

## Topics on Diffusion Generative Models

Original Text in Chinese by Jianlin Su<sup>1</sup> and Translated and Annotated by Jack Yang<sup>2</sup>

<sup>1)</sup> <https://kexue.fm>

<sup>2)</sup> *School of Materials Science and Engineering, University of New South Wales, Sydney, New South Wales 2052, Australia<sup>a)</sup>*

(Dated: 28 June 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.

---

<sup>a)</sup> Electronic mail: [jianliang.yang1@unsw.edu.au](mailto:jianliang.yang1@unsw.edu.au)

<b>1. DDPM=Building Demolition and Rebuilding</b>	4
1.1. A new starting point	4
1.2. Demolition and rebuilding	4
1.3. How should we demolish the building?	5
1.4. How to rebuild the building?	6
1.5. Reducing the square error	6
1.6. Regressive generations	7
1.7. Hyperparameter settings	7
<b>2. DDPM=Autoregressive VAE</b>	8
2.1. Solving the problem step-wise	8
2.2. The joint KL divergence	8
2.3. Divide-and-conquer	9
2.4. Reconstructing the diffusion model	9
2.5. Hyperparameter settings	10
<b>3. DDPM=Bayes Plus Denoising</b>	11
3.1. Recap on the model setup	11
3.2. Using the Bayes' theorem	11
3.3. The denoising process	12
3.4. Correcting the estimates	13
3.5. Remaining questions	13
<b>4. DDIM=High Level View of DDPM</b>	14
4.1. The thought process	14
4.2. Method of undetermined coefficients	14
4.3. Same procedure as before	15
4.4. A few examples	15
4.5. Accelerated generations	16
4.6. Differential equation	17
<b>5. The General Framework Based on Stochastic Differential Equation</b>	17
5.1. Stochastic differentiations	18
5.2. The reverse process	18
5.3. Score matching	19
5.4. The final results	20
<b>6. The General Framework Based on Ordinary Differential Equation</b>	22
6.1. Think again	22
6.2. The Dirac function	22
6.3. The Fokker-Planck equation	23
6.4. Equivalent transformation	23
6.5. Neural ODE	24
6.6. Revisiting DDIM	24
<b>7. Optimum Estimation of the Variance (Part I)</b>	25
7.1. Uncertainty in the reconstruction process	26
7.2. Optimisation for the mean	26
7.3. Estimation of the variance: part I	27
7.4. Estimation of the variance: part II	28
7.5. Further thoughts	28
<b>8. Optimum Estimation of the Variance (Part II)</b>	29
8.1. Revisiting the results	29
8.2. Different approaches to improve the results	29
8.3. Maximum likelihood	30
8.4. Conditional variance	30
8.5. Two stages	31
<b>9. Conditional Generation</b>	31
9.1. Theoretical analysis	31
9.2. Conditional input	32
9.3. Approximated distribution	32
9.4. Scaling the classifier gradient	33
9.5. Similarity control	33
9.6. Continuous scenarios	34
9.7. Classifier-free approach	35
<b>10. Unified Diffusion Model: The Theoretical Framework</b>	35
10.1. The forward process	35
10.2. The reverse process	36
10.3. Training target	36
10.4. Conditional probability	37

10.5. The thought process	38
<b>11. Unified Diffusion Model: Applications</b>	38
11.1. Recap on the general framework	38
11.2. Hot diffusion	39
11.3. Cold diffusion	40
11.4. The editing model	40
11.5. The masking model	41
11.6. The encoding model	42
<b>12. Conquering the Diffusion ODE</b>	42
12.1. Differential equation	42
12.2. The Jacobian	43
12.3. Taylor approximation	43
12.4. The equation of heat conduction	44
12.5. Solving for the probability distribution	44
12.6. Completing the model design	45

## 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<sup>1</sup>, VQ-VAE<sup>2</sup> 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.

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.

The popularity of modern diffusion generative models started from the **DDPM (Denoising Diffusion Probabilistic Model)**<sup>3</sup> 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<sup>4</sup> 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).

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

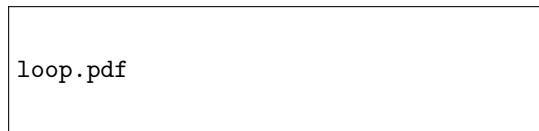


FIG. 1. Building construction analogy to the DDPM model.

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

June 2022, at the time of writing.

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

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

We can rethink of the process as a building process, where the noises  $\mathbf{z}$  are the raw materials, in this case, the bricks and mortar, and the samples  $\mathbf{x}$  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  $\mathbf{x}_0$  be the fully-constructed building (initial sample) and  $\mathbf{x}_T$  being the fully deconstructed bricks (random noises). Assuming the demolition takes  $T$  steps, then the whole demolition process can be written as

$$\mathbf{x} = \mathbf{x}_0 \rightarrow \mathbf{x}_1 \rightarrow \mathbf{x}_2 \rightarrow \cdots \mathbf{x}_{T-1} \rightarrow \mathbf{x}_T = \mathbf{z}. \quad (1.1)$$

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

### 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:

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

in particular,  $\alpha_t$  and  $\beta_t > 0$ ,  $\alpha_t^2 + \beta_t^2 = 1$ . Usually, we chose  $\beta_t \rightarrow 0$ , 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  $\mathbf{x}_{t-1}$  into  $\alpha_t \mathbf{x}_{t-1}$  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

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

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  $(\alpha_t \alpha_{t-1} \cdots \alpha_2) \beta_1$ ,  $(\alpha_t \cdots \alpha_3) \beta_2$ ,  $\dots$ ,  $\alpha_t \beta_{t-1}^2$  and  $\beta_t^2$ , 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

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

This means that, effectively, we have

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

which has now significantly simplified the computations of  $\mathbf{x}_t$ . On the other hand, by choosing a suitable form of  $\alpha_t$  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  $\boldsymbol{\varepsilon}$ .

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 architectures, 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.

This part is the sum of multiple independent Gaussian noises.

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.

#### 1.4. How to rebuild the building?

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

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

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  $\mathbf{x}_{t-1}$  the subject:

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

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

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

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

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

Here,  $\beta_t^2/\alpha_t^2$  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  $\mathbf{x}_t$ :

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

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

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

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  $\mathbf{x}_0$  from the inputs.
2. Sample  $\bar{\boldsymbol{\varepsilon}}_{t-1}$  and  $\bar{\boldsymbol{\varepsilon}}_t$  from a normal distribution  $\mathcal{N}(0, \mathbf{I})$ , 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  $\bar{\beta}_t \boldsymbol{\varepsilon} \sim \mathcal{N}(0, \mathbf{I})$ . Equally, we have  $\beta_t \bar{\boldsymbol{\varepsilon}}_{t-1} - \alpha_t \bar{\beta}_{t-1} \boldsymbol{\varepsilon}_t$  that is equivalent to  $\bar{\beta}_t \mathbf{w} \sim \mathcal{N}(0, \mathbf{I})$ . Furthermore, it can be proven that  $\mathbb{E}[\boldsymbol{\varepsilon} \mathbf{w}^T] = 0$ , so they are two independent normal distributions. With this, we can express  $\boldsymbol{\varepsilon}_t$  with  $\boldsymbol{\varepsilon}$  and  $\mathbf{w}$ :

Not quite sure where this comes from?

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

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

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

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

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

Ignoring the weight and constant again, we arrive at:

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

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  $\mathbf{x}_T \sim \mathcal{N}(0, \mathbf{I})$ , and repeat the process for  $T$  steps according to Eq. (1.7):

$$\mathbf{x}_{t-1} = \frac{1}{\alpha_t} (\mathbf{x}_t - \beta_t \epsilon_\theta(\mathbf{x}_t, t)). \quad (1.15)$$

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

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

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  $\alpha_t$ :

$$\alpha_t = \sqrt{1 - \frac{0.02t}{T}}, \quad (1.17)$$

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

This basically says, we must first sample a noise from the normal distribution  $\mathcal{N}(0, \mathbf{I})$ , from which we can generate the noisy sample  $\mathbf{x}_t = \bar{\alpha}_t \mathbf{x}_0 + \bar{\beta}_t \epsilon$ . Then we plug this  $\mathbf{x}_t$  into the denoising network  $\epsilon_\theta(\mathbf{x}_t)$  to recover  $\epsilon_t$ . This expression is equivalent to the results<sup>5</sup> derived using KL divergence between the destruction and regeneration process:  $\text{argmin}_\theta \mathcal{D}_{KL}(q(\mathbf{x}_{t-1}|\mathbf{x}_t, \mathbf{x}_0) \| p(\mathbf{x}_{t-1}|\mathbf{x}_t))$  from evidence lower bound, which upon further reparameterisation, gives the following expression for the KL divergence:  $\frac{(1-\alpha_t)^2}{2\sigma_q^2(t)} \frac{\|\epsilon_0 - \hat{\epsilon}_\theta(\mathbf{x}, t)\|_2^2}{(1-\bar{\alpha}_t)\alpha_t}$ .

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  $\alpha_t$ . When  $t$  is small,  $\mathbf{x}_t$  is close to the real image, so we need to decrease the distance between  $\mathbf{x}_{t-1}$  and  $\mathbf{x}_t$ , making it more suitable to apply the Euclidean distance in Eq. (1.6). This gives large  $\alpha_t$ . When  $t$  is very large,  $\mathbf{x}_t$  becomes very close to a true Gaussian noise, which can be modelled with the Euclidean distance, thus we can increase the distance between  $\mathbf{x}_{t-1}$  and  $\mathbf{x}_t$  with smaller  $\alpha_t$ . Can we use a large  $\alpha_t$  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

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

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  $\alpha_t$  throughout, then we will have to increase  $T$ .

Finally, note that in the loss function [Eq. (1.14),  $\varepsilon_\theta(\bar{\alpha}_t \mathbf{x}_0 + \bar{\beta}_t \varepsilon, t)$ ], 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<sup>3</sup>, 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:

$$\text{ENCODER: } \mathbf{x} \rightarrow \mathbf{z}; \quad \text{GENERATION: } \mathbf{z} \rightarrow \mathbf{x}. \quad (2.1)$$

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

In this case, we only need to deal with three data distributions: the **encoding distribution**  $p(\mathbf{z}|\mathbf{x})$ , the **generative distribution**  $q(\mathbf{x}|\mathbf{z})$ , and the prior distribution  $q(\mathbf{z})$ . The advantage behind such a formulation is its simplicity. There also exists a simple projective relationship between the sample  $\mathbf{x}$  and the latent variable  $\mathbf{z}$ , 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:

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

In this way, every transition probability  $p(\mathbf{x}_t|\mathbf{x}_{t-1})$  or  $q(\mathbf{x}_{t-1}|\mathbf{x}_t)$  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  $\Delta \mathbf{x}$ . 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  $p(\mathbf{x}_t|\mathbf{x}_{t-1})$  for every encoding step and  $q(\mathbf{x}_{t-1}|\mathbf{x}_t)$  for every generation step. This allows us to write down the **joint probability**

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



distributions:

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

Here,  $\mathbf{x}_0$  represents the input samples, thus  $\tilde{p}(\mathbf{x}_0)$  represents the sample distribution, whereas  $q(\mathbf{x}_T)$  is the prior distribution, as  $\mathbf{x}_T$  is the final encoded sample. The rest probability distributions of the type  $p(\mathbf{x}_t|\mathbf{x}_{t-1})$  and  $q(\mathbf{x}_{t-1}|\mathbf{x}_t)$  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:

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

This is, therefore, the optimisation target for DDPM. What is left is to consolidate the forms of  $p(\mathbf{x}_t|\mathbf{x}_{t-1})$  and  $q(\mathbf{x}_{t-1}|\mathbf{x}_t)$  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:

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

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  $\mathbf{x}_{t-1}$  that is multiplied by a *scalar*  $\alpha_t$ . Hence, DDPM has completely abandoned the capability to encode the data, and only established a generative model. The transition probability for the generating process  $q(\mathbf{x}_{t-1}|\mathbf{x}_t)$  has now been formulated as a learnable model  $\mathcal{N}(\mathbf{x}_{t-1}; \boldsymbol{\mu}(\mathbf{x}_t), \sigma_t^2 \mathbf{I})$ . Here,  $\alpha_t$ ,  $\beta_t$  and  $\sigma_t$  **are all not learnables but parameterised**. Hence,  $\boldsymbol{\mu}(\mathbf{x}_t)$  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

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

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

$$\begin{aligned} & - \int \underbrace{p(\mathbf{x}_T|\mathbf{x}_{T-1}) \cdots p(\mathbf{x}_1|\mathbf{x}_0)\tilde{p}(\mathbf{x}_0)}_{T+1 \text{ terms}} \log q(\mathbf{x}_{t-1}|\mathbf{x}_t) d\mathbf{x}_0 \cdots d\mathbf{x}_T \\ &= - \int \underbrace{p(\mathbf{x}_t|\mathbf{x}_{t-1}) \cdots p(\mathbf{x}_1|\mathbf{x}_0)\tilde{p}(\mathbf{x}_0)}_{t+1 \text{ terms}} \log q(\mathbf{x}_{t-1}|\mathbf{x}_t) d\mathbf{x}_0 \cdots d\mathbf{x}_t \\ &= - \int p(\mathbf{x}_t|\mathbf{x}_{t-1})p(\mathbf{x}_{t-1}|\mathbf{x}_0)\tilde{p}(\mathbf{x}_0) \log q(\mathbf{x}_{t-1}|\mathbf{x}_t) d\mathbf{x}_0 d\mathbf{x}_{t-1} \mathbf{x}_t. \end{aligned} \quad (2.6)$$

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

Similarly, the integral is also independent from  $\mathbf{x}_1, \dots, \mathbf{x}_{t-2}$ .

### 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  $-\log q(\mathbf{x}_{t-1}|\mathbf{x}_t)$  which gives the term  $\frac{1}{2\sigma_t^2} \|\mathbf{x}_{t-1} - \boldsymbol{\mu}(\mathbf{x}_t)\|_2^2$  in the target function for optimisation.

2. The transition probability  $p(\mathbf{x}_{t-1}|\mathbf{x}_0)$  can be equivalently written as  $\mathbf{x}_{t-1} = \bar{\alpha}_{t-1}\mathbf{x}_0 + \bar{\beta}_{t-1}\bar{\boldsymbol{\varepsilon}}_{t-1}$ , similarly  $p(\mathbf{x}_t|\mathbf{x}_{t-1})$  implies  $\mathbf{x}_t = \alpha_t\mathbf{x}_{t-1} + \beta_t\boldsymbol{\varepsilon}_t$ , in which  $\bar{\boldsymbol{\varepsilon}}_{t-1}$  and  $\boldsymbol{\varepsilon}_t \sim \mathcal{N}(0, \mathbf{I})$ .
3. From  $\mathbf{x}_{t-1} = \frac{1}{\alpha_t}(\mathbf{x}_t - \beta_t\boldsymbol{\varepsilon}_t)$ , it inspires us to parameterise  $\boldsymbol{\mu}(\mathbf{x}_t)$  as  $\boldsymbol{\mu}(\mathbf{x}_t) = \frac{1}{\alpha_t}(\mathbf{x}_t - \beta_t\boldsymbol{\varepsilon}_\theta(\mathbf{x}_t, t))$ .

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

This is equivalent to Eq. (1.10).

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

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).

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

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 multivariate integral [Eq. (2.5)]. Although the derivation is longer, it is logically tractable. In comparison to the original DDPM paper<sup>3</sup>, where a conditional probability  $q(\mathbf{x}_{t-1}|\mathbf{x}_t, \mathbf{x}_0)$  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  $\alpha_t$ ,  $\beta_t$  and  $\sigma_t$ . For  $p(\mathbf{x}_t|\mathbf{x}_{t-1})$ , 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:

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

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  $\mathcal{N}(\mathbf{x}_t; \bar{\alpha}_t\mathbf{x}_0, \bar{\beta}_t^2\mathbf{I})$  means we have  $\mathbf{x}_t = \alpha_t\mathbf{x}_{t-1} + \beta_t\boldsymbol{\varepsilon}_t$  with  $\boldsymbol{\varepsilon}_t \sim \mathcal{N}(0, \mathbf{I})$ , 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  $q(\mathbf{x}_T)$  as a standard normal distribution  $\mathcal{N}(\mathbf{x}_T; 0, \mathbf{I})$ , 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:

$$\begin{aligned} q(\mathbf{x}_T) &= \int p(\mathbf{x}_T|\mathbf{x}_{T-1})p(\mathbf{x}_{T-1}|\mathbf{x}_{T-2}) \cdots p(\mathbf{x}_1|\mathbf{x}_0)\tilde{p}(\mathbf{x}_0)d\mathbf{x}_0d\mathbf{x}_1 \cdots d\mathbf{x}_{T-1} \\ &= \int p(\mathbf{x}_T|\mathbf{x}_0)\tilde{p}(\mathbf{x}_0)d\mathbf{x}_0. \end{aligned} \quad (2.10)$$

In general, the sample distribution is arbitrary, such that the only way to make Eq. (2.10) hold is to equate  $p(\mathbf{x}_T|\mathbf{x}_0)$  to become  $q(\mathbf{x}_T)$  [Note that the integral in Eq. (2.10) is independent of  $\mathbf{x}_T$ !]. **It means to evolve  $\tilde{p}(\mathbf{x}_0)$  into a Gaussian distribution that is independent from  $\mathbf{x}_0$ .** Recall that  $q(\mathbf{x}_T) \sim \mathcal{N}(\mathbf{x}_T; \bar{\alpha}_t\mathbf{x}_0, \bar{\beta}_t^2\mathbf{I})$ , this would require  $\bar{\alpha}_t \approx 0$  when  $t \rightarrow T$ . More discussions on the choice of  $\alpha_t$  can be referred back to Section 1.1.4.

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

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

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

### 3. DDPM=Bayes Plus Denoising

In the previous two sections, we have provided two different mathematical derivations for DDPM. The first one is based on an analogous comparison to the building of a house, which is easy to understand, but difficult to be extended theoretically. In the second approach, we draw our conclusions from the framework of autoregressive variational autoencoder. Although theoretically more robust, it is rather formal and lack of inspiration for further improvements. In this section, we will give another derivation of the DDPM, which applies the Bayes' rule to simplify the computations. The derivation contains more reasoning, which makes it more inspirational. Such derivation also has a deeper relationship with the DDIM, which is to be discussed in the next section.

Original blog post see <https://www.spaces.ac.cn/archives/9164>

Diffusion Denoising Implicit Model

#### 3.1. Recap on the model setup

Recall that in the DDPM models, the forward process is modelled as a process of progressive transformation:

$$\mathbf{x} = \mathbf{x}_0 \rightarrow \mathbf{x}_1 \rightarrow \mathbf{x}_2 \rightarrow \cdots \rightarrow \mathbf{x}_{T-1} \rightarrow \mathbf{x}_T = \mathbf{z} \quad (3.1)$$

in which the process gradually adds noise to the sample  $\mathbf{x}$  and transform it into a pure noise  $\mathbf{z}$ , whereas the reverse process gradually denoises  $\mathbf{z}$  to recover the sample data  $\mathbf{x}$ . This reverse process is the generative model that we aim to build.

The forward process is simple to achieve, in which every step is given by

$$\mathbf{x}_t = \alpha_t \mathbf{x}_{t-1} + \beta_t \boldsymbol{\varepsilon}_t \quad \boldsymbol{\varepsilon}_t \sim \mathcal{N}(0, \mathbf{I}), \quad (3.2)$$

which can be equivalently written as  $p(\mathbf{x}_t | \mathbf{x}_{t-1}) = \mathcal{N}(\mathbf{x}_t; \alpha_t \mathbf{x}_{t-1}, \beta_t^2 \mathbf{I})$ . Under the constraint of  $\alpha_t^2 + \beta_t^2 = 1$ , we have the following:

$$\begin{aligned} \mathbf{x}_t &= \alpha_t \mathbf{x}_{t-1} + \beta_t \boldsymbol{\varepsilon}_t \\ &= \alpha_t (\alpha_{t-1} \mathbf{x}_{t-2} + \beta_{t-1} \boldsymbol{\varepsilon}_{t-1}) + \beta_t \boldsymbol{\varepsilon}_t \\ &= \cdots \\ &= (\alpha_t \cdots \alpha_1) \mathbf{x}_0 + \underbrace{(\alpha_t \cdots \alpha_2) \beta_1 \boldsymbol{\varepsilon}_1 + (\alpha_t \cdots \alpha_3) \beta_2 \boldsymbol{\varepsilon}_2 + \cdots + \alpha_t \beta_{t-1} \boldsymbol{\varepsilon}_{t-1} + \beta_t \boldsymbol{\varepsilon}_t}_{\sim \mathcal{N}(0, (1 - \alpha_t^2 \cdots \alpha_1^2) \mathbf{I})} \end{aligned} \quad (3.3)$$

From which we have  $p(\mathbf{x}_t | \mathbf{x}_0) = \mathcal{N}(\mathbf{x}_t; \bar{\alpha}_t \mathbf{x}_0, \bar{\beta}_t^2 \mathbf{I})$  with  $\bar{\alpha}_t = \alpha_1 \cdots \alpha_t$  and  $\bar{\beta}_t = \sqrt{1 - \bar{\alpha}_t^2}$ . What DDPM is trying to achieve, is to figure out how to obtain  $p(\mathbf{x}_{t-1} | \mathbf{x}_t)$  for the reverse generation process using the information above. With this, we could start from any arbitrary  $\mathbf{z} = \mathbf{x}_T$ , and through sampling  $\mathbf{x}_{T-1}, \mathbf{x}_{T-2}, \dots, \mathbf{x}_1$  stepwise, we could eventually generate a new sample  $\mathbf{x} = \mathbf{x}_0$ .

#### 3.2. Using the Bayes' theorem

According to the Bayes' theorem, we have, in this case

$$p(\mathbf{x}_{t-1} | \mathbf{x}_t) = \frac{p(\mathbf{x}_t | \mathbf{x}_{t-1}) p(\mathbf{x}_{t-1})}{p(\mathbf{x}_t)}. \quad (3.4)$$

However, we do not know what  $p(\mathbf{x}_{t-1})$  and  $p(\mathbf{x})$  look like, so we cannot move further from this expression. However, we know that these distributions are tractable once they are conditioned on the input  $\mathbf{x}_0$ . In this case, we have

$$p(\mathbf{x}_{t-1} | \mathbf{x}_t, \mathbf{x}_0) = \frac{p(\mathbf{x}_t | \mathbf{x}_{t-1}) p(\mathbf{x}_{t-1} | \mathbf{x}_0)}{p(\mathbf{x}_t | \mathbf{x}_0)}. \quad (3.5)$$

As such, in what follows, we will focus on deriving expressions for  $p(\mathbf{x}_{t-1} | \mathbf{x}_t)$

Since we know the expressions for  $p(\mathbf{x}_t|\mathbf{x}_{t-1})$ ,  $p(\mathbf{x}_{t-1}|\mathbf{x}_0)$  and  $p(\mathbf{x}_t|\mathbf{x}_0)$  (the latter two from one-shot sampling), Eq. (3.5) can be simplified into:

$$p(\mathbf{x}_{t-1}|\mathbf{x}_t, \mathbf{x}_0) = \mathcal{N}\left(\mathbf{x}_{t-1}; \frac{\alpha_t \bar{\beta}_{t-1}^2}{\beta_t^2} \mathbf{x}_t + \frac{\bar{\alpha}_{t-1} \beta_t^2}{\bar{\beta}_t^2} \mathbf{x}_0, \frac{\bar{\beta}_{t-1}^2 \beta_t^2}{\bar{\beta}_t^2} \mathbf{I}\right). \quad (3.6)$$

To prove Eq. (3.6), note that the exponent in Eq. (3.5), ignoring the factor  $\frac{1}{2}$ , is

$$\frac{\|\mathbf{x}_t - \alpha_t \mathbf{x}_{t-1}\|_2^2}{\beta_t^2} + \frac{\|\mathbf{x}_{t-1} - \bar{\alpha}_{t-1} \mathbf{x}_0\|_2^2}{\bar{\beta}_{t-1}^2} - \frac{\|\mathbf{x}_t - \bar{\alpha}_t \mathbf{x}_0\|_2^2}{\bar{\beta}_t^2}. \quad (3.7)$$

This expression is quadratic in  $\mathbf{x}_{t-1}$ , so the final distribution must also be a normal distribution. This means that we only need to determine the corresponding mean and variance. It can be shown that the coefficient for the  $\|\mathbf{x}_{t-1}\|_2^2$  term is

$$\frac{\alpha_t^2}{\beta_t^2} + \frac{1}{\bar{\beta}_{t-1}^2} = \frac{\alpha_t^2 \bar{\beta}_{t-1}^2 + \beta_t^2}{\beta_t^2 \bar{\beta}_{t-1}^2} = \frac{\alpha_t^2 (1 - \bar{\alpha}_{t-1}^2) + (1 - \alpha_t^2)}{\beta_t^2 \bar{\beta}_{t-1}^2} = \frac{1 - \bar{\alpha}_t^2}{\beta_t^2 \bar{\beta}_{t-1}^2} = \frac{\bar{\beta}_t^2}{\beta_t^2 \bar{\beta}_{t-1}^2} \quad (3.8)$$

This means that the final exponent must be of the form  $\frac{\bar{\beta}_t^2}{\beta_t^2 \bar{\beta}_{t-1}^2} \|\mathbf{x}_{t-1} - \bar{\boldsymbol{\mu}}(\mathbf{x}_t, \mathbf{x}_0)\|_2^2$ , such that the covariance matrix is  $\frac{\beta_t^2 \bar{\beta}_{t-1}^2}{\bar{\beta}_t^2} \mathbf{I}$ . On the other hand, taking out the linear term in  $\mathbf{x}_{t-1}$ , the coefficient is  $-2 \left( \frac{\alpha_t}{\beta_t} \mathbf{x}_t + \frac{\bar{\alpha}_{t-1}}{\bar{\beta}_{t-1}^2} \mathbf{x}_0 \right)$ , which, upon dividing by  $-\frac{2\bar{\beta}_t^2}{\beta_t^2 \bar{\beta}_{t-1}^2}$ , we get the following:

$$\bar{\boldsymbol{\mu}}(\mathbf{x}_t, \mathbf{x}_0) = \frac{\alpha_t \bar{\beta}_{t-1}^2}{\beta_t} \mathbf{x}_t + \frac{\bar{\alpha}_{t-1} \beta_t^2}{\bar{\beta}_t} \mathbf{x}_0. \quad (3.9)$$

By now, we have acquired all the information we need for the transition probability  $p(\mathbf{x}_{t-1}|\mathbf{x}_t, \mathbf{x}_0)$ .

### 3.3. The denoising process

Although we now have an expression for  $p(\mathbf{x}_{t-1}|\mathbf{x}_t, \mathbf{x}_0)$ , this is not the final answer that we want. This is because ultimately we would like to predict  $\mathbf{x}_{t-1}$  from  $\mathbf{x}_t$  that is independent from  $\mathbf{x}_0$ . If, on the other hand, we can predict  $\mathbf{x}_0$  from  $\mathbf{x}_t$ , then we would be able to estimate  $\mathbf{x}_0$  in  $p(\mathbf{x}_{t-1}|\mathbf{x}_t, \mathbf{x}_0)$  and make it only dependent upon  $\mathbf{x}_t$ . This can be achieved by letting  $\mathbf{x}_0 = \bar{\boldsymbol{\mu}}(\mathbf{x}_t)$  as a learnable, with a loss function of  $\|\mathbf{x}_0 - \bar{\boldsymbol{\mu}}(\mathbf{x}_t)\|_2^2$ . After the model has been trained, we can write

$$p(\mathbf{x}_{t-1}|\mathbf{x}_t, \mathbf{x}_0) \approx p(\mathbf{x}_{t-1}|\mathbf{x}_t, \mathbf{x}_0 = \bar{\boldsymbol{\mu}}(\mathbf{x}_t)) = \mathcal{N}\left(\mathbf{x}_{t-1}; \frac{\alpha_t \bar{\beta}_{t-1}^2}{\beta_t^2} \mathbf{x}_t + \frac{\bar{\alpha}_{t-1} \beta_t^2}{\bar{\beta}_t^2} \bar{\boldsymbol{\mu}}(\mathbf{x}_t), \frac{\bar{\beta}_{t-1}^2 \beta_t^2}{\bar{\beta}_t^2} \mathbf{I}\right). \quad (3.10)$$

In the expression  $\|\mathbf{x}_0 - \bar{\boldsymbol{\mu}}(\mathbf{x}_t)\|_2^2$ ,  $\mathbf{x}_0$  represents the original data,  $\mathbf{x}_t$  is the noised data. Hence, we are training a denoising model! This is the meaning of the first D in DDPM.

More specifically,  $p(\mathbf{x}_t|\mathbf{x}_0) = \mathcal{N}(\mathbf{x}_t; \bar{\alpha}_t \mathbf{x}_0, \bar{\beta}_t^2 \mathbf{I})$ , which means  $\mathbf{x}_t = \bar{\alpha}_t \mathbf{x}_0 + \bar{\beta}_t \boldsymbol{\varepsilon}$  with  $\boldsymbol{\varepsilon} \sim \mathcal{N}(0, \mathbf{I})$ , or equivalently,  $\mathbf{x}_0 = \frac{1}{\bar{\alpha}_t} (\mathbf{x}_t - \bar{\beta}_t \boldsymbol{\varepsilon})$ . This inspires us to parameterise  $\bar{\boldsymbol{\mu}}(\mathbf{x}_t)$  in the form of

$$\bar{\boldsymbol{\mu}}(\mathbf{x}_t) = \frac{1}{\bar{\alpha}_t} (\mathbf{x}_t - \bar{\beta}_t \boldsymbol{\varepsilon}_\theta(\mathbf{x}_t, t)). \quad (3.11)$$

In this case, the loss function for training  $\bar{\boldsymbol{\mu}}(\mathbf{x}_t)$  becomes

$$\|\mathbf{x}_0 - \bar{\boldsymbol{\mu}}(\mathbf{x}_t)\|_2^2 = \frac{\bar{\beta}_t^2}{\bar{\alpha}_t^2} \|\boldsymbol{\varepsilon} - \bar{\beta}_t \boldsymbol{\varepsilon}_\theta(\bar{\alpha}_t \mathbf{x}_0 + \bar{\beta}_t \boldsymbol{\varepsilon}, t)\|_2^2. \quad (3.12)$$

Upon ignoring the coefficient  $\bar{\beta}_t^2/\bar{\alpha}_t^2$ , we thus recovered the loss function in the original DDPM paper. It can be seen that the derivation here from  $\mathbf{x}_t$  to  $\mathbf{x}_{t-1}$  is more straight forward, which avoids the manipulations of integrals as done in the previous section. Now we can substitute Eq. (3.11) back into Eq. (3.10) and simplify, which lead to

$$p(\mathbf{x}_{t-1}|\mathbf{x}_t, \mathbf{x}_0) \approx p(\mathbf{x}_{t-1}|\mathbf{x}_t, \mathbf{x}_0 = \bar{\boldsymbol{\mu}}(\mathbf{x}_t)) = \mathcal{N}\left(\mathbf{x}_{t-1}; \frac{1}{\alpha_t} \left( \mathbf{x}_t - \frac{\bar{\beta}_t}{\bar{\alpha}_t} \boldsymbol{\varepsilon}_\theta(\mathbf{x}_t, t) \right), \frac{\bar{\beta}_{t-1}^2 \beta_t^2}{\bar{\beta}_t^2} \mathbf{I}\right). \quad (3.13)$$

Once trained, this will be the probability distribution that will be used for the reverse sampling

process.

To derive this result, note the following:

$$\frac{\bar{\alpha}_{t-1}\beta_t^2}{\bar{\beta}_t^2} \frac{1}{\bar{\alpha}_t} (\mathbf{x}_t - \bar{\beta}_t \boldsymbol{\varepsilon}_\theta(\mathbf{x}_t, t)) = \frac{\beta_t^2}{\bar{\beta}_t^2 \alpha_t} \mathbf{x}_t - \frac{\beta_t^2}{\bar{\beta}_t \alpha_t} \bar{\beta}_t \boldsymbol{\varepsilon}_\theta(\mathbf{x}_t, t).$$

Using the fact that  $\bar{\alpha}_t = \bar{\alpha}_{t-1} \alpha_t$ . Then for the coefficient of  $\mathbf{x}_t$ , we have

$$\frac{\alpha_t \bar{\beta}_{t-1}^2}{\bar{\beta}_t^2} + \frac{\beta_t^2}{\bar{\beta}_t^2 \alpha_t} = \frac{\alpha_t^2 \bar{\beta}_{t-1}^2 + \beta_t^2}{\bar{\beta}_t^2 \alpha_t} = \frac{\alpha_t^2 (1 - \bar{\alpha}_{t-1}^2) + \beta_t^2}{\bar{\beta}_t^2 \alpha_t} = \frac{(\alpha_t^2 + \beta_t^2) - \alpha_t \bar{\alpha}_{t-1}^2}{\bar{\beta}_t^2 \alpha_t} = \frac{1 - \bar{\alpha}_t^2}{\alpha_t \bar{\beta}_t^2} = \frac{\bar{\beta}_t^2}{\alpha_t \bar{\beta}_t^2} = \frac{1}{\alpha_t}.$$

### 3.4. Correcting the estimates

An interesting question arises here, which is related to the introduction of  $\bar{\boldsymbol{\mu}}(\mathbf{x}_t)$  that is used for estimating  $\mathbf{x}_0$  when approximating  $p(\mathbf{x}_{t-1}|\mathbf{x}_t, \mathbf{x}_0)$  with  $p(\mathbf{x}_{t-1}|\mathbf{x}_t)$ . If such a model can be used for estimating  $\mathbf{x}_0$  in on-shot, then why would we still need to transform  $\mathbf{x}_t$  to  $\mathbf{x}_0$  slowly in a stepwise manner?

The problem is that, in reality, estimating  $\mathbf{x}_0$  from  $\bar{\boldsymbol{\mu}}(\mathbf{x}_t)$  will not give very accurate result, at least in the first two steps of the reverse process. It only plays a forecasting roles to enable us taking a small step  $p(\mathbf{x}_{t-1}|\mathbf{x}_t)$  backward. This is similar to the idea of ‘estimate-and-correct’ in numerical calculations, in which we start from a rough estimate and iterate forward until an accurate answer is reached. Similar idea can be found in Hinton’s paper ‘*Lookahead optimiser: k steps forward, 1 step back*’.<sup>6</sup>

### 3.5. Remaining questions

In the previous section, we pointed out that the challenge behind the direct application of Eq. (3.4) is the existence of the terms  $p(\mathbf{x}_{t-1})$  and  $p(\mathbf{x}_t)$  that are numerically intractable. This is because, according to the following definition using the marginalisation:

$$p(\mathbf{x}_t) = \int p(\mathbf{x}_t|\mathbf{x}_0) \tilde{p}(\mathbf{x}_0) d\mathbf{x}_0, \quad (3.14)$$

in which we know the distribution  $p(\mathbf{x}_t|\mathbf{x}_0)$  but  $\tilde{p}(\mathbf{x}_0)$  is not known a priori, so we cannot compute  $p(\mathbf{x}_t)$  directly, except in two special cases, which will be discussed here. It also provides the proofs for the choices of the standard deviation  $\sigma_t$  that was introduced in the previous section.

In the first case, we have only one sample in the entire dataset. Without lost of generality, we assume this sample is  $\mathbf{0}$ , then  $\tilde{p}(\mathbf{x}_0) = \delta(\mathbf{x}_0)$ . In this case, we have  $p(\mathbf{x}_t|\mathbf{0}) = p(\mathbf{x}_t)$ . Substitute this into Eq. (3.4) shows that this corresponds to the conditional distributions  $p(\mathbf{x}_{t-1}|\mathbf{x}_t, \mathbf{x}_0)$  under the special condition of  $\mathbf{x}_0 = \mathbf{0}$ . As such, we now have

This can be obtained by setting  $\boldsymbol{\mu}(\mathbf{x}_t) \equiv \mathbf{0}$  in Eq. (3.10).

$$p(\mathbf{x}_{t-1}|\mathbf{x}_t) = p(\mathbf{x}_{t-1}|\mathbf{x}_t, \mathbf{x}_0 = \mathbf{0}) = \mathcal{N}\left(\mathbf{x}_{t-1}; \frac{\alpha_t \bar{\beta}_{t-1}^2}{\bar{\beta}_t^2} \mathbf{x}_t, \frac{\bar{\beta}_{t-1}^2 \beta_t^2}{\bar{\beta}_t^2} \mathbf{I}\right). \quad (3.15)$$

This shows that, in this case, we have  $\sigma_t = \frac{\bar{\beta}_{t-1}^2 \beta_t^2}{\bar{\beta}_t^2}$ , which is one particular choice of the standard deviation for the reverse sampling process.

In the second case, the sample follows a normal distribution  $\tilde{p}(\mathbf{x}_0) = \mathcal{N}(\mathbf{x}_0; \mathbf{0}, \mathbf{I})$ . From the one-shot sampling in the forward diffusion  $p(\mathbf{x}_t|\mathbf{x}_0) = \mathcal{N}(\mathbf{x}_t; \bar{\alpha}_t \mathbf{x}_0, \bar{\beta}_t \mathbf{I})$ , we have  $\mathbf{x}_t = \bar{\alpha}_t \mathbf{x}_0 + \bar{\beta}_t \boldsymbol{\varepsilon}$ , with  $\boldsymbol{\varepsilon} \sim \mathcal{N}(\mathbf{0}, \mathbf{I})$ . Since  $\tilde{p}(\mathbf{x}_0) \sim \mathcal{N}(\mathbf{0}, \mathbf{I})$ ,  $p(\mathbf{x}_t)$  should also follow a normal distribution. Substituting the expression of a normal distribution into Eq. (3.4) and removing the factor of  $-\frac{1}{2}$ , the result is

$$\frac{\|\mathbf{x}_t - \alpha_{t-1} \mathbf{x}_{t-1}\|^2}{\beta_t^2} + \|\mathbf{x}_{t-1}\|^2 - \|\mathbf{x}_t\|^2. \quad (3.16)$$

Similar to how we derived the expression for  $p(\mathbf{x}_{t-1}|\mathbf{x}_t, \mathbf{x}_0)$ , the corresponding distribution can be written as

$$p(\mathbf{x}_{t-1}|\mathbf{x}_t) = \mathcal{N}(\mathbf{x}_{t-1}; \alpha_t \mathbf{x}_t, \beta_t^2 \mathbf{I}), \quad (3.17)$$

which shows the variant  $\sigma_t = \beta_t^2$ , giving another choice if  $\sigma_t$  for the sampling process.

## 4. DDIM=High Level View of DDPM

Till now, we have shown three different interpretations of DDPM, each from a different perspective. So, is there a higher-level view of DDPM, from which we could acquire some new understanding on the diffusion model? The answer is yes. This leads to the DDIM that was introduced in the paper entitled “Denoising Diffusion Implicit Models”<sup>7</sup>, which will be discussed in this section.

Original blog post see <https://www.spaces.ac.cn/archives/9181>

### 4.1. The thought process

In the previous sections, we have mentioned that the derivation of DDPM is closely related to DDIM. More specifically, the derivations presented in the previous sections can be briefly summarised as following:

$$p(\mathbf{x}_t|\mathbf{x}_{t-1}) \xrightarrow{\text{derive}} p(\mathbf{x}_t|\mathbf{x}_0) \xrightarrow{\text{derive}} p(\mathbf{x}_{t-1}|\mathbf{x}_t, \mathbf{x}_0) \xrightarrow{\text{approximate}} p(\mathbf{x}_{t-1}|\mathbf{x}_t). \quad (4.1)$$

The whole process is progressed in a stepwise manner. However, it is not difficult to find specific characteristics in the above formulation:

1. The loss function for model training is only dependent on  $p(\mathbf{x}_t|\mathbf{x}_0)$ ;
2. The sampling process only relies on  $p(\mathbf{x}_{t-1}|\mathbf{x}_t)$ .

It means that, although the entire derivation for the DDPM results is started from  $p(\mathbf{x}_t|\mathbf{x}_{t-1})$  and progress forwards in a stepwise manner, the final results have actually nothing to do with  $p(\mathbf{x}_t|\mathbf{x}_{t-1})$ . This makes us to ask the following question:

#### High-Level Viewpoint 1

Since the final results for the diffusion generative model is independent from  $p(\mathbf{x}_t|\mathbf{x}_{t-1})$ , can this term be completely removed from the entire derivation?

This is the key idea that led to the birth of DDIM.

### 4.2. Method of undetermined coefficients

Some readers may find such an idea contradicts the Baye’s rule that was applied in the derivation presented in the previous chapter:

$$p(\mathbf{x}_{t-1}|\mathbf{x}_t, \mathbf{x}_0) = \frac{p(\mathbf{x}_t|\mathbf{x}_{t-1})p(\mathbf{x}_{t-1}|\mathbf{x}_0)}{p(\mathbf{x}_t|\mathbf{x}_0)}. \quad (4.2)$$

In this case, if we do not have an ansatz for  $p(\mathbf{x}_t|\mathbf{x}_{t-1})$ , then how will we be able to obtain  $p(\mathbf{x}_{t-1}|\mathbf{x}_t, \mathbf{x}_0)$ ? This question is actually raised from a very narrow mindset. In fact, without specifying  $p(\mathbf{x}_t|\mathbf{x}_{t-1})$ , the solution space for  $p(\mathbf{x}_{t-1}|\mathbf{x}_t, \mathbf{x}_0)$  is even larger. In certain aspects, it even provides a much simpler derivation, because now  $p(\mathbf{x}_{t-1}|\mathbf{x}_t, \mathbf{x}_0)$  will only need to satisfy the following condition for the marginal distribution:

$$\int p(\mathbf{x}_{t-1}|\mathbf{x}_t, \mathbf{x}_0)p(\mathbf{x}_t|\mathbf{x}_0)d\mathbf{x}_t = p(\mathbf{x}_{t-1}|\mathbf{x}_0). \quad (4.3)$$

We can solve this equation using the method of undetermined coefficients. In the previous section, we have shown that the solution for  $p(\mathbf{x}_{t-1}|\mathbf{x}_t, \mathbf{x}_0)$  is a normal distribution, so here, we can generalise this even further and let

$$p(\mathbf{x}_{t-1}|\mathbf{x}_t, \mathbf{x}_0) = \mathcal{N}(\mathbf{x}_{t-1}; \kappa_t \mathbf{x}_t + \lambda_t \mathbf{x}_0, \sigma_t^2 \mathbf{I}), \quad (4.4)$$

in which  $\kappa_t$ ,  $\lambda_t$  and  $\sigma_t$  are all coefficients to be determined. In order not to retrain the model, we keep  $p(\mathbf{x}_{t-1}|\mathbf{x}_0)$  and  $p(\mathbf{x}_t|\mathbf{x}_0)$  as before. As such, we have in which  $\boldsymbol{\varepsilon}$ ,  $\boldsymbol{\varepsilon}_1$ ,  $\boldsymbol{\varepsilon}_2 \sim \mathcal{N}(0, \mathbf{I})$ , and based on the superposition of normal distributions, we have  $\kappa_t \bar{\beta}_t \boldsymbol{\varepsilon}_1 + \sigma_t \boldsymbol{\varepsilon}_2 \sim \sqrt{\kappa_t^2 \bar{\beta}_t^2 + \sigma_t^2} \boldsymbol{\varepsilon}$ . By comparing the two different sampling approaches for computing  $\mathbf{x}_{t-1}$ , we found that, in order for Eq. (4.3) to hold, we must equate

$$\bar{\alpha}_{t-1} = \kappa_t \bar{\alpha}_t + \lambda_t, \quad \bar{\beta}_{t-1} = \sqrt{\kappa_t^2 \bar{\beta}_t^2 + \sigma_t^2}. \quad (4.5)$$

Notation	Representative Distribution	Sampling
$p(\mathbf{x}_{t-1} \mathbf{x}_0)$	$\mathcal{N}(\mathbf{x}_{t-1}; \bar{\alpha}_{t-1}\mathbf{x}_0, \beta_{t-1}^2\mathbf{I})$	$\mathbf{x}_{t-1} = \bar{\alpha}_{t-1}\mathbf{x}_0 + \beta_{t-1}\boldsymbol{\varepsilon}$
$p(\mathbf{x}_t \mathbf{x}_0)$	$\mathcal{N}(\mathbf{x}_t; \bar{\alpha}_t\mathbf{x}_0, \beta_t^2\mathbf{I})$	$\mathbf{x}_t = \bar{\alpha}_t\mathbf{x}_0 + \beta_t\boldsymbol{\varepsilon}_1$
$p(\mathbf{x}_{t-1} \mathbf{x}_t, \mathbf{x}_0)$	$\mathcal{N}(\mathbf{x}_{t-1}; \kappa_t\mathbf{x}_t + \lambda_t\mathbf{x}_0, \sigma_t^2\mathbf{I})$	$\mathbf{x}_{t-1} = \kappa_t\mathbf{x}_t + \lambda_t\mathbf{x}_0 + \sigma_t\boldsymbol{\varepsilon}_2$
$\int p(\mathbf{x}_{t-1} \mathbf{x}_t, \mathbf{x}_0) \cdot p(\mathbf{x}_t \mathbf{x}_0) d\mathbf{x}_t$		$\mathbf{x}_{t-1} = \kappa_t\mathbf{x}_t + \lambda_t\mathbf{x}_0 + \sigma_t\boldsymbol{\varepsilon}_2$ $= \kappa_t(\bar{\alpha}_t\mathbf{x}_0 + \beta_t\boldsymbol{\varepsilon}_1) + \lambda_t\mathbf{x}_0 + \sigma_t\boldsymbol{\varepsilon}_2$ $= (\kappa_t\bar{\alpha}_t + \lambda_t)\mathbf{x}_0 + (\kappa_t\beta_t\boldsymbol{\varepsilon}_1 + \sigma_t\boldsymbol{\varepsilon}_2)$

In which we have three unknowns but only two equations. This is why the solution space for  $p(\mathbf{x}_{t-1}|\mathbf{x}_t, \mathbf{x}_0)$  is even larger if we do not specify  $p(\mathbf{x}_t|\mathbf{x}_{t-1})$ . By making  $\sigma_t$  as an independent variable, we have the following solutions:

$$\kappa_t = \frac{\sqrt{\beta_{t-1}^2 - \sigma_t^2}}{\beta_{t-1}^2}, \quad \lambda_t = \bar{\alpha}_{t-1} - \frac{\bar{\alpha}_t\sqrt{\beta_{t-1}^2 - \sigma_t^2}}{\beta_t^2} \quad (4.6)$$

or equivalently,

$$p(\mathbf{x}_{t-1}|\mathbf{x}_t, \mathbf{x}_0) = \mathcal{N}\left(\mathbf{x}_{t-1}; \frac{\sqrt{\beta_{t-1}^2 - \sigma_t^2}}{\beta_{t-1}^2}\mathbf{x}_t + \left(\bar{\alpha}_{t-1} - \frac{\bar{\alpha}_t\sqrt{\beta_{t-1}^2 - \sigma_t^2}}{\beta_t^2}\right)\mathbf{x}_0, \sigma_t^2\mathbf{I}\right). \quad (4.7)$$

For convenience, we choose  $\bar{\alpha}_0 = 1$  and  $\beta_0 = 0$ . More specifically, we don't necessarily need to constrain  $\alpha_t^2 + \beta_t^2 = 1$  here. But in order to align with the previous results, we still set  $\alpha_t^2 + \beta_t^2 = 1$ .

#### 4.3. Same procedure as before

The above derivation shows that, without specifying  $p(\mathbf{x}_t|\mathbf{x}_{t-1})$ , but given  $p(\mathbf{x}_t|\mathbf{x}_0)$  and  $p(\mathbf{x}_{t-1}|\mathbf{x}_0)$ , we arrive at a set of solutions for  $p(\mathbf{x}_{t-1}|\mathbf{x}_t, \mathbf{x}_0)$  that are dependent on the free parameter  $\sigma_t$ . Using the analogy of demolishing and rebuilding from Section 1, it means now we know how the building will finally be demolished into  $[p(\mathbf{x}_t|\mathbf{x}_0), p(\mathbf{x}_{t-1}|\mathbf{x}_0)]$ , but we do not know how each demolition step  $[p(\mathbf{x}_t|\mathbf{x}_{t-1})]$  is achieved. From here, we want to learn how to rebuild the building through  $p(\mathbf{x}_{t-1}|\mathbf{x}_t)$ . Of course, if we want to know how the demolition process works, we could use the Baye's theorem in the opposite way,

$$p(\mathbf{x}_t|\mathbf{x}_{t-1}, \mathbf{x}_0) = \frac{p(\mathbf{x}_{t-1}|\mathbf{x}_t, \mathbf{x}_0)p(\mathbf{x}_t|\mathbf{x}_0)}{p(\mathbf{x}_{t-1}|\mathbf{x}_0)}. \quad (4.8)$$

What follows is the same as shown in the previous chapter. Since we would like to use  $p(\mathbf{x}_{t-1}|\mathbf{x}_t)$  for the generation process, not  $p(\mathbf{x}_{t-1}|\mathbf{x}_t, \mathbf{x}_0)$  which is conditioned on  $\mathbf{x}_0$ , we want to parameterise  $\mathbf{x}_0$  with

$$\bar{\boldsymbol{\mu}}(\mathbf{x}_t) = \frac{1}{\bar{\alpha}_t} (\mathbf{x}_t - \bar{\beta}_t\boldsymbol{\varepsilon}_\theta(\mathbf{x}_t, t)). \quad (4.9)$$

Because we did not change  $p(\mathbf{x}_t|\mathbf{x}_0)$ , the target function for the training the model is still  $\|\boldsymbol{\varepsilon} - \boldsymbol{\varepsilon}_\theta(\bar{\alpha}_t\mathbf{x}_0 + \bar{\beta}_t\boldsymbol{\varepsilon}, t)\|_2^2$ . This means that the training process has not changed. We can use the model trained from DDPM and replace  $\mathbf{x}_0$  in Eq. (4.7) with  $\bar{\boldsymbol{\mu}}(\mathbf{x}_t)$ , from which we get:

$$\begin{aligned} p(\mathbf{x}_{t-1}|\mathbf{x}_t, \mathbf{x}_0) &= (\mathbf{x}_{t-1}|\mathbf{x}_t, \mathbf{x}_0 = \bar{\boldsymbol{\mu}}(\mathbf{x}_t)) \\ &= \mathcal{N}\left(\mathbf{x}_{t-1}; \frac{1}{\alpha_t} \left(\mathbf{x}_t - \left(\bar{\beta}_t - \alpha_t\sqrt{\bar{\beta}_{t-1}^2 - \sigma_t^2}\right)\boldsymbol{\varepsilon}_\theta(\mathbf{x}_t, t)\right), \sigma_t^2\mathbf{I}\right). \end{aligned} \quad (4.10)$$

This gives  $p(\mathbf{x}_{t-1}|\mathbf{x}_t)$  that is needed for the sampling procedure, where  $\alpha_t = \bar{\alpha}_t/\bar{\alpha}_{t-1}$ . Here, we did not modify the training procedure, as such, the model also did not change. However, the sampling process now has a new parameter  $\sigma_t$ , which brings some fresh results to DDPM.

#### 4.4. A few examples

Refer back to the discussions on the choices of standard deviations  $\sigma_t$  in the previous two sections.



In principle, there is no restriction on the choice of the parameter  $\sigma_t$ . However, different choices of  $\sigma_t$  will lead to completely different sampling process. Here we will give a few examples.

First, let  $\sigma_t = \bar{\beta}_{t-1}\beta_t/\bar{\beta}_t$ , where  $\beta_t = \sqrt{1 - \alpha_t^2}$ . Substituting this back into Eq. (4.10) and we have

$$\begin{aligned} p(\mathbf{x}_{t-1}|\mathbf{x}_t) &\approx p(\mathbf{x}_{t-1}|\mathbf{x}_t, \mathbf{x}_0 = \boldsymbol{\mu}(\mathbf{x}_t)) \\ &= \mathcal{N}\left(\mathbf{x}_{t-1}; \frac{1}{\alpha_t} \left(\mathbf{x}_t - \frac{\beta_t^2}{\bar{\beta}_t} \boldsymbol{\varepsilon}_\theta(\mathbf{x}_t, t)\right), \frac{\bar{\beta}_{t-1}^2 \beta_t^2}{\bar{\beta}_t^2} \mathbf{I}\right), \end{aligned} \quad (4.11)$$

which is identical to the result presented in Eq. (3.13). In particular, in the DDIM paper, the results from setting different  $\sigma_t = \eta \bar{\beta}_{t-1} \beta_t / \bar{\beta}_t$  with  $\eta \in [0, 1]$  had been compared.

In the second example, we have  $\sigma_t = \beta_t$ , which is the other choice of  $\sigma_t$  that has been discussed previously. However, this does not lead to a more simplified version of Eq. (4.10). Nevertheless, the DDIM shows good performances in the numerical experiments using this standard choice of  $\sigma_t$  from DDPM.

In the most extreme case, we can let  $\sigma_t = 0$ . Then the transformation from  $\mathbf{x}_t$  to  $\mathbf{x}_0$  becomes a well-defined one:

$$\mathbf{x}_{t-1} = \frac{1}{\alpha_t} \left(\mathbf{x}_t - (\bar{\beta}_t - \alpha_t \bar{\beta}_{t-1}) \boldsymbol{\varepsilon}_\theta(\mathbf{x}_t, t)\right). \quad (4.12)$$

This is the case that was the focus of the original DDIM paper. Strictly speaking, DDIM is referred to this particular case of  $\sigma_t = 0$ , where by the I (implicit) in DDIM means this is an **implicit probabilistic model**. This is because, now the generated  $\mathbf{x}_0$  from a given  $\mathbf{x}_T = \mathbf{z}$  is no longer random, which we shall see, this has some advantages both theoretically and practically.

#### 4.5. Accelerated generations

It is worth reiterating that, in this section, we aim to eliminate  $p(\mathbf{x}_t|\mathbf{x}_{t-1})$  in the derivation, as a result, all the expressions derived here become dependent on the hyperparameters  $\bar{\alpha}_t$  and  $\bar{\beta}_t$ , whereas  $\alpha_t$  and  $\beta_t$  can be derived from  $\alpha_t = \bar{\alpha}_t/\bar{\alpha}_{t-1}$  and  $\beta_t = \sqrt{1 - \alpha_t^2}$ . From the loss function  $\|\boldsymbol{\varepsilon} - \boldsymbol{\varepsilon}_\theta(\bar{\alpha}_t \mathbf{x}_0 + \bar{\beta}_t \boldsymbol{\varepsilon}, t)\|_2^2$ , it can be seen that the training process is fixed once  $\bar{\alpha}_t$  is specified.

Based on this, it is noticed in DDIM that

##### High-Level Viewpoint 2

The training outcomes of DDPM in fact include the training outcomes of the parameters for any of its sub-sequences.

More specifically, let  $\tau = [\tau_1, \tau_2, \dots, \tau_{\dim(\tau)}]$  being an arbitrary sub-sequence of  $[1, 2, \dots, \tau]$ . If now we train a DDPM with parameters  $\bar{\alpha}_{\tau_1}, \bar{\alpha}_{\tau_2}, \dots, \bar{\alpha}_{\tau_{\dim(\tau)}}$  with  $\dim(\tau)$  steps, then the target functions are a subset of the target functions for the original DDPM model with parameters  $\bar{\alpha}_1, \bar{\alpha}_2, \dots, \bar{\alpha}_\tau$  that is trained for  $\tau$  steps. As such, when the model has been well trained, it actually includes all the parameters for models that consist of any of its arbitrary subsequences.

This means that, we can extract a smaller model of  $\dim(\tau)$  steps, and then sample according to Eq. (4.10) as

$$p(\mathbf{x}_{\tau_{i-1}} | \mathbf{x}_{\tau_i}) \approx \mathcal{N}\left(\mathbf{x}_{\tau_{i-1}}; \frac{\bar{\alpha}_{\tau_{i-1}}}{\bar{\alpha}_{\tau_i}} \left(\mathbf{x}_{\tau_i} - \left(\bar{\beta}_{\tau_i} - \frac{\bar{\alpha}_{\tau_i}}{\bar{\alpha}_{\tau_{i-1}}} \sqrt{\bar{\beta}_{\tau_{i-1}}^2 - \tilde{\sigma}_{\tau_i}^2}\right) \boldsymbol{\varepsilon}_\theta(\mathbf{x}_{\tau_i}, \tau_i)\right), \tilde{\sigma}_{\tau_i}^2 \mathbf{I}\right) \quad (4.13)$$

This is the process of accelerated sampling, which reduces the original  $\tau$  diffusion steps to  $\dim(\tau)$  steps. Notice that we cannot directly replace  $\alpha_t$  in Eq. (4.10) by  $\alpha_{\tau_i}$ , as we mentioned,  $\alpha_t$  is only a derived, auxiliary notation and equivalently, we should have  $\alpha_{\tau_i} = \bar{\alpha}_{\tau_i}/\bar{\alpha}_{\tau_{i-1}}$ . Similarly,  $\bar{\sigma}_{\tau_i}$  is not  $\sigma_{\tau_i}$ , which must now be determined by changing its notations into  $\bar{\alpha}_t$  and  $\bar{\beta}_t$ , followed by replacing  $t$  and  $t-1$  by  $\tau_i$  and  $\tau_{i-1}$ , respectively. For instance,

$$\sigma_t = \frac{\bar{\beta}_{t-1} \beta_t}{\bar{\beta}_t} = \frac{\bar{\beta}_{t-1}}{\bar{\beta}_t} \sqrt{1 - \frac{\bar{\alpha}_t^2}{\bar{\alpha}_{t-1}^2}} \rightarrow \frac{\bar{\beta}_{\tau_{i-1}}}{\bar{\beta}_{\tau_i}} \sqrt{1 - \frac{\bar{\alpha}_{\tau_i}^2}{\bar{\alpha}_{\tau_{i-1}}^2}} = \tilde{\sigma}_{\tau_i}. \quad (4.14)$$

So why don't we just train a model of  $\dim(\tau)$  steps, but must train a  $\tau > \dim(\tau)$  model and

Recall that in the previous section, the result shown in Eq. (4.11) was obtained by assuming

$$p(\mathbf{x}_t|\mathbf{x}_{t-1}) \sim \mathcal{N}\left(\mathbf{x}_{t-1}; \frac{\alpha_t \bar{\beta}_{t-1}^2}{\bar{\beta}_t^2} \mathbf{x}_t, \frac{\bar{\beta}_{t-1}^2 \beta_t^2}{\bar{\beta}_t^2} \mathbf{I}\right)$$

and then apply the Bayes' rule with the known distributions of  $p(\mathbf{x}_t|\mathbf{x}_0)$  and  $p(\mathbf{x}_{t-1}|\mathbf{x}_0)$ . The  $\sigma_t$  parameter can be traced back to the VAE formalism, in which we used the probability  $q(\mathbf{x}_{t-1}|\mathbf{x}_t)$  for the decoder to calculate the KL-divergence. By taking the logarithm of this parameterised normal distribution, we get  $-\log[q(\mathbf{x}_{t-1}|\mathbf{x}_t)] \sim \frac{1}{\sigma_t^2} \|\mathbf{x}_{t-1} - \bar{\boldsymbol{\mu}}(\mathbf{x}_t)\|_2^2$ .



then sample a sub-sequence? Possible explanation is that, on one hand, this enhances the model generalisability. On the other hand, it allows us to try other approaches of accelerated sampling without increasing the training costs.

#### 4.6. Differential equation

We now go back to re-examine the case of  $\sigma_t = 0$ . In this case, Eq. (4.12) can be written equivalently as

$$\frac{\mathbf{x}_t}{\bar{\alpha}_t} - \frac{\mathbf{x}_{t-1}}{\bar{\alpha}_{t-1}} = \left( \frac{\bar{\beta}_t}{\bar{\alpha}_t} - \frac{\bar{\beta}_{t-1}}{\bar{\alpha}_{t-1}} \right) \varepsilon_\theta(\mathbf{x}_t, t). \quad (4.15)$$

When  $T$  is sufficiently large, or when  $\bar{\alpha}_t$  and  $\bar{\alpha}_{t-1}$  are sufficiently small, the above equation becomes the finite-difference form of an ordinary differential equation. More specifically, by introducing an imaginary time variable  $s$ , we have

$$\frac{d}{ds} \left( \frac{\mathbf{x}(s)}{\bar{\alpha}(s)} \right) = \varepsilon_\theta(\mathbf{x}(s), t(s)) \frac{d}{ds} \left( \frac{\bar{\beta}(s)}{\bar{\alpha}(s)} \right). \quad (4.16)$$

Without loss of generality, let  $s \in [0, 1]$ , in which  $s = 0$  corresponds to  $t = 0$  and  $s = 1$  corresponds to  $t = T$ . Note that in the original DDPM paper, the imaginary time was set to  $\frac{\bar{\beta}(s)}{\bar{\alpha}(s)}$ . This is a rather unsuitable choice, as it lies within  $[0, \infty)$ . This unbounded limit is rather not well suitable for numerical solutions.

What we need to do now is to solve for  $\mathbf{x}(0)$  given  $\mathbf{x}(1) \sim \mathcal{N}(0, \mathbf{I})$ . The generation process of DDPM and DDIM corresponding to solving this ordinary differential equation (ODE, Eq. (4.15)) with Euler's method, which is known to be the slowest iterative method. Other approaches such as the Heun method and the R-K method may be the accelerated alternatives. This means that writing the generating process as ODE opens up new avenues to achieve accelerated samplings.

For example, using the default DDPM parameters with  $T = 1000$  and  $\alpha_t = \sqrt{1 - \frac{0.02t}{T}}$ , repeating the process outlined in Section 1, we have

$$\log \bar{\alpha}_t = \sum_{i=k}^t \log \alpha_k = \frac{1}{2} \sum_{i=k}^t \log \left( 1 - \frac{0.02k}{T} \right) < \frac{1}{2} \sum_{i=k}^t \log \left( -\frac{0.02k}{T} \right) = -\frac{0.005t(t+1)}{T}. \quad (4.17)$$

The above estimation is rather good given that every  $\alpha_k$  is very close to 1. On the other hand, since the starting point for this chapter is  $p(\mathbf{x}_t|\mathbf{x}_0)$ , so we should start from  $\bar{\alpha}_t$ . According to the approximation given above, let us now take

$$\bar{\alpha}_t = \exp \left( -\frac{0.005t^2}{T} \right) = \exp \left( -\frac{5t^2}{T^2} \right). \quad (4.18)$$

If we let  $s = \frac{t}{T}$ , then  $s \in [0, 1]$  (which is how we set it in the first place), and  $\bar{\alpha}(s) = e^{-5s^2}$ . Substituting this into Eq. (4.16) and simplify, we get:

$$\frac{d\mathbf{x}(s)}{ds} = 10s \left( \frac{\varepsilon_\theta(\mathbf{x}(s), sT)}{\sqrt{1 - e^{-10s^2}}} - \mathbf{x}(s) \right). \quad (4.19)$$

We can also take  $s = \frac{t^2}{T^2}$ , now we still have  $s \in [0, 1]$  but with  $\bar{\alpha}(s) = e^{-5s}$ . Substituting this into Eq. (4.16) and simplify, we get:

$$\frac{d\mathbf{x}(s)}{ds} = 5 \left( \frac{\varepsilon_\theta(\mathbf{x}(s), \sqrt{s}T)}{\sqrt{1 - e^{-10s}}} - \mathbf{x}(s) \right). \quad (4.20)$$

### 5. The General Framework Based on Stochastic Differential Equation

When writing Section 1 for this series of blog post on diffusion model, some readers had already recommended the paper entitled “Score-Based Generative Modelling through Stochastic Differential Equation”<sup>8</sup> published by Dr Yang Song. This paper has established a rather general theoretical framework for the diffusion model, which has linked up many results derived from DDPM, Stochastic Differential Equations (SDE) and Ordinary Differential Equation (ODE). Although this is a very good paper, it is not well suited for new comers in this field. This

Original blog post see <https://www.spaces.ac.cn/archives/9209>

is because this paper has applied directly, and extensively, results from SDE, Fokker-Planck equations and score matching, which are rather challenging to be understood. However, with the foundations built from the previous few sections, we can now try to understand this particular paper. In the following chapters, we will try to recover the results derived in this paper, with minimum prerequisite knowledge.

### 5.1. Stochastic differentiations

In the DDPM formalism, the diffusion process has been divided into a fixed number of  $T$  steps. Using the analogy of house demolition and rebuilding that was established in Section 1, it means that both the demolition and rebuilding processes have been divided into  $T$  steps beforehand with  $T$  being a model hyperparameter. Such a choice is rather artificial. In reality, the entire process should be a transformative one that is continuous in time, which can be described by an SDE.

With this understanding, the forward process can be described by the following SDE:

$$d\mathbf{x} = f_t(\mathbf{x})dt + g_t d\mathbf{w}. \quad (5.1)$$

Some readers may be rather not familiar with this equation, this is of no problem. The equation can be regarded as the limiting case at  $\Delta t \rightarrow 0$  for the following equation of finite differences:

$$\mathbf{x}_{t+\Delta t} - \mathbf{x}_t = f_t(\mathbf{x})\Delta t + g_t\sqrt{\Delta t}\boldsymbol{\varepsilon}, \quad \boldsymbol{\varepsilon} \sim \mathcal{N}(0, \mathbf{I}). \quad (5.2)$$

In a more straightforward language, if the entire demolition process takes one day, then it is a transformative process from  $t = 0$  to  $t = 1$ , in which every single step can be described by the above equation of finite differences. There is no specific restriction on  $\Delta t$ , except that the smaller  $\Delta t$  is, the better will Eq. (5.2) approximate the original SDE [Eq. (5.1)]. If we choose  $\Delta t = 0.001$ , then it corresponds to  $T = 1000$  in the original DDPM formulation, and similarly  $\Delta t = 0.01$  translates to  $T = 100$ . Therefore, under the viewpoint of a SDE in the continuous time domain, *different choices of  $T$  represent different levels of discretising the SDE*, but should all automatically lead to identical results. As such, *it is not necessary to pre-fix the  $T$  values, but changing it based on the requirement of the accuracy levels for different problems*. Therefore, the fundamental advantage of the SDE formalism of the diffusion model is *to separate theoretical analysis from numerical implementations*. We can conduct theoretical analysis based on the mathematical tools of SDE in the continuous time domain, while implementing it with a suitable discretisation strategy for numerical calculations.

Examining Eq. (5.2) in details, some readers may wonder why the first term in the RHS is  $\mathcal{O}(\Delta t)$ , whereas the second term is  $\mathcal{O}(\sqrt{\Delta t})$ . In another word, why the deterministic term is of a higher order than the random term? This is indeed a puzzling point in SDE. Simply speaking,  $\boldsymbol{\varepsilon}$  always follows a normal distribution if this random term also follows  $\mathcal{O}(\sqrt{\Delta t})$ . Since the normal distribution has a mean of 0 and a standard deviation of  $\mathbf{I}$ , the neighbouring random effect could cancel out each other, and the long-term effect of the random process will only show up if we magnify it to  $\mathcal{O}(\sqrt{\Delta t})$ .

### 5.2. The reverse process

In the language of the probability theory, Eq. (5.2) is equivalent to the conditional probability

$$\begin{aligned} p(\mathbf{x}_{t+\Delta t}|\mathbf{x}_t) &= \mathcal{N}(\mathbf{x}_{t+\Delta t}; \mathbf{x}_t + f_t(\mathbf{x}_t)\Delta t, g_t^2\Delta t\mathbf{I}) \\ &\propto \exp\left(-\frac{\|\mathbf{x}_{t+\Delta t} - \mathbf{x}_t - f_t(\mathbf{x}_t)\Delta t\|_2^2}{2g_t^2\Delta t}\right), \end{aligned} \quad (5.3)$$

which, for simplicity, we have ignored the normalisation constant. Following DDPM, we want to know how to rebuild from the demolition process, i.e., we want to determine  $p(\mathbf{x}_t|\mathbf{x}_{t+\Delta t})$ . Again, from the Bayes's rule,

$$\begin{aligned} p(\mathbf{x}_t|\mathbf{x}_{t+\Delta t}) &= \frac{p(\mathbf{x}_{t+\Delta t}|\mathbf{x}_t)p(\mathbf{x}_t)}{p(\mathbf{x}_{t+\Delta t})} = p(\mathbf{x}_{t+\Delta t}|\mathbf{x}_t)\exp[\log p(\mathbf{x}_t) - \log p(\mathbf{x}_{t+\Delta t})] \\ &\propto \exp\left(-\frac{\|\mathbf{x}_{t+\Delta t} - \mathbf{x}_t - f_t(\mathbf{x}_t)\Delta t\|_2^2}{2g_t^2\Delta t} + \log p(\mathbf{x}_t) - \log p(\mathbf{x}_{t+\Delta t})\right). \end{aligned} \quad (5.4)$$

It can be seen that, when  $\Delta t$  is sufficiently small,  $p(\mathbf{x}_t|\mathbf{x}_{t+\Delta t})$  will be non-zero only when  $\mathbf{x}_{t+\Delta t}$  is very close to  $\mathbf{x}_t$ . Similar argument applies to  $p(\mathbf{x}_{t+\Delta t}|\mathbf{x}_t)$ .

This subsection discuss how the diffusion model, that was established in the discrete time domain, can be generalised into the continuous time domain.

In this case, we can ignore  $\log p(\mathbf{x}_t) - \log p(\mathbf{x}_{t+\Delta t}) \rightarrow 0$ , then the first term in Eq. (5.4) dominates. When  $\Delta t \rightarrow 0$ , the fraction can diverge to  $\infty$ , making  $p(\mathbf{x}_t|\mathbf{x}_{t+\Delta t})$  vanishes. This can be avoided if  $\mathbf{x}_t$  is sufficiently close to  $\mathbf{x}_{t+\Delta t}$ , such that  $\|\mathbf{x}_{t+\Delta t} - \mathbf{x}_t\|_2^2 \sim \Delta t$  (i.e. they are of similar magnitude, so that  $\|\mathbf{x}_{t+\Delta t} - \mathbf{x}_t\|_2^2 / 2g_t^2\Delta t \sim 1$ , to avoid vanishing transition probability).

In this case, we can make a Taylor expansion for  $\log p(\mathbf{x}_{t+\Delta t})$  as an approximation when  $\mathbf{x}_{t+\Delta t}$  is sufficiently close to  $\mathbf{x}_t$ :

$$\log p(\mathbf{x}_{t+\Delta t}) \approx \log p(\mathbf{x}_t) + (\mathbf{x}_{t+\Delta t} - \mathbf{x}_t) \nabla_{\mathbf{x}_t} \log p(\mathbf{x}_t) + \Delta t \frac{\partial}{\partial t} \log p(\mathbf{x}_t). \quad (5.5)$$

Note that the last  $\frac{\partial}{\partial t}$ -term cannot be ignored, this is because  $p(\mathbf{x}_t)$  is the probability of the random variable being equal to  $\mathbf{x}_t$  at time  $t$ , and  $p(\mathbf{x}_{t+\Delta t})$  is the probability of the random variable being equal to  $\mathbf{x}_{t+\Delta t}$  at time  $t + \Delta t$ . In this case, we must include the partial derivative with respect to the time variable. With this, we can now substitute the expression of  $\log p(\mathbf{x}_{t+\Delta t})$  from Eq. (5.5) into Eq. (5.4) and complete the square, we then have,

$$\begin{aligned} p(\mathbf{x}_t | \mathbf{x}_{t+\Delta t}) &\propto \exp \left( -\frac{\|\mathbf{x}_{t+\Delta t} - \mathbf{x}_t - f_t(\mathbf{x}_t) \Delta t\|_2^2}{2g_t^2 \Delta t} - (\mathbf{x}_{t+\Delta t} - \mathbf{x}_t) \nabla_{\mathbf{x}_t} \log p(\mathbf{x}_t) - \Delta t \frac{\partial}{\partial t} \log p(\mathbf{x}_t) \right) \\ &\propto \exp \left( -\frac{\|\mathbf{x}_{t+\Delta t} - \mathbf{x}_t - f_t(\mathbf{x}_t) \Delta t\|_2^2 - 2g_t^2 (\mathbf{x}_{t+\Delta t} - \mathbf{x}_t) \nabla_{\mathbf{x}_t} \log p(\mathbf{x}_t) \Delta t - 2g_t^2 \Delta t^2 \frac{\partial}{\partial t} \log p(\mathbf{x}_t)}{2g_t^2 \Delta t} \right) \\ &\propto \exp \left( -\frac{\|\mathbf{x}_{t+\Delta t} - \mathbf{x}_t - [f_t(\mathbf{x}_t) - g_t^2 \nabla_{\mathbf{x}_t} \log p(\mathbf{x}_t)] \Delta t\|_2^2}{2g_t^2 \Delta t} - \frac{1}{2} \nabla_{\mathbf{x}_t} \log p(\mathbf{x}_t) \Delta t + \frac{\partial}{\partial t} \log p(\mathbf{x}_t) \Delta t \right) \\ &\propto \exp \left( -\frac{\|\mathbf{x}_{t+\Delta t} - \mathbf{x}_t - [f_t(\mathbf{x}_t) - g_t^2 \nabla_{\mathbf{x}_t} \log p(\mathbf{x}_t)] \Delta t\|_2^2}{2g_t^2 \Delta t} + \mathcal{O}(\Delta t) \right). \end{aligned} \quad (5.6)$$

When  $\Delta t \rightarrow 0$ ,  $\mathcal{O}(\Delta t)$  plays a negligible role, in this case,

$$\begin{aligned} p(\mathbf{x}_t | \mathbf{x}_{t+\Delta t}) &\approx \exp \left( -\frac{\|\mathbf{x}_{t+\Delta t} - \mathbf{x}_t - [f_t(\mathbf{x}_t) - g_t^2 \nabla_{\mathbf{x}_t} \log p(\mathbf{x}_t)] \Delta t\|_2^2}{2g_t^2 \Delta t} \right) \\ &\approx \exp \left( -\frac{\|\mathbf{x}_t - \mathbf{x}_{t+\Delta t} - [f_{t+\Delta t}(\mathbf{x}_{t+\Delta t}) - g_{t+\Delta t}^2 \nabla_{\mathbf{x}_{t+\Delta t}} \log p(\mathbf{x}_{t+\Delta t})] \Delta t\|_2^2}{2g_{t+\Delta t}^2 \Delta t} \right). \end{aligned} \quad (5.7)$$

This shows that  $p(\mathbf{x}_t | \mathbf{x}_{t+\Delta t})$  is approximately a normal distribution with the following mean:

$$\boldsymbol{\mu} = \mathbf{x}_{t+\Delta t} - [f_{t+\Delta t}(\mathbf{x}_{t+\Delta t}) - g_{t+\Delta t}^2 \nabla_{\mathbf{x}_{t+\Delta t}} \log p(\mathbf{x}_{t+\Delta t})] \Delta t,$$

and standard deviation of  $\boldsymbol{\sigma} = g_{t+\Delta t}^2 \Delta t \mathbf{I}$ . Taking the limit of  $\Delta t \rightarrow 0$ , then the corresponding SDE for the reverse process is

$$d\mathbf{x} = [f_t(\mathbf{x}_t) - g_t^2 \nabla_{\mathbf{x}} \log p_t(\mathbf{x})] dt + g_t d\mathbf{w}. \quad (5.8)$$

This first occurred in the paper entitled ‘‘Reverse-Time Diffusion Equation Model’’<sup>9</sup>. Here, the subscript  $t$  has been included in  $p$  to signify that this is the distribution at time  $t$ .

### 5.3. Score matching

Now we have obtained the SDE for the reverse diffusion process [Eq. (5.8)], if we further knows the quantity  $\nabla_{\mathbf{x}} \log p_t(\mathbf{x})$ , then we can rebuild the sample from the following discretised version of the SDE:

$$\mathbf{x}_t - \mathbf{x}_{t+\Delta t} = [f_{t+\Delta t}(\mathbf{x}_{t+\Delta t}) - g_{t+\Delta t}^2 \nabla_{\mathbf{x}_{t+\Delta t}} \log p(\mathbf{x}_{t+\Delta t})] \Delta t - g_{t+\Delta t} \sqrt{\Delta t} \boldsymbol{\varepsilon}, \quad (5.9)$$

in which  $\boldsymbol{\varepsilon} \sim \mathcal{N}(0, \mathbf{I})$ . This then completes the derivation of the diffusion generative model. In this case, how would we compute  $\nabla_{\mathbf{x}} \log p_t(\mathbf{x})$ ? Note that the probability distribution  $p_t(\mathbf{x})$  at  $t$  is identical to  $p(\mathbf{x}_t)$  in the previous sections, it represents the marginal distribution at time  $t$ . This means that we can write the following:

$$p(\mathbf{x}_t | \mathbf{x}_0) = \lim_{\Delta t \rightarrow 0} \int \cdots \int p(\mathbf{x}_t | \mathbf{x}_{t-\Delta t}) p(\mathbf{x}_{t-\Delta t} | \mathbf{x}_{t-2\Delta t}) \cdots p(\mathbf{x}_{\Delta t} | \mathbf{x}_0) d\mathbf{x}_{t-\Delta t} d\mathbf{x}_{t-2\Delta t} d\mathbf{x}_{\Delta t}, \quad (5.10)$$

which may be computed directly. This is because in real applications, we will design a model in which  $p(\mathbf{x}_t | \mathbf{x}_0)$  has an analytical solution. This is possible when  $f_t(\mathbf{x})$  is a linear function

in  $\mathbf{x}$ . With this, we have

$$p(\mathbf{x}_t) = \int p(\mathbf{x}_t|\mathbf{x}_0)\tilde{p}(\mathbf{x}_0)d\mathbf{x}_0 = \mathbb{E}_{\mathbf{x}_0}[p(\mathbf{x}_t|\mathbf{x}_0)], \quad (5.11)$$

and

$$\nabla_{\mathbf{x}_t} \log p(\mathbf{x}_t) = \frac{\mathbb{E}_{\mathbf{x}_0}[\nabla_{\mathbf{x}_t} p(\mathbf{x}_t|\mathbf{x}_0)]}{\mathbb{E}_{\mathbf{x}_0}[p(\mathbf{x}_t|\mathbf{x}_0)]} = \frac{\mathbb{E}_{\mathbf{x}_0}[p(\mathbf{x}_t|\mathbf{x}_0)\nabla_{\mathbf{x}_t} \log p(\mathbf{x}_t|\mathbf{x}_0)]}{\mathbb{E}_{\mathbf{x}_0}[p(\mathbf{x}_t|\mathbf{x}_0)]}. \quad (5.12)$$

It can be seen that the last expression in Eq. (5.12) contains a weighted average of  $\nabla_{\mathbf{x}_t} \log p(\mathbf{x}_t|\mathbf{x}_0)$ . As we assume  $p(\mathbf{x}_t|\mathbf{x}_0)$  has an analytical solution, in principle, Eq. (5.12) will also become computable. However, practically, it involves an average across all the training data  $\mathbf{x}_0$ , which is both heavy in computational costs and lack of good generalisability. Hence we would like to train a neural network  $s_{\theta}(\mathbf{x}_t, t)$ , which can be used for computing  $\nabla_{\mathbf{x}_t} \log p(\mathbf{x}_t|\mathbf{x}_0)$ .

Some readers may be familiar with the following expression:

$$\mathbb{E}[\mathbf{x}] = \underset{\boldsymbol{\mu}}{\operatorname{argmin}} \mathbb{E}_{\mathbf{x}} [\|\boldsymbol{\mu} - \mathbf{x}\|_2^2], \quad (5.13)$$

meaning that, in order to make  $\boldsymbol{\mu}$  to equal to be the mean of  $\mathbf{x}$ , we only need to minimise the mean of  $\|\boldsymbol{\mu} - \mathbf{x}\|_2^2$ . Similarly, in order to equate  $s_{\theta}(\mathbf{x}_t, t)$  to the weighted average of  $\nabla_{\mathbf{x}_t} \log p(\mathbf{x}_t|\mathbf{x}_0)$  (*i.e.*  $\nabla_{\mathbf{x}_t} \log p(\mathbf{x}_t)$ ), we only need to minimise the weighted average of  $\|s_{\theta}(\mathbf{x}_t, t) - \nabla_{\mathbf{x}_t} \log p(\mathbf{x}_t|\mathbf{x}_0)\|_2^2$ , *i.e.*

$$\frac{\mathbb{E}_{\mathbf{x}_0} [p(\mathbf{x}_t|\mathbf{x}_0)\|s_{\theta}(\mathbf{x}_t, t) - \nabla_{\mathbf{x}_t} \log p(\mathbf{x}_t|\mathbf{x}_0)\|_2^2]}{\mathbb{E}_{\mathbf{x}_0}[p(\mathbf{x}_t|\mathbf{x}_0)]}. \quad (5.14)$$

Again, the purpose here is to seek a way to drop  $\mathbf{x}_0$  in  $\nabla_{\mathbf{x}_t} \log p(\mathbf{x}_t|\mathbf{x}_0)$  for the sampling process such that it is not conditioned on  $\mathbf{x}_0$ .

The denominator  $\mathbb{E}_{\mathbf{x}_0}[p(\mathbf{x}_t|\mathbf{x}_0)]$  will only contribute to a weight to the loss function, which can be dropped in the following discussions without affecting the final solution. Finally, we integrate the loss function over  $\mathbf{x}_t$ , meaning we must minimise the loss function over every  $\mathbf{x}_t$ , from which we reached the final form of the loss function:

$$\begin{aligned} & \int \mathbb{E}_{\mathbf{x}_0} [p(\mathbf{x}_t|\mathbf{x}_0)\|s_{\theta}(\mathbf{x}_t, t) - \nabla_{\mathbf{x}_t} \log p(\mathbf{x}_t|\mathbf{x}_0)\|_2^2] d\mathbf{x}_t \\ &= \mathbb{E}_{\mathbf{x}_0, \mathbf{x}_t \sim p(\mathbf{x}_t|\mathbf{x}_0)\tilde{p}(\mathbf{x}_0)} [\|s_{\theta}(\mathbf{x}_t, t) - \nabla_{\mathbf{x}_t} \log p(\mathbf{x}_t|\mathbf{x}_0)\|_2^2]. \end{aligned} \quad (5.15)$$

This expression provides a physically meaningful explanation to the loss function, which is not necessarily the form that is numerically implemented in most cases.

Eq. (5.15) represents the loss function for the “**(conditional) score-matching**” **generative model**. The analytical solutions to the denoising loss function that was derived in Section 2 can be regarded as a special case for this result. The concept of score matching could be first traced back to the 2005 paper entitled “*Estimation of Non-Normalised Statistical Models by Score Matching*”<sup>10</sup>. As for the conditional score matching, the author believes that the earliest paper is “*A Connection between Score Matching and Denoising Autoencoder*”<sup>11</sup> that was published in 2011.

Although the results presented in this section is the same as the score matching, in our derivation, we did not use the concept of scores. Instead, the results come naturally from manipulating the loss function. The author thinks such an approach is more inspirational to the readers to establish its connection to the generative models. Hopefully, such an approach can also reduce the difficulties for understanding the score matching model.

#### 5.4. The final results

Hence, we have now established the general procedure of a diffusion generative model:

1. Define the forward (demolition) process based on the SDE [Eq. (5.1)].
2. Find the expression for  $p(\mathbf{x}_t|\mathbf{x}_0)$ .
3. Train the score function  $s_{\theta}(\mathbf{x}_t, t)$  using the loss function Eq. (5.15) (score-matching).
4. Substitute  $s_{\theta}(\mathbf{x}_t, t)$  into Eq. (5.8), replacing the term  $\nabla_{\mathbf{x}_t} \log p(\mathbf{x}_t)$  to complete the reverse process of re-building.

Most readers may be frightened by the term such as the stochastic differential equations. However, fundamentally, SDE was only used as a bridge for our subsequent discussions. Once the SDE has been discretised into Eq. (5.2) and Eq. (5.3), we can basically forget about SDE, making it much easier to be understood conceptually.

It is also not difficult to see that, it is easy to define an SDE [Eq. (5.1)], but it is rather challenging to solve for  $p(\mathbf{x}_t|\mathbf{x}_0)$  from it. In the remaining part of the original paper<sup>8</sup>, it focused on deriving the results for two practical examples. However, if solving for  $p(\mathbf{x}_t|\mathbf{x}_0)$  from a pre-defined SDE is too difficult, it will be more trivial to first define  $p(\mathbf{x}_t|\mathbf{x}_0)$ , just like the case of DDIM, and then deduce the corresponding SDE.

For example, let us first define

$$p(\mathbf{x}_t|\mathbf{x}_0) = \mathcal{N}(\mathbf{x}_t; \bar{\alpha}_t \mathbf{x}_0, \bar{\beta}_t^2 \mathbf{I}). \quad (5.16)$$

Without loss of generality, let us also assume the starting point is  $t = 0$  and the end point being  $t = 1$ , then the corresponding boundary conditions for  $\bar{\alpha}_t$  and  $\bar{\beta}_t$  are:

$$\bar{\alpha}_0 = 1, \quad \bar{\alpha}_1 = 0, \quad \bar{\beta}_0 = 0, \quad \bar{\beta}_1 = 1. \quad (5.17)$$

In reality, we only need to supply these boundary conditions approximately. For instance, in the previous section, DDPM is equivalent to the choice of  $\bar{\alpha}_t = e^{-5t^2}$ , such that when  $t = 1$ ,  $e^{-5} \approx 0$ .

Once we have  $p(\mathbf{x}_t|\mathbf{x}_0)$ , we can deduce Eq. (5.1) in the reverse order, which, in principle, is to seek a solution for  $p(\mathbf{x}_{t+\Delta t}|\mathbf{x}_t)$ , which satisfy the condition of

$$p(\mathbf{x}_{t+\Delta t}|\mathbf{x}_0) = \int p(\mathbf{x}_{t+\Delta t}|\mathbf{x}_t)p(\mathbf{x}_t|\mathbf{x}_0)d\mathbf{x}_t. \quad (5.18)$$

Considering only the linear solution, then we have

$$d\mathbf{x} = f_t \mathbf{x} dt + g_t d\mathbf{w}. \quad (5.19)$$

Similar to Section 4, we have the following:

Notation	Representative Distribution	Sampling
$p(\mathbf{x}_{t+\Delta t} \mathbf{x}_0)$	$\mathcal{N}(\mathbf{x}_{t+\Delta t}; \bar{\alpha}_{t+\Delta t} \mathbf{x}_0, \bar{\beta}_{t+\Delta t}^2 \mathbf{I})$	$\mathbf{x}_{t+\Delta t} = \bar{\alpha}_{t+\Delta t} \mathbf{x}_0 + \bar{\beta}_{t+\Delta t} \boldsymbol{\varepsilon}$
$p(\mathbf{x}_t \mathbf{x}_0)$	$\mathcal{N}(\mathbf{x}_t; \bar{\alpha}_t \mathbf{x}_0, \bar{\beta}_t^2 \mathbf{I})$	$\mathbf{x}_t = \bar{\alpha}_t \mathbf{x}_0 + \bar{\beta}_t \boldsymbol{\varepsilon}_1$
$p(\mathbf{x}_{t+\Delta t} \mathbf{x}_0)$	$\mathcal{N}(\mathbf{x}_{t+\Delta t}; \mathbf{x}_t + f_t(\mathbf{x}_t)\Delta t, g_t^2 \Delta t \mathbf{I})$	$\mathbf{x}_{t+\Delta t} = (1 + f_t \Delta t) \mathbf{x}_t + g_t \sqrt{\Delta t} \boldsymbol{\varepsilon}_2$
$\int p(\mathbf{x}_{t+\Delta t} \mathbf{x}_t) \cdot p(\mathbf{x}_t \mathbf{x}_0) d\mathbf{x}_t$		$\mathbf{x}_{t+\Delta t}$ $= (1 + f_t \Delta t) \mathbf{x}_t + g_t \sqrt{\Delta t} \boldsymbol{\varepsilon}_2$ $= (1 + f_t \Delta t) (\bar{\alpha}_t \mathbf{x}_0 + \bar{\beta}_t \boldsymbol{\varepsilon}_1) + g_t \sqrt{\Delta t} \boldsymbol{\varepsilon}_2$ $= (1 + f_t \Delta t) \bar{\alpha}_t \mathbf{x}_0 + ((1 + f_t \Delta t) \bar{\beta}_t \boldsymbol{\varepsilon}_1 + g_t \sqrt{\Delta t} \boldsymbol{\varepsilon}_2)$

From here, we get the

$$\begin{aligned} \bar{\alpha}_{t+\Delta t} &= (1 + f_t \Delta t) \bar{\alpha}_t \\ \bar{\beta}_{t+\Delta t}^2 &= (1 + f_t \Delta t)^2 \bar{\beta}_t^2 + g_t^2 \Delta t. \end{aligned} \quad (5.20)$$

By letting  $\Delta t \rightarrow 0$ , then we can solve for  $f_t$  and  $g_t$  as

$$\begin{aligned} f_t &= \frac{d}{dt}(\ln \bar{\alpha}_t) = \frac{1}{\bar{\alpha}_t} \frac{d\bar{\alpha}_t}{dt} \\ g_t &= \bar{\alpha}_t^2 \frac{d}{dt} \left( \frac{\bar{\beta}_t^2}{\bar{\alpha}_t^2} \right) = 2\bar{\alpha}_t \bar{\beta}_t \frac{d}{dt} \left( \frac{\bar{\beta}_t}{\bar{\alpha}_t} \right). \end{aligned} \quad (5.21)$$

If we take  $\bar{\alpha}_t \equiv 1$ , then the results corresponds to the variance-exploding SDE (VE-SDE). On the other hand, if  $\bar{\alpha}_t^2 + \bar{\beta}_t^2 = 1$ , this corresponds to the variance-preserving SDE (VP-SDE).

The first equation may be derived as following. From Eq. (5.20) we have

$$\bar{\alpha}_t + d\bar{\alpha}_t = (1 + f_t dt) \bar{\alpha}_t$$

$$d\bar{\alpha}_t = f_t dt \bar{\alpha}_t$$

$$\frac{d\bar{\alpha}_t}{\bar{\alpha}_t} = f_t dt$$

As for the loss function, now we have

$$\nabla_{\mathbf{x}_t} \log p(\mathbf{x}_t | \mathbf{x}_0) = -\frac{\mathbf{x}_t - \bar{\alpha}_t \mathbf{x}_0}{\bar{\beta}_t^2} = -\frac{\boldsymbol{\varepsilon}}{\bar{\beta}_t}, \quad (5.22)$$

in which the second equal sign comes from the fact that  $\mathbf{x}_t = \bar{\alpha}_t \mathbf{x}_0 + \bar{\beta}_t \boldsymbol{\varepsilon}$ . To align with the previous results, let  $s_{\boldsymbol{\theta}} = -\frac{1}{\bar{\beta}_t} \boldsymbol{\varepsilon}_{\boldsymbol{\theta}}(\mathbf{x}_t, t)$ , then Eq. (5.15) becomes

$$\frac{1}{\bar{\beta}_t^2} \mathbb{E}_{\mathbf{x}_0 \sim \bar{p}(\mathbf{x}_0), \boldsymbol{\varepsilon} \sim \mathcal{N}(0, \mathbf{I})} [\|\boldsymbol{\varepsilon}_{\boldsymbol{\theta}}(\bar{\alpha}_t \mathbf{x}_0 + \bar{\beta}_t \boldsymbol{\varepsilon}, t) - \boldsymbol{\varepsilon}\|_2^2]. \quad (5.23)$$

This is the same loss function as the DDPM after ignoring the coefficients. By replacing  $\nabla_{\mathbf{x}_{t+\Delta t}} \log p(\mathbf{x}_{t+\Delta t})$  with  $-\frac{1}{\bar{\beta}_{t+\Delta t}} \boldsymbol{\varepsilon}_{\boldsymbol{\theta}}(\mathbf{x}_{t+\Delta t}, t + \Delta t)$ , the result has the same form (to the first-order approximation) as for the reverse sampling process in DDPM.

## 6. The General Framework Based on Ordinary Differential Equation

In the previous chapter, we have discussed the paper entitled “*Score-Based Generative Modelling through Stochastic Differential Equation*” by Dr Yang Song<sup>8</sup>. However, as the title suggested, we had only discussed the part in the original paper that is related to the SDE, but have missed out the part that is related to the discussion around the probability flow ordinary differential equations (ODE), which will be discussed in details in this section.

In fact, the ODE part only constitutes a small portion of the original paper, but we need to discuss it with a separate section. This is because, after thinking about the derivations for a long time, there is no way that it can bypass the Fokker-Planck equation in the whole process. In this case, we must spend some lengths in this section to discuss the Fokker-Planck equation first, before moving on to the main topic of ODE.

### 6.1. Think again

Let us first summarise the results from the previous section. First of all, we have defined a forward (demolition) process with an SDE:

$$d\mathbf{x} = f_t(\mathbf{x})dt + g_t d\mathbf{w}. \quad (6.1)$$

Then, we have derived the SDE for the corresponding reverse (building) process:

$$d\mathbf{x} = [f_t(\mathbf{x}) - g_t^2 \nabla_{\mathbf{x}} \log p_t(\mathbf{x})] dt + g_t d\mathbf{w}. \quad (6.2)$$

Finally, we use a neural network  $s_{\boldsymbol{\theta}}(\mathbf{x}, t)$  to approximate  $\nabla_{\mathbf{x}} \log p_t(\mathbf{x})$  with the following loss function (score-matching):

$$\mathbb{E}_{\mathbf{x}_0, \mathbf{x}_t \sim p(\mathbf{x}_t | \mathbf{x}_0) \bar{p}(\mathbf{x}_0)} [\|s_{\boldsymbol{\theta}}(\mathbf{x}_t, t) - \nabla_{\mathbf{x}_t} \log p(\mathbf{x}_t | \mathbf{x}_0)\|_2^2]. \quad (6.3)$$

At this point, we have established the general framework for training and inferencing the diffusion model, which is the most generalised framework for DDPM. However, as introduced Section 4, is there a similar higher level view on SDE as the generalisation for DDPM? The answer is yes, which is the probability flow ODE to be discussed here.

### 6.2. The Dirac function

What have we achieved in DDIM? Simply speaking, it was found that the training of DDPM is targeted at  $p(\mathbf{x}_t | \mathbf{x}_0)$ , that is independent from the transition probability  $p(\mathbf{x}_t | \mathbf{x}_{t-1})$ . Therefore, DDIM starts from  $p(\mathbf{x}_t | \mathbf{x}_0)$  to seek for more general solutions for  $p(\mathbf{x}_{t-1} | \mathbf{x}_t, \mathbf{x}_0)$  and  $p(\mathbf{x}_t | \mathbf{x}_{t-1}, \mathbf{x}_0)$ . The idea of probability flow ODE is similar, which asks the following question:

In the SDE framework, for a fixed  $p(\mathbf{x}_t)$ , how many different choices of  $p(\mathbf{x}_{t+\Delta t} | \mathbf{x}_t)$  are possible, or, can we find different SDEs for the forward process?

To answer this question, similar to the previous section, let us first rewrite the SDE [Eq. (6.1)] in its finite-difference form:

$$\mathbf{x}_{t+\Delta t} = \mathbf{x}_t + f_t(\mathbf{x})\Delta t + g_t \sqrt{\Delta t} \boldsymbol{\varepsilon}, \quad \boldsymbol{\varepsilon} \sim \mathcal{N}(0, \mathbf{I}). \quad (6.4)$$

Original blog post see <https://www.spaces.ac.cn/archives/9228>

Eq. (6.4) describes the relationship amongst the random variables  $\mathbf{x}_{t+\Delta t}$ ,  $\mathbf{x}_t$  and  $\boldsymbol{\varepsilon}$ . Although we can easily find the expectation values for both sides, this is not what we want, as we are after the distribution  $p(\mathbf{y}_t)$  and the relationship that it should satisfy. To achieve this, we can use the Dirac function:

$$p(\mathbf{x}) = \int \delta(\mathbf{x} - \mathbf{y})p(\mathbf{y})d\mathbf{y} = \mathbb{E}_{\mathbf{y}}[\delta(\mathbf{x} - \mathbf{y})]. \quad (6.5)$$

To strictly define the Dirac function, we need to use the tools from functional analysis. But generally, we can obtain the correct result if we just treat it as a normal function. From Eq. (6.5), we also know that the following holds for any  $f(\mathbf{x})$ :

$$p(\mathbf{x})f(\mathbf{x}) = \int \delta(\mathbf{x} - \mathbf{y})p(\mathbf{y})f(\mathbf{y})d\mathbf{y} = \mathbb{E}_{\mathbf{y}}[f(\mathbf{y})\delta(\mathbf{x} - \mathbf{y})]. \quad (6.6)$$

Taking the derivative of both sides, we have

$$\nabla_{\mathbf{x}}[p(\mathbf{x})f(\mathbf{x})] = \mathbb{E}_{\mathbf{y}}[\nabla_{\mathbf{x}}(f(\mathbf{y})\delta(\mathbf{x} - \mathbf{y}))] = \mathbb{E}_{\mathbf{y}}[f(\mathbf{y}) \cdot \nabla_{\mathbf{x}}\delta(\mathbf{x} - \mathbf{y})]. \quad (6.7)$$

This is an important relationship that will be used latter. It can be seen that, fundamentally, it should that the derivative of a Dirac function can be transferred to the function that it multiplies to via integration.

### 6.3. The Fokker-Planck equation

With the preliminary knowledges discussed above, we can now write out the following based on Eq. (6.4):

$$\begin{aligned} & \delta(\mathbf{x} - \mathbf{x}_{t+\Delta t}) \\ &= \delta\left(\mathbf{x} - \mathbf{x}_t - f_t(\mathbf{x})\Delta t - g_t\sqrt{\Delta t}\boldsymbol{\varepsilon}\right) \\ &= \delta(\mathbf{x} - \mathbf{x}_t) - \left(f_t(\mathbf{x})\Delta t + g_t\sqrt{\Delta t}\boldsymbol{\varepsilon}\right) \nabla_{\mathbf{x}}\delta(\mathbf{x} - \mathbf{x}_t) + \frac{1}{2}\left(g_t\sqrt{\Delta t}\boldsymbol{\varepsilon}\nabla_{\mathbf{x}}\right)^2 \delta(\mathbf{x} - \mathbf{x}_t), \end{aligned} \quad (6.8)$$

in which we have Taylor expanded  $\Delta(\mathbf{x})$  as a normal function, and kept the terms that are not exceeding  $\mathcal{O}(\Delta t)$ . Now we take the expectation values of both sides:

$$\begin{aligned} & p_{t+\Delta t}(\mathbf{x}) \\ &= \mathbb{E}_{\mathbf{x}_t+\Delta t}[\delta(\mathbf{x} - \mathbf{x}_{t+\Delta t})] \\ &\approx \mathbb{E}_{\mathbf{x}_t, \boldsymbol{\varepsilon}}\left[\delta(\mathbf{x} - \mathbf{x}_t) - \left(f_t(\mathbf{x})\Delta t + g_t\sqrt{\Delta t}\boldsymbol{\varepsilon}\right) \nabla_{\mathbf{x}}\delta(\mathbf{x} - \mathbf{x}_t) + \frac{1}{2}\left(g_t\sqrt{\Delta t}\boldsymbol{\varepsilon}\nabla_{\mathbf{x}}\right)^2 \delta(\mathbf{x} - \mathbf{x}_t)\right] \\ &= \mathbb{E}_{\mathbf{x}_t}\left[\delta(\mathbf{x} - \mathbf{x}_t) - f_t(\mathbf{x}_t)\Delta t \nabla_{\mathbf{x}}\delta(\mathbf{x} - \mathbf{x}_t) + \frac{1}{2}g_t^2\Delta t \nabla_{\mathbf{x}} \cdot \nabla_{\mathbf{x}}\delta(\mathbf{x} - \mathbf{x}_t)\right] \\ &= p_t(\mathbf{x}) - \nabla_{\mathbf{x}}[f_t(\mathbf{x})\Delta t p_t(\mathbf{x})] + \frac{1}{2}g_t^2\Delta t \nabla_{\mathbf{x}} \cdot \nabla_{\mathbf{x}}p_t(\mathbf{x}). \end{aligned} \quad (6.9)$$

Now use the fact that  $\boldsymbol{\varepsilon} \sim \mathcal{N}(0, \mathbf{I})$ , such that  $\mathbb{E}_{\boldsymbol{\varepsilon}}[\boldsymbol{\varepsilon}] = 0$

Now if we divide both side by  $\Delta t$  and take the limit of  $\Delta t \rightarrow 0$ , we then get the Fokker-Planck equation corresponding to Eq. (6.1):

$$\frac{\partial}{\partial t}p_t(\mathbf{x}) = -\nabla_{\mathbf{x}}f_t(\mathbf{x})p_t(\mathbf{x}) + \frac{1}{2}g_t^2\nabla_{\mathbf{x}} \cdot \nabla_{\mathbf{x}}p_t(\mathbf{x}), \quad (6.10)$$

which is the partial differential equation that describes the time-dependent evolution of the marginal probability distribution.

### 6.4. Equivalent transformation

There is no need to worry about the appearance of another partial differential equation, as we are not going to seek for its solution but only borrow it to perform an equivalent transformation. In Eq. (6.10), we can add in another function  $\sigma_t^2$  to the second term and then subtract it out again. Providing that  $\sigma_t^2 \leq g_t^2$ , Eq. (6.10) is fully equivalent to

$$\frac{\partial}{\partial t}p_t(\mathbf{x}) = -\nabla_{\mathbf{x}}\left[f_t(\mathbf{x})p_t(\mathbf{x}) - \frac{1}{2}(g_t^2 - \sigma_t^2)\nabla_{\mathbf{x}}p_t(\mathbf{x})\right] + \frac{1}{2}\sigma_t^2\nabla_{\mathbf{x}} \cdot \nabla_{\mathbf{x}}p_t(\mathbf{x})$$



$$= -\nabla_x \left[ \left( f_t(\mathbf{x}) - \frac{1}{2}(g_t^2 - \sigma_t^2) \nabla_x \log p_t(\mathbf{x}) \right) p_t(\mathbf{x}) \right] + \frac{1}{2} \sigma_t^2 \nabla_x \cdot \nabla_x p_t(\mathbf{x}). \quad (6.11)$$

In the second line, we have used the fact that  $\nabla_x \log p(\mathbf{x}) \cdot p(\mathbf{x}) = \left[ \frac{1}{p(\mathbf{x})} \nabla_x p(\mathbf{x}) \right] p(\mathbf{x}) = \nabla_x p(\mathbf{x})$ .

In terms of the functional form, Eq. (6.11) is equivalent to changing the term of  $f_t(\mathbf{x})$  in Eq. (6.10) to  $f_t(\mathbf{x}) - \frac{1}{2}(g_t^2 - \sigma_t^2) \nabla_x \log p_t(\mathbf{x})$ , and  $g_t^2$  into  $\sigma_t^2$ . Since Eq. (6.10) is to be matched with Eq. (6.1), this means that Eq. (6.11) is equivalent to:

$$d\mathbf{x} = \left( f_t(\mathbf{x}) - \frac{1}{2}(g_t^2 - \sigma_t^2) \nabla_x \log p_t(\mathbf{x}) \right) dt + \sigma_t d\mathbf{w}. \quad (6.12)$$

However, do not forget that Eq. (6.10) is fully equivalent to Eq. (6.11), this means that the corresponding marginal distributions  $p_t(\mathbf{x})$  that come from Eq. (6.1) and Eq. (6.12) are also fully equivalent to each other. **This means that we have two forward processes with different variances, which will produce identical marginal distributions!**

This result is also equivalent to an updated version of DDIM, which we shall prove it later, that Eq. (6.12) is fully equivalent to DDIM when  $f_t(\mathbf{x})$  is a linear function. Furthermore, according to the SDE results derived in the previous section, the corresponding reverse SDE for Eq. (6.12) is

$$d\mathbf{x} = \left( f_t(\mathbf{x}) - \frac{1}{2}(g_t^2 + \sigma_t^2) \nabla_x \log p_t(\mathbf{x}) \right) dt + \sigma_t d\mathbf{w}. \quad (6.13) \quad \text{Not quite sure how to derive this.}$$

## 6.5. Neural ODE

**The main advantage of Eq. (6.12) is that it allows us to change the variance during the sampling process.** Here, we consider the extreme case of  $\sigma_t \equiv 0$ , in which case the SDE falls back to an ODE:

$$d\mathbf{x} = \left( f_t(\mathbf{x}) - \frac{1}{2}g_t^2 \nabla_x \log p_t(\mathbf{x}) \right) dt. \quad (6.14)$$

This is the so-called **Probability Flow ODE**. Practically, since  $\nabla_x \log p_t(\mathbf{x})$  needs to be approximated by a neural network  $s_\theta(\mathbf{x}, t)$ , so the above equation also corresponds to a neural ODE.

Why do we need to investigate the special case when  $\sigma_t \equiv 0$ ? This is because now the propagation of our samples are noiseless, meaning we now have a well-defined transformation from  $\mathbf{x}_0$  to  $\mathbf{x}_T$ . In this case, we can obtain the inverse transformation from  $\mathbf{x}_T$  to  $\mathbf{x}_0$  by directly solving the ODE in the reverse order, which is also a definitive process. This is consistent with the **flow models**, which transforms noises into samples through an invertible transformation. As such, by using probability flow ODE allows us to link the diffusion model with the flow model. For example, in the original paper<sup>8</sup>, it has mentioned that, by applying the probability flow ODE, we can accurately estimate the likelihood, as well as acquiring the latent variables. These are all the advantages of a flow model. By harnessing the invertibility of the flow model, we can also perform image editing in the latent space.

On the other hand, when the transformation from  $\mathbf{x}_T$  to  $\mathbf{x}_0$  is governed by an ODE, we can apply different higher-order numerical methods for solving the ODE to accelerate the transformation from  $\mathbf{x}_T$  to  $\mathbf{x}_0$ . Although there are some methods to accelerate the solving of SDE, they are far less well investigated compared to ODE. Overall, compared to SDE, ODE is more straightforward for both theoretical analysis and practical solutions.

## 6.6. Revisiting DDIM

In the last part of Section 4, we demonstrated that the continuous version of the DDIM corresponds to the ODE:

$$\frac{d}{ds} \left( \frac{\mathbf{x}(s)}{\bar{\alpha}(s)} \right) = \varepsilon_\theta(\mathbf{x}(s), t(s)) \frac{d}{ds} \left( \frac{\bar{\beta}(s)}{\bar{\alpha}(s)} \right). \quad (6.15)$$



We will show below, that this result is a special case when we take the function  $f_t(\mathbf{x})$  in Eq. (6.14) as a linear function in  $\mathbf{x}$ . As the end of Section 5, we had the following relationships:

$$\begin{cases} f_t = \frac{1}{\bar{\alpha}_t} \frac{\partial \bar{\alpha}_t}{\partial t} \\ g^2(t) = 2\bar{\alpha}_t \bar{\beta}_t \frac{\partial}{\partial t} \left( \frac{\bar{\beta}_t}{\bar{\alpha}_t} \right) \\ s_{\theta}(\mathbf{x}, t) = -\frac{\varepsilon_{\theta}(\mathbf{x}, t)}{\bar{\beta}_t}. \end{cases} \quad (6.16)$$

Substituting these relationships into Eq. (6.14) and replacing  $\nabla_{\mathbf{x}} \log p_t(\mathbf{x})$  with  $s_{\theta}(\mathbf{x}, t)$ . With further simplification, we obtain:

$$\frac{1}{\bar{\alpha}_t} \frac{d\mathbf{x}}{dt} - \frac{\mathbf{x}}{\bar{\alpha}_t^2} \frac{d\bar{\alpha}_t}{dt} = \varepsilon_{\theta}(\mathbf{x}, t) \frac{d}{dt} \left( \frac{\bar{\beta}_t}{\bar{\alpha}_t} \right), \quad (6.17)$$

in which the L.H.S. can be further simplified into  $\frac{d}{dt} \left( \frac{\mathbf{x}}{\bar{\alpha}_t} \right)$ , such that Eq. (6.17) becomes fully equivalent to Eq. (6.15).

As such, in this section, we have shown how to obtain a more general form of the differential equation for the forward process, from which we derived the probability flow ODE and demonstrated DDIM as one of its special case.

## 7. Optimum Estimation of the Variance (Part I)

For the diffusion models, how to choose the variance as one of its parameters is a critical issue, as the quality of the generated samples strongly depend on the different choices of the variances that are used for the model.

Original blog post see <https://www.spaces.ac.cn/archives/9245>

*Recall in Section 1, it was given that, with the trained network  $\varepsilon_{\theta}(\mathbf{x}_t, t)$ , the greedy search in the backward direction is  $\mathbf{x}_{t-1} = \frac{1}{\bar{\alpha}_t}(\mathbf{x}_t - \beta_t \varepsilon_{\theta}(\mathbf{x}_t, t))$ . This can be turned into a random search if we add a noise term with variance  $\sigma_t$ , such that  $\mathbf{x}_{t-1} = \frac{1}{\bar{\alpha}_t}(\mathbf{x}_t - \beta_t \varepsilon_{\theta}(\mathbf{x}_t, t)) + \sigma_t \mathbf{z}$  with  $\mathbf{z} \sim \mathcal{N}(0, \mathbf{I})$ , and for practicality, we select  $\sigma_t = \beta_t$ .*

In Section 2, we assumed two different sample distributions from which we deduced two different choices of  $\sigma_t$ .

*Recall that when  $\tilde{p}(\mathbf{x}) = \delta(\mathbf{x}_0)$ , i.e. when there exists only a single sample, then the optimum choice of  $\sigma_t = \bar{\beta}_{t-1} \beta_t^2 / \bar{\beta}_t^2$ . These are determined from  $p(\mathbf{x}_{t-1} | \mathbf{x}_0)$  which can be computed from  $p(\mathbf{x}_{t-1} | \mathbf{x}_t, \mathbf{x}_0)$  when  $\tilde{p}(\mathbf{x}_0)$  is known.*

In Section 4, when we discussed DDIM, we make the variance as a hyperparameter, and we even had allowed the variance to be zero, although such choice would deteriorate the quality of the generated samples.

*Recall that in DDIM, the variance was first introduced in the generalised transition probability for the reversed process (conditioned on  $\mathbf{x}_0$ ):*

$$p(\mathbf{x}_{t-1} | \mathbf{x}_t, \mathbf{x}_0) = \mathcal{N}(\mathbf{x}_{t-1}; \kappa_t \mathbf{x}_t + \lambda_t \mathbf{x}_0, \sigma_t^2 \mathbf{I}),$$

*which we showed that both  $\kappa_t$  and  $\lambda_t$  can be written as a function of  $\sigma_t$ . This choice of the reverse transition probability has significantly expanded the solution space.*

In Section 5, when we discussed the general framework of SDE, we showed that the variances for the forward and reverse SDE should be the same, but in principle, this only holds when  $\Delta t \rightarrow 0$ .

*Recall that in Section 5, we derived  $p(\mathbf{x}_t | \mathbf{x}_{t+\Delta t})$  for the reverse process using Baye's rule, Taylor expanded  $\log p(\mathbf{x}_{t+\Delta t})$ , and combined the Gaussian functions. When  $\Delta t$  is non-zero,  $\sigma_t = g_{t+\Delta t}^2 \Delta t \mathbf{I}$ , and when  $\Delta t \rightarrow 0$ , the stochastic term falls back to  $g_t d\mathbf{w}$ , which is the same as the forward process.*

In the paper entitled “Improved Denoising Diffusion Probabilistic Models”<sup>12</sup>, the variance was set to be a **learnable parameter**, but this increased the model training difficulties.

Therefore, how should we choose the variance after all? The two papers entitled “*Analytic-DPM: An Analytic Estimate of the Optimal Reverse Variance in Diffusion Probabilistic Models*”<sup>13</sup> and “*Estimating the Optimal Covariance with Imperfect Mean in Diffusion Probabilistic Models*”<sup>14</sup> have provided a perfect solution to this problem, which will be discussed here.

### 7.1. Uncertainty in the reconstruction process

The two publications mentioned above came from the same team. The first article, which will be dubbed as **Analytic-DPM**, derived an analytical solution for the unconditioned variance based on DDIM. In the second paper, which will be dubbed as the **Extended-Analytic-DPM**, had weakened the hypothesis in the first paper, and proposed an optimisation procedure for the conditional variance.

In Section 4, when we presented the DDIM model, we showed that, for a given transition probability  $p(\mathbf{x}_t|\mathbf{x}_0) = \mathcal{N}(\mathbf{x}_t; \bar{\alpha}_t\mathbf{x}_0, \bar{\beta}_t^2\mathbf{I})$ , the general solution for the corresponding conditional transition probability for the reverse process  $p(\mathbf{x}_{t-1}|\mathbf{x}_t, \mathbf{x}_0)$  is

$$p(\mathbf{x}_{t-1}|\mathbf{x}_t, \mathbf{x}_0) = \mathcal{N}\left(\mathbf{x}_{t-1}; \frac{\sqrt{\bar{\beta}_{t-1}^2 - \sigma_t^2}}{\bar{\beta}_t}\mathbf{x}_t + \gamma_t\mathbf{x}_0, \sigma_t^2\mathbf{I}\right), \quad (7.1)$$

in which  $\gamma_t = \bar{\alpha}_{t-1} - \bar{\alpha}_t\sqrt{\bar{\beta}_{t-1}^2 - \sigma_t^2}/\bar{\beta}_t$ , and  $\sigma_t$  is an adjustable variance. In what follows, we applied  $\bar{\boldsymbol{\mu}}(\mathbf{x}_t)$  to estimate  $\mathbf{x}_0$ , and approximated

$$p(\mathbf{x}_{t-1}|\mathbf{x}_t) \approx p(\mathbf{x}_{t-1}|\mathbf{x}_t, \mathbf{x}_0 = \bar{\boldsymbol{\mu}}(\mathbf{x}_t)). \quad (7.2)$$

However, from the viewpoint of Bayesian statistics, such a treatment is not very appropriate, as the prediction of  $\mathbf{x}_0$  from  $\mathbf{x}_t$  cannot be very accurate but with some degrees of uncertainties. Strictly speaking, we should describe it with a probability distribution rather than a function. In fact, we have the following:

Marginalisation over all possible transitions from  $\mathbf{x}_t$  to  $\mathbf{x}_0$ .

$$p(\mathbf{x}_{t-1}|\mathbf{x}_t) = \int p(\mathbf{x}_{t-1}|\mathbf{x}_t, \mathbf{x}_0)p(\mathbf{x}_0|\mathbf{x}_t)d\mathbf{x}_0. \quad (7.3)$$

There is no way we can acquire an exact solution for  $p(\mathbf{x}_0|\mathbf{x}_t)$ , but we only need a rough estimation. In this case, we can use a normal distribution  $\mathcal{N}(\mathbf{x}_0; \bar{\boldsymbol{\mu}}(\mathbf{x}_t), \bar{\sigma}_t^2\mathbf{I})$  to regress  $p(\mathbf{x}_0|\mathbf{x}_t)$  (details later). With this approximated distribution, we have,

$$\begin{aligned} \mathbf{x}_{t-1} &= \frac{\sqrt{\bar{\beta}_{t-1}^2 - \sigma_t^2}}{\bar{\beta}_t}\mathbf{x}_t + \gamma_t\mathbf{x}_0 + \sigma_t\boldsymbol{\varepsilon}_1 \\ &= \frac{\sqrt{\bar{\beta}_{t-1}^2 - \sigma_t^2}}{\bar{\beta}_t}\mathbf{x}_t + \gamma_t(\bar{\boldsymbol{\mu}}(\mathbf{x}_t) + \bar{\sigma}_t\boldsymbol{\varepsilon}_2) + \sigma_t\boldsymbol{\varepsilon}_1 \\ &= \left(\frac{\sqrt{\bar{\beta}_{t-1}^2 - \sigma_t^2}}{\bar{\beta}_t}\mathbf{x}_t + \gamma_t\bar{\boldsymbol{\mu}}(\mathbf{x}_t)\right) + \underbrace{(\sigma_t\boldsymbol{\varepsilon}_1 + \gamma_t\bar{\sigma}_t\boldsymbol{\varepsilon}_2)}_{\sim \sqrt{\sigma_t^2 + \gamma_t^2\bar{\sigma}_t^2}\boldsymbol{\varepsilon}}, \end{aligned} \quad (7.4)$$

Directly drawing sample from Eq. (7.1)

Draw  $\mathbf{x}_0$  from the approximated normal distribution

in which  $\boldsymbol{\varepsilon}_1, \boldsymbol{\varepsilon}_2$  and  $\boldsymbol{\varepsilon} \sim \mathcal{N}(0, \mathbf{I})$ . By comparing Eq. (7.1), Eq. (7.2) and Eq. (7.4), it can be seen that  $p(\mathbf{x}_{t-1}|\mathbf{x}_t)$  is much closer to a normal distribution with the mean of  $\frac{\sqrt{\bar{\beta}_{t-1}^2 - \sigma_t^2}}{\bar{\beta}_t}\mathbf{x}_t + \gamma_t\bar{\boldsymbol{\mu}}(\mathbf{x}_t)$  and standard deviation of  $(\sigma_t^2 + \gamma_t^2\bar{\sigma}_t^2)\mathbf{I}$ . In particular, the mean is identical to that in Eq. (7.1), except now the standard deviation contains an extra term of  $\gamma_t^2\bar{\sigma}_t^2$ , which will not be zero even if  $\sigma_t = 0$ . This extra term is the correction term to the optimum variance in the **Analytic-DPM**.

### 7.2. Optimisation for the mean

Now we move on to discuss why we can use a normal distribution  $\mathcal{N}(\mathbf{x}_0; \bar{\boldsymbol{\mu}}(\mathbf{x}_t), \bar{\sigma}_t^2\mathbf{I})$  to regress  $p(\mathbf{x}_0|\mathbf{x}_t)$ . This problem is equivalent of determining the mean and the variance for the probability distribution  $p(\mathbf{x}_0|\mathbf{x}_t)$ . For the mean  $\bar{\boldsymbol{\mu}}(\mathbf{x}_t)$ , it depends on  $\mathbf{x}_t$ , so it must be regressed with a model. To do this, we will need a loss function, which we can start with

$$\mathbb{E}_{\mathbf{x}}[\mathbf{x}] = \underset{\boldsymbol{\mu}}{\operatorname{argmin}} \mathbb{E}_{\mathbf{x}}[\|\mathbf{x} - \boldsymbol{\mu}\|_2^2], \quad (7.5)$$

from which we have

$$\begin{aligned}
\bar{\mu}(\mathbf{x}_t) &= \mathbb{E}_{\mathbf{x}_0 \sim p(\mathbf{x}_0|\mathbf{x}_t)}[\mathbf{x}_0] \\
&= \underset{\mu}{\operatorname{argmin}} \mathbb{E}_{\mathbf{x}_0 \sim p(\mathbf{x}_0|\mathbf{x}_t)}[\|\mathbf{x}_0 - \mu\|_2^2] \\
&= \underset{\mu(t)}{\operatorname{argmin}} \mathbb{E}_{\mathbf{x}_0 \sim p(\mathbf{x}_0|\mathbf{x}_t), \mathbf{x}_t \sim p(\mathbf{x}_t)}[\|\mathbf{x}_0 - \mu(\mathbf{x}_t)\|_2^2] \\
&= \underset{\mu(t)}{\operatorname{argmin}} \mathbb{E}_{\mathbf{x}_0 \sim \tilde{p}(\mathbf{x}_0), \mathbf{x}_t \sim p(\mathbf{x}_t|\mathbf{x}_0)}[\|\mathbf{x}_0 - \mu(\mathbf{x}_t)\|_2^2].
\end{aligned} \tag{7.6}$$

This expression gives the formal definition of for the expectation value of  $\mathbf{x}_0$ , if we can draw sample from the distribution of  $p(\mathbf{x}_0|\mathbf{x}_t)$ .

Applying Eq. (7.5).

Now we also need to sample  $\mathbf{x}_t$  as  $\mu(\mathbf{x}_t)$  is a function of it

This is the target function for training  $\mu(\mathbf{x}_t)$ . If we introduce the parameterisation procedure as before, where we let

$$\mu(\mathbf{x}_t) = \frac{1}{\bar{\alpha}_t} (\mathbf{x}_t - \bar{\beta}_t \boldsymbol{\varepsilon}_\theta(\mathbf{x}_t, t)), \tag{7.7}$$

then, we recover the loss function for training the DDPM model, which is  $\|\boldsymbol{\varepsilon} - \boldsymbol{\varepsilon}_\theta(\bar{\alpha}_t \mathbf{x}_0 + \bar{\beta}_t \boldsymbol{\varepsilon}, t)\|_2^2$ . So the procedure from optimising the mean leads to the identical results as before.

### 7.3. Estimation of the variance: part I

Similar to the above discussions, the covariance matrix is given by

$$\begin{aligned}
\Sigma(\mathbf{x}_t) &= \mathbb{E}_{\mathbf{x}_0 \sim p(\mathbf{x}_0|\mathbf{x}_t)}[(\mathbf{x}_0 - \bar{\mu}(\mathbf{x}_t))(\mathbf{x}_0 - \bar{\mu}(\mathbf{x}_t))^\top] \\
&= \mathbb{E}_{\mathbf{x}_0 \sim p(\mathbf{x}_0|\mathbf{x}_t)}\left[\left((\mathbf{x}_0 - \mu) - (\bar{\mu}(\mathbf{x}_t) - \mu)\right)\left((\mathbf{x}_0 - \mu) - (\bar{\mu}(\mathbf{x}_t) - \mu)\right)^\top\right] \\
&= \mathbb{E}_{\mathbf{x}_0 \sim p(\mathbf{x}_0|\mathbf{x}_t)}[(\mathbf{x}_0 - \mu_0)(\mathbf{x}_0 - \mu_0)^\top] - (\bar{\mu}(\mathbf{x}_t) - \mu_0)(\bar{\mu}(\mathbf{x}_t) - \mu_0)^\top,
\end{aligned} \tag{7.8}$$

in which  $\mu_0$  is an arbitrary vector quantity. The last equation holds due to the translational invariance of the covariance matrix.

The above derivation gives us the full covariance matrix, but this is not what we want, since we would like to use  $\mathcal{N}(\mathbf{x}_0; \bar{\mu}(\mathbf{x}_t), \bar{\sigma}_t^2 \mathbf{I})$  to regress  $p(\mathbf{x}_{t-1}|\mathbf{x}_t)$ , in which, by design, the covariance matrix is  $\bar{\sigma}_t^2 \mathbf{I}$ . The latter has two characters:

1. It is independent from  $\mathbf{x}_t$ , as such, in order to eliminate its dependency on  $\mathbf{x}_t$ , we can take the average over all  $\mathbf{x}_t$ , such that  $\Sigma_t = \mathbb{E}_{\mathbf{x}_t \sim p(\mathbf{x}_t)}[\Sigma(\mathbf{x}_t)]$ ;
2. It is a multiple of the identity matrix, meaning we only need to consider the diagonal elements and take the mean of the trace, *i.e.*  $\bar{\sigma}_t^2 = \operatorname{Tr}(\Sigma_t)/d$ , with  $d = \dim(\mathbf{x})$ .

In this case, we have

$$\begin{aligned}
\bar{\sigma}_t^2 &= \mathbb{E}_{\mathbf{x}_t \sim p(\mathbf{x}_t)} \mathbb{E}_{\mathbf{x}_0 \sim p(\mathbf{x}_0|\mathbf{x}_t)} \left[ \frac{\|\mathbf{x}_0 - \mu_0\|_2^2}{d} \right] - \mathbb{E}_{\mathbf{x}_t \sim p(\mathbf{x}_t)} \left[ \frac{\|\bar{\mu}(\mathbf{x}_t) - \mu_0\|_2^2}{d} \right] \\
&= \frac{1}{d} \mathbb{E}_{\mathbf{x}_0 \sim \tilde{p}(\mathbf{x}_0)} \mathbb{E}_{\mathbf{x}_t \sim p(\mathbf{x}_t|\mathbf{x}_0)} [\|\mathbf{x}_0 - \mu_0\|_2^2] - \frac{1}{d} \mathbb{E}_{\mathbf{x}_t \sim p(\mathbf{x}_t)} [\|\bar{\mu}(\mathbf{x}_t) - \mu_0\|_2^2] \\
&= \frac{1}{d} \mathbb{E}_{\mathbf{x}_0 \sim \tilde{p}(\mathbf{x}_0)} [\|\mathbf{x}_0 - \mu_0\|_2^2] - \frac{1}{d} \mathbb{E}_{\mathbf{x}_t \sim p(\mathbf{x}_t)} [\|\bar{\mu}(\mathbf{x}_t) - \mu_0\|_2^2]
\end{aligned} \tag{7.9}$$

This is one possible analytical form of  $\bar{\sigma}_t^2$  that was derived by the author. When the model of  $\bar{\mu}(\mathbf{x}_t)$  has been trained, it can be approximately evaluated by first sampling a series of samples  $\mathbf{x}_0$  and  $\mathbf{x}_t$ . Most importantly, if we select  $\mu_0 = \mathbb{E}_{\mathbf{x}_0 \sim \tilde{p}(\mathbf{x}_0)}[\mathbf{x}_0]$ , then we have the following:

$$\bar{\sigma}_t^2 = \operatorname{Var}[\mathbf{x}_0] - \frac{1}{d} \mathbb{E}_{\mathbf{x}_t \sim p(\mathbf{x}_t)} [\|\bar{\mu}(\mathbf{x}_t) - \mu_0\|_2^2]. \tag{7.10}$$

Here,  $\operatorname{Var}[\mathbf{x}_0]$  is the variance of the training data at the pixel level. If the pixel level falls into the range of  $[a, b]$  for every pixel, then its variance will not be larger than  $\left(\frac{b-a}{2}\right)^2$ , such that

we have the following inequality:

$$\bar{\sigma}_t^2 \leq \text{Var}[\mathbf{x}_0] \leq \left( \frac{b-a}{2} \right). \quad (7.11)$$

#### 7.4. Estimation of the variance: part II

The above derivation is a relatively straightforward one that was came up by the author (J. Su). In the original Analytic-DPM paper, a slightly different, but less straightforward solution was given. Substituting Eq. (7.7) into the definition of the covariance matrix, we have

$$\begin{aligned} \Sigma(\mathbf{x}_t) &= \mathbb{E}_{\mathbf{x}_0 \sim p(\mathbf{x}_0|\mathbf{x}_t)} [(\mathbf{x}_0 - \bar{\boldsymbol{\mu}}(\mathbf{x}_t))(\mathbf{x}_0 - \bar{\boldsymbol{\mu}}(\mathbf{x}_t))^\top] \\ &= \mathbb{E}_{\mathbf{x}_0 \sim p(\mathbf{x}_0|\mathbf{x}_t)} \left[ \left( \left( \mathbf{x}_0 - \frac{\mathbf{x}_t}{\bar{\alpha}_t} \right) + \frac{\bar{\beta}_t}{\bar{\alpha}_t} \boldsymbol{\varepsilon}_\theta(\mathbf{x}_t, t) \right) \left( \left( \mathbf{x}_0 - \frac{\mathbf{x}_t}{\bar{\alpha}_t} \right) + \frac{\bar{\beta}_t}{\bar{\alpha}_t} \boldsymbol{\varepsilon}_\theta(\mathbf{x}_t, t) \right)^\top \right] \\ &= \mathbb{E}_{\mathbf{x}_0 \sim p(\mathbf{x}_0|\mathbf{x}_t)} \left[ \left( \mathbf{x}_0 - \frac{\mathbf{x}_t}{\bar{\alpha}_t} \right) \left( \mathbf{x}_0 - \frac{\mathbf{x}_t}{\bar{\alpha}_t} \right)^\top \right] - \frac{\bar{\beta}_t^2}{\bar{\alpha}_t^2} \boldsymbol{\varepsilon}_\theta(\mathbf{x}_t, t) \boldsymbol{\varepsilon}_\theta(\mathbf{x}_t, t)^\top \\ &= \frac{1}{\bar{\alpha}_t^2} \mathbb{E}_{\mathbf{x}_0 \sim p(\mathbf{x}_0|\mathbf{x}_t)} [(\mathbf{x}_t - \bar{\alpha}_t \mathbf{x}_0)(\mathbf{x}_t - \bar{\alpha}_t \mathbf{x}_0)^\top] - \frac{\bar{\beta}_t^2}{\bar{\alpha}_t^2} \boldsymbol{\varepsilon}_\theta(\mathbf{x}_t, t) \boldsymbol{\varepsilon}_\theta(\mathbf{x}_t, t)^\top. \end{aligned} \quad (7.12)$$

Now we take the average over  $\mathbf{x}_t$  on both sides, we get

$$\begin{aligned} &\mathbb{E}_{\mathbf{x}_t \sim p(\mathbf{x}_t)} \mathbb{E}_{\mathbf{x}_0 \sim p(\mathbf{x}_0|\mathbf{x}_t)} [(\mathbf{x}_t - \bar{\alpha}_t \mathbf{x}_0)(\mathbf{x}_t - \bar{\alpha}_t \mathbf{x}_0)^\top] \\ &= \mathbb{E}_{\mathbf{x}_0 \sim \tilde{p}(\mathbf{x}_0)} \mathbb{E}_{\mathbf{x}_t \sim p(\mathbf{x}_t|\mathbf{x}_0)} [(\mathbf{x}_t - \bar{\alpha}_t \mathbf{x}_0)(\mathbf{x}_t - \bar{\alpha}_t \mathbf{x}_0)^\top]. \end{aligned} \quad (7.13)$$

Since  $p(\mathbf{x}_t|\mathbf{x}_0) = \mathcal{N}(\mathbf{x}_t; \bar{\alpha}_t \mathbf{x}_0, \bar{\beta}_t^2 \mathbf{I})$ , so  $\bar{\alpha}_t \mathbf{x}_0$  is the mean for the distribution  $p(\mathbf{x}_t|\mathbf{x}_0)$ , therefore,  $\mathbb{E}_{\mathbf{x}_t \sim p(\mathbf{x}_t|\mathbf{x}_0)} [(\mathbf{x}_t - \bar{\alpha}_t \mathbf{x}_0)(\mathbf{x}_t - \bar{\alpha}_t \mathbf{x}_0)^\top]$  is the covariance matrix for  $p(\mathbf{x}_t|\mathbf{x}_0)$  which is obviously  $\bar{\beta}_t^2 \mathbf{I}$ . Therefore,

$$\mathbb{E}_{\mathbf{x}_0 \sim \tilde{p}(\mathbf{x}_0)} \mathbb{E}_{\mathbf{x}_t \sim p(\mathbf{x}_t|\mathbf{x}_0)} [(\mathbf{x}_t - \bar{\alpha}_t \mathbf{x}_0)(\mathbf{x}_t - \bar{\alpha}_t \mathbf{x}_0)^\top] = \mathbb{E}_{\mathbf{x}_0 \sim \tilde{p}(\mathbf{x}_0)} [\bar{\beta}_t^2 \mathbf{I}] = \bar{\beta}_t^2 \mathbf{I}. \quad (7.14)$$

Furthermore, we have

$$\Sigma_t = \mathbb{E}_{\mathbf{x} \sim p(\mathbf{x}_t)} [\Sigma(\mathbf{x}_t)] = \frac{\bar{\beta}_t^2}{\bar{\alpha}_t^2} (\mathbf{I} - \mathbb{E}_{\mathbf{x} \sim p(\mathbf{x}_t)} [\boldsymbol{\varepsilon}_\theta(\mathbf{x}_t, t) \boldsymbol{\varepsilon}_\theta(\mathbf{x}_t, t)^\top]). \quad (7.15)$$

Taking the trace and then dividing  $d$  on both sides, we obtain the following upper limit:

$$\bar{\sigma}_t^2 = \frac{\bar{\beta}_t^2}{\bar{\alpha}_t^2} \left( 1 - \frac{1}{d} \mathbb{E}_{\mathbf{x} \sim p(\mathbf{x}_t)} [\|\boldsymbol{\varepsilon}_\theta(\mathbf{x}_t, t)\|_2^2] \right) \leq \frac{\bar{\beta}_t^2}{\bar{\alpha}_t^2}, \quad (7.16)$$

which is the original result from Analytic-DPM.

#### 7.5. Further thoughts

Till this point, we have completely finished introducing Analytic-DPM. The entire derivation is a bit tricky but nonetheless, not too difficult and relatively clear in its logic. This is much friendly to be understood compared to the original paper, which completed the derivations using 7 pages involving 13 Lemmas in the Appendix.

Surely, we admire the observations made by the original authors, who first obtained analytical solutions for the variance. However, the entire original derivation for Analytic-DPM is rather cumbersome and lack of inspirations. For example, the notation  $\nabla_{\mathbf{x}_t} p(\mathbf{x}_t)$  had been used throughout the original paper, which causes three problems: Firstly, the whole derivation was not straightforward, making it difficult to understand how the authors came up with the ideas in the first place. Secondly, it makes the understanding of score-matching a prerequisite. Finally, one must change  $\nabla_{\mathbf{x}_t} p(\mathbf{x}_t)$  back to either  $\bar{\boldsymbol{\mu}}(\mathbf{x}_t)$  or  $\boldsymbol{\varepsilon}_\theta(\mathbf{x}_t, t)$  in practical applications, which is an unnecessary detour.

The starting point for our derivations here, is to estimate the parameters for the normal distributions, which its moment estimation is the same as estimating the maximum likelihood. As such, we only need to estimate the corresponding mean and covariance. There is no real need to match the result with those in the score-matching scheme, since the baseline model for Analytic-DPM is DDIM, which the latter was not based on the score-matching model anyway. Hence it is not beneficial to derive the Analytic-DPM results using the score-

matching formalism both theoretically and practically. Using  $\bar{\mu}(\mathbf{x}_t)$  or  $\varepsilon_{\theta}(\mathbf{x}_t, t)$  directly in the derivation, leads to more obvious mathematical form, and much straightforward for numerical implementation.

## 8. Optimum Estimation of the Variance (Part II)

In the previous section, we have introduced the framework of Analytic-DPM, which leads to an optimum estimation of the variance in the diffusion model that has already been trained. More importantly, the optimum variance was derived as an analytical solution. Numerical experimentations showed that this newly estimated optimum variance could indeed lead to improved qualities of the samples that are generated from the diffusion models.

In this section, we will continue to discuss the upgraded version of Analytic-DPM. This idea came from the paper entitled “*Estimating the Optimum Covariance with Imperfect Mean in Diffusion Probabilistic Models*”<sup>14</sup> published by the same team, and it was referred to as the **Extended-Analytic-DPM**, a term that will also be used here.

### 8.1. Revisiting the results

We had, in the previous section, started from DDIM, and derived the optimum variance for the generation process being

$$\sigma_t^2 + \gamma_t^2 \bar{\sigma}_t^2, \quad (8.1)$$

in which  $\bar{\sigma}_t$  is the variance for  $p(\mathbf{x}_0|\mathbf{x}_t)$ . It has the following estimate (from the second approach):

$$\bar{\sigma}_t^2 = \frac{\bar{\beta}_t^2}{\bar{\alpha}_t^2} \left( 1 - \frac{1}{d} \mathbb{E}_{\mathbf{x} \sim p(\mathbf{x}_t)} [\|\varepsilon_{\theta}(\mathbf{x}_t, t)\|_2^2] \right). \quad (8.2)$$

In retrospect, it was not difficult to come up with such a result. Let us assume that

$$\bar{\mu}(\mathbf{x}_t) = \frac{1}{\bar{\alpha}_t} (\mathbf{x}_t - \bar{\beta}_t \varepsilon_{\theta}(\mathbf{x}_t, t)) \quad (8.3)$$

has already provided us with an accurate estimate of the mean vector for the distribution  $p(\mathbf{x}_0|\mathbf{x}_t)$ , then according to the definition of the covariance matrix:

$$\begin{aligned} \Sigma(\mathbf{x}_t) &= \mathbb{E}_{\mathbf{x}_0 \sim p(\mathbf{x}_0|\mathbf{x}_t)} [(\mathbf{x}_0 - \bar{\mu}(\mathbf{x}_t))(\mathbf{x}_0 - \bar{\mu}(\mathbf{x}_t))^{\top}] \\ &= \frac{1}{\bar{\alpha}_t^2} \mathbb{E}_{\mathbf{x}_0 \sim p(\mathbf{x}_0|\mathbf{x}_t)} [(\mathbf{x}_t - \bar{\alpha}_t \mathbf{x}_0)(\mathbf{x}_t - \bar{\alpha}_t \mathbf{x}_0)^{\top}] - \frac{\bar{\beta}_t^2}{\bar{\alpha}_t^2} \varepsilon_{\theta}(\mathbf{x}_t, t) \varepsilon_{\theta}(\mathbf{x}_t, t)^{\top}. \end{aligned} \quad (8.4)$$

To remove the explicit dependency on  $\mathbf{x}_t$ , we then take the average over  $\mathbf{x}_t \sim p(\mathbf{x}_t)$ , which gives

$$\Sigma_t = \mathbb{E}_{\mathbf{x} \sim p(\mathbf{x}_t)} [\Sigma(\mathbf{x}_t)] = \frac{\bar{\beta}_t^2}{\bar{\alpha}_t^2} (\mathbf{I} - \mathbb{E}_{\mathbf{x} \sim p(\mathbf{x}_t)} [\varepsilon_{\theta}(\mathbf{x}_t, t) \varepsilon_{\theta}(\mathbf{x}_t, t)^{\top}]). \quad (8.5)$$

If we then tool the trace of the diagonal element and average it over the matrix dimension, i.e.  $\bar{\sigma}_t^2 = \text{Tr}(\Sigma_t)/d$ , then we recover the scalar variance, as given in Eq. (8.2).

### 8.2. Different approaches to improve the results

Before we formally introduce the Extended-Analytic-DPM, we can briefly reflect, are there any other rooms to improve Analytic-DPM? Actually there are quite a few possible improvements.

For example, in Analytic-DPM, we assumed that the normal distribution that we used to approximate  $p(\mathbf{x}_0|\mathbf{x}_t)$  has a covariance matrix of  $\sigma_t^2 \mathbf{I}$ , in which all the diagonal matrix elements are identical. The most trivial improvement would be to make the diagonal elements different from each other, i.e.  $\text{diag}(\bar{\sigma}_t^2)$ . Here, we restrict the vector multiplications are all being carried out with the Hamamard product, for example,  $\mathbf{x}^2 = \mathbf{x} \otimes \mathbf{x}$ . In this case, we only need to consider the diagonal part of  $\Sigma_t$ , and from Eq. (8.5), the corresponding estimate for the variance is

$$\bar{\sigma}_t^2 = \frac{\bar{\beta}_t^2}{\bar{\alpha}_t^2} (\mathbf{1}_d - \mathbb{E}_{\mathbf{x} \sim p(\mathbf{x}_t)} [\varepsilon_{\theta}^2(\mathbf{x}_t, t)]), \quad (8.6)$$

in which  $\mathbf{1}_d$  is a  $d$ -dimensional unit vector. A further improvement is to retain the dependency

Original blog post see <https://www.spaces.ac.cn/archives/9246>

of  $\bar{\sigma}_t^2$  on  $\mathbf{x}_t$ , *i.e.* we consider  $\bar{\sigma}_t^2(\mathbf{x}_t)$ , making it similar to  $\bar{\mu}(\mathbf{x}_t)$ , which now  $\bar{\sigma}_t^2(\mathbf{x}_t)$  also becomes a learnable model on the input  $\mathbf{x}_t$ .

On the other hand, can we consider using the full matrix of  $\Sigma_t$ ? This is workable only in principle but not very practical. This is because  $\Sigma_t$  is a  $(d \times d)$  matrix. For high resolution images, as  $d$  is the total number of pixels, it makes the storage and computation of such a large matrix very costly.

However, as  $\bar{\mu}(\mathbf{x}_t)$  is learnt from the model, it may not be able to accurately represent  $\mathbb{E}_{\mathbf{x} \sim p(\mathbf{x}_0|\mathbf{x}_t)}[\mathbf{x}_0]$ . This is the meaning of the **imperfect mean** in the title for the Extended-Analytic-DPM paper. Therefore, it would be more meaningful and practical to further improve the model under the ideal of imperfect mean.

### 8.3. Maximum likelihood

Suppose the model for the mean  $\bar{\mu}(\mathbf{x}_t)$  has been trained, then, in the distribution  $p(\mathbf{x}_0|\mathbf{x}_t) \sim \mathcal{N}(\mathbf{x}_0; \bar{\mu}(\mathbf{x}_t), \bar{\sigma}_t^2 \mathbf{I})$ , the only remaining parameter will be  $\bar{\sigma}_t^2$ . The corresponding negative log-likelihood is

$$\begin{aligned} & \mathbb{E}_{\mathbf{x}_t \sim p(\mathbf{x}_t)} \mathbb{E}_{\mathbf{x}_0 \sim p(\mathbf{x}_0|\mathbf{x}_t)} [-\log \mathcal{N}(\mathbf{x}_0; \bar{\mu}(\mathbf{x}_t), \bar{\sigma}_t^2 \mathbf{I})] \\ &= \frac{1}{2\bar{\sigma}_t^2} \mathbb{E}_{\mathbf{x}_t, \mathbf{x}_0 \sim p(\mathbf{x}_t|\mathbf{x}_0)\bar{p}(\mathbf{x}_0)} [\|\mathbf{x}_0 - \bar{\mu}(\mathbf{x}_t)\|_2^2] + \frac{d}{2} \log \bar{\sigma}_t^2 + \frac{d}{2} \log 2\pi, \end{aligned} \quad (8.7)$$

for which the minimum is reached when

$$\bar{\sigma}_t^2 = \frac{1}{d} \mathbb{E}_{\mathbf{x}_t, \mathbf{x}_0 \sim p(\mathbf{x}_t|\mathbf{x}_0)\bar{p}(\mathbf{x}_0)} [\|\mathbf{x}_0 - \bar{\mu}(\mathbf{x}_t)\|_2^2]. \quad (8.8)$$

Here,  $\bar{\mu}(\mathbf{x}_t)$  may not be sufficiently accurate, so the second equality in Eq. (8.4) does not really hold, we can only work from the first equality in Eq. (8.4), so we substitute Eq. (8.3) into Eq. (8.8) and we reached:

$$\bar{\sigma}_t^2 = \frac{\bar{\beta}_t^2}{\bar{\alpha}_t^2 d} \mathbb{E}_{\mathbf{x}_0 \sim \bar{p}(\mathbf{x}_0), \boldsymbol{\varepsilon} \sim \mathcal{N}(0, \mathbf{I})} [\|\boldsymbol{\varepsilon} - \boldsymbol{\varepsilon}_\theta(\bar{\alpha}_t \mathbf{x}_0 + \bar{\beta}_t \boldsymbol{\varepsilon}, t)\|_2^2]. \quad (8.9)$$

This is the result when the covariance matrix is a scaled identity matrix, *i.e.*,  $\bar{\sigma}_t^2 \mathbf{I}$ . For a more general case where the covariance matrix is a diagonal one, *i.e.*,  $\mathcal{N}(\mathbf{x}_0; \bar{\mu}(\mathbf{x}_t), \text{diag}(\bar{\sigma}_t^2))$ , the corresponding result is:

$$\bar{\sigma}_t^2 = \frac{\bar{\beta}_t^2}{\bar{\alpha}_t^2} \mathbb{E}_{\mathbf{x}_0 \sim \bar{p}(\mathbf{x}_0), \boldsymbol{\varepsilon} \sim \mathcal{N}(0, \mathbf{I})} [\|\boldsymbol{\varepsilon} - \boldsymbol{\varepsilon}_\theta(\bar{\alpha}_t \mathbf{x}_0 + \bar{\beta}_t \boldsymbol{\varepsilon}, t)\|_2^2]. \quad (8.10)$$

### 8.4. Conditional variance

If we want to obtain the covariance  $\text{diag}(\bar{\sigma}_t^2(\mathbf{x}_t))$  that is conditioned on  $\mathbf{x}_t$ , then we need to eliminate the step that takes the average over  $\mathbf{x}_t$ , from which we get

$$\bar{\sigma}_t^2(\mathbf{x}_t) = \mathbb{E}_{\mathbf{x}_0 \sim p(\mathbf{x}_0|\mathbf{x}_t)} [(\mathbf{x}_0 - \bar{\mu}(\mathbf{x}_t))^2] = \frac{\bar{\beta}_t^2}{\bar{\alpha}_t^2} \mathbb{E}_{\mathbf{x}_0 \sim p(\mathbf{x}_0|\mathbf{x}_t)} [(\boldsymbol{\varepsilon}_t - \boldsymbol{\varepsilon}_\theta(\mathbf{x}_t, t))^2], \quad (8.11)$$

in which  $\boldsymbol{\varepsilon}_t = (\mathbf{x}_t - \bar{\alpha}_t \mathbf{x}_0)/\bar{\beta}_t$ . Same as in the previous section, taking the use of

$$\mathbb{E}_{\mathbf{x}}[\mathbf{x}] = \underset{\boldsymbol{\mu}}{\text{argmin}} \mathbb{E}_{\mathbf{x}} [\|\mathbf{x} - \boldsymbol{\mu}\|_2^2], \quad (8.12)$$

we get the following

$$\begin{aligned} \bar{\sigma}_t^2(\mathbf{x}_t) &= \frac{\bar{\beta}_t^2}{\bar{\alpha}_t^2} \mathbb{E}_{\mathbf{x}_0 \sim p(\mathbf{x}_0|\mathbf{x}_t)} [(\boldsymbol{\varepsilon}_t - \boldsymbol{\varepsilon}_\theta(\mathbf{x}_t, t))^2] \\ &= \frac{\bar{\beta}_t^2}{\bar{\alpha}_t^2} \underset{\mathbf{g}}{\text{argmin}} \mathbb{E}_{\mathbf{x}_0 \sim p(\mathbf{x}_0|\mathbf{x}_t)} [\|(\boldsymbol{\varepsilon}_t - \boldsymbol{\varepsilon}_\theta(\mathbf{x}_t, t))^2 - \mathbf{g}\|_2^2] \\ &= \frac{\bar{\beta}_t^2}{\bar{\alpha}_t^2} \underset{\mathbf{g}(\mathbf{x}_t)}{\text{argmin}} \mathbb{E}_{\mathbf{x}_t \sim p(\mathbf{x}_t)} \mathbb{E}_{\mathbf{x}_0 \sim p(\mathbf{x}_0|\mathbf{x}_t)} [\|(\boldsymbol{\varepsilon}_t - \boldsymbol{\varepsilon}_\theta(\mathbf{x}_t, t))^2 - \mathbf{g}(\mathbf{x}_t)\|_2^2] \\ &= \frac{\bar{\beta}_t^2}{\bar{\alpha}_t^2} \underset{\mathbf{g}(\mathbf{x}_t)}{\text{argmin}} \mathbb{E}_{\mathbf{x}_t, \mathbf{x}_0 \sim p(\mathbf{x}_t|\mathbf{x}_0)\bar{p}(\mathbf{x}_0)} [\|(\boldsymbol{\varepsilon}_t - \boldsymbol{\varepsilon}_\theta(\mathbf{x}_t, t))^2 - \mathbf{g}(\mathbf{x}_t)\|_2^2]. \end{aligned} \quad (8.13)$$

The key idea here is, since we know  $\mathbf{x}_0$ , and thus  $\bar{p}(\mathbf{x}_0)$  very well, we should seek an estimate of  $\bar{\sigma}_t$  directly from sampling  $\mathbf{x}_0$  rather than sample from  $\mathbf{x}_t$  and map it back to  $\mathbf{x}_0$ .

This also seems to be the formal definition for the variance.

This is the “NPR-DPM” scheme for learning the conditional variance in the Extended-Analytic-DPM. In addition, the paper also introduced another variance estimation scheme, dubbed “SN-DPM”, which was based on the assumption of a “perfect mean” rather than an imperfect one. However, the result shows that SN-DPM gives better results than NPR-DPM. What this means is that even though the paper was originally designed to solve the imperfect mean problem, the better results of SN-DPM suggests that the perfect mean is more close to practical realisms, such that the problem of imperfect mean is somehow meaningless.

### 8.5. Two stages

It was mentioned in the paper “*Improved Denoising Diffusion Probabilistic Models*”<sup>12</sup> that, by treating the covariance as a learnable will increase the difficulty in model training. In this case, why we need to train another model for the variance in the Extended-Analytic-DPM?

In the DDPM framework, there exist two possible choices for the variance, namely  $\sigma_t = (\bar{\beta}_{t-1}/\bar{\beta}_t)\beta_t$  and  $\sigma_t = \beta_t$ , which can already provide reasonably good generative outcomes. In such case, further fine tuning the variance will not affect the generative outcomes significantly, at least for a generative process with a large number of steps. The model training, in this case, should still be focusing on learning the mean  $\bar{\mu}(\mathbf{x}_t)$ . On the other hand, if both the mean and variance are learnt simultaneously, then, as the training progresses, the constantly changing variance can interfere with the learning of the mean, which violates the goal to focus on training the mean  $\bar{\mu}(\mathbf{x}_t)$ .

The Extended-Analytic-DPM has devised a genius solution to overcome the problem through a two-staged approach, in which the original variance was used to train the model for the mean  $\bar{\mu}(\mathbf{x}_t)$ , then the trained model was fixed, and some of its parameters were used to train another model for the variance. The main advantages of this approach are:

1. Reducing the number of parameters and the corresponding training costs.
2. Enabling the re-usage of the pre-trained model for the mean, and
3. Achieving a more stable training process.

## 9. Conditional Generation

In the previous few sections, we have mainly been focusing on discussing the theoretical frameworks for the diffusion models. In this section, we shall focus on a more practical topic, that is the conditional generation of new samples with the diffusion models.

As a generative model, diffusion model shares a similar developmental track as VAE, GAN and the flow model, for which the conditional generative model was soon proposed after the birth of the unconditional generative model. Generally speaking, the unconditional generation can facilitate the exploration of the upper limit in the qualities of the samples that are generated by the model, whereas the conditional generation is more for practical applications, as now we can control the samples that are being generated based on our specific needs. Since the birth of DDPM, many different conditional diffusion generative models have been developed. The capability to conditionally control the diffusion generative process is the driving force that makes the diffusion models become so popular these days, such as the text-to-image models DALL.E2 and Imagen.

In this section, we will briefly study and summarise the theoretical foundations for the conditional diffusion models.

### 9.1. Theoretical analysis

Practically, there are two pathways to realise conditional generation, which are usually dubbed as either **Classifier-Guidance** or **Classifier-Free**.

For most people, the cost for training a diffusion model at the SOTA-level is too large. On the other hand, the cost associated with training a classifier is more or less acceptable. As such, most people wants to use a pre-trained diffusion model and add in an additional classifier to realise the conditional sample generation, This is the *ad-hoc* **Classifier-Guidance** approach. Nevertheless, for Tech Giants such as Google and OpenAI, that are not lack of both data and computational power, they are more keen to include the conditional signals during the training of the diffusion model, in order to achieve better sample qualities, this is the so-called **Classifier-Free** approach.

The **Classifier-Guidance** approach was first proposed in the article “*Diffusion Models Beats GANs on Image Synthesis*”<sup>15</sup>, which was aimed to achieve sample generations according to different classifications. Subsequently, in the paper “*More Control for Free! Image Synthesis with Semantic Diffusion Guidance*”<sup>16</sup>, the concept of the classifier has been further generalised,

Original blog post see <https://www.spaces.ac.cn/archives/9257>



making it possible to generate samples that are guided by either images or texts. Generally speaking, the training costs for the Classifier-Guidance models are lower. However, the cost for inferencing is higher and the sample quality is lower compared the Classifier-Free approach.

As for the **Classifier-Free** approach, it was first proposed in the article “*Classifier-Free Diffusion Guidance*”<sup>17</sup>. Models such as DALL.E2 and Imagen are all based on this approach. Strictly speaking, there is not much technical tricks involved in the Classifier-Free approach. It is the most straightforward generalisation of diffusion model to its conditional variant. It appears later than the Classifier-Guidance approach most likely because of its high training cost. Nevertheless, when there exists sufficient data and computational power for training, the samples generated from the Classifier-Free approach have demonstrated amazing capability in controlling the details in the process of sample generation.

## 9.2. Conditional input

Simply put, there is not much technicalities in the Classifier-Free approach, except for its high training costs. As such, we will focus the subsequent discussions on the Classifier-Guidance approach, and briefly outline the Classifier-Free approach at the end.

Based on the analysis from the previous sections, it should be clear to the readers that the key step in the generation process is to build the transition probability  $p(\mathbf{x}_{t-1}|\mathbf{x}_t)$ . As such, for sample generation based on some condition  $\mathbf{y}$ , all we need to do is introduce  $\mathbf{y}$  into the generation process via the conditional probability  $p(\mathbf{x}_{t-1}|\mathbf{x}_t, \mathbf{y})$  which replaces  $p(\mathbf{x}_{t-1}|\mathbf{x}_t)$ . In order to reuse the  $p(\mathbf{x}_{t-1}|\mathbf{x}_t)$ . from the trained unconditional generative model, we apply the Baye’s rule

$$p(\mathbf{x}_{t-1}|\mathbf{y}) = \frac{p(\mathbf{x}_{t-1})p(\mathbf{y}|\mathbf{x}_{t-1})}{p(\mathbf{y})}, \quad (9.1)$$

and putting  $\mathbf{x}_t$  back into every term, we get:

$$p(\mathbf{x}_{t-1}|\mathbf{x}_t, \mathbf{y}) = \frac{p(\mathbf{x}_{t-1}|\mathbf{x}_t)p(\mathbf{y}|\mathbf{x}_{t-1}, \mathbf{x}_t)}{p(\mathbf{y}|\mathbf{x}_t)}. \quad (9.2)$$

Note that  $\mathbf{x}_t$  is generated from  $\mathbf{x}_{t-1}$  by adding the noise in the forward process. In this case, adding  $\mathbf{x}_t$  will not bring any benefit to the classifier, as such,  $p(\mathbf{y}|\mathbf{x}_{t-1}, \mathbf{x}_t) = p(\mathbf{y}|\mathbf{x}_{t-1})$ , this leads to the following:

$$p(\mathbf{x}_{t-1}|\mathbf{x}_t, \mathbf{y}) = \frac{p(\mathbf{x}_{t-1}|\mathbf{x}_t)p(\mathbf{y}|\mathbf{x}_{t-1})}{p(\mathbf{y}|\mathbf{x}_t)} = p(\mathbf{x}_{t-1}|\mathbf{x}_t) \exp [\log p(\mathbf{y}|\mathbf{x}_{t-1}) - \log p(\mathbf{y}|\mathbf{x}_t)]. \quad (9.3)$$

## 9.3. Approximated distribution

Readers who are familiar with the derivations in Section 5 on SDE will not be surprised by the following derivations. When  $T$  is sufficiently large, the standard deviations for the distribution  $p(\mathbf{x}_t|\mathbf{x}_{t-1})$  will be sufficiently small. This is saying that the transition probability  $p(\mathbf{x}_t|\mathbf{x}_{t-1})$  will only be significantly larger than zero when  $\mathbf{x}_t$  is very close to  $\mathbf{x}_{t-1}$ . This is also true for the conditional probabilities, *i.e.* either  $p(\mathbf{x}_{t-1}|\mathbf{x}_t, \mathbf{y})$  or  $p(\mathbf{x}_t|\mathbf{x}_{t-1}, \mathbf{y})$  will only be significantly larger than zero when  $\mathbf{x}_t$  is very close to  $\mathbf{x}_{t-1}$ . As such, we only need to consider the change in the probability distributions within this small range of  $\mathbf{x}$ . With this, we can apply the Taylor expansion as

$$\log p(\mathbf{y}|\mathbf{x}_{t-1}) - \log p(\mathbf{y}|\mathbf{x}_t) \approx (\mathbf{x}_{t-1} - \mathbf{x}_t) \nabla_{\mathbf{x}_t} \log p(\mathbf{y}|\mathbf{x}_t). \quad (9.4)$$

Strictly speaking, there should also exist an extra  $t$ -dependent term, but it is irrelevant to  $\mathbf{x}_{t-1}$ , as it is a constant that will not change the probability of  $\mathbf{x}_{t-1}$ . Therefore, this term is ignored in Eq. (9.4). Assuming that we already have  $p(\mathbf{x}_{t-1}|\mathbf{x}_t) = \mathcal{N}(\mathbf{x}_{t-1}; \boldsymbol{\mu}(\mathbf{x}_t), \sigma_t^2 \mathbf{I}) \propto \exp [-\|\mathbf{x}_{t-1} - \boldsymbol{\mu}(\mathbf{x}_t)\|^2 / 2\sigma_t^2]$ , then we have the following approximation:

$$\begin{aligned} p(\mathbf{x}_{t-1}|\mathbf{x}_t, \mathbf{y}) &\propto \exp [-\|\mathbf{x}_{t-1} - \boldsymbol{\mu}(\mathbf{x}_t)\|^2 / 2\sigma_t^2 + (\mathbf{x}_{t-1} - \mathbf{x}_t) \nabla_{\mathbf{x}_t} \log p(\mathbf{y}|\mathbf{x}_t)] \\ &\propto \exp [-\|\mathbf{x}_{t-1} - \boldsymbol{\mu}(\mathbf{x}_t) - \sigma_t^2 \nabla_{\mathbf{x}_t} \log p(\mathbf{y}|\mathbf{x}_t)\|^2 / 2\sigma_t^2 \log p(\mathbf{y}|\mathbf{x}_t)]. \end{aligned} \quad (9.5)$$

This result shows that the distribution  $p(\mathbf{x}_{t-1}|\mathbf{x}_t, \mathbf{y})$  is approximately a normal distribution of  $\mathcal{N}(\mathbf{x}_{t-1}; \boldsymbol{\mu}(\mathbf{x}_t) + \sigma_t^2 \nabla_{\mathbf{x}_t} \log p(\mathbf{y}|\mathbf{x}_t), \sigma_t^2 \mathbf{I})$ . In this case, we only need to modify the sampling



process to:

$$\mathbf{x}_{t-1} = \boldsymbol{\mu}(\mathbf{x}_t) + \sigma_t^2 \nabla_{\mathbf{x}_t} \log p(\mathbf{y}|\mathbf{x}_t) + \sigma_t \boldsymbol{\varepsilon}, \quad \boldsymbol{\varepsilon} \sim \mathcal{N}(0, \mathbf{I}). \quad (9.6)$$

This is the core result in the Classifier-Guidance Approach. Note that  $\mathbf{x}_t$  is the sample after noises are being added, which is the input for  $p(\mathbf{y}|\mathbf{x}_t)$ . It means that we need a model that can predict the noised sample. If we only have a model to predict the noise-free sample  $p_0(\mathbf{y}|\mathbf{x}_0)$ , then what we may consider is

$$p(\mathbf{y}|\mathbf{x}_0) = p_0(\mathbf{y}|\boldsymbol{\mu}(\mathbf{x}_t)), \quad (9.7)$$

i.e., we use  $\boldsymbol{\mu}(\cdot)$  to denoise  $\mathbf{x}_t$  before plugging it into  $p(\mathbf{y}|\mathbf{x}_t)$ . This avoids the cost of training a classifier with noisy samples.

#### 9.4. Scaling the classifier gradient

In the original paper entitled “*Diffusion Model Beats GAN on Image Synthesis*”, it was found that if a scaling parameter  $\gamma$  was introduced into the gradient of the classifier, one can better attenuate the generated samples. In this case, we have:

$$\mathbf{x}_{t-1} = \boldsymbol{\mu}(\mathbf{x}_t) + \sigma_t^2 \gamma \nabla_{\mathbf{x}_t} \log p(\mathbf{y}|\mathbf{x}_t) + \sigma_t \boldsymbol{\varepsilon}, \quad \boldsymbol{\varepsilon} \sim \mathcal{N}(0, \mathbf{I}). \quad (9.8)$$

When  $\gamma > 1$ , the generation process will utilise more information from the classifier. This will increase the correlation between the output and the input condition  $\mathbf{y}$ , at the price of reducing the diversity of the output samples. On the other hand, when  $\gamma < 1$ , the generative process will reduce the correlation between the generated output and the conditional input, thus leading to more diversified outcomes.

How should we understand this parameter? In the original paper, it was proposed that  $\gamma$  could be understood by adjusting the form of the probability distribution through the use of a power operation:

$$\tilde{p}(\mathbf{y}|\mathbf{x}_t) = \frac{p^\gamma(\mathbf{y}|\mathbf{x}_t)}{\mathcal{Z}(\mathbf{x}_t)}, \quad \mathcal{Z}(\mathbf{x}_t) = \sum_{\mathbf{y}} p^\gamma(\mathbf{y}|\mathbf{x}_t). \quad (9.9)$$

As  $\gamma$  increases,  $\tilde{p}(\mathbf{y}|\mathbf{x}_t)$  will slowly merge into an one-hot distribution. If this distribution was used to substitute  $p(\mathbf{y}|\mathbf{x}_0)$  in the classifier as the guidance, then the generative process will tend to select samples that have higher confidence level in being correctly classified. Although this explanation can facilitate our understanding to some extent, it was not completely right. This is because

$$\nabla_{\mathbf{x}_t} \log \tilde{p}(\mathbf{y}|\mathbf{x}_t) = \gamma \nabla_{\mathbf{x}_t} \log p(\mathbf{y}|\mathbf{x}_t) - \nabla_{\mathbf{x}_t} \mathcal{Z}(\mathbf{x}_t) \neq \gamma \nabla_{\mathbf{x}_t} \log p(\mathbf{y}|\mathbf{x}_t). \quad (9.10)$$

The authors of the original paper had wrongly took  $\mathcal{Z}(\mathbf{x}_t)$  as a constant, which gave  $\nabla_{\mathbf{x}_t} \mathcal{Z}(\mathbf{x}_t) = 0$ . However, when  $\gamma \neq 1$ , it is obvious that  $\mathcal{Z}(\mathbf{x}_t)$  will also be dependent on  $\mathbf{x}_t$ . Unfortunately, there does not seem to be a suitable solution to overcome this problem, we can only roughly assume that the property of the gradient function at  $\gamma = 1$  can be generalised to the case when  $\gamma \neq 1$ .

#### 9.5. Similarity control

As a matter of fact, the best way to understand the case when  $\gamma \neq 1$  is to abandon the Bayes’ rule [Eq. (9.2) and Eq. (9.3)] when trying to understand the term  $p(\mathbf{x}_{t-1}|\mathbf{x}_t, \mathbf{y})$ . Instead, we could straightforwardly define

$$p(\mathbf{x}_{t-1}|\mathbf{x}_t, \mathbf{y}) = \frac{p(\mathbf{x}_{t-1}|\mathbf{x}_t) e^{\gamma \cdot \text{sim}(\mathbf{x}_{t-1}, \mathbf{y})}}{\mathcal{Z}(\mathbf{x}_t, \mathbf{y})}, \quad \mathcal{Z}(\mathbf{x}_t|\mathbf{y}) = \sum_{\mathbf{x}_{t-1}} p(\mathbf{x}_{t-1}|\mathbf{x}_t) e^{\gamma \cdot \text{sim}(\mathbf{x}_{t-1}, \mathbf{y})}, \quad (9.11)$$

in which  $\text{sim}(\mathbf{x}_{t-1}, \mathbf{y})$  is a similarity measure between the generated sample  $\mathbf{x}_{t-1}$  and the conditional input  $\mathbf{y}$ . From this perspective,  $\gamma$  comes naturally from the definition of  $p(\mathbf{x}_{t-1}|\mathbf{x}_t, \mathbf{y})$ , which directly controls the correlation between the output and the condition. As  $\gamma$  increases, the model will generate  $\mathbf{x}_{t-1}$  that has stronger correlation with  $\mathbf{y}$ .

In order to arrive at an approximation that will further facilitate samplings, we could expand the similarity term approximately the point  $\mathbf{x}_{t-1}$  that is close to  $\mathbf{x}_t$  as:

$$e^{\gamma \cdot \text{sim}(\mathbf{x}_{t-1}, \mathbf{y})} \approx e^{\gamma \cdot \text{sim}(\mathbf{x}_t, \mathbf{y}) + \gamma(\mathbf{x}_{t-1} - \mathbf{x}_t) \nabla_{\mathbf{x}_t} \text{sim}(\mathbf{x}_t, \mathbf{y})}. \quad (9.12)$$

Assuming that this approximation is sufficiently accurate, we can then eliminate the term that are irrelevant to  $\mathbf{x}_{t-1}$ , from which we get

$$p(\mathbf{x}_{t-1}|\mathbf{x}_t, \mathbf{y}) \propto p(\mathbf{x}_{t-1}|\mathbf{x}_t) e^{\gamma(\mathbf{x}_{t-1}-\mathbf{x}_t) \nabla_{\mathbf{x}_t} \text{sim}(\mathbf{x}_t, \mathbf{y})}. \quad (9.13)$$

Similarly, substituting in  $p(\mathbf{x}_{t-1}|\mathbf{x}_t) = \mathcal{N}(\mathbf{x}_{t-1}; \boldsymbol{\mu}(\mathbf{x}_t), \sigma_t^2 \mathbf{I})$  and completing the square, we get

$$p(\mathbf{x}_{t-1}|\mathbf{x}_t) \propto \mathcal{N}(\mathbf{x}_{t-1}; \boldsymbol{\mu}(\mathbf{x}_t) + \gamma \sigma_t^2 \nabla_{\mathbf{x}_t} \text{sim}(\mathbf{x}_t, \mathbf{y}), \sigma_t^2 \mathbf{I}), \quad (9.14)$$

which now we no longer need to worry about the probabilistic interpretation of  $p(\mathbf{y}|\mathbf{x}_t)$ , but only a suitable definition of the similarity measure  $\text{sim}(\mathbf{x}_t, \mathbf{y})$ . Here,  $\mathbf{y}$  becomes no longer restricted to a class, it could be any type of inputs (image or text). The general treatment is to encode them into a vector representation using their own encoder and then apply the cosine similarity:

$$\text{sim}(\mathbf{x}_t, \mathbf{y}) = \frac{E_1(\mathbf{x}_t) \cdot E_2(\mathbf{y})}{\|E_1(\mathbf{x}_t)\| \cdot \|E_2(\mathbf{y})\|}. \quad (9.15)$$

It should be noted that, since the intermediate state  $\mathbf{x}_t$  contains Gaussian noise, it is generally better not to use the encoder for the noise-free sample but fine-tune it with noised samples. Above is the core idea in the article “*More Control for Free! Image Synthesis with Semantic Diffusion Guidance*”.

## 9.6. Continuous scenarios

We saw that in all the previous derivations, that the correction to the mean is either  $\sigma_t^2 \gamma \nabla_{\mathbf{x}_t} \log p(\mathbf{y}|\mathbf{x}_t)$  or  $\gamma \sigma_t^2 \nabla_{\mathbf{x}_t} \text{sim}(\mathbf{x}_t, \mathbf{y})$ , which both contain the term  $\sigma_t$ , meaning that when  $\sigma_t = 0$ , the correcting term will also be zero, making the conditional input playing no effect on sample generation.

So in this case, can we still set  $\sigma_t$  to be zero? The answer is yes. For example, in DDIM introduced in Section 4, it has its corresponding generation process with  $\sigma_t = 0$ . How should we realise conditional generation? In this case, we need to apply the SDE formalism introduced in Section 6, in which we showed that for a forward SDE:

$$d\mathbf{x} = f_t(\mathbf{x})dt + g_t d\mathbf{w}, \quad (9.16)$$

with the corresponding reverse SDE

$$d\mathbf{x} = \left( f_t(\mathbf{x}) - \frac{1}{2}(g_t^2 + \sigma_t^2) \nabla_{\mathbf{x}_t} \log p_t(\mathbf{x}) \right) dt + \sigma_t d\mathbf{w}. \quad (9.17)$$

Here, we are allowed to choose  $\sigma_t$  freely. Both DDPM and DDIM can be considered as its special cases. Especially when  $\sigma_t = 0$ , this corresponds to a generalised DDIM. It can be seen that, in the reverse SDE, the term that is input-dependent is  $\nabla_{\mathbf{x}_t} \log p_t(\mathbf{x})$ . If now we want to perform conditional generation, we should just need to change it to  $\nabla_{\mathbf{x}} \log p_t(\mathbf{x}|\mathbf{y})$ . From Bayes’ rule, we have

$$\nabla_{\mathbf{x}} \log p_t(\mathbf{x}|\mathbf{y}) = \nabla_{\mathbf{x}} \log p_t(\mathbf{x}) + \nabla_{\mathbf{x}} \log p_t(\mathbf{y}|\mathbf{x}), \quad (9.18)$$

noting that  $\nabla_{\mathbf{x}} \log p_t(\mathbf{y}) = 0$ . With the parameterisation trick that we had used before, we have  $\nabla_{\mathbf{x}} \log p_t(\mathbf{x}) = -\boldsymbol{\varepsilon}_{\boldsymbol{\theta}}(\mathbf{x}_t, t)/\bar{\beta}_t$ , thus we have

$$\nabla_{\mathbf{x}} \log p_t(\mathbf{x}|\mathbf{y}) = -\frac{\boldsymbol{\varepsilon}_{\boldsymbol{\theta}}(\mathbf{x}_t, t)}{\bar{\beta}_t} + \nabla_{\mathbf{x}} \log p_t(\mathbf{y}|\mathbf{x}) = -\frac{\boldsymbol{\varepsilon}_{\boldsymbol{\theta}}(\mathbf{x}_t, t) - \bar{\beta}_t \nabla_{\mathbf{x}} \log p_t(\mathbf{y}|\mathbf{x})}{\bar{\beta}_t}. \quad (9.19)$$

This means that, regardless of what the variance  $\sigma_t$  is, all we need to do is to replace  $\boldsymbol{\varepsilon}_{\boldsymbol{\theta}}(\mathbf{x}_t, t)$  with  $\boldsymbol{\varepsilon}_{\boldsymbol{\theta}}(\mathbf{x}_t, t) - \bar{\beta}_t \nabla_{\mathbf{x}} \log p_t(\mathbf{y}|\mathbf{x})$ , and we can realise conditional generation. As such, under the unified framework of SDE, we could arrive at the generation framework of Classifier-Guidance very easily.

### 9.7. Classifier-free approach

Finally, we will briefly introduce the Classifier-Free approach. It is actually very simple, we just need to define

$$p(\mathbf{x}_{t-1}|\mathbf{x}_t, \mathbf{y}) = \mathcal{N}(\mathbf{x}_{t-1}; \boldsymbol{\mu}(\mathbf{x}_t, \mathbf{y}), \sigma_t^2 \mathbf{I}). \quad (9.20)$$

Using the DDPM results from the previous sections, we can parameterise  $\boldsymbol{\mu}(\mathbf{x}_t, \mathbf{y})$  generally as

$$\boldsymbol{\mu}(\mathbf{x}_t, \mathbf{y}) = \frac{1}{\alpha_t} \left( \mathbf{x}_t - \frac{\beta_t^2}{\bar{\beta}_t^2} \boldsymbol{\varepsilon}_\theta(\mathbf{x}_t, \mathbf{y}, t) \right), \quad (9.21)$$

with the corresponding loss function

$$\mathbb{E}_{\mathbf{x}_0, \mathbf{y} \sim \tilde{p}(\mathbf{x}_0, \mathbf{y}), \boldsymbol{\varepsilon} \sim \mathcal{N}(0, \mathbf{I})} [\|\boldsymbol{\varepsilon}\|_2^2 - \boldsymbol{\varepsilon}_\theta(\mathbf{x}_t, \mathbf{y}, t)]. \quad (9.22)$$

The advantage here is that the additional input  $\mathbf{y}$  is already included in the training. Theoretically, the more information we have, the easier it is to train. On the other hand, this also brings a major disadvantage, whereby the model needs to be retrained for every new set of conditional input signals.

More specifically, like the Classifier-Guidances, the Classifier-Free approach also introduced a  $\gamma$ -parameter to balance specificity and diversity. Practically, we rewrite Eq. (9.8) using the following modified mean:

$$\boldsymbol{\mu}(\mathbf{x}_t) + \sigma_t^2 \gamma \nabla_{\mathbf{x}_t} \log p(\mathbf{y}|\mathbf{x}_t) = \gamma [\boldsymbol{\mu}(\mathbf{x}_t) + \sigma_t^2 \nabla_{\mathbf{x}_t} \log p(\mathbf{y}|\mathbf{x}_t)] - (\gamma - 1) \boldsymbol{\mu}(\mathbf{x}_t), \quad (9.23)$$

which shows that the Classifier-Free approach is equivalent to the usage of a trainable model to directly regress the term  $\boldsymbol{\mu}(\mathbf{x}_t) + \sigma_t^2 \nabla_{\mathbf{x}_t} \log p(\mathbf{y}|\mathbf{x}_t)$ . In comparison, we can also introduce  $\omega = \gamma - 1$  in the Classifier-Guidance and use the following

$$\hat{\boldsymbol{\varepsilon}}_\theta = (1 + \omega) \boldsymbol{\varepsilon}_\theta(\mathbf{x}_t, \mathbf{y}, t) - \omega \boldsymbol{\varepsilon}_\theta(\mathbf{x}_t, t) \quad (9.24)$$

to replace  $\boldsymbol{\varepsilon}_\theta(\mathbf{x}_t, \mathbf{y}, t)$  in the generation. How should we obtain the unconditioned  $\boldsymbol{\varepsilon}_\theta(\mathbf{x}_t, t)$  in this case? We can introduce a specific input  $\phi$ , which the target image is all the image samples. Once this is added for model training, we can treat that  $\boldsymbol{\varepsilon}_\theta(\mathbf{x}_t, t) = \boldsymbol{\varepsilon}_\theta(\mathbf{x}_t, \phi, t)$ .

## 10. Unified Diffusion Model: The Theoretical Framework

Through the previous sections, we had provided a thorough introduction to the diffusion generative models. It can be seen that there were a lot of theoretical contents involved. Nevertheless, it is clear that in all previous derivations, we have primarily been focusing on diffusion models that dealt with samples that were continuous, and the forward process was built upon a noising process draws noises from a standard normal distribution. In this section, we hope to establish a framework of Unified Diffusion Model (UDM) that may overcome the above limitations, in which

Original blog post see <https://www.spaces.ac.cn/archives/9262>

1. We impose no restriction on the the type of the samples, it can be either discrete or continuous.
2. We impose no restriction on how to realise the forward process. It could be noising, blurring, masking or deletions.
3. We impose no restriction on the nature of the time variable  $t$ , it can be either discrete or continuous.
4. It will include all the known results, including DDPM, DDIM, SDE and ODE that had been discussed previously.

It is the aim of this section that we will develop a unified framework that satisfies the above desires.

### 10.1. The forward process

From the discussions in the previous sections, it is clear that, in order to build a diffusion model, we will need three components, namely, the forward deconstruction process, the reverse

reconstruction process and a loss function for training. In this subsection, we begin our analysis on the forward process.

In the earliest DDPM model, the forward process was established based on the transition probability  $p(\mathbf{x}_t|\mathbf{x}_{t-1})$ . In the subsequent works on DDIM, it was realised that both the training targets and the generative process do not have any direct connection with  $p(\mathbf{x}_t|\mathbf{x}_{t-1})$ , but the probability  $p(\mathbf{x}_t|\mathbf{x}_0)$ . It was also found that, to derive  $p(\mathbf{x}_t|\mathbf{x}_0)$  from  $p(\mathbf{x}_t|\mathbf{x}_{t-1})$  was also rather difficult. Therefore, **a more practical approach is to directly treat  $p(\mathbf{x}_t|\mathbf{x}_0)$  as the forward process.**

The main purpose of establishing  $p(\mathbf{x}_t|\mathbf{x}_0)$  is to build the training data for the diffusion model, so **the basic requirement for  $p(\mathbf{x}_t|\mathbf{x}_0)$  is that sampling from such a distribution must be easy.** For this purpose, we could re-parameterise the forward process as

$$\mathbf{x}_t = \mathcal{F}_t(\mathbf{x}_0, \boldsymbol{\varepsilon}), \quad (10.1)$$

in which  $\mathcal{F}$  is a function of  $t$ ,  $\mathbf{x}_0$  and  $\boldsymbol{\varepsilon}$ . In particular,  $\boldsymbol{\varepsilon}$  is a random variable that is sampled from some standard distribution  $q(\boldsymbol{\varepsilon})$ , with normal distribution being the most common choice, although other distributions are also feasible. It could be anticipated that Eq. (10.1) is **a generalisation of the forward process that includes all possible transformations from  $\mathbf{x}_0$  to  $\mathbf{x}_t$ , without any restriction on the data type.** Generally, the only restriction is imposed on  $t$ , such that the smaller the time variable  $t$  is,  $\mathcal{F}_t(\mathbf{x}_0, \boldsymbol{\varepsilon})$  will contain more information about  $\mathbf{x}_0$ , and it becomes easier to reconstruct  $\mathbf{x}_0$  from  $\mathcal{F}_t(\mathbf{x}_0, \boldsymbol{\varepsilon})$ . Conversely, it becomes more difficult to reconstruct  $\mathbf{x}_0$  at large  $t$ . At a certain upper limit  $T$ ,  $\mathcal{F}_T(\mathbf{x}_0, \boldsymbol{\varepsilon})$  will contain almost no information about  $\mathbf{x}_0$ , from which it becomes literally impossible to reconstruct  $\mathbf{x}_0$ .

## 10.2. The reverse process

In the reverse process, we aim to generate realistic samples through multiple steps of denoising. The key quantity involves is the reverse transition probability  $p(\mathbf{x}_{t-1}|\mathbf{x}_t)$ . In general, we have the following:

$$p(\mathbf{x}_{t-1}|\mathbf{x}_t) = \int p(\mathbf{x}_{t-1}|\mathbf{x}_t, \mathbf{x}_0)p(\mathbf{x}_0|\mathbf{x}_t)d\mathbf{x}_0. \quad (10.2)$$

If  $\mathbf{x}_0$  are discrete data, then we can write Eq. (10.2) equivalently by changing the integral into a discrete summation. Since we would like the sampling from  $p(\mathbf{x}_{t-1}|\mathbf{x}_t)$  to be easy to achieve numerically, we will also require the sampling from both  $p(\mathbf{x}_{t-1}|\mathbf{x}_t, \mathbf{x}_0)$  and  $p(\mathbf{x}_0|\mathbf{x}_t)$  to be easy. In this case, the sampling from the distribution  $p(\mathbf{x}_{t-1}|\mathbf{x}_t)$  can be achieved via the following procedure:

$$\hat{\mathbf{x}}_0 \sim p(\mathbf{x}_0|\mathbf{x}_t) \quad \& \quad \mathbf{x}_{t-1} \sim p(\mathbf{x}_{t-1}|\mathbf{x}_t, \mathbf{x}_0 = \hat{\mathbf{x}}_0) \quad \rightarrow \quad p(\mathbf{x}_{t-1}|\mathbf{x}_t). \quad (10.3)$$

From this formalism, every sampling step  $\mathbf{x}_t \rightarrow \mathbf{x}_{t-1}$  contains two sub-steps:

1. **Estimation:** make an estimate of  $\mathbf{x}_0$  from  $p(\mathbf{x}_0|\mathbf{x}_t)$ .
2. **Correction:** put the estimated  $\mathbf{x}_0$  into  $p(\mathbf{x}_{t-1}|\mathbf{x}_t, \mathbf{x}_0)$ , which will be incorporated with  $p(\mathbf{x}_0|\mathbf{x}_t)$  to obtain  $p(\mathbf{x}_{t-1}|\mathbf{x}_t)$ .

Hence, reverse sampling is an iterative process of estimation followed by correction. By iteratively incorporating the estimate of  $\mathbf{x}_t \rightarrow \mathbf{x}_0$  into the reverse process, we obtain the corrected regression sequence of  $\mathbf{x}_T \rightarrow \cdots \mathbf{x}_t \rightarrow \mathbf{x}_{t-1} \rightarrow \cdots \rightarrow \mathbf{x}_0$ . This breaks the generative process, which is difficult to be achieved in one-shot, into many sequentially dependent small steps.

## 10.3. Training target

Of course, this proposed reverse sampling process is rather abstractive, as we do not have any knowledge on the distributions of either  $p(\mathbf{x}_{t-1}|\mathbf{x}_t, \mathbf{x}_0)$  or  $p(\mathbf{x}_0|\mathbf{x}_t)$ . In this subsection, we will discuss the choice of  $p(\mathbf{x}_0|\mathbf{x}_t)$ . Obviously,  $p(\mathbf{x}_0|\mathbf{x}_t)$  is the probabilistic model that allows one to predict  $\mathbf{x}_0$  from  $\mathbf{x}_t$ , and it must be a distribution that is easy to be calculated and sampled from. When  $\mathbf{x}_0$  is continuous data, there are not many choices of suitable distributions, and the most common choice is a normal distribution with a trainable mean:

$$p(\mathbf{x}_0|\mathbf{x}_t) \approx q(\mathbf{x}_t|\mathbf{x}_0) = \mathcal{N}(\mathbf{x}_0; \mathcal{G}_t(\mathbf{x}_t), \bar{\sigma}_t^2 \mathbf{I}). \quad (10.4)$$

Reminder: The advantage of writing  $p(\mathbf{x}_{t-1}|\mathbf{x}_t)$  in this integral form is that, the conditional probability  $p(\mathbf{x}_{t-1}|\mathbf{x}_t, \mathbf{x}_0)$  has a much well defined solution space, and we can approximate  $p(\mathbf{x}_0|\mathbf{x}_t)$  with  $q(\mathbf{x}_t|\mathbf{x}_0)$  from the forward process.

To reduce the training cost, we usually do not treat the standard deviation  $\bar{\sigma}_t^2$  as a trainable, but using the approaches outlined in Section 7 to estimate it afterward. On the other hand, when  $\mathbf{x}_0$  are discrete data, we can use either auto-regressive or non-auto-regressive language models (Seq2Seq) to build our model. Both building the probabilistic models and samplings are relatively much easier for discrete samples. With a concrete form of  $q(\mathbf{x}_0|\mathbf{x}_t)$ , the training target is easy to establish. A natural choice is to use the cross-entropy

$$\mathbb{E}_{\mathbf{x}_0 \sim \tilde{p}(\mathbf{x}_0), \mathbf{x}_t \sim p(\mathbf{x}_t|\mathbf{x}_0)} [-\log q(\mathbf{x}_0|\mathbf{x}_t)] = \mathbb{E}_{\mathbf{x}_0 \sim \tilde{p}(\mathbf{x}_0), \varepsilon \sim q(\varepsilon)} [-\log q(\mathbf{x}_0|\mathcal{F}_t(\mathbf{x}_0, \varepsilon))]. \quad (10.5)$$

As such, this solves the problem of estimating  $p(\mathbf{x}_0|\mathbf{x}_t)$  and designing the corresponding training target. If  $q(\mathbf{x}_0|\mathbf{x}_t)$  follows the normal distribution, then after eliminating the constants, the training target simplifies to

$$\mathbb{E}_{\mathbf{x}_0 \sim \tilde{p}(\mathbf{x}_0), \varepsilon \sim q(\varepsilon)} \left[ \frac{1}{2\sigma_t^2} \|\mathbf{x}_0 - \mathcal{G}_t(\mathcal{F}_t(\mathbf{x}_0, \varepsilon))\|_2^2 \right]. \quad (10.6)$$

#### 10.4. Conditional probability

Now we are left with the term  $p(\mathbf{x}_{t-1}|\mathbf{x}_t, \mathbf{x}_0)$  [in Eq. (10.2)] to be dealt with, which is the probability of estimating  $\mathbf{x}_{t-1}$  from  $\mathbf{x}_t$  and  $\mathbf{x}_0$ . There is also some room for designing this probability distribution. However, any given distributions for  $p(\mathbf{x}_{t-1}|\mathbf{x}_t, \mathbf{x}_0)$  must satisfy the following equality for marginal distribution:

$$\int p(\mathbf{x}_{t-1}|\mathbf{x}_t, \mathbf{x}_0) p(\mathbf{x}_t|\mathbf{x}_0) d\mathbf{x}_t = p(\mathbf{x}_{t-1}|\mathbf{x}_0). \quad (10.7)$$

Obviously, the simplest choice that satisfies the above constrain is to let

$$p(\mathbf{x}_{t-1}|\mathbf{x}_t, \mathbf{x}_0) = p(\mathbf{x}_{t-1}|\mathbf{x}_0), \quad (10.8)$$

which means now we have made  $p(\mathbf{x}_{t-1}|\mathbf{x}_t, \mathbf{x}_0)$  become completely independent from  $\mathbf{x}_t$ . In principle, there is no problem of making such a choice. However, the generated outcome based on such a simplification is usually not very good. This is because  $\mathbf{x}_{t-1}$  now becomes completely dependent upon  $\mathbf{x}_0$ , which represents the original sample. In the reverse process, however, we can only sample  $\mathbf{x}_0$  approximately from the approximated distribution of  $q(\mathbf{x}_0|\mathbf{x}_t)$ . Since  $q(\mathbf{x}_0|\mathbf{x}_t)$  is usually not very accurate, errors in the generative process will continuously accumulate throughout. At the same time, the distribution  $p(\mathbf{x}_{t-1}|\mathbf{x}_0)$  will carry noises in the sampling process, which may significantly affect the estimated  $\hat{\mathbf{x}}_0$ , thus worsening the generative outcomes.

Fortunately, under most situations, we can derive a new result based on this simple choice of  $p(\mathbf{x}_{t-1}|\mathbf{x}_t, \mathbf{x}_0) = p(\mathbf{x}_{t-1}|\mathbf{x}_0)$ . Based on Eq. (10.1), we have the following:

$$\begin{aligned} \mathbf{x}_{t-1} \sim p(\mathbf{x}_{t-1}|\mathbf{x}_0) &\Leftrightarrow \mathbf{x}_{t-1} = \mathcal{F}_{t-1}(\mathbf{x}_0, \varepsilon) \\ \mathbf{x}_t \sim p(\mathbf{x}_t|\mathbf{x}_0) &\Leftrightarrow \mathbf{x}_t = \mathcal{F}_t(\mathbf{x}_0, \varepsilon). \end{aligned} \quad (10.9)$$

Assuming that  $\mathcal{F}_t(\mathbf{x}_0, \varepsilon)$  is an invertible function of  $\varepsilon$ , then we have the solution  $\varepsilon = \mathcal{F}_t^{-1}(\mathbf{x}_0, \mathbf{x}_t)$ . With this, we can now replace  $\varepsilon$  in  $\mathcal{F}_{t-1}(\mathbf{x}_0, \varepsilon)$ , which gives us

$$\mathbf{x}_{t-1} = \mathcal{F}_{t-1}(\mathbf{x}_0, \mathcal{F}_t^{-1}(\mathbf{x}_0, \mathbf{x}_t)), \quad (10.10)$$

which is equivalent to

$$p(\mathbf{x}_{t-1}|\mathbf{x}_t, \mathbf{x}_0) = \delta(\mathbf{x}_{t-1} - \mathcal{F}_{t-1}(\mathbf{x}_0, \mathcal{F}_t^{-1}(\mathbf{x}_0, \mathbf{x}_t))). \quad (10.11)$$

This is a design of  $p(\mathbf{x}_{t-1}|\mathbf{x}_t, \mathbf{x}_0)$  that simultaneously depends on both  $\mathbf{x}_0$  and  $\mathbf{x}_t$ , in which some dependency of  $\mathbf{x}_{t-1}$  on  $\mathbf{x}_0$  has been spread to  $\mathbf{x}_t$ , and we also eliminated the noise, such that we have a more stabilised reversed process to achieve better generative outcomes. Furthermore, if  $q(\varepsilon)$  was a normal distribution, we may arrive at a more general result. Based on the superposition of normal distributions, we get

$$\mathbf{x}_{t-1} = \mathcal{F}_{t-1}(\mathbf{x}_0, \varepsilon) \Leftrightarrow \mathbf{x}_{t-1} = \mathcal{F}_{t-1}\left(\mathbf{x}_0, \sqrt{1 - \bar{\sigma}_t^2} \varepsilon_1 + \bar{\sigma}_t \varepsilon_2\right). \quad (10.12)$$

In this case, the solution of  $\varepsilon$  from  $\mathbf{x}_0$  and  $\mathbf{x}_t$  can only be used to substitute one of  $\varepsilon_1$  and  $\varepsilon_2$ .

Reminder: The cross-entropy between two distributions  $p(x)$  and  $q(x)$  is defined as  $-\mathbb{E}_{x \sim p(x)} [\log q(x)] = -\int p(x) \log q(x) dx$ , in which  $\mathbb{E}_{x \sim p(x)}$  represents the expectation value taken over the true data distribution. In this case, Eq. (10.5) is trying to minimise the discrepancy between the distribution  $p(\mathbf{x}_0|\mathbf{x}_t)$  that is what we try to learn and the true distribution  $q(\mathbf{x}_t|\mathbf{x}_0)$  from training data generated from forward process.

This result is analogous to the general expression for  $p(\mathbf{x}_{t-1}|\mathbf{x}_t, \mathbf{x}_0)$  in Eq. (4.7) from DDIM.

The final sampling process follows:

$$\mathbf{x}_{t-1} = \mathcal{F}_{t-1} \left( \mathbf{x}_0, \sqrt{1 - \tilde{\sigma}_t^2} \mathcal{F}_t^{-1}(\mathbf{x}_0, \mathbf{x}_t) + \tilde{\sigma}_t \boldsymbol{\varepsilon} \right). \quad (10.13)$$

## 10.5. The thought process

By now, we have completed the building of a unified theoretical framework for the diffusion generative model. In the next section, we will provide some concrete examples of how to derive the results of the existing diffusion models from this unified framework, as well as some new outcomes from the new framework. Here, we will provide some further analysis on the derivation process that has been presented in this section.

Some readers may feel confused with the derivations here. This is because it was established based on author's understanding on all the theoretical details of the diffusion model that have been discussed in the previous nine sections, from which a unified framework may be established. It is not technically challenging but logically not trivial to be understood. Foremost, our goal is to design a unified theoretical framework that satisfies the targets outlined at the beginning of this section. The key in our design is to **understand and control parts of the model that can be freely adjusted and those that must be fixed under certain constraints**.

More specifically, for those readers who are familiar with the diffusion model, they would appreciate that the core idea behind the diffusion model is to “*learn how to reconstruct from destruction*”. As such, we could choose any arbitrary destruction protocols in theory, but must be able to learn how to reconstruct from it. Nevertheless, the destruction process is not completely free of constraints. Generally, one must destruct the samples progressively in order for us to learn how to rebuild them, also progressively. This allows us to design the forward destruction process as presented in Eq. (10.1), where  $\mathcal{F}$  can be any destruction protocol and  $t$  represents the progress of the destruction process. There is no specific constraints on the type of the original data  $\mathbf{x}_0$ . Finally,  $\boldsymbol{\varepsilon}$  encodes the random factor that may exist in the forward process. With these, we have established the most generic form of the destruction process.

As for the backward reconstruction process, we first deconstructed  $p(\mathbf{x}_{t-1}|\mathbf{x}_t)$  with Eq. (10.2) derived from the probability theory, which can be regarded as a constraint. How did it come up? In retrospect, since the forward process follows  $\mathbf{x}_0 \rightarrow \mathbf{x}_t$ , then it is natural to match it with the reverse process with  $\mathbf{x}_t \rightarrow \mathbf{x}_0$ . Eq. (10.2) contains two parts, in which  $p(\mathbf{x}_0|\mathbf{x}_t)$  is very clear, it represents the probability of predicting  $\mathbf{x}_0$  from  $\mathbf{x}_t$ , which does not have much room for further simplification and can only be acquired from a learnable model. The freedom comes from  $p(\mathbf{x}_{t-1}|\mathbf{x}_t, \mathbf{x}_0)$ , which must be a distribution that is easy to be sampled from, while satisfying the constraint imposed by Eq. (10.7) from the probability theory, which had taken the author a rather long time to come up with a satisfactory solution.

## 11. Unified Diffusion Model: Applications

In the previous section, we have tried to establish a UDM, which allows the application of a more generic diffusion model and data type. In this section, we will be providing some concrete examples to showcase how we may achieve our model design targets.

Original blog post see <https://www.spaces.ac.cn/archives/9271>

### 11.1. Recap on the general framework

Firstly, UDM builds the forward process by choosing a suitable noise distribution  $q(\boldsymbol{\varepsilon})$  and a transformation  $\mathcal{F}$ :

$$\mathbf{x}_t = \mathcal{F}_t(\mathbf{x}_0, \boldsymbol{\varepsilon}), \quad \boldsymbol{\varepsilon} \sim q(\boldsymbol{\varepsilon}). \quad (11.1)$$

Then, through the following partition

$$\hat{\mathbf{x}}_0 \sim p(\mathbf{x}_0|\mathbf{x}_t) \quad \& \quad \mathbf{x}_{t-1} \sim p(\mathbf{x}_{t-1}|\mathbf{x}_t, \mathbf{x}_0 = \hat{\mathbf{x}}_0), \quad (11.2)$$

we will achieve the sampling  $\mathbf{x}_{t-1} \sim p(\mathbf{x}_{t-1}|\mathbf{x}_t)$  in the reverse process. Here  $p(\mathbf{x}_0|\mathbf{x}_t)$  is the probability of estimating  $\mathbf{x}_0$  from  $\mathbf{x}_t$ , which we can generally approximate it with a simple distribution such as  $q(\mathbf{x}_0|\mathbf{x}_t)$ . In this case, the corresponding training target will be  $-\log q(\mathbf{x}_0|\mathbf{x}_t)$  or some simple variants of it. When  $\mathbf{x}_0$  is of the continuous type, we can choose the conditional normal distribution for  $q(\mathbf{x}_0|\mathbf{x}_t)$ . When  $\mathbf{x}_0$  are discrete data, we could approximate  $q(\mathbf{x}_0|\mathbf{x}_t)$  with either an autoregressive or a non-autoregressive model.

As for the conditional distribution  $p(\mathbf{x}_{t-1}|\mathbf{x}_t, \mathbf{x}_0)$ , the most basic choice is

$$p(\mathbf{x}_{t-1}|\mathbf{x}_t, \mathbf{x}_0) = p(\mathbf{x}_{t-1}|\mathbf{x}_0) \Leftrightarrow \mathbf{x}_{t-1} = \mathcal{F}_{t-1}(\mathbf{x}_0, \boldsymbol{\varepsilon}). \quad (11.3)$$

From this starting point, we can arrive at different results under different assumptions. When  $\mathcal{F}_t(\mathbf{x}_0, \boldsymbol{\varepsilon})$  is an invertible function of  $\boldsymbol{\varepsilon}$ , we can get a solution of  $\boldsymbol{\varepsilon} = \mathcal{F}_t^{-1}(\mathbf{x}_0, \mathbf{x}_t)$ , from which we can establish a sampling process with better certainties:

$$\mathbf{x}_{t-1} = \mathcal{F}_{t-1}(\mathbf{x}_0, \mathcal{F}_t^{-1}(\mathbf{x}_0, \mathbf{x}_t)). \quad (11.4)$$

Moving one step forward, if  $q(\boldsymbol{\varepsilon})$  is a standard normal distribution, then we have

$$\mathbf{x}_{t-1} = \mathcal{F}_{t-1} \left( \mathbf{x}_0, \sqrt{1 - \bar{\sigma}_t^2} \mathcal{F}_t^{-1}(\mathbf{x}_0, \mathbf{x}_t) + \bar{\sigma}_t \boldsymbol{\varepsilon} \right). \quad (11.5)$$

## 11.2. Hot diffusion

Here, we shall first prove that the concept of *hot diffusion* is a special case of UDM, which corresponds to the mainstream diffusion models such as DDPM and DDIM. This idea came from the paper that introduced *cold diffusion*, which will be discussed next.

In the mainstream diffusion models, we are primarily dealing with continuous data, in which the forward process corrupts the data by adding Gaussian noise:

$$\mathbf{x}_t = \bar{\alpha}_t \mathbf{x}_0 + \bar{\beta}_t \boldsymbol{\varepsilon}, \quad \boldsymbol{\varepsilon} \sim \mathcal{N}(0, \mathbf{I}), \quad (11.6)$$

and subsequently we choose  $\mathcal{N}(\mathbf{x}_0; \bar{\boldsymbol{\mu}}(\mathbf{x}_t), \bar{\sigma}_t \mathbf{I})$  for  $q(\mathbf{x}_0|\mathbf{x}_t)$ . Generally, we do not treat  $\bar{\sigma}_t$  as a trainable parameter, such that when we ignore the constant term, we arrive at the following target function:

$$-\log q(\mathbf{x}_0|\mathbf{x}_t) = \frac{1}{2\bar{\sigma}_t^2} \|\mathbf{x}_0 - \bar{\boldsymbol{\mu}}(\mathbf{x}_t)\|_2^2. \quad (11.7)$$

If we further introduce the parameterisation of  $\bar{\boldsymbol{\mu}}(\mathbf{x}_t) = \frac{1}{\bar{\alpha}_t}(\mathbf{x}_t - \bar{\beta}_t \boldsymbol{\varepsilon}_\theta(\mathbf{x}_t, t))$  and combine with  $\mathbf{x}_t = \bar{\alpha}_t \mathbf{x}_0 + \bar{\beta}_t \boldsymbol{\varepsilon}$ , then we have

$$-\log q(\mathbf{x}_0|\mathbf{x}_t) = \frac{\bar{\beta}_t^2}{2\bar{\sigma}_t^2 \bar{\alpha}_t^2} \|\boldsymbol{\varepsilon} - \boldsymbol{\varepsilon}_\theta(\bar{\alpha}_t \mathbf{x}_0 + \bar{\beta}_t \boldsymbol{\varepsilon}, t)\|_2^2. \quad (11.8)$$

Numerical experiments shows that the performance is even better if we ignore the prefactor, such that the final training target is  $\|\boldsymbol{\varepsilon} - \boldsymbol{\varepsilon}_\theta(\bar{\alpha}_t \mathbf{x}_0 + \bar{\beta}_t \boldsymbol{\varepsilon}, t)\|_2^2$ . The choices of the standard derivations follows the discussions in Section 4 and Section 8. Finally, for the distribution of  $p(\mathbf{x}_{t-1}|\mathbf{x}_t, \mathbf{x}_0)$ , we have

$$\begin{aligned} \mathbf{x}_{t-1} &= \bar{\alpha}_{t-1} \mathbf{x}_0 + \bar{\beta}_{t-1} \boldsymbol{\varepsilon} \\ &\sim \bar{\alpha}_{t-1} \mathbf{x}_0 + \sqrt{\bar{\beta}_{t-1}^2 - \bar{\sigma}_t^2} \boldsymbol{\varepsilon}_1 + \bar{\sigma}_t \boldsymbol{\varepsilon}_2, \end{aligned} \quad (11.9)$$

in which  $\boldsymbol{\varepsilon}$ ,  $\boldsymbol{\varepsilon}_1$  and  $\boldsymbol{\varepsilon}_2 \sim \mathcal{N}(0, \mathbf{I})$ . From  $\mathbf{x}_t = \bar{\alpha}_t \mathbf{x}_0 + \bar{\beta}_t \boldsymbol{\varepsilon}$ , we have  $\boldsymbol{\varepsilon} = (\mathbf{x}_t - \bar{\alpha}_t \mathbf{x}_0)/\bar{\beta}_t$ , which can be used for substituting  $\boldsymbol{\varepsilon}_1$ . This leads to the following general form of  $p(\mathbf{x}_{t-1}|\mathbf{x}_t, \mathbf{x}_0)$ :

$$\mathbf{x}_{t-1} = \bar{\alpha}_{t-1} \mathbf{x}_0 + \sqrt{\bar{\beta}_{t-1}^2 - \bar{\sigma}_t^2} \frac{\mathbf{x}_t - \bar{\alpha}_t \mathbf{x}_0}{\bar{\beta}_t} + \bar{\sigma}_t \boldsymbol{\varepsilon}, \quad \boldsymbol{\varepsilon} \sim \mathcal{N}(0, \mathbf{I}). \quad (11.10)$$

On the other hand,  $\hat{\mathbf{x}}_0 \sim p(\mathbf{x}_0|\mathbf{x}_t)$  implies that

$$\hat{\mathbf{x}}_0 = \bar{\boldsymbol{\mu}}(\mathbf{x}_t) + \bar{\sigma}_t \boldsymbol{\varepsilon} = \frac{1}{\bar{\alpha}_t} (\mathbf{x}_t - \bar{\beta}_t \boldsymbol{\varepsilon}_\theta(\mathbf{x}_t, t)) + \bar{\sigma}_t \boldsymbol{\varepsilon}. \quad (11.11)$$

By combining Eq. (11.10) and Eq. (11.11), we get the generating process for all mainstream diffusion models, in which DDPM has chosen  $\bar{\sigma}_t = 0$  and  $\sigma_t = \bar{\beta}_{t-1} \bar{\beta}_t / \bar{\beta}_t$ , DDIM has chosen  $\bar{\sigma}_t = 0$ ,  $\sigma_t = 0$ , whereas in Analytic-DP, the value of  $\bar{\sigma}_t$  had been further optimised.

Note how such a choice corresponds to an invertible function  $\mathcal{F}$  of the noise  $\boldsymbol{\varepsilon}$ .



### 11.3. Cold diffusion

We now move on to show that the idea of *cold diffusion*, introduced in the paper entitled “Cold Diffusion: Inverting Arbitrary Image Transforms without Noise”<sup>18</sup>, is also a special case of the UDM. Cold diffusion also focuses on modelling continuous data, and it can be seen from the paper title that, this method aims at building a forward process with any arbitrary noiseless transformations. According to our best knowledge, this is the first work that tries to build a generic forward process, which has inspired us in formulating the UDM.

Cold diffusion builds the forward process with a definitive transformation  $\mathbf{x}_t = \mathcal{F}(\mathbf{x}_0)$ . To facilitate the subsequent discussions, let us generalise this forward process further by introducing the noise term as

$$\mathbf{x}_t = \mathcal{F}(\mathbf{x}_0) + \sigma\boldsymbol{\varepsilon}, \quad \boldsymbol{\varepsilon} \sim q(\boldsymbol{\varepsilon}). \quad (11.12)$$

Here,  $\mathcal{F}$  represents any arbitrary way to destruct the raw data. For images, this could be either blurring, painting, snowification, and so on. For a definitive transformation, we only need to let  $\sigma \rightarrow 0$ . Subsequently, we choose  $q(\mathbf{x}_0|\mathbf{x}_t)$  as a normal distribution with  $\ell_1$ -norm:

$$q(\mathbf{x}_0|\mathbf{x}_t) = \frac{1}{\mathcal{Z}(\tau)} \int e^{-\|\mathbf{x}_0 - \mathcal{G}_t(\mathbf{x}_t)\|_1/\tau} d\mathbf{x}_0, \quad (11.13)$$

in which  $\mathcal{Z}(\tau)$  is the normalisation constant. If we choose  $\tau$  to be a fixed constant, then ignore the constant term, we have  $-\log q(\mathbf{x}_0|\mathbf{x}_t) \propto \|\mathbf{x}_0 - \mathcal{G}_t(\mathbf{x}_t)\|_1$ . Combining this with  $\mathbf{x}_t = \mathcal{F}(\mathbf{x}_0)$ , we have the following training target, which is to minimise

$$\|\mathbf{x}_0 - \mathcal{G}_t(\mathcal{F}(\mathbf{x}_0))\|_1. \quad (11.14)$$

For the reverse process, the variance in  $q(\mathbf{x}_0|\mathbf{x}_t)$  is completely ignored, *i.e.* we let  $\tau \rightarrow 0$ . As such, we have  $\hat{\mathbf{x}}_0 = \mathcal{G}_t(\mathbf{x}_t)$ . If we further choose  $p(\mathbf{x}_{t-1}|\mathbf{x}_t, \mathbf{x}_0) = p(\mathbf{x}_{t-1}|\mathbf{x}_0)$ , *i.e.*  $\mathbf{x}_{t-1} = \mathcal{F}_{t-1}(\mathbf{x}_0) + \sigma\boldsymbol{\varepsilon}$ , then we substitute in  $\hat{\mathbf{x}}_0$  and take the limit of  $\sigma \rightarrow 0$ , we reached

$$\hat{\mathbf{x}}_0 = \mathcal{G}_t(\mathbf{x}_t), \quad \mathbf{x}_{t-1} = \mathcal{F}_{t-1}(\hat{\mathbf{x}}_0). \quad (11.15)$$

This result corresponds to the **naïve sampling** proposed in the paper. On the other hand, if we started from  $\mathbf{x}_t = \mathcal{F}(\mathbf{x}_0) + \sigma\boldsymbol{\varepsilon}$  and solved  $\boldsymbol{\varepsilon} = (\mathbf{x}_t - \mathcal{F}(\mathbf{x}_0))/\sigma$ , before we substitute this into  $\mathbf{x}_{t-1} = \mathcal{F}_{t-1}(\mathbf{x}_0) + \sigma\boldsymbol{\varepsilon}$ , we arrived at the result of **improved sampling**:

$$\hat{\mathbf{x}}_0 = \mathcal{G}_t(\mathbf{x}_t), \quad \mathbf{x}_{t-1} = \mathbf{x}_t + \mathcal{F}_{t-1}(\hat{\mathbf{x}}_0) - \mathcal{F}_t(\hat{\mathbf{x}}_0). \quad (11.16)$$

Overall, cold diffusion was successful in realising a generic forward diffusion process. However, it over emphasised the ideal of “without noise”, which has led to shortages that are almost impossible to overcome theoretically. For example, when using a blurring operation in the forward process, it blurs an  $(w \times w \times 3)$  dimensional image data into a 3-dimensional vector with a definitive transformation. As the same time, another definitive reverse process is also applied to rebuild the  $3w^2$ -dimensional image from a 3-dimensional vector. It is unavoidable that a lot of informations would have been lost in this process, which limited the resolutions of the reconstructed images. To solve this problem, we cannot eliminate the noise term in the diffusion process. This is because noises represent uncertainties, which promote a “many-to-one” mapping in the forward process. In another word, it allows information loss. As a matter of fact, this problem has been made aware of in the implementation of the cold diffusion, in which a  $3w^3$ -dimensional random noise had been injected into the 3-dimensional vector. Numerical experiments showed that this has indeed led to improved image qualities.

### 11.4. The editing model

The two examples given above are both dealing with continuous data. However, we have mentioned that the purpose of UDM is to not restricting the data type. Therefore, in this subsection, we will discuss an example with discrete data. We aim to show that the text generation model for editing can also be viewed as a special case of UDM.

For simplicity, we consider a sentence with a fixed length  $\ell$ . It is possible also to deal with variable length but it is slightly more difficult. With this, we define the forward process  $\mathbf{x}_t = \mathcal{F}(\mathbf{x}_0, \boldsymbol{\varepsilon})$  as ‘random substitution’, *i.e.*, we randomly select  $t$  tokens in the sentence and substitute them with other tokens. When  $t = \ell$ ,  $\mathbf{x}_t$  is a combination of  $\ell$  random tokens. In this case,  $q(\mathbf{x}_0|\mathbf{x}_t)$  is a model that predicts the original token sequence from a sequence of fully randomised tokens, which can be trained with regressive models and cross-entropy as the loss



function. Note that now the forward process  $\mathcal{F}(\mathbf{x}_0, \varepsilon)$  is certainly not invertible, as there are multiple pathways to evolve  $\mathbf{x}_0$  to  $\mathbf{x}_t$ . As such, we can only apply  $p(\mathbf{x}_{t-1}|\mathbf{x}_t, \mathbf{x}_0) = p(\mathbf{x}_{t-1}|\mathbf{x}_0)$ , which leads to the following generative process:

1. Randomly choose  $\ell$  tokens as the initial  $\mathbf{x}_\ell$ .
2. Make an estimate of  $\hat{\mathbf{x}}_0$  from  $q(\mathbf{x}_0|\mathbf{x}_t)$ .
3. Randomly choose  $\ell - 1$  tokens to substitute other tokens to form  $\mathbf{x}_{t-1}$ .
4. Repeat steps 2 and 3 until a final  $\mathbf{x}_0$  is obtained.

However, this algorithm will not lead to very good outcomes. This is because the estimate from step 2 may be significantly destroyed by the noises from step 3. In order to improve the sampling quality, we need to use a better sampling procedure, this requires  $\mathcal{F}(\mathbf{x}_0, \varepsilon)$  to be an invertible function of  $\varepsilon$ . This means that from a give pair of  $\mathbf{x}_0$  and  $\mathbf{x}_t$ , we can figure out how the forward transformation was achieved. To achieve this, we redefined the forward process as “*randomly selecting  $t$  tokens in the sentence and substitute them with different ones*”. The main difference is now we forces the new tokens must substitute with ones that are different from the original ones. Otherwise, one might have sampled the same tokens as before. With this restriction, we could directly compare  $\mathbf{x}_0$  and  $\mathbf{x}_t$  to figure out what the transformation was. In this case, we change step 3 into a transformation from  $\hat{\mathbf{x}}_0$  to  $\mathbf{x}_t$ :

1. Randomly choose  $\ell$  tokens as the initial  $\mathbf{x}_\ell$ .
2. Make an estimate of  $\hat{\mathbf{x}}_0$  from  $q(\mathbf{x}_0|\mathbf{x}_t)$ , under the condition that  $t$  tokens in  $\hat{\mathbf{x}}_0$  must be different from  $\mathbf{x}_t$ , which may be easily achievable with a non-autoregressive model.
3. Randomly choose a token in  $\mathbf{x}_t$  that is different from  $\hat{\mathbf{x}}_0$ , substitute it with the token of  $\hat{\mathbf{x}}_0$  in the corresponding positions to form  $\mathbf{x}_{t-1}$ .
4. Repeat steps 2 and 3 until a final  $\mathbf{x}_0$  is obtained.

With this sampling procedure, in every sampling step, the part in  $\hat{\mathbf{x}}_0$  that is identical to  $\mathbf{x}_t$  can be kept. Comparing  $\mathbf{x}_{t-1}$  and  $\mathbf{x}_t$ , only one token would have been changed, so we have a much more stable progressive generative process.

### 11.5. The masking model

Here, we provide another example to further help the readers in understanding the models. Similar to above, we consider a sentence with  $\ell$  tokens, and define the forward process  $\mathbf{x}_t = \mathcal{F}(\mathbf{x}_0, \varepsilon)$  as random masking, *i.e.*, we select  $t$  tokens and randomly replace them with [MASK], in which  $t \leq \ell$ . When  $t = \ell$ ,  $\mathbf{x}_t$  is a sentence with  $\ell$  [MASK].

In this case,  $q(\mathbf{x}_0|\mathbf{x}_t)$  is a model that predicts the original sequence from a sequence that contains [MASK]. This can be achieved with non-autoregressive models such as BERT based on MLM, with cross-entropy as the loss function. The basic generative process is:

1. Using a sequence containing  $\ell$  [MASK] as the starting  $\mathbf{x}_\ell$ .
2. Sample  $\hat{\mathbf{x}}_0$  from  $q(\mathbf{x}_0|\mathbf{x}_t)$ .
3. Randomly selecting  $t - 1$  tokens in  $\hat{\mathbf{x}}_0$  and replace it with [MASK], this gives  $\mathbf{x}_{t-1}$ .
4. Repeat steps 2 and 3 until a final  $\mathbf{x}_0$  is obtained.

Noticing that  $\mathcal{F}_t(\mathbf{x}_0, \varepsilon)$  is an invertible function of  $\varepsilon$ , meaning that we can figure our how the sequence has been transformed from  $\mathbf{x}_0$  to  $\mathbf{x}_t$ , *i.e.* which tokens have been changed into [MASK]. In this case, we have the following improved sampling procedure:

1. Using a sequence containing  $\ell$  [MASK] as the starting  $\mathbf{x}_\ell$ .
2. Sample  $\hat{\mathbf{x}}_0$  from  $q(\mathbf{x}_0|\mathbf{x}_t)$ , where we only need to sample for the tokens that were originally the [MASK], and do not change those that were not [MASK].
3. Select  $t-1$  positions randomly from  $\mathbf{x}_t$  in which the  $t$  positions were occupied by [MASK]. Changing those into the corresponding token at  $\hat{\mathbf{x}}_0$  and make this to be  $\mathbf{x}_{t-1}$ .
4. Repeat steps 2 and 3 until a final  $\mathbf{x}_0$  is obtained.

In which step 2 and 3 can be combined into a single step:

Randomly selecting a position from  $t$  possible positions that are occupied by [MASK] in  $\mathbf{x}_t$ , and according to the probability  $q(\mathbf{x}_0|\mathbf{x}_t)$ , sample a token and replace the [MASK] at that position, giving a new  $\mathbf{x}_{t-1}$ .

This is basically the same as the Gibbs sampling in MLM model. From these two examples, we could appreciate that many models based on progressive changes can all be described by the UDM, or equivalently, using USM to guide the design of new models based on progressive changes.

### 11.6. The encoding model

In all the models described above, there exists no trainable parameters in the forward process, meaning these forward processes are pre-designed. This is not completely necessary, which we can generalise the forward diffusion in DDPM into

$$\mathbf{x}_t = \bar{\alpha}_t \mathcal{F}(\mathbf{x}_0) + \bar{\beta}_t \boldsymbol{\varepsilon}, \quad \boldsymbol{\varepsilon} \sim \mathcal{N}(0, \mathbf{I}), \quad (11.17)$$

in which  $\mathcal{F}(\mathbf{x}_0)$  is the encoding model for  $\mathbf{x}_0$ , which can contain trainable parameter. The corresponding training target is

$$-\log q(\mathbf{x}_0|\mathbf{x}_t) = -\log q(\mathbf{x}_0|\bar{\alpha}_t \mathcal{F}(\mathbf{x}_0) + \bar{\beta}_t \boldsymbol{\varepsilon}). \quad (11.18)$$

The only difference is that now  $\mathcal{F}$  is also parameterised. The reverse sampling process is also similar, except now the last step  $\hat{\mathbf{x}}_0 \sim q(\mathbf{x}_0|\mathbf{x}_1)$  will return  $\hat{\mathbf{x}}_0$  directly. Most importantly, because now we have an encoding model  $\mathcal{F}(\mathbf{x}_0)$ , the input data  $\mathbf{x}_0$  could be either continuous or discrete, which is similar to VAE that transforms the data distribution into the normal distribution of the latent variables.

## 12. Conquering the Diffusion ODE

In Section 5, we aimed at developing an understanding of the diffusion model from the perspective of SDE. Subsequently, in Section 6, we showed that, in the diffusion model that was established based on SDE, there exists an underlying ODE model. Coincidentally, in Section 4 where we discussed DDIM, we have demonstrated that the original DDPM model which performs stochastic samples, also implicitly contains a DDIM model, in which the sampling procedure is deterministic, and its continuous limit also corresponds to an ODE.

Thinking of the above arguments in detail, it is not difficult to discover that both DDPM→DDIM and SDE→ODE are procedures that convert a stochastic sampling model into a deterministic one. On the other hand, if our initial goal was directly based on ODE, then these procedures of establishing a deterministic sampling framework from a stochastic starting point are obviously too cumbersome. In this section, we will try to provide a direct derivation for the diffusion model based on ODE, and demonstrate its linkages with the Jacobian and the diffusion equations for heat conduction.

### 12.1. Differential equation

The core idea behind the generative models, such as GAN, is to find a deterministic transformation that is capable of transforming a random variable sampled from a simple probabilistic distribution, such as the standard normal distribution, into a realistic sample that follows a more data-specific distribution. The flow model is another type of generative model, which follows the opposite design concept. It tries to find a deterministic transformation that transforms the unknown data distribution into a simple one, then it solves for the corresponding

Original blog post see <https://www.spaces.ac.cn/archives/9280>

inverse transformation to establish the generative model.

The traditional flow model achieves this invertible transformation through a careful design of the coupling layers. However, it was later realised that the same type of transformation can be achieved with the usage of differential equations, and it is theoretically more elegant. This leads to the new field of neural ODE which builds generative models by combining neural network with differential equations.

Consider the first-order ordinary differential equation on  $\mathbf{x}_t \in \mathbb{R}^d$ :

$$\frac{d\mathbf{x}_t}{dt} = f_t(\mathbf{x}_t). \quad (12.1)$$

Suppose  $t \in [0, T]$ , then with a given  $\mathbf{x}_0$ , (under some conditions that are easy to realise) we could solve for  $\mathbf{x}_T$  deterministically. It implies that this ODE describes a definitive transformation from  $\mathbf{x}_0$  to  $\mathbf{x}_T$ . More specifically, this transformation will be invertible, meaning that we could also solve the ODE in the reverse order to obtain the transformation from  $\mathbf{x}_T$  to  $\mathbf{x}_0$ . Therefore, differential equations lead to an elegant theoretical solution for constructing an invertible transformation.

### 12.2. The Jacobian

Similar to the diffusion models discussed before, we treat  $\mathbf{x}_0$  as a sample from the input data, and  $\mathbf{x}_T$  as a sample drawn from a simple probability distribution. Our goal here is to obtain a transformation from  $\mathbf{x}_0$  to  $\mathbf{x}_T$  through the differential equation.

To do this, we first write out the finite-difference form for the differential equation [Eq. (12.1)] as:

$$\mathbf{x}_{t+\Delta t} - \mathbf{x}_t = f_t(\mathbf{x}_t)\Delta t. \quad (12.2)$$

Since this is a definitive transformation, we have the following expression for the probability distributions at  $\mathbf{x}_t$  and  $\mathbf{x}_{t+\Delta t}$ :

$$p_t(\mathbf{x}_t)d\mathbf{x}_t = p_{t+\Delta t}(\mathbf{x}_{t+\Delta t})d\mathbf{x}_{t+\Delta t} = p_{t+\Delta t}(\mathbf{x}_{t+\Delta t}) \left| \frac{\partial \mathbf{x}_{t+\Delta t}}{\partial \mathbf{x}_t} \right| d\mathbf{x}_t. \quad (12.3)$$

Here,  $\frac{\partial \mathbf{x}_{t+\Delta t}}{\partial \mathbf{x}_t}$  is the Jacobian matrix for the transformation and  $|\cdot|$  takes its determinant. By taking the derivatives on both sides of Eq. (12.2), we have

$$\frac{\partial \mathbf{x}_{t+\Delta t}}{\partial \mathbf{x}_t} = \mathbf{I} + \frac{\partial f_t(\mathbf{x}_t)}{\partial \mathbf{x}_t} \Delta t, \quad (12.4)$$

and based on the definition on the derivative of a determinant, we have

$$\left| \frac{\partial \mathbf{x}_{t+\Delta t}}{\partial \mathbf{x}_t} \right| \approx 1 + \text{Tr} \left[ \frac{\partial f_t(\mathbf{x}_t)}{\partial \mathbf{x}_t} \right] \Delta t = 1 + \nabla_{\mathbf{x}_t} \cdot f_t(\mathbf{x}_t) \Delta t \approx \exp [\nabla_{\mathbf{x}_t} \cdot f_t(\mathbf{x}_t) \Delta t]. \quad (12.5)$$

By taking the logarithm to both sides of Eq. (12.3) with the use of Eq. (12.5), we obtain the following equation for the change in the probability distributions in the finite-difference form:

$$\log p_{t+\Delta t}(\mathbf{x}_{t+\Delta t}) - \log p_t(\mathbf{x}_t) \approx -\nabla_{\mathbf{x}_t} \cdot f_t(\mathbf{x}_t) \Delta t. \quad (12.6)$$

### 12.3. Taylor approximation

Now suppose  $p_t(\mathbf{x}_t)$  defines a series of probability density functions that vary continuously with respect to the time variable  $t$ , in which  $p_0(\mathbf{x}_0)$  is the true sample distribution and  $p_T(\mathbf{x}_T)$  is a simple distribution. When both  $\Delta t$  and  $\mathbf{x}_{t+\Delta t} - \mathbf{x}_t$  are sufficiently small, we can write the following first-order Taylor approximation for Eq. (12.6):

$$\log p_{t+\Delta t}(\mathbf{x}_{t+\Delta t}) - \log p_t(\mathbf{x}_t) \approx (\mathbf{x}_{t+\Delta t} - \mathbf{x}_t) \nabla_{\mathbf{x}_t} \cdot \log p_t(\mathbf{x}_t) + \Delta t \frac{\partial}{\partial t} \log p_t(\mathbf{x}_t). \quad (12.7)$$

Substituting the expression for  $\mathbf{x}_{t+\Delta t} - \mathbf{x}_t$  from Eq. (12.2) and compare the result to Eq. (12.6), we get the equation which  $f_t(\mathbf{x}_t)$  must satisfy:

$$-\nabla_{\mathbf{x}_t} \cdot f_t(\mathbf{x}_t) = f_t(\mathbf{x}_t) \nabla_{\mathbf{x}_t} \cdot \log p_t(\mathbf{x}_t) + \Delta t \frac{\partial}{\partial t} \log p_t(\mathbf{x}_t). \quad (12.8)$$

In other words, any arbitrary function  $f_t(\mathbf{x}_t)$  that satisfies the above equation will be a suitable candidate for constructing the ODE of the form shown in Eq. (12.1), solving which will allow us to achieve the transformation between the data and a simple probability distribution. Rearranging Eq. (12.8), we get the following

$$\frac{\partial}{\partial t} \log p_t(\mathbf{x}_t) = -\nabla_{\mathbf{x}_t} (f_t(\mathbf{x}_t) p_t(\mathbf{x}_t)), \quad (12.9)$$

which is actually the special case of the Fokker-Planck equation introduced in Section 6 when  $g_t = 0$ .

#### 12.4. The equation of heat conduction

Now consider a solution to Eq. (12.9) that is of the following form:

$$f_t(\mathbf{x}_t) = -D_t(\mathbf{x}_t) \nabla_{\mathbf{x}_t} \log p_t(\mathbf{x}_t), \quad (12.10)$$

in which  $D_t(\mathbf{x}_t)$  could be either a matrix or a scalar, depending on the complexity of the problem. Why should we consider this particular form of the solution? The author's initial intention was to match with the results from DDIM. It was later found that, after some generalisation, it can be matched with the diffusion model to be discussed below. In retrospect, if  $D_t(\mathbf{x}_t)$  is a non-negative scalar function and if we substitute Eq. (12.10) into Eq. (12.2), we found that it led to a form that is similar to the case of gradient descent, in which the transformation from  $\mathbf{x}_0$  to  $\mathbf{x}_T$  is a process that progressively searches the regions of low probabilities, and conversely, the transformation from  $\mathbf{x}_T$  to  $\mathbf{x}_0$  gradually leads to the searches over the regions of high probabilities. This follows our basic intuitions, which may be considered as the inspiration behind Eq. (12.10).

Now by substituting Eq. (12.10) into Eq. (12.9), we get

$$\frac{\partial}{\partial t} p_t(\mathbf{x}_t) = \nabla_{\mathbf{x}_t} \cdot \left( D_t(\mathbf{x}_t) \nabla_{\mathbf{x}_t} p_t(\mathbf{x}_t) \right). \quad (12.11)$$

This is the diffusion equation in differential equations. Here, we will only consider the most simplified case, in which  $D_t(\mathbf{x}_t)$  is a scalar function that is independent on  $\mathbf{x}_t$ , such that the diffusion equation simplifies into

$$\frac{\partial}{\partial t} p_t(\mathbf{x}_t) = D_t \nabla_{\mathbf{x}_t}^2 p_t(\mathbf{x}_t). \quad (12.12)$$

This gives the equation for heat conduction, which forms the basis for our subsequent discussions.

#### 12.5. Solving for the probability distribution

Using the Fourier transformation, we can convert the equation for heat conduction into an ODE, from which we can solve for  $p_t(\mathbf{x}_t)$ , and the result is

$$p_t(\mathbf{x}_t) = \int \frac{1}{(2\pi\sigma_t^2)^{\frac{d}{2}}} \exp\left(-\frac{\|\mathbf{x}_t - \mathbf{x}_0\|_2^2}{2\sigma_t^2}\right) p_0(\mathbf{x}_0) d\mathbf{x}_0 = \int \mathcal{N}(\mathbf{x}_t; \mathbf{x}_0, \sigma_t^2 \mathbf{I}) p_0(\mathbf{x}_0) d\mathbf{x}_0, \quad (12.13)$$

in which  $\sigma_t^2 = 2 \int_0^t D_s ds$  or  $D_t = \dot{\sigma}_t \sigma_t$  (with  $\sigma_0 = 0$ ). From which we see that the solution is a Gaussian mixture model with  $p_0(\mathbf{x}_0)$  as the initial distribution.

**The Procedure for solving Eq. (12.12)** Here, we shall briefly introduce the procedure for solving the equation of heat conduction, which can be skipped by the readers who are not interested or who are familiar with the equation.

It is not difficult to solve Eq. (12.12) with the use of Fourier transformation. Taking the Fourier transformation to the variable  $\mathbf{x}_t$  on both sides of the equation and use the fact that  $\nabla_{\mathbf{x}_t} \rightarrow i\omega$ , the result is:

$$\frac{\partial}{\partial t} \mathcal{F}_t(\omega) = -D_t \omega^2 \mathcal{F}_t(\omega). \quad (12.14)$$

This is an ordinary differential equation with the following solution:

$$\mathcal{F}_t(\omega) = \mathcal{F}_0(\omega) \exp\left(-\frac{\sigma_t^2 \omega^2}{2}\right), \quad (12.15)$$

in which  $\sigma_t^2 = 2 \int_0^t D_s ds$ , and  $\mathcal{F}_0(\omega)$  is the Fourier transformation of  $p_0(\mathbf{x}_0)$ . Now we take the inverse Fourier transformation on both size, for which  $\mathcal{F}_t(\omega)$  transforms back to  $p_t(\mathbf{x}_t)$ ,  $\mathcal{F}_0(\omega)$  transforms back to  $p_0(\mathbf{x}_0)$  and  $\exp(-\sigma_t^2 \omega^2/2)$  corresponds to the normal distribution  $\mathcal{N}(\mathbf{x}_t; \mathbf{x}_0, \sigma_t^2 \mathbf{I})$ . The final form of the solution given in Eq. (1.12) can be obtained by following the convolution property of the Fourier transformation.

## 12.6. Completing the model design

Let us briefly summarise the results so far. From solving the equation for heat conduction, we have obtained

$$p_t(\mathbf{x}_t) = \int \mathcal{N}(\mathbf{x}_t; \mathbf{x}_0, \sigma_t^2 \mathbf{I}) p_0(\mathbf{x}_0) d\mathbf{x}_0, \quad (12.16)$$

which corresponds to the solution for the following equation

$$\frac{d\mathbf{x}_t}{dt} = -\dot{\sigma}_t \sigma_t \nabla_{\mathbf{x}_t} \log p_t(\mathbf{x}_t). \quad (12.17)$$

This gives a definitive transformation from  $p_0(\mathbf{x}_0)$  to  $p_T(\mathbf{x}_T)$ . If  $p_T(\mathbf{x}_T)$  is a distribution that is easy to sample from, and we know how to compute the score function  $\nabla_{\mathbf{x}_t} \log p_t(\mathbf{x}_t)$ , then we can perform a random sampling from  $\mathbf{x}_T \sim p_T(\mathbf{x}_T)$ , followed by solving the differential equation in the reverse order, to synthesise new sample  $\mathbf{x}_0 \sim p_0(\mathbf{x}_0)$ .

The first equation is, when is  $p_T(\mathbf{x}_T)$  a distribution that is easy to sample from? Based on Eq. (12.16), we know that

$$\mathbf{x}_T \sim p_T(\mathbf{x}_T) \Leftrightarrow \mathbf{x}_T = \mathbf{x}_0 + \sigma_T \varepsilon, \quad \mathbf{x}_0 \sim p_0(\mathbf{x}_0), \quad \varepsilon \sim \mathcal{N}(0, \mathbf{I}). \quad (12.18)$$

When  $\sigma_T$  is sufficiently large, the influence of  $\mathbf{x}_0$  upon  $\mathbf{x}_T$  will be negligible, at which point we can simply take

$$\mathbf{x}_T \sim p_T(\mathbf{x}_T) \Leftrightarrow \mathbf{x}_T = \sigma_T \varepsilon \quad \varepsilon \sim \mathcal{N}(0, \mathbf{I}), \quad (12.19)$$

which makes  $p_T(\mathbf{x}_T)$  a distribution that is easy to sample from. As such, the choice of  $\sigma_t$  should follow the general rule of  $\sigma = 0$ ,  $\sigma_T \gg 1$ , with  $\sigma_t$  being a continuous growing function over  $[0, T]$ .

The second question is, how should we compute  $\nabla_{\mathbf{x}_t} \log p_t(\mathbf{x}_t)$ ? This is the same as the score-matching that was discussed in Section 5, in which we use a neural network  $s_\theta(\mathbf{x}_t, t)$  to surrogate it, with the following training target:

$$\mathbb{E}_{\mathbf{x}_0, \mathbf{x}_t \sim \mathcal{N}(\mathbf{x}_t; \mathbf{x}_0, \sigma_t^2 \mathbf{I}) p_0(\mathbf{x}_0)} \left[ \|s_\theta(\mathbf{x}_t, t) - \nabla_{\mathbf{x}_t} \log \mathcal{N}(\mathbf{x}_t; \mathbf{x}_0, \sigma_t^2 \mathbf{I})\|_2^2 \right]. \quad (12.20)$$

This is called the conditional score matching, which has been derived before and will not be repeated here.

In summary, we have provided a different approach to derive the ODE-based diffusion model. Starting from the ODE, by combining the Jacobian we obtained the first-order approximation to the probability distribution. By comparing this approximation to that obtained directly from the first-order Taylor expansion, we obtained the equation that the  $f_t(\mathbf{x}_t)$  term should satisfy. Subsequently, this equation can be transformed into the equation of diffusion or heat conduction, that is to be solved. Compared to the previous derivation from the SDE and FP equations, the procedure presented here should be much more straight forward.

## References

- <sup>1</sup>Y. Lipman, R. T. Chen, H. Ben-Hamu, M. Nickel, and M. Le, “Flow matching for generative modeling,” arXiv preprint arXiv:2210.02747 (2022).
- <sup>2</sup>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).
- <sup>3</sup>J. Ho, A. Jain, and P. Abbeel, “Denoising diffusion probabilistic models,” Adv. Neural Info. Proc. Sys. **33**, 6840 (2020).
- <sup>4</sup>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.
- <sup>5</sup>C. Luo, “Understanding diffusion models: A unified perspective,” arXiv:2208.11970 (2022).
- <sup>6</sup>M. Zhang, J. Lucas, J. Ba, and G. E. Hinton, “Lookahead optimizer: k steps forward, 1 step back,” Adv. Neural Info. Proc. Sys. **32** (2019).
- <sup>7</sup>J. Song, C. Meng, and S. Ermon, “Denoising diffusion implicit models,” arXiv:2010.02502 (2020).
- <sup>8</sup>Y. Song, J. Sohl-Dickstein, D. P. Kingma, A. Kumar, S. Ermon, and B. Poole, “Score-based generative modeling through stochastic differential equations,” arXiv:2011.13456 (2020).
- <sup>9</sup>B. D. Anderson, “Reverse-time diffusion equation models,” Stochastic Processes and their Applications **12**, 313 (1982).
- <sup>10</sup>A. Hyvärinen and P. Dayan, “Estimation of non-normalized statistical models by score matching,” J. Mach. Learn. Res. **6** (2005).
- <sup>11</sup>P. Vincent, “A connection between score matching and denoising autoencoders,” Neural Comput. **23**, 1661 (2011).
- <sup>12</sup>A. Q. Nichol and P. Dhariwal, “Improved denoising diffusion probabilistic models,” in *Inter. Conf. Machine Learning* (PMLR, 2021) p. 8162.
- <sup>13</sup>F. Bao, C. Li, J. Zhu, and B. Zhang, “Analytic-DPM: an analytic estimate of the optimal reverse variance in diffusion probabilistic models,” arXiv:2201.06503 (2022).
- <sup>14</sup>F. Bao, C. Li, J. Sun, J. Zhu, and B. Zhang, “Estimating the optimal covariance with imperfect mean in diffusion probabilistic models,” arXiv:2206.07309 (2022).
- <sup>15</sup>P. Dhariwal and A. Nichol, “Diffusion models beat GANs on image synthesis,” Adv. Neural Info. Proc. Sys. **34**, 8780 (2021).
- <sup>16</sup>X. Liu, D. H. Park, S. Azadi, G. Zhang, A. Chopikyan, Y. Hu, H. Shi, A. Rohrbach, and T. Darrell, “More control for free! Image synthesis with semantic diffusion guidance,” in *Proceedings of the IEEE/CVF winter conference on applications of computer vision* (2023) p. 289.
- <sup>17</sup>J. Ho and T. Salimans, “Classifier-free diffusion guidance,” arXiv:2207.12598 (2022).
- <sup>18</sup>A. Bansal, E. Borgnia, H.-M. Chu, J. Li, H. Kazemi, F. Huang, M. Goldblum, J. Geiping, and T. Goldstein, “Cold diffusion: Inverting arbitrary image transforms without noise,” Adv. Neural Info. Proc. Sys. **36**, 41259 (2023).