Topics on Diffusion Generative Models

Original Text in Chinese by Jianlin Su[1] and Translated and Annotated by Jack Yang[2]
[1] *https://kexue.fm*
[2] *School of Materials Science and Engineering, University of New South Wales, Sydney, New South Wales 2052, Australia*[a]

(Dated: 27 March 2025)

Diffusion generative models, originally developed for high-quality image synthesis, have gradually gained popularity in other scientific research domains, including materials science for innovative materials discoveries. Compared with other deep learning models, however, diffusion models are much more mathematically involved, thus more challenging to be understood by novelists. Nevertheless, the ideal of diffusion generative models have strong link with the diffusion processes, which can be described by stochastic diffusion equations and are very known across many fields of fundamental science. In light of these, the translator (JY) sees huge potential in the applications of both the theoretical frameworks and the numerical implementations of the diffusion generative models in the future. Throughout the past few years, the original author (JS) has written a series of blog articles on diffusion generative models in Chinese, providing a more approachable understanding on the mathematical motivations and derivations behind this family of generative models, of which JY believes an English translated version will bring greater benefit to the large community who are interested in utilising these models in their research and development. This document is, therefore, prepared for this purpose while JY was studying JS's blog articles. Some further annotations and notes are added by JY to facilitate the understanding of the original materials.

[a] Electronic mail: jianliang.yang1@unsw.edu.au

Contents

1. DDPM=Building Demolition and Rebuilding 2
1.1. A new starting point 2
1.2. Demolition and rebuilding 3
1.3. How should we demolish the building? 3
1.4. How to rebuild the building? 4
1.5. Reducing the square error 4
1.6. Regressive generations 5
1.7. Hyperparameter settings 6

2. DDPM=Autoregressive VAE 6
2.1. Solving the problem step-wise 6
2.2. The joint KL Divergence 7
2.3. Divide-and-conquer 7
2.4. Reconstructing the diffusion model 8
2.5. Hyperparameter settings 8

1. DDPM=Building Demolition and Rebuilding

Generative Adversarial Networks (GAN) and Variational Autoencoders (VAE) are some of the well-known generative models in the deep learning community. Some of the less popular models, including the flow models[1], VQ-VAE[2] and so on, have also become increasingly attractive in the past few years. This is particularly true for VQ-VAE and its variant VQ-GAN, which have risen to become the image tokeniser, that can be adapted to (pre-)train the NLP models. Apart from these models, an even less known choice, namely, the **diffusion model**, has become increasingly popular in the area of artificial image generations. The two most successful text-to-image generation models to date, DALL-E2 from OpenAI and Imagen from Google, are both diffusion generative models.

Original blog post see https://www.spaces.ac.cn/archives/9119

June 2022, at the time of writing.

Hereafter, we start to discuss the recent progresses in the development of the diffusion model. Generally, it was believed that the mathematical derivations behind the diffusion models were very complex thus lacking intuitions. Here, we shall try to unpick these complex mathematical derivations, aiming at obtaining an intuitively clearer understanding behind the diffusion model.

1.1. A new starting point

Generally, when discussing the diffusion model, most literatures will refer to concepts such as **the energy-based model**, **score-matching**, **Langevin dynamics**, and so on. In a simple word, what diffusion model is trying to do is to *utilise the technique of score-matching to train an energy model, and then sample new items from this trained model based on the Langevin equation.*

Theoretically, this is a very mature proposal, whereby we can, in principle, apply it to generate and sample any continuous objects such as images and audio. Practically, however, it is rather difficult to train an energy function, especially when the dimensionality of the data is very high (*e.g.* high-resolution images). The main challenge is to obtain a fully converged energy function. On the other hand, there exist lots of uncertainties when we perform samplings based on the energy model using the Langevin equation, which results in noisy samples. As such, for a long time, the traditional ideas behind the diffusion model had only been experimented on generating low-resolution images.

In statistical thermodynamics, it means we need to sample all possible states $\{\boldsymbol{x}_i\}$ to obtain converged configurational partition function $\mathcal{Z}=\sum_{i=1}^{N}\exp[-U(\boldsymbol{x}_i)/k_BT]$, where $U(\boldsymbol{x})$ is the energy function, from which we can then gain access to the probability of individual sample $P(\boldsymbol{x}_i)=\exp[-U(\boldsymbol{x}_i)/k_BT]/\mathcal{Z}$. However, it is generally impossible to obtain the converged $\mathcal{Z}$ through sampling!

The popularity of modern diffusion generative models started from the **DDPM (Denoising Diffusion Probabilistic Model)**[3] first proposed in 2020. Although it had been named as a diffusion model, the formalism of DDPM is completely different from the traditional diffusion model that was based on the Langevin equation. More precisely, this new type of diffusion model should really be named as **models of progressive changes**, whereas calling it the diffusion model can actually cause lots of misunderstanding on the essence of this model. The concepts in the traditional diffusion model, such as the energy function, score matching and Langevin dynamics actually *have nothing to do with* DDPM and its subsequent variants. Interestingly, before 2020, the fundamental mathematical framework for DDPM had already been published in an ICML2015 article[4] entitled "*Deep Unsupervised Learning Using Nonequilibrium Thermodynamics*", but the model has only become popular after its capacity in generating high-resolution images was demonstrated. This shows that the birth and popularisation of a new model require both time and opportunity.

1.2. Demolition and rebuilding

When introducing the DDPM, many literatures adapt the probabilistic interpretation and dive straight into the change in the probability distribution that is followed by variational inference, which scares many readers with the mathematical equations. This is further compounded by the perceptions from the traditional diffusion models, thinking that the maths involved are very complicated. As a matter of fact, DDPM can be more intuitively explained, in a way that is no more complicated than the idea of 'fake it and then discriminate it' behind GAN (Generative Adversarial Network).

Although, on the other hand, this proves that DDPM is a variational autoencoder (VAE) rather than a diffusion model.

Firstly, let's think of a GAN-like model, it is actually a model that converts a random noise $\boldsymbol{z}$ into an actual sample $\boldsymbol{x}$, as shown in Fig. 1.

$$\begin{tikzcd}
\text{Random noise } \mathbf{z} \arrow[r, "\text{convert}"] \arrow[d, "\text{equivalent to}"] & \text{Sample } \mathbf{x} \arrow[d, "\text{equivalent to}"] \\
\text{Bricks and mortar} \arrow[r, "\text{build}"] & \text{Building}
\end{tikzcd}$$

FIG. 1. Building construction analogy to the DDPM model.

We can rethink of the process as a building process, where the noises $\boldsymbol{z}$ are the raw materials, in this case, the bricks and mortar, and the samples $\boldsymbol{x}$ are the final buildings being built. In this case, the generative model becomes the construction team.

The reconstruction process is bound to be difficult, and this is why there are so many researches involved in the development of generative models. On the other hand, the demolition process is very easy. Considering a **step-wise process** that gradually demolishing a building into bricks, in this case, let $\boldsymbol{x}_0$ be the fully-constructed building (initial sample) and $\boldsymbol{x}_T$ being the fully deconstructed bricks (random noises). Assuming the demolition takes T steps, then the whole demolition process can be written as

$$\boldsymbol{x} = \boldsymbol{x}_0 \to \boldsymbol{x}_1 \to \boldsymbol{x}_2 \to \cdots \boldsymbol{x}_{T-1} \to \boldsymbol{x}_T = \boldsymbol{z}. \tag{1.1}$$

Within this framework, the difficulties in rebuilding a building is that, the jump from the raw materials $\boldsymbol{x}_T$ to the final building $\boldsymbol{x}_0$ is too large for a novelist to comprehend how this process can be achieved. However, if we have the intermediate states from the demolition processes $\boldsymbol{x}_1$, $\boldsymbol{x}_2$ till $\boldsymbol{x}_{T-1}$, and we know $\boldsymbol{x}_{t-1} \to \boldsymbol{x}_t$ represents one small demolition step, then why can't we utilise these information to learn the reverse process $\boldsymbol{x}_t \to \boldsymbol{x}_{t-1}$ to rebuild the building? This means that if we can learn the relationship between the two $\boldsymbol{\mu}(\boldsymbol{x}_t) = \boldsymbol{x}_{t-1}$, then starting from $\boldsymbol{x}_T$, by repetitively applying $\boldsymbol{x}_{T-1} = \boldsymbol{\mu}(\boldsymbol{x}_T)$, $\boldsymbol{x}_{T-2} = \boldsymbol{\mu}(\boldsymbol{x}_{T-1})$ and so on, wouldn't we be able to rebuild the whole building $\boldsymbol{x}_0$?

It is important to take a note of this at this point, as this is a recurrent trick in the diffusion generative model that tries to learn the reverse construction process. This can be parameterised by any major NN architetures, including convolutional NN, transformer, graph and message-passing NNs for learning atomistic structures. In this regard, diffusion model has very little to do with the designs of the NN but more with the manipulations in the probability distributions of the data.

1.3. How should we demolish the building?

Just like we need to eat our meals bite by bite, we also need to rebuild our buildings stepwise. The process of generation in DDPM is actually fully consistent with the the **analogy of "demolition followed by reconstruction"** for DDPM. It is a process that first gradually deconstruct the data samples into random noises, then considers a reverse process which is carried out repetitively to (re)generated the (original) new samples. As mentioned before, this is why DDPM is more like a model of **progressive change** rather than a diffusion model.

More specifically, the demolition process in DDPM takes the following form:

$$\boldsymbol{x}_t = \alpha_t \boldsymbol{x}_{t-1} + \beta_t \boldsymbol{\varepsilon}_t, \quad \text{with} \quad \boldsymbol{\varepsilon}_t \sim \mathcal{N}(0, \boldsymbol{I}), \tag{1.2}$$

in particular, α_t and $\beta_t > 0$, $\alpha_t^2 + \beta_t^2 = 1$. Usually, we chose $\beta_t \to 0$, it represents how much damage we will make to the building during each step of the demolition process. The introduction of the **noise** $\boldsymbol{\varepsilon}_t$ represents the destruction to the original signal. We can also think of it as the raw materials, such that in each step of the demolition process, we break $\boldsymbol{x}_{t-1}$ into $\alpha_t \boldsymbol{x}_{t-1}$ part of the original building plus $\beta_t \boldsymbol{\varepsilon}_t$ part of the raw materials.

If we repeat the above process, then we have

$$\begin{aligned}
\boldsymbol{x}_t &= \alpha_t \boldsymbol{x}_{t-1} + \beta_t \boldsymbol{\varepsilon}_t \\
&= \alpha_t \left(\alpha_{t-1} \boldsymbol{x}_{t-2} + \beta_{t-1} \boldsymbol{\varepsilon}_{t-1}\right) + \beta_t \boldsymbol{\varepsilon}_t \\
&= \cdots \\
&= (\alpha_t \alpha_{t-1} \cdots \alpha_1) \boldsymbol{x}_0 + (\alpha_t \alpha_{t-1} \cdots \alpha_2) \beta_1 \boldsymbol{\varepsilon}_1 + (\alpha_t \cdots \alpha_3) \beta_2 \boldsymbol{\varepsilon}_2 + \cdots + \beta_t \boldsymbol{\varepsilon}_t.
\end{aligned} \tag{1.3}$$

This part is the sum of multiple independent Gaussian noises.

So why do we require the coefficients must satisfy the constraint $\alpha_t^2+\beta_t^2=1$? Let us try to answer this question. In Eq. (1.3), the part that is highlighted in red represents the sum of multiple independent Gaussian noises with the mean of zero, and variances being $(\alpha_t\alpha_{t-1}\cdots\alpha_2)\beta_1$, $(\alpha_t\cdots\alpha_3)\beta_2$, …, $\alpha_t\beta_{t-1}^2$ and β_t^2, respectively. Then, based on the superposition of Gaussians from the probability theory, the sum of Gaussians returns a new Gaussian, which, in the case is of mean zero and variance given by $(\alpha_t\alpha_{t-1}\cdots\alpha_2)\beta_1+(\alpha_t\cdots\alpha_3)\beta_2+\cdots+\beta_t$. Providing that $\alpha_t^2+\beta_t^2=1$, it is not difficult to prove that

$$(\alpha_t\alpha_{t-1}\cdots\alpha_2)\beta_1+(\alpha_t\cdots\alpha_3)\beta_2+\cdots+\beta_t=1. \tag{1.4}$$

This means that, effectively, we have

$$\boldsymbol{x}_t=\underbrace{(\alpha_t\cdots\alpha_1)}_{\equiv\bar{\alpha}_t}\boldsymbol{x}_0+\underbrace{\sqrt{1-(\alpha_t\cdots\alpha_1)^2}}_{\equiv\bar{\beta}_t}\bar{\boldsymbol{\varepsilon}}_t, \qquad \bar{\boldsymbol{\varepsilon}}_t\sim\mathcal{N}(0,\boldsymbol{I}), \tag{1.5}$$

Take note of the definitions $\bar{\alpha}_t=\prod_{i=1}^t\alpha_i$ and $\bar{\beta}_t=\sqrt{1-\bar{\alpha}_t^2}$, which are commonly used in the diffusion models.

which has now significantly simplified the computations of $\boldsymbol{x}_t$. On the other hand, by choosing a suitable form of α_t in DDPM to ensure $\bar{\alpha}_t\approx 0$ will guarantee that, after T steps of demolitions, the entire building will be fully deconstructed into its raw materials $\boldsymbol{\varepsilon}$.

1.4. How to rebuild the building?

During the demolition process $\boldsymbol{x}_{t-1}\to\boldsymbol{x}_t$, we can obtain many pairs of data $(\boldsymbol{x}_{t-1},\boldsymbol{x}_t)$, therefore, rebuilding is simply a process that learns a model to enable the reverse process $\boldsymbol{x}_t\to\boldsymbol{x}_{t-1}$ to occur. Suppose this model is $\boldsymbol{\mu}(\boldsymbol{x}_t)$, then the most straightforward learning protocol is to minimise the Euclidean distance

$$\|\boldsymbol{\mu}(\boldsymbol{x}_t)-\boldsymbol{x}_{t-1}\|_2^2. \tag{1.6}$$

This simple formulation is actually very close to the final DDPM model. Here, we can refine it further. If we first rearrange Eq. (1.2) from the demolition process into a reconstructing one by making $\boldsymbol{x}_{t-1}$ the subject:

$$\boldsymbol{x}_{t-1}=\frac{1}{\alpha_t}\left(\boldsymbol{x}_t-\beta_t\boldsymbol{\varepsilon}_t\right),$$

this then inspires us to express the parameterised reconstruction process analogously as

$$\boldsymbol{\mu}(\boldsymbol{x}_t)=\frac{1}{\alpha_t}\left(\boldsymbol{x}_t-\beta_t\boldsymbol{\varepsilon}_\theta(\boldsymbol{x}_t,t)\right), \tag{1.7}$$

in which θ is the learnable parameter. Substituting Eq. (1.7) into the loss function Eq. (1.6), we get

$$\|\boldsymbol{\mu}(\boldsymbol{x}_t)-\boldsymbol{x}_{t-1}\|_2^2=\frac{\beta_t^2}{\alpha_t^2}\|\boldsymbol{\varepsilon}_t-\boldsymbol{\varepsilon}_\theta(\boldsymbol{x}_t,t)\|_2^2. \tag{1.8}$$

Here, β_t^2/α_t^2 represents the weight of the loss function, which we can ignore it for the time being. Substituting Eq. (1.2) and Eq. (1.5) for the expression of $\boldsymbol{x}_t$:

$$\boldsymbol{x}_t=\alpha_t\boldsymbol{x}_{t-1}+\beta_t\boldsymbol{\varepsilon}_t=\alpha_t(\bar{\alpha}_{t-1}\boldsymbol{x}_0+\bar{\beta}_{t-1}\bar{\boldsymbol{\varepsilon}}_{t-1})+\beta_t\boldsymbol{\varepsilon}_t=\alpha_t\boldsymbol{x}_0+\alpha_t\bar{\beta}_{t-1}\bar{\boldsymbol{\varepsilon}}_{t-1}+\beta_t\boldsymbol{\varepsilon}_t, \tag{1.9}$$

from which we arrive at a new form of the loss function:

$$\|\boldsymbol{\varepsilon}_t-\boldsymbol{\varepsilon}_\theta(\alpha_t\boldsymbol{x}_0+\alpha_t\bar{\beta}_{t-1}\bar{\boldsymbol{\varepsilon}}_{t-1}+\beta_t\boldsymbol{\varepsilon}_t,t)\|_2^2. \tag{1.10}$$

Some readers may wonder why, in Eq. (1.9), we have to go back on step to $t-1$, rather than just go from t to 0 directly? The answer is that, since we already sampled $\boldsymbol{\varepsilon}_t$ in Eq. (1.8), and $\bar{\boldsymbol{\varepsilon}}_t$ is not independent from $\boldsymbol{\varepsilon}_t$, we cannot sample $\bar{\boldsymbol{\varepsilon}}_t$ completely as an independent variable, in which case we end up minimising towards a target that is sample dependent.

This process of stepping back one more step will occur recurrently in many derivations in the subsequent chapters.

1.5. Reducing the square error

In principle, we could train DDPM directly using the loss function Eq. (1.10). But in real practice, this is not very tractable, which can lead to slow convergence. This is because Eq. (1.10) involves **four** random variables that can only be obtained through samplings:

1. A random sample $\boldsymbol{x}_0$ from the inputs.
2. Sample $\bar{\boldsymbol{\varepsilon}}_{t-1}$ and $\bar{\boldsymbol{\varepsilon}}_t$ from a normal distribution $\mathcal{N}(0, \boldsymbol{I})$, giving extra two variables.
3. Choosing a t from 1 to T.

The more we need to sample, the more challenge it is to compute the loss function accurately! Fortunately, this problem can be partially mitigated by combining $\boldsymbol{\varepsilon}_t$ and $\bar{\boldsymbol{\varepsilon}}_{t-1}$ into a single random variable sampled from a normal distribution using the integration trick. Because the product of two normal distributions is still a normal distribution, so we have $\alpha_t\bar{\beta}_{t-1}\bar{\boldsymbol{\varepsilon}}_{t-1}+\beta_t\boldsymbol{\varepsilon}_t$ is equivalent to $\bar{\beta}_t\boldsymbol{\varepsilon}|\boldsymbol{\varepsilon} \sim \mathcal{N}(0, I)$. Equally, we have $\beta_t\bar{\boldsymbol{\varepsilon}}_{t-1} - \alpha_t\bar{\beta}_{t-1}\boldsymbol{\varepsilon}_t$ that is equivalent to $\bar{\beta}_t\boldsymbol{w}|\boldsymbol{w} \sim \mathcal{N}(0, I)$. Furthermore, it can be proven that $\mathbb{E}[\boldsymbol{\varepsilon}\boldsymbol{w}^T] = 0$, so they are two independent normal distributions. With this, we can express $\boldsymbol{\varepsilon}_t$ with $\boldsymbol{\varepsilon}$ and $\boldsymbol{w}$:

Not quite sure where this comes from?

$$\boldsymbol{\varepsilon}_t = \frac{(\beta_t\boldsymbol{\varepsilon} - \alpha_t\bar{\beta}_{t-1}\boldsymbol{w})\bar{\beta}_t}{\beta_t^2 + \alpha_t^2\bar{\beta}_{t-1}^2} = \frac{\beta_t\boldsymbol{\varepsilon} - \alpha_t\bar{\beta}_{t-1}\boldsymbol{w}}{\bar{\beta}_t}. \tag{1.11}$$

Substitute Eq. (1.11) into Eq. (1.10), we get

$$\begin{aligned}&\mathbb{E}_{\bar{\boldsymbol{\varepsilon}}_{t-1},\boldsymbol{\varepsilon}_t\sim\mathcal{N}(0,\boldsymbol{I})}\left[\left\|\boldsymbol{\varepsilon}_t - \boldsymbol{\varepsilon}_\theta(\alpha_t\boldsymbol{x}_0 + \alpha_t\bar{\beta}_{t-1}\bar{\boldsymbol{\varepsilon}}_{t-1} + \beta_t\boldsymbol{\varepsilon}_t, t)\right\|_2^2\right]\\ &= \mathbb{E}_{\boldsymbol{w},\boldsymbol{\varepsilon}\sim\mathcal{N}(0,\boldsymbol{I})}\left[\left\|\frac{\beta_t\boldsymbol{\varepsilon} - \alpha_t\bar{\beta}_{t-1}\boldsymbol{w}}{\bar{\beta}_t} - \boldsymbol{\varepsilon}_\theta(\bar{\alpha}_t\boldsymbol{x}_0 + \bar{\beta}_t\boldsymbol{\varepsilon}, t)\right\|_2^2\right].\end{aligned} \tag{1.12}$$

Note that the loss function is quadratic in $\boldsymbol{w}$, which we can expand the square and then determine the expectation value, this gives,

$$\frac{\beta_t^2}{\bar{\beta}_t^2}\mathbb{E}_{\boldsymbol{\varepsilon}\sim\mathcal{N}(0,\boldsymbol{I})}\left[\left\|\boldsymbol{\varepsilon} - \frac{\bar{\beta}_t}{\beta_t}\boldsymbol{\varepsilon}_\theta(\bar{\alpha}_t\boldsymbol{x}_0 + \bar{\beta}_t\boldsymbol{\varepsilon}, t)\right\|_2^2\right] + \mathcal{C}. \tag{1.13}$$

Ignoring the weight and constant again, we arrive at:

$$\mathbb{E}_{\boldsymbol{\varepsilon}\sim\mathcal{N}(0,\boldsymbol{I})}\left[\left\|\boldsymbol{\varepsilon} - \frac{\bar{\beta}_t}{\beta_t}\boldsymbol{\varepsilon}_\theta(\bar{\alpha}_t\boldsymbol{x}_0 + \bar{\beta}_t\boldsymbol{\varepsilon}, t)\right\|_2^2\right], \tag{1.14}$$

This basically says, we must first sample a noise from the normal distribution $\mathcal{N}(0, \boldsymbol{I})$, from which we can generate the noisy sample $\boldsymbol{x}_t = \bar{\alpha}_t\boldsymbol{x}_0 + \bar{\beta}_t\boldsymbol{\varepsilon}$. Then we plug this $\boldsymbol{x}_t$ into the denoising network $\boldsymbol{\varepsilon}_\theta(\boldsymbol{x}_t)$ to recover $\boldsymbol{\varepsilon}_t$. This expression is equivalent to the results[5] derived using KL divergence between the destruction and regeneration process: $\text{argmin}_\theta\, \mathcal{D}_{KL}(q(\boldsymbol{x}_{t-1}|\boldsymbol{x}_t, \boldsymbol{x}_0) \| p(\boldsymbol{x}_{t-1}|\boldsymbol{x}_t))$ from evidence lower bound, which upon further reparameterisation, gives the following expression for the KL divergence: $\frac{(1-\alpha_t)^2}{2\sigma_q^2(t)}\frac{\|\boldsymbol{\varepsilon}_0 - \hat{\boldsymbol{\varepsilon}}_\theta(\boldsymbol{x},t)\|_2^2}{(1-\bar{\alpha}_t)\alpha_t}$.

which is the final form of the lost function for DDPM.

1.6. Regressive generations

Above, we have clarified the training procedures for DDPM. The derivation is a bit lengthy and not immediately trivial. While it is not easy to understand, it is not totally difficult either, since it has not applied the mathematical tools that are needed in the traditional diffusion models, but only an analogous model of building demolition and reconstruction plus some basic knowledge in the probability theory. As such, this newly formulated diffusion generated model, as represented by DDPM, is not as mathematically complicated as one may think.

After the model is being trained, we can perform samplings to generate new samples. Starting from a random noise $\boldsymbol{x}_T \sim \mathcal{N}(0, \boldsymbol{I})$, and repeat the process for T steps according to Eq. (1.7):

$$\boldsymbol{x}_{t-1} = \frac{1}{\alpha_t}(\boldsymbol{x}_t - \beta_t\boldsymbol{\varepsilon}_\theta(\boldsymbol{x}_t, t)). \tag{1.15}$$

This corresponds to the **greedy search** in VAE. For random sampling, we can add an additional noise term:

$$\boldsymbol{x}_{t-1} = \frac{1}{\alpha_t}(\boldsymbol{x}_t - \beta_t\boldsymbol{\varepsilon}_\theta(\boldsymbol{x}_t, t)) + \sigma_t\boldsymbol{z} \qquad \boldsymbol{z} \sim \mathcal{N}(0, \boldsymbol{I}). \tag{1.16}$$

In practice, we can keep $\beta_t = \sigma_t$, i.e., the variances are the same for both the forward and reverse diffusions.

It should be noted that the sampling process in DDPM is different from the one performed with Langevin equation. In DDPM, each sampling process starts from a new random noise, and is iterated for T steps to generate a new sample. On the other hand, Langevin sampling starts from a single random point and iterates definitively. In principle, all samples can be generated through infinite iterations. Therefore, although DDPM shares similarity with the Langevin sampling, they are fundamentally different models.

1.7. Hyperparameter settings

In DDPM, the upper limit T is usually set to 1000, which is much larger than what people usually expect. Why such a large value is necessary? In the original publication, the following function is used to set α_t:

$$\alpha_t = \sqrt{1 - \frac{0.02t}{T}}, \tag{1.17}$$

which is a monotonically decreasing function. Why such a choice is also necessary?

The above two questions are inter-dependent, and is related to the nature of the data. When the Euclidean distance is used for constructing the loss function, experience of image reconstructions using VAE shows that, only when the input and output images are sufficiently close to each other, we can generate new images of higher quality. Thus, using large time can make sure the input and output in each time step are sufficiently similar to each other, reducing the blurring effects from using the Euclidean distance in the loss function.

A similar reason applies to the choice of a monotonically decreasing function for α_t. When t is small, $\boldsymbol{x}_t$ is close to the real image, so we need to decrease the distance between $\boldsymbol{x}_{t-1}$ and $\boldsymbol{x}_t$, making it more suitable to apply the Euclidean distance in Eq. (1.6). This gives large α_t. When t is very large, xt becomes very close to a true Gaussian noise, which can be modelled with the Euclidean distance, thus we can increase the distance between $\boldsymbol{x}_{t-1}$ and $\boldsymbol{x}_t$ with smaller α_t. Can we use a large α_t throughout the forward process then? In principle the answer is yes, but we need to increase T. Recall that when we derived Eq. (1.5), we wanted $\bar{\alpha}_T \approx 0$, which we can evaluate

$$\log \bar{\alpha}_T = \frac{1}{2}\sum_{t=1}^{T} \log\left(1 - \frac{0.02t}{T}\right) < \frac{1}{2}\sum_{t=1}^{T} \log\left(-\frac{0.02t}{T}\right) = -0.005(T+1) \tag{1.18}$$

With $T = 1000$, $\bar{\alpha}_T \approx 10^{-5}$, which is nearly zero as we wished. As a result, if we want to use a large α_t throughout, then we will have to increase T.

Finally, note that in the loss function [Eq. (1.14), $\boldsymbol{\varepsilon}_\theta(\bar{\alpha}_t \boldsymbol{x}_0 + \bar{\beta}_t \boldsymbol{\varepsilon}, t)$], the time variable t has been explicitly included. This is because, at different t, we are dealing with data of different noise levels. Therefore, **we are required to have T different reconstruction models**, meaning **T is also a model hyperparameter**.

2. DDPM=Autoregressive VAE

In Section 1, we had established an analogy for the diffusion generative model DDPM, in which we described it as the process of demolition followed by reconstruction. Based on this analogy, we have completed a particular form of the mathematical derivations for the theoretical foundation of DDPM. We also pointed out that, fundamentally, DDPM is not the traditional diffusion model, but more like a VAE. In fact, in the seminal paper for DDPM[3], it was derived based on the VAE framework. Therefore, in this section, we will reintroduce the DDPM diffusion model based on the framework of VAE.

Original blog post see https://www.spaces.ac.cn/archives/9152

2.1. Solving the problem step-wise

In the standard formulation of the VAE, both the encoding and generative processes are completed in a single step:

In the language of VAE, $\boldsymbol{z}$ is usually referred to as the *latent variable.*

$$\text{ENCODER:} \quad \boldsymbol{x} \to \boldsymbol{z}; \quad \text{GENERATION:} \quad \boldsymbol{z} \to \boldsymbol{x}. \tag{2.1}$$

In this case, we only need to deal with three data distributions: the **encoding distribution** $p(\boldsymbol{z}|\boldsymbol{x})$, the **generative distribution** $q(\boldsymbol{x}|\boldsymbol{z})$, and the prior distribution $q(\boldsymbol{z})$. The advantage behind such a formulation is its simplicity. There also exists a simple projective relationship between the sample $\boldsymbol{x}$ and the latent variable $\boldsymbol{z}$, such that we can train and obtain both the encoder and the decoder (generator) simultaneously. it also allows us to manipulate the data in the latent space. However, VAE also has its notable limitations. Since all three distributions are usually modelled as the normal distributions, it has resulted in the limited expressive capability of the VAE model.

To overcome this limitation, DDPM breaks these processes into multiple steps:

$$\begin{aligned}&\text{ENCODER:} \quad \boldsymbol{x} = \boldsymbol{x}_0 \to \boldsymbol{x}_1 \to \boldsymbol{x}_2 \to \cdots \to \boldsymbol{x}_{T-1} \to \boldsymbol{x}_T = \boldsymbol{z} \\ &\text{DECODER:} \quad \boldsymbol{z} = \boldsymbol{x}_T \to \boldsymbol{x}_{T-1} \to \boldsymbol{x}_{T-2} \to \cdots \to \boldsymbol{x}_1 \to \boldsymbol{x}_0 = \boldsymbol{x}.\end{aligned} \tag{2.2}$$

In this way, every transition probability $p(\boldsymbol{x}_t|\boldsymbol{x}_{t-1})$ or $q(\boldsymbol{x}_{t-1}|\boldsymbol{x}_t)$ is only responsible for making a small change to the sample by the model. One may ask, if both p and q are normal distributions, why this alternative model will work better than the single-stepped one [Eq. (2.1)]? This is because, *Gaussian distributions work better in approximating small incremental changes*, which is similar to approximating a curve with linear relationship that will only work within a small region of $\Delta \boldsymbol{x}$. As such, theoretically, the formulation proposed in Eq. (2.2) is able to improve the performance of a single-shot VAE.

2.2. The joint KL Divergence

To arrive at a step-wise formulation for VAE, we have $p(\boldsymbol{x}_t|\boldsymbol{x}_{t-1})$ for every encoding step and $q(\boldsymbol{x}_{t-1}|\boldsymbol{x}_t)$ for every generation step. This allows us to write down the **joint probability distributions**:

In the following derivations, the convention behind VAE will be adopted, in which we will use p for the transition probability of the forward (encoding) process and q for the backward (decoding) process, which is **opposite** to the notations used in the DDPM literatures!

$$\begin{aligned}p(\boldsymbol{x}_0, \boldsymbol{x}_1, \boldsymbol{x}_2, \ldots, \boldsymbol{x}_T) &= p(\boldsymbol{x}_T|\boldsymbol{x}_{T-1}) \cdots p(\boldsymbol{x}_2|\boldsymbol{x}_1) p(\boldsymbol{x}_1|\boldsymbol{x}_0) \tilde{p}(\boldsymbol{x}_0) \\ q(\boldsymbol{x}_0, \boldsymbol{x}_1, \boldsymbol{x}_2, \ldots, \boldsymbol{x}_T) &= q(\boldsymbol{x}_0|\boldsymbol{x}_1) \cdots q(\boldsymbol{x}_{T-2}|\boldsymbol{x}_{T-1}) q(\boldsymbol{x}_{T-1}|\boldsymbol{x}_T) q(\boldsymbol{x}_T).\end{aligned} \tag{2.3}$$

Here, $\boldsymbol{x}_0$ represents the input samples, thus $\tilde{p}(\boldsymbol{x}_0)$ represents the sample distribution, whereas $q(\boldsymbol{x}_T)$ is the prior distribution, as $\boldsymbol{x}_T$ is the final encoded sample. The rest probability distributions of the type $p(\boldsymbol{x}_t|\boldsymbol{x}_{t-1})$ and $q(\boldsymbol{x}_{t-1}|\boldsymbol{x}_t)$ represent each small step in the encoding and generating processes, respectively.

The simplest way to understand VAE, is to minimise the KL divergence between the joint probability distributions, which is equally applicable to DDPM:

$$D_{KL}(p\|q) = \int p(\boldsymbol{x}_T|\boldsymbol{x}_{T-1}) \cdots p(\boldsymbol{x}_1|\boldsymbol{x}_0) \tilde{p}(\boldsymbol{x}_0) \log \frac{p(\boldsymbol{x}_T|\boldsymbol{x}_{T-1}) \cdots p(\boldsymbol{x}_1|\boldsymbol{x}_0) \tilde{p}(\boldsymbol{x}_0)}{q(\boldsymbol{x}_0|\boldsymbol{x}_1) \cdots q(\boldsymbol{x}_{T-1}|\boldsymbol{x}_T) q(\boldsymbol{x}_T)} d\boldsymbol{x}_0 \cdots d\boldsymbol{x}_T. \tag{2.4}$$

This is, therefore, the optimisation target for DDPM. What is left is to consolidate the forms of $p(\boldsymbol{x}_t|\boldsymbol{x}_{t-1})$ and $q(\boldsymbol{x}_{t-1}|\boldsymbol{x}_t)$ to simplify Eq. (2.4) further, making it possible to be computed numerically.

2.3. Divide-and-conquer

First of all, DDPM is only interested in the generating part of the process, therefore, in the encoder process, every step is modelled as a simple normal distribution:

$$p(\boldsymbol{x}_t|\boldsymbol{x}_{t-1}) = \mathcal{N}(\boldsymbol{x}_t; \alpha_t \boldsymbol{x}_{t-1}, \beta_t^2 \boldsymbol{I}).$$

This is an alternative way of writing Eq. (1.2)

This is much simpler than the VAE, the medium and variance of which are both learnt by a neural network. Here, the medium is basically $\boldsymbol{x}_{t-1}$ that is multiplied by a *scalar* α_t. Hence, DDPM has completely abandoned the capability to encode the data, and only established a generative model. The transition probability for the generating process $q(\boldsymbol{x}_{t-1}|\boldsymbol{x}_t)$ has now been formulated as a learnable model $\mathcal{N}(\boldsymbol{x}_{t-1}; \boldsymbol{\mu}(\boldsymbol{x}_t), \sigma_t^2 \boldsymbol{I})$. Here, **$\alpha_t$, β_t and σ_t are all not learnables but parameterised.** Hence, $\boldsymbol{\mu}(\boldsymbol{x}_t)$ is the only trainable object in the entire diffusion model.

Since p is not parameterised, the integration in Eq. (2.4) that involves purely the p distributions can be factored out as a constant. This makes Eq. (2.4) become equivalent to

$$\begin{aligned}&-\int p(\boldsymbol{x}_T|\boldsymbol{x}_{T-1}) \cdots p(\boldsymbol{x}_1|\boldsymbol{x}_0) \tilde{p}(\boldsymbol{x}_0) \underbrace{\log[q(\boldsymbol{x}_0|\boldsymbol{x}_1) \cdots q(\boldsymbol{x}_{T-1}|\boldsymbol{x}_T) q(\boldsymbol{x}_T)]}_{\text{make this into a summation}} d\boldsymbol{x}_0 \cdots d\boldsymbol{x}_T \\ =&-\int p(\boldsymbol{x}_T|\boldsymbol{x}_{T-1}) \cdots p(\boldsymbol{x}_1|\boldsymbol{x}_0) \tilde{p}(\boldsymbol{x}_0) \left[\log q(\boldsymbol{x}_T) + \sum_{t=1}^{T} \log q(\boldsymbol{x}_{t-1}|\boldsymbol{x}_t)\right] d\boldsymbol{x}_0 \cdots d\boldsymbol{x}_T.\end{aligned} \tag{2.5}$$

Since $q(\boldsymbol{x}_T)$ is usually modelled as a normal distribution, it is also parameter-free. As such, we only need to compute *every term* of the form

$$-\int \underbrace{p(\boldsymbol{x}_T|\boldsymbol{x}_{T-1})\cdots p(\boldsymbol{x}_1|\boldsymbol{x}_0)\tilde{p}(\boldsymbol{x}_0)}_{T+1 \text{ terms}} \log q(\boldsymbol{x}_{t-1}|\boldsymbol{x}_t) d\boldsymbol{x}_0 \cdots d\boldsymbol{x}_T$$

This integral ismaximally dependent up to $\boldsymbol{x}_t$, so all the integrations from $\boldsymbol{x}_{t+1}$ to $\boldsymbol{x}_T$ gives unity, allowing us to remove some terms involving p.

$$= -\int \underbrace{p(\boldsymbol{x}_t|\boldsymbol{x}_{t-1})\cdots p(\boldsymbol{x}_1|\boldsymbol{x}_0)\tilde{p}(\boldsymbol{x}_0)}_{t+1 \text{ terms}} \log q(\boldsymbol{x}_{t-1}|\boldsymbol{x}_t) d\boldsymbol{x}_0 \cdots d\boldsymbol{x}_t$$

$$= -\int p(\boldsymbol{x}_t|\boldsymbol{x}_{t-1}) p(\boldsymbol{x}_{t-1}|\boldsymbol{x}_0)\tilde{p}(\boldsymbol{x}_0) \log q(\boldsymbol{x}_{t-1}|\boldsymbol{x}_t) d\boldsymbol{x}_0 d\boldsymbol{x}_{t-1}\boldsymbol{x}_t. \quad (2.6)$$

Similarly, the integral is also independent from $x_1, \ldots, x_{t-2}$.

2.4. Reconstructing the diffusion model

What follows from here is basically the same as discussed in Section 1 1.4:

1. Remove all the constants that are irrelevant to the optimisation problem, this left us with the term $-\log q(\boldsymbol{x}_{t-1}|\boldsymbol{x}_t)$ which gives the term $\frac{1}{2\sigma_t^2}\|\boldsymbol{x}_{t-1} - \boldsymbol{\mu}(\boldsymbol{x}_t)\|_2^2$ in the target function for optimisation.
2. The transition probability $p(\boldsymbol{x}_{t-1}|\boldsymbol{x}_0)$ can be equivalently written as $\boldsymbol{x}_{t-1} = \bar{\alpha}_{t-1}\boldsymbol{x}_0 + \bar{\beta}_{t-1}\bar{\boldsymbol{\varepsilon}}_{t-1}$, similarly $p(\boldsymbol{x}_t|\boldsymbol{x}_{t-1})$ implies $\boldsymbol{x}_t = \alpha_t \boldsymbol{x}_{t-1} + \beta_t \boldsymbol{\varepsilon}_t$, in which $\bar{\boldsymbol{\varepsilon}}_{t-1}$ and $\boldsymbol{\varepsilon}_t \sim \mathcal{N}(0, \boldsymbol{I})$.
3. From $\boldsymbol{x}_{t-1} = \frac{1}{\alpha_t}(\boldsymbol{x}_t - \beta_t\boldsymbol{\varepsilon}_t)$, it inspires us to parameterise $\boldsymbol{\mu}(\boldsymbol{x}_t)$ as $\boldsymbol{\mu}(\boldsymbol{x}_t) = \frac{1}{\alpha_t}(\boldsymbol{x}_t - \beta_t\boldsymbol{\varepsilon}_\theta(\boldsymbol{x}_t, t))$.

Equipped with these transformations, the target function for optimisation can be rewritten in the following form:

This is equivalent to Eq. (1.10).

$$\frac{\beta_t^2}{\alpha_t^2\sigma_t^2}\mathbb{E}_{\bar{\boldsymbol{\varepsilon}}_{t-1},\boldsymbol{\varepsilon}_t\sim\mathcal{N}(0,\boldsymbol{I}),\boldsymbol{x}_0\sim\tilde{p}(\boldsymbol{x}_0)}\left[\left\|\boldsymbol{\varepsilon}_t - \boldsymbol{\varepsilon}_\theta(\bar{\alpha}_t\boldsymbol{x}_0 + \alpha_t\bar{\beta}_{t-1}\boldsymbol{\varepsilon}_{t-1} + \beta_t\boldsymbol{\varepsilon}_t, t)\right\|_2^2\right]. \quad (2.7)$$

This is followed by the trick of changing the variables applied in Section 1 1.5, which gives

This is equivalent to Eq. (1.14).

$$\frac{\beta_t^4}{\alpha_t^2\sigma_t^4}\mathbb{E}_{\boldsymbol{\varepsilon}\sim\mathcal{N}(0,\boldsymbol{I}),\boldsymbol{x}_0\sim\tilde{p}(\boldsymbol{x}_0)}\left[\left\|\boldsymbol{\varepsilon}_t - \frac{\bar{\beta}_t}{\beta_t}\boldsymbol{\varepsilon}_\theta(\bar{\alpha}_t\boldsymbol{x}_0 + \bar{\beta}_t\boldsymbol{\varepsilon}, t)\right\|_2^2\right] \quad (2.8)$$

This leads to the loss function for DDPM. **As tested in the original paper, the model performance is even better if we drop the coefficient $\frac{\beta_t^4}{\alpha_t^2\sigma_t^4}$ in the numerical implementation.** The above derivation starts from the loss function for VAE and sequentially simplifies the multivariant integral [Eq. (2.5)]. Although the derivation is longer, it is logically tractable. In comparison to the original DDPM paper[3], where a conditional probability $q(\boldsymbol{x}_{t-1}|\boldsymbol{x}_t, \boldsymbol{x}_0)$ was introduced which then split the terms that were subsequently cancelled out. This original derivation came somehow 'out of the blue', which may not be easily understood by many readers.

2.5. Hyperparameter settings

Now we move on to discuss the choice of the hyperparameters α_t, β_t and σ_t. For $p(\boldsymbol{x}_t|\boldsymbol{x}_{t-1})$, it is conventional to set $\alpha_t^2 + \beta_t^2 = 1$, which already allows us to reduce the number of hyperparameters by half, and helps to simplify the expressions. This has already been discussed in Section 1. From the superposition of normal distributions, we have the following formula one-shot sampling:

$$p(\boldsymbol{x}_t|\boldsymbol{x}_0) = \int p(\boldsymbol{x}_t|\boldsymbol{x}_{t-1})\cdots p(\boldsymbol{x}_1|\boldsymbol{x}_0) d\boldsymbol{x}_t d\boldsymbol{x}_{t-1}\cdots d\boldsymbol{x}_1 = \mathcal{N}(\boldsymbol{x}_t; \bar{\alpha}_t\boldsymbol{x}_0, \bar{\beta}_t^2\boldsymbol{I}), \quad (2.9)$$

where $\bar{\alpha}_t = \alpha_1\alpha_2\cdots\alpha_t$, and $\bar{\beta}_t = \sqrt{1-\bar{\alpha}_t^2}$.

So what inspires us to set $\alpha_t^2 + \beta_t^2 = 1$? The choice of $\mathcal{N}(\boldsymbol{x}_t; \bar{\alpha}_t\boldsymbol{x}_0, \bar{\beta}_t^2\boldsymbol{I})$ means we have $\boldsymbol{x}_t = \alpha_t\boldsymbol{x}_{t-1} + \beta_t\boldsymbol{\varepsilon}_t$ with $\boldsymbol{\varepsilon}_t \sim \mathcal{N}(0, \boldsymbol{I})$, then it would require $\alpha_t^2 + \beta_t^2 = 1$ based on the normalisation condition for the linear combinations of normal distributions.

As mentioned before, we usually represent $q(\boldsymbol{x}_T)$ as a standard normal distribution

$\mathcal{N}(\boldsymbol{x}_T; 0, \boldsymbol{I})$, and the training goal is to minimise the KL divergence of the two joint probability distributions, i.e. $p = q$. This means that the marginal distributions should also equal each other:

$$q(\boldsymbol{x}_T) = \int p(\boldsymbol{x}_T|\boldsymbol{x}_{T-1})p(\boldsymbol{x}_{T-1}|\boldsymbol{x}_{T-2})\cdots p(\boldsymbol{x}_1|\boldsymbol{x}_0)\tilde{p}(\boldsymbol{x}_0)d\boldsymbol{x}_0 d\boldsymbol{x}_1\cdots d\boldsymbol{x}_{T-1}$$
$$= \int p(\boldsymbol{x}_T|\boldsymbol{x}_0)\tilde{p}(\boldsymbol{x}_0)d\boldsymbol{x}_0. \tag{2.10}$$

In general, the sample distribution is arbitrary, such that the only way to make Eq. (2.10) hold is to equate $p(\boldsymbol{x}_T|\boldsymbol{x}_0)$ to become $q(\boldsymbol{x}_T)$ [Note that the integral in Eq. (2.10) is independent of $\boldsymbol{x}_T$!]. **It means to evolve $\tilde{p}(\boldsymbol{x}_0)$ into a Gaussian distribution that is independent from $\boldsymbol{x}_0$.** Recall that $q(\boldsymbol{x}_T) \sim \mathcal{N}(\boldsymbol{x}_t; \bar{\alpha}_t\boldsymbol{x}_0, \bar{\beta}_t^2\boldsymbol{I})$, this would require $\bar{\alpha}_t \approx 0$ when $t \to T$. More discussions on the choice of α_t can be referred back to Section 1 1.4.

For σ_t, in theory, different sample distributions $\tilde{p}(\boldsymbol{x}_0)$ will lead to a different optimised σ_t. However, we do not really want to train the variance as an extra parameter. In this case, we can choose two special examples.

σ_t comes from the definition of $q(\boldsymbol{x}_{t-1}|\boldsymbol{x}_t)$, which we formulated it as a learnable distribution $\mathcal{N}(\boldsymbol{x}_{t-1}; \boldsymbol{\mu}(\boldsymbol{x}_t), \sigma_t^2\boldsymbol{I})$, this is needed for the reverse sampling process.

1. Assume there is only one training sample, $\boldsymbol{x}_*$. In this case, we have $\tilde{p}(\boldsymbol{x}_0) = \delta(\boldsymbol{x} - \boldsymbol{x}_*)$, the Dirac distribution. This will give an optimum value of $\sigma_t = \bar{\beta}_{t-1}\beta_t/\bar{\beta}_t$.
2. Assume $\tilde{p}(\boldsymbol{x}_0) \sim \mathcal{N}(0, \boldsymbol{I})$, then the optimum $\sigma_t = \beta_t$.

Both choices lead to comparable performances in practice. The proofs of these results are rather complicated, which will be reserved for the discussions in the next section.

References

[1]Y. Lipman, R. T. Chen, H. Ben-Hamu, M. Nickel, and M. Le, “Flow matching for generative modeling,” arXiv preprint arXiv:2210.02747 (2022).
[2]A. Razavi, A. Van den Oord, and O. Vinyals, “Generating diverse high-fidelity images with VQ-VAE-2,” Adv. Neural Info. Proc. Sys. **32** (2019).
[3]J. Ho, A. Jain, and P. Abbeel, “Denoising diffusion probabilistic models,” Adv. Neural Info. Proc. Sys. **33**, 6840 (2020).
[4]J. Sohl-Dickstein, E. Weiss, N. Maheswaranathan, and S. Ganguli, “Deep unsupervised learning using nonequilibrium thermodynamics,” in *International conference on machine learning* (pmlr, 2015) p. 2256.
[5]C. Luo, “Understanding diffusion models: A unified perspective,” arXiv:2208.11970 (2022).