Research on Denoising Diffusion Probabilistic Models

Yiyang Jiang *

University of Chinese Academy of Sciences, Beijing, 100000, China

* Corresponding author: jiangyiyang19@ucas.ac.cn

Abstract. Diffusion models represent the latest state-of-the-art in the domain of deep generative models, boasting remarkable performance across a broad spectrum of applications. Despite the widespread success of diffusion models in various tasks, the original formulations of these models exhibit notable limitations. The article uses DDPM as an example, thoroughly and deeply exploring and deriving the mathematical principles of the model from two different perspectives. Additionally, this article explores the relationship between diffusion models and five other types of generative models: Generative Adversarial Networks (GANs), Variational Autoencoders (VAEs), Autoregressive models, Normalizing flows, and Energy-based models. Concluding with open questions for future research, the paper offers insights into the prospective algorithmic and application-oriented developments of diffusion models. Diffusion models have become a powerful framework capable of competing with Generative Adversarial Networks (GANs) in most applications without resorting to adversarial training. For specific tasks, understanding why and when diffusion models are more effective than other networks, and comprehending the differences between diffusion models and other generative models, will help clarify why diffusion models can produce high-quality samples with high likelihood.

Keywords: Machine Learning; generative models; stochastic differential equation.

1. Introduction

Diffusion models are the new state-of-the-art (SOTA) within the realm of deep generative models. These models have surpassed the previous SOTA, Generative Adversarial Networks (GANs), in image generation tasks and have demonstrated outstanding performance in a wide range of applications [1]. Furthermore, diffusion models maintain a close relationship with various research fields, such as robust learning, representation learning, and reinforcement learning [2]. Given the rapid progress in this area, a detailed and systematic review is essential. This paper primarily references the comprehensive review article by Song et al., which was the first to provide an extensive overview of diffusion models [4].

This article primarily focuses on the foundational theory of Denoising Diffusion Probabilistic Model (DDPM), providing a detailed derivation of the mathematical principles in the original text [5]. Additionally, it offers a derivation of DDPM principles from another perspective, that of Score-Based Generative Models. Comparing these two perspectives enhances the understanding of diffusion models and many details in their training process [6].

Despite the impressive performance of diffusion models across various tasks, the original models have their shortcomings. Their sampling speed is slow, often requiring thousands of evaluation steps to generate a single sample; their maximum likelihood estimation does not compare favorably with likelihood-based models; and their ability to generalize across different data types is limited. Many efforts have been made to address these limitations from practical applications or to analyze the capabilities of the models from a theoretical perspective, resulting in significant improvements to diffusion models.

2. Diffusion Models

Diffusion models did not receive much attention when they were first introduced, as they were not as straightforward and easy to understand as GANs. However, in recent years, they have made a

sudden rise in the field of generative models [7]. The two most advanced text-to-image generation systems currently, OpenAI's DALL-E 2 and Google's Imagen are both based on diffusion models.

The current popularity of generative diffusion models began with the Denoising Diffusion Probabilistic Model (DDPM) proposed in 2020 [8]. The groundbreaking paper on DDPM, published only in 2020, demonstrated the capabilities of diffusion models to the world. It outperformed GANs in image synthesis, prompting a shift in focus towards DDPM research within the image generation field.

2.1. Diffusion Models Explanation

Since they are called generative models, this means that Diffusion Models are used to generate data similar to the training data. Fundamentally, the working principle of Diffusion Models involves continuously adding Gaussian noise to corrupt the training data, and then learning to recover the data by reversing this noise process. After training, random noise samples can be input into the Diffusion Models, and data can be generated through the learned denoising process [9, 10].

More specifically, diffusion models are a type of latent variable model that use a Markov chain (MC) to map to the latent space. Through the Markov chain, noise is gradually added to the data x_i at each time step t to obtain the posterior probability $q(x_{1:T}|x_0)$, where $x_{1:T}$ represents the input data and is also the latent space. That is to say, the latent space of the Diffusion Models has the same dimensions as the input data.

Diffusion Models are divided into a forward diffusion process and a reverse denoising process. The Figure 3 below represents the diffusion process, where the transition from x_0 to the final x_T is a Markov chain, indicating a stochastic process in the state space that transitions from one state to another. The subscript corresponds to the image diffusion process in the Diffusion Models. Ultimately, the real image input as x_0 is asymptotically transformed into a pure Gaussian noise image x_T through the Diffusion Models.

2.2.1. Forward process

The forward process, or the diffusion process, utilizes a fixed form of Markov chain, which means gradually adding Gaussian noise to the image:

$$q(x_t|x_{t-1})\& = N\left(x_t; \sqrt{1-\beta_t}x_{t-1}, \beta_{tI}\right),\ q(x_{1:T}|x_0)\& = \prod_{t=1}^{T} q(x_t|x_{t-1}). \tag{1}$$

In DDPM, β_t is a predetermined fixed value parameter. The diffusion process has an important property that allows us to directly sample the noisy outcome x_t at any given moment t. When substituting $\alpha_t \coloneqq 1-\beta_t$ and $\overline{\alpha}_t \coloneqq \prod_{s=1}^{t} \alpha_s$ into the equation, we can derive the following result:

$$q(x_t|x_0) \coloneqq N\left(x_t; \sqrt{\overline{\alpha}_t}x_0, (1-\overline{\alpha}_t)I\right). \tag{2}$$

This analytical formula allows us to directly obtain images with any degree of added noise, facilitating subsequent training.

2.2.2. Reverse process

The reverse process starts with a random gaussian noise image x_T, and through gradual denoising, generates the final result x_0. This process is a Markov Chain, which can be defined as:

$$p_\theta(x_{0:T}) \coloneqq p(x_T)\prod_{t=1}^{T} p_\theta(x_{t-1}|x_t)\,, p_\theta(x_{t-1}|x_t) = N\left(x_{t-1}; \mu_\theta(x_t,t), \Sigma_\theta(x_t,t)\right). \tag{3}$$

2.2.3. Model training

To implement generation based on diffusion models, DDPM employs a U-Net structured Autoencoder to predict the noise at time t, that is, $\epsilon_\theta(x_t,t)$. The training objective used during network training is very straightforward:

$$\left|\left|\epsilon_\theta\left(\sqrt{\overline{\alpha}_t}x_0 + \sqrt{1-\overline{\alpha}_t}\epsilon\right) - \epsilon\right|\right|. \tag{4}$$

Here, ϵ represents Gaussian noise. The noise prediction network takes the noisy image as input, with the goal of predicting the added noise. The training objective aims for the predicted noise to be consistent with the actual noise. In DDPM, the definition of the mean μ_θ is as follows:

$$\mu_\theta(x_t, t) = \frac{1}{\sqrt{\alpha_t}}\left(x_t - \frac{1-\alpha_t}{\sqrt{1-\bar{\alpha}_t}}\epsilon_\theta(x_t, t)\right). \tag{5}$$

In DDPM, the variance term Σ_θ of the Gaussian distribution in the reverse process is taken as a constant term. Subsequent work has utilized a separate network branch to predict the variance term independently, achieving improved generative effects.

2.2. Score-based Generative Models (SGMs)

The above DDPM can be considered as a discrete form of SGM. SGM constructs a stochastic differential equation (SDE) to smoothly perturb the data distribution, transforming the original data distribution into a known prior distribution: $dx = f(x,t)\,dt + g(t)\,d\overline{w}$. The corresponding reverse SDE to transform the prior distribution back to the original data distribution is given by:

$$dx = [f(x,t) - g(t)^2 \nabla_x \log q_t(x)]\,dt + g(t)\,d\overline{w}. \tag{6}$$

Therefore, to reverse the diffusion process and generate data, the only information we need is the score function at each time point. The author can learn the score function using score-matching techniques with the following loss function:

$$E_{t,x_0,x_t}\left[\lambda(t)\left|\left|s_\theta(x_t,t) - \nabla_{x_t} \log q_{0t}(x_t|x_0)\right|\right|^2\right]. \tag{7}$$

2.3. Further Generalization

DDPMs and SGMs can be further generalized to the case of infinite time steps or noise levels, where the perturbation and denoising processes are solutions to stochastic differential equations (SDEs). We call this formulation Score SDE, as it leverages SDEs for noise perturbation and sample generation, and the denoising process requires estimating score functions of noisy data distributions. As the following objective:

$$E_{t\sim U[0,T],x_0\sim q(x_0),x_t\sim q(x_t|x_0)}\left[\lambda(t)\left|\left|s_\theta(x_t,t) - \nabla_{x_t} \log q_{0t}(x_t|x_0)\right|\right|^2\right], \tag{8}$$

Where $U[0,T]$ denotes the uniform distribution over $[0,T]$, and the remaining notations follow Equation (7).

3. Derivation on Diffusion Models

3.1. Denoising Diffusion Probabilistic Models (DDPM)

3.1.1. Forward process (Adding noise)

The variable x_0 represents the original image, and the process $x_{t-1} \to x_t$ is one of progressively adding noise. The relationship between two adjacent variables is linear, and we can model it as follows:

$$x_t = a_t x_{t-1} + b_t \varepsilon_t, \quad \varepsilon_t \sim N(0, I). \tag{9}$$

Let's first examine these two coefficients. Since x_{t-1} contains more information, a_t is a decay coefficient, whose value should satisfy $0 < a_t < 1$; similarly, the noise coefficient also satisfies $0 < b_t < 1$. When we continuously expand the expression using $x_{t-1} = a_{t-1}x_{t-2} + b_{t-1}\varepsilon_{t-1}$, we can obtain the following:

$$x_t = a_t x_{t-1} + b_t \varepsilon_t = a_t(a_{t-1}x_{t-2} + b_{t-1}\varepsilon_{t-1}) + b_t\varepsilon_t = a_t a_{t-1}x_{t-2} + a_t b_{t-1}\varepsilon_{t-1} + b_t\varepsilon_t = \cdots = (a_t \ldots a_1)x_0 + (a_t \ldots a_2)b_1\varepsilon_1 + (a_t \ldots a_3)b_2\varepsilon_2 + \cdots + a_t b_{t-1}\varepsilon_{t-1} + b_t\varepsilon_t. \quad (10)$$

Excluding the first term, the subsequent terms are sums of multiple independent normal noises. By additivity, their sum is also a normal distribution with a mean of 0, and the variance is the sum of the squares of the coefficients, which is $(a_t \ldots a_2)^2 b_1^2 + (a_t \ldots a_3)^2 b_2^2 + \cdots + a_t^2 b_{t-1}^2 + b_t^2$.

$$x_t = (a_t \ldots a_1)x_0 + \sqrt{(a_t \ldots a_2)^2 b_1^2 + (a_t \ldots a_3)^2 b_2^2 + \cdots + a_t^2 b_{t-1}^2 + b_t^2}\,\overline{\varepsilon}_t, \quad \overline{\varepsilon}_t \sim N(0, I) \quad (11)$$

There's a detail to note here. If we sum up the squared coefficients:

$$(a_t \ldots a_1)^2 + (a_t \ldots a_2)^2 b_1^2 + (a_t \ldots a_3)^2 b_2^2 + \cdots + a_t^2 b_{t-1}^2 + b_t^2 = (a_t \ldots a_2)^2 a_1^2 + (a_t \ldots a_2)^2 b_1^2 + (a_t \ldots a_3)^2 b_2^2 + \cdots + a_t^2 b_{t-1}^2 + b_t^2 = (a_t \ldots a_2)^2(a_1^2 + b_1^2) + (a_t \ldots a_3)^2 b_2^2 + \cdots + a_t^2 b_{t-1}^2 + b_t^2 = (a_t \ldots a_3)^2(a_2^2(a_1^2 + b_1^2) + b_2^2) + \cdots + a_t^2 b_{t-1}^2 + b_t^2 = a_t^2(a_{t-1}^2(\ldots(a_2^2(a_1^2 + b_1^2) + b_2^2) + \cdots) + b_{t-1}^2) + b_t^2. \quad (12)$$

The author finds that if we add a constraint, it will greatly simplify the expression, which is to require that $a_t^2 + b_t^2 = 1$. This way, the sum of squares becomes 1. At the same time, if we denote $\overline{a}_t = (a_t \ldots a_1)^2$, the latter part of the sum of squares (which is also the variance part of Equation (11)) can be represented as $1 - \overline{a}_t$. Thus, we can rewrite Equation (11) as:

$$x_t = \sqrt{\overline{a}_t}x_0 + \sqrt{1 - \overline{a}_t}\overline{\varepsilon}_t, \quad \overline{\varepsilon}_t \sim N(0, I). \quad (13)$$

This form is very neat and easier to handle. At this point, the author can rewrite Equation (9), substituting a_t with $\sqrt{\alpha_t}$ and b_t with $\sqrt{1 - \alpha_t}$, which is consistent with the original paper:

$$x_t = \sqrt{\alpha_t}x_{t-1} + \sqrt{1 - \alpha_t}\varepsilon_t, \quad \varepsilon_t \sim N(0, I). \quad (14)$$

The reason for detailing this so extensively is that other materials often present this formula directly, without explaining why the coefficients are set up this way (to have a sum of squares equal to 1). In addition, there's another point to note: the final $x_T \sim \mathbf{N}(\mathbf{0}, \mathbf{I})$ is assumed to follow a standard normal distribution, which implies that the preceding term's a_T is close to 0, and the latter term should be designed in the form of $(1 - a_t)$. It has also been mentioned in some articles that such a design ensures that the variances are on the same scale, which is known as variance-preserving.

The linear process mentioned above can also be viewed as sampling from a Gaussian distribution, which in practice is facilitated by the reparameterization trick. We can write out the complete forward transition process:

$$x_t \sim q(x_t|x_{t-1}) = N\left(x_t; \sqrt{\alpha_t}x_{t-1}, (1 - \alpha_t)I\right). \quad (15)$$

Note that α_t is not a learned parameter. In the original article, there is another symbol β_t and their relationship is $\alpha_t = 1 - \beta_t$ .We can also express Equation (13) in probabilistic form, substituting a with α.

$$\boldsymbol{x}_t \sim q(\boldsymbol{x}_t \mid \boldsymbol{x}_0) = \mathcal{N}\left(\boldsymbol{x}_t; \sqrt{\bar{\alpha}_t}\boldsymbol{x}_0, (1 - \bar{\alpha}_t)\boldsymbol{I}\right)\Big), \; \bar{\alpha}_t = \prod_{s=1}^{t} \alpha_s \quad (16)$$

The entire forward process is a posterior estimation, which is represented as: (based on the joint probability density and the properties of Markov chains).

$$q(x_{1:T}|x_0) = \prod_{t=1}^{T} q(x_t|x_{t-1}). \quad (17)$$

3.1.2. Reverse process (Denoising and generation)

The Variational Autoencoder (VAE), a well-known generative model, employs a straightforward encoding and decoding (generation) process, which can be summarized as:

$$\text{encoding: } x \to z, \quad \text{decoding: } z \to x. \quad (18)$$

In the context of Variational Autoencoders (VAEs), there are three key distributions involved: the encoder model $q(z|x)$, which approximates the posterior distribution of the latent variables z given the data x; the decoder model $p(x|z)$, which models the likelihood of the data given the latent variables; and the prior distribution $q(z)$, which is typically chosen to be a simple distribution, like a Gaussian, that the latent variables are assumed to follow.

The simplicity of this structure, while making VAEs conceptually straightforward and computationally feasible, also imposes limitations on their generative capabilities. This is where diffusion models come into play, which can be viewed as a form of Hierarchical Variational Autoencoder (HVAE). This interpretation expands the encoding and decoding (generation) process across multiple steps T, effectively creating a more complex and gradual transition between the data and the latent space:

$$\text{encoding: } \mathrm{x}_0 \to \mathrm{x}_1 \to \mathrm{x}_2 \to \cdots \to \mathrm{x}_\mathrm{T}, \tag{19}$$

$$\text{decoding: } \mathrm{x}_\mathrm{T} \to \mathrm{x}_{\mathrm{T}-1} \to \mathrm{x}_{\mathrm{T}-2} \to \cdots \to \mathrm{x}_0. \tag{20}$$

The distribution relationship is similar, except that each step is characterized by $q(x_t|x_{t-1}), p(x_{t-1}|x_t)$. The advantage of this approach is somewhat akin to using piecewise linear functions to approximate a curve, which enhances the generative performance.

We model each step of the reverse process as $p_\theta(x_{t-1}|x_t)$ similar to the forward process introduced earlier. The linear relationship between them can also be modeled as a Gaussian distribution. However, unlike the forward process where the parameters are deterministic, here they are not fixed, so we denote it as $p_\theta(x_{t-1}|x_t) = \mathbf{N}\big(x_{t-1}; \mu_\theta(x_t,t), \Sigma_\theta(x_t,t)\big)$. The entire reverse process is represented as a joint probability distribution $p_\theta(x_{0:T}) = p(x_T)\prod_{t=1}^{T} p_\theta(x_{t-1}|x_t)$. According to the law of total probability, we can write the likelihood function we want to solve as:

$$\begin{aligned}
&\log p_\theta(\boldsymbol{x}_0) = \log \int p_\theta(\boldsymbol{x}_0, \boldsymbol{x}_1, \dots, \boldsymbol{x}_T) d\boldsymbol{x}_1 d\boldsymbol{x}_2 \dots d\boldsymbol{x}_T \\
&= \log \int p_\theta(\boldsymbol{x}_{0:T}) d\boldsymbol{x}_{1:T} \\
&= \log \int \frac{p_\theta(x_{0:T}) q(x_{1:T}|x_0)}{q(x_{1:T}|x_0)} d\boldsymbol{x}_{1:T} \\
&= \log \mathbb{E}_{q(x_{1:T}|x_0)}\left[\frac{p_\theta(x_{0:T})}{q(x_{1:T}|x_0)}\right] \\
&\geq \mathbb{E}_{q(x_{1:T}|x_0)}\left[\log \frac{p_\theta(x_{0:T})}{q(x_{1:T}|x_0)}\right] \\
&= \mathbb{E}_{q(x_{1:T}|x_0)}\left[\log \frac{p(x_T)\prod_{t=1}^{T} p_\theta(x_{t-1}|x_t)}{\prod_{t=1}^{T} q(x_t|x_{t-1})}\right] \\
&= \mathbb{E}_{q(x_{1:T}|x_0)}\left[\log \frac{p(x_T) p_\theta(x_0|x_1)\prod_{t=2}^{T} p_\theta(x_{t-1}|x_t)}{q(x_T|x_{T-1})\prod_{t=1}^{T-1} q(x_t|x_{t-1})}\right] \\
&= \mathbb{E}_{q(x_{1:T}|x_0)}\left[\log \frac{p(x_T) p_\theta(x_0|x_1)\prod_{t=1}^{T-1} p_\theta(x_t|x_{t+1})}{q(x_T|x_{T-1})\prod_{t=1}^{T-1} q(x_t|x_{t-1})}\right] \\
&= \mathbb{E}_{q(x_{1:T}|x_0)}\left[\log \frac{p(x_T) p_\theta(x_0|x_1)}{q(x_T|x_{T-1})}\right] + \mathbb{E}_{q(x_{1:T}|x_0)}\left[\log \prod_{t=1}^{T-1} \frac{p_\theta(x_t|x_{t+1})}{q(x_t|x_{t-1})}\right] \\
&= \mathbb{E}_{q(x_{1:T}|x_0)}[\log p_\theta(\boldsymbol{x}_0 \mid \boldsymbol{x}_1)] + \mathbb{E}_{q(x_{1:T}|x_0)}\left[\log \frac{p(x_T)}{q(x_T|x_{T-1})}\right] + \mathbb{E}_{q(x_{1:T}|x_0)}\left[\sum_{t=1}^{T-1} \log \frac{p_\theta(x_t|x_{t+1})}{q(x_t|x_{t-1})}\right] \\
&= \mathbb{E}_{q(x_{1:T}|x_0)}[\log p_\theta(\boldsymbol{x}_0 \mid \boldsymbol{x}_1)] + \mathbb{E}_{q(x_{1:T}|x_0)}\left[\log \frac{p(x_T)}{q(x_T|x_{T-1})}\right] + \sum_{t=1}^{T-1} \mathbb{E}_{q(x_{1:T}|x_0)}\left[\log \frac{p_\theta(x_t|x_{t+1})}{q(x_t|x_{t-1})}\right] \\
&= \mathbb{E}_{q(x_1|x_0)}[\log p_\theta(\boldsymbol{x}_0 \mid \boldsymbol{x}_1)] + \mathbb{E}_{q(x_{T-1},x_T|x_0)}\left[\log \frac{p(x_T)}{q(x_T|x_{T-1})}\right] + \sum_{t=1}^{T-1} \mathbb{E}_{q(x_{t-1},x_t,x_{t+1}|x_0)}\left[\log \frac{p_\theta(x_t|x_{t+1})}{q(x_t|x_{t-1})}\right] \\
&= \mathbb{E}_{q(x_1|x_0)}[\log p_\theta(\boldsymbol{x}_0 \mid \boldsymbol{x}_1)] - \mathbb{E}_{q(x_{T-1}|x_0)}\mathbb{E}_{q(x_T|x_{T-1})}\left[\log \frac{q(x_T|x_{T-1})}{p(x_T)}\right] \\
&- \sum_{t=1}^{T-1} \mathbb{E}_{q(x_{t-1},x_{t+1}|x_0)}\mathbb{E}_{q(x_t|x_0)}\left[\log \frac{q(x_t|x_{t-1})}{p_\theta(x_t|x_{t+1})}\right] \\
&= \underbrace{\mathbb{E}_{q(x_1|x_0)}[\log p_\theta(\boldsymbol{x}_0 \mid \boldsymbol{x}_1)]}_{\text{reconstruction term}} - \underbrace{\mathbb{E}_{q(x_{T-1}|x_0)}\big[D_{\mathrm{KL}}\big(q(\boldsymbol{x}_T \mid \boldsymbol{x}_{T-1}) \parallel p(\boldsymbol{x}_T)\big)\big]}_{\text{prior matching term}} \\
&- \sum_{t=1}^{T-1} \underbrace{\mathbb{E}_{q(x_{t-1},x_{t+1}|x_0)}\big[D_{\mathrm{KL}}\big(q(\boldsymbol{x}_t \mid \boldsymbol{x}_{t-1}) \parallel p_\theta(\boldsymbol{x}_t \mid \boldsymbol{x}_{t+1})\big)\big]}_{\text{consistency term}}
\end{aligned} \tag{21}$$

The optimization of the above objective function uses Monte Carlo estimation. However, the last term involves calculating $q(x_t|x_{t-1}), p_\theta(x_t|x_{t+1})$, which involves three random variables and significantly increases the variance of the estimation. Therefore, we hope to mitigate this issue.

Given that x_0 is fixed, and due to the properties of the Markov chain, adding an extra term does not affect the outcome. Therefore, by leveraging this information:

$$q(x_t|x_{t-1}, x_0) = \frac{q(x_{t-1}|x_t, x_0)q(x_t|x_0)}{q(x_{t-1}|x_0)} \tag{22}$$

We can rewrite the above likelihood optimization objective, with changes starting from the fourth line:

$$\begin{aligned}
&\log\, p_\theta(\boldsymbol{x}_0) \geq \mathbb{E}_{q(x_{1:T}|x_0)}\left[\log \frac{p_\theta(x_{0:T})}{q(x_{1:T}|x_0)}\right] \\
&= \mathbb{E}_{q(x_{1:T}|x_0)}\left[\log \frac{p(x_T)\prod_{t=1}^{T} p_\theta(x_{t-1}|x_t)}{\prod_{t=1}^{T} q(x_t|x_{t-1})}\right] \\
&= \mathbb{E}_{q(x_{1:T}|x_0)}\left[\log \frac{p(x_T)p_\theta(x_0|x_1)\prod_{t=2}^{T} p_\theta(x_{t-1}|x_t)}{q(x_1|x_0)\prod_{t=2}^{T} q(x_t|x_{t-1})}\right] \\
&= \mathbb{E}_{q(x_{1:T}|x_0)}\left[\log \frac{p(x_T)p_\theta(x_0|x_1)\prod_{t=2}^{T} p_\theta(x_{t-1}|x_t)}{q(x_1|x_0)\prod_{t=2}^{T} q(x_t|x_{t-1},x_0)}\right] \\
&= \mathbb{E}_{q(x_{1:T}|x_0)}\left[\log \frac{p_\theta(x_T)p_\theta(x_0|x_1)}{q(x_1|x_0)} + \log \prod_{t=2}^{T} \frac{p_\theta(x_{t-1}|x_t)}{q(x_t|x_{t-1},x_0)}\right] \\
&= \mathbb{E}_{q(x_{1:T}|x_0)}\left[\log \frac{p(x_T)p_\theta(x_0|x_1)}{q(x_1|x_0)} + \log \prod_{t=2}^{T} \frac{p_\theta(x_{t-1}|x_t)}{\frac{q(x_{t-1}|x_t,x_0)q(x_t|x_0)}{q(x_{t-1}|x_0)}}\right] \\
&= \mathbb{E}_{q(x_{1:T}|x_0)}\left[\log \frac{p(x_T)p_\theta(x_0|x_1)}{q(x_1|x_0)} + \log \prod_{t=2}^{T} \frac{q(x_{t-1}|x_0)}{q(x_t|x_0)} + \log \prod_{t=2}^{T} \frac{p_\theta(x_{t-1}|x_t)}{q(x_{t-1}|x_t,x_0)}\right] \\
&= \mathbb{E}_{q(x_{1:T}|x_0)}\left[\log \frac{p(x_T)p_\theta(x_0|x_1)}{q(x_1|x_0)} + \log \frac{q(x_1|x_0)}{q(x_2|x_0)}\frac{q(x_2|x_0)}{q(x_3|x_0)}\ldots\frac{q(x_{T-1}|x_0)}{q(x_T|x_0)} + \log \prod_{t=2}^{T} \frac{p_\theta(x_{t-1}|x_t)}{q(x_{t-1}|x_t,x_0)}\right] \\
&= \mathbb{E}_{q(x_{1:T}|x_0)}\left[\log \frac{p(x_T)p_\theta(x_0|x_1)}{q(x_1|x_0)} + \log \frac{q(x_1|x_0)}{q(x_T|x_0)} + \log \prod_{t=2}^{T} \frac{p_\theta(x_{t-1}|x_t)}{q(x_{t-1}|x_t,x_0)}\right] \\
&= \mathbb{E}_{q(x_{1:T}|x_0)}\left[\log \frac{p(x_T)p_\theta(x_0|x_1)}{q(x_1|x_0)}\frac{q(x_1|x_0)}{q(x_T|x_0)} + \log \prod_{t=2}^{T} \frac{p_\theta(x_{t-1}|x_t)}{q(x_{t-1}|x_t,x_0)}\right] \\
&= \mathbb{E}_{q(x_{1:T}|x_0)}\left[\log \frac{p(x_T)p_\theta(x_0|x_1)}{q(x_T|x_0)} + \sum_{t=2}^{T} \log \frac{p_\theta(x_{t-1}|x_t)}{q(x_{t-1}|x_t,x_0)}\right] \\
&= \mathbb{E}_{q(x_{1:T}|x_0)}[\log\, p_\theta(\boldsymbol{x}_0 \mid \boldsymbol{x}_1)] + \mathbb{E}_{q(x_{1:T}|x_0)}\left[\log \frac{p(x_T)}{q(x_T|x_0)}\right] + \sum_{t=2}^{T} \mathbb{E}_{q(x_{1:T}|x_0)}\left[\log \frac{p_\theta(x_{t-1}|x_t)}{q(x_{t-1}|x_t,x_0)}\right] \\
&= \mathbb{E}_{q(x_1|x_0)}[\log\, p_\theta(\boldsymbol{x}_0 \mid \boldsymbol{x}_1)] + \mathbb{E}_{q(x_T|x_0)}\left[\log \frac{p(x_T)}{q(x_T|x_0)}\right] + \sum_{t=2}^{T} \mathbb{E}_{q(x_{t-1},x_t|x_0)}\left[\log \frac{p_\theta(x_{t-1}|x_t)}{q(x_{t-1}|x_t,x_0)}\right] \\
&= \mathbb{E}_{q(x_1|x_0)}[\log\, p_\theta(\boldsymbol{x}_0 \mid \boldsymbol{x}_1)] - \mathbb{E}_{q(x_T|x_0)}\left[\log \frac{q(x_T|x_0)}{p(x_T)}\right] - \sum_{t=2}^{T} \mathbb{E}_{q(x_t|x_0)}\mathbb{E}_{q(x_{t-1}|x_t,x_0)}\left[\log \frac{q(x_{t-1}|x_t,x_0)}{p_\theta(x_{t-1}|x_t)}\right] \\
&= \underbrace{\mathbb{E}_{q(x_1|x_0)}[\log\, p_\theta(\boldsymbol{x}_0 \mid \boldsymbol{x}_1)]}_{\text{reconstruction term}} - \underbrace{D_{\mathrm{KL}}\big(q(\boldsymbol{x}_T \mid \boldsymbol{x}_0) \parallel p(\boldsymbol{x}_T)\big)}_{\text{prior matching term}} - \sum_{t=2}^{T} \underbrace{\mathbb{E}_{q(x_t|x_0)}\big[D_{\mathrm{KL}}\big(q(\boldsymbol{x}_{t-1} \mid \boldsymbol{x}_t, \boldsymbol{x}_0) \parallel p_\theta(\boldsymbol{x}_{t-1} \mid \boldsymbol{x}_t)\big)\big]}_{\text{denoising matching term}}
\end{aligned} \tag{23}$$

Now, we aim to further model $q(x_{t-1}|x_t, x_0)$ and $p_\theta(x_{t-1}|x_t)$ so that we can calculate the KL divergence between them:

$$
\begin{aligned}
&q(\boldsymbol{x}_{t-1} \mid \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)} \\
&= \frac{\mathcal{N}(\boldsymbol{x}_t;\sqrt{\alpha_t}\boldsymbol{x}_{t-1},(1-\alpha_t)\mathbf{I})\mathcal{N}(\boldsymbol{x}_{t-1};\sqrt{\bar{\alpha}_{t-1}}\boldsymbol{x}_0,(1-\bar{\alpha}_{t-1})\mathbf{I})}{\mathcal{N}(\boldsymbol{x}_t;\sqrt{\bar{\alpha}_t}\boldsymbol{x}_0,(1-\bar{\alpha}_t)\mathbf{I})} \\
&\propto \exp\left\{-\left[\frac{\left(\boldsymbol{x}_t-\sqrt{\alpha_t}\boldsymbol{x}_{t-1}\right)^2}{2(1-\alpha_t)} + \frac{\left(\boldsymbol{x}_{t-1}-\sqrt{\bar{\alpha}_{t-1}}\boldsymbol{x}_0\right)^2}{2(1-\bar{\alpha}_{t-1})} - \frac{\left(\boldsymbol{x}_t-\sqrt{\bar{\alpha}_t}\boldsymbol{x}_0\right)^2}{2(1-\bar{\alpha}_t)}\right]\right\} \\
&= \exp\left\{-\frac{1}{2}\left[\frac{\left(\boldsymbol{x}_t-\sqrt{\alpha_t}\boldsymbol{x}_{t-1}\right)^2}{1-\alpha_t} + \frac{\left(\boldsymbol{x}_{t-1}-\sqrt{\bar{\alpha}_{t-1}}\boldsymbol{x}_0\right)^2}{1-\bar{\alpha}_{t-1}} - \frac{\left(\boldsymbol{x}_t-\sqrt{\bar{\alpha}_t}\boldsymbol{x}_0\right)^2}{1-\bar{\alpha}_t}\right]\right\} \\
&= \exp\left\{-\frac{1}{2}\left[-\frac{2\sqrt{\alpha_t}\boldsymbol{x}_t\boldsymbol{x}_{t-1}}{1-\alpha_t} + \frac{\alpha_t\boldsymbol{x}_{t-1}^2}{1-\alpha_t} + \frac{\boldsymbol{x}_{t-1}^2}{1-\bar{\alpha}_{t-1}} - \frac{2\sqrt{\bar{\alpha}_{t-1}}\boldsymbol{x}_{t-1}\boldsymbol{x}_0}{1-\bar{\alpha}_{t-1}} + C(\boldsymbol{x}_t,\boldsymbol{x}_0)\right]\right\} \\
&= \exp\left\{-\frac{1}{2}\left[\left(\frac{\alpha_t}{1-\alpha_t} + \frac{1}{1-\bar{\alpha}_{t-1}}\right)\boldsymbol{x}_{t-1}^2 - 2\left(\frac{\sqrt{\alpha_t}\boldsymbol{x}_t}{1-\alpha_t} + \frac{\sqrt{\bar{\alpha}_{t-1}}\boldsymbol{x}_0}{1-\bar{\alpha}_{t-1}}\right)\boldsymbol{x}_{t-1} + C(\boldsymbol{x}_t,\boldsymbol{x}_0)\right]\right\} \\
&= \exp\left\{-\frac{1}{2}\left[\frac{\alpha_t(1-\bar{\alpha}_{t-1})+1-\alpha_t}{(1-\alpha_t)(1-\bar{\alpha}_{t-1})}\boldsymbol{x}_{t-1}^2 - 2\left(\frac{\sqrt{\alpha_t}\boldsymbol{x}_t}{1-\alpha_t} + \frac{\sqrt{\bar{\alpha}_{t-1}}\boldsymbol{x}_0}{1-\bar{\alpha}_{t-1}}\right)\boldsymbol{x}_{t-1} + C(\boldsymbol{x}_t,\boldsymbol{x}_0)\right]\right\} \\
&= \exp\left\{-\frac{1}{2}\left[\frac{\alpha_t-\bar{\alpha}_t+1-\alpha_t}{(1-\alpha_t)(1-\bar{\alpha}_{t-1})}\boldsymbol{x}_{t-1}^2 - 2\left(\frac{\sqrt{\alpha_t}\boldsymbol{x}_t}{1-\alpha_t} + \frac{\sqrt{\bar{\alpha}_{t-1}}\boldsymbol{x}_0}{1-\bar{\alpha}_{t-1}}\right)\boldsymbol{x}_{t-1} + C(\boldsymbol{x}_t,\boldsymbol{x}_0)\right]\right\} \\
&= \exp\left\{-\frac{1}{2}\left[\frac{1-\bar{\alpha}_t}{(1-\alpha_t)(1-\bar{\alpha}_{t-1})}\boldsymbol{x}_{t-1}^2 - 2\left(\frac{\sqrt{\alpha_t}\boldsymbol{x}_t}{1-\alpha_t} + \frac{\sqrt{\bar{\alpha}_{t-1}}\boldsymbol{x}_0}{1-\bar{\alpha}_{t-1}}\right)\boldsymbol{x}_{t-1}\right] - \frac{1}{2}C(\boldsymbol{x}_t,\boldsymbol{x}_0)\right\} \\
&= \exp\left\{-\frac{1}{2}\left(\frac{1-\bar{\alpha}_t}{(1-\alpha_t)(1-\bar{\alpha}_{t-1})}\right)\left[\boldsymbol{x}_{t-1}^2 - 2\frac{\left(\frac{\sqrt{\alpha_t}\boldsymbol{x}_t}{1-\alpha_t}+\frac{\sqrt{\bar{\alpha}_{t-1}}\boldsymbol{x}_0}{1-\bar{\alpha}_{t-1}}\right)}{\frac{1-\bar{\alpha}_t}{(1-\alpha_t)(1-\bar{\alpha}_{t-1})}}\boldsymbol{x}_{t-1}\right] - \frac{1}{2}C(\boldsymbol{x}_t,\boldsymbol{x}_0)\right\} \\
&= \exp\left\{-\frac{1}{2}\left(\frac{1-\bar{\alpha}_t}{(1-\alpha_t)(1-\bar{\alpha}_{t-1})}\right)\left[\boldsymbol{x}_{t-1}^2 - 2\frac{\left(\frac{\sqrt{\alpha_t}\boldsymbol{x}_t}{1-\alpha_t}+\frac{\sqrt{\bar{\alpha}_{t-1}}\boldsymbol{x}_0}{1-\bar{\alpha}_{t-1}}\right)(1-\alpha_t)(1-\bar{\alpha}_{t-1})}{1-\bar{\alpha}_t}\boldsymbol{x}_{t-1}\right] - \frac{1}{2}C(\boldsymbol{x}_t\right. \\
&= \exp\left\{-\frac{1}{2}\left(\frac{1}{\frac{(1-\alpha_t)(1-\bar{\alpha}_{t-1})}{1-\bar{\alpha}_t}}\right)\left[\boldsymbol{x}_{t-1}^2 - 2\frac{\sqrt{\alpha_t}(1-\bar{\alpha}_{t-1})\boldsymbol{x}_t+\sqrt{\bar{\alpha}_{t-1}}(1-\alpha_t)\boldsymbol{x}_0}{1-\bar{\alpha}_t}\boldsymbol{x}_{t-1}\right] - \frac{1}{2}C(\boldsymbol{x}_t,\boldsymbol{x}_0)\right\} \\
&= \exp\left\{-\frac{\left(\boldsymbol{x}_{t-1}-\frac{\sqrt{\alpha_t}(1-\bar{\alpha}_{t-1})\boldsymbol{x}_t+\sqrt{\bar{\alpha}_{t-1}}(1-\alpha_t)\boldsymbol{x}_0}{1-\bar{\alpha}_t}\right)^2}{2\left(\frac{(1-\alpha_t)(1-\bar{\alpha}_{t-1})}{1-\bar{\alpha}_t}\right)}\right\} \\
&\propto \mathcal{N}\left(\boldsymbol{x}_{t-1};\underbrace{\frac{\sqrt{\alpha_t}(1-\bar{\alpha}_{t-1})\boldsymbol{x}_t+\sqrt{\bar{\alpha}_{t-1}}(1-\alpha_t)\boldsymbol{x}_0}{1-\bar{\alpha}_t}}_{\mu_q(\boldsymbol{x}_t,\boldsymbol{x}_0)},\underbrace{\frac{(1-\alpha_t)(1-\bar{\alpha}_{t-1})}{1-\bar{\alpha}_t}}_{\boldsymbol{\Sigma}_q(t)}\mathbf{I}\right)
\end{aligned}
\tag{24}
$$

Therefore, we model $q(x_{t-1}|x_t, x_0)$, which is also referred to as the posterior distribution of the forward process. The C in the above expression can be accurately substituted after derivation. We also present the form as it appears in the original paper: $x_{t-1} \sim q(x_{t-1}|x_t, x_0) = \mathbf{N}(\mu_q, \Sigma_q) = \mathbf{N}\left(\frac{\sqrt{\alpha_t}(1-\overline{\alpha}_{t-1})x_t+\sqrt{\overline{\alpha}_{t-1}}(1-\alpha_t)x_0}{1-\overline{\alpha}_t}, \frac{(1-\alpha_t)(1-\overline{\alpha}_{t-1})}{1-\overline{\alpha}_t}\mathbf{I}\right) = \mathbf{N}\left(\frac{\sqrt{\overline{\alpha}_{t-1}}\beta_t}{1-\overline{\alpha}_t}x_0 + \frac{\sqrt{\alpha_t}(1-\overline{\alpha}_{t-1})}{1-\overline{\alpha}_t}x_t, \frac{1-\overline{\alpha}_{t-1}}{1-\overline{\alpha}_t}\beta_{t\mathbf{I}}\right)$. Now, we also need to model $p_\theta(x_{t-1}|x_t) = \mathbf{N}(\mu_\theta, \Sigma_\theta)$. The variance can be modeled directly as the same, $\Sigma_\theta = \Sigma_q$. Based on: $D_{\mathrm{KL}}\left(\mathcal{N}(\boldsymbol{x};\boldsymbol{\mu}_x,\boldsymbol{\Sigma}_x) \parallel \mathcal{N}(\boldsymbol{y};\boldsymbol{\mu}_y,\boldsymbol{\Sigma}_y)\right) = \frac{1}{2}\left[\log\frac{|\boldsymbol{\Sigma}_y|}{|\boldsymbol{\Sigma}_x|} - d + \mathrm{tr}\left(\boldsymbol{\Sigma}_y^{-1}\boldsymbol{\Sigma}_x\right) + \left(\boldsymbol{\mu}_y - \boldsymbol{\mu}_x\right)^T\boldsymbol{\Sigma}_y^{-1}\left(\boldsymbol{\mu}_y - \boldsymbol{\mu}_x\right)\right]$. We can derive:

$$\begin{aligned}
&\arg\min_{\boldsymbol{\theta}} D_{\mathrm{KL}}\left(q(\boldsymbol{x}_{t-1} \mid \boldsymbol{x}_t, \boldsymbol{x}_0) \parallel p_{\boldsymbol{\theta}}(\boldsymbol{x}_{t-1} \mid \boldsymbol{x}_t)\right) \\
=&\arg\min_{\boldsymbol{\theta}} D_{\mathrm{KL}}\left(\mathcal{N}\left(\boldsymbol{x}_{t-1}; \boldsymbol{\mu}_q, \boldsymbol{\Sigma}_q(t)\right) \parallel \mathcal{N}\left(\boldsymbol{x}_{t-1}; \boldsymbol{\mu}_{\boldsymbol{\theta}}, \boldsymbol{\Sigma}_q(t)\right)\right) \\
=&\arg\min_{\boldsymbol{\theta}} \frac{1}{2}\left[\log \frac{|\boldsymbol{\Sigma}_q(t)|}{|\boldsymbol{\Sigma}_q(t)|} - d + \operatorname{tr}\left(\boldsymbol{\Sigma}_q(t)^{-1}\boldsymbol{\Sigma}_q(t)\right) + \left(\boldsymbol{\mu}_{\boldsymbol{\theta}} - \boldsymbol{\mu}_q\right)^T \boldsymbol{\Sigma}_q(t)^{-1}\left(\boldsymbol{\mu}_{\boldsymbol{\theta}} - \boldsymbol{\mu}_q\right)\right] \\
=&\arg\min_{\boldsymbol{\theta}} \frac{1}{2}\left[\left(\boldsymbol{\mu}_{\boldsymbol{\theta}} - \boldsymbol{\mu}_q\right)^T \boldsymbol{\Sigma}_q(t)^{-1}\left(\boldsymbol{\mu}_{\boldsymbol{\theta}} - \boldsymbol{\mu}_q\right)\right] \\
=&\arg\min_{\boldsymbol{\theta}} \frac{1}{2}\left[\left(\boldsymbol{\mu}_{\boldsymbol{\theta}} - \boldsymbol{\mu}_q\right)^T \left(\sigma_q^2(t)\mathbf{I}\right)^{-1}\left(\boldsymbol{\mu}_{\boldsymbol{\theta}} - \boldsymbol{\mu}_q\right)\right] \\
=&\arg\min_{\boldsymbol{\theta}} \frac{1}{2\sigma_q^2(t)}\left[\left\|\boldsymbol{\mu}_{\boldsymbol{\theta}} - \boldsymbol{\mu}_q\right\|_2^2\right]
\end{aligned} \tag{25}$$

From this conclusion, we can see that what we ultimately need to compare are the means of the two models, where μ_q is known, and μ_θ is what the model needs to learn. Let's look at the form of μ_q:

$$\mu_q = \frac{\sqrt{\overline{\alpha}_{t-1}}\beta_t}{1-\overline{\alpha}_t} x_0 + \frac{\sqrt{\alpha_t}(1-\overline{\alpha}_{t-1})}{1-\overline{\alpha}_t} x_t \tag{26}$$

In the forward process, both x_0 and x_t are known, and they can be transformed into each other. The mean of the reverse process, $\mu_\theta(x_t, t)$, is a function of x_t. The most general way to model this is to have a neural network learn the true value of μ_q directly based on (x_t, t), which can obviously be quite complex. Because if μ_q and μ_θ are very similar in form, then it would only be necessary to differentiate between their differences, significantly simplifying the optimization process. A very natural idea is to let the neural network learn the true value of x_0, that is, $\widehat{x_0} = f_\theta(x_t, t)$. Therefore, we model it as:

$$\mu_\theta = \frac{\sqrt{\overline{\alpha}_{t-1}}\beta_t f_\theta(x_t,t)}{1-\overline{\alpha}_t} + \frac{\sqrt{\alpha_t}(1-\overline{\alpha}_{t-1})x_t}{1-\overline{\alpha}_t} \tag{27}$$

We can calculate:

$$\begin{aligned}
&arg\min_{\boldsymbol{\theta}} D_{KL}\left(q(\boldsymbol{x}_{t-1} \mid \boldsymbol{x}_t, \boldsymbol{x}_0) \parallel p_{\boldsymbol{\theta}}(\boldsymbol{x}_{t-1} \mid \boldsymbol{x}_t)\right) \\
=&arg\min_{\boldsymbol{\theta}} \frac{1}{2\sigma_q^2(t)}\left[\left\|\boldsymbol{\mu}_{\boldsymbol{\theta}} - \boldsymbol{\mu}_q\right\|_2^2\right] \\
=&arg\min_{\boldsymbol{\theta}} \frac{1}{2\sigma_q^2(t)}\left[\left\|\frac{\sqrt{\bar{\alpha}_{t-1}}\beta_t}{1-\bar{\alpha}_t}\left(f_{\boldsymbol{\theta}}(\boldsymbol{x}_t, t) - \boldsymbol{x}_0\right)\right\|_2^2\right] \\
=&arg\min_{\boldsymbol{\theta}} \frac{1}{2\sigma_q^2(t)}\frac{\bar{\alpha}_{t-1}\beta_t^2}{(1-\bar{\alpha}_t)^2}\left[\left\|f_{\boldsymbol{\theta}}(\boldsymbol{x}_t, t) - \boldsymbol{x}_0\right\|_2^2\right]
\end{aligned} \tag{28}$$

Up to this point, we can conclude that, in fact, it can be understood as learning to restore the original image from any step t of noise. In summary, we are now able to solve Equation (20). However, this conclusion's form does not match the original papers. Go back to Equation (25), and with the help of Equation (13), we can eliminate x_0 to get:

$$\begin{aligned}\boldsymbol{\mu}_q &= \frac{\sqrt{\alpha_{t-1}}\beta_t}{1-\bar{\alpha}_t}\boldsymbol{x}_0 + \frac{\sqrt{\alpha_t}(1-\bar{\alpha}_{t-1})}{1-\bar{\alpha}_t}\boldsymbol{x}_t \\ &= \frac{\sqrt{\bar{\alpha}_{t-1}}\beta_t}{1-\bar{\alpha}_t}\frac{\boldsymbol{x}_t-\sqrt{1-\bar{\alpha}_t}\varepsilon_t}{\sqrt{\bar{\alpha}_t}} + \frac{\sqrt{\alpha_t}(1-\bar{\alpha}_{t-1})}{1-\bar{\alpha}_t}\boldsymbol{x}_t \\ &= \left[\frac{\sqrt{\bar{\alpha}_{t-1}}\beta_t}{(1-\bar{\alpha}_t)\sqrt{\bar{\alpha}_t}} + \frac{\sqrt{\alpha_t}(1-\bar{\alpha}_{t-1})}{1-\bar{\alpha}_t}\right]\boldsymbol{x}_t - \frac{1-\alpha_t}{\sqrt{1-\bar{\alpha}_t}\sqrt{\alpha_t}}\boldsymbol{\varepsilon}_t \\ &= \frac{(1-\alpha_t)+\alpha_t(1-\bar{\alpha}_{t-1})}{(1-\bar{\alpha}_t)\sqrt{\alpha_t}}\boldsymbol{x}_t - \frac{1-\alpha_t}{\sqrt{1-\bar{\alpha}_t}\sqrt{\alpha_t}}\boldsymbol{\varepsilon}_t \\ &= \frac{1-\alpha_t+\alpha_t-\bar{\alpha}_t}{(1-\bar{\alpha}_t)\sqrt{\alpha_t}}\boldsymbol{x}_t - \frac{1-\alpha_t}{\sqrt{1-\bar{\alpha}_t}\sqrt{\alpha_t}}\boldsymbol{\varepsilon}_t \\ &= \frac{1}{\sqrt{\alpha_t}}\boldsymbol{x}_t - \frac{1-\alpha_t}{\sqrt{1-\bar{\alpha}_t}\sqrt{\alpha_t}}\varepsilon_t\end{aligned} \tag{29}$$

Now, we have the neural network learn the noise ε_t added during the transformation process from x_0 to x_t, that is, $\widehat{\varepsilon_t} = f_\theta(x_t, t)$. With this approach, the optimization objective for Equation (24) can be written as $\arg\min_{\boldsymbol{\theta}} D_{\mathrm{KL}}\big(q(\boldsymbol{x}_{t-1} \mid \boldsymbol{x}_t, \boldsymbol{x}_0) \parallel p_{\boldsymbol{\theta}}(\boldsymbol{x}_{t-1} \mid \boldsymbol{x}_t)\big)$. This formula now matches the formula in the original text as Equation (30).

$$\mathbb{E}_{x_0,\epsilon}\left[\frac{\beta_t^2}{2\sigma_t^2\alpha_t(1-\hat{\alpha}_t)}\left\|\epsilon - \epsilon_0\big(\sqrt{\hat{\alpha}_t}x_0 + \sqrt{1-\hat{\alpha}_t}\epsilon, t\big)\right\|^2\right] \tag{30}$$

The difference between Equation (29) and Equation (27) lies in that one estimates the image, while the other estimates the noise. However, experiments have shown that estimating the noise yields better results. At the same time, the original paper mentions that omitting the preceding coefficients and randomly sampling time t during training can lead to improved generation outcomes. The formula in the original text is written as: $L_{\text{simple}}(\theta) := \mathbb{E}_{t,\boldsymbol{x}_0,\boldsymbol{\epsilon}}\left[\left\|\boldsymbol{\epsilon} - \boldsymbol{\epsilon}_\theta\big(\sqrt{\bar{\alpha}_t}\boldsymbol{x}_0 + \sqrt{1-\bar{\alpha}_t}\boldsymbol{\epsilon}, t\big)\right\|^2\right]$.

3.2. Score-Based Generative Models(SGM)

As mentioned above, the Diffusion Model can be interpreted not only from the perspective of DDPM but also can be given a fresh interpretation through the Score-based SDE approach.

Note: To be consistent with the literature, here we use $s_\theta(x, t)$ to denote the neural network's predicted values, rather than $f_\theta(x, t)$. To simplify the notation, $\nabla \log p_\theta(x)$ is used to represent $\nabla_x \log p_\theta(x)$. Also, $p(x;\theta)$ is equivalent to $p_\theta(x)$. Start with some conclusions from DDPM. According to Equation (16), for the forward process:

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

According to Tweedie's Formula, for a Gaussian variable $z \sim \mathbf{N}(z; \mu_z, \Sigma_z)$, we have the following conclusion:

$$\mu_z = z + \Sigma_z \nabla \log p(z) \tag{32}$$

Substituting it into Equation (32), we then have:

$$\sqrt{\overline{\alpha}_t}x_0 = x_t + (1-\overline{\alpha}_t)\nabla \log p(x_t) \tag{33}$$

Expanding Equation (32) yields $x_t = \sqrt{\overline{\alpha}_t}x_0 + \sqrt{1-\overline{\alpha}_t}\varepsilon_t$, so let's make an equivalent transformation for x_0:

$$\boldsymbol{x}_0 = \frac{\boldsymbol{x}_t + (1-\bar{\alpha}_t)\nabla \log\, p(\boldsymbol{x}_t)}{\sqrt{\bar{\alpha}_t}} = \frac{\boldsymbol{x}_t - \sqrt{1-\bar{\alpha}_t}\boldsymbol{\varepsilon}_t}{\sqrt{\bar{\alpha}_t}} \Rightarrow \nabla \log\, p(\boldsymbol{x}_t) = -\frac{\boldsymbol{\varepsilon}_t}{\sqrt{1-\bar{\alpha}_t}} \tag{34}$$

Similarly, when we derive the optimization objective in DDPM, we need to first model μ_q, and then use a neural network to learn and approximate it. Recall Equation (28) we derived above:

$$\mu_q = \frac{1}{\sqrt{\alpha_t}} x_t - \frac{1-\alpha_t}{\sqrt{1-\overline{\alpha}_t}\sqrt{\alpha_t}} \varepsilon_t \tag{35}$$

Using Equation (35) to replace ε_t allows us to obtain a new modeling approach:

$$\boldsymbol{\mu}_q = \frac{1}{\sqrt{\alpha_t}} \boldsymbol{x}_t + \frac{1-\alpha_t}{\sqrt{\alpha_t}} \nabla log\ p(\boldsymbol{x}_t) \tag{36}$$

Similarly, we can model the reverse estimation process as:

$$\boldsymbol{\mu}_\theta = \frac{1}{\sqrt{\alpha_t}} \boldsymbol{x}_t + \frac{1-\alpha_t}{\sqrt{\alpha_t}} s_\theta(\boldsymbol{x}_t, t) \tag{37}$$

Re-derive the optimization objective in DDPM, we will introduce the Score-based Generative Model. Returning to the estimation problem of the data distribution $p(x)$, any probability distribution can be represented in a more general form: $p_\theta(x) = \frac{e^{-f_\theta(x)}}{Z(\theta)}$.

This originates from Energy-Based Models. $Z(\theta) = \int^{e^{-f_\theta(x)}} dx$ represents a normalization constant, to ensure the p.d.f. integral $\int p_\theta(\boldsymbol{x}) d\boldsymbol{x}$ equals 1. The usual estimation method is Maximum Likelihood Estimation (MLE), but it necessitates considering the denominator first since it is unknown. How can the effect of this normalization term $Z(\theta)$ be eliminated? There are the following methods:

Normalizing flow restricts Z to 1. VAE and GAN do not eliminate Z but directly estimate it using some measure methods. VAE finds a lower bound of the evidence distance, while GAN directly uses neural networks to learn the distance metric.

Score-based Generative Model directly eliminates Z by taking the log gradient of p. The method of Score-based Generative Model is to take the log gradient of Equation (38):

$$\nabla log\ p_\theta(\boldsymbol{x}) = -\nabla f_\theta(\boldsymbol{x}) - \nabla log\ Z(\theta)\ \ldots (Z(\theta)\ \textit{is independent of}\ \boldsymbol{x}) = -\nabla f_\theta(\boldsymbol{x}) \tag{38}$$

And the optimization process can be directly approximated with a neural network $s_\theta(x)$, by sampling a variable from the real data $p(x)$, and then taking the log gradient, which provides an optimization direction:

$$\mathbb{E}_{p(\boldsymbol{x})}\left[\|s_\theta(\boldsymbol{x}) - \nabla log\ p(\boldsymbol{x})\|_2^2\right]. \tag{39}$$

In this way, every point in the data space is assigned a score vector, starting from an initial point and iterating continuously, finally reaching the region where the target data is located. Once we know this score (the gradient of $p(x)$ with respect to the input x), we can sample data using Langevin dynamics, the process is:

$$\boldsymbol{x}_{t+1} \leftarrow \boldsymbol{x}_t + \delta \nabla log\ p(\boldsymbol{x}_t) + \sqrt{2\delta}\varepsilon_t,\ t = 0,1,\cdots,T \tag{40}$$

Where x_0 is a random initial point; δ is a scale coefficient; $\varepsilon_t \sim \mathbf{N}(\mathbf{0}, \mathbf{I})$ is a perturbation term to prevent the sampling result from collapsing to a fixed point. When δ is sufficiently small and T is sufficiently large, the final x_T is the real data.

Can Equation (39) be optimized? No, because it requires knowing the true data's score function, which is obviously unattainable. However, other researchers have developed methods for optimization without knowing the true value as an alternative, collectively known as 'score matching'. Common methods include denoising score matching and sliced score matching. Although solvable, it still faces a problem, namely:

Samples in low-density areas are too few, making the estimates for these samples inaccurate, and the probability of certain sampling points may even be 0. Then taking the logarithm makes no sense, meaning the training effect will be poor.

How should this problem be solved? Here is a solution provided. Add Gaussian noise. This reduces the occurrence of zero probability, allowing more samples in low-density areas. Then, determining the size of the noise becomes the key issue. The current proposal is to add a series of noises of different

scales, and then train the score matching network of each scale step by step. The specific process is to introduce a series of noises $\sigma_{tt=1}^{T}$, and then the corresponding noise-perturbed distribution can be obtained:

$$p_{\sigma_t}(\boldsymbol{x}_t) = \int p(\boldsymbol{x})\mathcal{N}(\boldsymbol{x}_t; \boldsymbol{x}, \sigma_t^2 \boldsymbol{I})d\boldsymbol{x} \tag{41}$$

Here, a diagram is used to aid understanding. What we need to train are the arrows in the diagram. As $\sigma_1 < \sigma_2 < \sigma_3$ increases, the arrows learned become more and more refined. Then Equation (39) can be rewritten as:

$$arg\ \underset{\boldsymbol{\theta}}{min} \sum_{t=1}^{T} \lambda(t)\mathbb{E}_{p_{\sigma_t}(\boldsymbol{x})}\left[\left\|s_\theta(\boldsymbol{x}, t) - \nabla log\ p_{\sigma_t}(\boldsymbol{x})\right\|_2^2\right] \tag{42}$$

Where $\lambda(t)$ is a weight vector about t, which implements the training at different noise scales mentioned earlier. In this setting, Langevin dynamics sampling is also done on a scale basis, but in reverse, from $t = T$ to $t = T - 1$, and finally to $t = 1$. Intuitively, high noise provides better guidance in low-density areas, and low noise provides better guidance in high-density areas, making the entire sampling process more stable. This perspective better explains the effective principle of the Diffusion Model, as well as the necessity of the diffusion process and hierarchical noise addition.

What would happen if the aforementioned finite number of steps T were extended to an infinite number of steps? Because experimental verification has shown that the larger T is, the more accurate the likelihood estimate and the better the quality of the results. In this way, the perturbation of data in continuous time can be modeled as a stochastic differential equation (SDE). There are many forms of SDEs, and one form given by Dr. Song Yang in the paper is:

$$dx = f(x,t)dt + g(t)dw \tag{43}$$

Here $f(\cdot)$ is called the drift coefficient, $g(t)$ is called the diffusion coefficient, w is a standard Brownian motion, and $\mathrm{d}w$ can be viewed as a white noise. The solution to this stochastic differential equation is a set of continuous random variables $x(t)$, $t \in [0, T]$, where t makes the above discrete form $1,2,\ldots,T$ continuous. Meanwhile, $p_t(x)$ represents the probability density function of $x(t)$, which corresponds to the earlier $p\sigma_t(x_t)$, $p_0(x) = p(x)$ is the original data distribution, and $p_T(x) = \mathbf{N}(\mathbf{0}, \mathbf{I})$ is the final state after noise perturbation, turning into white noise.

Let's look at the connection between DDPM and SDE. Here we use the form of β, and to distinguish from the subscript t in the formulas below, we rewrite it as subscript i:

$$x_i = \sqrt{1-\beta_i}x_{i-1} + \sqrt{\beta_i}\varepsilon_{i-1}, \quad \varepsilon_{i-1} \sim N(0, I) \tag{44}$$

This is a discrete process; how can we make it continuous to get an SDE? The process from x_{i-1} to x_i involves sparsely adjacent variables, and we can make the sparse distribution denser by dividing the variable index by T (where T tends to infinity), thus changing the variable index from $i \in 0,1,\ldots,T$ to $t \in 0, \frac{1}{T}, \ldots, \frac{T-1}{T}, 1$. Then do some transfer.

$$x_{i-1} \to x'(t), \varepsilon_{i-1} \to \varepsilon'(t), x_i \to x'\left(t + \frac{1}{T}\right), \beta_i \to \frac{1}{T}\beta'\left(t + \frac{1}{T}\right) \tag{45}$$

In the above transformation, only the last β_i is scaled by a factor of N, this is to facilitate the derivation of Δt later on. At this moment, $\Delta t = \frac{1}{T}$, so now we can rewrite Equation (44):

$$\boldsymbol{x}'\left(t + \frac{1}{T}\right) = \sqrt{1 - \frac{1}{T}\beta'\left(t + \frac{1}{T}\right)}\boldsymbol{x}'(t) + \sqrt{\frac{1}{T}\beta'\left(t + \frac{1}{T}\right)}\varepsilon'(t) \tag{46}$$

$$(t + \Delta t) = \sqrt{1 - \beta(t + \Delta t)\Delta t}\boldsymbol{x}(t) + \sqrt{\beta(t + \Delta t)\Delta t}\varepsilon(t) \tag{47}$$

Using the conclusion of the first-order Taylor expansion $\sqrt{1-x} \approx 1 - \frac{x}{2}$, Equation (46) can be written as:

$$\begin{aligned} \boldsymbol{x}(t+\Delta t) &\approx \left[1-\frac{\beta(t+\Delta t)\Delta t}{2}\right]\boldsymbol{x}(t)+\sqrt{\beta(t+\Delta t)\Delta t}\boldsymbol{\varepsilon}(t) \\ d\boldsymbol{x}(t) &\approx -\frac{\beta(t)\Delta t}{2}\boldsymbol{x}(t)+\sqrt{\beta(t)\Delta t}\boldsymbol{\varepsilon}(t) \\ d\boldsymbol{x}(t) &\approx -\frac{\beta(t)x(t)}{2}\,dt+\sqrt{\beta(t)}\sqrt{dt}\varepsilon(t) \end{aligned} \tag{48}$$

Thus, we have derived a result in the form of Equation (43):

$$d\boldsymbol{x}=-\frac{\beta(t)x}{2}d\boldsymbol{t}+\sqrt{\beta(t)}d\boldsymbol{w} \tag{49}$$

In the previous discrete process, we used Langevin dynamics to generate new samples. Now we have derived the forward process of the SDE, but to generate samples, we need the reverse process. Is there a Reverse SDE? Yes! The reverse process of an SDE is still an SDE, and its expression is as follows:

$$d\boldsymbol{x}=[\boldsymbol{f}(\boldsymbol{x},t)-g(t)^2\nabla log\ p_t(\boldsymbol{x})]dt+g(t)d\boldsymbol{w} \tag{50}$$

The solution to the above equation yields a score-based generative model. The part marked in red is the score function we discussed earlier, which means the task is indeed to estimate this score function, consistently training a network to approximate $f_\theta(x,t)\approx\nabla\log p_t(x)$. Rewriting Equation (39) under the continuous process:

$$arg\min_{\boldsymbol{\theta}}\mathbb{E}_{t\in U(0,T)}\mathbb{E}_{p_t(\boldsymbol{x})}\left[\lambda(t)\|s_\theta(\boldsymbol{x},t)-\nabla log\ p_t(\boldsymbol{x})\|_2^2\right] \tag{51}$$

Regarding ordinary differential equations (ODE), the original paper demonstrates the transformation from SDE to ODE. On one hand, we can utilize more ODE solvers to aid in faster solving (since ODEs lack randomness, larger step sizes can be used for solving), and on the other hand, they can help us calculate a more accurate log-likelihood. The ODE corresponding to Equation (50) is shown as follows:

$$d\boldsymbol{x}=\left[f(x,t)-\frac{1}{2}g(t)^2\nabla\log p_t(x)\right]d\boldsymbol{t} \tag{52}$$

From the perspective of DDPM, it's difficult to derive the form of conditional probability, which complicates representing conditional generation. From the perspective of SDE, this issue can be effectively addressed. By employing Bayes' theorem:

$$p(x|y)=\frac{p(x)p(y|x)}{p(y)} \tag{53}$$

Taking the partial derivative with respect to x on both sides' yields:

$$\nabla\log p(x|y)=\nabla\log p(x)+\nabla\log p(y|x) \tag{54}$$

The last two terms are score functions that we can estimate, so we can generate $p(x|y)$ through Langevin-MCMC. To add here: $p_t(x)$ or $p(x_t)$ represents the marginal distribution at time t.

$$p(x_t)=\int p(x_t\mid x_0)p(x_0)dx_0=\mathbb{E}_{x_0}[p(x_t\mid x_0)] \tag{55}$$

Then

$$\begin{aligned} \nabla_{x_t}log\ p(x_t) &= \frac{\nabla_{x_t}p(x_t)}{p(x_t)}=\frac{\nabla_{x_t}\mathbb{E}_{x_0}[p(x_t|x_0)]}{\mathbb{E}_{x_0}[p(x_t|x_0)]} \\ &= \frac{\mathbb{E}_{x_0}[p(x_t|x_0)\nabla_{x_t}log\ p(x_t|x_0)]}{\mathbb{E}_{x_0}[p(x_t|x_0)]} \end{aligned} \tag{56}$$

The goal is to transform $\nabla_{x_t}\log p(x_t)\to\nabla_{x_t}\log p(x_t|x_0)$. Through derivation, it is found to be a relationship of weighted average, which can also be omitted without affecting the final optimization. Therefore, the neural network is essentially minimizing $\left|s_\theta(x_t,t)-\nabla_{x_t}\log p(x_t|x_0)\right|^2$.

4. Conclusion

This article has offered a thorough examination of diffusion models from multiple perspectives. It starts with an autonomous introduction to the three key formulations: DDPMs, SGMs and Score SDEs. The discussion then shifts to recent advancements aimed at enhancing diffusion models, emphasizing three principal trends: enhancing sampling efficiency, maximizing likelihood, and developing novel approaches for data with unique structures. Additionally, the article delves into the relationships between diffusion models and other types of generative models. The varied applications presented across numerous fields showcase the broad applicability and potential of diffusion models.

References

[1] Ho J, Jain A, Abbeel P. Denoising Diffusion Probabilistic Models. In Proceedings of the 34th International Conference on Neural Information Processing Systems, 2020.

[2] Ho J, et al. Improved Denoising Diffusion Probabilistic Models. Working paper, 2022.

[3] Song Y, Ermon S. Diffusion Models: A Comprehensive Survey of Methods and Applications. Working paper, 2021.

[4] Sohl-Dickstein J, Weiss E, Maheswaran than N, Ganguli S. Deep Unsupervised Learning using Nonequilibrium Thermodynamics. In Proceedings of the 32nd International Conference on Machine Learning, 2015.

[5] Kingma D P, Welling M. Auto-Encoding Variational Bayes. In Proceedings of the 2nd International Conference on Learning Representations (ICLR), 2013.

[6] Goodfellow I, et al. Generative Adversarial Nets. In Advances in Neural Information Processing Systems, 2014, 2672 - 2680.

[7] Vaswani A, et al. Attention is All You Need. In Advances in Neural Information Processing Systems, 2017, 30: 5998 - 6008.

[8] Nichol, A. Q., et al. GLIDE: Towards Photorealistic Image Generation and Editing with Text-Guided Diffusion Models. Working paper, 2021.

[9] Robin Rombach, Andreas Blattmann, Dominik Lorenz, Patrick Esser, and Björn Ommer. High-resolution image synthesis with latent diffusion models. In IEEE Conference on Computer Vision and Pattern Recognition, 2022, 10684 - 10695.

[10] Levon Khachatryan, Andranik Movsisyan, Vahram Tadevosyan, Roberto Henschel, Zhangyang Wang, Shant Navasardyan, Humphrey Shi. Text2video-zero: Text-to-image diffusion models are zero-shot video generators. Working paper, 2023.