Topics on Diffusion Generative Models

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

(Dated: 2 October 2025)

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

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

Contents

1. DDPM=Building Demolition and Rebuilding

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

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

June 2022, at the time of writing.

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

1.1. A new starting point

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

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

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

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

1.2. Demolition and rebuilding

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

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

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

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

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

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

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

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

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

1.3. How should we demolish the building?

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

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

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

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

If we repeat the above process, then we have

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

This part is the sum of multiple independent Gaussian noises.

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

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

This means that, effectively, we have

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

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

which has now significantly simplified the computations of $\boldsymbol{x}_t$. On the other hand, by choosing a suitable form of α_t in DDPM to ensure $\bar{\alpha}_t \approx 0$ will guarantee that, after T steps of

demolitions, the entire building will be fully deconstructed into its raw materials ε.

1.4. How to rebuild the building?

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

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

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

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

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

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

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

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

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

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

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

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

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

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

1.5. Reducing the square error

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

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

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

Not quite sure where this comes from?

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

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

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

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

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

Ignoring the weight and constant again, we arrive at:

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

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

This basically says, we must first sample a noise from the normal distribution $\mathcal{N}(0,\boldsymbol{I})$, from which we can generate the noisy sample $\boldsymbol{x}_t=\bar{\alpha}_t\boldsymbol{x}_0+\bar{\beta}_t\boldsymbol{\varepsilon}$. Then we plug this $\boldsymbol{x}_t$ into the denoising network $\varepsilon_\theta(\boldsymbol{x}_t)$ to recover ε_t. This expression is equivalent to the results[5] derived using KL divergence between the destruction and regeneration process: $\operatorname{argmin}_\theta\mathcal{D}_{KL}(q(\boldsymbol{x}_{t-1}|\boldsymbol{x}_t,\boldsymbol{x}_0)\|p(\boldsymbol{x}_{t-1}|\boldsymbol{x}_t))$ from evidence lower bound, which upon further reparameterisation, gives the following expression for the KL divergence: $\frac{(1-\alpha_t)^2}{2\sigma_q^2(t)}\frac{\|\varepsilon_0-\hat{\varepsilon}_\theta(\boldsymbol{x}_t,t)\|_2^2}{(1-\bar{\alpha}_t)\alpha_t}$.

1.6. Regressive generations

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

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

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

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

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

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

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

1.7. Hyperparameter settings

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

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

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

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

make sure the input and output in each time step are sufficiently similar to each other, reducing the blurring effects from using the Euclidean distance in the loss function.

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

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

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

Finally, note that in the loss function [Eq. (1.14), $\varepsilon_\theta(\bar{\alpha}_t \boldsymbol{x}_0 + \bar{\beta}_t \varepsilon, t)$], 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

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

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

2.1. Solving the problem step-wise

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

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

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

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

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

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

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

2.2. The joint KL divergence

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

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:

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

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

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

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

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

2.3. Divide-and-conquer

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

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

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

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

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

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

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

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

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

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

2.4. Reconstructing the diffusion model

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

1. Remove all the constants that are irrelevant to the optimisation problem, this left us with the term $-\log q(\boldsymbol{x}_{t-1}|\boldsymbol{x}_t)$ which gives the term $\frac{1}{2\sigma_t^2}||\boldsymbol{x}_{t-1} - \boldsymbol{\mu}(\boldsymbol{x}_t)||_2^2$ in the target function for optimisation.

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

3. From $\boldsymbol{x}_{t-1} = \frac{1}{\alpha_t}(\boldsymbol{x}_t - \beta_t\boldsymbol{\varepsilon}_t)$, it inspires us to parameterise $\boldsymbol{\mu}(\boldsymbol{x}_t)$ as $\boldsymbol{\mu}(\boldsymbol{x}_t) = \frac{1}{\alpha_t}(\boldsymbol{x}_t - \beta_t\boldsymbol{\varepsilon}_\theta(\boldsymbol{x}_t, t))$.

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

This is equivalent to Eq. (1.10).

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

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

This is equivalent to Eq. (1.14).

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

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

2.5. Hyperparameter settings

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

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

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

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

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

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

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

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

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

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

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

3. DDPM=Bayes Plus Denoising

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

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.

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:

$$\boldsymbol{x} = \boldsymbol{x}_0 \to \boldsymbol{x}_1 \to \boldsymbol{x}_2 \to \cdots \to \boldsymbol{x}_{T-1} \to \boldsymbol{x}_T = \boldsymbol{z} \tag{3.1}$$

in which the process gradually adds noise to the sample $\boldsymbol{x}$ and transform it into a pure noise $\boldsymbol{z}$, whereas the reverse process gradually denoises $\boldsymbol{z}$ to recover the sample data $\boldsymbol{x}$. 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

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

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

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

From which we have $p(\boldsymbol{x}_t|\boldsymbol{x}_0) = \mathcal{N}(\boldsymbol{x}_t; \bar{\alpha}_t \boldsymbol{x}_0, \bar{\beta}_t^2 \boldsymbol{I})$ 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 $p(\boldsymbol{x}_{t-1}|\boldsymbol{x}_t)$ for the reverse generation process using the information above. With this, we could start from any arbitrary $\boldsymbol{z} = \boldsymbol{x}_T$, and through sampling $\boldsymbol{x}_{T-1}$, $\boldsymbol{x}_{T-2}$, ..., $\boldsymbol{x}_1$ stepwise, we could eventually generate a new sample $\boldsymbol{x} = \boldsymbol{x}_0$.

3.2. Using the Bayes' theorem

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

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

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

However, we do not know what $p(\boldsymbol{x}_{t-1})$ and $p(\boldsymbol{x})$ 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 $\boldsymbol{x}_0$. In this case, we have

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

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

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

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

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

This expression is quadratic in $\boldsymbol{x}_{t-1}$, so the finally 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 $\|\boldsymbol{x}_{t-1}\|_2^2$ term is

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

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

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

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

3.3. The denoising process

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

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

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

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

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

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

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

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 $\boldsymbol{x}_t$ to $\boldsymbol{x}_{t-1}$ 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

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

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

sampling process.

To derive this result, note the following:

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

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

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

3.4. Correcting the estimates

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

The problem is that, in reality, estimating $\boldsymbol{x}_0$ from $\bar{\boldsymbol{\mu}}(\boldsymbol{x}_t)$ 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 $p(\boldsymbol{x}_{t-1}|\boldsymbol{x}_t)$ 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*'.[6]

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 $p(\boldsymbol{x}_{t-1})$ and $p(\boldsymbol{x}_t)$ that are numerically intractable. This is because, according to the following definition using the marginalisation:

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

in which we know the distribution $p(\boldsymbol{x}_t|\boldsymbol{x}_0)$ but $\tilde{p}(\boldsymbol{x}_0)$ is not known a priori, so we cannot compute $p(\boldsymbol{x}_t)$ directly, except in two special cases, which will be discussed here. It also provides the proofs for the choices of the standard deviation σ_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 $\tilde{p}(\boldsymbol{x}_0) = \delta(\boldsymbol{x}_0)$. In this case, we have $p(\boldsymbol{x}_t|\mathbf{0}) = p(\boldsymbol{x}_t)$. Substitute this into Eq. (3.4) shows that this corresponds to the conditional distributions $p(\boldsymbol{x}_{t-1}|\boldsymbol{x}_t, \boldsymbol{x}_0)$ under the special condition of $\boldsymbol{x}_0 = 0$. As such, we now have

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

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

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 $\tilde{p}(\boldsymbol{x}_0) = \mathcal{N}(\boldsymbol{x}_0; 0, \boldsymbol{I})$. From the one-shot sampling in the forward diffusion $p(\boldsymbol{x}_t|\boldsymbol{x}_0) = \mathcal{N}(\boldsymbol{x}_t; \bar{\alpha}_t\boldsymbol{x}_0, \bar{\beta}_t\boldsymbol{I})$, we have $\boldsymbol{x}_t = \bar{\alpha}_t\boldsymbol{x}_0 + \bar{\beta}_t\boldsymbol{\varepsilon}$, with $\boldsymbol{\varepsilon} \sim \mathcal{N}(0, \boldsymbol{I})$. Since $\tilde{p}(\boldsymbol{x}_0) \sim \mathcal{N}(0, \boldsymbol{I})$, $p(\boldsymbol{x}_t)$ 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

$$\frac{\|\boldsymbol{x}_t - \alpha_{t-1}\boldsymbol{x}_{t-1}\|^2}{\beta_t^2} + \|\boldsymbol{x}_{t-1}\|^2 - \|\boldsymbol{x}_t\|^2. \tag{3.16}$$

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

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

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

4. DDIM=High Level View of DDPM

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

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"[7], which will be discussed in this section.

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:

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

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 $p(\boldsymbol{x}_t|\boldsymbol{x}_0)$;
2. The sampling process only relies on $p(\boldsymbol{x}_{t-1}|\boldsymbol{x}_t)$.

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

High-Level Viewpoint 1
Since the final results for the diffusion generative model is independent from $p(\boldsymbol{x}_t|\boldsymbol{x}_{t-1})$, 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:

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

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

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

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

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

in which κ_t, λ_t and σ_t are all coefficients to be determined. In order not to retrain the model, we keep $p(\boldsymbol{x}_{t-1}|\boldsymbol{x}_0)$ and $p(\boldsymbol{x}_t|\boldsymbol{x}_0)$ as before. As such, we have in which ε, ε_1, $\varepsilon_2 \sim \mathcal{N}(0,\boldsymbol{I})$, and based on the superposition of normal distributions, we have $\kappa_t\bar{\beta}_t\varepsilon_1 + \sigma_t\varepsilon_2 \sim \sqrt{\kappa_t^2\bar{\beta}_t^2 + \sigma_t^2}\varepsilon$. By comparing the two different sampling approaches for computing $\boldsymbol{x}_{t-1}$, we found that, in order for Eq. (4.3) to hold, we must equate

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

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

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

$$\kappa_t = \frac{\sqrt{\bar{\beta}_{t-1}^2 - \sigma_t^2}}{\bar{\beta}_{t-1}^2}, \qquad \lambda_t = \bar{\alpha}_{t-1} - \frac{\bar{\alpha}_t\sqrt{\bar{\beta}_{t-1}^2 - \sigma_t^2}}{\bar{\beta}_t^2} \tag{4.6}$$

or equivalently,

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

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 $p(\boldsymbol{x}_t|\boldsymbol{x}_{t-1})$, but given $p(\boldsymbol{x}_t|\boldsymbol{x}_0)$ and $p(\boldsymbol{x}_{t-1}|\boldsymbol{x}_0)$, we arrive at a set of solutions for $p(\boldsymbol{x}_{t-1}|\boldsymbol{x}_t, \boldsymbol{x}_0)$ that are dependent on the free parameter σ_t. Using the analogy of demolishing and rebuiding from Section 1, it means now we know how the building will finally be demolished into $[p(\boldsymbol{x}_t|\boldsymbol{x}_0), p(\boldsymbol{x}_{t-1}|\boldsymbol{x}_0)]$, but we do not know how each demolition step $[p(\boldsymbol{x}_t|\boldsymbol{x}_{t-1})]$ is achieved. From here, we want to learn how to rebuild the building through $p(\boldsymbol{x}_{t-1}|\boldsymbol{x}_t)$. Of course, if we want to know how the demolition process works, we could use the Baye's theorem in the opposite way,

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

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

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

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

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

This gives $p(\boldsymbol{x}_{t-1}|\boldsymbol{x}_t)$ 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 σ_t, which brings some fresh results to DDPM.

4.4. A few examples

Refer back to the discussions on the choices of standard deviations σ_t in the previous two sections.

In principle, there is no restriction on the choice of the parameter σ_t. However, different choices of σ_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

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

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 σ_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 σ_t from DDPM.

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

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

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 $\boldsymbol{x}_0$ from a given $\boldsymbol{x}_T = \boldsymbol{z}$ is no longer random, which we shall see, this has some advantages both theoretically and practically.

Recall that in the previous section, the result shown in Eq. (4.11) was obtained by assuming $p(\boldsymbol{x}_t|\boldsymbol{x}_{t-1}) \sim \mathcal{N}\left(\boldsymbol{x}_{t-1}; \frac{\alpha_t\bar{\beta}_{t-1}^2}{\bar{\beta}_t^2}\boldsymbol{x}_t, \frac{\bar{\beta}_{t-1}^2\beta_t^2}{\bar{\beta}_t^2}\boldsymbol{I}\right)$ and then apply the Bayes' rule with the known distributions of $p(\boldsymbol{x}_t|\boldsymbol{x}_0)$ and $p(\boldsymbol{x}_{t-1}|\boldsymbol{x}_0)$. The σ_t parameter can be traced back to the VAE formalism, in which we used the probability $q(\boldsymbol{x}_{t-1}|\boldsymbol{x}_t)$ for the decoder to calculate the KL-divergence. By taking the logarithm of this parameterised normal distribution, we get $-\log[q(\boldsymbol{x}_{t-1}|\boldsymbol{x}_t)] \sim \frac{1}{\sigma_t^2}\|\boldsymbol{x}_{t-1} - \bar{\boldsymbol{\mu}}(\boldsymbol{x}_t)\|_2^2$.

4.5. Accelerated generations

It is worth reiterating that, in this section, we aim to eliminate $p(\boldsymbol{x}_t|\boldsymbol{x}_{t-1})$ in the derivation, as a result, all the expressions derived here become dependent on the hyperparameters $\bar{\alpha}_t$ and $\bar{\beta}_t$, whereas α_t and β_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 $\|\boldsymbol{\varepsilon} - \boldsymbol{\varepsilon}_\theta(\bar{\alpha}_t\boldsymbol{x}_0 + \bar{\beta}_t\boldsymbol{\varepsilon}, t)\|_2^2$, 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 $\tau = [\tau_1, \tau_2, \ldots, \tau_{\dim(\tau)}]$ being an arbitrary sub-sequence of $[1, 2, \ldots, \tau]$. If now we train a DDPM with parameters $\bar{\alpha}_{\tau_1}, \bar{\alpha}_{\tau_2}, \ldots, \bar{\alpha}_{\tau_{\dim(\tau)}}$ with $\dim(\tau)$ steps, then the target functions are a subset of the target functions for the original DDPM model with parameters $\bar{\alpha}_1, \bar{\alpha}_2, \ldots, \bar{\alpha}_\tau$ that is trained for τ 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

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

This is the process of accelerated sampling, which reduces the original τ diffusion steps to $\dim(\tau)$ steps. Notice that we cannot directly replace α_t in Eq. (4.10) by α_{τ_i}, as we mentioned, α_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 σ_{τ_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 τ_i and τ_{i-1}, respectively. For instance,

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

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

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

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

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

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 $\boldsymbol{x}(0)$ given $\boldsymbol{x}(1) \sim \mathcal{N}(0, \boldsymbol{I})$. 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

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

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

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

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:

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

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:

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

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*"[8] 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 Equa-

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

tion (ODE). Although this is a very good paper, it is not well suited for new comers in this field. This 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

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

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:

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

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 $\Delta t \to 0$ for the following equation of finite differences:

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

In a more straightfoward 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 Δt, except that the smaller Δt 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 $\boldsymbol{I}$, 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

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

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 $p(\boldsymbol{x}_t|\boldsymbol{x}_{t+\Delta t})$. Again, from the Baye's rule,

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

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

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

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

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

Note that the last $\frac{\partial}{\partial t}$-term cannot be ignored, this is because $p(\boldsymbol{x}_t)$ is the probability of the random variable being equal to $\boldsymbol{x}_t$ at time t, and $p(\boldsymbol{x}_{t+\Delta t})$ is the probability of the random variable being equal to $\boldsymbol{x}_{t+\Delta t}$ 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 $\log p(\boldsymbol{x}_{t+\Delta t})$ from Eq. (5.5) into Eq. (5.4) and complete the square, we then have,

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

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

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

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

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

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

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

This first occurred in the paper entitled "Reverse-Time Diffusion Equation Model"[9]. 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 $\nabla_{\boldsymbol{x}}\log p_t(\boldsymbol{x})$, then we can rebuild the sample from the following discretised version of the SDE:

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

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

$$p(\boldsymbol{x}_t|\boldsymbol{x}_0) = \lim_{\Delta t\to 0}\int\cdots\iint p(\boldsymbol{x}_t|\boldsymbol{x}_{t-\Delta t})p(\boldsymbol{x}_{t-\Delta t}|\boldsymbol{x}_{t-2\Delta t})\cdots p(\boldsymbol{x}_{\Delta t}|\boldsymbol{x}_0)d\boldsymbol{x}_{t-\Delta t}d\boldsymbol{x}_{t-2\Delta t}d\boldsymbol{x}_{\Delta t}, \tag{5.10}$$

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

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

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

and

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

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

Some readers may be familiar with the following expression:

$$\mathbb{E}[\boldsymbol{x}] = \underset{\boldsymbol{\mu}}{\operatorname{argmin}}\,\mathbb{E}_{\boldsymbol{x}}\left[\|\boldsymbol{\mu} - \boldsymbol{x}\|_2^2\right], \tag{5.13}$$

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

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

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

The denominator $\mathbb{E}_{\boldsymbol{x}_0}[p(\boldsymbol{x}_t|\boldsymbol{x}_0)]$ 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 $\boldsymbol{x}_t$, meaning we must minimise the loss function over every $\boldsymbol{x}_t$, from which we reached the final form of the loss function:

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

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*"[10]. As for the conditional score matching, the author believes that the earliest paper is "*A Connection between Score Matching and Denoising Autoencoder*"[11] 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 difficulities 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 (demolision) process based on the SDE [Eq. (5.1)].
2. Find the expression for $p(\boldsymbol{x}_t|\boldsymbol{x}_0)$.
3. Train the score function $s_{\boldsymbol{\theta}}(\boldsymbol{x}_t, t)$ using the loss function Eq. (5.15) (score-matching).
4. Substitute $s_{\boldsymbol{\theta}}(\boldsymbol{x}_t, t)$ into Eq. (5.8), replacing the term $\nabla_{\boldsymbol{x}_t}\log p(\boldsymbol{x}_t)$ 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 out 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 $p(\boldsymbol{x}_t|\boldsymbol{x}_0)$ from it. In the remaining part of the original paper[8], it focused on deriving the results for two practical examples. However, if solving for $p(\boldsymbol{x}_t|\boldsymbol{x}_0)$ from a pre-defined SDE is too difficult, it will be more trivial to first define $p(\boldsymbol{x}_t|\boldsymbol{x}_0)$, just like the case of DDIM, and then deduce the corresponding SDE.

For example, let us first define

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

Without lost 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:

$$\bar{\alpha}_0 = 1, \quad \bar{\alpha}_t = 0, \quad \bar{\beta}_0 = 0, \quad \bar{\beta}_t = 1. \tag{5.17}$$

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

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

Considering only the linear solution, then we have

$$d\boldsymbol{x} = f_t \boldsymbol{x} dt + g_t d\boldsymbol{w}. \tag{5.19}$$

Similar to Section 4, we have the following:

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

From here, we get the

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

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

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

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

$$\begin{aligned}\bar{\alpha}_t + d\bar{\alpha}_t &= (1 + f_t dt)\bar{\alpha}_t\\ d\bar{\alpha}_t &= f_t dt\bar{\alpha}_t\\ \frac{d\bar{\alpha}_t}{\bar{\alpha}_t} &= f_t dt\end{aligned}$$

As for the loss function, now we have

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

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

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

This is the same loss function as the DDPM after ignoring the coefficients. By replacing $\nabla_{\boldsymbol{x}_{t+\Delta t}} \log p(\boldsymbol{x}_{t+\Delta t})$ with $-\frac{1}{\bar{\beta}_{t+\Delta t}}\boldsymbol{\varepsilon}_{\boldsymbol{\theta}}(\boldsymbol{x}_{t+\Delta t}, t+\Delta t)$, 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

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

In the previous chapter, we have discussed the paper entitled "*Score-Based Generative Modelling through Stochastic Differential Equation* by Dr Yang Song[8]. 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:

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

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

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

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

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

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 $p(\boldsymbol{x}_t|\boldsymbol{x}_0)$, that is independent from the transition probability $p(\boldsymbol{x}_t|\boldsymbol{x}_{t-1})$. Therefore, DDIM starts from $p(\boldsymbol{x}_t|\boldsymbol{x}_0)$ to seek for more general solutions for $p(\boldsymbol{x}_{t-1}|\boldsymbol{x}_t, \boldsymbol{x}_0)$ and $p(\boldsymbol{x}_t|\boldsymbol{x}_{t-1}, \boldsymbol{x}_0)$. The idea of probability flow ODE is similar, which asks the following question:

> In the SDE framework, for a fixed $p(\boldsymbol{x}_t)$, how many different choices of $p(\boldsymbol{x}_{t+\Delta t}|\boldsymbol{x}_t)$ 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:

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

Eq. (6.4) describes the relationship amongst the random variables $\boldsymbol{x}_{t+\Delta t}$, $\boldsymbol{x}_t$ 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 $p(\boldsymbol{y}_t)$ and the relationship that it should satisfy. To achieve this, we can use the Dirac function:

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

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 $f(\boldsymbol{x})$:

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

Taking the derivative of both sides, we have

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

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):

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

in which we have Taylor expanded $\Delta(\boldsymbol{x})$ 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:

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

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

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

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

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 σ_t^2 to the second term and then subtract it out again. Providing that $\sigma_t^2 \le g_t^2$, Eq. (6.10) is fully equivalent to

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

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

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

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

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

However, do not forget that Eq. (6.10) is fully equivalent to Eq. (6.11), this means that the corresponding marginal distributions $p_t(\boldsymbol{x})$ 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 $f_t(\boldsymbol{x})$ is a linear function. Furthermore, according to the SDE results derived in the previous section, the corresponding reverse SDE for Eq. (6.12) is

$$d\boldsymbol{x} = \left(f_t(\boldsymbol{x}) - \frac{1}{2}(g_t^2 + \sigma_t^2)\nabla_{\boldsymbol{x}} \log p_t(\boldsymbol{x}) \right) dt + \sigma_t d\boldsymbol{w}. \tag{6.13}$$

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:

$$d\boldsymbol{x} = \left(f_t(\boldsymbol{x}) - \frac{1}{2} g_t^2 \nabla_{\boldsymbol{x}} \log p_t(\boldsymbol{x}) \right) dt. \tag{6.14}$$

This is the so-called **Probability Flow ODE**. Practically, since $\nabla_{\boldsymbol{x}} \log p_t(\boldsymbol{x})$ needs to be approximated by a neural network $s_{\boldsymbol{\theta}}(\boldsymbol{x}, t)$, 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 $\boldsymbol{x}_0$ to $\boldsymbol{x}_T$. In this case, we can obtain the inverse transformation from $\boldsymbol{x}_T$ to $\boldsymbol{x}_0$ 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[8], 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 $\boldsymbol{x}_T$ to $\boldsymbol{x}_0$ is governed by an ODE, we can apply different higher-order numerical methods for solving the ODE to accelerate the transformation from $\boldsymbol{x}_T$ to $\boldsymbol{x}_0$. 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:

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

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

$$\begin{cases} f_t = \dfrac{1}{\bar{\alpha}_t}\dfrac{\partial \bar{\alpha}_t}{\partial t} \\ g^2(t) = 2\bar{\alpha}_t\bar{\beta}_t\dfrac{\partial}{\partial t}\left(\dfrac{\bar{\beta}_t}{\bar{\alpha}_t}\right) \\ s_{\boldsymbol{\theta}}(\boldsymbol{x},t) = -\dfrac{\boldsymbol{\varepsilon}_{\boldsymbol{\theta}}(\boldsymbol{x},t)}{\bar{\beta}_t}. \end{cases} \tag{6.16}$$

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

$$\frac{1}{\bar{\alpha}_t}\frac{d\boldsymbol{x}}{dt} - \frac{\boldsymbol{x}}{\bar{\alpha}_t^2}\frac{d\bar{\alpha}_t}{dt} = \boldsymbol{\varepsilon}_{\boldsymbol{\theta}}(\boldsymbol{x},t)\frac{d}{dt}\left(\frac{\bar{\beta}_t}{\bar{\alpha}_t}\right), \tag{6.17}$$

in which the L.H.S. can be further simplified into $\frac{d}{dt}\left(\frac{\boldsymbol{x}}{\bar{\alpha}_t}\right)$, 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)

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

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.

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

> *Recall that when $\tilde{p}(\boldsymbol{x}) = \delta(\boldsymbol{x}_0)$, 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 $p(\boldsymbol{x}_{t-1}|\boldsymbol{x}_0)$ which can be computed from $p(\boldsymbol{x}_{t-1}|\boldsymbol{x}_t,\boldsymbol{x}_0)$ when $\tilde{p}(\boldsymbol{x}_0)$ 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 $\boldsymbol{x}_0$):*
>
> $$p(\boldsymbol{x}_{t-1}|\boldsymbol{x}_t,\boldsymbol{x}_0) = \mathcal{N}(\boldsymbol{x}_{t-1};\kappa_t\boldsymbol{x}_t + \lambda_t\boldsymbol{x}_0, \sigma_t^2\boldsymbol{I}),$$
>
> *which we showed that both κ_t and λ_t can be written as a function of σ_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 $\Delta t\to 0$.

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

In the paper entitled "*Improved Denoising Diffusion Probabilitic Models*"[12], 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*"[13] and "*Estimating the Optimal Covariance with Imperfect Mean in Diffusion Probabilistic Models*"[14] 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 $p(\boldsymbol{x}_t|\boldsymbol{x}_0) = \mathcal{N}(\boldsymbol{x}_t; \bar{\alpha}_t \boldsymbol{x}_0, \bar{\beta}_t^2 \boldsymbol{I})$, the general solution for the corresponding conditional transition probability for the reverse process $p(\boldsymbol{x}_{t-1}|\boldsymbol{x}_t, \boldsymbol{x}_0)$ is

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

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

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

However, from the viewpoint of Bayesian statistics, such a treatment is not very appropriate, as the prediction of $\boldsymbol{x}_0$ from $\boldsymbol{x}_t$ 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 $\boldsymbol{x}_t$ to $\boldsymbol{x}_0$..

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

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

$$\begin{aligned}
\boldsymbol{x}_{t-1} &= \frac{\sqrt{\bar{\beta}_{t-1}^2 - \sigma_t^2}}{\bar{\beta}_t}\boldsymbol{x}_t + \gamma_t \boldsymbol{x}_0 + \sigma_t \varepsilon_1 && \text{Directly drawing sample from Eq. (7.1)} \\
&= \frac{\sqrt{\bar{\beta}_{t-1}^2 - \sigma_t^2}}{\bar{\beta}_t}\boldsymbol{x}_t + \gamma_t \left(\bar{\boldsymbol{\mu}}(\boldsymbol{x}_t) + \bar{\sigma}_t \varepsilon_2\right) + \sigma_t \varepsilon_1 && \text{Draw } \boldsymbol{x}_0 \text{ from the approximated normal distribution} \\
&= \left(\frac{\sqrt{\bar{\beta}_{t-1}^2 - \sigma_t^2}}{\bar{\beta}_t}\boldsymbol{x}_t + \gamma_t \bar{\boldsymbol{\mu}}(\boldsymbol{x}_t)\right) + \underbrace{(\sigma_t \varepsilon_1 + \gamma_t \bar{\sigma}_t \varepsilon_2)}_{\sim\sqrt{\sigma_t^2 + \gamma_t^2 \bar{\sigma}_t{}^2}\varepsilon}, && (7.4)
\end{aligned}$$

in which ε_1, ε_2 and $\varepsilon \sim \mathcal{N}(0, \boldsymbol{I})$. By comparing Eq. (7.1), Eq. (7.2) and Eq. (7.4), it can be seen that $p(\boldsymbol{x}_{t-1}|\boldsymbol{x}_t)$ is much closer to a normal distribution with the mean of $\frac{\sqrt{\bar{\beta}_{t-1}^2 - \sigma_t^2}}{\bar{\beta}_t}\boldsymbol{x}_t + \gamma_t \bar{\boldsymbol{\mu}}(\boldsymbol{x}_t)$ and standard deviation of $(\sigma_t^2 + \gamma_t^2 \bar{\sigma}_t{}^2)\boldsymbol{I}$. 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 $\mathcal{N}(\boldsymbol{x}_0; \bar{\boldsymbol{\mu}}(\boldsymbol{x}_t), \bar{\sigma}_t{}^2 \boldsymbol{I})$ to regress $p(\boldsymbol{x}_0|\boldsymbol{x}_t)$. This problem is equivalent of determining the mean and the variance for the probability distribution $p(\boldsymbol{x}_0|\boldsymbol{x}_t)$. For the mean $\bar{\boldsymbol{\mu}}(\boldsymbol{x}_t)$, it depends on $\boldsymbol{x}_t$, so it must be

regressed with a model. To do this, we will need a loss function, which we can start with

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

from which we have

$$\begin{aligned}
\bar{\boldsymbol{\mu}}(\boldsymbol{x}_t) &= \mathbb{E}_{\boldsymbol{x}_0 \sim p(\boldsymbol{x}_0|\boldsymbol{x}_t)}[\boldsymbol{x}_0] && \text{This expression gives the formal definition of for the expectation value of } \boldsymbol{x}_0, \text{ if we can draw sample from the distribution of} p(\boldsymbol{x}_0|\boldsymbol{x}_t). \\
&= \underset{\boldsymbol{\mu}}{\operatorname{argmin}}\, \mathbb{E}_{\boldsymbol{x}_0 \sim p(\boldsymbol{x}_0|\boldsymbol{x}_t)}[\|\boldsymbol{x}_0 - \boldsymbol{\mu}\|_2^2] && \text{Applying Eq. (7.5).} \\
&= \underset{\boldsymbol{\mu}(t)}{\operatorname{argmin}}\, \mathbb{E}_{\boldsymbol{x}_0 \sim p(\boldsymbol{x}_0|\boldsymbol{x}_t), \boldsymbol{x}_t \sim p(\boldsymbol{x}_t)}[\|\boldsymbol{x}_0 - \boldsymbol{\mu}(\boldsymbol{x}_t)\|_2^2] && \text{Now we also need to sample } \boldsymbol{x}_t \text{ as } \boldsymbol{\mu}(\boldsymbol{x}_t) \text{ is a function of it} \\
&= \underset{\boldsymbol{\mu}(t)}{\operatorname{argmin}}\, \mathbb{E}_{\boldsymbol{x}_0 \sim \tilde{p}(\boldsymbol{x}_0), \boldsymbol{x}_t \sim p(\boldsymbol{x}_t|\boldsymbol{x}0)}[\|\boldsymbol{x}_0 - \boldsymbol{\mu}(\boldsymbol{x}_t)\|_2^2]. && (7.6)
\end{aligned}$$

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

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

then, we recover the loss function for training the DDPM model, which is $\|\boldsymbol{\varepsilon} - \boldsymbol{\varepsilon}_{\boldsymbol{\theta}}(\bar{\alpha}_t \boldsymbol{x}_0 + \bar{\beta}_t \boldsymbol{\varepsilon}, t)\|_2^2$. 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

$$\begin{aligned}
\boldsymbol{\Sigma}(\boldsymbol{x}_t) &= \mathbb{E}_{\boldsymbol{x}_0 \sim p(\boldsymbol{x}_0|\boldsymbol{x}_t)}\left[(\boldsymbol{x}_0 - \bar{\boldsymbol{\mu}}(\boldsymbol{x}_t))(\boldsymbol{x}_0 - \bar{\boldsymbol{\mu}}(\boldsymbol{x}_t))^{\mathsf{T}}\right] \\
&= \mathbb{E}_{\boldsymbol{x}_0 \sim p(\boldsymbol{x}_0|\boldsymbol{x}_t)}\left[((\boldsymbol{x}_0 - \boldsymbol{\mu}) - (\bar{\boldsymbol{\mu}}(\boldsymbol{x}_t) - \boldsymbol{\mu}))((\boldsymbol{x}_0 - \boldsymbol{\mu}) - (\bar{\boldsymbol{\mu}}(\boldsymbol{x}_t) - \boldsymbol{\mu}))^{\mathsf{T}}\right] \\
&= \mathbb{E}_{\boldsymbol{x}_0 \sim p(\boldsymbol{x}_0|\boldsymbol{x}_t)}\left[(\boldsymbol{x}_0 - \boldsymbol{\mu}_0)(\boldsymbol{x}_0 - \boldsymbol{\mu}_0)^{\mathsf{T}}\right] - (\bar{\boldsymbol{\mu}}(\boldsymbol{x}_t) - \boldsymbol{\mu}_0)(\bar{\boldsymbol{\mu}}(\boldsymbol{x}_t) - \boldsymbol{\mu}_0)^{\mathsf{T}}, \qquad (7.8)
\end{aligned}$$

in which $\boldsymbol{\mu}_0$ 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 $\mathcal{N}(\boldsymbol{x}_0; \bar{\boldsymbol{\mu}}(\boldsymbol{x}_t), \bar{\sigma_t}^2 \boldsymbol{I})$ to regress $p(\boldsymbol{x}_{t-1}|\boldsymbol{x}_t)$, in which, by design, the covariance matrix is $\bar{\sigma_t}^2 \boldsymbol{I}$. The latter has two characters:

1. It is independent from $\boldsymbol{x}_t$, as such, in order to eliminate its dependency on $\boldsymbol{x}_t$, we can take the average over all $\boldsymbol{x}_t$, such that $\boldsymbol{\Sigma}_t = \mathbb{E}_{\boldsymbol{x}_t \sim p(\boldsymbol{x}_t)}[\boldsymbol{\Sigma}(\boldsymbol{x}_t)]$;
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.* $\bar{\sigma_t}^2 = \operatorname{Tr}(\boldsymbol{\Sigma}_t)/d$, with $d = \dim(\boldsymbol{x})$.

In this case, we have

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

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

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

Here, $\mathrm{Var}[\boldsymbol{x}_0]$ 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:

$$\bar{\sigma_t}^2 \leq \mathrm{Var}[\boldsymbol{x}_0] \leq \left(\frac{b-a}{2}\right). \tag{7.11}$$

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

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

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

$$\begin{aligned}
&\mathbb{E}_{\boldsymbol{x}_t\sim p(\boldsymbol{x}_t)}\mathbb{E}_{\boldsymbol{x}_0\sim p(\boldsymbol{x}_0|\boldsymbol{x}_t)}\left[(\boldsymbol{x}_t - \bar{\alpha}_t\boldsymbol{x}_0)(\boldsymbol{x}_t - \bar{\alpha}_t\boldsymbol{x}_0)^{\mathsf{T}}\right] \\
=\ &\mathbb{E}_{\boldsymbol{x}_0\sim \tilde{p}(\boldsymbol{x}_0)}\mathbb{E}_{\boldsymbol{x}_t\sim p(\boldsymbol{x}_t|\boldsymbol{x}_0)}\left[(\boldsymbol{x}_t - \bar{\alpha}_t\boldsymbol{x}_0)(\boldsymbol{x}_t - \bar{\alpha}_t\boldsymbol{x}_0)^{\mathsf{T}}\right].
\end{aligned} \tag{7.13}$$

Since $p(\boldsymbol{x}_t|x_0) = \mathcal{N}(\boldsymbol{x}_t; \bar{\alpha}_t\boldsymbol{x}_0, \bar{\beta}_t^2\boldsymbol{I})$, so $\bar{\alpha}_t\boldsymbol{x}_0$ is the mean for the distribution $p(\boldsymbol{x}_t|x_0)$, therefore, $\mathbb{E}_{\boldsymbol{x}_t\sim p(\boldsymbol{x}_t|\boldsymbol{x}_0)}\left[(\boldsymbol{x}_t - \bar{\alpha}_t\boldsymbol{x}_0)(\boldsymbol{x}_t - \bar{\alpha}_t\boldsymbol{x}_0)^{\mathsf{T}}\right]$ is the covariance matrix for $p(\boldsymbol{x}_t|x_0)$ which is obviously $\bar{\beta}_t^2\boldsymbol{I}$. Therefore,

$$\mathbb{E}_{\boldsymbol{x}_0\sim \tilde{p}(\boldsymbol{x}_0)}\mathbb{E}_{\boldsymbol{x}_t\sim p(\boldsymbol{x}_t|\boldsymbol{x}_0)}\left[(\boldsymbol{x}_t - \bar{\alpha}_t\boldsymbol{x}_0)(\boldsymbol{x}_t - \bar{\alpha}_t\boldsymbol{x}_0)^{\mathsf{T}}\right] = \mathbb{E}_{\boldsymbol{x}_0\sim \tilde{p}(\boldsymbol{x}_0)}[\bar{\beta}_t^2\boldsymbol{I}] = \bar{\beta}_t^2\boldsymbol{I}. \tag{7.14}$$

Furthermore, we have

$$\boldsymbol{\Sigma}_t = \mathbb{E}_{\boldsymbol{x}\sim p(\boldsymbol{x}_t)}[\boldsymbol{\Sigma}(\boldsymbol{x}_t)] = \frac{\bar{\beta}_t^2}{\bar{\alpha}_t^2}\left(\boldsymbol{I} - \mathbb{E}_{\boldsymbol{x}\sim p(\boldsymbol{x}_t)}\left[\boldsymbol{\varepsilon}_{\boldsymbol{\theta}}(\boldsymbol{x}_t, t)\boldsymbol{\varepsilon}_{\boldsymbol{\theta}}(\boldsymbol{x}_t, t)^{\mathsf{T}}\right]\right). \tag{7.15}$$

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

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

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 Lammas 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 $\nabla_{\boldsymbol{x}_t}p(\boldsymbol{x}_t)$ 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 $\nabla_{\boldsymbol{x}_t}p(\boldsymbol{x}_t)$ back to either $\bar{\boldsymbol{\mu}}(\boldsymbol{x}_t)$ or $\boldsymbol{\varepsilon}_{\boldsymbol{\theta}}(\boldsymbol{x}_t, t)$ 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 $\bar{\boldsymbol{\mu}}(\boldsymbol{x}_t)$ or $\boldsymbol{\varepsilon}_{\boldsymbol{\theta}}(\boldsymbol{x}_t, t)$ directly in the derivation, leads to more obvious mathematical form, and much straightforward for numerical implementation.

8. Optimum Estimation of the Variance (Part II)

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

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*"[14] 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

$$\sigma_t^2 + \gamma_t^2 \bar{\sigma}_t^2, \tag{8.1}$$

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

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

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

$$\bar{\boldsymbol{\mu}}(\boldsymbol{x}_t) = \frac{1}{\bar{\alpha}_t}\left(\boldsymbol{x}_t - \bar{\beta}_t \boldsymbol{\varepsilon}_{\boldsymbol{\theta}}(\boldsymbol{x}_t, t)\right) \tag{8.3}$$

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

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

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

$$\boldsymbol{\Sigma}_t = \mathbb{E}_{\boldsymbol{x}\sim p(\boldsymbol{x}_t)}[\boldsymbol{\Sigma}(\boldsymbol{x}_t)] = \frac{\bar{\beta}_t^2}{\bar{\alpha}_t^2}\left(\boldsymbol{I} - \mathbb{E}_{\boldsymbol{x}\sim p(\boldsymbol{x}_t)}\left[\boldsymbol{\varepsilon}_{\boldsymbol{\theta}}(\boldsymbol{x}_t, t)\boldsymbol{\varepsilon}_{\boldsymbol{\theta}}(\boldsymbol{x}_t, t)^{\top}\right]\right). \tag{8.5}$$

If we then tool the trace of the diagonal element and average it over the matrix dimension, *i.e.* $\bar{\sigma}_t^2 = \mathrm{Tr}(\boldsymbol{\Sigma}_t)/d$, 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 $p(\boldsymbol{x}_0|\boldsymbol{x}_t)$ has a covariance matrix of $\sigma_t^2\boldsymbol{I}$, 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. $\mathrm{diag}(\bar{\boldsymbol{\sigma}}_t^2)$. Here, we restrict the vector multiplications are all being carried out with the Hamamard product, for example, $\boldsymbol{x}^2 = \boldsymbol{x}\otimes\boldsymbol{x}$. In this case, we only need to consider the diagonal part of $\boldsymbol{\Sigma}_t$, and from Eq. (8.5), the corresponding estimate for the

variance is

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

in which $\mathbf{1}_d$ is a d-dimensional unit vector. A further improvement is to retain the dependency of $\bar{\boldsymbol{\sigma}}_t^2$ on $\boldsymbol{x}_t$, *i.e.* we consider $\bar{\boldsymbol{\sigma}}_t^2(\boldsymbol{x}_t)$, making it similar to $\bar{\boldsymbol{\mu}}(\boldsymbol{x}_t)$, which now $\bar{\boldsymbol{\sigma}}_t^2(\boldsymbol{x}_t)$ also becomes a learnable model on the input $\boldsymbol{x}_t$.

On the other hand, can we consider using the full matrix of $\boldsymbol{\Sigma}_t$? This is workable only in principle but not very practical. This is because $\boldsymbol{\Sigma}_t$ 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 $\bar{\boldsymbol{\mu}}(\boldsymbol{x}_t)$ is learnt from the model, it may not be able to accurately represent $\mathbb{E}_{\boldsymbol{x}\sim p(\boldsymbol{x}_0|\boldsymbol{x}_t)}[\boldsymbol{x}_0]$. 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 $\bar{\boldsymbol{\mu}}(\boldsymbol{x}_t)$ has been trained, then, in the distribution $p(\boldsymbol{x}_0|\boldsymbol{x}_t) \sim \mathcal{N}(\boldsymbol{x}_0; \bar{\boldsymbol{\mu}}(\boldsymbol{x}_t), \bar{\sigma}_t^2\boldsymbol{I})$, the only remaining parameter will be $\bar{\sigma}_t^2$. The corresponding negative log-likelihood is

The key idea here is, since we know $x0$, and thus $\tilde{p}(\boldsymbol{x}_0)$ very well, we should seek an estimate of $\bar{\sigma}_t$ directly from sampling $\boldsymbol{x}_0$ rather than sample from $\boldsymbol{x}_t$ and map it back to $\boldsymbol{x}_0$.

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

for which the minimum is reached when

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

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

Here, $\bar{\boldsymbol{\mu}}(\boldsymbol{x}_t)$ 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:

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

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

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

8.4. Conditional variance

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

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

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

$$\mathbb{E}_{\boldsymbol{x}}[\boldsymbol{x}] = \operatorname*{argmin}_{\boldsymbol{\mu}}\mathbb{E}_{\boldsymbol{x}}[\|\boldsymbol{x} - \boldsymbol{\mu}\|_2^2], \tag{8.12}$$

we get the following

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

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

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*"[12] 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 $\bar{\boldsymbol{\mu}}(\boldsymbol{x}_t)$. 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 $\bar{\boldsymbol{\mu}}(\boldsymbol{x}_t)$.

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 $\bar{\boldsymbol{\mu}}(\boldsymbol{x}_t)$, 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. Acheiving a more stable training process.

9. Conditional Generation

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

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*"[15], 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"[16], the concept of the classifier has been further generalised, 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*"[17]. 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 $p(\boldsymbol{x}_{t-1}|\boldsymbol{x}_t)$. As such, for sample generation based on some condition $\boldsymbol{y}$, all we need to do is introduce $\boldsymbol{y}$ into the generation process via the conditional probability $p(\boldsymbol{x}_{t-1}|\boldsymbol{x}_t, \boldsymbol{y})$ which replaces $p(\boldsymbol{x}_{t-1}|\boldsymbol{x}_t)$. In order to reuse the $p(\boldsymbol{x}_{t-1}|\boldsymbol{x}_t)$. from the trained unconditional generative model, we apply the Baye's rule

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

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

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

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

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

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 $p(\boldsymbol{x}_t|\boldsymbol{x}_{t-1})$ will be sufficiently small. This is saying that the transition probability $p(\boldsymbol{x}_t|\boldsymbol{x}_{t-1})$ will only be significantly larger than zero when $\boldsymbol{x}_t$ is very close to $\boldsymbol{x}_{t-1}$. This is also true for the conditional probabilities, *i.e.* either $p(\boldsymbol{x}_{t-1}|\boldsymbol{x}_t, \boldsymbol{y})$ or $p(\boldsymbol{x}_t|\boldsymbol{x}_{t-1}, \boldsymbol{y})$ will only be significantly larger than zero when $\boldsymbol{x}_t$ is very close to $\boldsymbol{x}_{t-1}$. As such, we only need to consider the change in the probability distributions within this small range of $\boldsymbol{x}$. With this, we can apply the Taylor expansion as

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

Strictly speaking, there should also exist an extra t-dependent term, but it is irrelevant to $\boldsymbol{x}_{t-1}$, as it is a constant that will not change the probability of $\boldsymbol{x}_{t-1}$. Therefore, this term

is ignored in Eq. (9.4). Assuming that we already have $p(\boldsymbol{x}_{t-1}|\boldsymbol{x}_t) = \mathcal{N}(\boldsymbol{x}_{t-1}; \boldsymbol{\mu}(\boldsymbol{x}_t), \sigma_t^2 \boldsymbol{I}) \propto \exp\left[-\|\boldsymbol{x}_{t-1} - \boldsymbol{\mu}(\boldsymbol{x}_t)\|^2 / 2\sigma_t^2\right]$, then we have the following approximation:

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

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

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

This is the core result in the Classifier-Guidance Approach. Note that $\boldsymbol{x}_t$ is the sample after noises are being added, which is the input for $p(\boldsymbol{y}|\boldsymbol{x}_t)$. 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 $p_0(\boldsymbol{y}|\boldsymbol{x}_0)$, then what we way consider is

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

i.e., we use $\boldsymbol{\mu}(\cdot)$ to denoise $\boldsymbol{x}_t$ before plugging it into $p(\boldsymbol{y}|\boldsymbol{x}_t)$. 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 γ was introduced into the gradient of the classifier, one can better attenuate the generated samples. In this case, we have:

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

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 $\boldsymbol{y}$, 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 γ could be understood by adjusting the form of the probability distribution through the use of a power operation:

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

As γ increases, $\tilde{p}(\boldsymbol{y}|\boldsymbol{x}_t)$ will slowly merge into an one-hot distribution. If this distribution was used to substitute $p(\boldsymbol{y}|\boldsymbol{x}_0)$ 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

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

The authors of the original paper had wrongly took $\mathcal{Z}(\boldsymbol{x}_t)$ as a constant, which gave $\nabla_{\boldsymbol{x}_t} \mathcal{Z}(\boldsymbol{x}_t) = 0$. However, when $\gamma \neq 1$, it is obvious that $\mathcal{Z}(\boldsymbol{x}_t)$ will also be dependent on $\boldsymbol{x}_t$. 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 $p(\boldsymbol{x}_{t-1}|\boldsymbol{x}_t, \boldsymbol{y})$. Instead, we could straightforwardly define

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

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

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

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

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

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

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

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

which now we no longer need to worry about the probabilistic interpretation of $p(\boldsymbol{y}|\boldsymbol{x}_t)$, but only a suitable definition of the similarity measure $\text{sim}(\boldsymbol{x}_t, \boldsymbol{y})$. Here, $\boldsymbol{y}$ 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:

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

It should be noted that, since the intermediate state $\boldsymbol{x}_t$ 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 $\sigma_t^2 \gamma \nabla_{\boldsymbol{x}_t} \log p(\boldsymbol{y}|\boldsymbol{x}_t)$ or $\gamma\sigma_t^2 \nabla_{\boldsymbol{x}_t}\text{sim}(\boldsymbol{x}_t, \boldsymbol{y})$, which both contain the term σ_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 σ_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:

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

with the corresponding reverse SDE

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

Here, we are allowed to choose σ_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 $\nabla_{\boldsymbol{x}_t} \log p_t(\boldsymbol{x})$. If now we want to perform conditional generation, we should just need to change it to $\nabla_{\boldsymbol{x}} \log p_t(\boldsymbol{x}|\boldsymbol{y})$. From Bayes' rule, we have

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

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

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

This means that, regardless of what the variance σ_t is, all we need to do is to replace $\varepsilon_{\boldsymbol{\theta}}(\boldsymbol{x}_t, t)$ with $\varepsilon_{\boldsymbol{\theta}}(\boldsymbol{x}_t, t) - \bar{\beta}_t \nabla_{\boldsymbol{x}} \log p_t(\boldsymbol{y}|\boldsymbol{x})$, 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

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

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

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

with the corresponding loss function

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

The advantage here is that the additional input $\boldsymbol{y}$ 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 γ-parameter to balance specificity and diversity. Practically, we rewrite Eq. (9.8) using the following modified mean:

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

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

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

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

10. Unified Diffusion Model: The Theoretical Framework

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

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

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 $p(\boldsymbol{x}_t|\boldsymbol{x}_{t-1})$. 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 $p(\boldsymbol{x}_t|\boldsymbol{x}_{t-1})$, but the probability $p(\boldsymbol{x}_t|\boldsymbol{x}_0)$. It was also found that, to derive $p(\boldsymbol{x}_t|\boldsymbol{x}_0)$ from $p(\boldsymbol{x}_t|\boldsymbol{x}_{t-1})$ was also rather difficult. Therefore, **a more practical approach is to directly treat $p(\boldsymbol{x}_t|\boldsymbol{x}_0)$ as the forward process.**

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

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

in which $\mathcal{F}$ is a function of t, $\boldsymbol{x}_0$ 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 $\boldsymbol{x}_0$ to $\boldsymbol{x}_t$, without any restriction on the data type**. Generally, the only restriction is imposed on t, such that the smaller the time variable t is, $\mathcal{F}_t(\boldsymbol{x}_0, \boldsymbol{\varepsilon})$ will contain more information about $\boldsymbol{x}_0$, and it becomes easier to reconstruct $\boldsymbol{x}_0$ from $\mathcal{F}_t(\boldsymbol{x}_0, \boldsymbol{\varepsilon})$. Conversely, it becomes more difficult to reconstruct $\boldsymbol{x}_0$ at large t. At a certain upper limit T, $\mathcal{F}_T(\boldsymbol{x}_0, \boldsymbol{\varepsilon})$ will contain almost no information about $\boldsymbol{x}_0$, from which it becomes literally impossible to reconstruct $\boldsymbol{x}_0$.

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 $p(\boldsymbol{x}_{t-1}|\boldsymbol{x}_t)$. In general, we have the following:

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

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

If $\boldsymbol{x}_0$ 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 $p(\boldsymbol{x}_{t-1}|\boldsymbol{x}_t)$ to be easy to achieve numerically, we will also require the sampling from both $p(\boldsymbol{x}_{t-1}|\boldsymbol{x}_t, \boldsymbol{x}_0)$ and $p(\boldsymbol{x}_0|\boldsymbol{x}_t)$ to be easy. In this case, the sampling from the distribution $p(\boldsymbol{x}_{t-1}|\boldsymbol{x}_t)$ can be achieved via the following procedure:

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

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

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

Hence, reverse sampling is an iterative process of estimation followed by correction. By iteratively incorporating the estimate of $\boldsymbol{x}_t \to \boldsymbol{x}_0$ into the reverse process, we obtain the corrected regression sequence of $\boldsymbol{x}_T \to \cdots \boldsymbol{x}_t \to \boldsymbol{x}_{t-1} \to \cdots \to \boldsymbol{x}_0$. 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 $p(\boldsymbol{x}_{t-1}|\boldsymbol{x}_t, \boldsymbol{x}_0)$ or $p(\boldsymbol{x}_0|\boldsymbol{x}_t)$. In this subsection, we will discuss the choice of $p(\boldsymbol{x}_0|\boldsymbol{x}_t)$. Obviously, $p(\boldsymbol{x}_0|\boldsymbol{x}_t)$ is the probabilistic model that allows one

to predict $\boldsymbol{x}_0$ from $\boldsymbol{x}_t$, and it must be a distribution that is easy to be calculated and sampled from. When $\boldsymbol{x}_0$ is continuous data, there are not many choices of suitable distributions, and the most common choice is a normal distribution with a trainable mean:

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

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 $\boldsymbol{x}_0$ 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 $q(\boldsymbol{x}_0|\boldsymbol{x}_t)$, the training target is easy to establish. A natural choice is to use the cross-entropy

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

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

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

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 $p(\boldsymbol{x}_0|\boldsymbol{x}_t)$ that is what we try to learn and the true distribution $q(\boldsymbol{x}_t|\boldsymbol{x}0)$ from training data generated from forward process.

10.4. Conditional probability

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

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

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

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

which means now we have made $p(\boldsymbol{x}_{t-1}|\boldsymbol{x}_t,\boldsymbol{x}_0)$ become completely independent from $\boldsymbol{x}_t$. 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 $\boldsymbol{x}_{t-1}$ now becomes completely dependent upon $\boldsymbol{x}_0$, which represents the original sample. In the reverse process, however, we can only sample $\boldsymbol{x}_0$ approximately from the approximated distribution of $q(\boldsymbol{x}_0|\boldsymbol{x}_t)$. Since $q(\boldsymbol{x}_0|\boldsymbol{x}_t)$ is usually not very accurate, errors in the generative process will continuously accumulate throughout. At the same time, the distribution $p(\boldsymbol{x}_{t-1}|\boldsymbol{x}_0)$ will carry noises in the sampling process, which may significantly affect the estimated $\hat{\boldsymbol{x}}_0$, thus worsening the generative outcomes.

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

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

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

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

which is equivalent to

$$p(\boldsymbol{x}_{t-1}|\boldsymbol{x}_t,\boldsymbol{x}_0) = \delta\left(\boldsymbol{x}_{t-1} - \mathcal{F}_{t-1}(\boldsymbol{x}_0, \mathcal{F}_t^{-1}(\boldsymbol{x}_0,\boldsymbol{x}_t))\right). \tag{10.11}$$

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

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

on the superposition of normal distributions, we get

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

In this case, the solution of ε from $\boldsymbol{x}_0$ and $\boldsymbol{x}_t$ can only be used to substitute one of ε_1 and ε_2. The final sampling process follows:

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

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. Foremostly, 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 $\boldsymbol{x}_0$. Finally, ε 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 $p(\boldsymbol{x}_{t-1}|\boldsymbol{x}_t)$ 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 $\boldsymbol{x}_0 \to \boldsymbol{x}_t$, then it is natural to match it with the reverse process with $\boldsymbol{x}_t \to \boldsymbol{x}_0$. Eq. (10.2) contains two parts, in which $p(\boldsymbol{x}_0|\boldsymbol{x}_t)$ is very clear, it represents the probability of predicting $\boldsymbol{x}_0$ from $\boldsymbol{x}_t$, which does not have much room for further simplification and can only be acquired from a learnable model. The freedom comes from $p(\boldsymbol{x}_{t-1}|\boldsymbol{x}_t, \boldsymbol{x}_0)$, 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 $q(\varepsilon)$ and a transformation $\mathcal{F}$:

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

Then, through the following partition

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

we will achieve the sampling $\boldsymbol{x}_{t-1} \sim p(\boldsymbol{x}_{t-1}|\boldsymbol{x}_t)$ in the reverse process. Here $p(\boldsymbol{x}_0|\boldsymbol{x}_t)$ is the probability of estimating $\boldsymbol{x}_0$ from $\boldsymbol{x}_t$, which we can generally approximate it with a

simple distribution such as $q(\boldsymbol{x}_0|\boldsymbol{x}_t)$. In this case, the corresponding training target will be $-\log q(\boldsymbol{x}_0|\boldsymbol{x}_t)$ or some simple variants of it. When $\boldsymbol{x}_0$ is of the continuous type, we can choose the conditional normal distribution for $q(\boldsymbol{x}_0|\boldsymbol{x}_t)$. When $\boldsymbol{x}_0$ are discrete data, we could approximate $q(\boldsymbol{x}_0|\boldsymbol{x}_t)$ with either an autoregressive or a non-autoregressive model.

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

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

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

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

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

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

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:

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

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

and subsequently we choose $\mathcal{N}(\boldsymbol{x}_0;\bar{\boldsymbol{\mu}}(\boldsymbol{x}_t),\bar{\sigma}_t\boldsymbol{I})$ for $q(\boldsymbol{x}_0|\boldsymbol{x}_t)$. 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:

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

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

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

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

$$\begin{aligned}\boldsymbol{x}_{t-1} &= \bar{\alpha}_{t-1}\boldsymbol{x}_0 + \bar{\beta}_{t-1}\varepsilon \\ &\sim \bar{\alpha}_{t-1}\boldsymbol{x}_0 + \sqrt{\bar{\beta}_{t-1}^2 - \bar{\sigma}_t^2}\varepsilon_1 + \sigma_t\varepsilon_2,\end{aligned} \tag{11.9}$$

in which ε, ε_1 and $\varepsilon_2\sim\mathcal{N}(0,\boldsymbol{I})$. From $\boldsymbol{x}_t = \bar{\alpha}_t\boldsymbol{x}_0 + \bar{\beta}_t\varepsilon$, we have $\varepsilon = (\boldsymbol{x}_t - \bar{\alpha}_t\boldsymbol{x}_0)/\bar{\beta}_t$, which can be used for substituting ε_1. This leads to the following general from of $p(\boldsymbol{x}_{t-1}|\boldsymbol{x}_t,\boldsymbol{x}_0)$:

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

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

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

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 $\sigma_t = \bar{\beta}_{t-1}\beta_t/\bar{\beta}_t$, DDIM has chosen

$\bar{\sigma}_t = 0$, $\sigma_t = 0$, whereas in Analytic-DP$_\dagger$. the value of $\bar{\sigma}_t$ had been further optimised.

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*"[18], 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 $\boldsymbol{x}_t = \mathcal{F}(\boldsymbol{x}_0)$. To facilitate the subsequent discussions, let us generalise this forward process further by introducing the noise term as

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

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 $\sigma \to 0$. Subsequently, we choose $q(\boldsymbol{x}_0|\boldsymbol{x}_t)$ as a normal distribution with ℓ_1-norm:

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

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

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

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

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

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

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

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 let 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 ℓ. It is possible also to deal with variable length but it is slightly more difficult. With this, we define the forward process $\boldsymbol{x}_t = \mathcal{F}(\boldsymbol{x}_0, \varepsilon)$ as 'random substitution', *i.e.*, we randomly select t tokens in the sentence and substitute them with other tokens. When $t = \ell$, $\boldsymbol{x}_t$ is a combination of ℓ random tokens. In

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

1. Randomly choose ℓ tokens as the initial $\boldsymbol{x}_\ell$.
2. Make an estimate of $\hat{\boldsymbol{x}}_0$ from $q(\boldsymbol{x}_0|\boldsymbol{x}_t)$.
3. Randomly choose $\ell - 1$ tokens to substitute other tokens to form $\boldsymbol{x}_{t-1}$.
4. Repeat steps 2 and 3 until a final $\boldsymbol{x}_0$ 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 $\mathcal{F}(\boldsymbol{x}_0, \varepsilon)$ to be an invertible function of ε. This means that from a give pair of $\boldsymbol{x}_0$ and $\boldsymbol{x}_t$, 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 $\boldsymbol{x}_0$ and $\boldsymbol{x}_t$ to figure out what the transformation was. In this case, we change step 3 into a transformation from $\hat{\boldsymbol{x}}_0$ to $\boldsymbol{x}_t$:

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

With this sampling procedure, in every sampling step, the part in $\hat{\boldsymbol{x}}_0$ that is identical to $\boldsymbol{x}_t$ can be kept. Comparing $\boldsymbol{x}_{t-1}$ and $\boldsymbol{x}_t$, 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 ℓ tokens, and define the forward process $\boldsymbol{x}_t = \mathcal{F}(\boldsymbol{x}_0, \varepsilon)$ as random masking, *i.e.*, we select t tokens and randomly replace them with [MASK], in which $t \leq \ell$. When $t = \ell$, $\boldsymbol{x}_t$ is a sentence with ℓ [MASK].

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

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

1. Using a sequence containing ℓ [MASK] as the starting $\boldsymbol{x}_\ell$.
2. Sample $\hat{\boldsymbol{x}}_0$ from $q(\boldsymbol{x}_0|\boldsymbol{x}_t)$, 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 $\boldsymbol{x}_t$ in which the t positions were occupied by [MASK]. Changing those into the corresponding token at $\hat{\boldsymbol{x}}_0$ and make this to be $\boldsymbol{x}_{t-1}$.
4. Repeat steps 2 and 3 until a final $\boldsymbol{x}_0$ 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 $\boldsymbol{x}_t$, and according to the probability $q(\boldsymbol{x}_0|\boldsymbol{x}_t)$, sample a token and replace the [MASK] at that position, giving a new $\boldsymbol{x}_{t-1}$.

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

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

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

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

The only difference is that now $\mathcal{F}$ is also parameterised. The reverse sampling process is also similar, except now the last step $\hat{\boldsymbol{x}}_0 \sim q(\boldsymbol{x}_0|\boldsymbol{x}_1)$ will return $\hat{\boldsymbol{x}}_0$ directly. Most importantly, because now we have an encoding model $\mathcal{F}(\boldsymbol{x}_0)$, the input data $\boldsymbol{x}_0$ 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

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

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 mode 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

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 $\boldsymbol{x}_t \in \mathbb{R}^d$:

$$\frac{d\boldsymbol{x}_t}{dt} = f_t(\boldsymbol{x}_t). \tag{12.1}$$

Suppose $t \in [0, T]$, then with a given $\boldsymbol{x}_0$, (under some conditions that are easy to realise) we could solve for $\boldsymbol{x}_T$ deterministically. It implies that this ODE describes a definitive transformation from $\boldsymbol{x}_0$ to $\boldsymbol{x}_T$. More specifically, this transformation will be invertible, meaning that we could also solve the ODE in the reverse order to obtain the transformation from $\boldsymbol{x}_T$ to $\boldsymbol{x}_0$. 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 $\boldsymbol{x}_0$ as a sample from the input data, and $\boldsymbol{x}_T$ as a sample drawn from a simple probability distribution. Our goal here is to obtain a transformation from $\boldsymbol{x}_0$ to $\boldsymbol{x}_T$ through the differential equation.

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

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

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

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

Here, $\frac{\partial \boldsymbol{x}_{t+\Delta t}}{\partial \boldsymbol{x}_t}$ 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

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

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

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

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:

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

12.3. Taylor approximation

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

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

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

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

In other words, any arbitrary function $f_t(\boldsymbol{x}_t)$ 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

$$\frac{\partial}{\partial t}\log p_t(\boldsymbol{x}_t) = -\nabla_{\boldsymbol{x}_t}\left(f_t(\boldsymbol{x}_t)p_t(\boldsymbol{x}_t)\right), \tag{12.9}$$

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:

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

in which $D_t(\boldsymbol{x}_t)$ 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 $D_t(\boldsymbol{x}_t)$ 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 $\boldsymbol{x}_0$ to $\boldsymbol{x}_T$ is a process that progressively searches the regions of low probabilities, and conversely, the transformation from $\boldsymbol{x}_T$ to $\boldsymbol{x}_0$ 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

$$\frac{\partial}{\partial t}p_t(\boldsymbol{x}_t) = \nabla_{\boldsymbol{x}_t}\cdot\Big(D_t(\boldsymbol{x}_t)\nabla_{\boldsymbol{x}_t}p_t(\boldsymbol{x}_t)\Big). \tag{12.11}$$

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

$$\frac{\partial}{\partial t}p_t(\boldsymbol{x}_t) = D_t\nabla^2_{\boldsymbol{x}_t}p_t(\boldsymbol{x}_t). \tag{12.12}$$

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 $p_t(\boldsymbol{x}_t)$, and the result is

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

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 $p_0(\boldsymbol{x}_0)$ 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 no 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 $\boldsymbol{x}_t$ on both sides of the equation and use the fact that $\nabla_{\boldsymbol{x}_t} \to i\omega$, the result is:

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

This is an ordinary differential equation with the following solution:

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

in which $\sigma_t^2 = 2\int_0^t D_s ds$, and $\mathcal{F}_0(\omega)$ is the Fourier transformation of $p_0(\boldsymbol{x}_0)$. Now we take the inverse Fourier transformation on both size, for which $\mathcal{F}_t(\omega)$ transforms back to $p_t(\boldsymbol{x}_t)$, $\mathcal{F}_0(\omega)$ transforms back to $p_0(\boldsymbol{x}_0)$ and $\exp(-\sigma_t^2\omega^2/2)$ corresponds to the normal distribution $\mathcal{N}(\boldsymbol{x}_t; \boldsymbol{x}_0, \sigma_t^2 \boldsymbol{I})$. 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

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

which corresponds to the solution for the following equation

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

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

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

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

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

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

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

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

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

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 $f_t(\boldsymbol{x}_t)$ 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.

13. From the Law of Gravitational Force to Diffusion Model

Original blog post see https://www.spaces.ac.cn/archives/9305

For many readers, the diffusion generative model is probably the first deep-learning model that they have encountered, which has utilised so many different mathematical methods. In this series of articles, we have demonstrated the strong connection between the diffusion models and many different fields in mathematics, including mathematical analysis, probability theory,

statistics, ordinary and stochastic partial differential equations. It is reasonable to say, even for researchers in the field of pure theory, such as mathematical physics, it will be very likely for them to find their skills useful in the development of diffusion generative models.

In this section, we shall introduce another diffusion model that has a strong link with mathematical physics, that is the ODE-based diffusion model that was inspired by the law of the gravitational force. It cam from the paper entitled "**Poisson Flow Generative Model**"[19] (PFGM for short), which has provided a new perspective for constructing an ODE-based diffusion model.

13.1. The gravitational force

We have all learned about the law of gravitational force in high school. Basically, it states that, the attractive force between two point masses, is proportional to the product of their masses and the inverse square of the distance between them. Here, we ignore the masses and the constant of the gravitational field, but focus on the relationship between the gravitational force with distance and direction. Assuming the source of this attractive force is at $\boldsymbol{y}$, then the force experienced by an object at $\boldsymbol{x}$ is given by:

$$\boldsymbol{F}(\boldsymbol{x}) = -\frac{1}{4\pi}\frac{\boldsymbol{x}-\boldsymbol{y}}{\|\boldsymbol{x}-\boldsymbol{y}\|_2^3}. \tag{13.1}$$

We can further ignore the factor of $1/4\pi$ for the time being, as it will not affect our subsequent discussions. Strictly speaking, the above equation describes the gravitational field in the three-dimensional space. For the d-dimensional case, the corresponding gravitational field is

$$\boldsymbol{F}(\boldsymbol{x}) = -\frac{1}{S_d(1)}\frac{\boldsymbol{x}-\boldsymbol{y}}{\|\boldsymbol{x}-\boldsymbol{y}\|_2^d}, \tag{13.2}$$

in which $S_d(1)$ is the surface area of a d-dimensional hypersphere. This expression corresponds to the gradient of the Green's function for the d-dimensional Poisson distribution, thus the word "Poisson" in the title of the paper mentioned above.

13.2. Following the lines of the gravitational field

If there exist multiple sources of attractions, we just need to add up the attractive forces from different sources, based on the additivity of the forces. Generally, we observe that, for most of the field lines, they start from infinitely far away, and end at the attractors. Now, an intuitive but smart idea arises. If every source of the gravitational force represents a real sample to be generated, then for an arbitrary far away point, it will evolve into a real sample if it moves by following the field line. This is the key genius thought behind the article "Poisson Flow generative Model".

13.3. Equivalent centre-of-mass

Nevertheless, to convert a smart idea into a workable model, there are still many details that will need to be filled in. For instance, when we say "an arbitrarily far away point", which corresponds to the prior distribution in the diffusion model, how far is really far enough, and how should we sample an arbitrary point? Apparently, if the sampling procedure is too complicated, then it becomes valueless.

Fortunately, there exists an equivalence property for the gravitational field: *The gravitational field experienced by a point that is infinitive far away from multiple sources of attraction, is equivalent to the field generated from a single centre-of-mass with the sum of all masses.*

This is saying that, when the distance is sufficiently large, we can treat the field as originating from a single point source of attraction. What is the property of the gravitational field from a single point mass? It is isotropic in space! This means that when the radius is sufficiently large, we can consider the gravitational field lines will pass the surface of the sphere centred at the point mass evenly. Therefore, for the procedure of initial sampling, it should be performed on the surface of a hypersphere with a sufficiently large radius, and we shall discuss later how large is sufficiently large.

13.4. Mode Collapsing

Has this completed the generative model? Not yet! The isotropic nature of the gravitational field has made the initial sampling easy. However, it can also cause the gravity sources cancelling each other, leading to the so-called phenomenon of "mode-collapsing". More specifically, we consider the field distribution around a spherical shell, in which it can be observed

that, outside the shell, the gravitational field distributes isotropically. However, inside the shell, the field is empty! It shows that the fields cancel each other, which essentially forms a vacuum inside the shell.

the cancellation of the field implies that, because the gravitational forces form uniformly distributed sources on a spherical shell cancel out each other, it is equivalent to the case where these sources did not exist in the first place. On the other hand, the core idea of the generative model discussed here, is to let a point far away to move along the gravitational field until it reaches a particular source. If, the gravitational field cancels each other, it means that we may never reach certain sources of the gravitational field, or equivalently, we may never be able to synthesise some of the realistic samples. In this case, we lost the diversity in the generative outcomes. This is the phenomenon of mode collapse.

13.5. Adding one extra dimension

It seems like the problem of mode collapsing is unavoidable anyhow. This is because, when we are building the generative model, we usually assume that the real samples follow a continuous distribution. In this case, we can always choose a sphere, even though the samples do not distribute uniformly on its surface, we can always find a subset from this distribution that is uniformly distributed, such that the forces from this subset cancel out each other. This is equivalent to saying this subset of samples being non-existent, leading to the problem of mode collapse.

Does this mean we are stuck? No, as this brings to the second genius idea behind PFGM, that is, to add an one extra dimension.

From our previous analysis, the problem of mode collapse become unavoidable, is because the continuous sample distribution which we assumed in the first place has led to the situation in which the isotropy in the field distribution becomes unavoidable. In order to avoid the mode collapse, we need to try avoiding isotropy in the sample distribution. However, the sample distribution is our target distribution, which we cannot change. Nevertheless, we can add one extra dimension to the samples. If now we tackle the problem in the $(d+1)$-dimensional space, we can no longer have isotropy on a place. Using a low dimensional analogous example, we know that a circle in the two-dimensional space is isotropic. However, in the three-dimensional space, only a sphere is isotropic, a circle that is isotropic in the two-dimensional space is no longer isotropic in the three-dimensional space.

In this regard, let us assume that the real samples that we would like to generate are $\boldsymbol{x} \in \mathbb{R}^d$, and we can introduce another new dimension t, such that the sample data now change into the form of $(\boldsymbol{x}, t) \in \mathbb{R}^{d+1}$ Correspondingly, the original sample follows a distribution of $\boldsymbol{x} \sim \tilde{p}(\boldsymbol{x})$. Now the distribution becomes $(\boldsymbol{x}, t) \sim \tilde{p}(\boldsymbol{x})\delta(t)$, in which $\delta(t)$ is the Dirac distribution. This is a transformation that places the real sample onto the $t = 0$ plane in the $(d+1)$-dimensional space. With this, in the $(d+1)$-dimensional space, all the realistic samples will always have $t = 0$, which cannot be isotropically distributed any more.

13.6. Everything clicked

From the first loo of it, adding one extra dimension is only a small mathematical trick. However, as one thinks more about it, we could find more brilliant points behind this approach. Many problems that are difficult to be solved in the d-dimensional space have suddenly become easy when we moved to the $(d+1)$-dimension.

According to Eq. (13.2) and the linear superposition of the gravitational fields, the corresponding field in the $(d+1)$-dimensional space is

$$\begin{aligned}
\boldsymbol{F}(\boldsymbol{x}, t) &= -\frac{1}{S_{d+1}(1)} \iint \frac{(\boldsymbol{x} - \boldsymbol{x}_0, t - t_0)}{(\|\boldsymbol{x} - \boldsymbol{x}_0\|^2 + (t - t_0)^2)^{(d+1)/2}} \delta(t_0)\tilde{p}(\boldsymbol{x}_0) d\boldsymbol{x}_0 dt_0 \\
&= -\frac{1}{S_{d+1}(1)} \iint \frac{(\boldsymbol{x} - \boldsymbol{x}_0, t)}{(\|\boldsymbol{x} - \boldsymbol{x}_0\|^2 + t^2)^{(d+1)/2}} \delta(t_0)\tilde{p}(\boldsymbol{x}_0) d\boldsymbol{x}_0 \qquad (13.3) \\
&\triangleq (\boldsymbol{F}_{\boldsymbol{x}}, \boldsymbol{F}_t),
\end{aligned}$$

in which $\boldsymbol{F}_{\boldsymbol{x}}$ is the first d components of $\boldsymbol{F}(\boldsymbol{x}, t)$ and $\boldsymbol{F}_t$ is its $(d+1)$-th component. We will discuss later on how to learn $\boldsymbol{F}(\boldsymbol{x}, t)$. Assuming that we have already know the form of $\boldsymbol{F}(\boldsymbol{x}, t)$, then we need to move along the field line, meaning that the trajectories of our sample movement should always follow the direction of $\boldsymbol{F}(\boldsymbol{x}, t)$, *i.e.*,

$$(d\boldsymbol{x}, dt) = (\boldsymbol{F}_{\boldsymbol{x}}, \boldsymbol{F}_t) d\tau \quad \Rightarrow \quad \frac{d\boldsymbol{x}}{dt} = \frac{\boldsymbol{F}_{\boldsymbol{x}}}{\boldsymbol{F}_t}. \qquad (13.4)$$

This is the differential equation that we will need for the generative model. Before, in the d-dimensional space, apart from the problem of mode collapse, when to stop the movement is also a problem. Intuitively, we just need to follow the field line and stop when we have collided with the sample. But how do we judge whether we have collided with a real sample? This is not a trivial question to answer. However, in the $(d+1)$-dimensional space,all the real samples lie on the plane of $t = 0$, hence we should stop when $t = 0$ is reached.

As for the prior distribution, based on the discussions above, it should be a uniform distribution on a $(d+1)$-dimensional hypersphere with a sufficiently large radius. However, since we use $t = 0$ as our stop signal, then we could fix a sufficiently large τ, in the order of 40 to 100, and then sample on the $t = T$ plane. In this case, the generative process becomes solving the equation of motion from $t = T$ to $t = 0$ as given by the differential equation. This gives well-defined starting and end points for the generative process.

If we perform sampling on the $t = T$ place, it will not be uniform. In fact, we have the following:

$$p_{prior}(\boldsymbol{x}) \propto \frac{1}{(\|\boldsymbol{x}\|^2 + T^2)^{(d+1)/2}}. \tag{13.5}$$

It can be seen that the probability density is only dependent on the modulus $\|\boldsymbol{x}\|$. Hence, for sampling, we need to first sample the modulus from a given distribution, then we perform uniform sampling for different directions and combine the two together. For sampling the modulus, let $r = \|\boldsymbol{x}\|$ and converts to hyperspherical harmonics, then we have $p_{prior}(r) \propto r^{d-1}(r^2 + \tau^2)^{(d+1)/2}$, from which we could sample from this distribution with the inverse cumulative distribution function.

13.7. Training the field

Now, we have the prior distribution and the differential equation, all we have left is to develop a protocol to train the function of a vector field $F(\boldsymbol{x}, t)$. Based on the differential equation (13.4), it can be seen that the movement of the particle is only dependent on the relative value of the vector field. As such, the results will not be affected if we shrink or expand the vector field. Based on Eq. (13.3), we can write the vector field as an expectation value:

$$\boldsymbol{F}(\boldsymbol{x}, t) = \mathbb{E}_{\boldsymbol{x}_0 \sim \tilde{p}(\boldsymbol{x}_0)} \left[-\frac{(\boldsymbol{x} - \boldsymbol{x}_0, t)}{(\|\boldsymbol{x} - \boldsymbol{x}_0\|^2 + t^2)^{(d+1)/2}} \right]. \tag{13.6}$$

Using the following equality that we had applied many times:

$$\mathbb{E}_{\boldsymbol{x}} = \underset{\boldsymbol{\mu}}{\operatorname{argmin}} \, \mathbb{E}_{\boldsymbol{x}}[\|\boldsymbol{x} - \boldsymbol{\mu}\|^2], \tag{13.7}$$

we can introduce a trainable function $s_\theta(\boldsymbol{x}, t)$ to learn $\boldsymbol{F}(\boldsymbol{x}, t)$ with the following target function:

$$\mathbb{E}_{\boldsymbol{x}_0 \sim \tilde{p}(\boldsymbol{x}_0)} \left[\left\| s_\theta(\boldsymbol{x}, t) + \frac{(\boldsymbol{x} - \boldsymbol{x}_0, t)}{(\|\boldsymbol{x} - \boldsymbol{x}_0\|^2 + t^2)^{(d+1)/2}} \right\|^2 \right]. \tag{13.8}$$

However, we still need to perform sampling on $\boldsymbol{x}$ and t in order to evaluate the above target function, and there is no clear definition on how this should be done. This is a major character behind PFGM, it defined the reverse generative process, without the need of defining the forward process. This one-step sampling is practically equivalent to the forward process. As such, the original paper considered an approach to construct samples of $\boldsymbol{x}$ and t through perturbing each real sample as:

$$\boldsymbol{x} = \boldsymbol{x}_0 + \|\varepsilon_{\boldsymbol{x}}\|(1+\tau)^m \boldsymbol{u}, \qquad t = \|\varepsilon_t\|(1+\tau)^m, \tag{13.9}$$

in which $(\varepsilon_{\boldsymbol{x}}, \varepsilon_t) \sim \mathcal{N}(0, \sigma^2 \boldsymbol{I}_{(d+1)\times(d+1)})$ and $m \sim \mathcal{U}[0, M]$. $\boldsymbol{u}$ is the unit vector that distribute uniformly on the d-dimensional hypersphere. τ, σ and M are all constants.

Finally, the training target in the original PFGM paper is slightly different from Eq. (13.8), it is approximately equivalent to

$$\left\| s_\theta(\boldsymbol{x}, t) + \text{Normalise}\left(\mathbb{E}_{\boldsymbol{x}_0 \sim \tilde{p}(\boldsymbol{x}_0)} \left[\frac{(\boldsymbol{x} - \boldsymbol{x}_0, t)}{(\|\boldsymbol{x} - \boldsymbol{x}_0\|^2 + t^2)^{(d+1)/2}} \right] \right) \right\|^2 \tag{13.10}$$

In the real implementation, we can only sample a finite number of $\boldsymbol{x}_0$ evaluate the expectation value inside the bracket. As such, this target function contains some bias. Although a biased target function is not necessarily worse than an unbiased one. Why such a target function was chosen in the original paper was unclear. Our best guess is because the biased estimation has introduced the normalisation operations in the vectors, which has made the training process numerically more stable. But this will also increase the batch size to achieve better accuracy, thus the training cost.

Overall, in this section, we have introduced an ODE-based diffusion model that is inspired by the law of the gravitational forces. It has overcome the reliance on the Gaussian distributions on the diffusion models discussed before, and is a brand new theoretical framework that builds ODE-based diffusion model from the field theory, which is worth to be studied in details.

14. The General Procedure of Constructing an ODE (Part I)

Original blog post see https://www.spaces.ac.cn/archives/9370

Continuing from our discussions in the previous section, in which we have introduced an ODE-based diffusion generative model that was inspired by the gravitational field with a clear geometric meaning. Some readers may ask, it seems like the gravitational field may not be the only choice, is it possible to use other type of forces to construct the diffusion model under the same physical picture? On the other hand, even though the model is physically very trivial, there is a lack of mathematical proof that we could indeed learn the real sample distribution from it.

Here, we aim to provide a mathematically more accurate answer on what type of force field is suitable for constructing an ODE-based diffusion generative model.

14.1. The basic conclusions from the previous sections

To answer this question, we need to use a conclusion that we have derived in Section 12, which finds the corresponding ODE for describing the change in the probability distributions from the starting ODE that describes the evolution of the samples.

Consider the first-order ODE on $\boldsymbol{x}_t \in \mathbb{R}^d$ and $t \in [0, T]$:

$$\frac{d\boldsymbol{x}_t}{dt} = f_t(\boldsymbol{x}_t), \tag{14.1}$$

which describes an (invertible) transformation from $\boldsymbol{x}_0$ to $\boldsymbol{x}_T$. If $\boldsymbol{x}_0$ was a random variable, then $\boldsymbol{x}_t$ will also be a random variable throughout the transformation. The corresponding changes in the probability distribution can be described by the following ODE:

$$\frac{d}{dt}p_t(\boldsymbol{x}_t) = -\nabla_{\boldsymbol{x}_t}\Big(f_t(\boldsymbol{x}_t)p_t(\boldsymbol{x}_t)\Big). \tag{14.2}$$

This equation could be derived by using the Jacobian and Taylor expansion as shown in Section 12, or as we have done in Section 6, we derived the full Fokker-Plank equation first, and then set $g_t = 0$. It should be pointed out that, Eq. (14.2) is the famous **continuity equation** in physics, which reflects many different conservation laws.

Returning to the diffusion model, our goal here, is to construct a transformation, that is capable of transforming a simple distribution into the target sample distribution. In principle, we could apply Eq. (14.2) to solve for a possible $f_t(\boldsymbol{x}_t)$ with a given distribution $p_t(\boldsymbol{x}_t)$, then using Eq. (14.1) to complete the generation process. However, noticing that we only have one equation [Eq. (14.2)] here but the target function $f_t(\boldsymbol{x}_t)$ has d components, so this is an indeterminate equation. In principle, we can solve the equation by assigning an arbitrary complete set of $p_t(\boldsymbol{x}_t)$ and solve for $f_t(\boldsymbol{x}_t)$, not just the distributions at the boundaries corresponding to $t = 0$ and $t = T$.

As such, the construction of an ODE-based diffusion model is simply to solve an ODE with almost no constraints. Although this is conceptually true, it is particularly difficult to achieve in coding. Therefore, a more accurate way to rephrase the question is how can we solve for some practically more useful solutions from Eq. (14.2).

14.2. Simplifying the equation

Noticing that, Eq. (14.2) can be rewritten in the following form:

$$\underbrace{\left(\frac{\partial}{\partial t}, \nabla_{\boldsymbol{x}_t}\right)}_{\nabla_{(t,\boldsymbol{x}_t)}} \cdot \underbrace{\Big(p_t(\boldsymbol{x}_t), f_t(\boldsymbol{x}_t)p_t(\boldsymbol{x}_t)\Big)}_{\boldsymbol{u}\in\mathbb{R}^{d+1}} = 0. \tag{14.3}$$

As shown above, we can treat $\left(\frac{\partial}{\partial t}, \nabla_{\boldsymbol{x}_t}\right)$ as the gradient $\nabla_{(t,\boldsymbol{x}_t)}$ in the $(d+1)$-dimensional space, and $\Big(p_t(\boldsymbol{x}_t), f_t(\boldsymbol{x}_t)p_t(\boldsymbol{x}_t)\Big)$ can be written as a $(d+1)$-dimensional vector, such that Eq. (14.2) can be casted into a simple divergence equation:

$$\nabla_{(t,\boldsymbol{x}_t)}\boldsymbol{u}(t,\boldsymbol{x}_t) = 0. \tag{14.4}$$

Under this form, we have

$$\frac{d\boldsymbol{x}_t}{dt} = f_t(\boldsymbol{x}_t) = \frac{\boldsymbol{u}_{>1}(t,\boldsymbol{x}_t)}{\boldsymbol{u}_1(t,\boldsymbol{x}_t)}, \tag{14.5}$$

in which $\boldsymbol{u}_1$ and $\boldsymbol{u}_{>1}$ represent the first and last d components of $\boldsymbol{u}(t,\boldsymbol{x}_t)$. of course, we cannot forget the boundary conditions:

$$\begin{cases} \boldsymbol{u}_1(0,\boldsymbol{x}_0) = p_0(\boldsymbol{x}_0) & \text{(initial condition)} \\ \int \boldsymbol{u}_1(t,\boldsymbol{x}_t)dt = 1 & \text{(integral condition)} \end{cases} \tag{14.6}$$

in which $p_0(\boldsymbol{x}_0)$ is the target sample distribution. For the distribution at $t = T$, we only require it to be a distribution that is easy to sample from, with no other constraint to be imposed at this stage.

14.3. The Green's function

With the above transformation, we can treat $\boldsymbol{u}(t,\boldsymbol{x}_t)$ as a vector field in the $(d+1)$-dimensional space, and Eq. (14.5) determines the trajectory of a point mass that is moving along the field direction. this then becomes consistent with the physical picture established in the last section based on the gravitational forces.

In order to solve for a general solution for $\boldsymbol{u}(t,\boldsymbol{x}_t)$, we can apply the mathematical tool of the Green's function. Firstly, let us try to solve the following problem

$$\begin{cases} \nabla_{(t,\boldsymbol{x}_t)} \cdot G(t,0;\boldsymbol{x}_t,\boldsymbol{x}_0) = 0 \\ G_1(0,0;,\boldsymbol{x}_t,\boldsymbol{x}_0) = \delta(\boldsymbol{x}_t - \boldsymbol{x}_0) \\ \int G_1(t,0;\boldsymbol{x}_t,\boldsymbol{x}_0)d\boldsymbol{x}_t = 1. \end{cases} \tag{14.7}$$

It is not difficult to prove that, if the above equation holds, then we have

$$\boldsymbol{u}(t,\boldsymbol{x}_t) = \int G(t,0;\boldsymbol{x}_t,\boldsymbol{x}_0)p_0(\boldsymbol{x}_0)d\boldsymbol{x}_0 = \mathbb{E}_{\boldsymbol{x}_0\sim p_0(\boldsymbol{x}_0)}[G(t,0;\boldsymbol{x}_t,\boldsymbol{x}_0)], \tag{14.8}$$

which is the solution to Eq. (14.4) that satisfies the corresponding constraints. In this case, we have successfully expressed $u(t,\boldsymbol{x}_t)$ as some expectation value over the training samples, which will help the model training. It is not difficult to see that the Greeen's function $G(t,0;\boldsymbol{x}_t,\boldsymbol{x}_0)$ is indeed the conditional probability $p(\boldsymbol{x}_t|\boldsymbol{x}_0)$ in the diffusion model.

As a matter of face, the Green's function $G(t,0;\boldsymbol{x}_t,\boldsymbol{x}_0)$ that is defined in Eq. (14.7) is not the green's function that is usually defined. In the traditional definition, the Green's function corresponds to the solution at the point source of the field, whereas here, we have put the source at the boundary points. Even so, the Green's function $G(t,0;\boldsymbol{x}_t,\boldsymbol{x}_0)$ defined here still possesses the properties of a normal Green's function, which is equivalent to a forcefield that is sourced from a point and is continuously distributed in the space.

14.4. The gravitational force

Now we will seek for some concrete solutions based on the above established mathematical framework. As we have mentioned before, both Eq. (14.4) and Eq. (14.7) are undetermined

equations that give one equation to solve for $d+1$ unknowns. Theoretically, there exist infinitely many solutions of different forms. Therefore, to actually solve these equations, we must introduce some additional assumptions in order to arrive at a more concrete solution. The first assumption is the isotropic assumption, which corresponds to the results given in Section 13.

14.4.1. An ansatz

We note that the isotropic property is referring to the state $(t, \boldsymbol{x}_t)$ lying in the $(d+1)$-dimensional space being isotropic. It means that $G(t, 0; \boldsymbol{x}_t, \boldsymbol{x}_0)$ points to the origin at $(0, \boldsymbol{x}_0)$, and the modulus is only depending on $R = \sqrt{t^2 + \|\boldsymbol{x}_t - \boldsymbol{x}_0\|^2}$. As such, we can let

$$G(t, 0; \boldsymbol{x}_t, \boldsymbol{x}_0) = \varphi(R)(t, \boldsymbol{x}_t - \boldsymbol{x}_0). \tag{14.9}$$

From which we have

The first boundary condition as in Eq. (14.7).

$$\begin{aligned}
0 &= \nabla_{(t,\boldsymbol{x}_t)} \cdot G(t, 0; \boldsymbol{x}_t, \boldsymbol{x}_0) \\
&= \left[\nabla_{(t,\boldsymbol{x}_t)} \varphi(R)\right] \cdot (t, \boldsymbol{x}_t - \boldsymbol{x}_0) + \varphi(R) \left[\nabla_{(t,\boldsymbol{x}_t)} (t, \boldsymbol{x}_t - \boldsymbol{x}_0)\right] \quad \text{(product rule)} \\
&= \varphi'(R) \cdot \frac{(t, \boldsymbol{x}_t - \boldsymbol{x}_0)}{R} \cdot (t, \boldsymbol{x}_t - \boldsymbol{x}_0) + (d+1)\varphi(R) \quad \text{(chain rule)} \\
&= \varphi'(R) R + (d+1)\varphi(R) \quad \text{(definition of R)} \\
&= \frac{[\varphi(R) R^{d+1}]'}{R^d}. \quad \text{(reversing the product rule)}
\end{aligned} \tag{14.10}$$

This implies that we must have $[\varphi(R) R^{d+1}]' = 0$ or $\varphi(R) R^{d+1} = C$, *i.e.* $\varphi(R) = C R^{-(d+1)}$. As such, an ansatz for Eq. (14.7) is

$$G(t, 0; \boldsymbol{x}_t, \boldsymbol{x}_0) = C \times \frac{(t, \boldsymbol{x}_t - \boldsymbol{x}_0)}{(t^2 + \|\boldsymbol{x}_t - \boldsymbol{x}_0\|)^{(d+1)/2}}. \tag{14.11}$$

14.4.2. Imposing the constraints

it can be seen from the above derivation, that under the assumption of isotropy, the gravitational field becomes the only possible ansatz. To further prove that this is a feasible solution, we must also check that it satisfies the other constraints, in which the key one is

$$\int G_1(t, 0; \boldsymbol{x}_t, \boldsymbol{x}_0) d\boldsymbol{x}_t = C \times \int \frac{t}{(t^2 + \|\boldsymbol{x}_t - \boldsymbol{x}_0\|)^{(d+1)/2}} d\boldsymbol{x}_t = 1. \tag{14.12}$$

To prove this, we only need to show the integral is independent from $\boldsymbol{x}_0$ and t, and provided that a suitable normalisation constant C is chosen, we can always make the value of the integral to be unity. For any $t > 0$, we can make the substitution of $\boldsymbol{z} = (\boldsymbol{x}_t - \boldsymbol{x}_0)/t$. Since $\boldsymbol{x}_t \in \mathbb{R}^d$, we also have $\boldsymbol{z} \in \mathbb{R}^d$. With this, Eq. (14.12) becomes

$$\int G_1(t, 0; \boldsymbol{x}_t, \boldsymbol{x}_0) d\boldsymbol{x}_t = C \times \int \frac{1}{(1 + \|\boldsymbol{z}\|)^{(d+1)/2}} d\boldsymbol{z}, \tag{14.13}$$

which we can see now the integral is completely independent from $\boldsymbol{x}_0$ and t, so we just need to choose a suitable normalisation constant C to make the integral to become unity. In the following discussions, we assume that such C has already been chosen.

As for the initial values, we need to prove the limit $\lim_{t \to 0^+} G_1(t, 0; \boldsymbol{x}_t, \boldsymbol{x}_0) = \delta(\boldsymbol{x}_t - \boldsymbol{x}_0)$, which we only need to show this based on the definition of the Dirac function: (1) When $\boldsymbol{x}_t \neq \boldsymbol{x}_0$, the limit is obviously zero. (2) When $\boldsymbol{x}_t = \boldsymbol{x}_0$, the limit is obviously 1. (3) As we have already shown, the integral of $G_1(t, 0; \boldsymbol{x}_t, \boldsymbol{x}_0)$ over $\boldsymbol{x}_t$ is always unity. These three points are the basic properties, or simply put, the definition of the Dirac function, so we have make sure the function has the correct initial values.

14.4.3. Result analysis

According to Eq. (14.8), we now have

$$\boldsymbol{u}(t, \boldsymbol{x}_t) = C \times \mathbb{E}_{\boldsymbol{x}_0 \sim p(\boldsymbol{x}_0)} \left[\frac{(t, \boldsymbol{x}_t - \boldsymbol{x}_0)}{(t^2 + \|\boldsymbol{x}_t - \boldsymbol{x}_0\|^2)^{(d+1)/2}} \right], \tag{14.14}$$

which we can further apply the relationship $\mathbb{E}_{\boldsymbol{x}}[\boldsymbol{x}] = \underset{\boldsymbol{\mu}}{\operatorname{argmin}}\mathbb{E}_{\boldsymbol{x}}[\|\boldsymbol{x}-\boldsymbol{\mu}\|^2]$ to construct a training target that is similar to score-matching. which will not be repeated here.

It has been mentioned previously, that $G_1(t,0;\boldsymbol{x}_t,\boldsymbol{x}_0)$ is equivalent to $p_t(\boldsymbol{x}_t|\boldsymbol{x}_0)$, which we have now known the exact form

$$p_t(\boldsymbol{x}_t|\boldsymbol{x}_0) \propto \frac{t}{(t^2+\|\boldsymbol{x}_t-\boldsymbol{x}_0\|^2)^{(d+1)/2}}, \tag{14.15}$$

when T is sufficiently large, the influence of $\boldsymbol{x}_0$ upon $p_t(\boldsymbol{x}_t|\boldsymbol{x}_0)$ becomes negligible, in which case $p_t(\boldsymbol{x}_t|\boldsymbol{x}_0)$ becomes the prior distribution that is independent from $\boldsymbol{x}_0$:

$$p_{prior}(\boldsymbol{x}_T) \propto \frac{T}{(T^2+\|\boldsymbol{x}_T\|^2)^{(d+1)/2}}. \tag{14.16}$$

This is the same prior distribution as derived in Section 13 but it comes more naturally under the current framework. Since now we have $p_t(\boldsymbol{x}_t|\boldsymbol{x}_0)$, in principle, we can also achieve the sampling of $\boldsymbol{x}_t \sim p(\boldsymbol{x}_t|\boldsymbol{x}_0)$. From Eq. (14.13), we know that if we make the substitution of $\boldsymbol{z} = (\boldsymbol{x}_t - \boldsymbol{x}_0)/t$, then we have

$$p(\boldsymbol{z}) \propto \frac{1}{(1+\|\boldsymbol{z}\|^2)^{(d+1)/2}}, \tag{14.17}$$

From which we can first sample from $p(\boldsymbol{z})$ and then obtain $\boldsymbol{x}_t$ from $\boldsymbol{x}_t = \boldsymbol{x}_0 + \boldsymbol{z}t$. As for the actual sampling of $p(\boldsymbol{z})$, since it only depends on the value of the modulus, we could first sample the modulus using the inverse cumulative function and then randomly sample an orientation to finish the entire sampling process, which is the same as sampling from the prior distribution. Nevertheless, when considering the remaining problems, we have discovered the following surprise!

14.4.4. Revisiting the problem

In Section 13, we have mentioned that the sampling procedure provided in the original paper was of the following form,

$$\boldsymbol{x}_t = \boldsymbol{x}_0 + \|\boldsymbol{\varepsilon}_{\boldsymbol{x}}\|(1+\tau)^m\boldsymbol{u}, \qquad t = \|\varepsilon_t\|(1+\tau)^m, \tag{14.18}$$

in which $(\boldsymbol{\varepsilon}_{\boldsymbol{x}},\varepsilon_t) \sim \mathcal{N}(0,\sigma^2\boldsymbol{I}_{(d+1)\times(d+1)})$ and $m \sim \mathcal{U}[0,M]$. $\boldsymbol{u}$ is the unit vector that distribute uniformly on the d-dimensional hypersphere. τ, σ and M are all constants.

Initially, it was thought that such a sampling procedure was purely a subjective design from the original authors, with not much reasoning. Subsequently, we found a surprising coincident, that this sampling procedure is a way to achieve the sampling from the distribution given in Eq. (14.17). We shall prove this in the following discussions. First of all, we substitute the second half of Eq. (14.18) into the first half, which we get:

$$\boldsymbol{x}_t = \boldsymbol{x}_0 + t\frac{\|\boldsymbol{\varepsilon}_{\boldsymbol{x}}\|}{|\varepsilon_t|}\cdot\boldsymbol{u}, \tag{14.19}$$

that is of exactly the same form of $\boldsymbol{x}_t = \boldsymbol{x}_0 + t\boldsymbol{z}$ derived above, and $\boldsymbol{u}$ is an isotropic random variable. In this case, the question becomes whether $\frac{\|\boldsymbol{\varepsilon}_{\boldsymbol{x}}\|}{|\varepsilon_t|}$ follows the same probability distribution as $\|\boldsymbol{\varepsilon}\|$. The answer is yes. Paying attention to the face that when converting the probability distribution from the Cartesian to spherical coordinates, we need to multiply an extra term that is proportional to the radius to the power of $d-1$, as such, according to Eq. (14.17), we have

$$p(\|\boldsymbol{z}\|) \propto \frac{\|\boldsymbol{z}\|^{d-1}}{(1+\|\boldsymbol{z}\|^2)^{(d+1)/2}}. \tag{14.20}$$

On the other hand, according to the choice of $(\boldsymbol{\varepsilon}_{\boldsymbol{x}},\varepsilon_t) \sim \mathcal{N}(0,\boldsymbol{I}_{(d+1)\times(d+1)})$ (Here, because we are only interested in the ratio between two probability distributions, the standard deviations cancel out each other, thus, for simplicity, we can let $\sigma = 1$), so we have the following

$$p(\|\boldsymbol{\varepsilon}_{\boldsymbol{x}}\|) \propto \|\boldsymbol{\varepsilon}_{\boldsymbol{x}}\|^{d-1}e^{-\|\boldsymbol{\varepsilon}_{\boldsymbol{x}}\|^2/2}, \qquad p(|\varepsilon_t|) \propto e^{-|\varepsilon_t|^2/2}. \tag{14.21}$$

Let $r = \|\boldsymbol{\varepsilon}_{\boldsymbol{x}}\|/|\varepsilon_t|$, such that $\|\boldsymbol{\varepsilon}_{\boldsymbol{x}}\| = r|\varepsilon_t|$, then take the usage of equality in probability, we have

$$\begin{aligned}
p(r)dr &= \mathbb{E}_{|\varepsilon_t|\sim p(|\varepsilon_t|)}\left[p(\|\boldsymbol{\varepsilon}_{\boldsymbol{x}}\| = r|\varepsilon_t|)d(r|\varepsilon_t|)\right] \\
&\propto \mathbb{E}_{|\varepsilon_t|\sim p(|\varepsilon_t|)}\left[r^{d-1}|\varepsilon_t|^d e^{-r^2|\varepsilon_t|^2/2}dr\right] \\
&\propto \int_0^\infty r^{d-1}|\varepsilon_t|^d e^{-r^2|\varepsilon_t|^2/2}e^{-|\varepsilon_t|^2/2}d|\varepsilon_t|dr \\
&\propto \int_0^\infty r^{d-1}|\varepsilon_t|^d e^{-(r^2+1)|\varepsilon_t|^2/2}d|\varepsilon_t|dr \\
&= \frac{r^{d-1}}{(1+r^2)^{(d+1)/2}}\int_0^\infty s^d e^{-s^2/2}dsdr \qquad (s = |\varepsilon_t|\sqrt{r^2+1}) \\
&\propto \frac{r^{d-1}}{(1+r^2)^{(d+1)/2}}
\end{aligned} \tag{14.22}$$

This shows that $p(r)$ is fully equivalent to Eq. (14.20). As such, using $(\|\boldsymbol{\varepsilon}_{\boldsymbol{x}}\|/|\varepsilon_t|)\boldsymbol{u}$ has indeed provided an effective sampling strategy for $\boldsymbol{z}$, which is numerically much easier than using the inverse cumulative function. But this was not explicitly mentioned in the original paper.

14.5. Space-time separation

What we have done so far is to solve for $(t, \boldsymbol{x}_t)$ under the assumption of isotropy is the $(d+1)$-dimensional space. To some extent, this counts for the simplest possible solution, although some readers might find it difficult to accept, as this gravitational field-based generative model is seemingly more complicated mathematically. However, when solving the mathematical physics equations, it is very common that we to try solving the problem first by starting from a simple isotropic ansatz.

Nevertheless, it is indeed difficult to rationalise the isotropic nature of $(t, \boldsymbol{x}_t)$ when taking the space-time variables together as a whole. We are more comfortable in dealing with isotropy in space by separating out the time variable. Here, we shall seek for the solution under the separation of time and space.

14.5.1. An ansatz

What the above discussions imply here, are that the isotropy should now apply only to the spatial variable $\boldsymbol{x}_t$ in the d-dimensional space, and we will separate $G(t, 0; \boldsymbol{x}_t, \boldsymbol{x}_0)$ into two parts $G_1(t, 0; \boldsymbol{x}_t, \boldsymbol{x}_0)$ and $G_{>1}(t, 0; \boldsymbol{x}_t, \boldsymbol{x}_0)$ to help with our understanding. Here, $G_1(t, 0; \boldsymbol{x}_t, \boldsymbol{x}_0)$ is only a scalar, in which case, isotropy implies that it only depends on the modulus $r = \|\boldsymbol{x}_t - \boldsymbol{x}_0\|$, which we shall denote it as $\phi_t(r)$. Whereas $G_{>1}(t, 0; \boldsymbol{x}_t, \boldsymbol{x}_0)$ is a d-dimensional vector, and isotropy implies that it points to $\boldsymbol{x}_0$ from all directions, and the modulus is only dependent upon $r = \|\boldsymbol{x}_t - \boldsymbol{x}_0\|$. As such, we can let

$$G_{>1}(t, 0; \boldsymbol{x}_t, \boldsymbol{x}_0) = \varphi_t(r)(\boldsymbol{x}_t - \boldsymbol{x}_0). \tag{14.23}$$

With this, we have [from Eq. (14.4)]:

$$\begin{aligned}
0 &= \frac{\partial}{\partial t}\phi_t(r) + \nabla_{\boldsymbol{x}_t}\Big(\varphi_t(r)(\boldsymbol{x}_t - \boldsymbol{x}_0)\Big) \\
&= \frac{\partial}{\partial t}\phi_t(r) + r\frac{\partial}{\partial r}\varphi_t(r) + d\varphi_t(r) \\
&= \frac{\partial}{\partial t}\phi_t(r) + \frac{1}{r^{d-1}}\frac{\partial}{\partial r}\Big(\varphi_t(r)r^d\Big).
\end{aligned} \tag{14.24}$$

Here, we have two undetermined functions $\phi_t(r)$ and $\varphi_t(r)$ but only one differential equation, which makes it easier to solve. Since the conditional constraint is applied to $G_1(t, 0; \boldsymbol{x}_t, \boldsymbol{x}_0)$, that is, to the function $\phi_t(r)$ not $\varphi_t(r)$, in this case, we could, for simplicity, solve for $\varphi_t(r)$ with a given function of $\phi_t(r)$ that satisfies the constraints. With this, the solution for Eq. (14.24) is

$$\varphi_t(r) = -\frac{1}{r^d}\int\frac{\partial}{\partial t}\phi_t(r)r^{d-1}dr = -\frac{1}{r^d}\frac{\partial}{\partial t}\int\phi_t(r)r^{d-1}dr. \tag{14.25}$$

14.5.2. Gaussian diffusion

Here, we shall further prove that, the common ODE-based diffusion model that builds upon the Gaussian distribution is a special case for Eq. (14.25). For the assumption that is based on the Gaussian distribution, we have the following transition probability:

$$G_1(t,0;\boldsymbol{x}_t,\boldsymbol{x}_0)=p_t(\boldsymbol{x}_t|\boldsymbol{x}_0)=\frac{1}{(2\pi\sigma_t^2)^{d/2}}e^{-\|\boldsymbol{x}_t-\boldsymbol{x}_0\|^2/2\sigma_t^2}, \tag{14.26}$$

i.e. $\phi_t(r)=\frac{1}{(2\pi\sigma_t^2)^{d/2}}e^{-r^2/2\sigma_t^2}$, in which σ_t is a monotonically increasing function of the time variable t. It satisfies the boundary condition that $\sigma_0=0$ and σ_T being sufficiently large. $\sigma_0=0$ is set to satisfy the initial value condition, and σ_T is to be made sufficiently large in order to make the prior distribution becomes independent from the sample distribution. As for the unity integral constraint, this is naturally satisfied by the Gaussian distribution.

Substituting Eq. (14.26) into Eq. (14.25) we have the following solution

$$\varphi_t(r)=\frac{\dot{\sigma}_t}{(2\pi\sigma_t^2)^{d/2}\sigma_t}e^{-r^2/2\sigma_t^2}=\frac{\dot{\sigma}_t}{\sigma_t}\phi_t(r), \tag{14.27}$$

in which the integration over r has taken the usage of the incomplete Gamma function that the author has derived it with the help of Mathematica. With this solution, we have

$$\begin{aligned}
\boldsymbol{u}_1(t,0;\boldsymbol{x}_t,\boldsymbol{x}_0)&=\int p_t(\boldsymbol{x}_t|\boldsymbol{x}_0)p(\boldsymbol{x}_0)d\boldsymbol{x}_0=p_t(\boldsymbol{x}_t)\qquad\text{[from Eq. (14.26)]}\\
\boldsymbol{u}_{>1}(t,0;\boldsymbol{x}_t,\boldsymbol{x}_0)&=\int\frac{\dot{\sigma}_t}{\sigma_t}(\boldsymbol{x}_t-\boldsymbol{x}_0)p_t(\boldsymbol{x}_t|\boldsymbol{x}_0)p_0(\boldsymbol{x}_0)d\boldsymbol{x}_0\\
&=-\dot{\sigma}_t\sigma_t\int\nabla_{\boldsymbol{x}_t}p_t(\boldsymbol{x}_t|\boldsymbol{x}_0)p_0(\boldsymbol{x}_0)d\boldsymbol{x}_0\\
&=-\dot{\sigma}_t\sigma_t\nabla_{\boldsymbol{x}_t}p_t(\boldsymbol{x}_t),
\end{aligned} \tag{14.28}$$

in which, based on Eq. (14.5), we have

$$f_t(\boldsymbol{x}_t)=\frac{\boldsymbol{u}_{>1}(t,\boldsymbol{x}_t)}{\boldsymbol{u}_1(t,\boldsymbol{x}_t)}=-\dot{\sigma}_t\sigma_t\nabla_{\boldsymbol{x}_t}\log p_t(\boldsymbol{x}_t). \tag{14.29}$$

This result is identical to those derived in Section 12

14.6. Reverse engineering

Solving for $\varphi_t(r)$ with a given $\phi_t(r)$ is theoretically simple, but there are two major challenges: (1) It is difficult to construct a function $\phi_t(r)$ that is simultaneously satisfying both the initial value and the integral conditions, and (2) One cannot guarantee that the integration over r will always have a simple analytical form. As such, we need to come up with a method of reverse engineering to construct $\phi_t(r)$.

We know that $\phi_t(r)$ is the probability density under the Cartesian coordinates, which, upon transformation into the spherical harmonics, one must multiply it with $C_d r^{d-1}$, in which C_d is a d-dependent constant. Based on Eq. (14.5) that the final result is a ratio that will not be affected by this constant, so we can ignore it for simplicity. This leads straight to the integrand in Eq. (14.25). As such the integral in Eq. (14.25)

$$\int\phi_t(r)r^{d-1}dr \tag{14.30}$$

is a cumulative probability function. It is not always easy to calculate the cumulative probability function through integration, but going the other way around via differentiation is very easy. As such, we could build the cumulative probability function, and then solve for the corresponding $\phi_t(r)$ and $\varphi_t(r)$, bypassing the challenge of integration.

To build the cumulative probability function $\psi_t(r)$, it must satisfy the following conditions: (1) $\psi_t(0)=0$ and $\psi_t(\infty)=1$. (2) $\psi(r)$ is a monotonically increasing function of r. (3) $\forall r>0,\ \lim_{t\to 0^+}\psi_t(r)=1$. For people who are familiar with the activation function, it should not be difficult to construct such functions, it is basically the smoothed approximation to the step function such as $\tanh(r/t)$ and $1-e^{-r/t}$. With the cumulative function $\psi_t(r)$, and based

on Eq. (14.25), we have

$$\phi_t(r) = \frac{1}{r^{d-1}}\frac{\partial}{\partial r}\psi_t(r), \qquad \varphi_t(r) = -\frac{1}{r^d}\frac{\partial}{\partial t}\Big(\phi_t(r) + \lambda_t\Big), \tag{14.31}$$

in which λ_t is an arbitrary function of t, which can be set as zero in the usual situation. Certainly, all of the isotropic solutions are fundamentally equivalent to each other, including the results derived for the gravitational diffusions. They all could be included in the equations above. We can also derive different outcomes using coordinate transformation, as the cumulative function $\psi_t(r)$ is only a function of a scalar variable, and we can transform the cumulative probability function for different distributions, providing they are all well-behaved monotonically increasing functions.

14.7. Summary

To summarise, here, we have constructed a general framework of building ODE-based diffusion model. Theoretically speaking, the framework encompassed all ODE-based diffusion models, from which we can construct any type of new and funky diffusion models. For examples, the current derivations are all based on the isotropic assumption, which we could change $\varphi(R)$ into a more generic form of $\varphi(t;\boldsymbol{x}_t,\boldsymbol{x}_0)$ and solve the partial differential equation using the method of characteristic, to obtain a family of new models. Overall, this is really a factory of ODE-based diffusion models.

Some readers may ask, what is the purpose of deriving so many variants for diffusion models, if we just want a diffusion generative model that is practically workable. As a matter of fact, we want to discover and understand the fundamental principles behind the constructions of the diffusion generative models, such that we may discover other diffusion models that are more efficient. This is a never-ending process of perfectionism.

The numerical experimental results from the previously discussed gravitational diffusion model, as an ODE-based one, already led to a slightly better result than the conventional Gaussian diffusion model. This shows that, even they are all based on the isotropy assumption, the practical performances of these mathematically equivalent models can still be different from each other. Therefore, answering the question of what is a better design for the diffusion model would be a very meaningful research direction in the future.

15. The General Procedure of Constructing an ODE (Part II)

Original blog post see https://www.spaces.ac.cn/archives/9379

When writing the previous section, the author believed that we had already found a general procedure for constructing an ODE-based diffusion generative model. However, a reader has later provided a more efficient and straightforward alternative, which the author was truly amazed by the inspiration behind the model. After some discussions and thoughts, the author discovered that the fundamental idea behind this new approach is based on the **method of characteristics** for solving the first-order partial differential equations, in which we first construct a particular vector field to enforce the initial condition, and then the differential equation will be solved to match the boundary conditions that are to be satisfied by the end points. As such, the conditions for both the starting and end points that need to be matched are satisfied simultaneously. This is a really smart approach, which the author will summarise his thoughts in this section, as a sequel to the previous section.

15.1. Revisiting the previous results

We shall first briefly revise the results from the previous section. Suppose a random variable $\boldsymbol{x}_0 \in \mathbb{R}^d$ will transform itself continuously in space into $\boldsymbol{x}_T$ by following an ODE of the form,

$$\frac{d\boldsymbol{x}_t}{dt} = f_t(\boldsymbol{x}_t), \tag{15.1}$$

then the corresponding probability distribution $p_t(\boldsymbol{x}_t)$ at any given time t shall follow the following continuity equation:

$$\frac{\partial}{\partial t}p_t(\boldsymbol{x}_t) = -\nabla_{\boldsymbol{x}_t}\cdot\Big(f_t(\boldsymbol{x}_t)p_t(\boldsymbol{x}_t)\Big). \tag{15.2}$$

By letting $\boldsymbol{u}(t, \boldsymbol{x}_t) = \big(p_t(\boldsymbol{x}_t), f_t(\boldsymbol{x}_t)p_t(\boldsymbol{x}_t)\big) \in \mathbb{R}^{d+1}$, then the above equation can be simplified into

$$\begin{cases} \nabla_{(t,\boldsymbol{x}_t)} \cdot \boldsymbol{u}(t, \boldsymbol{x}_t) = 0 \\ \boldsymbol{u}_1(0, \boldsymbol{x}_0) = p_0(\boldsymbol{x}_0), \quad \int \boldsymbol{u}_1(t, \boldsymbol{x}_t) d\boldsymbol{x}_t = 1. \end{cases} \tag{15.3}$$

To solve this equation, we applied the technique of Green's function, in which we first solve

$$\begin{cases} \nabla_{(t,\boldsymbol{x}_t)} \cdot G(t, 0; \boldsymbol{x}_t, \boldsymbol{x}_0) = 0 \\ G_1(0, 0; \boldsymbol{x}_t, \boldsymbol{x}_0) = \delta(\boldsymbol{x}_t - \boldsymbol{x}_0), \qquad \int G_1(t, 0; \boldsymbol{x}_t, \boldsymbol{x}_0) = 1. \end{cases} \tag{15.4}$$

From here, we have the following solution

$$\boldsymbol{u}(t, \boldsymbol{x}_t) = \int G(t, 0; \boldsymbol{x}_t, \boldsymbol{x}_0) p_0(\boldsymbol{x}_0) d\boldsymbol{x}_0 = \mathbb{E}_{\boldsymbol{x}_0 \sim p_0(\boldsymbol{x}_0)} \left[G(t, 0; \boldsymbol{x}_t, \boldsymbol{x}_0) \right], \tag{15.5}$$

which satisfies our boundary conditions.

15.2. Geometric interpretations

The idea behind Green's function is simple. First of all, let us forget about synthesising a complex set of data, by just assuming we only have one data point $\boldsymbol{x}_0$ and try to figure out how we should synthesise this single data. This may look simple, in which we can simply take $\boldsymbol{x}_T \times 0 + \boldsymbol{x}_0$. However, this is an oversimplification, because we want a slowly-varying and continuous transformation, such that for any arbitrary $\boldsymbol{x}_t$ at $t < T$, it will follow a smooth trajectory to $\boldsymbol{x}_0$ at $t = 0$.

Since we are only interested in building a diffusion generative model, in principle, we are not interested in the actual shape of the trajectory, as long as they all pass through $\boldsymbol{x}_0$. In this case, we could choose any clusters of trajectories that we prefer, which will be collectively denoted as

$$\varphi_t(\boldsymbol{x}_t | \boldsymbol{x}_0) = \boldsymbol{x}_T. \tag{15.6}$$

Let us emphasise again that this represents a cluster of trajectories that take $\boldsymbol{x}_0$ as the starting point and the $\boldsymbol{x}_T$ as the end point. The independent and dependent variables in each trajectory are t and $\boldsymbol{x}_t$, respectively. The starting point $\boldsymbol{x}_0$ is always fixed, whereas the final endpoint $\boldsymbol{x}_T$ is variable. There is also no restriction on the shape of the trajectory, we could choose a straight line, a parabola or anything else.

Now we take the total derivatives to the both sides of Eq. (15.6). As $\boldsymbol{x}_T$ can be changed arbitrarily, it is equivalent to the integration constant in the differential equation, and its derivative is simply 0, as such, we have,

Not quite sure about this. This probably comes from the total derivative of function $f(\boldsymbol{x}(t), t)$ with respect to t is given by $\frac{\partial f}{\partial \boldsymbol{x}} \frac{d\boldsymbol{x}}{dt} + \frac{\partial f}{\partial t}$.

$$\frac{\partial \varphi_t(\boldsymbol{x}_t | \boldsymbol{x}_0)}{\partial \boldsymbol{x}_t} \frac{d\boldsymbol{x}_t}{dt} + \frac{\partial \varphi_t(\boldsymbol{x}_t | \boldsymbol{x}_0)}{\partial t} = \boldsymbol{0}, \tag{15.7}$$

which gives,

$$\frac{d\boldsymbol{x}_t}{dt} = - \left(\frac{\partial \varphi_t(\boldsymbol{x}_t | \boldsymbol{x}_0)}{\partial \boldsymbol{x}_t} \right)^{-1} \frac{\partial \varphi_t(\boldsymbol{x}_t | \boldsymbol{x}_0)}{\partial t}.$$

Comparing this to Eq. (15.1), we obtain

$$f_t(\boldsymbol{x}_t | \boldsymbol{x}_0) = - \left(\frac{\partial \varphi_t(\boldsymbol{x}_t | \boldsymbol{x}_0)}{\partial \boldsymbol{x}_t} \right)^{-1} \frac{\partial \varphi_t(\boldsymbol{x}_t | \boldsymbol{x}_0)}{\partial t}. \tag{15.8}$$

Here, we have changed the original function of $f_t(\boldsymbol{x}_t)$ into $f_t(\boldsymbol{x}_t | \boldsymbol{x}_0)$ to signify that all the trajectories have a common starting point of $\boldsymbol{x}_0$. This implies that, the trajectories from the force field built from this ODE will always pass through $\boldsymbol{x}_0$. This makes sure the initial value condition for the Green's function is always satisfied.

15.3. The method of characteristics

Once we have the condition for the initial value, we could ask further on how to satisfy the condition for the final value, which we want $\boldsymbol{x}_T$ to follow a simple distribution that is independent from $\boldsymbol{x}_0$. The limitation in the derivations shown in the previous section is that we cannot guarantee the final value follows a simple distribution, and must be analysed case-by-case in a post-hoc fashion. Here, our design principle is to satisfy the condition that is imposed on the initial values directly through the design of $\varphi_t(\boldsymbol{x}_t|\boldsymbol{x}_0)$, which leaves us with plenty of rooms to satisfy the conditions imposed on the final value. Once these conditions are satisfied, the integration constraint will be naturally satisfied provided that the continuity equation [Eq. (15.2)] is also satisfied.

Mathematically speaking, what we want to do is to solve Eq. (15.2) with a given $f_t(\boldsymbol{x}_t|\boldsymbol{x}_0)$ and $P_T(\boldsymbol{x}_T)$. This is a first-order partial differential equation which may be solved using the method of characteristics. Firstly, we rewrite Eq. (15.2) into the following equivalent form:

$$\frac{\partial}{\partial t}p_t(\boldsymbol{x}_t|\boldsymbol{x}_0) + \nabla_{\boldsymbol{x}_t}p_t(\boldsymbol{x}_t|\boldsymbol{x}_0)\cdot f_t(\boldsymbol{x}_t|\boldsymbol{x}_0) = -p_t(\boldsymbol{x}_t|\boldsymbol{x}_0)\nabla_{\boldsymbol{x}_t}f_t(\boldsymbol{x}_t|\boldsymbol{x}_0). \tag{15.9}$$

Similarly as before, we seek for its solution with a fixed starting point $\boldsymbol{x}_0$, such that we have rewritten $p_t(\boldsymbol{x}_t)$ as $p_t(\boldsymbol{x}_t|\boldsymbol{x}_0)$ to signify this as the solution with starting point $\boldsymbol{x}_0$.

The idea behind the method of characteristics is to first seek the solution of a partial differential equation along a specified trajectory, which transforms the partial differential equation into an ordinary differential equation, making it easier to solve. More specifically, we assume that $\boldsymbol{x}_t$ is a function of t, and solve the partial differential equation along the trajectory that is defined by Eq. (15.1). As Eq. (15.1) holds, we could replace $f_t(\boldsymbol{x}_t|\boldsymbol{x}_0)$ in Eq. (15.9) by $\frac{d\boldsymbol{x}_t}{dt}$, such that the L.H.S. of Eq. (15.9) becomes the total derivative of $P_t(\boldsymbol{x}_t|\boldsymbol{x}_0)$, which gives us

$$\frac{d}{dt}p_t(\boldsymbol{x}_t|\boldsymbol{x}_0) = -p_t(\boldsymbol{x}_t|\boldsymbol{x}_0)\nabla_{\boldsymbol{x}_t}f_t(\boldsymbol{x}_t|\boldsymbol{x}_0). \tag{15.10}$$

Noted that, at this point, all $\boldsymbol{x}_t$ should be replaced by a function with respect to t, which, in principle, can be solved from the equation for the trajectory [Eq. (15.6)]. With this substitution, both p and f in the above equation become purely the functions of t, such that the above equation is just linear ordinary differential equation of p, which should have the following solution:

$$p_t(\boldsymbol{x}_t|\boldsymbol{x}_0) = C\exp\left(\int_t^T \nabla_{\boldsymbol{x}_s}f_s(\boldsymbol{x}_s|\boldsymbol{x}_0)ds\right). \tag{15.11}$$

Substituting the condition for the final value $p_T(\boldsymbol{x}_T)$, we have $C = p_T(\boldsymbol{x}_T)$, such that

$$p_t(\boldsymbol{x}_t|\boldsymbol{x}_0) = p_T(\boldsymbol{x}_T)\exp\left(\int_t^T \nabla_{\boldsymbol{x}_s}f_s(\boldsymbol{x}_s|\boldsymbol{x}_0)ds\right). \tag{15.12}$$

If we further substitute into the equation for the trajectory [Eq. (15.6)], we obtain a function that will only include $\boldsymbol{x}_0$, $\boldsymbol{x}_t$ and t. This is the final result for the Green's function that we are trying to solve, i.e. $G_1(t,0;\boldsymbol{x}_t,\boldsymbol{x}_0)$. Correspondingly, $G_{>1}(t,0;\boldsymbol{x}_t,\boldsymbol{x}_0) = p_t(\boldsymbol{x}_t|\boldsymbol{x}_0)f_t(\boldsymbol{x}_t|\boldsymbol{x}_0)$.

15.4. The training target

From the Green's function, we obtain the following:

$$\begin{aligned}\boldsymbol{u}_1(t,\boldsymbol{x}_t) &= \int p_t(\boldsymbol{x}_t|\boldsymbol{x}_0)p_0(\boldsymbol{x}_0)d\boldsymbol{x}_0 = p_t(\boldsymbol{x}_t)\\ \boldsymbol{u}_{>1}(t,\boldsymbol{x}_t) &= \int f_t(\boldsymbol{x}_t|\boldsymbol{x}_0)p_t(\boldsymbol{x}_t|\boldsymbol{x}_0)p_0(\boldsymbol{x}_0)d\boldsymbol{x}_0,\end{aligned} \tag{15.13}$$

which leads to

$$\begin{aligned}f_t(\boldsymbol{x}_t) &= \frac{\boldsymbol{u}_{>1}(t,\boldsymbol{x}_t)}{\boldsymbol{u}_1(t,\boldsymbol{x}_t)}\\ &= \int f_t(\boldsymbol{x}_t|\boldsymbol{x}_0)\frac{p_t(\boldsymbol{x}_t|\boldsymbol{x}_0)p_0(\boldsymbol{x}_0)}{p_t(\boldsymbol{x}_t)}d\boldsymbol{x}_0\end{aligned}$$

$$= \int f_t(\boldsymbol{x}_t|\boldsymbol{x}_0)p_t(\boldsymbol{x}_0|\boldsymbol{x}_t)d\boldsymbol{x}_0$$
$$= \mathbb{E}_{\boldsymbol{x}_0\sim p_t(\boldsymbol{x}_0|\boldsymbol{x}_t)}[f_t(\boldsymbol{x}_t|\boldsymbol{x}_0)]. \tag{15.14}$$

Based on the discussion in Section 5, in which we built the training function based on score-matching, we can build our training target here similarly as

$$\mathbb{E}_{\boldsymbol{x}_t\sim p(\boldsymbol{x}_t)}\left[\mathbb{E}_{\boldsymbol{x}_0\sim p_t(\boldsymbol{x}_0|\boldsymbol{x}_t)}\left[||v_\theta(\boldsymbol{x}_t,t) - f_t(\boldsymbol{x}_t|\boldsymbol{x}_0)||_2^2\right]\right]$$
$$= \mathbb{E}_{\boldsymbol{x}_0,\boldsymbol{x}_t\sim p_t(\boldsymbol{x}_t|\boldsymbol{x}_0)p(\boldsymbol{x}_0)}\left[||v_\theta(\boldsymbol{x}_t,t) - f_t(\boldsymbol{x}_t|\boldsymbol{x}_0)||_2^2\right], \tag{15.15}$$

which has the same form as the **conditional flow matching** given in the article entitled "Flow Matching for Generative Modelling"[1]. As we shall see later, the results from this article can all be derived using the method proposed above. After model training, sampling can be achieved simply by solving the differential equation $\frac{d\boldsymbol{x}_t}{dt} = v_\theta(\boldsymbol{x}_t,t)$. It can be seen from Eq. (15.15) that, the only requirement that we have here is to have a distribution $p_t(\boldsymbol{x}_t|\boldsymbol{x}_0)$ that is easy to sample from.

15.5. A few examples

Probably the abstract results from above are not easy to be understood. In what follows, we will provide some examples to help developing some intuitive understanding.

15.5.1. Linear trajectory

As the simplest example, we assume that $\boldsymbol{x}_T$ transforms into $\boldsymbol{x}_0$ by following a linear trajectory. For simplicity, we can also set $T = 1$. Without loss of generality, we can write the equation for $\boldsymbol{x}_t$ as

$$\boldsymbol{x}_t = (\boldsymbol{x}_1 - \boldsymbol{x}_0)t + \boldsymbol{x}_0 \quad \Rightarrow \quad \frac{\boldsymbol{x}_t - \boldsymbol{x}_0}{t} + \boldsymbol{x}_0 = \boldsymbol{x}_1. \tag{15.16}$$

According to Eq. (15.8), we have

$$f_t(\boldsymbol{x}_t|\boldsymbol{x}_0) = \frac{\boldsymbol{x}_t - \boldsymbol{x}_0}{t}, \tag{15.17}$$

and now $\nabla_{\boldsymbol{x}_t}\cdot f_t(\boldsymbol{x}_t|\boldsymbol{x}_0) = \frac{d}{t}$, such that, according to Eq. (15.12), we have

$$p_t(\boldsymbol{x}_t|\boldsymbol{x}_0) = \frac{p_1(\boldsymbol{x}_1)}{t^d}. \tag{15.18}$$

Now substituting in the expression for $\boldsymbol{x}_1$ from Eq. (15.16), we have

$$p_t(\boldsymbol{x}_t|\boldsymbol{x}_0) = \frac{p_1\left(\frac{\boldsymbol{x}_t-\boldsymbol{x}_0}{t} + \boldsymbol{x}_0\right)}{t^d}. \tag{15.19}$$

More specifically, if $p_1(\boldsymbol{x}_1)$ follows the standard normal distribution, then the above equation should follow $p_t(\boldsymbol{x}|\boldsymbol{x}_0) \sim \mathcal{N}(\boldsymbol{x}_t;(1-t)\boldsymbol{x}_0,t^2\boldsymbol{I})$, which corresponds to the result from the Gaussian diffusion model. The new result under this framework, *is to allow us to choose more general prior distributions*, such as the uniform distribution. Also, when introducing the score-matching in Eq. (15.15), we already mentioned that all we need to know is how to sample from the distribution of $p_t(\boldsymbol{x}_t|\boldsymbol{x}_0)$, and Eq. (15.19) tells us that this is easily achievable if the prior distribution is on that is easy to sample from. This is because

$$\boldsymbol{x}_t \sim p_t(\boldsymbol{x}_t|\boldsymbol{x}_0) \quad \Leftrightarrow \quad \boldsymbol{x}_t = (1+t)\boldsymbol{x}_0 + t\boldsymbol{\varepsilon} \quad \boldsymbol{\varepsilon}\sim p_1(\boldsymbol{\varepsilon}). \tag{15.20}$$

15.5.2. A further generalisation

The above result can be further generalised into the following:

$$\boldsymbol{x}_t = \boldsymbol{\mu}_t(\boldsymbol{x}_t) + \sigma_t\boldsymbol{x}_1 \quad \Rightarrow \quad \boldsymbol{x}_1 = \frac{\boldsymbol{x}_t - \boldsymbol{\mu}_t(\boldsymbol{x}_t)}{\sigma_t}. \tag{15.21}$$

Here, $\boldsymbol{\mu}_t(\boldsymbol{x}_t)$ is any arbitrary function that satisfies the boundary conditions of $\boldsymbol{\mu}_0(\boldsymbol{x}_0) = \boldsymbol{x}_0$ and $\boldsymbol{\mu}_1(\boldsymbol{x}_0) = 0$, that is defined over the domain of $\mathbb{R}^d \to \mathbb{R}^d$. σ_t is a monotonically increasing

function from $\sigma_0 = 0$ to $\sigma_1 = 1$. According to Eq. (15.8), we have the following

$$f_t(\boldsymbol{x}_t|\boldsymbol{x}_0) = \dot{\boldsymbol{\mu}}_t(\boldsymbol{x}_0) + \frac{\dot{\sigma_t}}{\sigma_t}\Big(\boldsymbol{x}_t - \boldsymbol{\mu}_t(\boldsymbol{x}_0)\Big). \tag{15.22}$$

This is equivalent to Eq. (15) in the article "Flow Matching for Generative Modelling". Now $\nabla_{\boldsymbol{x}_t} f_t(\boldsymbol{x}_t|\boldsymbol{x}_0) = d\dot{\sigma_t}/\sigma_t$, and from eq:15.12, we have

$$p_t(\boldsymbol{x}_t|\boldsymbol{x}_0) = p_1(\boldsymbol{x}_1)/\sigma_t^d. \tag{15.23}$$

Substituting in the expression for $\boldsymbol{x}_1$, we have the following final result:

$$p_t(\boldsymbol{x}_t|\boldsymbol{x}_0) = \frac{p_1\left(\frac{\boldsymbol{x}_t - \boldsymbol{\mu}_t(\boldsymbol{x}_0)}{\sigma_t}\right)}{\sigma_t^d}. \tag{15.24}$$

This is the general result for linear ODE based diffusion model. It include prior distributions that can be either Gaussian or non-Gaussian.

15.5.3. Even more complicated?

In all the examples above, the transformations (or the trajectories) from $\boldsymbol{x}_0$ to $\boldsymbol{x}_t$ are all established based on a simple linear interpolation function of t. A natural question to ask is, then, can we consider more complicated trajectories? This is theoretically possible. However, using more complex trajectories implies the need to embed more assumptions in the model, and it is usually difficult for us to prove whether all the target data follow these assumptions. Therefore, it is usually not necessary to consider more complex trajectories. In addition, for complex trajectories, it becomes more difficult to derive analytical solutions, making it impossible to proceed both theoretically and practically.

Most importantly, we are now only dealing with the trajectories for synthesising a single sample. It is not difficult to show that, even though this trajectory is a straight line, for synthesising multiple samples, the trajectories will lead to a more complex curves. In this case, if we already made the trajectory for synthesising a single sample to be unnecessarily complicated, then for multiple samples, the complexities behind the trajectories will be extremely high, making the model more likely to become unstable.

15.6. Summary

In summary, we have followed the context from the previous section, and further discussed the idea behind building an ODE-based diffusion model. This time, starting from a geometric intuition, we build specific vector fields to ensure the results follows the distributions of the initial values, and satisfy those for the final values by solving the differential equations. This allows us to solve for the Green's function that satisfies the conditions that both the initial and final values should follow. More specifically, it allows us to use any simple distributions as the prior distributions, thus overcoming the reliance on the Gaussian distributions in constructing the diffusion models.

16. Wasserstein Distance $\leq$ Score-Matching

Original blog post see https://www.spaces.ac.cn/archives/9467

The Wasserstein distance, is a distance measure that quantifies the differences between two probability distributions defined based on the optimum transport theory. Many readers first came across the Wasserstein distance were probably from the birth of WGAN in 2017, which opened up a new branch for understanding GAN from the perspective of the optimum transport theory. It also made this theory start to play more important role in machine learning. For a long time, GAN has been the leader in the generative modelling, until the rise of the diffusion models in the past few years, even though GAN still remains as a compelling generative model.

The fundamental model architectures behind the diffusion models and GAN are very different, therefore, the research on these two models have been relatively independent from each other. However, the article entitled "*Score-Based Generative Modelling Secretly Minimizes the Wasserstein Distance*"[20] published at the end of 2022 had broken this barrier. It proved that the loss function for the diffusion model derived from the score-matching can be written as the upper limit of the Wasserstein distance. This implies that, when weminimise the loss function for the diffusion model, what we do is effectively the same as WGAN, in which the Wasserstein distance between two probability distributions is also minimised.

16.1. Result analysis

More specifically, the results presented in the original paper, was directly aligned with the SDE-based diffusion generative models discussed previously in Section 5. The core result is the following inequality:

$$\mathcal{W}_2[p_0, q_0] \leq \int_0^T g_t^2 I_t \Big(\mathbb{E}_{\boldsymbol{x}_t \sim p_t(\boldsymbol{x}_t)} \Big[||\nabla_{\boldsymbol{x}_t} \log p_t(\boldsymbol{x}_t) - s_\theta(\boldsymbol{x}_t, t)||_2^2 \Big] \Big)^{\frac{1}{2}} dt + I_T \mathcal{W}_2[p_T, q_T], \quad (16.1)$$

in which I_t is a non-negative function of t, the meaning of which will be discussed later.

How should we understand this inequality? Firstly, a diffusion model can be understood as a motion from $t = T$ to $t = 0$ that is described by an SDE. The terms p_T and q_T on the far right-hand-side are the random distribution for sampling at T, where p_T is usually the standard normal distribution. In real practices, we have $p_T = q_T$ such that $\mathcal{W}_2[p_T, q_T] = 0$. It was left in the equation to give the most theoretically generalised result. Secondly, p_0 on the left-hand-side, is obtained from solving the following SDE:

$$d\boldsymbol{x}_t = \Big[f_t(\boldsymbol{x}_t) - g_t^2 \nabla_{\boldsymbol{x}_t} \log p_t(\boldsymbol{x}_t) \Big] dt + g_t dw, \quad (16.2)$$

in the reversed order, starting from a random point $\boldsymbol{x}_T$ that is sampled from p_T. This is the target sample distribution that we would like to generate. On the other hand q_0 is the distribution obtained from solving the SDE:

$$d\boldsymbol{x}_t = \Big[f_t(\boldsymbol{x}_t) - g_t^2 s_\theta(\boldsymbol{x}_t, t) \Big] dt + g_t dw, \quad (16.3)$$

that starts from a random point sampled from q_T to reach the corresponding distribution at $t = 0$, in which $s_\theta(\boldsymbol{x}_t, t)$ is a neural network approximation to the score function $\nabla_{\boldsymbol{x}_t} p_t(\boldsymbol{x}_t)$. As such, q_0 is the sample distribution that is generated from the diffusion generative model, and $\mathcal{W}_2[p_0, q_0]$ represents the Wasserstein distance between the true and generated sample distributions. Lastly, the remainder is the integral term, in which the key part is the integrand:

$$\mathbb{E}_{\boldsymbol{x}_t \sim p_t(\boldsymbol{x}_t)} \Big[||\nabla_{\boldsymbol{x}_t} \log p_t(\boldsymbol{x}_t) - s_\theta(\boldsymbol{x}_t, t)||_2^2 \Big], \quad (16.4)$$

which is nothing but the "score-matching" loss for the diffusion generative model.

the above result shows that, when we train the diffusion model with the score-matching loss, we have indirectly minimised the Wasserstein distance between the sample and the generated distributions. Differently from WGAN, the latter optimises $\mathcal{W}_1[p_0, q_0]$ distance, whereas here, the $\mathcal{W}_2[p_0, q_0]$ distance is optimised.

Note: Strictly speaking, Eq. (16.4) is not the loss function for the diffusion model, the latter should really be the "conditional score-matching", and its relationship with score-matching is as following:

$$\begin{aligned}
&\mathbb{E}_{\boldsymbol{x}_t \sim p_t(\boldsymbol{x}_t)} \Big[||\nabla_{\boldsymbol{x}_t} \log p_t(\boldsymbol{x}_t) - s_\theta(\boldsymbol{x}_t, t)||_2^2 \Big] \\
&= \mathbb{E}_{\boldsymbol{x}_t \sim p_t(\boldsymbol{x}_t)} \Bigg[\Big\| \mathbb{E}_{\boldsymbol{x}_0 \sim p_t(\boldsymbol{x}_0|\boldsymbol{x}_t)} \Big[\nabla_{\boldsymbol{x}_t} \log p_t(\boldsymbol{x}_t|\boldsymbol{x}_0) \Big] - s_\theta(\boldsymbol{x}_t, t) \Big\|_2^2 \Bigg] \\
&\leq \mathbb{E}_{\boldsymbol{x}_t \sim p_t(\boldsymbol{x}_t)} \mathbb{E}_{\boldsymbol{x}_0 \sim p_t(\boldsymbol{x}_0|\boldsymbol{x}_t)} \Big[||\nabla_{\boldsymbol{x}_t} \log p_t(\boldsymbol{x}_t|\boldsymbol{x}_0) - s_\theta(\boldsymbol{x}_t, t)||_2^2 \Big] \\
&= \mathbb{E}_{\boldsymbol{x}_0 \sim p_0(\boldsymbol{x}_0), \boldsymbol{x}_t \sim p_t(\boldsymbol{x}_t|\boldsymbol{x}_0)} \Big[||\nabla_{\boldsymbol{x}_t} \log p_t(\boldsymbol{x}_t|\boldsymbol{x}_0) - s_\theta(\boldsymbol{x}_t, t)||_2^2 \Big],
\end{aligned} \quad (16.5)$$

in which the last line corresponds to the conditional score-matching loss function for the diffusion generative model. The first equality holds because $\nabla_{\boldsymbol{x}_t} \log p_t(\boldsymbol{x}_t) = \mathbb{E}_{\boldsymbol{x}_0 \sim p_t(\boldsymbol{x}_0|\boldsymbol{x}_t)}[\nabla_{\boldsymbol{x}_t} \log p_t(\boldsymbol{x}_t|\boldsymbol{x}_0)]$. The second inequality is based on the Jansen's inequality. Finally, the third equation is from the Bayesian formula. This implies that the conditional score-matching is the upper limit to score-matching, therefore, it should also be the upper limit to the Wasserstein distance.

From Eq. (16.1), we can also better rationalise why we need to drop the coefficients before the modulus in the target function for training the diffusion model. Here, the Wasserstein

distance is a well-defined metric for measuring the similarity between the two probability distribution functions, while $g_t^2 I_t$ in Eq. (16.1) is a monotonically increasing function of t. This means that we need to properly increase the score-matching loss when t is small. In Section 5, we had derived the final form of the score-matching loss being

$$\frac{1}{\bar{\beta}_t^2}\mathbb{E}_{\boldsymbol{x}_0\sim\tilde{p}_0(\boldsymbol{x}_0)}\Big[\big\|\varepsilon_\theta\left(\bar{\alpha}_t\boldsymbol{x}_0+\bar{\beta}_t\varepsilon,t\right)-\varepsilon\big\|_2^2\Big]. \tag{16.6}$$

Ignoring the coefficient $1/\bar{\beta}_t^2$ is equivalent to multiplying the loss function with $\bar{\beta}_t^2$, which is also a monotonically increasing function of t. As such, for simplicity, we can consider ignoring the coefficient is an approach to make it closer to the Wasserstein distance between the two probability distributions.

16.2. Preparatory works

Although the original article has provided the proof for Eq. (16.1), it requires many prerequisite knowledges such as the continuity equation, gradient flow, and so on. Especially it used one theory without proof, which as hidden in the Chapter 8 of a monograph on gradient flow, or Chapter 5 of a monograph on the optimum transport theory, which was too difficult for the author to understand. After some trial, the author had finally arrived a (partial) proof of Eq. (16.1), which only requires the usage of the definition for the Wasserstein distance, fundamentals of differential equations and the Cauchy inequality. This is much simpler than the original proof. After some modifications and improvements, it is presented as follows.

Before proceeding to the proof, let us first make some preparations, which we shall summarise the basic concepts and conclusions that we need to apply subsequently. Firstly, the Wasserstein distance is defined as

$$\mathcal{W}_\rho[p,q]=\left(\inf_{\gamma\in\Pi[p,q]}\iint\gamma(\boldsymbol{x},\boldsymbol{y})\|\boldsymbol{x}-\boldsymbol{y}\|^\rho d\boldsymbol{x}d\boldsymbol{y}\right)^{\frac{1}{\rho}}, \tag{16.7}$$

in which $\Pi[p,q]$ is the probability density function for all the joint probability distributions in which p, q are the corresponding marginal distributions. It describes the concrete proposals for the transportation of masses between the two distributions.

Here, we will only consider the case in which $\rho=2$, which is the only case that gives a simple derivation in what follows. Note that in the above definition for the Wasserstein distance, it includes the computation of the definitive lower bound (inf). This implies that, for any $\gamma\in\Pi[p,q]$ that we can find, we will always have

$$\mathcal{W}_2[p,q]\leq\left(\iint\gamma(\boldsymbol{x},\boldsymbol{y})\|\boldsymbol{x}-\boldsymbol{y}\|^2 d\boldsymbol{x}d\boldsymbol{y}\right)^{\frac{1}{2}}. \tag{16.8}$$

This is the key behind our proof here. To prove this bound, we shall be applying the Cauchy inequality:

$$\begin{aligned}\text{in vector form}\qquad&\boldsymbol{x}\cdot\boldsymbol{y}\leq\|\boldsymbol{x}\|\cdot\|\boldsymbol{y}\|,\\ \text{in the form of expectation}\qquad&\mathbb{E}_{\boldsymbol{x}}[f(\boldsymbol{x})\cdot g(\boldsymbol{x})]\leq\big(\mathbb{E}_x[f^2(\boldsymbol{x})]\big)^{\frac{1}{2}}\cdot\big(\mathbb{E}_x[g^2(\boldsymbol{x})]\big)^{\frac{1}{2}}.\end{aligned} \tag{16.9}$$

Additionally, in the following proof, we will also assume that the function $g_t(\boldsymbol{x})$ satisfies the single-sided Lipschitz constraint defined as

$$\Big(g_t(\boldsymbol{x})-g_t(\boldsymbol{y})\Big)\cdot(\boldsymbol{x}-\boldsymbol{y})\leq\mathcal{L}_t\|\boldsymbol{x}-\boldsymbol{y}\|^2, \tag{16.10}$$

which is a weaker version of the usual Lipschitz constraint, *i.e.* a function that satisfies the full Lipschitz constraint should also satisfy the single-sided Lipschitz constraint. Basically this will give a more appropriate definition of the Wasserstein distance and stable gradient during model training.

16.3. The initial trial

Eq. (16.1) is an overly generalised result, analysing which will not help us in rationalising and understanding the meaning behind. Therefore, we must first simplify the problem to see whether we can prove a weaker result. How should we simplify the problem? First of all, Eq. (16.1) considered the differences in the prior distributions (p_T and q_T), which we shall

Whilst the proof here seems not too challenge to understand, it does take a lot of inspirations to come up with such a proof!

consider them as being identical to each other. Secondly, the reverse process underpinning the derivation of Eq. (16.1) was based on an SDE [Eq. (16.2)], which we shall first consider a definitive transformation that is described by an ODE.

More specifically, let us consider a sample $\boldsymbol{z}$ that is drawn from the distribution $q(\boldsymbol{z})$ at $t = T$, which we shall treat it as our initial value, and then we shall let it evolve along trajectories defined by two different ODEs:

$$\frac{d\boldsymbol{x}_t}{dt} = f_t(\boldsymbol{x}_t) \qquad \frac{d\boldsymbol{y}_t}{dt} = g_t(\boldsymbol{y}_t). \tag{16.11}$$

Assuming that, at any given time t, the distributions of $\boldsymbol{x}_t$ and $\boldsymbol{y}_t$ are p_t and q_t, respectively. We shall try to estimate an upper bound for $\mathcal{W}_2[p_0, q_0]$.

Since both $\boldsymbol{x}_t$ and $\boldsymbol{y}_t$ are evolved from the same initial value of $\boldsymbol{z}$ with respect to their own ODEs, therefore, they are both well-defined functions of $\boldsymbol{z}$, which should be more precisely denoted as $\boldsymbol{x}_t(\boldsymbol{z})$ and $\boldsymbol{y}_t(\boldsymbol{z})$, but we have omitted $\boldsymbol{z}$ for brevity. This means that, for a given $\boldsymbol{x}$, $\boldsymbol{x}_t \leftrightarrow \boldsymbol{y}_t$ establishes a transport pathway between p_t and q_t. As such, based on Eq. (16.8), we can write

$$\mathcal{W}_2^2[p_t, q_t] \leq \mathbb{E}_{\boldsymbol{z}}\left[\|\boldsymbol{x}_t - \boldsymbol{y}_t\|_2^2\right] \triangleq \tilde{\mathcal{W}}_2^2[p_t, q_t]. \tag{16.12}$$

Now we shall try to find the bound for $\tilde{\mathcal{W}}_2^2[p_t, q_t]$. In order to link it with $f_t(\boldsymbol{x}_t)$ and $g_t(\boldsymbol{y}_t)$, we shall take the derivative of $\tilde{\mathcal{W}}_2^2[p_t, q_t]$ with respect to t:

$$\begin{aligned}
\pm\frac{d\left(\tilde{\mathcal{W}}_2^2[p_t, q_t]\right)}{dt} &= \pm 2\mathbb{E}_{\boldsymbol{z}}\left[(\boldsymbol{x}_t - \boldsymbol{y}_t)\cdot\left(\frac{d\boldsymbol{x}_t}{dt} - \frac{d\boldsymbol{y}_t}{dt}\right)\right] \\
&= \pm 2\mathbb{E}_{\boldsymbol{z}}\left[(\boldsymbol{x}_t - \boldsymbol{y}_t)\cdot(f_t(\boldsymbol{x}_t) - g_t(\boldsymbol{y}_t))\right] \\
&= \pm 2\mathbb{E}_{\boldsymbol{z}}\underbrace{\left[(\boldsymbol{x}_t - \boldsymbol{y}_t)\cdot(f_t(\boldsymbol{x}_t) - g_t(\boldsymbol{x}_t))\right]}_{\text{Cauchy inequality (vector form)}} \pm 2\mathbb{E}_{\boldsymbol{z}}\underbrace{\left[(\boldsymbol{x}_t - \boldsymbol{y}_t)\cdot(g_t(\boldsymbol{x}_t) - g_t(\boldsymbol{y}_t))\right]}_{\text{Lipschitz constraint}} \\
&\leq 2\mathbb{E}_{\boldsymbol{z}}\left[\|\boldsymbol{x}_t - \boldsymbol{y}_t\|\cdot\|f_t(\boldsymbol{x}_t) - g_t(\boldsymbol{x}_t)\|\right] + 2\mathbb{E}_{\boldsymbol{z}}\left[\mathcal{L}_t\|\boldsymbol{x}_t - \boldsymbol{y}_t\|^2\right] \\
&\leq 2\left(\mathbb{E}_{\boldsymbol{z}}\left[\|\boldsymbol{x}_t - \boldsymbol{y}_t\|^2\right]^2\right)^{\frac{1}{2}}\left(\mathbb{E}_{\boldsymbol{z}}\left[\|f_t(\boldsymbol{x}_t) - g_t(\boldsymbol{y}_t)\|^2\right]^2\right)^{\frac{1}{2}} + 2\mathbb{E}_{\boldsymbol{z}}\left[\mathcal{L}_t\|\boldsymbol{x}_t - \boldsymbol{y}_t\|^2\right] \\
&= 2\tilde{\mathcal{W}}_2[p_t, q_t]\left(\mathbb{E}_{\boldsymbol{z}}\left[\|f_t(\boldsymbol{x}_t) - g_t(\boldsymbol{y}_t)\|^2\right]^2\right)^{\frac{1}{2}} + 2\mathcal{L}_t\tilde{\mathcal{W}}_2^2[p_t, q_t],
\end{aligned} \tag{16.13}$$

in which, to derive the first inequality, we need to use the vector form of the Cauchy inequality and the Lipschitz constraint [Eq. (16.10)], and we applied the expectation form of the Cauchy inequality in deriving the second inequality. The $\pm$ sign implies that the final inequality will hold regardless of whether we take the $+$ or $-$ sign. In what follows, we take the results from the $-$ side. Using the fact that $(w^2)' = 2ww'$, in which w on both sides of the equation cancel out each other, we get

$$-\frac{d\left(\tilde{\mathcal{W}}_2[p_t, q_t]\right)}{dt} \leq \left(\mathbb{E}_{\boldsymbol{z}}\left[\|f_t(\boldsymbol{x}_t) - g_t(\boldsymbol{y}_t)\|_2^2\right]^2\right)^{\frac{1}{2}} + \mathcal{L}_t\tilde{\mathcal{W}}_2[p_t, q_t]. \tag{16.14}$$

Using the variation of parameters, let us further assume that $\tilde{\mathcal{W}}_2[p_t, q_t] = C_t\exp\left(\int_t^T \mathcal{L}_s ds\right)$, and substitute this into the equation above, we get

Not sure how the $\mathcal{L}_t$ term in Eq. (16.14) becomes eliminated here?

$$-\frac{dC_t}{dt} \leq \exp\left(-\int_t^T \mathcal{L}_s ds\right)\left(\mathbb{E}_{\boldsymbol{z}}\left[\|f_t(\boldsymbol{x}_t) - g_t(\boldsymbol{y}_t)\|_2^2\right]^2\right)^{\frac{1}{2}}. \tag{16.15}$$

Integrating both sides over the interval $[0, T]$, and using the initial condition $C_T = 0$ (identical prior distributions with distance between them being zero), we get:

$$C_0 \leq \int_0^T \exp\left(-\int_t^T \mathcal{L}_s ds\right)\left(\mathbb{E}_{\boldsymbol{z}}\left[\|f_t(\boldsymbol{x}_t) - g_t(\boldsymbol{y}_t)\|_2^2\right]^2\right)^{\frac{1}{2}} dt, \tag{16.16}$$

as such, This bit is also a bit puzzling.

$$\tilde{\mathcal{W}}_2[p_0, q_0] \leq C_0 \exp\left(\int_0^T \mathcal{L}_s ds\right) = \int_0^T I_t \left(\mathbb{E}_{\boldsymbol{z}}\left[\|f_t(\boldsymbol{x}_t) - g_t(\boldsymbol{y}_t)\|_2^2\right]^2\right)^{\frac{1}{2}} dt, \tag{16.17}$$

in which $I_t = \exp\left(\int_0^t \mathcal{L}_s ds\right)$. According to Eq. (16.12), this is also the upper limit for $\mathcal{W}_2[p_0, q_0]$. Finally, since the expectation value is a function of $\boldsymbol{x}_t$ and $\boldsymbol{x}_t$ is a function of $\boldsymbol{z}$, therefore, the expectation value taken over $\boldsymbol{z}$ is equivalent to the expectation value over $\boldsymbol{x}_t$, such that,

$$\tilde{\mathcal{W}}_2[p_0, q_0] \leq \int_0^T I_t \left(\mathbb{E}_{\boldsymbol{x}_t \sim p_t(\boldsymbol{x}_t)}\left[\|f_t(\boldsymbol{x}_t) - g_t(\boldsymbol{y}_t)\|_2^2\right]^2\right)^{\frac{1}{2}} dt. \tag{16.18}$$

16.4. Continuing the proof

As a matter of fact, the simplified version shown in the above inequality (16.18) does not have a fundamental difference compared to the more generalised form of (16.1). Its derivation has already included all the general ideas behind the full derivation of the generalised result. Now let us finalise the remaining part of the proof.

First, we need to generalise Eq. (16.18) to the case where the initial (prior) distributions are different from each other. Assuming that the two prior distributions are $p_T(\boldsymbol{z}_1)$ and $q_T(\boldsymbol{z}_2)$, we sample the initial value from $p_T(\boldsymbol{z}_1)$ to evolve $\boldsymbol{x}_t$, and from $q_T(\boldsymbol{z}_2)$ to evolve $\boldsymbol{y}_t$. As such, $\boldsymbol{x}_t$ and $\boldsymbol{y}_t$ are functions of $\boldsymbol{z}_1$ and $\boldsymbol{z}_2$, respectively, not the function of an identical $\boldsymbol{z}$ as we assumed before. In this case, we cannot straight away build a single transport protocol, but also need another transport protocol between $\boldsymbol{z}_1$ and $\boldsymbol{z}_2$. Let us choose the optimum transport protocol $\gamma^*(\boldsymbol{z}_1, \boldsymbol{z}_2)$ between the two distributions of $p_T(\boldsymbol{z}_1)$ and $q_T(\boldsymbol{z}_2)$. Similar to Eq. (16.12), we now have

$$\mathcal{W}_2^2[p_t, q_t] \leq \mathbb{E}_{\boldsymbol{z}_1, \boldsymbol{z}_2 \sim \gamma^*(\boldsymbol{z}_1, \boldsymbol{z}_2)}\left[\|\boldsymbol{x}_t - \boldsymbol{y}_t\|_2^2\right] \triangleq \tilde{\mathcal{W}}_2^2[p_t, q_t]. \tag{16.19}$$

Because we did not change the definitions, except the expectation from $\mathbb{E}_{\boldsymbol{z}}$ to $\mathbb{E}_{\boldsymbol{z}_1, \boldsymbol{z}_2}$, both the procedure shown in Eq. (16.13) and the inequalities (16.14) and (16.15) still hold. However, when we take the integral over $[0, T]$ on both sides of (16.15), we no longer have $C_T = 0$, but $C_T = \tilde{\mathcal{W}}_2[p_T, q_T] = \mathcal{W}_2[p_T, q_T]$, so the final result is

$$\tilde{\mathcal{W}}_2[p_0, q_0] \leq \int_0^T I_t \left(\mathbb{E}_{\boldsymbol{x}_t \sim p_t(\boldsymbol{x}_t)}\left[\|f_t(\boldsymbol{x}_t) - g_t(\boldsymbol{y}_t)\|_2^2\right]^2\right)^{\frac{1}{2}} dt + I_T \mathcal{W}_2[p_T, q_T]. \tag{16.20}$$

At this point, let us return to the diffusion model. In Section 6, we have shown that, for a given forward process, it has a *set* of corresponding reverse processes that are described by the following SDE:

$$d\boldsymbol{x} = \left(f_t(\boldsymbol{x}_t) - \frac{1}{2}\left(g_t^2 + \sigma_t^2\right)\nabla_{\boldsymbol{x}_t} \log p_t(\boldsymbol{x}_t)\right) dt + \sigma_t dw, \tag{16.21}$$

in which σ_t is a time-dependent function of standard deviation that we are free to choose, and Eq. (16.2) corresponds to the case where $\sigma_t = g_t$. As we have only considered ODE above, we will consider the case in which $\sigma_t = 0$. In this case, (16.20) still holds. What we only need to do is to change $f_t(\boldsymbol{x}_t)$ into $f_t(\boldsymbol{x}_t) - \frac{1}{2} g_t^2 \nabla_{\boldsymbol{x}_t} \log p_t(\boldsymbol{x}_t)$ and change $g_t(\boldsymbol{x}_t)$ into $f_t(\boldsymbol{x}_t) - \frac{1}{2} g_t^2 s_\theta(\boldsymbol{x}_t, t)$. Substituting these back into Eq. (16.20), we recover the result (16.1) shown at the beginning. Of course, do not forget that the single-sided Lipschitz constraint (16.10) that we must impose on $g_t(\boldsymbol{x}_t)$, which now we could further elaborate our assumption on $f_t(\boldsymbol{x}_t)$ and $s_\theta(\boldsymbol{x}_t, t)$, which we will not discuss these results here.

16.5. Further generalisation and its difficulties

Supposedly, we should carry on the work and finish the proof for the case where $\sigma_t \neq 0$. Unfortunately, the idea presented in this section cannot be fully generalised to the case of SDE, which we shall provide some analysis on why this is the case in what follows. As a matter of face, for most readers, it should be sufficient to understand the case of ODE-based derivation, the full details for SDE case are not that critically important.

For simplicity, let us take Eq. (16.2) as an example. What we need to do is to estimate the difference between the distributions for the evolving trajectories as described by the following

two SDEs:

$$\begin{cases} d\boldsymbol{x}_t = \Big[f_t(\boldsymbol{x}_t) - g_t^2 \nabla_{\boldsymbol{x}_t} \log p_t(\boldsymbol{x}_t)\Big] dt + g_t dw \\ d\boldsymbol{y}_t = \Big[f_t(\boldsymbol{y}_t) - g_t^2 s_\theta(\boldsymbol{y}_t, t)\Big] dt + g_t dw. \end{cases} \tag{16.22}$$

In another word, we want to know how much the final distribution will change if we replace the accurate score function $\nabla_{\boldsymbol{x}_t} \log p_t(\boldsymbol{x}_t)$ by the estimate $s_\theta(\boldsymbol{y}_t, t)$. We use the same idea of transforming the SDE into an ODE first, from which we can reuse the proof before. According to Eq. (16.21), we know that the corresponding ODE to the first SDE is

$$d\boldsymbol{x}_t = \left[f_t(\boldsymbol{x}_t) - \frac{1}{2} g_t^2 \nabla_{\boldsymbol{x}_t} \log p_t(\boldsymbol{x}_t)\right] dt. \tag{16.23}$$

Deriving the ODE for the second SDE is a bit tricky, which we need to first change it into the form of $-g_t^2 \nabla_{\boldsymbol{y}_t} \log g_t(\boldsymbol{y}_t)$ before applying Eq. (16.21):

$$\begin{aligned} d\boldsymbol{y}_t &= \left[\underbrace{f_t(\boldsymbol{y}_t) - g_t^2 s_\theta(\boldsymbol{y}_t, t) + g_t^2 \nabla_{\boldsymbol{y}_t} \log q_t(\boldsymbol{y}_t)}_{\text{treat as a whole}} - g_t^2 \nabla_{\boldsymbol{y}_t} \log q_t(\boldsymbol{y}_t)\right] dt + g_t dw \\ &= \left[f_t(\boldsymbol{y}_t) - g_t^2 s_\theta(\boldsymbol{y}_t, t) + g_t^2 \nabla_{\boldsymbol{y}_t} \log q_t(\boldsymbol{y}_t) - \frac{1}{2} g_t^2 \nabla_{\boldsymbol{y}_t} \log q_t(\boldsymbol{y}_t)\right] dt \\ &= \left[f_t(\boldsymbol{y}_t) - g_t^2 s_\theta(\boldsymbol{y}_t, t) + \frac{1}{2} g_t^2 \nabla_{\boldsymbol{y}_t} \log q_t(\boldsymbol{y}_t)\right] dt. \end{aligned} \tag{16.24}$$

Repeat the process shown in Eq. (16.13) for the above two ODEs, then the major difference is that we now have an extra term:

$$-\frac{1}{2} g_t^2 \mathbb{E}_{\boldsymbol{z}}\Big[(\boldsymbol{x}_t - \boldsymbol{y}_t)\left(\nabla_{\boldsymbol{x}_t} p_t(\boldsymbol{x}_t) - \nabla_{\boldsymbol{y}_t} \log q_t(\boldsymbol{y}_t)\right)\Big]. \tag{16.25}$$

If this term is negative, then the result from Eq. (16.13) will still hold. So the key question is whether we can prove the following:

$$\mathbb{E}_{\boldsymbol{z}}\Big[(\boldsymbol{x}_t - \boldsymbol{y}_t)\left(\nabla_{\boldsymbol{x}_t} p_t(\boldsymbol{x}_t) - \nabla_{\boldsymbol{y}_t} \log q_t(\boldsymbol{y}_t)\right)\Big] \geq 0.$$

However, we could prove that this does not generally hold with specific examples. In the original article, a similar term has also appeared, except that the expectation value was taken over the transport proposals between $\boldsymbol{x}_t$ and $\boldsymbol{y}_t$. The article has provided a simple proof, but for the newcomers with limited knowledge on the optimum transport theory, we can only stop here.

More specifically, we cannot impose the single-sided Lipschitz constraint over the score functions, as we can easily find examples where the constraint is violated. As such, to prove the above inequality, we can only follow the methodology provided in the original article, without imposing other constraints but obeying the properties of the data distributions.

17. The General Procedure of Constructing an ODE (Part II)

Original blog post see https://www.spaces.ac.cn/archives/9497

After completing Section 15, the author believed that a general procedure for constructing an ODE-based diffusion model had been completed. However, later, the author was brought into the attention of a new ICLR2023 article entitled "*Flow Straight and Fast, Learning to Generate and Transfer Data with Rectified Flow*"[21], which provided another proposal for building an ODE-based diffusion model that is extremely simple and elegant. This will be discussed in this section.

17.1. The intuitive results

As we know, the diffusion model is modelling a stochastic process that evolves $\boldsymbol{x}_T$ to $\boldsymbol{x}_0$, whereas an ODE-based diffusion model fixes such an evolution process, by enforcing the evolution to occur by following an ODE:

$$\frac{d\boldsymbol{x}_t}{dt} = f_t(\boldsymbol{x}_t). \tag{17.1}$$

The core of building an ODE-based diffusion model, is to design a function $f_t(\boldsymbol{x}_t)$ that describes the trajectory of the evolving sample, and simultaneously, constitutes a transformation between two given distributions $p_0(\boldsymbol{x}_0)$ and $p_T(\boldsymbol{x}_T)$. In other words, if we randomly sample $\boldsymbol{x}_T$ from $p_T(\boldsymbol{x}_T)$ and evolve it according to the above ODE, we would like to ensure the outcome $\boldsymbol{x}_0$ follows the sample distribution $p_0(\boldsymbol{x}_0)$. The idea from the original paper was very simple, we choose $\boldsymbol{x}_0 \sim p_0(\boldsymbol{x}_0)$ and $\boldsymbol{x}_T \sim p_T(\boldsymbol{x}_T)$ randomly, and assume they transform into each other following the trajectory of

$$\boldsymbol{x}_t = \boldsymbol{\varphi}_t(\boldsymbol{x}_t, \boldsymbol{x}_0). \tag{17.2}$$

Such a trajectory is defined by a known function that we are free to design. Theoretically, it only needs to satisfy the following boundary conditions:

$$\boldsymbol{x}_0 = \boldsymbol{\varphi}_0(\boldsymbol{x}_0, \boldsymbol{x}_T), \qquad \boldsymbol{x}_T = \boldsymbol{\varphi}_T(\boldsymbol{x}_0, \boldsymbol{x}_T). \tag{17.3}$$

while being a continuous function over $[\boldsymbol{x}_0, \boldsymbol{x}_T]$. From here, one can write out the differential equation that the trajectory function must satisfy:

$$\frac{d\boldsymbol{x}_t}{dt} = \frac{d\boldsymbol{\varphi}_t(\boldsymbol{x}_0, \boldsymbol{x}_T)}{dt}. \tag{17.4}$$

However, this differential equation does not have a practical value. This is because we want to generate $\boldsymbol{x}_0$ from a given $\boldsymbol{x}_T$, but the right-hand-side of Eq. (17.4) is also a function of $\boldsymbol{x}_0$, unless we have already known $\boldsymbol{x}_0$, in this case, we would not need a diffusion model in the first place. The ODE will only be practically useful if its right-hand-side is of a similar form as in Eq. (17.1), which is only a function of $\boldsymbol{x}_t$. (Theoretically, we can also make it dependent on $\boldsymbol{x}_T$, but we will not consider this case, in general.) In this case, an intuitive and innovative ideal was formed, that is, to learn a function $\boldsymbol{v}_\theta(\boldsymbol{x}_t, t)$ to regress the right-hand-side of the above ODE. To do this, we optimise the following target:

$$\mathbb{E}_{\boldsymbol{x}_0 \sim p_0(\boldsymbol{x}_0), \boldsymbol{x}_T \sim p_T(\boldsymbol{x}_T)} \left[\left\| \boldsymbol{v}_\theta(\boldsymbol{x}_t, t) - \frac{d\boldsymbol{\varphi}_t(\boldsymbol{x}_0, \boldsymbol{x}_T)}{dt} \right\|_2^2 \right]. \tag{17.5}$$

When $\boldsymbol{v}_\theta(\boldsymbol{x}_t, t)$ is a sufficiently accurate approximation for $\frac{d\boldsymbol{\varphi}_t(\boldsymbol{x}_0, \boldsymbol{x}_T)}{dt}$, we can think Eq. (17.4) will still hold even if we replace its right-hand-side by $\boldsymbol{v}_\theta(\boldsymbol{x}_t, t)$, from which we obtain a practically workable diffusion ODE:

$$\frac{d\boldsymbol{x}_t}{dt} = \boldsymbol{v}_\theta(\boldsymbol{x}_t, t). \tag{17.6}$$

17.2. A simple example

As a simple example, we assume $T = 1$ and the sample evolves along a linear trajectory:

$$\boldsymbol{x}_t = \boldsymbol{\varphi}_t(\boldsymbol{x}_0, \boldsymbol{x}_1) = (\boldsymbol{x}_1 - \boldsymbol{x}_0)t + \boldsymbol{x}_0. \tag{17.7}$$

In this case, we have

$$\frac{\partial \boldsymbol{\varphi}_t(\boldsymbol{x}_0, \boldsymbol{x}_1)}{\partial t} = \boldsymbol{x}_1 - \boldsymbol{x}_0, \tag{17.8}$$

which leads to the following training target

$$\mathbb{E}_{\boldsymbol{x}_0 \sim p_0(\boldsymbol{x}_0), \boldsymbol{x}_T \sim p_T(\boldsymbol{x}_T)} \left[\left\| \boldsymbol{v}_\theta\Big((\boldsymbol{x}_1 - \boldsymbol{x}_0)t + \boldsymbol{x}_0, t\Big) - (\boldsymbol{x}_1 - \boldsymbol{x}_0) \right\|_2^2 \right], \tag{17.9}$$

or equivalently written as

$$\mathbb{E}_{\boldsymbol{x}_0, \boldsymbol{x}_t \sim p(\boldsymbol{x}_0) p_t(\boldsymbol{x}_t | \boldsymbol{x}_0)} \left[\left\| \boldsymbol{v}_\theta(\boldsymbol{x}_t, t) - \frac{\boldsymbol{x}_t - \boldsymbol{x}_0}{t} \right\|_2^2 \right], \tag{17.10}$$

which completes the whole process of constructing a diffusion model. This result is identical to the result of linear trajectory as derived in Section 14, and it is the key model being investigated in the original article, which was dubbed at the "Rectified Flow". From this example

of linear trajectory, it can be seen that the process of constructing an ODE-based diffusion model using this idea only contains a few lines of derivations, which is almost incredible in the sense that it overthrows the perception that the diffusion model is very complicated.

17.3. Further proofs

Nevertheless, the results presented above can only be considered as an intuitive guess, because we have not proven that, theoretically, the outcomes of training the model based on Eq. (17.5), which leads to the ODE shown in Eq. (17.6), will indeed lead to a transformation between the two probability distributions of $p_0(\boldsymbol{x}_0)$ and $p_T(\boldsymbol{x}_T)$. To achieve this goal, the author's initial thought was to prove that the optimum solution for the target distribution from Eq. (17.5) satisfies the following equation of continuity:

$$\frac{\partial p_t(\boldsymbol{x}_t)}{\partial t} = -\nabla_{\boldsymbol{x}_t}\Big(p_t(\boldsymbol{x}_t)\boldsymbol{v}_\theta(\boldsymbol{x}_t, t)\Big). \tag{17.11}$$

If this is the case, then based on the correspondence between ODE and the equation of continuity derived in Section 12. Eq. (17.6) is indeed describing the transformation between two probability distributions of $p_0(\boldsymbol{x}_0)$ and $p_T(\boldsymbol{x}_T)$.

However, such a procedure is a bit cumbersome. Fundamentally, the equation of continuity itself is derived from the ODE through the use of the following relationship:

This is basically another way of writing Eq. (12.3).

$$\mathbb{E}_{\boldsymbol{x}_{t+\Delta t}}\Big[\phi\left(\boldsymbol{x}_{t+\Delta t}\right)\Big] = E_{\boldsymbol{x}_t}\Big[\phi\left(\boldsymbol{x}_t + f_t(\boldsymbol{x}_t)\Delta t\right)\Big]. \tag{17.12}$$

This means that Eq. (17.12) is a more fundamental starting point, for which we only need to prove the optimum solution for Eq. (17.5) satisfies this relationship, which contains $\boldsymbol{x}_t$ as the only independent variable. For this, we write out the following, (in which we have short-handed $\varphi_t(\boldsymbol{x}_0, \boldsymbol{x}_T)$ as φ_t):

$$\begin{aligned}
\mathbb{E}_{\boldsymbol{x}_{t+\Delta t}}\Big[\phi\left(\boldsymbol{x}_{t+\Delta t}\right)\Big] &= \mathbb{E}_{\boldsymbol{x}_0,\boldsymbol{x}_T}\Big[\phi\left(\varphi_{t+\Delta t}\right)\Big] \\
&= \mathbb{E}_{\boldsymbol{x}_0,\boldsymbol{x}_T}\left[\phi(\boldsymbol{x}_t) + \Delta t\frac{\partial \varphi_t}{\partial t}\cdot\nabla_\varphi\phi(\varphi_t)\right] \\
&= \mathbb{E}_{\boldsymbol{x}_0,\boldsymbol{x}_T}\left[\phi(\boldsymbol{x}_t)\right] + \Delta t\mathbb{E}_{\boldsymbol{x}_0,\boldsymbol{x}_T}\left[\frac{\partial \varphi_t}{\partial t}\cdot\nabla_{\boldsymbol{x}_t}\phi(\boldsymbol{x}_t)\right] \\
&= \mathbb{E}_{\boldsymbol{x}_t}\left[\phi(\boldsymbol{x}_t)\right] + \Delta t\mathbb{E}_{\boldsymbol{x}_0,\boldsymbol{x}_T}\left[\frac{\partial \varphi_t}{\partial t}\cdot\nabla_{\boldsymbol{x}_t}\phi(\boldsymbol{x}_t)\right].
\end{aligned} \tag{17.13}$$

Here, the first equality is derived from Eq. (17.2), the second equality comes from the first-order Taylor expansion, and the third equality also comes from Eq. (17.2). Finally, in the last equality, the expectation taken over the variables $\boldsymbol{x}_0$ and $\boldsymbol{x}_T$ is equivalent to the expectation taken over $\boldsymbol{x}_t$, since the latter is a well-defined function of $\boldsymbol{x}_0$ and $\boldsymbol{x}_T$.

Here, we can see that $\frac{\partial \varphi_t}{\partial t}$ is also a function of $\boldsymbol{x}_0$ and $\boldsymbol{x}_T$. We can make a further assumption that the function defined in Eq. (17.2) with respect to $\boldsymbol{x}_T$ is an invertible function, meaning we can solve for $\boldsymbol{x}_T = \psi(\boldsymbol{x}_0, \boldsymbol{x}_t)$ from Eq. (17.2). Substituting this into $\frac{\partial \varphi_t}{\partial t}$ enables us to write the above expectation as a function of $\boldsymbol{x}_0$ and $\boldsymbol{x}_t$. With this, we have,

$$\begin{aligned}
\mathbb{E}_{\boldsymbol{x}_{t+\Delta t}}\Big[\phi\left(\boldsymbol{x}_{t+\Delta t}\right)\Big] &= \mathbb{E}_{\boldsymbol{x}_t}\left[\phi(\boldsymbol{x}_t)\right] + \Delta t\mathbb{E}_{\boldsymbol{x}_0,\boldsymbol{x}_T}\left[\frac{\partial \varphi_t}{\partial t}\cdot\nabla_{\boldsymbol{x}_t}\phi(\boldsymbol{x}_t)\right] \\
&= \mathbb{E}_{\boldsymbol{x}_t}\left[\phi(\boldsymbol{x}_t)\right] + \Delta t\mathbb{E}_{\boldsymbol{x}_0,\boldsymbol{x}_t}\left[\frac{\partial \varphi_t}{\partial t}\cdot\nabla_{\boldsymbol{x}_t}\phi(\boldsymbol{x}_t)\right] \\
&= \mathbb{E}_{\boldsymbol{x}_t}\left[\phi(\boldsymbol{x}_t)\right] + \Delta t\mathbb{E}_{\boldsymbol{x}_t}\left[\underbrace{\mathbb{E}_{\boldsymbol{x}_0|\boldsymbol{x}_t}\left[\frac{\partial \varphi_t}{\partial t}\right]}_{\text{function of } \boldsymbol{x}_t}\cdot\nabla_\varphi\phi(\varphi_t)\right] \\
&= \mathbb{E}_{\boldsymbol{x}_t}\left[\phi\left(\boldsymbol{x}_t + \Delta t\mathbb{E}_{\boldsymbol{x}_0|\boldsymbol{x}_t}\left[\frac{\partial \varphi_t}{\partial t}\right]\right)\right].
\end{aligned} \tag{17.14}$$

The second equation holds because now $\frac{\partial \varphi_t}{\partial t}$ is also a function of $\boldsymbol{x}_0$ and $\boldsymbol{x}_t$, in this case, for the second expectation value, it can be rewritten as being taken over $\boldsymbol{x}_0$ and $\boldsymbol{x}_t$. The third

equality is equivalent to breaking down the joint distribution $p(\boldsymbol{x}_0, \boldsymbol{x}_t) = p(\boldsymbol{x}_0|\boldsymbol{x}_t)p(\boldsymbol{x}_t)$, in this case, it signifies that $\boldsymbol{x}_0$ and $\boldsymbol{x}_t$ are not independent from each other, such that we have to write $\boldsymbol{x}_0|\boldsymbol{x}_t$ to denote their inter-dependency. Note that $\frac{\partial \varphi_t}{\partial t}$ is a function of $\boldsymbol{x}_0$ and $\boldsymbol{x}_t$, such that when we take the expectation value of it over $\boldsymbol{x}_0$, we are left with $\boldsymbol{x}_t$ as the only independent variable. We shall see later that this is the function of $\boldsymbol{x}_t$ that we are looking for! The last equality combines the two terms above with Taylor expansion.

Now we have derived the result

$$\mathbb{E}_{\boldsymbol{x}_{t+\Delta t}}\left[\phi\left(\boldsymbol{x}_{t+\Delta t}\right)\right] = \mathbb{E}_{\boldsymbol{x}_t}\left[\phi\left(\boldsymbol{x}_t + \Delta t \mathbb{E}_{\boldsymbol{x}_0|\boldsymbol{x}_t}\left[\frac{\partial \varphi_t}{\partial t}\right]\right)\right], \tag{17.15}$$

which holds for any testing function ϕ. This implies that

$$\boldsymbol{x}_{t+\Delta t} = \boldsymbol{x}_t + \Delta t \mathbb{E}_{\boldsymbol{x}_0|\boldsymbol{x}_t}\left[\frac{\partial \varphi_t}{\partial t}\right] \quad \Rightarrow \quad \frac{\partial \boldsymbol{x}_t}{\partial t} = \mathbb{E}_{\boldsymbol{x}_0|\boldsymbol{x}_t}\left[\frac{\partial \varphi_t}{\partial t}\right]. \tag{17.16}$$

This is the ODE that we are looking for. According to

$$\mathbb{E}_{\boldsymbol{x}}[\boldsymbol{x}] = \underset{\boldsymbol{\mu}}{\operatorname{argmin}}\, \mathbb{E}_{\boldsymbol{x}}[\|\boldsymbol{x} - \boldsymbol{\mu}\|^2], \tag{17.17}$$

it shows that the right-hand-side of Eq. (17.16) is exactly the optimum solution of the training target given in Eq. (17.5). this proves that the optimum solution for Eq. (17.5), which leads to Eq. (17.6), is indeed providing a transformation between two distributions of $p_0(\boldsymbol{x}_0)$ and $p_T(\boldsymbol{x}_T)$.

18. Score-Matching = Conditional Score-Matching

Original blog post see https://www.spaces.ac.cn/archives/9509

In the previous discussions, we have encountered the terms "*score-matching*" and "*conditional score-matching*" multiple times. These are the concepts that appear frequently in the literature of diffusion and energy models. Particularly in many articles where the training targets for the diffusion models were based on the "score-matching", but for all the mainstreamed diffusion models, such as DDPM, the training target is actually established based on the conditional score-matching.

Therefore, what is the relationship between "*score-matching*" and "*conditional score-matching*"? Are they equivalent to each other? This section will try to answer these questions in details.

18.1. Score-matching

First of all, score-matching refers to the following target function for model training:

$$\mathbb{E}_{\boldsymbol{x}_t \sim p_t(\boldsymbol{x}_t)}\left[\|\nabla_{\boldsymbol{x}_t} \log p_t(\boldsymbol{x}_t) - s_\theta(\boldsymbol{x}_t, t)\|_2^2\right], \tag{18.1}$$

in which θ are the learnable parameters. Obviously, the objective of score-function is to train a model $s_\theta(\boldsymbol{x}_t, t)$ to approximate $\nabla_{\boldsymbol{x}_t} \log p_t(\boldsymbol{x}_t)$, in which we call call the term $\nabla_{\boldsymbol{x}_t} \log p_t(\boldsymbol{x}_t)$ as the *score*.

In the diffusion generative model, $p_t(\boldsymbol{x}_t)$ is given by the following expression:

$$p_t(\boldsymbol{x}_t) = \int p_t(\boldsymbol{x}_t|\boldsymbol{x}_0)p_0(\boldsymbol{x}_0)d\boldsymbol{x}_0 = \mathbb{E}_{\boldsymbol{x}_0 \sim p_0(\boldsymbol{x}_0)}[p_t(\boldsymbol{x}_t|\boldsymbol{x}_0)]. \tag{18.2}$$

Here, $p_t(\boldsymbol{x}_t|\boldsymbol{x}_0)$ is usually constructed as a simple distribution with analytical form (such as a conditional normal distribution). $p_0(\boldsymbol{x}_0)$ is a distribution that is fixed and given, which is usually the distribution of the training data, from which we can sample real data but its analytical form is generally unknown. From Eq. (18.2), we can derive the following:

$$\begin{aligned}\nabla_{\boldsymbol{x}_t} \log p_t(\boldsymbol{x}_t) &= \frac{\nabla_{\boldsymbol{x}_t} p_t(\boldsymbol{x}_t)}{p_t(\boldsymbol{x}_t)} \\ &= \frac{\int \nabla_{\boldsymbol{x}_t} p_t(\boldsymbol{x}_t|\boldsymbol{x}_0)p_0(\boldsymbol{x}_0)d\boldsymbol{x}_0}{\int p_t(\boldsymbol{x}_t|\boldsymbol{x}_0)p_0(\boldsymbol{x}_0)d\boldsymbol{x}_0} \\ &= \frac{\mathbb{E}_{\boldsymbol{x}_0 \sim p_0(\boldsymbol{x}_0)}[\nabla_{\boldsymbol{x}_t} p_t(\boldsymbol{x}_t|\boldsymbol{x}_0)]}{\mathbb{E}_{\boldsymbol{x}_0 \sim p_0(\boldsymbol{x}_0)}[p(\boldsymbol{x}_t|\boldsymbol{x}_0)]}.\end{aligned} \tag{18.3}$$

Based on our assumption, both $\nabla_{\boldsymbol{x}_t} p_t(\boldsymbol{x}_t|\boldsymbol{x}_0)$ and $p_t(\boldsymbol{x}_t|\boldsymbol{x}_0)$ have known analytical expressions, so in principle, we could estimate $\nabla_{\boldsymbol{x}_t} \log p_t(\boldsymbol{x}_t)$ from sampling $\boldsymbol{x}_0$. However, this involves taking the quotient of two expectation values, which leads to a biased estimate in general. As such, one will need a large batch-size when training the model using Eq. (18.1) to overcome the bias in order to reach a better outcome.

18.2. Conditional score-matching

In reality, most diffusion models apply the following conditional score-matching as the training target:

$$\mathbb{E}_{\boldsymbol{x}_0,\boldsymbol{x}_t\sim p_0(\boldsymbol{x}_0)p_t(\boldsymbol{x}_t|\boldsymbol{x}_0)}\Big[||\nabla_{\boldsymbol{x}_t} \log p_t(\boldsymbol{x}_t|\boldsymbol{x}_0) - s_\theta(\boldsymbol{x}_t,t)||_2^2\Big] \tag{18.4}$$

Based on our assumptions, we have an analytical form for $\nabla \boldsymbol{x}_t \log p_t(\boldsymbol{x}_t|\boldsymbol{x}_0)$, so that above training target is practically computable, which requires sampling many pairs of $(\boldsymbol{x}_0, \boldsymbol{x}_t)$ to estimate the target function. More importantly, this is an unbiased estimate, so its value is not strongly dependent upon the batch size, making it more workable in model training.

In order to reveal the relationship between score-matching and conditional score-matching, we will also need another expression for the score function:

$$\begin{aligned}
\nabla_{\boldsymbol{x}_t} \log p_t(\boldsymbol{x}_t) &= \frac{\nabla_{\boldsymbol{x}_t} p_t(\boldsymbol{x}_t)}{p_t(\boldsymbol{x}_t)} \\
&= \frac{\int p_0(\boldsymbol{x}_0)\nabla_{\boldsymbol{x}_t} p_t(\boldsymbol{x}_t|\boldsymbol{x}_0) d\boldsymbol{x}_0}{p_t(\boldsymbol{x}_t)} \\
&= \frac{\int p_0(\boldsymbol{x}_0)p_t(\boldsymbol{x}_t|\boldsymbol{x}_0)\nabla_{\boldsymbol{x}_t} \log p_t(\boldsymbol{x}_t|\boldsymbol{x}_0) d\boldsymbol{x}_0}{p_t(\boldsymbol{x}_t)} \\
&= \int p_t(\boldsymbol{x}_0|\boldsymbol{x}_t)\nabla_{\boldsymbol{x}_t} \log p_t(\boldsymbol{x}_t|\boldsymbol{x}_0) d\boldsymbol{x}_0 \\
&= \mathbb{E}_{\boldsymbol{x}_0\sim p_t(\boldsymbol{x}_0|\boldsymbol{x}_t)}\Big[\nabla_{\boldsymbol{x}_t} \log p_t(\boldsymbol{x}_t|\boldsymbol{x}_0)\Big].
\end{aligned} \tag{18.5}$$

18.3. The inequality relationship

First of all, we can very quickly prove that the conditional score-matching is an upper bound for score-matching, meaning when we minimise the loss of conditional score-matching, we are also minimising the score-matching loss to some extent. This is not difficult to prove, which has already be demonstrated in Section 16, which shall be represented as following:

$$\begin{aligned}
&\mathbb{E}_{\boldsymbol{x}_t\sim p_t(\boldsymbol{x}_t)}\Big[||\nabla_{\boldsymbol{x}_t} \log p_t(\boldsymbol{x}_t) - s_\theta(\boldsymbol{x}_t,t)||_2^2\Big] \\
=\ &\mathbb{E}_{\boldsymbol{x}_t\sim p_t(\boldsymbol{x}_t)}\left[\left\|\mathbb{E}_{\boldsymbol{x}_0\sim p_t(\boldsymbol{x}_0|\boldsymbol{x}_t)}\Big[\nabla_{\boldsymbol{x}_t} \log p_t(\boldsymbol{x}_t|\boldsymbol{x}_0)\Big] - s_\theta(\boldsymbol{x}_t,t)\right\|_2^2\right] \\
\leq\ &\mathbb{E}_{\boldsymbol{x}_t\sim p_t(\boldsymbol{x}_t)}\mathbb{E}_{\boldsymbol{x}_0\sim p_t(\boldsymbol{x}_0|\boldsymbol{x}_t)}\Big[||\nabla_{\boldsymbol{x}_t} \log p_t(\boldsymbol{x}_t|\boldsymbol{x}_0) - s_\theta(\boldsymbol{x}_t,t)||_2^2\Big] \\
=\ &\mathbb{E}_{\boldsymbol{x}_0\sim p_0(\boldsymbol{x}_0),\boldsymbol{x}_t\sim p_t(\boldsymbol{x}_t|\boldsymbol{x}_0)}\Big[||\nabla_{\boldsymbol{x}_t} \log p_t(\boldsymbol{x}_t|\boldsymbol{x}_0) - s_\theta(\boldsymbol{x}_t,t)||_2^2\Big],
\end{aligned} \tag{18.6}$$

in which the first equality comes from Eq. (18.5), the second inequality is derived based on the Jansen's inequality. Finally, the last equality is from the Baye's rule.

18.4. The equality relationship

Here, we shall further prove that the difference between the conditional score-matching and score-matching is a term that is independent from the trainable parameters.

To prove this, we start from score-matching, which we have

$$\begin{aligned}
&\mathbb{E}_{\boldsymbol{x}_t\sim p_t(\boldsymbol{x}_t)}\Big[||\nabla_{\boldsymbol{x}_t} \log p_t(\boldsymbol{x}_t) - s_\theta(\boldsymbol{x}_t,t)||^2\Big] \\
=\ &\mathbb{E}_{\boldsymbol{x}_t\sim p_t(\boldsymbol{x}_t)}\Big[||\nabla_{\boldsymbol{x}_t} \log p_t(\boldsymbol{x}_t)||^2 + ||s_\theta(\boldsymbol{x}_t,t)||^2 - 2s_\theta(\boldsymbol{x}_t,t)\cdot\nabla_{\boldsymbol{x}_t} \log p_t(\boldsymbol{x}_t)\Big].
\end{aligned} \tag{18.7}$$

Whereas for the conditional score-matching, we have

$$\mathbb{E}_{\boldsymbol{x}_0,\boldsymbol{x}_t\sim p_0(\boldsymbol{x}_0)p_t(\boldsymbol{x}_t|\boldsymbol{x}_0)}\Big[||\nabla_{\boldsymbol{x}_t} \log p_t(\boldsymbol{x}_t|\boldsymbol{x}_0) - s_\theta(\boldsymbol{x}_t,t)||^2\Big]$$

$$= \mathbb{E}_{\boldsymbol{x}_0,\boldsymbol{x}_t\sim p_0(\boldsymbol{x}_0)p_t(\boldsymbol{x}_t|\boldsymbol{x}_0)}\Big[||\nabla_{\boldsymbol{x}_t}\log p_t(\boldsymbol{x}_t|\boldsymbol{x}_0)||^2 + ||s_\theta(\boldsymbol{x}_t,t)||^2 - 2s_\theta(\boldsymbol{x}_t,t)\cdot\nabla_{\boldsymbol{x}_t}\log p_t(\boldsymbol{x}_t|\boldsymbol{x}_0)\Big] \tag{18.8}$$

$$= \mathbb{E}_{\boldsymbol{x}_t\sim p_t(\boldsymbol{x}_t),\boldsymbol{x}_0\sim p(\boldsymbol{x}_0|\boldsymbol{x}_t)}\Big[||\nabla_{\boldsymbol{x}_t}\log p_t(\boldsymbol{x}_t|\boldsymbol{x}_0)||^2 + ||s_\theta(\boldsymbol{x}_t,t)||^2 - 2s_\theta(\boldsymbol{x}_t,t)\cdot\nabla_{\boldsymbol{x}_t}\log p_t(\boldsymbol{x}_t|\boldsymbol{x}_0)\Big]$$

$$= \mathbb{E}_{\boldsymbol{x}_t\sim p_t(\boldsymbol{x}_t)}\Big[\mathbb{E}_{\boldsymbol{x}_0\sim p(\boldsymbol{x}_0|\boldsymbol{x}_t)}[||\nabla_{\boldsymbol{x}_t}\log p_t(\boldsymbol{x}_t|\boldsymbol{x}_0)||^2] + ||s_\theta(\boldsymbol{x}_t,t)||^2|| - 2s_\theta(\boldsymbol{x}_t,t)\cdot\mathbb{E}_{\boldsymbol{x}_0\sim p(\boldsymbol{x}_0|\boldsymbol{x}_t)}[||\nabla_{\boldsymbol{x}_t}\log p_t(\boldsymbol{x}_t|\boldsymbol{x}_0)||^2]\Big]$$

$$= \mathbb{E}_{\boldsymbol{x}_t\sim p_t(\boldsymbol{x}_t)}\Big[\mathbb{E}_{\boldsymbol{x}_0\sim p(\boldsymbol{x}_0|\boldsymbol{x}_t)}[||\nabla_{\boldsymbol{x}_t}\log p_t(\boldsymbol{x}_t|\boldsymbol{x}_0)||^2] + ||s_\theta(\boldsymbol{x}_t,t)||^2|| - 2s_\theta(\boldsymbol{x}_t,t)\nabla_{\boldsymbol{x}_t}\log p_t(\boldsymbol{x}_t)\Big].$$

Taking the difference between the two, the result is

$$\mathbb{E}_{\boldsymbol{x}_t\sim p_t(\boldsymbol{x}_t)}\Big[\mathbb{E}_{\boldsymbol{x}_0\sim p(\boldsymbol{x}_0|\boldsymbol{x}_t)}[||\nabla_{\boldsymbol{x}_t}\log p_t(\boldsymbol{x}_t|\boldsymbol{x}_0)||^2] - ||\nabla_{\boldsymbol{x}_t}\log p_t(\boldsymbol{x}_t)||^2\Big], \tag{18.9}$$

which is independent from the training parameters. This means that minimising the target function from conditional score-matching is fully equivalent to minimising the target function from score-matching. This result was first demonstrated in the article "*A Connection between Score Matching and Denoising Autoencoders*"[11].

Since the two approaches are theoretically equivalent to each other, does it mean our previous claim that score-matching requires larger batch-size for training than the conditional score-matching will not hold? This is not true. As estimating the score $\nabla_{\boldsymbol{x}_t}\log p_t(\boldsymbol{x}_t)$ using Eq. (18.3) for evaluating the score-matching loss function will still contain bias which needs to be overcome with the use of large batch-size. However, when we further expanded and simplified the target function [Eq. (18.1)], we have effectively changed the biased estimate into an unbiased one, making it less dependent on the batch size. This is saying, although the two training targets are theoretically equivalent, from a statistical point of view, they are two different statistical quantities and are only exactly equivalent to each other when the sample size approach infinity.

19. GAN and a Diffusion ODE

In Section 16, we have derived an inequality between the score-matching loss and the Wasserstein distance. It was showed that, to a certain extent, the optimisation targets for the diffusion generative models and those for the WGAN are very similar. In this section, we will further discuss the key results from the article "*Monoflow: Rethinking Divergence GANs via the Perspective of Wasserstein Gradient Flow*"[22], which demonstrated another subtle relationship between GAN and the diffusion generative models, whereby *GAN can be considered as the diffusion ODE on another time dimension*!

Original blog post see https://www.spaces.ac.cn/archives/9662

These discoveries show that, although GAN and diffusion models appear to be two completely different generative models, in fact, there exist many similarities between them, and we can borrow ideas from each other to facilitate the development of generative models in the future.

19.1. The basic ideas

As we know, the generator that is trained in a GAN model establishes a one-step deterministic transformation $g_{\boldsymbol{\theta}}(\boldsymbol{z})$ that transforms a noise $\boldsymbol{z}$ into a realistic sample. On the other hand, the distinct character of the diffusion model is its 'progressive generation', which corresponds to a procedure that samples from a series of slowly varying distributions $p_0(\boldsymbol{x}_0)$, $p_1(\boldsymbol{x}_1)$, $\cdots$, $p_T(\boldsymbol{x}_T)$. (Note that, in the diffusion models, we treat $\boldsymbol{x}_T$ as the noise, and $\boldsymbol{x}_0$ being the target sample. But to align with the following discussions, we shall treat them in the reverse order, in which $\boldsymbol{x}_0 \to \boldsymbol{x}_T$ represents a transformation from the noise to the sample.) On the surface, there really is not much similarity between the two models, so how can we link them up together?

Obviously, if we would like to understand GAN from the perspective of the diffusion model, we will need to think of a way to construct a series of continuously varying probability distributions. Although the generator $g_{\boldsymbol{\theta}}(\boldsymbol{z})$ itself is a one-shot transformation, in which the concept of 'slowly varying distributions' does not apply, the optimisation process of the generator is, on the other hand, a slowly varying process! As such, would it be possible to use the trajectory of $\boldsymbol{\theta}$, which shall be denoted as $\boldsymbol{\theta}_t$, to construct these distributions?

More specifically, assume that the generator was initialised with $\boldsymbol{\theta}_0$, after T steps of adversarial training, we shall obtain the optimised parameters $\boldsymbol{\theta}_T$, whereas the parameters along

the training trajectory are $\boldsymbol{\theta}_1$, $\boldsymbol{\theta}_2$, $\cdots$, $\boldsymbol{\theta}_{T-1}$. In this case, if we define $\boldsymbol{x}_t = g_{\boldsymbol{\theta}_t}(\boldsymbol{z})$, then equivalently, we would have correspondingly defined a series of gradually varying $\boldsymbol{x}_0$, $\boldsymbol{x}_1$, $\cdots$, $\boldsymbol{x}_T$, that are drawn from a series of slowly varying distributions $p_0(\boldsymbol{x}_0)$, $p_1(\boldsymbol{x}_1)$, $\cdots$, $p_T(\boldsymbol{x}_T)$.

If this concept is workable, then a GAN model can be effectively interpreted as a diffusion model on the dimension of an imaginary time that corresponds to the descending gradient of the parameters $\boldsymbol{\theta}_t$, and we shall try to investigate this further below.

19.2. The flow of gradient

First of all, we need to use an important result on the Wasserstein gradient flow, which shows that the following differential equation

$$\frac{\partial q_t(\boldsymbol{x})}{\partial t} = -\nabla_{\boldsymbol{x}}\Big(q_t(\boldsymbol{x})\nabla_{\boldsymbol{x}}\log r_t(\boldsymbol{x})\Big), \tag{19.1}$$

is effectively minimising the KL-divergence between $p(\boldsymbol{x})$ and $q_t(\boldsymbol{x})$, *i.e.* $\lim\limits_{t\to\infty} q_t(\boldsymbol{x}) = p(\boldsymbol{x})$, in which $r_t(\boldsymbol{x}) = \frac{p(\boldsymbol{x})}{q_t(\boldsymbol{x})}$. If $p(\boldsymbol{x})$ represents the distribution of the real samples, and furthermore, assuming that we can easily sample from the distributions of $q_t(\boldsymbol{x})$, then, when $t\to\infty$, we would be able to sample from the target distribution of $p(\boldsymbol{x})$. More specifically, sampling from the distribution of $q_t(\boldsymbol{x})$ may be achieved from the following ODE:

$$\frac{d\boldsymbol{x}}{dt} = \nabla_{\boldsymbol{x}}\log r_t(\boldsymbol{x}). \tag{19.2}$$

However, generally, $r_t(\boldsymbol{x})$ in the above ODE is unknown, such that we cannot directly sample $\boldsymbol{x}$ from it. Hence, we must first try to estimate $r_t(\boldsymbol{x})$.

19.3. The discriminator

Now we can come back to examine the discriminator in GAN. Take the earliest vanilla GAN as an example, it has the following training target:

$$\max_D \mathbb{E}_{\boldsymbol{x}\sim p(\boldsymbol{x})}\Big[\log\sigma(D(\boldsymbol{x}))\Big] + \mathbb{E}_{\boldsymbol{x}\sim q(\boldsymbol{x})}\Big[\log\big(1-\sigma(D(\boldsymbol{x}))\big)\Big]. \tag{19.3}$$

Here, D is the discriminator, $\sigma(t) = 1/(1+e^{-t})$ is the sigmoid function, $p(\boldsymbol{x})$ is the distribution of the real samples, and $q(\boldsymbol{x})$ is the distribution of the fake samples. It is possible to prove that, the optimum solution for the discriminator from the above training target is

$$D(\boldsymbol{x}) = \log\frac{p(\boldsymbol{x})}{q(\boldsymbol{x})}. \tag{19.4}$$

For the more generalised f-GAN, the result will be slightly different, but it is possible to prove that the optimum theoretical solution for the discriminator is always a function of $\frac{p(\boldsymbol{x})}{q(\boldsymbol{x})}$. This means that, as long as we could perform sampling from the distributions of $p(\boldsymbol{x})$ and $q_t(\boldsymbol{x})$, then from the trained discriminator of a GAN based on Eq. (19.3), we should be able to estimate $r_t(\boldsymbol{x}) = \frac{p(\boldsymbol{x})}{q_t(\boldsymbol{x})}$.

19.4. One step forward

Some readers may be puzzled, wouldn't the above argument fall into a chicken-and-egg dilemma? On one hand, we want to estimate $r_t(\boldsymbol{x})$ in order to achieve sampling from $q_t(\boldsymbol{x})$ using Eq. (19.2). On the other hand, we assumed that one could sample from $q_t(\boldsymbol{x})$ in order to estimate $r_t(\boldsymbol{x})$. The beauty of this comes as follows: suppose now we have the generator $g_{\boldsymbol{\theta}_t}(\boldsymbol{z})$, its generative outcome is equivalent to sampling from the distribution of $g_t(\boldsymbol{x})$, *i.e.*

$$\Big\{g_{\boldsymbol{\theta}_t}(\boldsymbol{z})\Big|\boldsymbol{z}\sim\mathcal{N}(0,\boldsymbol{I})\Big\} = \Big\{\boldsymbol{x}_t\Big|\boldsymbol{x}_t\sim g_t(\boldsymbol{x})\Big\}. \tag{19.5}$$

This will allow us to compute $r_t(\boldsymbol{x})$ in Eq. (19.3). Pay attention that this is only for $r_t(\boldsymbol{x})$ at time t, and we don't know $r_t(\boldsymbol{x})$ at the other times. So, in principle, we cannot complete the final sampling process directly from Eq. (19.2). Nevertheless, we could push a small step forward:

$$\boldsymbol{x}_{t+1} = \boldsymbol{x}_t + \varepsilon\nabla_{\boldsymbol{x}_t}\log r_t(\boldsymbol{x}_t) = \boldsymbol{x}_t + \varepsilon\nabla_{\boldsymbol{x}_t}D(\boldsymbol{x}_t), \tag{19.6}$$

in which ε is an infinitesimal positive number that represents the step size. Now we have the outcome from sampling the next step, which we want it also to be equivalent to the sampling outcome from the generator in the next step, *i.e.*

$$\begin{aligned}\Big\{g_{\boldsymbol{\theta}_{t+1}}(\boldsymbol{z})\Big|\boldsymbol{z}\sim\mathcal{N}(0,\boldsymbol{I})\Big\} &= \Big\{\boldsymbol{x}_{t+1}\Big|\boldsymbol{x}_{t+1}\sim g_{t+1}(\boldsymbol{x})\Big\}\\ &= \Big\{\boldsymbol{x}_{t+1}\Big|\boldsymbol{x}_t+\varepsilon\nabla_{\boldsymbol{x}_t}D(\boldsymbol{x}_t),\boldsymbol{x}_t\sim q_t(\boldsymbol{x})\Big\}.\end{aligned}\tag{19.7}$$

In other words, we want to change the motion of the samples in the diffusion model into the motion of the parameters in the generator of GAN. To achieve this, we train the following loss to optimise $\boldsymbol{\theta}_{t+1}$:

$$\boldsymbol{\theta}_{t+1} = \underset{\boldsymbol{\theta}}{\operatorname{argmin}}\,\mathbb{E}_{\boldsymbol{z}\sim\mathcal{N}(0,\boldsymbol{I})}\Big[\|g_{\boldsymbol{\theta}}(\boldsymbol{z}) - g_{\boldsymbol{\theta}_t}(\boldsymbol{z}) - \varepsilon\nabla_g D(g_{\boldsymbol{\theta}_t}(\boldsymbol{z}))\|_2^2\Big].\tag{19.8}$$

This is saying that, we first increment $\boldsymbol{x}_t = g_{\boldsymbol{\theta}_t}(\boldsymbol{z})$ forward to reach $\boldsymbol{x}_{t+1}$, and hope that the corresponding outcome from $g_{\boldsymbol{\theta}_{t+1}}(\boldsymbol{z})$ is also sufficiently close to $\boldsymbol{x}_{t+1}$. After this, we start the new iteration by replacing $\boldsymbol{\theta}_t$ with the new $\boldsymbol{\theta}_{t+1}$, meaning that we iterate through Eq. (19.3) and Eq. (19.8) repetitively. Doesn't this look very similar to GAN?

19.5. The final touch

If the above derivations are not sufficient, we could further improve it to make it become closer to the GAN framework. Noticing that the gradient of the function, over which the expectation value is taken in Eq. (19.8), can be computed as

$$\begin{aligned}&\nabla_{\boldsymbol{\theta}}\Big\|g_{\boldsymbol{\theta}}(\boldsymbol{z}) - g_{\boldsymbol{\theta}_t}(\boldsymbol{z}) - \varepsilon\nabla_g D(g_{\boldsymbol{\theta}_t}(\boldsymbol{z}))\Big\|_2^2\\ &= 2\Big\langle g_{\boldsymbol{\theta}}(\boldsymbol{z}) - g_{\boldsymbol{\theta}_t}(\boldsymbol{z}) - \varepsilon\nabla_g D(g_{\boldsymbol{\theta}_t}(\boldsymbol{z})), \nabla_{\boldsymbol{\theta}} g_{\boldsymbol{\theta}}(\boldsymbol{z})\Big\rangle.\end{aligned}\tag{19.9}$$

Substituting in the current value of $\boldsymbol{\theta}_t$, the result is

$$-2\varepsilon\Big\langle\nabla_g D(g_{\boldsymbol{\theta}_t}(\boldsymbol{z})), \nabla_{\boldsymbol{\theta}_t} g_{\boldsymbol{\theta}_t}(\boldsymbol{z})\Big\rangle = -2\varepsilon\nabla_{\boldsymbol{\theta}_t} D(g_{\boldsymbol{\theta}_t}(\boldsymbol{z})).\tag{19.10}$$

This is saying that, if we only perform one step of optimisation based on the gradient of the loss function, then the loss function based on Eq. (19.8) is fully equivalent to the following loss function (since the gradient only differs by a constant coefficient):

$$\boldsymbol{\theta}_{t+1} = \underset{\boldsymbol{\theta}}{\operatorname{argmin}}\,\mathbb{E}_{\boldsymbol{z}\sim\mathcal{N}(0,\boldsymbol{I})}\Big[-D(g_{\boldsymbol{\theta}}(\boldsymbol{z}))\Big].\tag{19.11}$$

This is one of the common loss functions for training the generator. By training the model through minimising Eq. (19.3) and **??** alternatively, it leads to a common variant of GAN. More specifically,in the original article, a more generic form for the loss function for training the generator was also given:

$$\boldsymbol{\theta}_{t+1} = \underset{\boldsymbol{\theta}}{\operatorname{argmin}}\,\mathbb{E}_{\boldsymbol{z}\sim\mathcal{N}(0,\boldsymbol{I})}\Big[-h\Big(D(g_{\boldsymbol{\theta}}(\boldsymbol{z}))\Big)\Big],\tag{19.12}$$

in which $h(\cdot)$ is any arbitrary monotonically increasing function. Correspondingly, in the Wasserstein gradient flow [Eq. (19.1)], we can change $\log r_t(\boldsymbol{x})$ into $h(\log r_t(\boldsymbol{x}))$. This is probably the origin behind the name *monoflow*. As for proving Eq. (19.12), we shall not further discuss the details here. Interested readers may refer back to the original article.

19.6. The significance behind this work

In summary, to treat GAN as the diffusion model, we consider the following process:

$$\cdots\to g_{\boldsymbol{\theta}_t}(\boldsymbol{z})\xrightarrow{\text{Eq. (19.3)}} r_t(\boldsymbol{x})\xrightarrow{\text{Eq. (19.6)}}\boldsymbol{x}_{t+1}\xrightarrow{\text{Eq. (19.8)}} g_{\boldsymbol{\theta}_{t+1}}(\boldsymbol{z})\to\cdots\tag{19.13}$$

in which the key equation is Eq. (19.6), which comes from Eq. (19.1) and Eq. (19.2) based on the Wasserstein gradient flow.

Some readers may ask, why we need to spend so much effort to understand GAN from this perspective if it has not led to something fundamentally new to the original GAN model?

First of all, the author believes that, understanding GAN from the perspective of the diffusion generative model, or unifying GAN and the diffusion models, is an interesting and fun endeavour, which is its most important significance, even though this may not lead to any practical significance.

Secondly, as the authors have already pointed out in their original paper, the theoretical derivations and the actual model training process for GAN are not fully consistent with each other. This is no longer the case if we try to understand GAN from the perspective of the diffusion model, which leads to a training process that is conceptually consistent with the model design. In other words, from the perspective of model training, the existing derivations for GAN were wrong, the right perspective is to treat it fundamentally similar to the diffusion model.

How should we understand this? Taking the GAN model above as an example, the training targets for the discriminator and the generator are:

$$\max_D \mathbb{E}_{\boldsymbol{x}\sim p(\boldsymbol{x})}\Big[\log \sigma(D(\boldsymbol{x}))\Big] + \mathbb{E}_{\boldsymbol{x}\sim q(\boldsymbol{x})}\Big[\log(1-\sigma(D(\boldsymbol{x})))\Big] \tag{19.14}$$

$$\min_q \mathbb{E}_{\boldsymbol{x}\sim q(\boldsymbol{x})}\Big[-D(\boldsymbol{x})\Big] \tag{19.15}$$

The nature way to prove these, is to prove that the optimum solution for D is $\log \frac{p(\boldsymbol{x})}{q(\boldsymbol{x})}$, and substitute this result into the loss function of the generator, which shows that effectively, it is minimising the KL divergence between p and q, such that the optimum solution is naturally $p(\boldsymbol{x}) = q(\boldsymbol{x})$. However, the corresponding training procedure for this should be to first solve the $\max_D$ problem for any arbitrary $q(\boldsymbol{x})$, which gives D as a function of $q(\boldsymbol{x})$ or the parameters $\boldsymbol{\theta}$ for the generator, from which we proceed to the $\min_q$ step. This is different from the alternating training steps that switch periodically between the generator and discriminator. The latter is more consistent with the design of the diffusion model, which itself is alternative in nature, this leads more naturally to the model training process for GAN.

Overall, understanding GAN from the perspective of the diffusion model is not only just a new way to understand GAN, but more importantly, a perspective that is more consistent with the model training process for GAN. For instance, we can now explain why training the GAN generator should not take too many steps, this is because Eq. (19.8) and Eq. (19.11) are equivalent to each other only when we take the single-shot optimisation, otherwise, one should really be using Eq. (19.8) as the training target.

References

[1] Y. Lipman, R. T. Chen, H. Ben-Hamu, M. Nickel, and M. Le, "Flow matching for generative modeling," arXiv preprint arXiv:2210.02747 (2022).

[2] A. Razavi, A. Van den Oord, and O. Vinyals, "Generating diverse high-fidelity images with VQ-VAE-2," Adv. Neural Info. Proc. Sys. **32** (2019).

[3] J. Ho, A. Jain, and P. Abbeel, "Denoising diffusion probabilistic models," Adv. Neural Info. Proc. Sys. **33**, 6840 (2020).

[4] J. Sohl-Dickstein, E. Weiss, N. Maheswaranathan, and S. Ganguli, "Deep unsupervised learning using nonequilibrium thermodynamics," in *International conference on machine learning* (pmlr, 2015) p. 2256.

[5] C. Luo, "Understanding diffusion models: A unified perspective," arXiv:2208.11970 (2022).

[6] 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).

[7] J. Song, C. Meng, and S. Ermon, "Denoising diffusion implicit models," arXiv:2010.02502 (2020).

[8] 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).

[9] B. D. Anderson, "Reverse-time diffusion equation models," Stochastic Processes and their Applications **12**, 313 (1982).

[10] A. Hyvärinen and P. Dayan, "Estimation of non-normalized statistical models by score matching." J. Mach. Learn. Res. **6** (2005).

[11] P. Vincent, "A connection between score matching and denoising autoencoders," Neural Comput. **23**, 1661 (2011).

[12] A. Q. Nichol and P. Dhariwal, "Improved denoising diffusion probabilistic models," in *Inter. Conf. Machine Learning* (PMLR, 2021) p. 8162.

[13] 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).

[14] 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).

[15] P. Dhariwal and A. Nichol, "Diffusion models beat GANs on image synthesis," Adv. Neural Info. Proc. Sys. **34**, 8780 (2021).

[16] 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.

[17] J. Ho and T. Salimans, "Classifier-free diffusion guidance," arXiv:2207.12598 (2022).

[18] 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).

[19] Y. Xu, Z. Liu, M. Tegmark, and T. Jaakkola, "Poisson flow generative models," Adv. Neural Info. Proc. Sys. **35**, 16782 (2022).

[20] D. Kwon, Y. Fan, and K. Lee, "Score-based generative modeling secretly minimizes the wasserstein distance," Adv. Neu. Info. Proc. Sys. **35**, 20205 (2022).

[21] X. Liu, C. Gong, and Q. Liu, "Flow straight and fast: Learning to generate and transfer data with rectified flow," arXiv:2209.03003 (2022).

[22] M. Yi, Z. Zhu, and S. Liu, "Monoflow: Rethinking divergence gans via the perspective of differential equations," in *Inter. Conf. Mach. Learn.* (Proceedings of Machine Learning Research, 2024) p. 39984.