theta(\sqrt{\bar{\alpha}_t}\boldsymbol{x}_0, \sqrt{1-\bar{\alpha}_t}\boldsymbol{\epsilon}, t)\|^2$
6: until converged
```

Algorithm 4.2 Sampling

```
1: $\boldsymbol{x}_T \sim \mathcal{N}(\mathbf{0}, \boldsymbol{I})$
2: for t=T,...,1 do
3:     $\boldsymbol{z} \sim \mathcal{N}(\mathbf{0}, \boldsymbol{I})$ if $t > 1$, else $\boldsymbol{z} = \mathbf{0}$
4:     $\boldsymbol{x}_{t-1} = \frac{1}{\sqrt{\alpha_t}}\left(\boldsymbol{x}_t - \frac{1-\alpha_t}{\sqrt{1-\bar{\alpha}_t}}\boldsymbol{\epsilon}_\theta(\boldsymbol{x}_t, t)\right) + \sigma_t \boldsymbol{z}$
5: end for
6: return $\boldsymbol{x}_0$
```

References

[1] Ian J. Goodfellow et al., *Generative Adversarial Nets*, NeurIPS 2014.

[2] J. Sohl-Dickstein et al., *Deep Unsupervised Learning using Nonequilibrium Thermodynamics*, ICML 2015

[3] J. Ho, et al., *Denoising Diffusion Probabilistic Models*, NeurIPS 2020