

Tutorial on Diffusion Models for Imaging and Vision

Stanley Chan¹

January 9, 2025

Abstract. The astonishing growth of generative tools in recent years has empowered many exciting applications in text-to-image generation and text-to-video generation. The underlying principle behind these generative tools is the concept of *diffusion*, a particular sampling mechanism that has overcome some longstanding shortcomings in previous approaches. The goal of this tutorial is to discuss the essential ideas underlying these diffusion models. The target audience of this tutorial includes undergraduate and graduate students who are interested in doing research on diffusion models or applying these tools to solve other problems.

Contents

1 Variational Auto-Encoder (VAE)	2
1.1 Building Blocks of VAE	2
1.2 Evidence Lower Bound	5
1.3 Optimization in VAE	10
1.4 Concluding Remark	17
2 Denoising Diffusion Probabilistic Model (DDPM)	18
2.1 Building Blocks	19
2.2 Evidence Lower Bound	25
2.3 Distribution of the Reverse Process	30
2.4 Training and Inference	34
2.5 Predicting Noise	37
2.6 Denoising Diffusion Implicit Model (DDIM)	39
2.7 Concluding Remark	43
3 Score-Matching Langevin Dynamics (SMLD)	44
3.1 Sampling from a Distribution	44
3.2 (Stein's) Score Function	48
3.3 Score Matching Techniques	49
3.4 Concluding Remark	53
4 Stochastic Differential Equation (SDE)	54
4.1 From Iterative Algorithms to Ordinary Differential Equations	54
4.2 What is an SDE?	56
4.3 Stochastic Differential Equation for DDPM and SMLD	58
4.4 Numerical Solvers for ODE and SDE	63
4.5 Concluding Remark	64
5 Langevin and Fokker-Planck Equations	66
5.1 Brownian Motion	66
5.2 Masters Equation	73
5.3 Kramers-Moyal Expansion	77
5.4 Fokker-Planck Equation	81
5.5 Concluding Remark	86
6 Conclusion	87

¹School of Electrical and Computer Engineering, Purdue University, West Lafayette, IN 47907.
Email: stanchan@purdue.edu.

1 Variational Auto-Encoder (VAE)

A long time ago, in a galaxy far far away, we wanted to build a generator — a generator that generates texts, speeches, or images from some inputs with which we give to the computer. While this may sound magical at first, the problem has actually been studied for a long time. To kick off the discussion of this tutorial, we shall first consider the **variational autoencoder** (VAE). VAE was proposed by Kingma and Welling in 2014 [23]. According to their 2019 tutorial [24], the VAE was inspired by the Helmholtz Machine [10] as the marriage of graphical models and deep learning. In what follows, we will discuss VAE’s problem setting, its building blocks, and the optimization tools associated with the training.

1.1 Building Blocks of VAE

We start by discussing the schematic diagram of a VAE. As shown in the figure below, the VAE consists of a pair of models (often realized by deep neural networks). The one located near the input is called an **encoder** whereas the one located near the output is called a **decoder**. We denote the input (typically an image) as a vector \mathbf{x} , and the output (typically another image) as a vector $\hat{\mathbf{x}}$. The vector located in the middle between the encoder and the decoder is called a **latent variable**, denoted as \mathbf{z} . The job of the encoder is to extract a meaningful representation for \mathbf{x} , whereas the job of the decoder is to generate a new image from the latent variable \mathbf{z} .

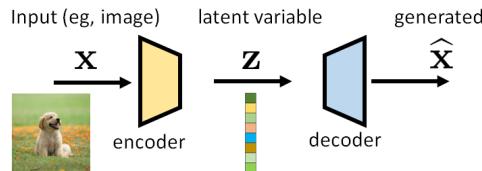


Figure 1.1: A variational autoencoder consists of an encoder that converts an input \mathbf{x} to a latent variable \mathbf{z} , and a decoder that synthesizes an output $\hat{\mathbf{x}}$ from the latent variable.

The latent variable \mathbf{z} has two special roles in this setup. With respect to the input, the latent variable encapsulates the information that can be used to describe \mathbf{x} . The encoding procedure could be a lossy process, but our goal is to preserve the important content of \mathbf{x} as much as we can. With respect to the output, the latent variable serves as the “seed” from which an image $\hat{\mathbf{x}}$ can be generated. Two different \mathbf{z} ’s should in theory give us two different generated images.

A slightly more formal definition of a latent variable is given below.

Definition 1.1. Latent Variables[24]. In a probabilistic model, latent variables \mathbf{z} are variables that we do not observe and hence are not part of the training dataset, although they are part of the model.

Example 1.1. Getting a latent representation of an image is not an alien thing. Back in the time of JPEG compression (which is arguably a dinosaur), we used discrete cosine transform (DCT) basis functions φ_n to encode the underlying image/patches of an image. The coefficient vector $\mathbf{z} = [z_1, \dots, z_N]^T$ is obtained by projecting the image \mathbf{x} onto the space spanned by the basis, via $z_n = \langle \varphi_n, \mathbf{x} \rangle$. So, given an image \mathbf{x} , we can produce a coefficient vector \mathbf{z} . From \mathbf{z} , we can use the inverse transform to recover (i.e. decode) the image.

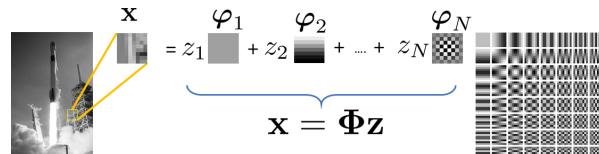


Figure 1.2: In discrete cosine transform (DCT), we can think of the encoder as taking an image \mathbf{x} and generating a latent variable \mathbf{z} by projecting \mathbf{x} onto the basis functions.

In this example, the coefficient vector \mathbf{z} is the latent variable. The encoder is the DCT transform, and the decoder is the inverse DCT transform.

The term “variational” in VAE is related to the subject of calculus of variations which studies optimization over functions. In VAE, we are interested in searching for the optimal probability distributions to describe \mathbf{x} and \mathbf{z} . In light of this, we need to consider a few distributions:

- $p(\mathbf{x})$: The true distribution of \mathbf{x} . It is never known. The whole universe of diffusion models is to find ways to draw samples from $p(\mathbf{x})$. If we knew $p(\mathbf{x})$ (say, we have a formula that describes $p(\mathbf{x})$), we can just draw a sample \mathbf{x} that maximizes $\log p(\mathbf{x})$.
- $p(\mathbf{z})$: The distribution of the latent variable. Typically, we make it a zero-mean unit-variance Gaussian $\mathcal{N}(0, \mathbf{I})$. One reason is that linear transformation of a Gaussian remains a Gaussian, and so this makes the data processing easier. Doersch [12] also has an excellent explanation. It was mentioned that any distribution can be generated by mapping a Gaussian through a sufficiently complicated function. For example, in a one-variable setting, the inverse cumulative distribution function (CDF) technique [7, Chapter 4] can be used for any continuous distribution with an invertible CDF. In general, as long as we have a sufficiently powerful function (e.g., a neural network), we can learn it and map the i.i.d. Gaussian to whatever latent variable needed for our problem.
- $p(\mathbf{z}|\mathbf{x})$: The conditional distribution associated with the **encoder**, which tells us the likelihood of \mathbf{z} when given \mathbf{x} . We have no access to it. $p(\mathbf{z}|\mathbf{x})$ itself is *not* the encoder, but the encoder has to do something so that it will behave consistently with $p(\mathbf{z}|\mathbf{x})$.
- $p(\mathbf{x}|\mathbf{z})$: The conditional distribution associated with the **decoder**, which tells us the posterior probability of getting \mathbf{x} given \mathbf{z} . Again, we have no access to it.

When we switch from the classical parameteric models to deep neural networks, the notion of latent variables is changed to *deep* latent variables. Kingma and Welling [24] gave a good definition below.

Definition 1.2. Deep Latent Variables[24]. Deep Latent Variables are latent variables whose distributions $p(\mathbf{z})$, $p(\mathbf{x}|\mathbf{z})$, or $p(\mathbf{z}|\mathbf{x})$ are parameterized by a neural network.

The advantage of deep latent variables is that they can model very complex data distributions $p(\mathbf{x})$ even though the structures of the prior distributions and the conditional distributions are relatively simple (e.g. Gaussian). One way to think about this is that the neural networks can be used to estimate the mean of a Gaussian. Although the Gaussian itself is simple, the mean is a function of the input data, which passes through a neural network to generate a data-dependent mean. So the expressiveness of the Gaussian is significantly improved.

Let's go back to the four distributions above. Here is a somewhat trivial but educational example that can illustrate the idea:

Example 1.2. Consider a random variable \mathbf{X} distributed according to a Gaussian mixture model with a latent variable $z \in \{1, \dots, K\}$ denoting the cluster identity such that $p_Z(k) = \mathbb{P}[Z = k] = \pi_k$ for $k = 1, \dots, K$. We assume $\sum_{k=1}^K \pi_k = 1$. Then, if we are told that we need to look at the k -th cluster only, the conditional distribution of \mathbf{X} given Z is

$$p_{\mathbf{X}|Z}(\mathbf{x}|k) = \mathcal{N}(\mathbf{x} | \boldsymbol{\mu}_k, \sigma_k^2 \mathbf{I}).$$

The marginal distribution of \mathbf{x} can be found using the law of total probability, giving us

$$p_{\mathbf{X}}(\mathbf{x}) = \sum_{k=1}^K p_{\mathbf{X}|Z}(\mathbf{x}|k)p_Z(k) = \sum_{k=1}^K \pi_k \mathcal{N}(\mathbf{x} | \boldsymbol{\mu}_k, \sigma_k^2 \mathbf{I}). \quad (1.1)$$

Therefore, if we start with $p_{\mathbf{X}}(\mathbf{x})$, the design question for the encoder is to build a magical encoder such that for every sample $\mathbf{x} \sim p_{\mathbf{X}}(\mathbf{x})$, the latent code will be $z \in \{1, \dots, K\}$ with a distribution $z \sim p_Z(k)$.

To illustrate how the encoder and decoder work, let's assume that the mean and variance are known and are fixed. Otherwise we will need to estimate the mean and variance through an expectation-

maximization (EM) algorithm. It is doable, but the tedious equations will defeat the educational purpose of this illustration.

Encoder: How do we obtain z from \mathbf{x} ? This is easy because at the encoder, we know $p_{\mathbf{X}}(\mathbf{x})$ and $p_Z(k)$. Imagine that you only have two classes $z \in \{1, 2\}$. Effectively you are just making a binary decision of where the sample \mathbf{x} should belong to. There are many ways you can do the binary decision. If you like the maximum-a-posteriori decision rule, you can check

$$p_{Z|\mathbf{X}}(1|\mathbf{x}) \gtrless_{\text{class } 2}^{\text{class } 1} p_{Z|\mathbf{X}}(2|\mathbf{x}),$$

and this will return you a simple decision: You give us \mathbf{x} , we tell you $z \in \{1, 2\}$.

Decoder: On the decoder side, if we are given a latent code $z \in \{1, \dots, K\}$, the magical decoder just needs to return us a sample \mathbf{x} which is drawn from $p_{\mathbf{X}|Z}(\mathbf{x}|k) = \mathcal{N}(\mathbf{x} | \boldsymbol{\mu}_k, \sigma_k^2 \mathbf{I})$. A different z will give us one of the K mixture components. If we have enough samples, the overall distribution will follow the Gaussian mixture.

This example is certainly oversimplified because real-world problems can be much harder than a Gaussian mixture model with known means and known variances. But one thing we realize is that if we want to find the magical encoder and decoder, we must have a way to find the two conditional distributions $p(\mathbf{z}|\mathbf{x})$ and $p(\mathbf{x}|\mathbf{z})$. However, they are both high-dimensional.

In order for us to say something more meaningful, we need to impose additional structures so that we can generalize the concept to harder problems. To this end, we consider the following two proxy distributions:

- $q_\phi(\mathbf{z}|\mathbf{x})$: The proxy for $p(\mathbf{z}|\mathbf{x})$, which is also the distribution associated with the *encoder*. $q_\phi(\mathbf{z}|\mathbf{x})$ can be any directed graphical model and it can be parameterized using deep neural networks [24, Section 2.1]. For example, we can define

$$\begin{aligned} (\boldsymbol{\mu}, \boldsymbol{\sigma}^2) &= \text{EncoderNetwork}_\phi(\mathbf{x}), \\ q_\phi(\mathbf{z}|\mathbf{x}) &= \mathcal{N}(\mathbf{z} | \boldsymbol{\mu}, \text{diag}(\boldsymbol{\sigma}^2)). \end{aligned} \quad (1.2)$$

This model is a widely used because of its tractability and computational efficiency.

- $p_\theta(\mathbf{x}|\mathbf{z})$: The proxy for $p(\mathbf{x}|\mathbf{z})$, which is also the distribution associated with the *decoder*. Like the encoder, the decoder can be parameterized by a deep neural network. For example, we can define

$$\begin{aligned} f_\theta(\mathbf{z}) &= \text{DecoderNetwork}_\theta(\mathbf{z}), \\ p_\theta(\mathbf{x}|\mathbf{z}) &= \mathcal{N}(\mathbf{x} | f_\theta(\mathbf{z}), \sigma_{\text{dec}}^2 \mathbf{I}), \end{aligned} \quad (1.3)$$

where σ_{dec} is a hyperparameter that can be pre-determined or it can be learned.

The relationship between the input \mathbf{x} and the latent \mathbf{z} , as well as the conditional distributions, are summarized in Figure 1.3. There are two nodes \mathbf{x} and \mathbf{z} . The “forward” relationship is specified by $p(\mathbf{z}|\mathbf{x})$ (and approximated by $q_\phi(\mathbf{z}|\mathbf{x})$), whereas the “reverse” relationship is specified by $p(\mathbf{x}|\mathbf{z})$ (and approximated by $p_\theta(\mathbf{x}|\mathbf{z})$).

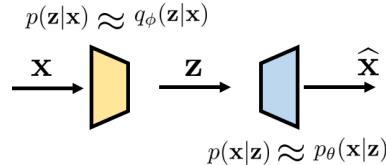


Figure 1.3: In a variational autoencoder, the variables \mathbf{x} and \mathbf{z} are connected by the conditional distributions $p(\mathbf{x}|\mathbf{z})$ and $p(\mathbf{z}|\mathbf{x})$. To make things work, we introduce proxy distributions $p_\theta(\mathbf{x}|\mathbf{z})$ and $q_\phi(\mathbf{z}|\mathbf{x})$.

Example 1.3. Suppose that we have a random variable $\mathbf{x} \in \mathbb{R}^d$ and a latent variable $\mathbf{z} \in \mathbb{R}^d$ such that

$$\begin{aligned} \mathbf{x} &\sim p(\mathbf{x}) = \mathcal{N}(\mathbf{x} | \boldsymbol{\mu}, \sigma^2 \mathbf{I}), \\ \mathbf{z} &\sim p(\mathbf{z}) = \mathcal{N}(\mathbf{z} | 0, \mathbf{I}). \end{aligned}$$

We want to construct a VAE. By this, we mean that we want to build two mappings $\text{Encoder}(\cdot)$ and $\text{Decoder}(\cdot)$. The encoder will take a sample \mathbf{x} and map it to the latent variable \mathbf{z} , whereas the decoder will take the latent variable \mathbf{z} and map it to the generated variable $\hat{\mathbf{x}}$. If we knew what $p(\mathbf{x})$ is, then there is a trivial solution where $\mathbf{z} = (\mathbf{x} - \boldsymbol{\mu})/\sigma$ and $\hat{\mathbf{x}} = \boldsymbol{\mu} + \sigma\mathbf{z}$. In this case, the true distributions can be determined and they can be expressed in terms of delta functions:

$$\begin{aligned} p(\mathbf{x}|\mathbf{z}) &= \delta(\mathbf{x} - (\sigma\mathbf{z} + \boldsymbol{\mu})), \\ p(\mathbf{z}|\mathbf{x}) &= \delta(\mathbf{z} - (\mathbf{x} - \boldsymbol{\mu})/\sigma). \end{aligned}$$

Suppose now that we do not know $p(\mathbf{x})$ so we need to build an encoder and a decoder to estimate \mathbf{z} and $\hat{\mathbf{x}}$. Let's first define the encoder. Our encoder in this example takes the input \mathbf{x} and generates a pair of parameters $\hat{\boldsymbol{\mu}}(\mathbf{x})$ and $\hat{\sigma}(\mathbf{x})^2$, denoting the parameters of a Gaussian. Then, we define $q_\phi(\mathbf{z}|\mathbf{x})$ as a Gaussian:

$$\begin{aligned} (\hat{\boldsymbol{\mu}}(\mathbf{x}), \hat{\sigma}(\mathbf{x})^2) &= \text{Encoder}_\phi(\mathbf{x}), \\ q_\phi(\mathbf{z}|\mathbf{x}) &= \mathcal{N}(\mathbf{z} | \hat{\boldsymbol{\mu}}(\mathbf{x}), \hat{\sigma}(\mathbf{x})^2 \mathbf{I}). \end{aligned}$$

For the purpose of discussion, we assume that $\hat{\boldsymbol{\mu}}$ is an affine function of \mathbf{x} such that $\hat{\boldsymbol{\mu}}(\mathbf{x}) = a\mathbf{x} + \mathbf{b}$ for some parameters a and \mathbf{b} . Similarly, we assume that $\hat{\sigma}(\mathbf{x})^2 = t^2$ for some scalar t . This will give us

$$q_\phi(\mathbf{z}|\mathbf{x}) = \mathcal{N}(\mathbf{z} | a\mathbf{x} + \mathbf{b}, t^2 \mathbf{I}).$$

For the decoder, we deploy a similar structure by considering

$$\begin{aligned} (\tilde{\boldsymbol{\mu}}(\mathbf{z}), \tilde{\sigma}(\mathbf{z})^2) &= \text{Decoder}_\theta(\mathbf{z}), \\ p_\theta(\mathbf{x}|\mathbf{z}) &= \mathcal{N}(\mathbf{x} | \tilde{\boldsymbol{\mu}}(\mathbf{z}), \tilde{\sigma}(\mathbf{z})^2 \mathbf{I}). \end{aligned}$$

Again, for the purpose of discussion, we assume that $\tilde{\boldsymbol{\mu}}$ is affine so that $\tilde{\boldsymbol{\mu}}(\mathbf{z}) = c\mathbf{z} + \mathbf{v}$ for some parameters c and \mathbf{v} and $\tilde{\sigma}(\mathbf{z})^2 = s^2$ for some scalar s . Therefore, $p_\theta(\mathbf{x}|\mathbf{z})$ takes the form of:

$$p_\theta(\mathbf{x}|\mathbf{z}) = \mathcal{N}(\mathbf{z} | c\mathbf{z} + \mathbf{v}, s^2 \mathbf{I}).$$

We will discuss how to determine the parameters later.

1.2 Evidence Lower Bound

How do we use these two proxy distributions to achieve our goal of determining the encoder and the decoder? If we treat ϕ and θ as optimization variables, then we need an objective function (or the loss function) so that we can optimize ϕ and θ through training samples. The loss function we use here is called the Evidence Lower BOund (ELBO) [24]:

Definition 1.3. (Evidence Lower Bound) The Evidence Lower Bound is defined as

$$\text{ELBO}(\mathbf{x}) \stackrel{\text{def}}{=} \mathbb{E}_{q_\phi(\mathbf{z}|\mathbf{x})} \left[\log \frac{p(\mathbf{x}, \mathbf{z})}{q_\phi(\mathbf{z}|\mathbf{x})} \right]. \quad (1.4)$$

You are certainly puzzled how on the Earth people can come up with this loss function!? Let's see what ELBO means and how it is derived.

In a nutshell, ELBO is a **lower bound** for the prior distribution $\log p(\mathbf{x})$ because we can show that

$$\begin{aligned}
 \log p(\mathbf{x}) &= \text{some magical steps to be derived} \\
 &= \mathbb{E}_{q_\phi(\mathbf{z}|\mathbf{x})} \left[\log \frac{p(\mathbf{x}, \mathbf{z})}{q_\phi(\mathbf{z}|\mathbf{x})} \right] + \mathbb{D}_{\text{KL}}(q_\phi(\mathbf{z}|\mathbf{x}) \| p(\mathbf{z}|\mathbf{x})) \\
 &\geq \mathbb{E}_{q_\phi(\mathbf{z}|\mathbf{x})} \left[\log \frac{p(\mathbf{x}, \mathbf{z})}{q_\phi(\mathbf{z}|\mathbf{x})} \right] \\
 &\stackrel{\text{def}}{=} \text{ELBO}(\mathbf{x}),
 \end{aligned} \tag{1.5}$$

where the inequality follows from the fact that the KL divergence is always non-negative. Therefore, ELBO is a valid lower bound for $\log p(\mathbf{x})$. Since we never have access to $\log p(\mathbf{x})$, if we somehow have access to ELBO and if ELBO is a good lower bound, then we can effectively maximize ELBO to achieve the goal of maximizing $\log p(\mathbf{x})$ which is the gold standard. Now, the question is how good the lower bound is. As you can see from the equation and also Figure 1.4, the inequality will become an equality when our proxy $q_\phi(\mathbf{z}|\mathbf{x})$ can match the true distribution $p(\mathbf{z}|\mathbf{x})$ exactly. So, part of the game is to ensure $q_\phi(\mathbf{z}|\mathbf{x})$ is close to $p(\mathbf{z}|\mathbf{x})$.

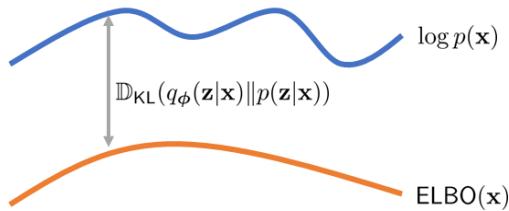


Figure 1.4: Visualization of $\log p(\mathbf{x})$ and ELBO. The gap between the two is determined by the KL divergence $\mathbb{D}_{\text{KL}}(q_\phi(\mathbf{z}|\mathbf{x}) \| p(\mathbf{z}|\mathbf{x}))$.

The derivation of Eqn (1.5) is as follows.

Theorem 1.1. Decomposition of Log-Likelihood. The log likelihood $\log p(\mathbf{x})$ can be decomposed as

$$\log p(\mathbf{x}) = \underbrace{\mathbb{E}_{q_\phi(\mathbf{z}|\mathbf{x})} \left[\log \frac{p(\mathbf{x}, \mathbf{z})}{q_\phi(\mathbf{z}|\mathbf{x})} \right]}_{\stackrel{\text{def}}{=} \text{ELBO}(\mathbf{x})} + \mathbb{D}_{\text{KL}}(q_\phi(\mathbf{z}|\mathbf{x}) \| p(\mathbf{z}|\mathbf{x})). \tag{1.6}$$

Proof. The trick is to use our magical proxy $q_\phi(\mathbf{z}|\mathbf{x})$ to poke around $p(\mathbf{x})$ and derive the bound.

$$\begin{aligned}
 \log p(\mathbf{x}) &= \log p(\mathbf{x}) \times \underbrace{\int q_\phi(\mathbf{z}|\mathbf{x}) d\mathbf{z}}_{=1} && \text{(multiply 1)} \\
 &= \int \underbrace{\log p(\mathbf{x})}_{\text{some constant wrt } \mathbf{z}} \times \underbrace{q_\phi(\mathbf{z}|\mathbf{x})}_{\text{distribution in } \mathbf{z}} d\mathbf{z} && \text{(move } \log p(\mathbf{x}) \text{ into integral)} \\
 &= \mathbb{E}_{q_\phi(\mathbf{z}|\mathbf{x})} [\log p(\mathbf{x})],
 \end{aligned} \tag{1.7}$$

where the last equality is the fact that $\int a \times p_Z(z) dz = \mathbb{E}[a] = a$ for any random variable Z and a scalar a .

See, we have already got $\mathbb{E}_{q_\phi(\mathbf{z}|\mathbf{x})}[\cdot]$. Just a few more steps. Let's use Bayes theorem which states

that $p(\mathbf{x}, \mathbf{z}) = p(\mathbf{z}|\mathbf{x})p(\mathbf{x})$:

$$\begin{aligned}
\mathbb{E}_{q_\phi(\mathbf{z}|\mathbf{x})}[\log p(\mathbf{x})] &= \mathbb{E}_{q_\phi(\mathbf{z}|\mathbf{x})} \left[\log \frac{p(\mathbf{x}, \mathbf{z})}{p(\mathbf{z}|\mathbf{x})} \right] && \text{(Bayes Theorem)} \\
&= \mathbb{E}_{q_\phi(\mathbf{z}|\mathbf{x})} \left[\log \frac{p(\mathbf{x}, \mathbf{z})}{p(\mathbf{z}|\mathbf{x})} \times \frac{q_\phi(\mathbf{z}|\mathbf{x})}{q_\phi(\mathbf{z}|\mathbf{x})} \right] && \text{(Multiply } \frac{q_\phi(\mathbf{z}|\mathbf{x})}{q_\phi(\mathbf{z}|\mathbf{x})} \text{)} \\
&= \underbrace{\mathbb{E}_{q_\phi(\mathbf{z}|\mathbf{x})} \left[\log \frac{p(\mathbf{x}, \mathbf{z})}{q_\phi(\mathbf{z}|\mathbf{x})} \right]}_{\text{ELBO}} + \underbrace{\mathbb{E}_{q_\phi(\mathbf{z}|\mathbf{x})} \left[\log \frac{q_\phi(\mathbf{z}|\mathbf{x})}{p(\mathbf{z}|\mathbf{x})} \right]}_{\mathbb{D}_{\text{KL}}(q_\phi(\mathbf{z}|\mathbf{x}) \| p(\mathbf{z}|\mathbf{x}))}, && (1.8)
\end{aligned}$$

where we recognize that the first term is exactly ELBO, whereas the second term is exactly the KL divergence. Comparing Eqn (1.8) with Eqn (1.5), we complete the proof.

Example 1.4. Using the previous example, we can minimize the gap between $\log p(\mathbf{x})$ and $\text{ELBO}(\mathbf{x})$ if we knew $p(\mathbf{z}|\mathbf{x})$. To see that, we note that $\log p(\mathbf{x})$ is

$$\log p(\mathbf{x}) = \text{ELBO}(\mathbf{x}) + \mathbb{D}_{\text{KL}}(q_\phi(\mathbf{z}|\mathbf{x}) \| p(\mathbf{z}|\mathbf{x})) \geq \text{ELBO}(\mathbf{x}).$$

The equality holds if and only if the KL-divergence term is zero. For the KL divergence to be zero, it is necessary that $q_\phi(\mathbf{z}|\mathbf{x}) = p(\mathbf{z}|\mathbf{x})$. However, since $p(\mathbf{z}|\mathbf{x})$ is a delta function, the only possibility is to have

$$\begin{aligned}
q_\phi(\mathbf{z}|\mathbf{x}) &= \mathcal{N}(\mathbf{z} \mid \frac{\mathbf{x}-\boldsymbol{\mu}}{\sigma}, \ 0) \\
&= \delta(\mathbf{z} - \frac{\mathbf{x}-\boldsymbol{\mu}}{\sigma}),
\end{aligned} \tag{1.9}$$

i.e., we set the standard deviation to be $t = 0$. To determine $p_\theta(\mathbf{x}|\mathbf{z})$, we need some additional steps to simplify ELBO.

We now have ELBO. But this ELBO is still not too useful because it involves $p(\mathbf{x}, \mathbf{z})$, something we have no access to. So, we need to do a little more work.

Theorem 1.2. Interpretation of ELBO. ELBO can be decomposed as

$$\text{ELBO}(\mathbf{x}) = \underbrace{\mathbb{E}_{q_\phi(\mathbf{z}|\mathbf{x})}[\log \overbrace{p_\theta(\mathbf{x}|\mathbf{z})}^{\text{a Gaussian}}]}_{\text{how good your decoder is}} - \underbrace{\mathbb{D}_{\text{KL}}\left(\overbrace{q_\phi(\mathbf{z}|\mathbf{x})}^{\text{a Gaussian}} \parallel \overbrace{p(\mathbf{z})}^{\text{a Gaussian}}\right)}_{\text{how good your encoder is}}. \tag{1.10}$$

Proof. Let's take a closer look at ELBO

$$\begin{aligned}
\text{ELBO}(\mathbf{x}) &\stackrel{\text{def}}{=} \mathbb{E}_{q_\phi(\mathbf{z}|\mathbf{x})} \left[\log \frac{p(\mathbf{x}, \mathbf{z})}{q_\phi(\mathbf{z}|\mathbf{x})} \right] && \text{(definition)} \\
&= \mathbb{E}_{q_\phi(\mathbf{z}|\mathbf{x})} \left[\log \frac{p(\mathbf{x}|\mathbf{z})p(\mathbf{z})}{q_\phi(\mathbf{z}|\mathbf{x})} \right] && (p(\mathbf{x}, \mathbf{z}) = p(\mathbf{x}|\mathbf{z})p(\mathbf{z})) \\
&= \mathbb{E}_{q_\phi(\mathbf{z}|\mathbf{x})} [\log p(\mathbf{x}|\mathbf{z})] + \mathbb{E}_{q_\phi(\mathbf{z}|\mathbf{x})} \left[\log \frac{p(\mathbf{z})}{q_\phi(\mathbf{z}|\mathbf{x})} \right] && \text{(split expectation)} \\
&= \mathbb{E}_{q_\phi(\mathbf{z}|\mathbf{x})} [\log p_\theta(\mathbf{x}|\mathbf{z})] - \mathbb{D}_{\text{KL}}(q_\phi(\mathbf{z}|\mathbf{x}) \| p(\mathbf{z})), && \text{(definition of KL)}
\end{aligned}$$

where we replaced the inaccessible $p(\mathbf{x}|\mathbf{z})$ by its proxy $p_\theta(\mathbf{x}|\mathbf{z})$.

This is a *beautiful* result. We just showed something very easy to understand. Let's look at the two terms in Eqn (1.10):

- **Reconstruction.** The first term is about the *decoder*. We want the decoder to produce a good image \mathbf{x} if we feed a latent \mathbf{z} into the decoder (of course!). So, we want to *maximize* $\log p_{\theta}(\mathbf{x}|\mathbf{z})$. It is similar to maximum likelihood where we want to find the model parameter to maximize the likelihood of observing the image. The expectation here is taken with respect to the samples \mathbf{z} (conditioned on \mathbf{x}). This shouldn't be a surprise because the samples \mathbf{z} are used to assess the quality of the decoder. It cannot be an arbitrary noise vector but a meaningful latent vector. So, \mathbf{z} needs to be sampled from $q_{\phi}(\mathbf{z}|\mathbf{x})$.
- **Prior Matching.** The second term is the KL divergence for the *encoder*. We want the encoder to turn \mathbf{x} into a latent vector \mathbf{z} such that the latent vector will follow our choice of distribution, e.g., $\mathbf{z} \sim \mathcal{N}(0, \mathbf{I})$. To be slightly more general, we write $p(\mathbf{z})$ as the target distribution. Because the KL divergence is a distance (which increases when the two distributions become more dissimilar), we need to put a negative sign in front so that it increases when the two distributions become more similar.

Example 1.5. Following up on the previous example, we continue to assume that we *knew* $p(\mathbf{z}|\mathbf{x})$. Then the reconstruction term in ELBO will give us

$$\begin{aligned}
\mathbb{E}_{q_{\phi}(\mathbf{z}|\mathbf{x})}[\log p_{\theta}(\mathbf{x}|\mathbf{z})] &= \mathbb{E}_{q_{\phi}(\mathbf{z}|\mathbf{x})}[\log \mathcal{N}(\mathbf{x} | c\mathbf{z} + \mathbf{v}, s^2\mathbf{I})] \\
&= \mathbb{E}_{q_{\phi}(\mathbf{z}|\mathbf{x})} \left[-\frac{1}{2} \log 2\pi - \log s - \frac{\|\mathbf{x} - (c\mathbf{z} + \mathbf{v})\|^2}{2s^2} \right] \\
&= -\frac{1}{2} \log 2\pi - \log s - \frac{c^2}{2s^2} \mathbb{E}_{q_{\phi}(\mathbf{z}|\mathbf{x})} [\|\mathbf{z} - \frac{\mathbf{x}-\mathbf{v}}{c}\|^2] \\
&= -\frac{1}{2} \log 2\pi - \log s - \frac{c^2}{2s^2} \mathbb{E}_{\delta(\mathbf{z} - \frac{\mathbf{x}-\mu}{\sigma})} [\|\mathbf{z} - \frac{\mathbf{x}-\mathbf{v}}{c}\|^2] \\
&= -\frac{1}{2} \log 2\pi - \log s - \frac{c^2}{2s^2} [\|\frac{\mathbf{x}-\mu}{\sigma} - \frac{\mathbf{x}-\mathbf{v}}{c}\|^2] \\
&\leq -\frac{1}{2} \log 2\pi - \log s,
\end{aligned}$$

where the upper bound is tight if and only if the norm-square term is zero, which holds when $\mathbf{v} = \boldsymbol{\mu}$ and $c = \sigma$. For the remaining terms, it is clear that $-\log s$ is a monotonically decreasing function in s with $-\log s \rightarrow \infty$ as $s \rightarrow 0$. Therefore, when $\mathbf{v} = \boldsymbol{\mu}$ and $c = \sigma$, it follows that $\mathbb{E}_{q_{\phi}(\mathbf{z}|\mathbf{x})}[\log p_{\theta}(\mathbf{x}|\mathbf{z})]$ is maximized when $s = 0$. This implies that

$$\begin{aligned}
p_{\theta}(\mathbf{x}|\mathbf{z}) &= \mathcal{N}(\mathbf{x} | \sigma\mathbf{z} + \boldsymbol{\mu}, 0) \\
&= \delta(\mathbf{x} - (\sigma\mathbf{z} + \boldsymbol{\mu})).
\end{aligned} \tag{1.11}$$

Limitation of ELBO. ELBO is practically useful, but it is *not* the same as the true likelihood $\log p(\mathbf{x})$. As we mentioned, ELBO is exactly equal to $\log p(\mathbf{x})$ if and only if $\mathbb{D}_{\text{KL}}(q_{\phi}(\mathbf{z}|\mathbf{x})||p(\mathbf{z}|\mathbf{x})) = 0$ which happens when $q_{\phi}(\mathbf{z}|\mathbf{x}) = p(\mathbf{z}|\mathbf{x})$. In the following example, we will show a case where the $q_{\phi}(\mathbf{z}|\mathbf{x})$ obtained from maximizing ELBO is not the same as $p(\mathbf{z}|\mathbf{x})$.

Example 1.6. (Limitation of ELBO). In the previous example, if we have no idea about $p(\mathbf{z}|\mathbf{x})$, we need to train the VAE by maximizing ELBO. However, since ELBO is only a lower bound of the true distribution $\log p(\mathbf{x})$, maximizing ELBO will not return us the delta functions as we hope. Instead, we will obtain something that is quite meaningful but not exactly the delta functions.

For simplicity, let's consider the distributions that will return us unbiased estimates of the mean but with unknown variances:

$$\begin{aligned}
q_{\phi}(\mathbf{z}|\mathbf{x}) &= \mathcal{N}(\mathbf{z} | \frac{\mathbf{x}-\boldsymbol{\mu}}{\sigma}, t^2\mathbf{I}), \\
p_{\theta}(\mathbf{x}|\mathbf{z}) &= \mathcal{N}(\mathbf{x} | \sigma\mathbf{z} + \boldsymbol{\mu}, s^2\mathbf{I}).
\end{aligned}$$

This is partially “cheating” because in theory we should not assume anything about the estimates of

the means. But from an intuitive angle, since $q_\phi(\mathbf{z}|\mathbf{x})$ and $p_\theta(\mathbf{x}|\mathbf{z})$ are proxies to $p(\mathbf{z}|\mathbf{x})$ and $p(\mathbf{x}|\mathbf{z})$, they must resemble some properties of the delta functions. The closest choice is to define $q_\phi(\mathbf{z}|\mathbf{x})$ and $p_\theta(\mathbf{x}|\mathbf{z})$ as Gaussians with means consistent with those of the two delta functions. The variances are unknown, and they are the subject of interest in this example.

Our focus here is to maximize ELBO which consists of the prior matching term and the reconstruction term. For the prior matching error, we want to minimize the KL-divergence:

$$\mathbb{D}_{\text{KL}}(q_\phi(\mathbf{z}|\mathbf{x}) \| p(\mathbf{z})) = \mathbb{D}_{\text{KL}}(\mathcal{N}(\mathbf{z} | \frac{\mathbf{x}-\boldsymbol{\mu}}{\sigma}, t^2 \mathbf{I}) \| \mathcal{N}(\mathbf{z} | 0, \mathbf{I})).$$

The KL-divergence of two multivariate Gaussians $\mathcal{N}(\mathbf{z}|\boldsymbol{\mu}_0, \Sigma_0)$ and $\mathcal{N}(\mathbf{z}|\boldsymbol{\mu}_1, \Sigma_1)$ has a closed form expression which can be found in Wikipedia:

$$\begin{aligned} & \mathbb{D}_{\text{KL}}(\mathcal{N}(\boldsymbol{\mu}_0, \Sigma_0) \| \mathcal{N}(\boldsymbol{\mu}_1, \Sigma_1)) \\ &= \frac{1}{2} \left(\text{Tr}(\Sigma_1^{-1} \Sigma_0) - d + (\boldsymbol{\mu}_1 - \boldsymbol{\mu}_0)^T \Sigma_1^{-1} (\boldsymbol{\mu}_1 - \boldsymbol{\mu}_0) + \log \frac{\det \Sigma_1}{\det \Sigma_0} \right). \end{aligned}$$

Using this result (and with some algebra), we can show that

$$\mathbb{D}_{\text{KL}}(\mathcal{N}(\mathbf{z} | \frac{\mathbf{x}-\boldsymbol{\mu}}{\sigma}, t^2 \mathbf{I}) \| \mathcal{N}(\mathbf{z} | 0, \mathbf{I})) = \frac{1}{2} [t^2 d - d + \|\frac{\mathbf{x}-\boldsymbol{\mu}}{\sigma}\|^2 - 2d \log t],$$

where d is the dimension of \mathbf{x} and \mathbf{z} . To minimize the KL-divergence, we take derivative with respect to t and show that

$$\frac{\partial}{\partial t} \left\{ \frac{1}{2} [t^2 d - d + \|\frac{\mathbf{x}-\boldsymbol{\mu}}{\sigma}\|^2 - 2d \log t] \right\} = t \cdot d - \frac{d}{t}.$$

Setting this to zero will give us $t = 1$. Therefore, we can show that

$$q_\phi(\mathbf{z}|\mathbf{x}) = \mathcal{N}(\mathbf{z} | \frac{\mathbf{x}-\boldsymbol{\mu}}{\sigma}, \mathbf{I}).$$

For the reconstruction term, we can show that

$$\begin{aligned} \mathbb{E}_{q_\phi(\mathbf{z}|\mathbf{x})}[\log p_\theta(\mathbf{x}|\mathbf{z})] &= \mathbb{E}_{q_\phi(\mathbf{z}|\mathbf{x})} \left[\log \frac{1}{(\sqrt{2\pi s^2})^d} \exp \left\{ -\frac{\|\mathbf{x} - (\sigma\mathbf{z} + \boldsymbol{\mu})\|^2}{2s^2} \right\} \right] \\ &= \mathbb{E}_{q_\phi(\mathbf{z}|\mathbf{x})} \left[-\frac{d}{2} \log 2\pi - d \log s - \frac{\|\mathbf{x} - (\sigma\mathbf{z} + \boldsymbol{\mu})\|^2}{2s^2} \right] \\ &= -\frac{d}{2} \log 2\pi - d \log s - \frac{\sigma^2}{2s^2} \mathbb{E}_{q_\phi(\mathbf{z}|\mathbf{x})} [\|\mathbf{z} - \frac{\mathbf{x}-\boldsymbol{\mu}}{\sigma}\|^2] \\ &= -\frac{d}{2} \log 2\pi - d \log s - \frac{\sigma^2}{2s^2} \text{Trace} \left\{ \mathbb{E}_{q_\phi(\mathbf{z}|\mathbf{x})} \left[(\mathbf{z} - \frac{\mathbf{x}-\boldsymbol{\mu}}{\sigma}) (\mathbf{z} - \frac{\mathbf{x}-\boldsymbol{\mu}}{\sigma})^T \right] \right\} \\ &= -\frac{d}{2} \log 2\pi - d \log s - \frac{\sigma^2}{2s^2} \cdot d, \end{aligned}$$

because the covariance of $\mathbf{z} \sim q_\phi(\mathbf{z}|\mathbf{x})$ is \mathbf{I} and so the trace will give us d . Taking derivatives with respect to s will give us

$$\frac{d}{ds} \left\{ -\frac{d}{2} \log 2\pi - d \log s - \frac{d\sigma^2}{2s^2} \right\} = -\frac{d}{s} + \frac{d\sigma^2}{s^3} = 0.$$

Equating this to zero will give us $s = \sigma$. Therefore,

$$p_\theta(\mathbf{x}|\mathbf{z}) = \mathcal{N}(\mathbf{x} | \sigma\mathbf{z} + \boldsymbol{\mu}, \sigma^2 \mathbf{I}).$$

As we can see in this example and the previous example, while the ideal distributions are delta functions, the proxy distributions we obtain have a finite variance. This finite variance adds additional randomness to the samples generated by the VAE. There is nothing wrong with this VAE — we do it correctly by maximizing ELBO. It is just that maximizing the ELBO is not the same as maximizing $\log p(\mathbf{x})$.

1.3 Optimization in VAE

In the previous two subsections we introduced the building blocks of VAE and ELBO. The goal of this subsection is to discuss how to train a VAE and how to do inference.

VAE is a model that aims to approximate the true distribution $p(\mathbf{x})$ so that we can draw samples. A VAE is parameterized by (ϕ, θ) . Therefore, training a VAE is equivalent to solving an optimization problem that encapsulates the essence of $p(\mathbf{x})$ while being tractable. However, since $p(\mathbf{x})$ is not accessible, the natural alternative is to optimize the ELBO which is the lower bound of $\log p(\mathbf{x})$. That means, the learning goal of VAE is to solve the following problem.

Definition 1.4. The optimization objective of VAE is to maximize the ELBO:

$$(\phi, \theta) = \underset{\phi, \theta}{\operatorname{argmax}} \sum_{\mathbf{x} \in \mathcal{X}} \text{ELBO}(\mathbf{x}), \quad (1.12)$$

where $\mathcal{X} = \{\mathbf{x}^{(\ell)} \mid \ell = 1, \dots, L\}$ is the training dataset.

Intractability of ELBO's Gradient. The challenge associated with the above optimization is that the gradient of ELBO with respect to (ϕ, θ) is intractable. Since the majority of today's neural network optimizers use first-order methods and backpropagate the gradient to update the network weights, an intractable gradient will pose difficulties in training the VAE.

Let's elaborate more about the intractability of the gradient. We first substitute Definition 1.3 into the above objective function. The gradient of ELBO is:²

$$\begin{aligned} \nabla_{\theta, \phi} \text{ELBO}(\mathbf{x}) &= \nabla_{\theta, \phi} \left\{ \mathbb{E}_{q_\phi(\mathbf{z}|\mathbf{x})} \left[\log \frac{p_\theta(\mathbf{x}, \mathbf{z})}{q_\phi(\mathbf{z}|\mathbf{x})} \right] \right\} \\ &= \nabla_{\theta, \phi} \left\{ \mathbb{E}_{q_\phi(\mathbf{z}|\mathbf{x})} \left[\log p_\theta(\mathbf{x}, \mathbf{z}) - \log q_\phi(\mathbf{z}|\mathbf{x}) \right] \right\}. \end{aligned} \quad (1.13)$$

The gradient contains two parameters. Let's first look at θ . We can show that

$$\begin{aligned} \nabla_{\theta} \text{ELBO}(\mathbf{x}) &= \nabla_{\theta} \left\{ \mathbb{E}_{q_\phi(\mathbf{z}|\mathbf{x})} \left[\log p_\theta(\mathbf{x}, \mathbf{z}) - \log q_\phi(\mathbf{z}|\mathbf{x}) \right] \right\} \\ &= \nabla_{\theta} \left\{ \int \left[\log p_\theta(\mathbf{x}, \mathbf{z}) - \log q_\phi(\mathbf{z}|\mathbf{x}) \right] \cdot q_\phi(\mathbf{z}|\mathbf{x}) d\mathbf{z} \right\} \\ &= \int \nabla_{\theta} \left\{ \log p_\theta(\mathbf{x}, \mathbf{z}) - \log q_\phi(\mathbf{z}|\mathbf{x}) \right\} \cdot q_\phi(\mathbf{z}|\mathbf{x}) d\mathbf{z} \\ &= \mathbb{E}_{q_\phi(\mathbf{z}|\mathbf{x})} \left[\nabla_{\theta} \left\{ \log p_\theta(\mathbf{x}, \mathbf{z}) - \log q_\phi(\mathbf{z}|\mathbf{x}) \right\} \right] \\ &= \mathbb{E}_{q_\phi(\mathbf{z}|\mathbf{x})} \left[\nabla_{\theta} \left\{ \log p_\theta(\mathbf{x}, \mathbf{z}) \right\} \right] \\ &\approx \frac{1}{L} \sum_{\ell=1}^L \nabla_{\theta} \left\{ \log p_\theta(\mathbf{x}, \mathbf{z}^{(\ell)}) \right\}, \quad (\text{where } \mathbf{z}^{(\ell)} \sim q_\phi(\mathbf{z}|\mathbf{x})) \end{aligned} \quad (1.14)$$

where the last equality is the Monte Carlo approximation of the expectation.

In the above equation, if $p_\theta(\mathbf{x}, \mathbf{z})$ is realized by a computable model such as a neural network, then its gradient $\nabla_{\theta} \{\log p_\theta(\mathbf{x}, \mathbf{z})\}$ can be computed via automatic differentiation. Thus, the maximization can be achieved by backpropagating the gradient.

²The original definition of ELBO uses the true joint distribution $p(\mathbf{x}, \mathbf{z})$. In practice, since $p(\mathbf{x}, \mathbf{z})$ is not accessible, we replace it by its proxy $p_\theta(\mathbf{x}, \mathbf{z})$ which is a computable distribution.

The gradient with respect to ϕ is more difficult. We can show that

$$\begin{aligned}
\nabla_{\phi} \text{ELBO}(\mathbf{x}) &= \nabla_{\phi} \left\{ \mathbb{E}_{q_{\phi}(\mathbf{z}|\mathbf{x})} \left[\log p_{\theta}(\mathbf{x}, \mathbf{z}) - \log q_{\phi}(\mathbf{z}|\mathbf{x}) \right] \right\} \\
&= \nabla_{\phi} \left\{ \int \left[\log p_{\theta}(\mathbf{x}, \mathbf{z}) - \log q_{\phi}(\mathbf{z}|\mathbf{x}) \right] \cdot q_{\phi}(\mathbf{z}|\mathbf{x}) d\mathbf{z} \right\} \\
&= \int \nabla_{\phi} \left\{ [\log p_{\theta}(\mathbf{x}, \mathbf{z}) - \log q_{\phi}(\mathbf{z}|\mathbf{x})] \cdot q_{\phi}(\mathbf{z}|\mathbf{x}) \right\} d\mathbf{z} \\
&\neq \int \nabla_{\phi} \left\{ \log p_{\theta}(\mathbf{x}, \mathbf{z}) - \log q_{\phi}(\mathbf{z}|\mathbf{x}) \right\} \cdot q_{\phi}(\mathbf{z}|\mathbf{x}) d\mathbf{z} \\
&= \mathbb{E}_{q_{\phi}(\mathbf{z}|\mathbf{x})} \left[\nabla_{\phi} \left\{ \log p_{\theta}(\mathbf{x}, \mathbf{z}) - \log q_{\phi}(\mathbf{z}|\mathbf{x}) \right\} \right] \\
&= \mathbb{E}_{q_{\phi}(\mathbf{z}|\mathbf{x})} \left[\nabla_{\phi} \left\{ -\log q_{\phi}(\mathbf{z}|\mathbf{x}) \right\} \right] \\
&\approx \frac{1}{L} \sum_{\ell=1}^L \nabla_{\phi} \left\{ -\log q_{\phi}(\mathbf{z}^{(\ell)}|\mathbf{x}) \right\}, \quad (\text{where } \mathbf{z}^{(\ell)} \sim q_{\phi}(\mathbf{z}|\mathbf{x})). \quad (1.15)
\end{aligned}$$

As we can see, even though we *wish* to maintain a similar structure as we did for θ , the expectation and the gradient operators in the above derivations cannot be switched. This forbids us from doing any backpropagation of the gradient to maximize ELBO.

Reparameterization Trick. The intractability of ELBO's gradient is inherited from the fact that we need to draw samples \mathbf{z} from a distribution $q_{\phi}(\mathbf{z}|\mathbf{x})$ which itself is a function of ϕ . As noted by Kingma and Welling [23], for continuous latent variables, it is possible to compute an unbiased estimate of $\nabla_{\theta, \phi} \text{ELBO}(\mathbf{x})$ so that we can approximately calculate the gradient and hence maximize ELBO. The idea is to employ a technique known as the *reparameterization trick* [23].

Recall that the latent variable \mathbf{z} is a sample drawn from the distribution $q_{\phi}(\mathbf{z}|\mathbf{x})$. The idea of reparameterization trick is to express \mathbf{z} as some differentiable and invertible transformation of another random variable ϵ whose distribution is independent of \mathbf{x} and ϕ . That is, we define a differentiable and invertible function \mathbf{g} such that

$$\mathbf{z} = \mathbf{g}(\epsilon, \phi, \mathbf{x}), \quad (1.16)$$

for some random variable $\epsilon \sim p(\epsilon)$. To make our discussions easier, we pose an additional requirement that

$$q_{\phi}(\mathbf{z}|\mathbf{x}) \cdot \left| \det \left(\frac{\partial \mathbf{z}}{\partial \epsilon} \right) \right| = p(\epsilon), \quad (1.17)$$

where $\frac{\partial \mathbf{z}}{\partial \epsilon}$ is the Jacobian, and $\det(\cdot)$ is the matrix determinant. This requirement is related to change of variables in multivariate calculus. The following example will make it clear.

Example 1.7. Suppose $\mathbf{z} \sim q_{\phi}(\mathbf{z}|\mathbf{x}) \stackrel{\text{def}}{=} \mathcal{N}(\mathbf{z} | \boldsymbol{\mu}, \text{diag}(\boldsymbol{\sigma}^2))$. We can define

$$\mathbf{z} = \mathbf{g}(\epsilon, \phi, \mathbf{x}) \stackrel{\text{def}}{=} \epsilon \odot \boldsymbol{\sigma} + \boldsymbol{\mu}, \quad (1.18)$$

where $\epsilon \sim \mathcal{N}(0, \mathbf{I})$ and “ \odot ” means elementwise multiplication. The parameter ϕ is $\phi = (\boldsymbol{\mu}, \boldsymbol{\sigma}^2)$. For this choice of the distribution, we can show that by letting $\epsilon = \frac{\mathbf{z} - \boldsymbol{\mu}}{\boldsymbol{\sigma}}$:

$$\begin{aligned}
q_{\phi}(\mathbf{z}|\mathbf{x}) \cdot \left| \det \left(\frac{\partial \mathbf{z}}{\partial \epsilon} \right) \right| &= \prod_{i=1}^d \frac{1}{\sqrt{2\pi\sigma_i^2}} \exp \left\{ -\frac{(z_i - \mu_i)^2}{2\sigma_i^2} \right\} \cdot \prod_{i=1}^d \sigma_i \\
&= \frac{1}{(\sqrt{2\pi})^d} \exp \left\{ -\frac{\|\epsilon\|^2}{2} \right\} = \mathcal{N}(0, \mathbf{I}) = p(\epsilon).
\end{aligned}$$

With this re-parameterization of \mathbf{z} by expressing it in terms of ϵ , we can look at $\nabla_{\phi} \mathbb{E}_{q_{\phi}(\mathbf{z}|\mathbf{x})}[f(\mathbf{z})]$ for some general function $f(\mathbf{z})$. (Later we will consider $f(\mathbf{z}) = -\log q_{\phi}(\mathbf{z}|\mathbf{x})$.) For notational simplicity, we

write $\mathbf{g}(\boldsymbol{\epsilon})$ instead of $\mathbf{g}(\boldsymbol{\epsilon}, \boldsymbol{\phi}, \mathbf{x})$ although we understand that \mathbf{g} has three inputs. By change of variables, we can show that

$$\begin{aligned}
\mathbb{E}_{q_{\boldsymbol{\phi}}(\mathbf{z}|\mathbf{x})}[f(\mathbf{z})] &= \int f(\mathbf{z}) \cdot q_{\boldsymbol{\phi}}(\mathbf{z}|\mathbf{x}) d\mathbf{z} \\
&= \int f(\mathbf{g}(\boldsymbol{\epsilon})) \cdot q_{\boldsymbol{\phi}}(\mathbf{g}(\boldsymbol{\epsilon})|\mathbf{x}) d\mathbf{g}(\boldsymbol{\epsilon}), && (\mathbf{z} = \mathbf{g}(\boldsymbol{\epsilon})) \\
&= \int f(\mathbf{g}(\boldsymbol{\epsilon})) \cdot q_{\boldsymbol{\phi}}(\mathbf{g}(\boldsymbol{\epsilon})|\mathbf{x}) \cdot \left| \det \left(\frac{\partial \mathbf{g}(\boldsymbol{\epsilon})}{\partial \boldsymbol{\epsilon}} \right) \right| d\boldsymbol{\epsilon} && (\text{Jacobian due to change of variable}) \\
&= \int f(\mathbf{z}) \cdot p(\boldsymbol{\epsilon}) d\boldsymbol{\epsilon} && (\text{use Eqn (1.17)}) \\
&= \mathbb{E}_{p(\boldsymbol{\epsilon})}[f(\mathbf{z})]. && (1.19)
\end{aligned}$$

So, if we want to take the gradient with respect to $\boldsymbol{\phi}$, we can show that

$$\begin{aligned}
\nabla_{\boldsymbol{\phi}} \mathbb{E}_{q_{\boldsymbol{\phi}}(\mathbf{z}|\mathbf{x})}[f(\mathbf{z})] &= \nabla_{\boldsymbol{\phi}} \mathbb{E}_{p(\boldsymbol{\epsilon})}[f(\mathbf{z})] = \nabla_{\boldsymbol{\phi}} \left\{ \int f(\mathbf{z}) \cdot p(\boldsymbol{\epsilon}) d\boldsymbol{\epsilon} \right\} \\
&= \int \nabla_{\boldsymbol{\phi}} \{f(\mathbf{z}) \cdot p(\boldsymbol{\epsilon})\} d\boldsymbol{\epsilon} \\
&= \int \{\nabla_{\boldsymbol{\phi}} f(\mathbf{z})\} \cdot p(\boldsymbol{\epsilon}) d\boldsymbol{\epsilon} \\
&= \mathbb{E}_{p(\boldsymbol{\epsilon})}[\nabla_{\boldsymbol{\phi}} f(\mathbf{z})], && (1.20)
\end{aligned}$$

which can be approximated by Monte Carlo. Substituting $f(\mathbf{z}) = -\log q_{\boldsymbol{\phi}}(\mathbf{z}|\mathbf{x})$, we can show that

$$\begin{aligned}
\nabla_{\boldsymbol{\phi}} \mathbb{E}_{q_{\boldsymbol{\phi}}(\mathbf{z}|\mathbf{x})}[-\log q_{\boldsymbol{\phi}}(\mathbf{z}|\mathbf{x})] &= \mathbb{E}_{p(\boldsymbol{\epsilon})}[-\nabla_{\boldsymbol{\phi}} \log q_{\boldsymbol{\phi}}(\mathbf{z}|\mathbf{x})] \\
&\approx -\frac{1}{L} \sum_{\ell=1}^L \nabla_{\boldsymbol{\phi}} \log q_{\boldsymbol{\phi}}(\mathbf{z}^{(\ell)}|\mathbf{x}), && (\text{where } \mathbf{z}^{(\ell)} = \mathbf{g}(\boldsymbol{\epsilon}^{(\ell)}, \boldsymbol{\phi}, \mathbf{x})) \\
&= -\frac{1}{L} \sum_{\ell=1}^L \nabla_{\boldsymbol{\phi}} \left[\log p(\boldsymbol{\epsilon}^{(\ell)}) - \log \left| \det \frac{\partial \mathbf{z}^{(\ell)}}{\partial \boldsymbol{\epsilon}^{(\ell)}} \right| \right] \\
&= \frac{1}{L} \sum_{\ell=1}^L \nabla_{\boldsymbol{\phi}} \left[\log \left| \det \frac{\partial \mathbf{z}^{(\ell)}}{\partial \boldsymbol{\epsilon}^{(\ell)}} \right| \right].
\end{aligned}$$

So, as long as the determinant is differentiable with respect to $\boldsymbol{\phi}$, the Monte Carlo approximation can be numerically computed.

Example 1.8. Suppose that the parameters and the distribution $q_{\boldsymbol{\phi}}$ are defined as follows:

$$\begin{aligned}
(\boldsymbol{\mu}, \boldsymbol{\sigma}^2) &= \text{EncoderNetwork}_{\boldsymbol{\phi}}(\mathbf{x}) \\
q_{\boldsymbol{\phi}}(\mathbf{z}|\mathbf{x}) &= \mathcal{N}(\mathbf{z} \mid \boldsymbol{\mu}, \text{diag}(\boldsymbol{\sigma}^2)).
\end{aligned}$$

We can define $\mathbf{z} = \boldsymbol{\mu} + \boldsymbol{\sigma} \odot \boldsymbol{\epsilon}$, with $\boldsymbol{\epsilon} \sim \mathcal{N}(0, \mathbf{I})$. Then, we can show that

$$\begin{aligned}
\log \left| \det \frac{\partial \mathbf{z}}{\partial \boldsymbol{\epsilon}} \right| &= \log \left| \det \left(\frac{\partial(\boldsymbol{\mu} + \boldsymbol{\sigma} \odot \boldsymbol{\epsilon})}{\partial \boldsymbol{\epsilon}} \right) \right| \\
&= \log \left| \det \left(\text{diag}\{\boldsymbol{\sigma}\} \right) \right| \\
&= \log \prod_{i=1}^d \sigma_i = \sum_{i=1}^d \log \sigma_i.
\end{aligned}$$

Therefore, we can show that

$$\begin{aligned}
\nabla_{\phi} \mathbb{E}_{q_{\phi}(\mathbf{z}|\mathbf{x})}[-\log q_{\phi}(\mathbf{z}|\mathbf{x})] &\approx \frac{1}{L} \sum_{\ell=1}^L \nabla_{\phi} \left[\log \left| \det \frac{\partial \mathbf{z}^{(\ell)}}{\partial \epsilon^{(\ell)}} \right| \right] \\
&= \frac{1}{L} \sum_{\ell=1}^L \nabla_{\phi} \left[\sum_{i=1}^d \log \sigma_i \right] \\
&= \nabla_{\phi} \left[\sum_{i=1}^d \log \sigma_i \right] \\
&= \frac{1}{\sigma} \odot \nabla_{\phi} \left\{ \sigma_{\phi}(\mathbf{x}) \right\},
\end{aligned}$$

where we emphasize that $\sigma_{\phi}(\mathbf{x})$ is the output of the encoder which is a neural network.

As we can see in the above example, for some specific choices of the distributions (e.g., Gaussian), the gradient of ELBO can be significantly easier to derive.

VAE Encoder. After discussing the reparameterizing trick, we can now discuss the specific structure of the encoder in VAE. To make our discussions focused, we assume a relatively common choice of the encoder:

$$\begin{aligned}
(\mu, \sigma^2) &= \text{EncoderNetwork}_{\phi}(\mathbf{x}) \\
q_{\phi}(\mathbf{z}|\mathbf{x}) &= \mathcal{N}(\mathbf{z} \mid \mu, \sigma^2 \mathbf{I}).
\end{aligned}$$

The parameters μ and σ are technically *neural networks* because they are the outputs of $\text{EncoderNetwork}_{\phi}(\cdot)$. Therefore, it will be helpful if we denote them as

$$\begin{aligned}
\mu &= \underbrace{\mu_{\phi}}_{\text{neural network}}(\mathbf{x}), \\
\sigma^2 &= \underbrace{\sigma_{\phi}^2}_{\text{neural network}}(\mathbf{x}),
\end{aligned}$$

Our notation is slightly more complicated because we want to emphasize that μ is a function of \mathbf{x} ; You give us an image \mathbf{x} , our job is to return you the parameters of the Gaussian (i.e., mean and variance). If you give us a different \mathbf{x} , then the parameters of the Gaussian should also be different. The parameter ϕ specifies that μ is controlled (or parameterized) by ϕ .

Suppose that we are given the ℓ -th training sample $\mathbf{x}^{(\ell)}$. From this $\mathbf{x}^{(\ell)}$ we want to generate a latent variable $\mathbf{z}^{(\ell)}$ which is a sample from $q_{\phi}(\mathbf{z}|\mathbf{x})$. Because of the Gaussian structure, it is equivalent to say that

$$\mathbf{z}^{(\ell)} \sim \mathcal{N}\left(\mathbf{z} \mid \mu_{\phi}(\mathbf{x}^{(\ell)}), \sigma_{\phi}^2(\mathbf{x}^{(\ell)})\mathbf{I}\right). \quad (1.21)$$

The interesting thing about this equation is that we use a neural network $\text{EncoderNetwork}_{\phi}(\cdot)$ to estimate the mean and variance of the Gaussian. Then, from this Gaussian we draw a sample $\mathbf{z}^{(\ell)}$, as illustrated in Figure 1.5.

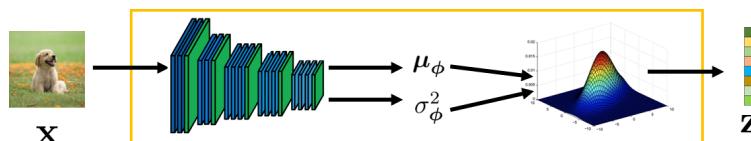


Figure 1.5: Implementation of a VAE encoder. We use a neural network to take the image \mathbf{x} and estimate the mean μ_{ϕ} and variance σ_{ϕ}^2 of the Gaussian distribution.

A more convenient way of expressing Eqn (1.21) is to realize that the sampling operation $\mathbf{z} \sim \mathcal{N}(\mu, \sigma^2 \mathbf{I})$ can be done using the reparameterization trick.

Reparameterization Trick for High-dimensional Gaussian:

$$\mathbf{z} \sim \mathcal{N}(\boldsymbol{\mu}, \sigma^2 \mathbf{I}) \iff \mathbf{z} = \boldsymbol{\mu} + \sigma \boldsymbol{\epsilon}, \quad \boldsymbol{\epsilon} \sim \mathcal{N}(0, \mathbf{I}). \quad (1.22)$$

Using the reparameterization trick, Eqn (1.21) can be written as

$$\mathbf{z}^{(\ell)} = \boldsymbol{\mu}_\phi(\mathbf{x}^{(\ell)}) + \sigma_\phi(\mathbf{x}^{(\ell)}) \boldsymbol{\epsilon}, \quad \boldsymbol{\epsilon} \sim \mathcal{N}(0, \mathbf{I}).$$

Proof. We will prove a general case for an arbitrary covariance matrix Σ instead of a diagonal matrix $\sigma^2 \mathbf{I}$.

For any high-dimensional Gaussian $\mathbf{z} \sim \mathcal{N}(\mathbf{z}|\boldsymbol{\mu}, \Sigma)$, the sampling process can be done via the transformation of white noise

$$\mathbf{z} = \boldsymbol{\mu} + \Sigma^{\frac{1}{2}} \boldsymbol{\epsilon}, \quad (1.23)$$

where $\boldsymbol{\epsilon} \sim \mathcal{N}(0, \mathbf{I})$. The half matrix $\Sigma^{\frac{1}{2}}$ can be obtained through eigen-decomposition or Cholesky factorization. If Σ has an eigen-decomposition $\Sigma = \mathbf{U}\mathbf{S}\mathbf{U}^T$, then $\Sigma^{\frac{1}{2}} = \mathbf{U}\mathbf{S}^{\frac{1}{2}}\mathbf{U}^T$. The square root of the eigenvalue matrix \mathbf{S} is well-defined because Σ is a positive semi-definite matrix.

We can calculate the expectation and covariance of \mathbf{x} :

$$\begin{aligned} \mathbb{E}[\mathbf{z}] &= \mathbb{E}[\boldsymbol{\mu} + \Sigma^{\frac{1}{2}} \boldsymbol{\epsilon}] = \boldsymbol{\mu} + \Sigma^{\frac{1}{2}} \underbrace{\mathbb{E}[\boldsymbol{\epsilon}]}_{=0} = \boldsymbol{\mu}, \\ \text{Cov}(\mathbf{z}) &= \mathbb{E}[(\mathbf{z} - \boldsymbol{\mu})(\mathbf{z} - \boldsymbol{\mu})^T] = \mathbb{E}\left[\Sigma^{\frac{1}{2}} \boldsymbol{\epsilon} \boldsymbol{\epsilon}^T (\Sigma^{\frac{1}{2}})^T\right] = \Sigma^{\frac{1}{2}} \underbrace{\mathbb{E}[\boldsymbol{\epsilon} \boldsymbol{\epsilon}^T]}_{=\mathbf{I}} (\Sigma^{\frac{1}{2}})^T = \Sigma. \end{aligned}$$

Therefore, for diagonal matrices $\Sigma = \sigma^2 \mathbf{I}$, the above is reduced to

$$\mathbf{z} = \boldsymbol{\mu} + \sigma \boldsymbol{\epsilon}, \quad \text{where } \boldsymbol{\epsilon} \sim \mathcal{N}(0, \mathbf{I}). \quad (1.24)$$

Given the VAE encoder structure and $q_\phi(\mathbf{z}|\mathbf{x})$, we can go back to ELBO. Recall that ELBO consists of the prior matching term and the reconstruction term. The prior matching term is measured in terms of the KL divergence $\mathbb{D}_{\text{KL}}(q_\phi(\mathbf{z}|\mathbf{x}) \| p(\mathbf{z}))$. Let's evaluate this KL divergence.

To evaluate the KL divergence, we (re)use a result which we summarize below:

Theorem 1.3. KL-Divergence of Two Gaussian.

The KL divergence for two d -dimensional Gaussian distributions $\mathcal{N}(\boldsymbol{\mu}_0, \Sigma_0)$ and $\mathcal{N}(\boldsymbol{\mu}_1, \Sigma_1)$ is

$$\begin{aligned} \mathbb{D}_{\text{KL}}\left(\mathcal{N}(\boldsymbol{\mu}_0, \Sigma_0) \| \mathcal{N}(\boldsymbol{\mu}_1, \Sigma_1)\right) &= \frac{1}{2} \left(\text{Tr}(\Sigma_1^{-1} \Sigma_0) - d + (\boldsymbol{\mu}_1 - \boldsymbol{\mu}_0)^T \Sigma_1^{-1} (\boldsymbol{\mu}_1 - \boldsymbol{\mu}_0) + \log \frac{\det \Sigma_1}{\det \Sigma_0} \right). \end{aligned} \quad (1.25)$$

Substituting our distributions by considering

$$\begin{aligned} \boldsymbol{\mu}_0 &= \boldsymbol{\mu}_\phi(\mathbf{x}), & \Sigma_0 &= \sigma_\phi^2(\mathbf{x}) \mathbf{I} \\ \boldsymbol{\mu}_1 &= 0, & \Sigma_1 &= \mathbf{I}, \end{aligned}$$

we can show that the KL divergence has an analytic expression

$$\mathbb{D}_{\text{KL}}\left(q_\phi(\mathbf{z}|\mathbf{x}) \| p(\mathbf{z})\right) = \frac{1}{2} \left(\sigma_\phi^2(\mathbf{x})d - d + \|\boldsymbol{\mu}_\phi(\mathbf{x})\|^2 - 2d \log \sigma_\phi(\mathbf{x}) \right), \quad (1.26)$$

where d is the dimension of the vector \mathbf{z} . The gradient of the KL-divergence with respect to ϕ does not have a closed form but they can be calculated numerically:

$$\nabla_\phi \mathbb{D}_{\text{KL}}\left(q_\phi(\mathbf{z}|\mathbf{x}) \| p(\mathbf{z})\right) = \frac{1}{2} \nabla_\phi \left(\sigma_\phi^2(\mathbf{x})d - d + \|\boldsymbol{\mu}_\phi(\mathbf{x})\|^2 - 2d \log \sigma_\phi(\mathbf{x}) \right). \quad (1.27)$$

The gradient with respect to θ is zero because there is nothing dependent on θ .

VAE Decoder. The decoder is implemented through a neural network. For notation simplicity, let's define it as $\text{DecoderNetwork}_\theta(\cdot)$ where θ denotes the network parameters. The job of the decoder network is to take a latent variable \mathbf{z} and generate an image $f_\theta(\mathbf{z})$:

$$f_\theta(\mathbf{z}) = \text{DecoderNetwork}_\theta(\mathbf{z}). \quad (1.28)$$

The distribution $p_\theta(\mathbf{x}|\mathbf{z})$ can be defined as

$$p_\theta(\mathbf{x}|\mathbf{z}) = \mathcal{N}(\mathbf{x} | f_\theta(\mathbf{z}), \sigma_{\text{dec}}^2 \mathbf{I}), \quad \text{for some hyperparameter } \sigma_{\text{dec}}. \quad (1.29)$$

The interpretation of $p_\theta(\mathbf{x}|\mathbf{z})$ is that we estimate $f_\theta(\mathbf{z})$ through a network and put it as the mean of the Gaussian. If we draw a sample \mathbf{x} from $p_\theta(\mathbf{x}|\mathbf{z})$, then by the reparameterization trick we can write the generated image $\hat{\mathbf{x}}$ as

$$\hat{\mathbf{x}} = f_\theta(\mathbf{z}) + \sigma_{\text{dec}} \epsilon, \quad \epsilon \sim \mathcal{N}(0, \mathbf{I}).$$

Moreover, if we take the log of the likelihood, we can show that

$$\begin{aligned} \log p_\theta(\mathbf{x}|\mathbf{z}) &= \log \mathcal{N}(\mathbf{x} | f_\theta(\mathbf{z}), \sigma_{\text{dec}}^2 \mathbf{I}) \\ &= \log \frac{1}{\sqrt{(2\pi\sigma_{\text{dec}}^2)^d}} \exp \left\{ -\frac{\|\mathbf{x} - f_\theta(\mathbf{z})\|^2}{2\sigma_{\text{dec}}^2} \right\} \\ &= -\frac{\|\mathbf{x} - f_\theta(\mathbf{z})\|^2}{2\sigma_{\text{dec}}^2} - \underbrace{\log \sqrt{(2\pi\sigma_{\text{dec}}^2)^d}}_{\text{independent of } \theta \text{ so we can drop it}}. \end{aligned} \quad (1.30)$$

Going back to ELBO, we want to compute $\mathbb{E}_{q_\phi(\mathbf{z}|\mathbf{x})}[\log p_\theta(\mathbf{x}|\mathbf{z})]$. If we straightly calculate the expectation, we will need to compute an integration

$$\begin{aligned} \mathbb{E}_{q_\phi(\mathbf{z}|\mathbf{x})}[\log p_\theta(\mathbf{x}|\mathbf{z})] &= \int \log [\mathcal{N}(\mathbf{x} | f_\theta(\mathbf{z}), \sigma_{\text{dec}}^2 \mathbf{I})] \cdot \mathcal{N}(\mathbf{z} | \mu_\phi(\mathbf{x}), \sigma_\phi^2(\mathbf{x})) d\mathbf{z} \\ &= - \int \frac{\|\mathbf{x} - f_\theta(\mathbf{z})\|^2}{2\sigma_{\text{dec}}^2} \cdot \mathcal{N}(\mathbf{z} | \mu_\phi(\mathbf{x}), \sigma_\phi^2(\mathbf{x})) d\mathbf{z} + C, \end{aligned}$$

where the constant C coming out of the log of the Gaussian can be dropped. By using the reparameterization trick, we write $\mathbf{z} = \mu_\phi(\mathbf{x}) + \sigma_\phi(\mathbf{x})\epsilon$ and substitute it into the above equation. This will give us³

$$\begin{aligned} \mathbb{E}_{q_\phi(\mathbf{z}|\mathbf{x})}[\log p_\theta(\mathbf{x}|\mathbf{z})] &= - \int \frac{\|\mathbf{x} - f_\theta(\mathbf{z})\|^2}{2\sigma_{\text{dec}}^2} \cdot \mathcal{N}(\mathbf{z} | \mu_\phi(\mathbf{x}), \sigma_\phi^2(\mathbf{x})) d\mathbf{z} \\ &\approx -\frac{1}{M} \sum_{m=1}^M \frac{\|\mathbf{x} - f_\theta(\mathbf{z}^{(m)})\|^2}{2\sigma_{\text{dec}}^2} \\ &= -\frac{1}{M} \sum_{m=1}^M \frac{\|\mathbf{x} - f_\theta(\mu_\phi(\mathbf{x}) + \sigma_\phi(\mathbf{x})\epsilon^{(m)})\|^2}{2\sigma_{\text{dec}}^2}. \end{aligned} \quad (1.31)$$

The approximation above is due to Monte Carlo where the randomness is based on the sampling of the $\epsilon \sim \mathcal{N}(\epsilon | 0, \mathbf{I})$. The index M specifies the number of Monte Carlo samples we want to use to approximate the expectation. Note that the input image \mathbf{x} is fixed because $\mathbb{E}_{q_\phi(\mathbf{z}|\mathbf{x})}[\log p_\theta(\mathbf{x}|\mathbf{z})]$ is a function of \mathbf{x} .

The gradient of $\mathbb{E}_{q_\phi(\mathbf{z}|\mathbf{x})}[\log p_\theta(\mathbf{x}|\mathbf{z})]$ with respect to θ is relatively easy to compute. Since only f_θ depends on θ , we can do automatic differentiation. The gradient with respect to ϕ is slightly harder, but it is still computable because we use chain rule and go into $\mu_\phi(\mathbf{x})$ and $\sigma_\phi(\mathbf{x})$.

Inspecting Eqn (1.31), we notice one interesting thing that the loss function is simply the ℓ_2 norm between the reconstructed image $f_\theta(\mathbf{z})$ and the ground truth image \mathbf{x} . This means that if we have the generated image $f_\theta(\mathbf{z})$, we can do a direct comparison with the ground truth \mathbf{x} via the usual ℓ_2 loss as illustrated in Figure 1.6.

³The negative sign here is not a mistake. We want to *maximize* $\mathbb{E}_{q_\phi(\mathbf{z}|\mathbf{x})}[\log p_\theta(\mathbf{x}|\mathbf{z})]$, which is equivalent to *minimize* the negative of the ℓ_2 norm.

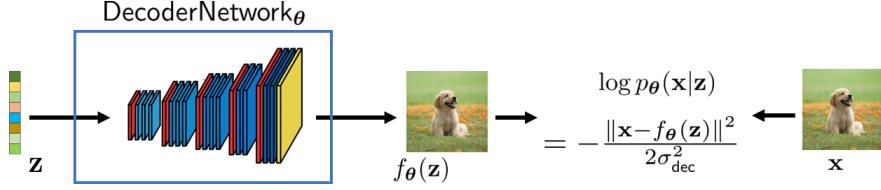


Figure 1.6: Implementation of a VAE decoder. We use a neural network to take the latent vector \mathbf{z} and generate an image $f_\theta(\mathbf{z})$. The log likelihood will give us a quadratic equation if we assume a Gaussian distribution.

Training the VAE. Given a training dataset $\mathcal{X} = \{(\mathbf{x}^{(\ell)})\}_{\ell=1}^L$ of clean images, the training objective of VAE is to maximize the ELBO

$$\operatorname{argmax}_{\theta, \phi} \sum_{\mathbf{x} \in \mathcal{X}} \text{ELBO}_{\phi, \theta}(\mathbf{x}),$$

where the summation is taken with respect to the entire training dataset. The individual ELBO is based on the sum of the terms we derived above

$$\text{ELBO}_{\phi, \theta}(\mathbf{x}) = \mathbb{E}_{q_\phi(\mathbf{z}|\mathbf{x})}[\log p_\theta(\mathbf{x}|\mathbf{z})] - \mathbb{D}_{\text{KL}}(q_\phi(\mathbf{z}|\mathbf{x}) \parallel p(\mathbf{z})). \quad (1.32)$$

Here, the reconstruction term is:

$$\mathbb{E}_{q_\phi(\mathbf{z}|\mathbf{x})}[\log p_\theta(\mathbf{x}|\mathbf{z})] \approx -\frac{1}{M} \sum_{m=1}^M \frac{\|\mathbf{x} - f_\theta(\boldsymbol{\mu}_\phi(\mathbf{x}) + \sigma_\phi(\mathbf{x})\epsilon^{(m)})\|^2}{2\sigma_{\text{dec}}^2}, \quad (1.33)$$

whereas the prior matching term is

$$\mathbb{D}_{\text{KL}}(q_\phi(\mathbf{z}|\mathbf{x}) \parallel p(\mathbf{z})) = \frac{1}{2} (\sigma_\phi^2(\mathbf{x})d - d + \|\boldsymbol{\mu}_\phi(\mathbf{x})\|^2 - 2\log \sigma_\phi(\mathbf{x})). \quad (1.34)$$

To optimize for θ and ϕ , we can run stochastic gradient descent. The gradients can be taken based on the tensor graphs of the neural networks. On computers, this is done automatically by the automatic differentiation.

Let's summarize these.

Theorem 1.4. (VAE Training). To train a VAE, we need to solve the optimization problem

$$\operatorname{argmax}_{\theta, \phi} \sum_{\mathbf{x} \in \mathcal{X}} \text{ELBO}_{\phi, \theta}(\mathbf{x}),$$

where

$$\begin{aligned} \text{ELBO}_{\phi, \theta}(\mathbf{x}) = & -\frac{1}{M} \sum_{m=1}^M \frac{\|\mathbf{x} - f_\theta(\boldsymbol{\mu}_\phi(\mathbf{x}) + \sigma_\phi(\mathbf{x})\epsilon^{(m)})\|^2}{2\sigma_{\text{dec}}^2} \\ & + \frac{1}{2} (\sigma_\phi^2(\mathbf{x})d - d + \|\boldsymbol{\mu}_\phi(\mathbf{x})\|^2 - 2d\log \sigma_\phi(\mathbf{x})). \end{aligned} \quad (1.35)$$

VAE Inference. The inference of an VAE is relatively simple. Once the VAE is trained, we can drop the encoder and only keep the decoder, as shown in Figure 1.7. To generate a new image from the model, we pick a random latent vector $\mathbf{z} \in \mathbb{R}^d$. By sending this \mathbf{z} through the decoder f_θ , we will be able to generate a new image $\hat{\mathbf{x}} = f_\theta(\mathbf{z})$.

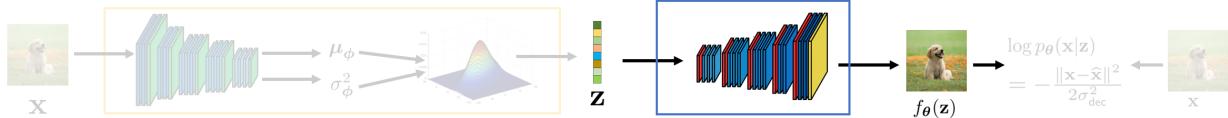


Figure 1.7: Using VAE to generate image is as simple as sending a latent noise code \mathbf{z} through the decoder.

1.4 Concluding Remark

For readers who are looking for additional references, we highly recommend the tutorial by Kingma and Welling [24] which is based on their original VAE paper [23]. A shorter tutorial by Doersch et al [12] can also be helpful. [24] includes a long list of good papers including a paper by Rezende and Mohamed [32] on normalizing flow which was published around the same time as Kingma and Welling's VAE paper.

VAE has many linkages to the classical variational inference and graphical models [45]. VAE is also relevant to the generative adversarial networks (GAN) by Goodfellow et al. [15]. Kingma and Welling commented in [24] that VAE and GAN have complementary properties; while GAN produces better perceptual quality images, there is a weaker linkage with the data likelihood. VAE can meet the data likelihood criterion better but the samples are at times not perceptually as good.

2 Denoising Diffusion Probabilistic Model (DDPM)

In this section, we discuss the diffusion models. There are many different perspectives on how the diffusion models can be derived, e.g., score matching, differential equation, etc. We will follow the approach outlined by the original paper on denoising diffusion probability model by Ho et al. [16].

Before we discuss the mathematical details, let's summarize DDPM from the perspective of VAE's extension:

Diffusion models are *incremental* updates where the assembly of the whole gives us the encoder-decoder structure.

Why increment? It's like turning the direction of a giant ship. You need to turn the ship slowly towards your desired direction or otherwise you will lose control. The same principle applies to your company HR and your university administration.

Bend one inch at a time.

Okay. Enough philosophy. Let's get back to our business.

DDPM has a lot of linkage to a piece of earlier work by Sohl-Dickstein et al in 2015 [38]. Sohl-Dickstein et al asked the question of how to convert from one distribution to another distribution. VAE provides one approach: Referring to the previous section, we can think of the source distribution being the latent variable $\mathbf{z} \sim p(\mathbf{z})$ and the target distribution being the input variable $\mathbf{x} \sim p(\mathbf{x})$. Then by setting up the proxy distributions $p_{\theta}(\mathbf{x}|\mathbf{z})$ and $q_{\phi}(\mathbf{z}|\mathbf{x})$, we can train the encoder and decoder so that the decoder will serve the goal of generating images. But VAE is largely a *one-step* generation — if you give us a latent code \mathbf{z} , we ask the neural network $f_{\theta}(\cdot)$ to immediately return us the generated signal $\mathbf{x} \sim \mathcal{N}(\mathbf{x} | f_{\theta}(\mathbf{z}), \sigma_{\text{dec}}^2 \mathbf{I})$. In some sense, this is asking a lot from the neural network. We are asking it to use a few layers of neurons to immediately convert from one distribution $p(\mathbf{z})$ to another distribution $p(\mathbf{x})$. This is too much.

The idea Sohl-Dickstein et al proposed was to construct a chain of conversions instead of a one-step process. To this end they defined two processes analogous to the encoder and decoder in a VAE. They call the encoder as the forward process, and the decoder as the reverse process. In both processes, they consider a sequence of variables $\mathbf{x}_0, \dots, \mathbf{x}_T$ whose joint distribution is denoted as $q_{\phi}(\mathbf{x}_{0:T})$ and $p_{\theta}(\mathbf{x}_{0:T})$ respectively for the forward and reverse processes. To make both processes tractable (and also flexible), they impose a Markov chain structure (i.e., memoryless) where

$$\begin{aligned} \text{forward from } \mathbf{x}_0 \text{ to } \mathbf{x}_T : \quad q_{\phi}(\mathbf{x}_{0:T}) &= q(\mathbf{x}_0) \prod_{t=1}^T q_{\phi}(\mathbf{x}_t | \mathbf{x}_{t-1}), \\ \text{reverse from } \mathbf{x}_T \text{ to } \mathbf{x}_0 : \quad p_{\theta}(\mathbf{x}_{0:T}) &= p(\mathbf{x}_T) \prod_{t=1}^T p_{\theta}(\mathbf{x}_{t-1} | \mathbf{x}_t). \end{aligned}$$

In both equations, the transition distributions are only dependent on its immediate previous stage. Therefore, if each transition is realized through some form of neural networks, the overall generation process is broken down into many smaller tasks. It does not mean that we will need T times more neural networks. We are just re-using one network for T times.

Breaking the overall process into smaller steps allows us to use simple distributions at each step. As will be discussed in the following subsections, we can use Gaussian distributions for the transitions. Thanks to the properties of a Gaussian, the posterior will remain a Gaussian if the likelihood and the prior are both Gaussians. Therefore, if each transitional distribution above is a Gaussian, the joint distribution is also a Gaussian. Since a Gaussian is fully characterized by the first two moments (mean and variance), the computation is highly tractable. In the original paper of Sohl-Dickstein et al, there is also a case study of binomial diffusion processes.

After providing a high-level overview of the concepts, let's talk about some details. The starting point of the diffusion model is to consider the VAE structure and make it a chain of incremental updates as shown in

Figure 2.1. This particular structure is called the **variational diffusion model**, a name given by Kingma et al in 2021 [22]. The variational diffusion model has a sequence of states $\mathbf{x}_0, \mathbf{x}_1, \dots, \mathbf{x}_T$ with the following interpretations:

- \mathbf{x}_0 : It is the original image, which is the same as \mathbf{x} in VAE.
- \mathbf{x}_T : It is the latent variable, which is the same as \mathbf{z} in VAE. As explained above, we choose $\mathbf{x}_T \sim \mathcal{N}(0, \mathbf{I})$ for simplicity, tractability, and computational efficiency.
- $\mathbf{x}_1, \dots, \mathbf{x}_{T-1}$: They are the intermediate states. They are also the latent variables, but they are not white Gaussian.

The structure of the variational diffusion model consists of two paths. The forward and the reverse paths are analogous to the paths of a single-step variational autoencoder. The difference is that the encoders and decoders have identical input-output dimensions. The assembly of all the forward building blocks will give us the encoder, and the assembly of all the reverse building blocks will give us the decoder.

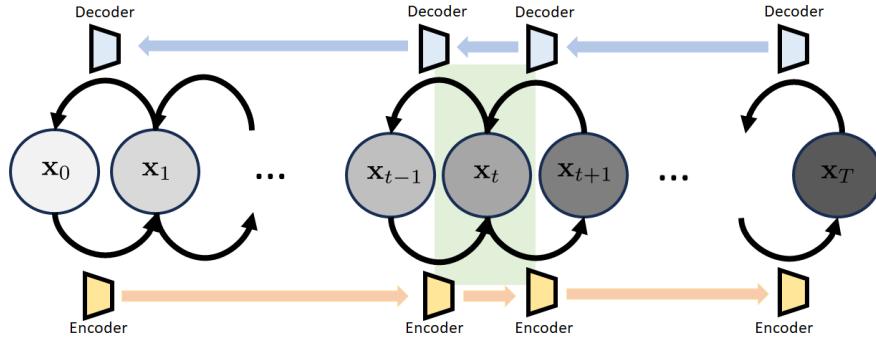


Figure 2.1: Variational diffusion model by Kingma et al [22]. In this model, the input image is \mathbf{x}_0 and the white noise is \mathbf{x}_T . The intermediate variables (or states) $\mathbf{x}_1, \dots, \mathbf{x}_{T-1}$ are latent variables. The transition from \mathbf{x}_{t-1} to \mathbf{x}_t is analogous to the forward step (encoder) in VAE, whereas the transition from \mathbf{x}_t to \mathbf{x}_{t-1} is analogous to the reverse step (decoder) in VAE. In variational diffusion models, the input dimension and the output dimension of the encoders/decoders are identical.

2.1 Building Blocks

Let's talk about the building blocks of the variational diffusion model. There are three classes of building blocks: the transition block, the initial block, and the final block.

Transition Block The t -th transition block consists of three states \mathbf{x}_{t-1} , \mathbf{x}_t , and \mathbf{x}_{t+1} . There are two possible paths to get to state \mathbf{x}_t , as illustrated in Figure 2.2.

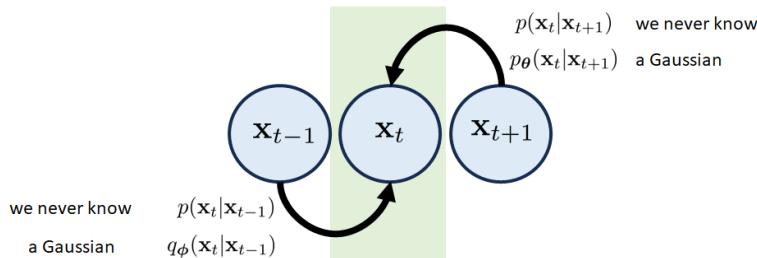


Figure 2.2: The transition block of a variational diffusion model consists of three nodes. The transition distributions $p(\mathbf{x}_t | \mathbf{x}_{t+1})$ and $p(\mathbf{x}_t | \mathbf{x}_{t-1})$ are not accessible, but we can approximate them by Gaussians.

- The first path is the forward transition going from \mathbf{x}_{t-1} to \mathbf{x}_t . The associated transition distribution is $p(\mathbf{x}_t|\mathbf{x}_{t-1})$. In plain words, if you tell us \mathbf{x}_{t-1} , we can draw a sample \mathbf{x}_t according to $p(\mathbf{x}_t|\mathbf{x}_{t-1})$. However, just like a VAE, the transition distribution $p(\mathbf{x}_t|\mathbf{x}_{t-1})$ is not accessible. We can approximate it by some simple distributions such as a Gaussian. The approximated distribution is denoted as $q_\phi(\mathbf{x}_t|\mathbf{x}_{t-1})$. We will discuss the exact form of q_ϕ later.
- The second path is the reverse transition going from \mathbf{x}_{t+1} to \mathbf{x}_t . Again, we do not know $p(\mathbf{x}_t|\mathbf{x}_{t+1})$ and so we have another proxy distribution, e.g. a Gaussian, to approximate the true distribution. This proxy distribution is denoted as $p_\theta(\mathbf{x}_t|\mathbf{x}_{t+1})$.

Initial Block The initial block of the variational diffusion model focuses on the state \mathbf{x}_0 . Since we start at \mathbf{x}_0 , we only need the reverse transition from \mathbf{x}_1 to \mathbf{x}_0 . The forward transition from \mathbf{x}_{-1} to \mathbf{x}_0 can be dropped. Therefore, we only need to consider $p(\mathbf{x}_0|\mathbf{x}_1)$. But since $p(\mathbf{x}_0|\mathbf{x}_1)$ is never accessible, we approximate it by a Gaussian $p_\theta(\mathbf{x}_0|\mathbf{x}_1)$ where the mean is computed through a neural network. See Figure 2.3 for illustration.

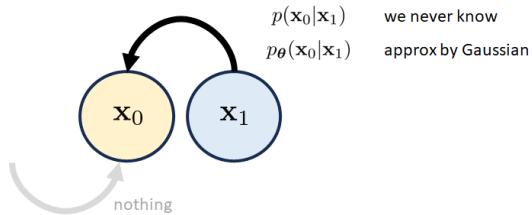


Figure 2.3: The initial block of a variational diffusion model focuses on the node \mathbf{x}_0 . Since there is no state before time $t = 0$, we only have a reverse transition from \mathbf{x}_1 to \mathbf{x}_0 .

Final Block. The final block focuses on the state \mathbf{x}_T . Remember that \mathbf{x}_T is supposed to be our final latent variable which is a white Gaussian noise vector. Because it is the final block, we only need a forward transition from \mathbf{x}_{T-1} to \mathbf{x}_T , and nothing such as \mathbf{x}_{T+1} to \mathbf{x}_T . The forward transition is approximated by $q_\phi(\mathbf{x}_T|\mathbf{x}_{T-1})$ which is a Gaussian. See Figure 2.4 for illustration.

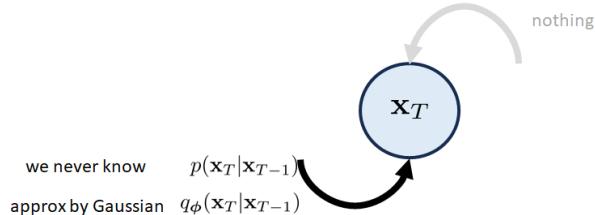


Figure 2.4: The final block of a variational diffusion model focuses on the node \mathbf{x}_T . Since there is no state after time $t = T$, we only have a forward transition from \mathbf{x}_{T-1} to \mathbf{x}_T .

Understanding the Transition Distribution. Before we proceed further, we need to explain the transition distribution $q_\phi(\mathbf{x}_t|\mathbf{x}_{t-1})$. We know that it is a Gaussian. But what is the mean and variance of this Gaussian?

Definition 2.1. Transition Distribution $q_\phi(\mathbf{x}_t|\mathbf{x}_{t-1})$. In a variational diffusion model (and also DDPM which we will discuss later), the transition distribution $q_\phi(\mathbf{x}_t|\mathbf{x}_{t-1})$ is defined as

$$q_\phi(\mathbf{x}_t|\mathbf{x}_{t-1}) \stackrel{\text{def}}{=} \mathcal{N}(\mathbf{x}_t | \sqrt{\alpha_t} \mathbf{x}_{t-1}, (1 - \alpha_t)\mathbf{I}). \quad (2.1)$$

In other words, $q_\phi(\mathbf{x}_t|\mathbf{x}_{t-1})$ is a Gaussian. The mean is $\sqrt{\alpha_t} \mathbf{x}_{t-1}$ and the variance is $1 - \alpha_t$. The choice of the scaling factor $\sqrt{\alpha_t}$ is to make sure that the variance magnitude is preserved so that it will not explode and vanish after many iterations.

Example 2.1. Let's consider a Gaussian mixture model

$$\mathbf{x}_0 \sim p_0(\mathbf{x}) = \pi_1 \mathcal{N}(\mathbf{x} | \mu_1, \sigma_1^2) + \pi_2 \mathcal{N}(\mathbf{x} | \mu_2, \sigma_2^2).$$

Given the transition probability, we know that if $\mathbf{x}_t \sim q_\phi(\mathbf{x}_t | \mathbf{x}_{t-1})$ then

$$\mathbf{x}_t = \sqrt{\alpha_t} \mathbf{x}_{t-1} + \sqrt{(1 - \alpha_t)} \boldsymbol{\epsilon}, \quad \text{where } \boldsymbol{\epsilon} \sim \mathcal{N}(0, \mathbf{I}).$$

Our goal is to see whether this iterative procedure (using the above transition probability) will give us a white Gaussian in the equilibrium state (i.e., when $t \rightarrow \infty$).

For a mixture model, it is not difficult to show that the probability distribution of \mathbf{x}_t can be calculated recursively via the algorithm for $t = 1, 2, \dots, T$: (the proof will be shown later)

$$\begin{aligned} \mathbf{x}_t \sim p_t(\mathbf{x}) &= \pi_1 \mathcal{N}(\mathbf{x} | \sqrt{\alpha_t} \mu_{1,t-1}, \alpha_t \sigma_{1,t-1}^2 + (1 - \alpha_t)) \\ &\quad + \pi_2 \mathcal{N}(\mathbf{x} | \sqrt{\alpha_t} \mu_{2,t-1}, \alpha_t \sigma_{2,t-1}^2 + (1 - \alpha_t)), \end{aligned} \quad (2.2)$$

where $\mu_{1,t-1}$ is the mean for class 1 at time $t - 1$, with $\mu_{1,0} = \mu_1$ being the initial mean. Similarly, $\sigma_{1,t-1}^2$ is the variance for class 1 at time $t - 1$, with $\sigma_{1,0}^2 = \sigma_1^2$ being the initial variance.

In the figure below, we show a numerical example where $\pi_1 = 0.3$, $\pi_2 = 0.7$, $\mu_1 = -2$, $\mu_2 = 2$, $\sigma_1 = 0.2$, and $\sigma_2 = 1$. The rate is defined as $\alpha_t = 0.97$ for all t . We plot the probability distribution function for different t .

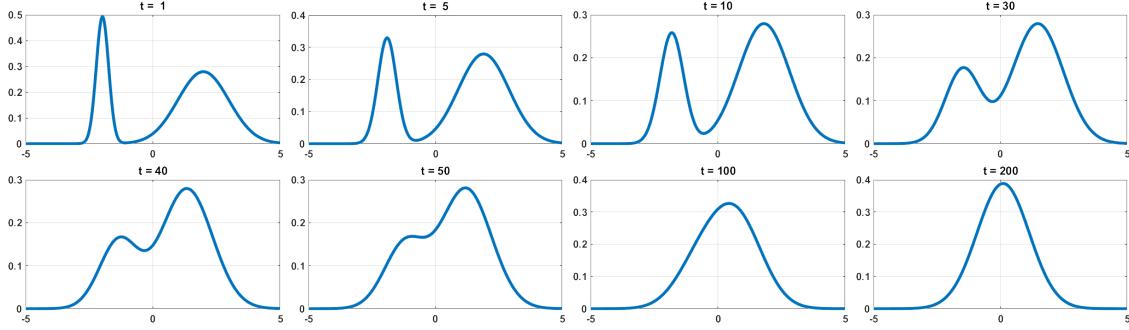


Figure 2.5: Evolution of the distribution $p_t(\mathbf{x})$. As time t progresses, the bimodal distribution gradually becomes a Gaussian.

Proof of Eqn (2.2). For those who would like to understand how we derive the probability density of a mixture model in Eqn (2.2), we can show a simple derivation. Consider a mixture model

$$p(\mathbf{x}) = \sum_{k=1}^K \pi_k \underbrace{\mathcal{N}(\mathbf{x} | \mu_k, \sigma_k^2 \mathbf{I})}_{p(\mathbf{x}|k)}.$$

If we consider a new variable $\mathbf{y} = \sqrt{\alpha} \mathbf{x} + \sqrt{1 - \alpha} \boldsymbol{\epsilon}$ where $\boldsymbol{\epsilon} \sim \mathcal{N}(0, \mathbf{I})$, then the distribution of \mathbf{y} can be derived by using the law of total probability:

$$p(\mathbf{y}) = \sum_{k=1}^K p(\mathbf{y}|k)p(k) = \sum_{k=1}^K \pi_k p(\mathbf{y}|k).$$

Since $\mathbf{y}|k = \sqrt{\alpha} \mathbf{x}|k + \sqrt{1 - \alpha} \boldsymbol{\epsilon}$ is a linear combination of a (conditioned) Gaussian random variable $\mathbf{x}|k$ and another Gaussian random variable $\boldsymbol{\epsilon}$, the sum $\mathbf{y}|k$ will remain as a Gaussian. The mean and

variance are

$$\begin{aligned}\mathbb{E}[\mathbf{y}|k] &= \sqrt{\alpha}\mathbb{E}[\mathbf{x}|k] + \sqrt{1-\alpha}\mathbb{E}[\boldsymbol{\epsilon}] = \sqrt{\alpha}\mu_k, \\ \text{Var}[\mathbf{y}|k] &= \alpha\text{Var}[\mathbf{x}|k] + (1-\alpha)\text{Var}[\boldsymbol{\epsilon}] = \alpha\sigma_k^2 + (1-\alpha),\end{aligned}$$

where we used the fact that $\mathbb{E}[\boldsymbol{\epsilon}] = 0$, and $\text{Var}[\boldsymbol{\epsilon}] = 1$. Since we just argued that $\mathbf{y}|k$ is a Gaussian, the distribution of $\mathbf{y}|k$ is completely specified once we know the mean and variance. Substituting the above derived results, we know that $p(\mathbf{y}|k) = \mathcal{N}(\mathbf{y}|\sqrt{\alpha}\mu_k, \alpha\sigma_k^2 + (1-\alpha))$. This completes the derivation.

The magical scalars $\sqrt{\alpha_t}$ and $1 - \alpha_t$. You may wonder how the genius people (the authors of the denoising diffusion papers) come up with the magical scalars $\sqrt{\alpha_t}$ and $(1 - \alpha_t)$ for the above transition probability. To demystify this, let's consider two unrelated scalars $a \in \mathbb{R}$ and $b \in \mathbb{R}$, and define the transition distribution as

$$q_{\phi}(\mathbf{x}_t | \mathbf{x}_{t-1}) = \mathcal{N}(\mathbf{x}_t | a\mathbf{x}_{t-1}, b^2\mathbf{I}). \quad (2.3)$$

Here is the finding:

Theorem 2.1. (Why $\sqrt{\alpha}$ and $1 - \alpha$?) Suppose that $q_{\phi}(\mathbf{x}_t | \mathbf{x}_{t-1}) = \mathcal{N}(\mathbf{x}_t | a\mathbf{x}_{t-1}, b^2\mathbf{I})$ for some constants a and b . If we want to choose a and b such that the distribution of \mathbf{x}_t will become $\mathcal{N}(0, \mathbf{I})$, then it is necessary that

$$a = \sqrt{\alpha} \quad \text{and} \quad b = \sqrt{1 - \alpha}.$$

Therefore, the transition distribution is

$$q_{\phi}(\mathbf{x}_t | \mathbf{x}_{t-1}) \stackrel{\text{def}}{=} \mathcal{N}(\mathbf{x}_t | \sqrt{\alpha}\mathbf{x}_{t-1}, (1 - \alpha)\mathbf{I}). \quad (2.4)$$

Remark: You can replace α by α_t , if you prefer a noise schedule.

Proof. We want to show that $a = \sqrt{\alpha}$ and $b = \sqrt{1 - \alpha}$. For the distribution shown in Eqn (2.3), the equivalent sampling step is:

$$\mathbf{x}_t = a\mathbf{x}_{t-1} + b\boldsymbol{\epsilon}_{t-1}, \quad \text{where} \quad \boldsymbol{\epsilon}_{t-1} \sim \mathcal{N}(0, \mathbf{I}). \quad (2.5)$$

We can carry on the recursion to show that

$$\begin{aligned}\mathbf{x}_t &= a\mathbf{x}_{t-1} + b\boldsymbol{\epsilon}_{t-1} \\ &= a(a\mathbf{x}_{t-2} + b\boldsymbol{\epsilon}_{t-2}) + b\boldsymbol{\epsilon}_{t-1} \quad (\text{substitute } \mathbf{x}_{t-1} = a\mathbf{x}_{t-2} + b\boldsymbol{\epsilon}_{t-2}) \\ &= a^2\mathbf{x}_{t-2} + ab\boldsymbol{\epsilon}_{t-2} + b\boldsymbol{\epsilon}_{t-1} \quad (\text{regroup terms}) \\ &= \vdots \\ &= a^t\mathbf{x}_0 + b\underbrace{[\boldsymbol{\epsilon}_{t-1} + a\boldsymbol{\epsilon}_{t-2} + a^2\boldsymbol{\epsilon}_{t-3} + \dots + a^{t-1}\boldsymbol{\epsilon}_0]}_{\stackrel{\text{def}}{=} \mathbf{w}_t}.\end{aligned} \quad (2.6)$$

The finite sum above is a sum of independent Gaussian random variables. The mean vector $\mathbb{E}[\mathbf{w}_t]$ remains zero because everyone has a zero mean. The covariance matrix (for a zero-mean vector) is

$$\begin{aligned}\text{Cov}[\mathbf{w}_t] &\stackrel{\text{def}}{=} \mathbb{E}[\mathbf{w}_t \mathbf{w}_t^T] \\ &= b^2(\text{Cov}(\boldsymbol{\epsilon}_{t-1}) + a^2\text{Cov}(\boldsymbol{\epsilon}_{t-2}) + \dots + (a^{t-1})^2\text{Cov}(\boldsymbol{\epsilon}_0)) \\ &= b^2(1 + a^2 + a^4 + \dots + a^{2(t-1)})\mathbf{I} \\ &= b^2 \cdot \frac{1 - a^{2t}}{1 - a^2} \mathbf{I}.\end{aligned}$$

As $t \rightarrow \infty$, $a^t \rightarrow 0$ for any $0 < a < 1$. Therefore, at the limit when $t = \infty$,

$$\lim_{t \rightarrow \infty} \text{Cov}[\mathbf{w}_t] = \frac{b^2}{1 - a^2} \mathbf{I}.$$

So, if we want $\lim_{t \rightarrow \infty} \text{Cov}[\mathbf{w}_t] = \mathbf{I}$ (so that the distribution of \mathbf{x}_t will approach $\mathcal{N}(0, \mathbf{I})$), then we need

$$1 = \frac{b^2}{1 - a^2},$$

or equivalently $b = \sqrt{1 - a^2}$. Now, if we let $a = \sqrt{\alpha}$, then $b = \sqrt{1 - \alpha}$. This will give us

$$\mathbf{x}_t = \sqrt{\alpha} \mathbf{x}_{t-1} + \sqrt{1 - \alpha} \boldsymbol{\epsilon}_{t-1}. \quad (2.7)$$

Distribution $q_\phi(\mathbf{x}_t | \mathbf{x}_0)$. With the understanding of the magical scalars, we can talk about the distribution $q_\phi(\mathbf{x}_t | \mathbf{x}_0)$. That is, we want to know how \mathbf{x}_t will be distributed if we are given \mathbf{x}_0 .

Theorem 2.2. (Conditional Distribution $q_\phi(\mathbf{x}_t | \mathbf{x}_0)$). The conditional distribution $q_\phi(\mathbf{x}_t | \mathbf{x}_0)$ is given by

$$q_\phi(\mathbf{x}_t | \mathbf{x}_0) = \mathcal{N}(\mathbf{x}_t | \sqrt{\bar{\alpha}_t} \mathbf{x}_0, (1 - \bar{\alpha}_t) \mathbf{I}), \quad (2.8)$$

where $\bar{\alpha}_t = \prod_{i=1}^t \alpha_i$.

Proof. To see how Eqn (2.8) is derived, we can re-do the recursion but this time we use $\sqrt{\alpha_t} \mathbf{x}_{t-1}$ and $(1 - \alpha_t) \mathbf{I}$ as the mean and covariance, respectively. This will give us

$$\begin{aligned} \mathbf{x}_t &= \sqrt{\alpha_t} \mathbf{x}_{t-1} + \sqrt{1 - \alpha_t} \boldsymbol{\epsilon}_{t-1} \\ &= \sqrt{\alpha_t} (\sqrt{\alpha_{t-1}} \mathbf{x}_{t-2} + \sqrt{1 - \alpha_{t-1}} \boldsymbol{\epsilon}_{t-2}) + \sqrt{1 - \alpha_t} \boldsymbol{\epsilon}_{t-1} \\ &= \sqrt{\alpha_t \alpha_{t-1}} \mathbf{x}_{t-2} + \underbrace{\sqrt{\alpha_t} \sqrt{1 - \alpha_{t-1}} \boldsymbol{\epsilon}_{t-2} + \sqrt{1 - \alpha_t} \boldsymbol{\epsilon}_{t-1}}_{\mathbf{w}_1}. \end{aligned} \quad (2.9)$$

Therefore, we have a sum of two Gaussians. But since sum of two Gaussians remains a Gaussian, we can just calculate its new covariance (because the mean remains zero). The new covariance is

$$\begin{aligned} \mathbb{E}[\mathbf{w}_1 \mathbf{w}_1^T] &= [(\sqrt{\alpha_t} \sqrt{1 - \alpha_{t-1}})^2 + (\sqrt{1 - \alpha_t})^2] \mathbf{I} \\ &= [\alpha_t(1 - \alpha_{t-1}) + 1 - \alpha_t] \mathbf{I} = [1 - \alpha_t \alpha_{t-1}] \mathbf{I}. \end{aligned}$$

Returning to Eqn (2.9), we can show that the recursion becomes a linear combination of \mathbf{x}_{t-2} and a noise vector $\boldsymbol{\epsilon}_{t-2}$:

$$\begin{aligned} \mathbf{x}_t &= \sqrt{\alpha_t \alpha_{t-1}} \mathbf{x}_{t-2} + \sqrt{1 - \alpha_t \alpha_{t-1}} \boldsymbol{\epsilon}_{t-2} \\ &= \sqrt{\alpha_t \alpha_{t-1} \alpha_{t-2}} \mathbf{x}_{t-3} + \sqrt{1 - \alpha_t \alpha_{t-1} \alpha_{t-2}} \boldsymbol{\epsilon}_{t-3} \\ &= \vdots \\ &= \left(\sqrt{\prod_{i=1}^t \alpha_i} \right) \mathbf{x}_0 + \left(\sqrt{1 - \prod_{i=1}^t \alpha_i} \right) \boldsymbol{\epsilon}_0. \end{aligned} \quad (2.10)$$

So, if we define $\bar{\alpha}_t = \prod_{i=1}^t \alpha_i$, we can show that

$$\mathbf{x}_t = \sqrt{\bar{\alpha}_t} \mathbf{x}_0 + \sqrt{1 - \bar{\alpha}_t} \boldsymbol{\epsilon}_0. \quad (2.11)$$

In other words, the distribution $q_\phi(\mathbf{x}_t | \mathbf{x}_0)$ is

$$\mathbf{x}_t \sim q_\phi(\mathbf{x}_t | \mathbf{x}_0) = \mathcal{N}(\mathbf{x}_t | \sqrt{\bar{\alpha}_t} \mathbf{x}_0, (1 - \bar{\alpha}_t) \mathbf{I}). \quad (2.12)$$

The utility of the new distribution $q_\phi(\mathbf{x}_t | \mathbf{x}_0)$ is its one-shot forward diffusion step compared to the chain $\mathbf{x}_0 \rightarrow \mathbf{x}_1 \rightarrow \dots \rightarrow \mathbf{x}_{T-1} \rightarrow \mathbf{x}_T$. In every step of the forward diffusion model, since we already know \mathbf{x}_0 and we assume that all subsequence transitions are Gaussian, we will know \mathbf{x}_t for any t . The situation can be understood from Figure 2.6.

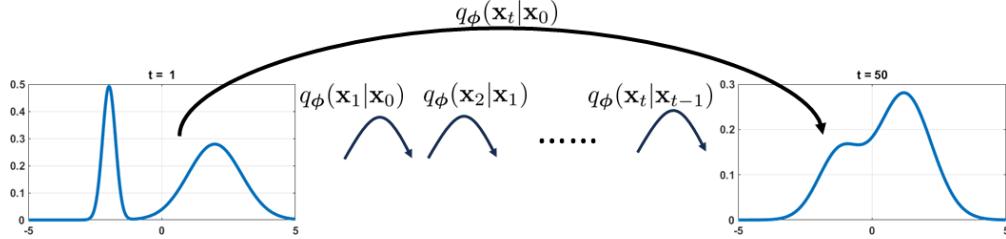


Figure 2.6: The difference between $q_\phi(\mathbf{x}_t | \mathbf{x}_{t-1})$ and $q_\phi(\mathbf{x}_t | \mathbf{x}_0)$.

Example 2.2. For a Gaussian mixture model such that $\mathbf{x}_0 \sim p_0(\mathbf{x}) = \sum_{k=1}^K \pi_k \mathcal{N}(\mathbf{x} | \boldsymbol{\mu}_k, \sigma_k^2 \mathbf{I})$, we can show that the distribution at time t is

$$\begin{aligned} \mathbf{x}_t \sim p_t(\mathbf{x}) &= \sum_{k=1}^K \pi_k \mathcal{N}(\mathbf{x} | \sqrt{\bar{\alpha}_t} \boldsymbol{\mu}_k, (1 - \bar{\alpha}_t) \mathbf{I} + \bar{\alpha}_t \sigma_k^2 \mathbf{I}) \\ &= \sum_{k=1}^K \pi_k \mathcal{N}(\mathbf{x} | \sqrt{\alpha^t} \boldsymbol{\mu}_k, (1 - \alpha^t) \mathbf{I} + \alpha^t \sigma_k^2 \mathbf{I}), \quad \text{if } \alpha_t = \alpha \text{ so that } \bar{\alpha}_t = \prod_{i=1}^t \alpha = \alpha^t. \end{aligned} \quad (2.13)$$

If you are curious about how the probability distribution p_t evolves over time t , we can visualize the trajectory of a Gaussian mixture distribution we discussed in Example 2.1. We use Eqn (2.13) to plot the heatmap. You can see that when $t = 0$, the initial distribution is a mixture of two Gaussians. As we progress by following the transition defined in Eqn (2.13), we can see that the distribution gradually becomes the single Gaussian $\mathcal{N}(0, \mathbf{I})$.

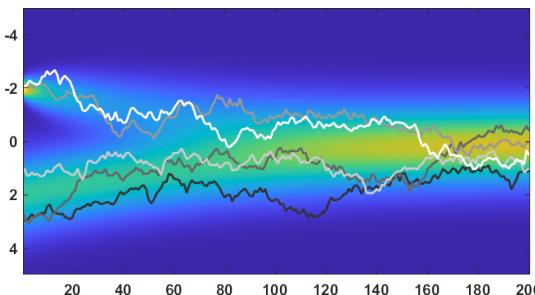


Figure 2.7: Realizations of random trajectories made by \mathbf{x}_t . The color map in the background indicates the probability distribution $p_t(\mathbf{x})$.

In the same plot, we overlay and show a few instantaneous trajectories of the random samples \mathbf{x}_t as a function of time t . The equation we used to generate the samples is

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

As you can see, the trajectories of \mathbf{x}_t more or less follow the distribution $p_t(\mathbf{x})$.

A confusing point to many readers is that if the goal is to convert from an image $p(\mathbf{x}_0)$ to white noise $p(\mathbf{x}_T)$, what is the point of deriving $q_\phi(\mathbf{x}_t | \mathbf{x}_{t-1})$ and $q_\phi(\mathbf{x}_t | \mathbf{x}_0)$? The answer is that so far we have been just talking about the *forward* process. In a diffusion model, the forward process is chosen such that they can be expressed in a closed form. The more interesting part is the *reverse* process. As will be discussed, the reverse process is realized through a chain of denoising operations. Each denoising step should be coupled with the corresponding step in the forward process. $q_\phi(\mathbf{x}_t | \mathbf{x}_0)$ just provides us a slightly more convenient way to implement the forward process.

2.2 Evidence Lower Bound

Now that we understand the structure of the variational diffusion model, we can write down the ELBO and hence train the model.

Theorem 2.3. (ELBO for Variational Diffusion Model). The ELBO for the variational diffusion model is

$$\begin{aligned} \text{ELBO}_{\phi, \theta}(\mathbf{x}) &= \mathbb{E}_{q_\phi(\mathbf{x}_1 | \mathbf{x}_0)} \left[\log \underbrace{p_\theta(\mathbf{x}_0 | \mathbf{x}_1)}_{\text{how good the initial block is}} \right] \\ &\quad - \mathbb{E}_{q_\phi(\mathbf{x}_{T-1} | \mathbf{x}_0)} \left[\underbrace{\mathbb{D}_{\text{KL}}(q_\phi(\mathbf{x}_T | \mathbf{x}_{T-1}) \| p(\mathbf{x}_T))}_{\text{how good the final block is}} \right] \\ &\quad - \sum_{t=1}^{T-1} \mathbb{E}_{q_\phi(\mathbf{x}_{t-1}, \mathbf{x}_{t+1} | \mathbf{x}_0)} \left[\underbrace{\mathbb{D}_{\text{KL}}(q_\phi(\mathbf{x}_t | \mathbf{x}_{t-1}) \| p_\theta(\mathbf{x}_t | \mathbf{x}_{t+1}))}_{\text{how good the transition blocks are}} \right], \end{aligned} \quad (2.14)$$

where $\mathbf{x}_0 = \mathbf{x}$, and $\mathbf{x}_T \sim \mathcal{N}(0, \mathbf{I})$.

If you are a casual reader of this tutorial, we hope that this equation does not throw you off. While it appears a monster, it does have structures. We just need to be patient when we try to understand it.

Reconstruction (Initial Block). Let's first look at the term

$$\mathbb{E}_{q_\phi(\mathbf{x}_1 | \mathbf{x}_0)} \left[\log p_\theta(\mathbf{x}_0 | \mathbf{x}_1) \right].$$

This term is based on the initial block and it is analogous to Eqn (1.10). The subject inside the expectation is the log-likelihood $\log p_\theta(\mathbf{x}_0 | \mathbf{x}_1)$. This log-likelihood measures how good the neural network (associated with p_θ) can recover \mathbf{x}_0 from the latent variable \mathbf{x}_1 .

The expectation is taken with respect to the samples drawn from $q_\phi(\mathbf{x}_1 | \mathbf{x}_0)$. Recall that $q_\phi(\mathbf{x}_1 | \mathbf{x}_0)$ is the distribution that generates \mathbf{x}_1 . We require \mathbf{x}_1 to be drawn from this distribution because \mathbf{x}_1 do not come from the sky but *created* by the forward transition $q_\phi(\mathbf{x}_1 | \mathbf{x}_0)$. The conditioning on \mathbf{x}_0 is needed here because we need to know what the original image is.

The reason why expectation is used here is that $p_\theta(\mathbf{x}_0 | \mathbf{x}_1)$ is a function of \mathbf{x}_1 (and so if \mathbf{x}_1 is random then $p_\theta(\mathbf{x}_0 | \mathbf{x}_1)$ is random too). For a different intermediate state \mathbf{x}_1 , the probability $p_\theta(\mathbf{x}_0 | \mathbf{x}_1)$ will be different. The expectation eliminates the dependency on \mathbf{x}_1 .

Prior Matching (Final Block). The prior matching term is

$$-\mathbb{E}_{q_\phi(\mathbf{x}_{T-1} | \mathbf{x}_0)} \left[\mathbb{D}_{\text{KL}}(q_\phi(\mathbf{x}_T | \mathbf{x}_{T-1}) \| p(\mathbf{x}_T)) \right], \quad (2.15)$$

and it is based on the final block. We use the KL divergence to measure the difference between $q_\phi(\mathbf{x}_T | \mathbf{x}_{T-1})$ and $p(\mathbf{x}_T)$. The distribution $q_\phi(\mathbf{x}_T | \mathbf{x}_{T-1})$ is the forward transition from \mathbf{x}_{T-1} to \mathbf{x}_T . This describes how \mathbf{x}_T is generated. The second distribution is $p(\mathbf{x}_T)$. Because of our laziness, we assume that $p(\mathbf{x}_T) = \mathcal{N}(0, \mathbf{I})$. We want $q_\phi(\mathbf{x}_T | \mathbf{x}_{T-1})$ to be as close to $\mathcal{N}(0, \mathbf{I})$ as possible.

When computing the KL-divergence, the variable \mathbf{x}_T is a dummy variable. However, since q_ϕ is conditioned on \mathbf{x}_{T-1} , the KL-divergence calculated here is a function of the conditioned variable \mathbf{x}_{T-1} . Where

does \mathbf{x}_{T-1} come from? It is generated by $q_\phi(\mathbf{x}_{T-1}|\mathbf{x}_0)$. We use a conditional distribution $q_\phi(\mathbf{x}_{T-1}|\mathbf{x}_0)$ because \mathbf{x}_{T-1} depends on what \mathbf{x}_0 we use in the first place. The expectation over $q_\phi(\mathbf{x}_{T-1}|\mathbf{x}_0)$ says that for each of the \mathbf{x}_{T-1} generated by $q_\phi(\mathbf{x}_{T-1}|\mathbf{x}_0)$, we will have a value of the KL divergence. We take the expectation over all the possible \mathbf{x}_{T-1} generated to eliminate the dependency.

Consistency. (Transition Blocks) The consistency term is

$$-\sum_{t=1}^{T-1} \mathbb{E}_{q_\phi(\mathbf{x}_{t-1}, \mathbf{x}_{t+1}|\mathbf{x}_0)} \left[\mathbb{D}_{\text{KL}} \left(q_\phi(\mathbf{x}_t|\mathbf{x}_{t-1}) \| p_\theta(\mathbf{x}_t|\mathbf{x}_{t+1}) \right) \right], \quad (2.16)$$

and it is based on the transition blocks. There are two directions if you recall Figure 2.2. The forward transition is determined by the distribution $q_\phi(\mathbf{x}_t|\mathbf{x}_{t-1})$ whereas the reverse transition is determined by another distribution $p_\theta(\mathbf{x}_t|\mathbf{x}_{t+1})$. The consistency term uses the KL divergence to measure the deviation.

The expectation is taken with respect to the pair of samples $(\mathbf{x}_{t-1}, \mathbf{x}_{t+1})$, drawn from $q_\phi(\mathbf{x}_{t-1}, \mathbf{x}_{t+1}|\mathbf{x}_0)$. The reason is that the KL divergence above is a function of \mathbf{x}_{t-1} and \mathbf{x}_{t+1} . (You can ignore \mathbf{x}_t because it is a dummy variable that will be eliminated during the integration process when we calculate the expectation.) Because of the dependencies on \mathbf{x}_{t-1} and \mathbf{x}_{t+1} , we need to take the expectation.

Proof of Theorem 2.3. Let's define the following notation: $\mathbf{x}_{0:T} = \{\mathbf{x}_0, \dots, \mathbf{x}_T\}$ means the collection of all state variables from $t = 0$ to $t = T$. We also recall that the prior distribution $p(\mathbf{x})$ is the distribution for the image \mathbf{x}_0 . So it is equivalent to $p(\mathbf{x}_0)$. With these in mind, we can show that

$$\begin{aligned} \log p(\mathbf{x}) &= \log p(\mathbf{x}_0) \\ &= \log \int p(\mathbf{x}_{0:T}) d\mathbf{x}_{1:T} && \text{(Marginalize by integrating over } \mathbf{x}_{1:T} \text{)} \\ &= \log \int p(\mathbf{x}_{0:T}) \frac{q_\phi(\mathbf{x}_{1:T}|\mathbf{x}_0)}{q_\phi(\mathbf{x}_{1:T}|\mathbf{x}_0)} d\mathbf{x}_{1:T} && \text{(Multiply and divide } q_\phi(\mathbf{x}_{1:T}|\mathbf{x}_0) \text{)} \\ &= \log \int q_\phi(\mathbf{x}_{1:T}|\mathbf{x}_0) \left[\frac{p(\mathbf{x}_{0:T})}{q_\phi(\mathbf{x}_{1:T}|\mathbf{x}_0)} \right] d\mathbf{x}_{1:T} && \text{(Rearrange terms)} \\ &= \log \mathbb{E}_{q_\phi(\mathbf{x}_{1:T}|\mathbf{x}_0)} \left[\frac{p(\mathbf{x}_{0:T})}{q_\phi(\mathbf{x}_{1:T}|\mathbf{x}_0)} \right] && \text{(Definition of expectation).} \end{aligned}$$

Now, we need to use Jensen's inequality, which states that for any random variable X and any concave function f , it holds that $f(\mathbb{E}[X]) \geq \mathbb{E}[f(X)]$. By recognizing that $f(\cdot) = \log(\cdot)$, we can show that

$$\log p(\mathbf{x}) = \log \mathbb{E}_{q_\phi(\mathbf{x}_{1:T}|\mathbf{x}_0)} \left[\frac{p(\mathbf{x}_{0:T})}{q_\phi(\mathbf{x}_{1:T}|\mathbf{x}_0)} \right] \geq \mathbb{E}_{q_\phi(\mathbf{x}_{1:T}|\mathbf{x}_0)} \left[\log \frac{p(\mathbf{x}_{0:T})}{q_\phi(\mathbf{x}_{1:T}|\mathbf{x}_0)} \right] \quad (2.17)$$

Let's take a closer look at $p(\mathbf{x}_{0:T})$. Inspecting Figure 2.2, we notice that if we want to decouple $p(\mathbf{x}_{0:T})$, we should do conditioning for $\mathbf{x}_{t-1}|\mathbf{x}_t$. This leads to:

$$p(\mathbf{x}_{0:T}) = p(\mathbf{x}_T) \prod_{t=1}^T p(\mathbf{x}_{t-1}|\mathbf{x}_t) = p(\mathbf{x}_T)p(\mathbf{x}_0|\mathbf{x}_1) \prod_{t=2}^T p(\mathbf{x}_{t-1}|\mathbf{x}_t). \quad (2.18)$$

As for $q_\phi(\mathbf{x}_{1:T}|\mathbf{x}_0)$, Figure 2.2 suggests that we need to do the conditioning for $\mathbf{x}_t|\mathbf{x}_{t-1}$. However, because of the sequential relationship, we can write

$$q_\phi(\mathbf{x}_{1:T}|\mathbf{x}_0) = \prod_{t=1}^T q_\phi(\mathbf{x}_t|\mathbf{x}_{t-1}) = q_\phi(\mathbf{x}_T|\mathbf{x}_{T-1}) \prod_{t=1}^{T-1} q_\phi(\mathbf{x}_t|\mathbf{x}_{t-1}). \quad (2.19)$$

Substituting Eqn (2.18) and Eqn (2.19) back to Eqn (2.17), we can show that

$$\begin{aligned}
\log p(\mathbf{x}) &\geq \mathbb{E}_{q_\phi(\mathbf{x}_{1:T}|\mathbf{x}_0)} \left[\log \frac{p(\mathbf{x}_{0:T})}{q_\phi(\mathbf{x}_{1:T}|\mathbf{x}_0)} \right] \\
&= \mathbb{E}_{q_\phi(\mathbf{x}_{1:T}|\mathbf{x}_0)} \left[\log \frac{p(\mathbf{x}_T)p(\mathbf{x}_0|\mathbf{x}_1)\prod_{t=2}^T p(\mathbf{x}_{t-1}|\mathbf{x}_t)}{q_\phi(\mathbf{x}_T|\mathbf{x}_{T-1})\prod_{t=1}^{T-1} q_\phi(\mathbf{x}_t|\mathbf{x}_{t-1})} \right] \\
&= \mathbb{E}_{q_\phi(\mathbf{x}_{1:T}|\mathbf{x}_0)} \left[\log \frac{p(\mathbf{x}_T)p(\mathbf{x}_0|\mathbf{x}_1)\prod_{t=1}^{T-1} p(\mathbf{x}_t|\mathbf{x}_{t+1})}{q_\phi(\mathbf{x}_T|\mathbf{x}_{T-1})\prod_{t=1}^{T-1} q_\phi(\mathbf{x}_t|\mathbf{x}_{t-1})} \right] \quad (\text{shift } t \text{ to } t+1) \\
&= \mathbb{E}_{q_\phi(\mathbf{x}_{1:T}|\mathbf{x}_0)} \left[\log \frac{p(\mathbf{x}_T)p(\mathbf{x}_0|\mathbf{x}_1)}{q_\phi(\mathbf{x}_T|\mathbf{x}_{T-1})} \right] + \mathbb{E}_{q_\phi(\mathbf{x}_{1:T}|\mathbf{x}_0)} \left[\log \prod_{t=1}^{T-1} \frac{p(\mathbf{x}_t|\mathbf{x}_{t+1})}{q_\phi(\mathbf{x}_t|\mathbf{x}_{t-1})} \right] \quad (\text{split expectation})
\end{aligned}$$

The first term above can be further decomposed into two expectations

$$\mathbb{E}_{q_\phi(\mathbf{x}_{1:T}|\mathbf{x}_0)} \left[\log \frac{p(\mathbf{x}_T)p(\mathbf{x}_0|\mathbf{x}_1)}{q_\phi(\mathbf{x}_T|\mathbf{x}_{T-1})} \right] = \underbrace{\mathbb{E}_{q_\phi(\mathbf{x}_{1:T}|\mathbf{x}_0)} \left[\log p(\mathbf{x}_0|\mathbf{x}_1) \right]}_{\text{Reconstruction}} + \underbrace{\mathbb{E}_{q_\phi(\mathbf{x}_{1:T}|\mathbf{x}_0)} \left[\log \frac{p(\mathbf{x}_T)}{q_\phi(\mathbf{x}_T|\mathbf{x}_{T-1})} \right]}_{\text{Prior Matching}}.$$

The Reconstruction term can be simplified as

$$\mathbb{E}_{q_\phi(\mathbf{x}_{1:T}|\mathbf{x}_0)} \left[\log p(\mathbf{x}_0|\mathbf{x}_1) \right] = \mathbb{E}_{q_\phi(\mathbf{x}_1|\mathbf{x}_0)} \left[\log p(\mathbf{x}_0|\mathbf{x}_1) \right],$$

where we used the fact that the conditioning $\mathbf{x}_{1:T}|\mathbf{x}_0$ is equivalent to $\mathbf{x}_1|\mathbf{x}_0$ when the subject of interest (i.e., $\log p(\mathbf{x}_0|\mathbf{x}_1)$) only involves \mathbf{x}_0 and \mathbf{x}_1 .

The Prior Matching term is

$$\mathbb{E}_{q_\phi(\mathbf{x}_{1:T}|\mathbf{x}_0)} \left[\log \frac{p(\mathbf{x}_T)}{q_\phi(\mathbf{x}_T|\mathbf{x}_{T-1})} \right] = \mathbb{E}_{q_\phi(\mathbf{x}_T, \mathbf{x}_{T-1}|\mathbf{x}_0)} \left[\log \frac{p(\mathbf{x}_T)}{q_\phi(\mathbf{x}_T|\mathbf{x}_{T-1})} \right],$$

where we note that the conditional expectation can be simplified to samples \mathbf{x}_T and \mathbf{x}_{T-1} only, because $\log \frac{p(\mathbf{x}_T)}{q_\phi(\mathbf{x}_T|\mathbf{x}_{T-1})}$ only depends on \mathbf{x}_T and \mathbf{x}_{T-1} . For the expectation term, chain rule of probability tells us that $q_\phi(\mathbf{x}_T, \mathbf{x}_{T-1}|\mathbf{x}_0) = q_\phi(\mathbf{x}_T|\mathbf{x}_{T-1}, \mathbf{x}_0)q_\phi(\mathbf{x}_{T-1}|\mathbf{x}_0)$. Since q_ϕ is Markovian, we can further write $q_\phi(\mathbf{x}_T|\mathbf{x}_{T-1}, \mathbf{x}_0) = q_\phi(\mathbf{x}_T|\mathbf{x}_{T-1})$. Therefore, the joint expectation $\mathbb{E}_{q_\phi(\mathbf{x}_T, \mathbf{x}_{T-1}|\mathbf{x}_0)}$ can be written as a product of two expectations $\mathbb{E}_{q_\phi(\mathbf{x}_{T-1}|\mathbf{x}_0)}\mathbb{E}_{q_\phi(\mathbf{x}_T|\mathbf{x}_{T-1})}$. This will give us

$$\begin{aligned}
\mathbb{E}_{q_\phi(\mathbf{x}_T, \mathbf{x}_{T-1}|\mathbf{x}_0)} \left[\log \frac{p(\mathbf{x}_T)}{q_\phi(\mathbf{x}_T|\mathbf{x}_{T-1})} \right] &= \mathbb{E}_{q_\phi(\mathbf{x}_{T-1}|\mathbf{x}_0)} \left\{ \mathbb{E}_{q_\phi(\mathbf{x}_T|\mathbf{x}_{T-1})} \left[\log \frac{p(\mathbf{x}_T)}{q_\phi(\mathbf{x}_T|\mathbf{x}_{T-1})} \right] \right\} \\
&= -\mathbb{E}_{q_\phi(\mathbf{x}_{T-1}|\mathbf{x}_0)} \left[\mathbb{D}_{\text{KL}}(q_\phi(\mathbf{x}_T|\mathbf{x}_{T-1}) \| p(\mathbf{x}_T)) \right].
\end{aligned}$$

Finally, we look at the product term. We can show that

$$\begin{aligned}
\mathbb{E}_{q_\phi(\mathbf{x}_{1:T}|\mathbf{x}_0)} \left[\log \prod_{t=1}^{T-1} \frac{p(\mathbf{x}_t|\mathbf{x}_{t+1})}{q_\phi(\mathbf{x}_t|\mathbf{x}_{t-1})} \right] &= \sum_{t=1}^{T-1} \mathbb{E}_{q_\phi(\mathbf{x}_{1:T}|\mathbf{x}_0)} \left[\log \frac{p(\mathbf{x}_t|\mathbf{x}_{t+1})}{q_\phi(\mathbf{x}_t|\mathbf{x}_{t-1})} \right] \\
&= \sum_{t=1}^{T-1} \mathbb{E}_{q_\phi(\mathbf{x}_{t-1}, \mathbf{x}_t, \mathbf{x}_{t+1}|\mathbf{x}_0)} \left[\log \frac{p(\mathbf{x}_t|\mathbf{x}_{t+1})}{q_\phi(\mathbf{x}_t|\mathbf{x}_{t-1})} \right],
\end{aligned}$$

where again we use the fact the expectation only needs \mathbf{x}_{t-1} , \mathbf{x}_t , and \mathbf{x}_{t+1} . Then, by using the same

conditional independence argument, we can show that

$$\begin{aligned} \sum_{t=1}^{T-1} \mathbb{E}_{q_\phi(\mathbf{x}_{t-1}, \mathbf{x}_t, \mathbf{x}_{t+1} | \mathbf{x}_0)} \left[\log \frac{p(\mathbf{x}_t | \mathbf{x}_{t+1})}{q_\phi(\mathbf{x}_t | \mathbf{x}_{t-1})} \right] &= \sum_{t=1}^{T-1} \mathbb{E}_{q_\phi(\mathbf{x}_{t-1}, \mathbf{x}_{t+1} | \mathbf{x}_0)} \left\{ \mathbb{E}_{q_\phi(\mathbf{x}_t | \mathbf{x}_0)} \left[\log \frac{p(\mathbf{x}_t | \mathbf{x}_{t+1})}{q_\phi(\mathbf{x}_t | \mathbf{x}_{t-1})} \right] \right\} \\ &= - \sum_{t=1}^{T-1} \mathbb{E}_{q_\phi(\mathbf{x}_{t-1}, \mathbf{x}_{t+1} | \mathbf{x}_0)} \left[\mathbb{D}_{\text{KL}}(q_\phi(\mathbf{x}_t | \mathbf{x}_{t-1}) \| p(\mathbf{x}_t | \mathbf{x}_{t+1})) \right]. \end{aligned}$$

By replacing $p(\mathbf{x}_0 | \mathbf{x}_1)$ with $p_\theta(\mathbf{x}_0 | \mathbf{x}_1)$ and $p(\mathbf{x}_t | \mathbf{x}_{t+1})$ with $p_\theta(\mathbf{x}_t | \mathbf{x}_{t+1})$, we are done.

Rewrite the Consistency Term. The nightmare of the above variational diffusion model is that we need to draw samples $(\mathbf{x}_{t-1}, \mathbf{x}_{t+1})$ from a joint distribution $q_\phi(\mathbf{x}_{t-1}, \mathbf{x}_{t+1} | \mathbf{x}_0)$. We don't know what $q_\phi(\mathbf{x}_{t-1}, \mathbf{x}_{t+1} | \mathbf{x}_0)$ is! It is a Gaussian by our choice, but we still need to use future samples \mathbf{x}_{t+1} to draw the current sample \mathbf{x}_t . This is odd.

Inspecting the consistency term, we notice that $q_\phi(\mathbf{x}_t | \mathbf{x}_{t-1})$ and $p_\theta(\mathbf{x}_t | \mathbf{x}_{t+1})$ are moving along two opposite directions. Thus, it is unavoidable that we need to use \mathbf{x}_{t-1} and \mathbf{x}_{t+1} . The question we need to ask is: Can we come up with something so that we do not need to handle two opposite directions while we are able to check consistency?

So, here is the simple trick called Bayes theorem which will give us

$$q(\mathbf{x}_t | \mathbf{x}_{t-1}) = \frac{q(\mathbf{x}_{t-1} | \mathbf{x}_t) q(\mathbf{x}_t)}{q(\mathbf{x}_{t-1})} \quad \text{condition on } \mathbf{x}_0 \implies q(\mathbf{x}_t | \mathbf{x}_{t-1}, \mathbf{x}_0) = \frac{q(\mathbf{x}_{t-1} | \mathbf{x}_t, \mathbf{x}_0) q(\mathbf{x}_t | \mathbf{x}_0)}{q(\mathbf{x}_{t-1} | \mathbf{x}_0)}. \quad (2.20)$$

With this change of the conditioning order, we can switch $q(\mathbf{x}_t | \mathbf{x}_{t-1}, \mathbf{x}_0)$ to $q(\mathbf{x}_{t-1} | \mathbf{x}_t, \mathbf{x}_0)$ by adding one more condition variable \mathbf{x}_0 . (If you do not condition on \mathbf{x}_0 , there is no way that we can draw samples from $q(\mathbf{x}_{t-1})$, for example, because the specific state of \mathbf{x}_{t-1} depends on the initial image \mathbf{x}_0 .) The direction $q(\mathbf{x}_{t-1} | \mathbf{x}_t, \mathbf{x}_0)$ is now parallel to $p_\theta(\mathbf{x}_{t-1} | \mathbf{x}_t)$ as shown in Figure 2.8. So, if we want to rewrite the consistency term, a natural option is to calculate the KL divergence between $q_\phi(\mathbf{x}_{t-1} | \mathbf{x}_t, \mathbf{x}_0)$ and $p_\theta(\mathbf{x}_{t-1} | \mathbf{x}_t)$.

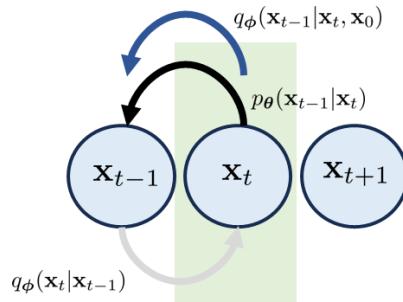


Figure 2.8: If we consider the Bayes theorem in Eqn (2.20), we can define a distribution $q_\phi(\mathbf{x}_{t-1} | \mathbf{x}_t, \mathbf{x}_0)$ that has a direction parallel to $p_\theta(\mathbf{x}_{t-1} | \mathbf{x}_t)$.

If we manage to go through a few (boring) algebraic derivations, we can show that the ELBO is now:

Theorem 2.4. (ELBO for Variational Diffusion Model). Let $\mathbf{x} = \mathbf{x}_0$, and $\mathbf{x}_T \sim \mathcal{N}(0, \mathbf{I})$. The ELBO for a variational diffusion model in Theorem 2.3 can be equivalently written as

$$\begin{aligned} \text{ELBO}_{\phi, \theta}(\mathbf{x}) &= \mathbb{E}_{q_\phi(\mathbf{x}_1 | \mathbf{x}_0)} [\underbrace{\log p_\theta(\mathbf{x}_0 | \mathbf{x}_1)}_{\text{same as before}}] - \underbrace{\mathbb{D}_{\text{KL}}(q_\phi(\mathbf{x}_T | \mathbf{x}_0) \| p(\mathbf{x}_T))}_{\text{new prior matching}} \\ &\quad - \sum_{t=2}^T \mathbb{E}_{q_\phi(\mathbf{x}_t | \mathbf{x}_0)} \left[\underbrace{\mathbb{D}_{\text{KL}}(q_\phi(\mathbf{x}_{t-1} | \mathbf{x}_t, \mathbf{x}_0) \| p_\theta(\mathbf{x}_{t-1} | \mathbf{x}_t))}_{\text{new consistency}} \right]. \quad (2.21) \end{aligned}$$

Let's quickly make three interpretations:

- **Reconstruction.** The new reconstruction term is the same as before. We are still maximizing the log-likelihood.
- **Prior Matching.** The new prior matching is simplified to the KL divergence between $q_\phi(\mathbf{x}_T|\mathbf{x}_0)$ and $p(\mathbf{x}_T)$. The change is due to the fact that we now condition upon \mathbf{x}_0 . Thus, there is no need to draw samples from $q_\phi(\mathbf{x}_{T-1}|\mathbf{x}_0)$ and take expectation.
- **Consistency.** The new consistency term is different from the previous one in two ways. Firstly, the running index t starts at $t = 2$ and ends at $t = T$. Previously it was from $t = 1$ to $t = T - 1$. Accompanied by this is the distribution matching, which is now between $q_\phi(\mathbf{x}_{t-1}|\mathbf{x}_t, \mathbf{x}_0)$ and $p_\theta(\mathbf{x}_{t-1}|\mathbf{x}_t)$. So, instead of asking a forward transition to match with a reverse transition, we use q_ϕ to construct a reverse transition and use it to match with p_θ .

Proof of Theorem 2.4. We begin with Eqn (2.17) by showing that

$$\begin{aligned}
\log p(\mathbf{x}) &\geq \mathbb{E}_{q_\phi(\mathbf{x}_{1:T}|\mathbf{x}_0)} \left[\log \frac{p(\mathbf{x}_{0:T})}{q_\phi(\mathbf{x}_{1:T}|\mathbf{x}_0)} \right] && \text{(By Eqn (2.17))} \\
&= \mathbb{E}_{q_\phi(\mathbf{x}_{1:T}|\mathbf{x}_0)} \left[\log \frac{p(\mathbf{x}_T)p(\mathbf{x}_0|\mathbf{x}_1)\prod_{t=2}^T p(\mathbf{x}_{t-1}|\mathbf{x}_t)}{q_\phi(\mathbf{x}_1|\mathbf{x}_0)\prod_{t=2}^T q_\phi(\mathbf{x}_t|\mathbf{x}_{t-1}, \mathbf{x}_0)} \right] && \text{(split the chain)} \\
&= \mathbb{E}_{q_\phi(\mathbf{x}_{1:T}|\mathbf{x}_0)} \left[\log \frac{p(\mathbf{x}_T)p(\mathbf{x}_0|\mathbf{x}_1)}{q_\phi(\mathbf{x}_1|\mathbf{x}_0)} \right] + \mathbb{E}_{q_\phi(\mathbf{x}_{1:T}|\mathbf{x}_0)} \left[\log \prod_{t=2}^T \frac{p(\mathbf{x}_{t-1}|\mathbf{x}_t)}{q_\phi(\mathbf{x}_t|\mathbf{x}_{t-1}, \mathbf{x}_0)} \right] && (2.22)
\end{aligned}$$

Let's consider the second term:

$$\begin{aligned}
\prod_{t=2}^T \frac{p(\mathbf{x}_{t-1}|\mathbf{x}_t)}{q_\phi(\mathbf{x}_t|\mathbf{x}_{t-1}, \mathbf{x}_0)} &= \prod_{t=2}^T \frac{p(\mathbf{x}_{t-1}|\mathbf{x}_t)}{\frac{q_\phi(\mathbf{x}_{t-1}|\mathbf{x}_t, \mathbf{x}_0)q_\phi(\mathbf{x}_t|\mathbf{x}_0)}{q_\phi(\mathbf{x}_{t-1}|\mathbf{x}_0)}} && \text{(Bayes rule, Eqn (2.20))} \\
&= \prod_{t=2}^T \frac{p(\mathbf{x}_{t-1}|\mathbf{x}_t)}{q_\phi(\mathbf{x}_{t-1}|\mathbf{x}_t, \mathbf{x}_0)} \times \prod_{t=2}^T \frac{q_\phi(\mathbf{x}_{t-1}|\mathbf{x}_0)}{q_\phi(\mathbf{x}_t|\mathbf{x}_0)} && \text{(Rearrange denominator)} \\
&= \prod_{t=2}^T \frac{p(\mathbf{x}_{t-1}|\mathbf{x}_t)}{q_\phi(\mathbf{x}_{t-1}|\mathbf{x}_t, \mathbf{x}_0)} \times \frac{q_\phi(\mathbf{x}_1|\mathbf{x}_0)}{q_\phi(\mathbf{x}_T|\mathbf{x}_0)}, && \text{(Recursion cancels terms)}
\end{aligned}$$

where the last equation uses the fact that for any sequence a_1, \dots, a_T , we have $\prod_{t=2}^T \frac{a_{t-1}}{a_t} = \frac{a_1}{a_2} \times \frac{a_2}{a_3} \times \dots \times \frac{a_{T-1}}{a_T} = \frac{a_1}{a_T}$. Going back to the Eqn (2.22), we can see that

$$\begin{aligned}
&\mathbb{E}_{q_\phi(\mathbf{x}_{1:T}|\mathbf{x}_0)} \left[\log \frac{p(\mathbf{x}_T)p(\mathbf{x}_0|\mathbf{x}_1)}{q_\phi(\mathbf{x}_1|\mathbf{x}_0)} \right] + \mathbb{E}_{q_\phi(\mathbf{x}_{1:T}|\mathbf{x}_0)} \left[\log \prod_{t=2}^T \frac{p(\mathbf{x}_{t-1}|\mathbf{x}_t)}{q_\phi(\mathbf{x}_t|\mathbf{x}_{t-1}, \mathbf{x}_0)} \right] \\
&= \mathbb{E}_{q_\phi(\mathbf{x}_{1:T}|\mathbf{x}_0)} \left[\log \frac{p(\mathbf{x}_T)p(\mathbf{x}_0|\mathbf{x}_1)}{q_\phi(\mathbf{x}_1|\mathbf{x}_0)} + \log \frac{q_\phi(\mathbf{x}_1|\mathbf{x}_0)}{q_\phi(\mathbf{x}_T|\mathbf{x}_0)} \right] + \mathbb{E}_{q_\phi(\mathbf{x}_{1:T}|\mathbf{x}_0)} \left[\log \prod_{t=2}^T \frac{p(\mathbf{x}_{t-1}|\mathbf{x}_t)}{q_\phi(\mathbf{x}_{t-1}|\mathbf{x}_t, \mathbf{x}_0)} \right] \\
&= \mathbb{E}_{q_\phi(\mathbf{x}_{1:T}|\mathbf{x}_0)} \left[\log \frac{p(\mathbf{x}_T)p(\mathbf{x}_0|\mathbf{x}_1)}{q_\phi(\mathbf{x}_T|\mathbf{x}_0)} \right] + \mathbb{E}_{q_\phi(\mathbf{x}_{1:T}|\mathbf{x}_0)} \left[\log \prod_{t=2}^T \frac{p(\mathbf{x}_{t-1}|\mathbf{x}_t)}{q_\phi(\mathbf{x}_{t-1}|\mathbf{x}_t, \mathbf{x}_0)} \right],
\end{aligned}$$

where we canceled $q_\phi(\mathbf{x}_1|\mathbf{x}_0)$ in the numerator and denominator since $\log \frac{a}{b} + \log \frac{b}{c} = \log \frac{a}{c}$ for any positive constants a , b , and c . This will give us

$$\begin{aligned}
\mathbb{E}_{q_\phi(\mathbf{x}_{1:T}|\mathbf{x}_0)} \left[\log \frac{p(\mathbf{x}_T)p(\mathbf{x}_0|\mathbf{x}_1)}{q_\phi(\mathbf{x}_T|\mathbf{x}_0)} \right] &= \mathbb{E}_{q_\phi(\mathbf{x}_{1:T}|\mathbf{x}_0)} [\log p(\mathbf{x}_0|\mathbf{x}_1)] + \mathbb{E}_{q_\phi(\mathbf{x}_{1:T}|\mathbf{x}_0)} \left[\log \frac{p(\mathbf{x}_T)}{q_\phi(\mathbf{x}_T|\mathbf{x}_0)} \right] \\
&= \underbrace{\mathbb{E}_{q_\phi(\mathbf{x}_1|\mathbf{x}_0)} [\log p(\mathbf{x}_0|\mathbf{x}_1)]}_{\text{reconstruction}} - \underbrace{\mathbb{D}_{\text{KL}}(q_\phi(\mathbf{x}_T|\mathbf{x}_0) \| p(\mathbf{x}_T))}_{\text{prior matching}}
\end{aligned}$$

The last term is

$$\begin{aligned}
\mathbb{E}_{q_\phi(\mathbf{x}_{1:T}|\mathbf{x}_0)} \left[\log \prod_{t=2}^T \frac{p(\mathbf{x}_{t-1}|\mathbf{x}_t)}{q_\phi(\mathbf{x}_{t-1}|\mathbf{x}_t, \mathbf{x}_0)} \right] &= \sum_{t=2}^T \mathbb{E}_{q_\phi(\mathbf{x}_t, \mathbf{x}_{t-1}|\mathbf{x}_0)} \log \frac{p(\mathbf{x}_{t-1}|\mathbf{x}_t)}{q_\phi(\mathbf{x}_{t-1}|\mathbf{x}_t, \mathbf{x}_0)} \\
&= \sum_{t=2}^T \iint \log \frac{p(\mathbf{x}_{t-1}|\mathbf{x}_t)}{q_\phi(\mathbf{x}_{t-1}|\mathbf{x}_t, \mathbf{x}_0)} \cdot q_\phi(\mathbf{x}_t, \mathbf{x}_{t-1}|\mathbf{x}_0) d\mathbf{x}_{t-1} d\mathbf{x}_t \\
&= \sum_{t=2}^T \iint \log \frac{p(\mathbf{x}_{t-1}|\mathbf{x}_t)}{q_\phi(\mathbf{x}_{t-1}|\mathbf{x}_t, \mathbf{x}_0)} \cdot q_\phi(\mathbf{x}_{t-1}|\mathbf{x}_t, \mathbf{x}_0) q_\phi(\mathbf{x}_t|\mathbf{x}_0) d\mathbf{x}_{t-1} d\mathbf{x}_t \\
&= \sum_{t=2}^T \int \left\{ \int \log \frac{p(\mathbf{x}_{t-1}|\mathbf{x}_t)}{q_\phi(\mathbf{x}_{t-1}|\mathbf{x}_t, \mathbf{x}_0)} \cdot q_\phi(\mathbf{x}_{t-1}, \mathbf{x}_t|\mathbf{x}_0) d\mathbf{x}_{t-1} \right\} q_\phi(\mathbf{x}_t|\mathbf{x}_0) d\mathbf{x}_t \\
&= - \underbrace{\sum_{t=2}^T \mathbb{E}_{q_\phi(\mathbf{x}_t|\mathbf{x}_0)} \mathbb{D}_{\text{KL}}(q_\phi(\mathbf{x}_{t-1}|\mathbf{x}_t, \mathbf{x}_0) \| p(\mathbf{x}_{t-1}|\mathbf{x}_t))}_{\text{consistency}}
\end{aligned}$$

Finally, replace $p(\mathbf{x}_{t-1}|\mathbf{x}_t)$ by $p_\theta(\mathbf{x}_{t-1}|\mathbf{x}_t)$, and $p(\mathbf{x}_0|\mathbf{x}_1)$ by $p_\theta(\mathbf{x}_0|\mathbf{x}_1)$. Done!

2.3 Distribution of the Reverse Process

Now that we know the new ELBO for the variational diffusion model, we should spend some time discussing its core component which is $q_\phi(\mathbf{x}_{t-1}|\mathbf{x}_t, \mathbf{x}_0)$. In a nutshell, what we want to show is that

- $q_\phi(\mathbf{x}_{t-1}|\mathbf{x}_t, \mathbf{x}_0)$ is still a Gaussian.
- Since it is a Gaussian, it is fully characterized by the mean and covariance. It turns out that

$$q_\phi(\mathbf{x}_{t-1}|\mathbf{x}_t, \mathbf{x}_0) = \mathcal{N}(\mathbf{x}_{t-1} | \heartsuit \mathbf{x}_t + \spadesuit \mathbf{x}_0, \clubsuit \mathbf{I}), \quad (2.23)$$

for some magical scalars \heartsuit , \spadesuit and \clubsuit defined below.

Theorem 2.5. The distribution $q_\phi(\mathbf{x}_{t-1}|\mathbf{x}_t, \mathbf{x}_0)$ takes the form of

$$q_\phi(\mathbf{x}_{t-1}|\mathbf{x}_t, \mathbf{x}_0) = \mathcal{N}(\mathbf{x}_{t-1} | \boldsymbol{\mu}_q(\mathbf{x}_t, \mathbf{x}_0), \boldsymbol{\Sigma}_q(t)), \quad (2.24)$$

where

$$\boldsymbol{\mu}_q(\mathbf{x}_t, \mathbf{x}_0) = \frac{(1 - \bar{\alpha}_{t-1})\sqrt{\alpha_t}}{1 - \bar{\alpha}_t} \mathbf{x}_t + \frac{(1 - \alpha_t)\sqrt{\alpha_{t-1}}}{1 - \bar{\alpha}_t} \mathbf{x}_0 \quad (2.25)$$

$$\boldsymbol{\Sigma}_q(t) = \frac{(1 - \alpha_t)(1 - \sqrt{\bar{\alpha}_{t-1}})}{1 - \bar{\alpha}_t} \mathbf{I} \stackrel{\text{def}}{=} \sigma_q^2(t) \mathbf{I}, \quad (2.26)$$

where $\bar{\alpha}_t = \prod_{i=1}^t \alpha_i$.

Eqn (2.25) reveals an interesting fact that the mean $\boldsymbol{\mu}_q(\mathbf{x}_t, \mathbf{x}_0)$ is a linear combination of \mathbf{x}_t and \mathbf{x}_0 . Geometrically, $\boldsymbol{\mu}_q(\mathbf{x}_t, \mathbf{x}_0)$ lives on the straight line connecting \mathbf{x}_t and \mathbf{x}_0 , as illustrated in Figure 2.9.

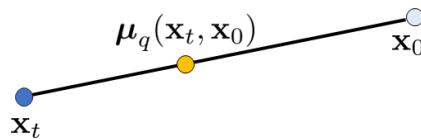


Figure 2.9: According to Eqn (2.25), the mean $\boldsymbol{\mu}_q(\mathbf{x}_t, \mathbf{x}_0)$ is a linear combination of \mathbf{x}_t and \mathbf{x}_0 .

Proof of Theorem 2.5. Using the Bayes theorem stated in Eqn (2.20), $q(\mathbf{x}_{t-1}|\mathbf{x}_t, \mathbf{x}_0)$ can be determined if we evaluate the following product of Gaussians

$$q(\mathbf{x}_{t-1}|\mathbf{x}_t, \mathbf{x}_0) = \frac{\mathcal{N}(\mathbf{x}_t|\sqrt{\alpha_t}\mathbf{x}_{t-1}, (1-\alpha_t)\mathbf{I})\mathcal{N}(\mathbf{x}_{t-1}|\sqrt{\bar{\alpha}_{t-1}}\mathbf{x}_0, (1-\bar{\alpha}_{t-1})\mathbf{I})}{\mathcal{N}(\mathbf{x}_t|\sqrt{\bar{\alpha}_t}\mathbf{x}_0, (1-\bar{\alpha}_t)\mathbf{I})}. \quad (2.27)$$

For simplicity we will treat the vectors are scalars. Then the above product of Gaussians will become

$$q(\mathbf{x}_{t-1}|\mathbf{x}_t, \mathbf{x}_0) \propto \exp \left\{ \frac{(\mathbf{x}_t - \sqrt{\alpha_t}\mathbf{x}_{t-1})^2}{2(1-\alpha_t)} + \frac{(\mathbf{x}_{t-1} - \sqrt{\bar{\alpha}_{t-1}}\mathbf{x}_0)^2}{2(1-\bar{\alpha}_{t-1})} - \frac{(\mathbf{x}_t - \sqrt{\bar{\alpha}_t}\mathbf{x}_0)^2}{2(1-\bar{\alpha}_t)} \right\}. \quad (2.28)$$

We consider the following mapping:

$$\begin{aligned} x &= \mathbf{x}_t, & a &= \alpha_t \\ y &= \mathbf{x}_{t-1}, & b &= \bar{\alpha}_{t-1} \\ z &= \mathbf{x}_0, & c &= \bar{\alpha}_t. \end{aligned}$$

Consider a quadratic function

$$f(y) = \frac{(x - \sqrt{ay})^2}{2(1-a)} + \frac{(y - \sqrt{bz})^2}{2(1-b)} - \frac{(x - \sqrt{cz})^2}{2(1-c)}. \quad (2.29)$$

We know that no matter how we rearrange the terms, the resulting function remains a quadratic equation. The minimizer of $f(y)$ is the mean of the resulting Gaussian. So, we can calculate the derivative of f and show that

$$f'(y) = \frac{1-ab}{(1-a)(1-b)}y - \left(\frac{\sqrt{a}}{1-a}x + \frac{\sqrt{b}}{1-b}z \right).$$

Setting $f'(y) = 0$ yields

$$y = \frac{(1-b)\sqrt{a}}{1-ab}x + \frac{(1-a)\sqrt{b}}{1-ab}z. \quad (2.30)$$

We note that $ab = \alpha_t\bar{\alpha}_{t-1} = \bar{\alpha}_t$. So,

$$\boldsymbol{\mu}_q(\mathbf{x}_t, \mathbf{x}_0) = \frac{(1-\bar{\alpha}_{t-1})\sqrt{\alpha_t}}{1-\bar{\alpha}_t}\mathbf{x}_t + \frac{(1-\alpha_t)\sqrt{\bar{\alpha}_{t-1}}}{1-\bar{\alpha}_t}\mathbf{x}_0. \quad (2.31)$$

Similarly, for the variance, we can check the curvature $f''(y)$. We can easily show that

$$f''(y) = \frac{1-ab}{(1-a)(1-b)} = \frac{1-\bar{\alpha}_t}{(1-\alpha_t)(1-\bar{\alpha}_{t-1})}.$$

Taking the reciprocal will give us

$$\boldsymbol{\Sigma}_q(t) = \frac{(1-\alpha_t)(1-\bar{\alpha}_{t-1})}{1-\bar{\alpha}_t}\mathbf{I}. \quad (2.32)$$

In combination weight in the above theorem deserves some study. Recall that

$$\boldsymbol{\mu}_q(\mathbf{x}_t, \mathbf{x}_0) = \frac{(1-\bar{\alpha}_{t-1})\sqrt{\alpha_t}}{1-\bar{\alpha}_t}\mathbf{x}_t + \frac{(1-\alpha_t)\sqrt{\bar{\alpha}_{t-1}}}{1-\bar{\alpha}_t}\mathbf{x}_0.$$

One question we can ask is how does the two coefficients behave as t goes from T to 1? We show an example in Figure 2.10. For this particular example, we use $\alpha_t = 0.9$ for all t . We plot the coefficients as a function of t . Figure 2.10 suggests that the coefficient for \mathbf{x}_t shrinks as t decreases from $t = T$ to $t = 1$, whereas the coefficient for \mathbf{x}_0 grows as t decreases.

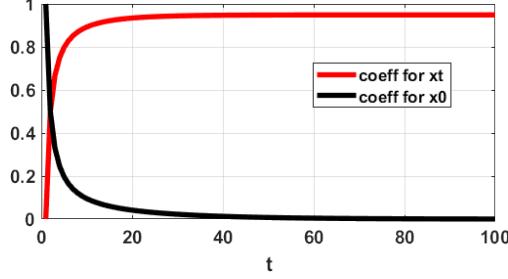


Figure 2.10: The trajectory of the coefficients for \mathbf{x}_t and for \mathbf{x}_0 , as t grows.

As t goes from T to 1, the variance $\sigma_q^2(t)$ will also change. Figure 2.11 shows the trajectory of \mathbf{x}_t as a function of t by sampling \mathbf{x}_t according to $q_\phi(\mathbf{x}_{t-1}|\mathbf{x}_t, \mathbf{x}_0)$. On the same plot, we show the radius of the Gaussian, defined by $\sigma_q^2(t)$. Our plot indicates that when $t = T$, the variance $\sigma_q^2(t)$ is fairly large so that \mathbf{x}_t is closer to white Gaussian noise. As t drops to $t = 1$, the variance $\sigma_q^2(t)$ also drops to zero. This makes sense because eventually we want \mathbf{x}_0 to be the clean image which is noise free.

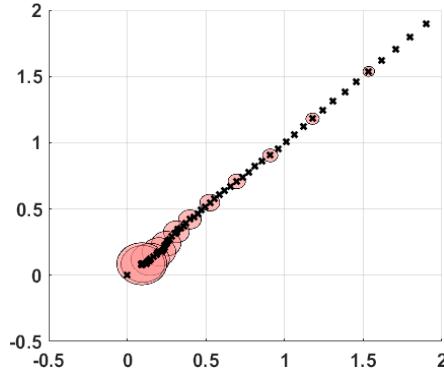


Figure 2.11: The trajectory of \mathbf{x}_t and the associated radius of the Gaussian $\sigma_q^2(t)$.

Constructing $p_\theta(\mathbf{x}_{t-1}|\mathbf{x}_t)$. The interesting part of Eqn (2.24) is that $q_\phi(\mathbf{x}_{t-1}|\mathbf{x}_t, \mathbf{x}_0)$ is *completely characterized* by \mathbf{x}_t and \mathbf{x}_0 . There is no neural network required to estimate the mean and variance! (You can compare this with VAE where a network is needed.) Since a network is not needed, there is really nothing to “learn”. The distribution $q_\phi(\mathbf{x}_{t-1}|\mathbf{x}_t, \mathbf{x}_0)$ is automatically determined if we know \mathbf{x}_t and \mathbf{x}_0 .

The realization here is important. Let’s look at the consistency term in Eqn (2.21):

$$\begin{aligned} \text{ELBO}_{\phi, \theta}(\mathbf{x}) &= \mathbb{E}_{q_\phi(\mathbf{x}_1|\mathbf{x}_0)}[\log \underbrace{p_\theta(\mathbf{x}_0|\mathbf{x}_1)}_{\text{same as before}}] - \underbrace{\mathbb{D}_{\text{KL}}(q_\phi(\mathbf{x}_T|\mathbf{x}_0) \| p(\mathbf{x}_T))}_{\text{new prior matching}} \\ &\quad - \sum_{t=2}^T \mathbb{E}_{q_\phi(\mathbf{x}_t|\mathbf{x}_0)} \left[\underbrace{\mathbb{D}_{\text{KL}}(q_\phi(\mathbf{x}_{t-1}|\mathbf{x}_t, \mathbf{x}_0) \| p_\theta(\mathbf{x}_{t-1}|\mathbf{x}_t))}_{\text{new consistency}} \right], \end{aligned} \quad (\text{from Eqn (2.21)})$$

where we just showed that

$$q_\phi(\mathbf{x}_{t-1}|\mathbf{x}_t, \mathbf{x}_0) = \mathcal{N}(\mathbf{x}_{t-1} | \boldsymbol{\mu}_q(\mathbf{x}_t, \mathbf{x}_0), \boldsymbol{\Sigma}_q(t)).$$

There is no “learning” for $q_\phi(\mathbf{x}_{t-1}|\mathbf{x}_t, \mathbf{x}_0)$ because it is defined once the hyperparameter α_t are defined. Therefore, the consistency term is a summation of many KL divergence terms where the t -th term is

$$\mathbb{D}_{\text{KL}}(\underbrace{q_\phi(\mathbf{x}_{t-1}|\mathbf{x}_t, \mathbf{x}_0)}_{\text{nothing to learn}} \| \underbrace{p_\theta(\mathbf{x}_{t-1}|\mathbf{x}_t)}_{\text{need to do something}}). \quad (2.33)$$

So, to compute the KL divergence, we need to do something about $p_{\theta}(\mathbf{x}_{t-1}|\mathbf{x}_t)$.

The big idea here is that $q_{\phi}(\mathbf{x}_{t-1}|\mathbf{x}_t, \mathbf{x}_0)$ is Gaussian. If we want to quickly calculate the KL divergence, then it would be good if $p_{\theta}(\mathbf{x}_{t-1}|\mathbf{x}_t)$ is also a Gaussian. So, we choose $p_{\theta}(\mathbf{x}_{t-1}|\mathbf{x}_t)$ to be a Gaussian. Moreover, we should match the form of the mean and variance! Therefore, we define

$$p_{\theta}(\mathbf{x}_{t-1}|\mathbf{x}_t) = \mathcal{N}\left(\mathbf{x}_{t-1} \mid \underbrace{\boldsymbol{\mu}_{\theta}(\mathbf{x}_t)}_{\text{neural network}}, \sigma_q^2(t)\mathbf{I}\right), \quad (2.34)$$

where we assume that the mean vector can be determined using a neural network. As for the variance, we choose the variance to be $\sigma_q^2(t)$. This is *identical* to Eqn (2.26)! Thus, if we put Eqn (2.24) side by side with $p_{\theta}(\mathbf{x}_{t-1}|\mathbf{x}_t)$, we notice a parallel relation between the two:

$$q_{\phi}(\mathbf{x}_{t-1}|\mathbf{x}_t, \mathbf{x}_0) = \mathcal{N}\left(\mathbf{x}_{t-1} \mid \underbrace{\boldsymbol{\mu}_q(\mathbf{x}_t, \mathbf{x}_0)}_{\text{known}}, \underbrace{\sigma_q^2(t)\mathbf{I}}_{\text{known}}\right), \quad (2.35)$$

$$p_{\theta}(\mathbf{x}_{t-1}|\mathbf{x}_t) = \mathcal{N}\left(\mathbf{x}_{t-1} \mid \underbrace{\boldsymbol{\mu}_{\theta}(\mathbf{x}_t)}_{\text{neural network}}, \underbrace{\sigma_q^2(t)\mathbf{I}}_{\text{known}}\right). \quad (2.36)$$

Therefore, the KL divergence is simplified to

$$\begin{aligned} & \mathbb{D}_{\text{KL}}\left(q_{\phi}(\mathbf{x}_{t-1}|\mathbf{x}_t, \mathbf{x}_0) \parallel p_{\theta}(\mathbf{x}_{t-1}|\mathbf{x}_t)\right) \\ &= \mathbb{D}_{\text{KL}}\left(\mathcal{N}(\mathbf{x}_{t-1} \mid \boldsymbol{\mu}_q(\mathbf{x}_t, \mathbf{x}_0), \sigma_q^2(t)\mathbf{I}) \parallel \mathcal{N}(\mathbf{x}_{t-1} \mid \boldsymbol{\mu}_{\theta}(\mathbf{x}_t), \sigma_q^2(t)\mathbf{I})\right) \\ &= \frac{1}{2\sigma_q^2(t)} \|\boldsymbol{\mu}_q(\mathbf{x}_t, \mathbf{x}_0) - \boldsymbol{\mu}_{\theta}(\mathbf{x}_t)\|^2, \end{aligned} \quad (2.37)$$

where we used the fact that the KL divergence between two identical-variance Gaussians is just the Euclidean distance square between the two mean vectors.

Substituting Eqn (2.37) to the definition of ELBO in Eqn (2.21), we can rewrite ELBO as follows.

Theorem 2.6. The **ELBO for a variational diffusion model** in Eqn (2.21) can be simplified to

$$\begin{aligned} \text{ELBO}_{\theta}(\mathbf{x}) &= \mathbb{E}_{q(\mathbf{x}_1|\mathbf{x}_0)}[\log p_{\theta}(\mathbf{x}_0|\mathbf{x}_1)] - \underbrace{\mathbb{D}_{\text{KL}}\left(q(\mathbf{x}_T|\mathbf{x}_0) \parallel p(\mathbf{x}_T)\right)}_{\text{nothing to train}} \\ &\quad - \sum_{t=2}^T \mathbb{E}_{q(\mathbf{x}_t|\mathbf{x}_0)}\left[\frac{1}{2\sigma_q^2(t)} \|\boldsymbol{\mu}_q(\mathbf{x}_t, \mathbf{x}_0) - \boldsymbol{\mu}_{\theta}(\mathbf{x}_t)\|^2\right], \end{aligned} \quad (2.38)$$

where $\mathbf{x} = \mathbf{x}_0$, and $\mathbf{x}_T \sim \mathcal{N}(0, \mathbf{I})$.

One remark for Theorem 2.6 is that the subscript ϕ is dropped because the distribution q_{ϕ} defined in Theorem 2.5 is fully characterized by \mathbf{x}_t and \mathbf{x}_0 . There is nothing to learn, and so the optimization does not need to include ϕ . Because of this, we can drop the KL-divergence term in Eqn (2.38). This leaves us the reconstruction term $\mathbb{E}_{q(\mathbf{x}_1|\mathbf{x}_0)}[\log p_{\theta}(\mathbf{x}_0|\mathbf{x}_1)]$ and transition term $\sum_{t=2}^T \mathbb{E}_{q(\mathbf{x}_t|\mathbf{x}_0)}\left[\frac{1}{2\sigma_q^2(t)} \|\boldsymbol{\mu}_q(\mathbf{x}_t, \mathbf{x}_0) - \boldsymbol{\mu}_{\theta}(\mathbf{x}_t)\|^2\right]$.

The reconstruction term can be simplified, since

$$\begin{aligned} \log p_{\theta}(\mathbf{x}_0|\mathbf{x}_1) &= \log \mathcal{N}(\mathbf{x}_0 \mid \boldsymbol{\mu}_{\theta}(\mathbf{x}_1), \sigma_q^2(1)\mathbf{I}) \\ &= \log \frac{1}{(\sqrt{2\pi\sigma_q^2(1)})^d} \exp\left\{-\frac{\|\mathbf{x}_0 - \boldsymbol{\mu}_{\theta}(\mathbf{x}_1)\|^2}{2\sigma_q^2(1)}\right\} \\ &= -\frac{\|\mathbf{x}_0 - \boldsymbol{\mu}_{\theta}(\mathbf{x}_1)\|^2}{2\sigma_q^2(1)} - \frac{d}{2} \log(2\pi\sigma_q^2(1)). \end{aligned}$$

So, as soon as we know \mathbf{x}_1 , we can send it to a network $\boldsymbol{\mu}_{\theta}(\mathbf{x}_1)$ to return us a mean estimate. The mean estimate will then be used to compute the likelihood.

2.4 Training and Inference

In this section we discuss how to train a variational diffusion model, and turn into a *denoising* diffusion probabilistic model.

We start by looking at the ELBO defined in Theorem 2.6. Eqn (2.38) suggests that we need to find a network μ_θ that can somehow minimize the loss:

$$\frac{1}{2\sigma_q^2(t)} \underbrace{\|\mu_q(\mathbf{x}_t, \mathbf{x}_0)\|_2^2}_{\text{known}} - \underbrace{\|\mu_\theta(\mathbf{x}_t)\|_2^2}_{\text{network}}. \quad (2.39)$$

Recall from Eqn (2.25) that

$$\mu_q(\mathbf{x}_t, \mathbf{x}_0) = \frac{(1 - \bar{\alpha}_{t-1})\sqrt{\alpha_t}}{1 - \bar{\alpha}_t} \mathbf{x}_t + \frac{(1 - \alpha_t)\sqrt{\bar{\alpha}_{t-1}}}{1 - \bar{\alpha}_t} \mathbf{x}_0. \quad (2.40)$$

We see that it is a function of \mathbf{x}_t and \mathbf{x}_0 . Therefore, $\mu_q(\mathbf{x}_t, \mathbf{x}_0)$ is known and determined once we know \mathbf{x}_t and \mathbf{x}_0 .

The subject of interest is μ_θ . Since μ_θ is our *design*, there is no reason why we cannot define it as something more convenient. So here is an option. We define

$$\underbrace{\mu_\theta(\mathbf{x}_t)}_{\text{a network}} \stackrel{\text{def}}{=} \frac{(1 - \bar{\alpha}_{t-1})\sqrt{\alpha_t}}{1 - \bar{\alpha}_t} \mathbf{x}_t + \frac{(1 - \alpha_t)\sqrt{\bar{\alpha}_{t-1}}}{1 - \bar{\alpha}_t} \underbrace{\hat{\mathbf{x}}_\theta(\mathbf{x}_t)}_{\text{another network}}. \quad (2.41)$$

Substituting Eqn (2.40) and Eqn (2.41) into Eqn (2.39) will give us

$$\begin{aligned} \frac{1}{2\sigma_q^2(t)} \|\mu_q(\mathbf{x}_t, \mathbf{x}_0) - \mu_\theta(\mathbf{x}_t)\|_2^2 &= \frac{1}{2\sigma_q^2(t)} \left\| \frac{(1 - \alpha_t)\sqrt{\bar{\alpha}_{t-1}}}{1 - \bar{\alpha}_t} (\hat{\mathbf{x}}_\theta(\mathbf{x}_t) - \mathbf{x}_0) \right\|_2^2 \\ &= \frac{1}{2\sigma_q^2(t)} \frac{(1 - \alpha_t)^2 \bar{\alpha}_{t-1}}{(1 - \bar{\alpha}_t)^2} \|\hat{\mathbf{x}}_\theta(\mathbf{x}_t) - \mathbf{x}_0\|_2^2. \end{aligned} \quad (2.42)$$

Therefore, ELBO can be written as

$$\text{ELBO}_\theta(\mathbf{x}) = \mathbb{E}_{q(\mathbf{x}_1|\mathbf{x}_0)}[\log p_\theta(\mathbf{x}_0|\mathbf{x}_1)] - \sum_{t=2}^T \mathbb{E}_{q(\mathbf{x}_t|\mathbf{x}_0)} \left[\frac{1}{2\sigma_q^2(t)} \frac{(1 - \alpha_t)^2 \bar{\alpha}_{t-1}}{(1 - \bar{\alpha}_t)^2} \|\hat{\mathbf{x}}_\theta(\mathbf{x}_t) - \mathbf{x}_0\|_2^2 \right], \quad (2.43)$$

where we dropped the term $\mathbb{D}_{\text{KL}}(q(\mathbf{x}_T|\mathbf{x}_0) \| p(\mathbf{x}_T))$.

Next, we want to simplify ELBO so that we can absorb $\mathbb{E}_{q(\mathbf{x}_1|\mathbf{x}_0)}[\log p_\theta(\mathbf{x}_0|\mathbf{x}_1)]$ into the summation. The following is the result.

Theorem 2.7. The ELBO for denoising diffusion probabilistic model is

$$\text{ELBO}_\theta(\mathbf{x}) = - \sum_{t=1}^T \frac{1}{2\sigma_q^2(t)} \frac{(1 - \alpha_t)^2 \bar{\alpha}_{t-1}}{(1 - \bar{\alpha}_t)^2} \mathbb{E}_{q(\mathbf{x}_t|\mathbf{x}_0)} [\|\hat{\mathbf{x}}_\theta(\mathbf{x}_t) - \mathbf{x}_0\|_2^2]. \quad (2.44)$$

Proof. Substituting Eqn (2.42) into Eqn (2.38), we can see that

$$\begin{aligned} \text{ELBO}_\theta(\mathbf{x}) &= \mathbb{E}_{q(\mathbf{x}_1|\mathbf{x}_0)}[\log p_\theta(\mathbf{x}_0|\mathbf{x}_1)] - \sum_{t=2}^T \mathbb{E}_{q(\mathbf{x}_t|\mathbf{x}_0)} \left[\frac{1}{2\sigma_q^2(t)} \|\mu_q(\mathbf{x}_t, \mathbf{x}_0) - \mu_\theta(\mathbf{x}_t)\|_2^2 \right] \\ &= \mathbb{E}_{q(\mathbf{x}_1|\mathbf{x}_0)}[\log p_\theta(\mathbf{x}_0|\mathbf{x}_1)] - \sum_{t=2}^T \mathbb{E}_{q(\mathbf{x}_t|\mathbf{x}_0)} \left[\frac{1}{2\sigma_q^2(t)} \frac{(1 - \alpha_t)^2 \bar{\alpha}_{t-1}}{(1 - \bar{\alpha}_t)^2} \|\hat{\mathbf{x}}_\theta(\mathbf{x}_t) - \mathbf{x}_0\|_2^2 \right]. \end{aligned} \quad (2.45)$$

The first term is

$$\begin{aligned}
\log p_{\theta}(\mathbf{x}_0 | \mathbf{x}_1) &= \log \mathcal{N}(\mathbf{x}_0 | \boldsymbol{\mu}_{\theta}(\mathbf{x}_1), \sigma_q^2(1)\mathbf{I}) \propto -\frac{1}{2\sigma_q^2(1)} \|\boldsymbol{\mu}_{\theta}(\mathbf{x}_1) - \mathbf{x}_0\|^2 && \text{(definition)} \\
&= -\frac{1}{2\sigma_q^2(1)} \left\| \frac{(1-\bar{\alpha}_0)\sqrt{\alpha_1}}{1-\bar{\alpha}_1} \mathbf{x}_1 + \frac{(1-\alpha_1)\sqrt{\alpha_0}}{1-\bar{\alpha}_1} \hat{\mathbf{x}}_{\theta}(\mathbf{x}_1) - \mathbf{x}_0 \right\|^2 && \text{(recall } \alpha_0 = 1) \\
&= -\frac{1}{2\sigma_q^2(1)} \left\| \frac{(1-\alpha_1)}{1-\bar{\alpha}_1} \hat{\mathbf{x}}_{\theta}(\mathbf{x}_1) - \mathbf{x}_0 \right\|^2 \\
&= -\frac{1}{2\sigma_q^2(1)} \|\hat{\mathbf{x}}_{\theta}(\mathbf{x}_1) - \mathbf{x}_0\|^2. && \text{(recall } \bar{\alpha}_1 = \alpha_1) \quad (2.46)
\end{aligned}$$

Substituting Eqn (2.46) into Eqn (2.45) will simplify ELBO as

$$\text{ELBO}_{\theta}(\mathbf{x}) = -\sum_{t=1}^T \mathbb{E}_{q(\mathbf{x}_t | \mathbf{x}_0)} \left[\frac{1}{2\sigma_q^2(t)} \frac{(1-\alpha_t)^2 \bar{\alpha}_{t-1}}{(1-\bar{\alpha}_t)^2} \|\hat{\mathbf{x}}_{\theta}(\mathbf{x}_t) - \mathbf{x}_0\|^2 \right].$$

The loss function defined in Eqn (2.44) is very intuitive. Ignoring the constants and expectations, the main subject of interest, for a particular \mathbf{x}_t , is

$$\operatorname{argmin}_{\theta} \|\hat{\mathbf{x}}_{\theta}(\mathbf{x}_t) - \mathbf{x}_0\|^2.$$

This is nothing but a denoising problem because we need to find a network $\hat{\mathbf{x}}_{\theta}$ such that the denoised image $\hat{\mathbf{x}}_{\theta}(\mathbf{x}_t)$ will be close to the ground truth \mathbf{x}_0 . What makes it not a typical denoiser is the following reasons:

- $\mathbb{E}_{q(\mathbf{x}_t | \mathbf{x}_0)}$: We are not trying to denoise any random noisy image. Instead, we are carefully choosing the noisy image to be

$$\begin{aligned}
\mathbf{x}_t &\sim q(\mathbf{x}_t | \mathbf{x}_0) = \mathcal{N}(\mathbf{x}_t | \sqrt{\bar{\alpha}_t} \mathbf{x}_0, (1-\bar{\alpha}_t)\mathbf{I}) \\
\Leftrightarrow \quad \mathbf{x}_t &= \sqrt{\bar{\alpha}_t} \mathbf{x}_0 + \sqrt{(1-\bar{\alpha}_t)} \boldsymbol{\epsilon}_t, \quad \text{where} \quad \boldsymbol{\epsilon}_t \sim \mathcal{N}(0, \mathbf{I}).
\end{aligned}$$

- $\frac{1}{2\sigma_q^2(t)} \frac{(1-\alpha_t)^2 \bar{\alpha}_{t-1}}{(1-\bar{\alpha}_t)^2}$: We do not weight the denoising loss equally for all steps. Instead, there is a scheduler to control the relative emphasis on each denoising loss. Considering this, and using Monte Carlo to approximate the expectation, we can write the optimization problem as

$$\begin{aligned}
&\operatorname{argmax}_{\theta} \sum_{\mathbf{x}_0 \in \mathcal{X}} \text{ELBO}(\mathbf{x}_0) \\
&= \operatorname{argmin}_{\theta} \sum_{\mathbf{x}_0 \in \mathcal{X}} \sum_{t=1}^T \mathbb{E}_{q(\mathbf{x}_t | \mathbf{x}_0)} \left[\frac{1}{2\sigma_q^2(t)} \frac{(1-\alpha_t)^2 \bar{\alpha}_{t-1}}{(1-\bar{\alpha}_t)^2} \left\| \hat{\mathbf{x}}_{\theta}\left(\sqrt{\bar{\alpha}_t} \mathbf{x}_0 + \sqrt{(1-\bar{\alpha}_t)} \boldsymbol{\epsilon}_t\right) - \mathbf{x}_0 \right\|^2 \right] \\
&= \operatorname{argmin}_{\theta} \sum_{\mathbf{x}_0 \in \mathcal{X}} \sum_{t=1}^T \frac{1}{M} \sum_{m=1}^M \frac{1}{2\sigma_q^2(t)} \frac{(1-\alpha_t)^2 \bar{\alpha}_{t-1}}{(1-\bar{\alpha}_t)^2} \left\| \hat{\mathbf{x}}_{\theta}\left(\sqrt{\bar{\alpha}_t} \mathbf{x}_0 + \sqrt{(1-\bar{\alpha}_t)} \boldsymbol{\epsilon}_t^{(m)}\right) - \mathbf{x}_0 \right\|^2, \quad (2.47)
\end{aligned}$$

where $\boldsymbol{\epsilon}_t^{(m)} \sim \mathcal{N}(0, \mathbf{I})$, and the summation over $\mathbf{x}_0 \in \mathcal{X}$ means we consider all samples in the training set \mathcal{X} . As you can see, the training of this model involves training a *denoiser* $\hat{\mathbf{x}}_{\theta}(\cdot)$. For this reason, the resulting model is known as the *denoising diffusion probabilistic model* (DDPM).

Forward Diffusion in DDPM. The training of a DDPM involves two parallel branches. The first branch is the forward diffusion. The goal of forward diffusion is to generate the intermediate variables $\mathbf{x}_1, \dots, \mathbf{x}_{T-1}$ by using

$$\mathbf{x}_t \sim q(\mathbf{x}_t | \mathbf{x}_0) = \mathcal{N}(\mathbf{x}_t | \sqrt{\bar{\alpha}_t} \mathbf{x}_0, (1-\bar{\alpha}_t)\mathbf{I}), \quad t = 1, \dots, T-1.$$

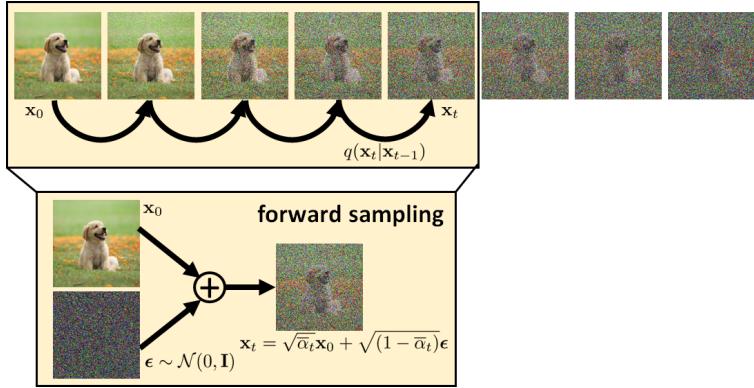


Figure 2.12: Forward diffusion process.

The forward diffusion does not require any training. If you give us the clean image \mathbf{x}_0 , we can run the forward diffusion and prepare the images $\mathbf{x}_1, \dots, \mathbf{x}_T$. A pictorial illustration is shown in Figure 2.12.

Training DDPM. Once the training samples $\mathbf{x}_0, \dots, \mathbf{x}_T$ are prepared, we can train the DDPM. The training of DDPM is summarized by the optimization in Eqn (2.47) and Figure 2.13. The goal is to train *one* denoiser for *all* noise levels. It is a one denoiser for all noise levels because each \mathbf{x}_t has a different variance $1 - \bar{\alpha}_t$. We are not interested in training many denoisers because it is computationally not feasible.

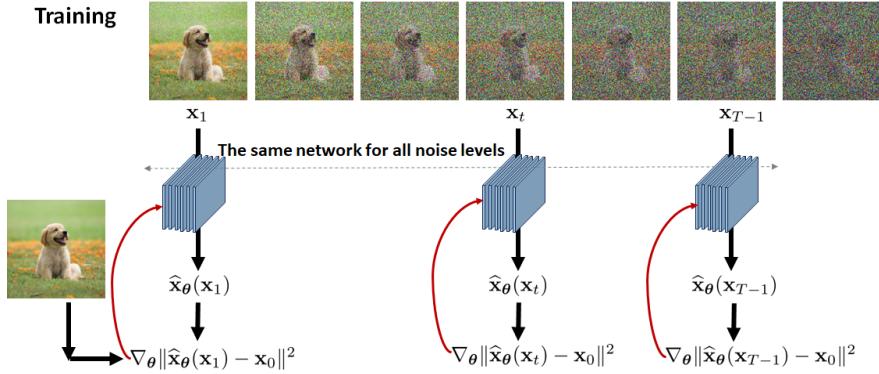


Figure 2.13: Training of a denoising diffusion probabilistic model. For the same neural network $\hat{\mathbf{x}}_\theta$, we send noisy inputs \mathbf{x}_t to the network. The gradient of the loss is back-propagated to update the network. Note that the noisy images are not arbitrary. They are generated according to the forward sampling process.

The training of a denoiser is no different than any conventional supervised learning. Given a pair of clean and noisy image, which in our case is \mathbf{x}_0 and \mathbf{x}_t , we train the denoiser $\hat{\mathbf{x}}_\theta(\cdot)$. The training loss in Eqn (2.47) has three summations. If we run stochastic gradient descent, we can simplify the above optimization into the following procedure.

Training Algorithm for DDPM. For every image \mathbf{x}_0 in your training dataset:

- Repeat the following steps until convergence.
- Pick a random time stamp $t \sim \text{Uniform}[1, T]$.
- Draw a sample $\mathbf{x}_t^{(m)} \sim \mathcal{N}(\mathbf{x}_t | \sqrt{\bar{\alpha}_t} \mathbf{x}_0, (1 - \bar{\alpha}_t) \mathbf{I})$, i.e.,

$$\mathbf{x}_t^{(m)} = \sqrt{\bar{\alpha}_t} \mathbf{x}_0 + \sqrt{(1 - \bar{\alpha}_t)} \boldsymbol{\epsilon}_t^{(m)}, \quad \boldsymbol{\epsilon}_t^{(m)} \sim \mathcal{N}(0, \mathbf{I}).$$

- Take gradient descent step on

$$\nabla_{\theta} \left\{ \frac{1}{M} \sum_{m=1}^M \left\| \hat{\mathbf{x}}_{\theta}(\mathbf{x}_t^{(m)}) - \mathbf{x}_0 \right\|^2 \right\}.$$

You can do this in batches, just like how you train any other neural networks. Note that, here, you are training **one** denoising network $\hat{\mathbf{x}}_{\theta}$ for **all** noisy conditions.

Inference of DDPM – the Reverse Diffusion. Once the denoiser $\hat{\mathbf{x}}_{\theta}$ is trained, we can apply it to do the inference. The inference is about sampling images from the distributions $p_{\theta}(\mathbf{x}_{t-1} | \mathbf{x}_t)$ over the sequence of states $\mathbf{x}_T, \mathbf{x}_{T-1}, \dots, \mathbf{x}_1$. Since it is the reverse diffusion process, we need to do it recursively via:

$$\mathbf{x}_{t-1} \sim p_{\theta}(\mathbf{x}_{t-1} | \mathbf{x}_t) = \mathcal{N}(\mathbf{x}_{t-1} | \mu_{\theta}(\mathbf{x}_t), \sigma_q^2(t)\mathbf{I}).$$

By reparameterization, we have

$$\begin{aligned} \mathbf{x}_{t-1} &= \mu_{\theta}(\mathbf{x}_t) + \sigma_q(t)\epsilon, & \text{where } \epsilon \sim \mathcal{N}(0, \mathbf{I}) \\ &= \frac{(1 - \bar{\alpha}_{t-1})\sqrt{\alpha_t}}{1 - \bar{\alpha}_t} \mathbf{x}_t + \frac{(1 - \alpha_t)\sqrt{\bar{\alpha}_{t-1}}}{1 - \bar{\alpha}_t} \hat{\mathbf{x}}_{\theta}(\mathbf{x}_t) + \sigma_q(t)\epsilon. \end{aligned}$$

This leads to the following inferencing algorithm. In plain words, we sequentially run the denoiser T times from the white noise vector \mathbf{x}_T back to the generated image $\hat{\mathbf{x}}_0$. A pictorial illustration is shown in Figure 2.14.

Inference of DDPM.

- You give us a white noise vector $\mathbf{x}_T \sim \mathcal{N}(0, \mathbf{I})$.
- Repeat the following for $t = T, T-1, \dots, 1$.
- We calculate $\hat{\mathbf{x}}_{\theta}(\mathbf{x}_t)$ using our trained denoiser.
- Update according to

$$\mathbf{x}_{t-1} = \frac{(1 - \bar{\alpha}_{t-1})\sqrt{\alpha_t}}{1 - \bar{\alpha}_t} \mathbf{x}_t + \frac{(1 - \alpha_t)\sqrt{\bar{\alpha}_{t-1}}}{1 - \bar{\alpha}_t} \hat{\mathbf{x}}_{\theta}(\mathbf{x}_t) + \sigma_q(t)\epsilon, \quad \epsilon \sim \mathcal{N}(0, \mathbf{I}). \quad (2.48)$$

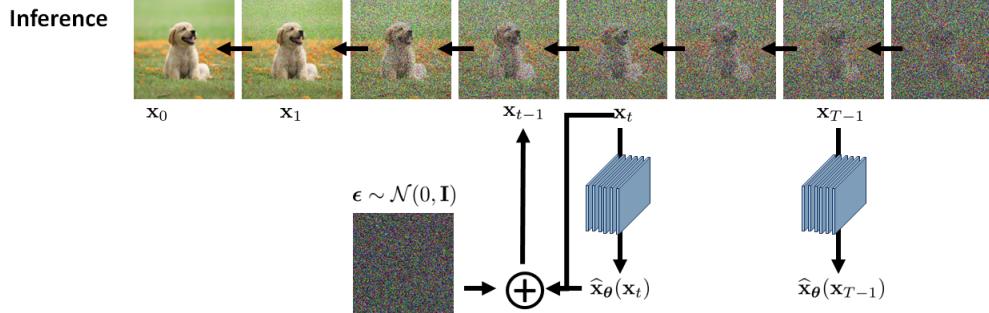


Figure 2.14: Inference of a denoising diffusion probabilistic model.

2.5 Predicting Noise

The final step of our derivation is to connect our results back to the original paper of Ho et al [16] so that the notations will be more consistent with the literature.

Training. If you are familiar with the denoising literature, you probably know the residue-type of algorithm that predicts the noise instead of the signal. The same spirit applies denoising diffusion, where

we can learn to predict the noise. To see why this is the case, we consider Eqn (2.11). If we re-arrange the terms we will obtain

$$\begin{aligned}\mathbf{x}_t &= \sqrt{\bar{\alpha}_t} \mathbf{x}_0 + \sqrt{1 - \bar{\alpha}_t} \boldsymbol{\epsilon}_0 \\ \Rightarrow \quad \mathbf{x}_0 &= \frac{\mathbf{x}_t - \sqrt{1 - \bar{\alpha}_t} \boldsymbol{\epsilon}_0}{\sqrt{\bar{\alpha}_t}}.\end{aligned}$$

Substituting this into $\boldsymbol{\mu}_q(\mathbf{x}_t, \mathbf{x}_0)$, we can show that

$$\begin{aligned}\boldsymbol{\mu}_q(\mathbf{x}_t, \mathbf{x}_0) &= \frac{\sqrt{\bar{\alpha}_t}(1 - \bar{\alpha}_{t-1})\mathbf{x}_t + \sqrt{\bar{\alpha}_{t-1}}(1 - \alpha_t)\mathbf{x}_0}{1 - \bar{\alpha}_t} \\ &= \frac{\sqrt{\bar{\alpha}_t}(1 - \bar{\alpha}_{t-1})\mathbf{x}_t + \sqrt{\bar{\alpha}_{t-1}}(1 - \alpha_t) \cdot \frac{\mathbf{x}_t - \sqrt{1 - \bar{\alpha}_t} \boldsymbol{\epsilon}_0}{\sqrt{\bar{\alpha}_t}}}{1 - \bar{\alpha}_t} \\ &= \text{a few more algebraic steps which we shall skip} \\ &= \frac{1}{\sqrt{\bar{\alpha}_t}} \mathbf{x}_t - \frac{1 - \alpha_t}{\sqrt{1 - \bar{\alpha}_t} \sqrt{\bar{\alpha}_t}} \boldsymbol{\epsilon}_0.\end{aligned}\tag{2.49}$$

In words, we have converted $\boldsymbol{\mu}_q(\mathbf{x}_t, \mathbf{x}_0)$ from a function of \mathbf{x}_0 to a function of $\boldsymbol{\epsilon}_0$.

Since we do this modification, naturally we should modify the mean estimator $\boldsymbol{\mu}_\theta$. In order to match the form of $\boldsymbol{\mu}_\theta$ with that of $\boldsymbol{\mu}_q(\mathbf{x}_t, \mathbf{x}_0)$, we choose

$$\boldsymbol{\mu}_\theta(\mathbf{x}_t) = \frac{1}{\sqrt{\bar{\alpha}_t}} \mathbf{x}_t - \frac{1 - \alpha_t}{\sqrt{1 - \bar{\alpha}_t} \sqrt{\bar{\alpha}_t}} \hat{\boldsymbol{\epsilon}}_\theta(\mathbf{x}_t).\tag{2.50}$$

Substituting Eqn (2.49) and Eqn (2.50) into Eqn (2.39) will give us a new ELBO

$$\begin{aligned}\text{ELBO}_\theta(\mathbf{x}_0, \boldsymbol{\epsilon}_0) &= - \sum_{t=1}^T \mathbb{E}_{q(\mathbf{x}_t | \mathbf{x}_0)} \left[\frac{1}{2\sigma_q^2(t)} \frac{(1 - \alpha_t)^2}{(1 - \bar{\alpha}_t)\alpha_t} \|\hat{\boldsymbol{\epsilon}}_\theta(\mathbf{x}_t) - \boldsymbol{\epsilon}_0\|^2 \right] \\ &= - \sum_{t=1}^T \mathbb{E}_{q(\mathbf{x}_t | \mathbf{x}_0)} \left[\frac{1}{2\sigma_q^2(t)} \frac{(1 - \alpha_t)^2}{(1 - \bar{\alpha}_t)\alpha_t} \left\| \hat{\boldsymbol{\epsilon}}_\theta \left(\sqrt{\bar{\alpha}_t} \mathbf{x}_0 + \sqrt{1 - \bar{\alpha}_t} \boldsymbol{\epsilon}_0 \right) - \boldsymbol{\epsilon}_0 \right\|^2 \right]\end{aligned}$$

We remark that this ELBO is a function of \mathbf{x}_0 and $\boldsymbol{\epsilon}_0$. Therefore, to train the denoiser, we need to solve the optimization

$$\begin{aligned}\operatorname{argmin}_\theta \mathbb{E}_{\mathbf{x}_0, \boldsymbol{\epsilon}_0} \text{ELBO}_\theta(\mathbf{x}_0, \boldsymbol{\epsilon}_0) \\ \approx \operatorname{argmin}_\theta \sum_{\mathbf{x}_0 \sim \mathcal{X}} \frac{1}{M} \sum_{m=1}^M \text{ELBO}_\theta(\mathbf{x}_0, \boldsymbol{\epsilon}_0^{(m)}),\end{aligned}$$

where the superscript denotes the m -th initial noise term. If we run stochastic gradient descent, we can simplify the above description to the following procedure.

Training DDPM using $\hat{\boldsymbol{\epsilon}}_\theta(\mathbf{x}_t)$. For every image \mathbf{x}_0 in your training dataset:

- Repeat the following steps until convergence.
- Pick a random time stamp $t \sim \text{Uniform}[1, T]$.
- Draw a sample $\boldsymbol{\epsilon}_0 \sim \mathcal{N}(0, \mathbf{I})$.
- Draw a sample $\mathbf{x}_t \sim \mathcal{N}(\mathbf{x}_t | \sqrt{\bar{\alpha}_t} \mathbf{x}_0, (1 - \bar{\alpha}_t) \mathbf{I})$, i.e.,

$$\mathbf{x}_t = \sqrt{\bar{\alpha}_t} \mathbf{x}_0 + \sqrt{1 - \bar{\alpha}_t} \boldsymbol{\epsilon}_0.$$

- Take gradient descent step on

$$\nabla_\theta \|\hat{\boldsymbol{\epsilon}}_\theta(\mathbf{x}_t) - \boldsymbol{\epsilon}_0\|^2.$$

Inference. For the inference, we know that

$$\mathbf{x}_{t-1} \sim p_{\theta}(\mathbf{x}_{t-1} | \mathbf{x}_t) = \mathcal{N}(\mathbf{x}_{t-1} | \boldsymbol{\mu}_{\theta}(\mathbf{x}_t), \sigma_q(t)^2 \mathbf{I}).$$

Using reparameterization, we can show that

$$\begin{aligned} \mathbf{x}_{t-1} &= \boldsymbol{\mu}_{\theta}(\mathbf{x}_t) + \sigma_q(t)\mathbf{z}, \quad \text{where } \mathbf{z} \sim \mathcal{N}(0, \mathbf{I}) \\ &= \left(\frac{1}{\sqrt{\alpha_t}} \mathbf{x}_t - \frac{1 - \alpha_t}{\sqrt{1 - \bar{\alpha}_t} \sqrt{\alpha_t}} \hat{\epsilon}_{\theta}(\mathbf{x}_t) \right) + \sigma_q(t)\mathbf{z} \\ &= \frac{1}{\sqrt{\alpha_t}} \left(\mathbf{x}_t - \frac{1 - \alpha_t}{\sqrt{1 - \bar{\alpha}_t}} \hat{\epsilon}_{\theta}(\mathbf{x}_t) \right) + \sigma_q(t)\mathbf{z}. \end{aligned} \quad (2.51)$$

Summarizing it here, we have

Inference of DDPM using $\hat{\epsilon}_{\theta}(\mathbf{x}_t)$.

- You give us a white noise vector $\mathbf{x}_T \sim \mathcal{N}(0, \mathbf{I})$.
- Repeat the following for $t = T, T-1, \dots, 1$.
- We calculate $\hat{\epsilon}_{\theta}(\mathbf{x}_t)$ using our trained denoiser.
- Update according to

$$\mathbf{x}_{t-1} = \frac{1}{\sqrt{\alpha_t}} \left(\mathbf{x}_t - \frac{1 - \alpha_t}{\sqrt{1 - \bar{\alpha}_t}} \hat{\epsilon}_{\theta}(\mathbf{x}_t) \right) + \sigma_q(t)\mathbf{z}, \quad \mathbf{z} \sim \mathcal{N}(0, \mathbf{I}). \quad (2.52)$$

2.6 Denoising Diffusion Implicit Model (DDIM)

From DDPM to DDIM. One of the most prevalent drawbacks of DDPM is that they need a large number of iterations to generate a reasonably good looking image. As mentioned by Song et al in [39], a DDPM would take more than 1000 hours to generate 50k images (of size 256×256) on a standard GPU. The reason is that when running the reverse diffusion steps, we need to perform denoising. If the reverse diffusion process intrinsically requires many steps to converge, then it will take us many denoising steps. Therefore, to speed up the computing, it is necessary to reduce the number of iterations. DDIM was an invention to overcome this difficulty.

Recall from Eqn (2.1) that the original DDPM transition probability takes the form

$$q(\mathbf{x}_t | \mathbf{x}_{t-1}) \stackrel{\text{def}}{=} \mathcal{N}\left(\mathbf{x}_t | \sqrt{\alpha_t} \mathbf{x}_{t-1}, (1 - \alpha_t) \mathbf{I}\right).$$

In addition, Eqn (2.8) shows that the probability of \mathbf{x}_t given \mathbf{x}_0 is

$$q(\mathbf{x}_t | \mathbf{x}_0) = \mathcal{N}\left(\mathbf{x}_t | \sqrt{\bar{\alpha}_t} \mathbf{x}_0, (1 - \bar{\alpha}_t) \mathbf{I}\right),$$

where $\bar{\alpha}_t = \prod_{i=0}^t \alpha_i$ for any decreasing sequence $0 < \alpha_t \leq 1$. One important observation here is that the transition probability $q(\mathbf{x}_t | \mathbf{x}_{t-1})$ follows a *Markov chain*, meaning that the probability of \mathbf{x}_t is purely dependent on \mathbf{x}_{t-1} but not the previous states \mathbf{x}_{t-2} and so on. The advantage of a Markovian structure is that the system is memoryless. Once we know \mathbf{x}_{t-1} , we will know \mathbf{x}_t . But the downside is that a Markov chain can take many steps to converge. DDIM overcomes this issue by departing from the Markovian structure to non-Markovian.

Probability Distributions in DDIM. To start our discussion, let's follow [39] by picking a special choice of parameters where we replace α_t by a ratio α_t/α_{t-1} . This means

$$q(\mathbf{x}_t | \mathbf{x}_{t-1}) \stackrel{\text{def}}{=} \mathcal{N}\left(\mathbf{x}_t | \sqrt{\frac{\alpha_t}{\alpha_{t-1}}} \mathbf{x}_{t-1}, (1 - \frac{\alpha_t}{\alpha_{t-1}}) \mathbf{I}\right).$$

There is no particularly strong physical meaning for this choice, except that it makes the notation simpler. With this choice, the product term is simplified to $\bar{\alpha}_t = \prod_{i=1}^t \frac{\alpha_i}{\alpha_{i-1}} = \alpha_t$ assuming that $\alpha_0 = 1$. Therefore,

$$q(\mathbf{x}_t | \mathbf{x}_0) = \mathcal{N}\left(\mathbf{x}_t \mid \sqrt{\alpha_t} \mathbf{x}_0, (1 - \alpha_t) \mathbf{I}\right). \quad (2.53)$$

Expressing Eqn (2.53) by means of reparametrization, we note that \mathbf{x}_t in Eqn (2.53) can be represented in terms of \mathbf{x}_0 as follows

$$\mathbf{x}_t = \sqrt{\alpha_t} \mathbf{x}_0 + \sqrt{1 - \alpha_t} \boldsymbol{\epsilon}, \quad \text{where } \boldsymbol{\epsilon} \sim \mathcal{N}(0, \mathbf{I}).$$

By the same argument, we can write

$$\mathbf{x}_{t-1} = \sqrt{\alpha_{t-1}} \mathbf{x}_0 + \sqrt{1 - \alpha_{t-1}} \boldsymbol{\epsilon}, \quad \text{where } \boldsymbol{\epsilon} \sim \mathcal{N}(0, \mathbf{I}). \quad (2.54)$$

So here comes an interesting trick. Let's replace $\boldsymbol{\epsilon}$ by something so that \mathbf{x}_{t-1} is no longer \mathbf{x}_0 perturbed by white noise. Perhaps we can consider the following derivation

$$\begin{aligned} \mathbf{x}_t &= \sqrt{\alpha_t} \mathbf{x}_0 + \sqrt{1 - \alpha_t} \boldsymbol{\epsilon} \\ \implies \sqrt{1 - \alpha_t} \boldsymbol{\epsilon} &= \mathbf{x}_t - \sqrt{\alpha_t} \mathbf{x}_0 \\ \implies \boldsymbol{\epsilon} &= \frac{\mathbf{x}_t - \sqrt{\alpha_t} \mathbf{x}_0}{\sqrt{1 - \alpha_t}}. \end{aligned}$$

So, substituting $\boldsymbol{\epsilon}$ into Eqn (2.54), we obtain

$$\begin{aligned} \mathbf{x}_{t-1} &= \sqrt{\alpha_{t-1}} \mathbf{x}_0 + \sqrt{1 - \alpha_{t-1}} \boldsymbol{\epsilon} \\ &= \sqrt{\alpha_{t-1}} \mathbf{x}_0 + \sqrt{1 - \alpha_{t-1}} \left(\frac{\mathbf{x}_t - \sqrt{\alpha_t} \mathbf{x}_0}{\sqrt{1 - \alpha_t}} \right). \end{aligned} \quad (2.55)$$

The difference between Eqn (2.55) and Eqn (2.54) is that in Eqn (2.54), the noise term is $\boldsymbol{\epsilon}$ which is $\mathcal{N}(0, \mathbf{I})$. It is this Gaussian that makes the derivations of DDPM easy, but it is also this Gaussian that makes the reverse diffusion slow. In contrast, Eqn (2.55) replaces the Gaussian by an estimate. This estimate uses the previous signal \mathbf{x}_t combined with the initial signal \mathbf{x}_0 . Of course, one can argue that in DDPM (e.g., Eqn (2.24)) also uses a combination of \mathbf{x}_t and \mathbf{x}_0 . The difference is that the combination in Eqn (2.55) allows us to do something that Eqn (2.24) does not, which is the derivation of the marginal distribution $q(\mathbf{x}_{t-1} | \mathbf{x}_0)$ and make it to a desired form.

Let's elaborate more on the marginal distribution. Referring to Eqn (2.55), we notice that we can *choose*

$$q(\mathbf{x}_{t-1} | \mathbf{x}_t, \mathbf{x}_0) = \mathcal{N}\left(\sqrt{\alpha_{t-1}} \mathbf{x}_0 + \sqrt{1 - \alpha_{t-1}} \left(\frac{\mathbf{x}_t - \sqrt{\alpha_t} \mathbf{x}_0}{\sqrt{1 - \alpha_t}} \right), \text{ something}\right).$$

where "something" stands for the variance of the Gaussian which can be made as $\sigma_t^2 \mathbf{I}$ for some hyperparameter σ_t . One important (likely the most important) argument in DDIM is that we want the marginal distribution $q(\mathbf{x}_{t-1} | \mathbf{x}_0)$ to have the same form as $q(\mathbf{x}_t | \mathbf{x}_0)$:

$$q(\mathbf{x}_{t-1} | \mathbf{x}_0) = \mathcal{N}(\sqrt{\alpha_{t-1}} \mathbf{x}_0, (1 - \alpha_{t-1}) \mathbf{I}).$$

The reason of aiming for this distribution is that ultimately we care about the marginal distribution $q(\mathbf{x}_t | \mathbf{x}_0)$ which we want it to become pure white noise when $t = T$ and it is the original image when $t = 0$. Therefore, while we can have millions of different choice of the transitional distribution $q(\mathbf{x}_{t-1} | \mathbf{x}_t, \mathbf{x}_0)$, only some very specialized transition probabilities can ensure that $q(\mathbf{x}_{t-1} | \mathbf{x}_0)$ takes a form we like.

Derivation of the Transition Distribution. With this goal in mind, we now state our mathematical problem. Suppose that

$$\begin{aligned} q(\mathbf{x}_t | \mathbf{x}_0) &= \mathcal{N}\left(\sqrt{\alpha_t} \mathbf{x}_0, (1 - \alpha_t) \mathbf{I}\right), \\ q(\mathbf{x}_{t-1} | \mathbf{x}_t, \mathbf{x}_0) &= \mathcal{N}\left(\sqrt{\alpha_{t-1}} \mathbf{x}_0 + \sqrt{1 - \alpha_{t-1}} \left(\frac{\mathbf{x}_t - \sqrt{\alpha_t} \mathbf{x}_0}{\sqrt{1 - \alpha_t}} \right), \sigma_t^2 \mathbf{I}\right), \end{aligned} \quad (2.56)$$

can we ensure that $q(\mathbf{x}_{t-1} | \mathbf{x}_0) = \mathcal{N}(\sqrt{\alpha_{t-1}} \mathbf{x}_0, (1 - \alpha_{t-1}) \mathbf{I})$? If not, what additional changes do we need?

The answer to this mathematical question requires some tools from textbooks. We recall the following result from Bishop's textbook [4].

Theorem 2.8. Bishop [4, Eqn 2.115] Suppose that we have two random variables \mathbf{x} and \mathbf{y} following the distributions

$$p(\mathbf{x}) = \mathcal{N}(\boldsymbol{\mu}, \boldsymbol{\Lambda}^{-1}),$$

$$p(\mathbf{y}|\mathbf{x}) = \mathcal{N}(\mathbf{Ax} + \mathbf{b}, \mathbf{L}^{-1}).$$

Then we can show that the marginal distribution is

$$p(\mathbf{y}) = \int p(\mathbf{y}|\mathbf{x})p(\mathbf{x})d\mathbf{x} = \mathcal{N}\left(\mathbf{A}\boldsymbol{\mu} + \mathbf{b}, \mathbf{L}^{-1} + \mathbf{A}\boldsymbol{\Lambda}^{-1}\mathbf{A}^T\right).$$

Let's see how we can apply this result to our problem. Looking at Eqn (2.56), we can identify the following qualities:

$$\mathbf{A} = \sqrt{\frac{1 - \alpha_{t-1}}{1 - \alpha_t}}, \quad \boldsymbol{\mu} = \sqrt{\alpha_t}\mathbf{x}_0, \quad \mathbf{b} = \sqrt{\alpha_{t-1}}\mathbf{x}_0 - \sqrt{\frac{1 - \alpha_{t-1}}{1 - \alpha_t}}\sqrt{\alpha_t}\mathbf{x}_0.$$

Suppose that $q(\mathbf{x}_{t-1}|\mathbf{x}_0) = \mathcal{N}(\boldsymbol{\mu}_{t-1}, \sigma_{t-1}^2 \mathbf{I})$ for some unknown choices of mean $\boldsymbol{\mu}_{t-1}$ and variance σ_{t-1}^2 . If we can show that $\boldsymbol{\mu}_{t-1} = \sqrt{\alpha_{t-1}}\mathbf{x}_0$ and $\sigma_{t-1}^2 = (1 - \alpha_{t-1})$, then we are done. To this end, we show that

$$\begin{aligned} \boldsymbol{\mu}_{t-1} &= \mathbf{A}\boldsymbol{\mu} + \mathbf{b} \\ &= \sqrt{\frac{1 - \alpha_{t-1}}{1 - \alpha_t}} \cdot \sqrt{\alpha_t}\mathbf{x}_0 + \sqrt{\alpha_{t-1}}\mathbf{x}_0 - \sqrt{\frac{1 - \alpha_{t-1}}{1 - \alpha_t}}\sqrt{\alpha_t}\mathbf{x}_0 \\ &= \sqrt{\alpha_{t-1}}\mathbf{x}_0. \end{aligned}$$

Oh, great news! We have shown that $\boldsymbol{\mu}_{t-1} = \sqrt{\alpha_{t-1}}\mathbf{x}_0$. That means for the transitional distribution $q(\mathbf{x}_{t-1}|\mathbf{x}_t, \mathbf{x}_0)$ we have chosen, the marginal distribution has the desired mean.

So it remains to check the variance. We show that

$$\begin{aligned} \sigma_{t-1}^2 &= \mathbf{L}^{-1} + \mathbf{A}\boldsymbol{\Lambda}^{-1}\mathbf{A}^T \\ &= \sigma_t^2 + \sqrt{\frac{1 - \alpha_{t-1}}{1 - \alpha_t}} \cdot (1 - \alpha_t) \cdot \sqrt{\frac{1 - \alpha_{t-1}}{1 - \alpha_t}} \\ &= \sigma_t^2 + (1 - \alpha_{t-1}). \end{aligned}$$

Oh, no! We cannot show that $\sigma_{t-1}^2 = 1 - \alpha_{t-1}$. There is an additional term σ_t^2 here. But this is not such a big deal. How about we do a quick fix by adding σ_t^2 into \mathbf{A} :

$$\begin{aligned} \sigma_{t-1}^2 &= \sigma_t^2 + \sqrt{\frac{1 - \alpha_{t-1} - \sigma_t^2}{1 - \alpha_t}} \cdot (1 - \alpha_t) \cdot \sqrt{\frac{1 - \alpha_{t-1} - \sigma_t^2}{1 - \alpha_t}} \\ &= \sigma_t^2 + 1 - \alpha_{t-1} - \sigma_t^2 \\ &= 1 - \alpha_{t-1}. \end{aligned}$$

Aha! σ_{t-1}^2 now takes the desired form such that $\sigma_{t-1}^2 = 1 - \alpha_{t-1}$. Let's do a quick check to make sure this additional σ_t^2 does not affect the mean:

$$\begin{aligned} \boldsymbol{\mu}_{t-1} &= \mathbf{A}\boldsymbol{\mu} + \mathbf{b} \\ &= \sqrt{\frac{1 - \alpha_{t-1} - \sigma_t^2}{1 - \alpha_t}} \cdot \sqrt{\alpha_t}\mathbf{x}_0 + \sqrt{\alpha_{t-1}}\mathbf{x}_0 - \sqrt{\frac{1 - \alpha_{t-1} - \sigma_t^2}{1 - \alpha_t}}\sqrt{\alpha_t}\mathbf{x}_0 \\ &= \sqrt{\alpha_{t-1}}\mathbf{x}_0. \end{aligned}$$

So, the mean remains the desired form despite we changed the variance term.

To summarize, we can choose $q(\mathbf{x}_{t-1}|\mathbf{x}_t, \mathbf{x}_0)$ to be the following.

Theorem 2.9. DDIM Transition Distribution. In DDIM, the transition distribution is defined as

$$q(\mathbf{x}_{t-1}|\mathbf{x}_t, \mathbf{x}_0) = \mathcal{N}\left(\sqrt{\alpha_{t-1}}\mathbf{x}_0 + \sqrt{1 - \alpha_{t-1} - \sigma_t^2} \frac{(\mathbf{x}_t - \sqrt{\alpha_t}\mathbf{x}_0)}{\sqrt{1 - \alpha_t}}, \sigma_t^2 \mathbf{I}\right). \quad (2.57)$$

If $q(\mathbf{x}_t|\mathbf{x}_0) = \mathcal{N}(\sqrt{\alpha_t}\mathbf{x}_0, (1-\alpha_t)\mathbf{I})$, then it follows from our derivation that $q(\mathbf{x}_{t-1}|\mathbf{x}_0) = \mathcal{N}(\sqrt{\alpha_{t-1}}\mathbf{x}_0, (1-\alpha_{t-1})\mathbf{I})$.

Inference for DDIM. The inference for DDIM is derived based on the transition distribution. Starting with the forward process, if we want to perform the reverse, we will need to find out \mathbf{x}_0 from the following equation:

$$\underbrace{\mathbf{x}_t}_{\text{given}} = \underbrace{\sqrt{\alpha_t}\mathbf{x}_0}_{\text{want to find}} + \underbrace{\sqrt{1 - \alpha_t}\boldsymbol{\epsilon}}_{\text{estimated by network}}.$$

By rearranging the terms, we see that

$$\begin{aligned} \mathbf{x}_0 &= \frac{1}{\sqrt{\alpha_t}} (\mathbf{x}_t - \sqrt{1 - \alpha_t}\boldsymbol{\epsilon}) \\ \Rightarrow f_{\boldsymbol{\theta}}^{(t)}(\mathbf{x}_t) &\stackrel{\text{def}}{=} \frac{1}{\sqrt{\alpha_t}} (\mathbf{x}_t - \sqrt{1 - \alpha_t}\boldsymbol{\epsilon}_{\boldsymbol{\theta}}^{(t)}(\mathbf{x}_t)). \end{aligned}$$

There are two new terms in this equation. The first one is $\boldsymbol{\epsilon}_{\boldsymbol{\theta}}^{(t)}(\mathbf{x}_t)$ which replaces $\boldsymbol{\epsilon}$. It is the estimate of the noise based on the current input \mathbf{x}_t . The second term is $f_{\boldsymbol{\theta}}^{(t)}(\mathbf{x}_t)$ which is defined as the equation $\frac{1}{\sqrt{\alpha_t}} (\mathbf{x}_t - \sqrt{1 - \alpha_t}\boldsymbol{\epsilon}_{\boldsymbol{\theta}}^{(t)}(\mathbf{x}_t))$. We can think of it as the estimate of the true signal \mathbf{x}_0 .

Going back to the transition distribution, we recall that it is denoted by $q(\mathbf{x}_{t-1}|\mathbf{x}_t, \mathbf{x}_0)$. If we do not have access to \mathbf{x}_0 , we can replace it $f_{\boldsymbol{\theta}}^{(t)}(\mathbf{x}_t)$. This means that

$$\begin{aligned} p_{\boldsymbol{\theta}}(\mathbf{x}_{t-1}|\mathbf{x}_t) &= q(\mathbf{x}_{t-1}|\mathbf{x}_t, \mathbf{x}_0) \\ \Rightarrow p_{\boldsymbol{\theta}}(\mathbf{x}_{t-1}|\mathbf{x}_t) &\stackrel{\text{def}}{=} q(\mathbf{x}_{t-1}|\mathbf{x}_t, f_{\boldsymbol{\theta}}^{(t)}(\mathbf{x}_t)) \\ &= \mathcal{N}\left(\sqrt{\alpha_{t-1}}f_{\boldsymbol{\theta}}^{(t)}(\mathbf{x}_t) + \sqrt{1 - \alpha_{t-1} - \sigma_t^2} \cdot \frac{\mathbf{x}_t - \sqrt{\alpha_t}f_{\boldsymbol{\theta}}^{(t)}(\mathbf{x}_t)}{\sqrt{1 - \alpha_t}}, \sigma_t^2 \mathbf{I}\right) \\ &= \mathcal{N}\left(\sqrt{\alpha_{t-1}} \cdot \frac{1}{\sqrt{\alpha_t}} (\mathbf{x}_t - \sqrt{1 - \alpha_t}\boldsymbol{\epsilon}_{\boldsymbol{\theta}}^{(t)}(\mathbf{x}_t)) + \right. \\ &\quad \left. + \sqrt{1 - \alpha_{t-1} - \sigma_t^2} \cdot \frac{\mathbf{x}_t - \sqrt{\alpha_t} \left(\frac{1}{\sqrt{\alpha_t}} (\mathbf{x}_t - \sqrt{1 - \alpha_t}\boldsymbol{\epsilon}_{\boldsymbol{\theta}}^{(t)}(\mathbf{x}_t)) \right)}{\sqrt{1 - \alpha_t}}, \sigma_t^2 \mathbf{I}\right) \\ &= \mathcal{N}\left(\sqrt{\alpha_{t-1}} \left(\frac{\mathbf{x}_t - \sqrt{1 - \alpha_t}\boldsymbol{\epsilon}_{\boldsymbol{\theta}}^{(t)}(\mathbf{x}_t)}{\sqrt{\alpha_t}} \right) + \sqrt{1 - \alpha_{t-1} - \sigma_t^2} \cdot \boldsymbol{\epsilon}_{\boldsymbol{\theta}}^{(t)}(\mathbf{x}_t), \sigma_t^2 \mathbf{I}\right). \quad (2.58) \end{aligned}$$

For the special case where $t = 1$, we define $p_{\boldsymbol{\theta}}(\mathbf{x}_{t-1}|\mathbf{x}_t) = \mathcal{N}(f_{\boldsymbol{\theta}}^{(1)}(\mathbf{x}_1), \sigma_1^2 \mathbf{I})$ so that the reverse process is supported everywhere. Looking at Eqn (2.58), we use reparametrization to write it as follows and interpret the equation according to [39].

$$(DDIM) \quad \mathbf{x}_{t-1} = \underbrace{\sqrt{\alpha_{t-1}} \left(\frac{\mathbf{x}_t - \sqrt{1 - \alpha_t}\boldsymbol{\epsilon}_{\boldsymbol{\theta}}^{(t)}(\mathbf{x}_t)}{\sqrt{\alpha_t}} \right)}_{\text{predicted } \mathbf{x}_0} + \underbrace{\sqrt{1 - \alpha_{t-1} - \sigma_t^2} \cdot \boldsymbol{\epsilon}_{\boldsymbol{\theta}}^{(t)}(\mathbf{x}_t)}_{\text{direction pointing to } \mathbf{x}_t} + \sigma_t \underbrace{\boldsymbol{\epsilon}_t}_{\sim \mathcal{N}(0, \mathbf{I})}. \quad (2.59)$$

It would be helpful to compare this equation with the DDPM equation in Eqn (2.52):

$$(DDPM) \quad \mathbf{x}_{t-1} = \frac{1}{\sqrt{\alpha_t}} \left(\mathbf{x}_t - \frac{1 - \alpha_t}{\sqrt{1 - \alpha_t}} \boldsymbol{\epsilon}_{\boldsymbol{\theta}}^{(t)}(\mathbf{x}_t) \right) + \sigma_t \boldsymbol{\epsilon}_t, \quad \boldsymbol{\epsilon}_t \sim \mathcal{N}(0, \mathbf{I}). \quad (2.60)$$

The main difference between DDPM and DDIM is subtle. While they both use \mathbf{x}_t and $\epsilon^{(t)}(\mathbf{x}_t)$ in their updates, the specific update formula makes a different convergence speed. In fact, later in the differential equation literature where people connect DDIM and DDPM with stochastic differential equations, it was observed that DDIM employed some special accelerated first-order numerical schemes when solving the differential equation.

2.7 Concluding Remark

The literature of DDPM is quickly exploding. The original paper by Sohl-Dickstein et al. [38] and Ho et al. [16] are the must-reads to understand the topic. For a more “user-friendly” version, we found that the tutorial by Luo very useful [27]. Some follow up works are highly cited, including the denoising diffusion implicit models by Song et al. [39]. In terms of application, people have been using DDPM for various image synthesis applications, e.g., [34, 35].

3 Score-Matching Langevin Dynamics (SMLD)

Score-based generative models [42] are alternative approaches to generate data from a desired distribution. There are several core ingredients: the Langevin equation, the (Stein) score function, and the score-matching loss. The focus of this section is more on the latter two. We will make a few hand-waving arguments about the Langevin equation and explain how to make it computationally feasible for our generative task. More in-depth discussions about the Langevin equation will be postponed to the next section.

3.1 Sampling from a Distribution

Imagine that we are given a distribution $p(\mathbf{x})$ and we want to draw samples from $p(\mathbf{x})$. If \mathbf{x} is a one-dimensional variable, we can achieve the goal easily by inverting the cumulative distribution function (CDF) [7, Chapter 4] and sending a uniform $[0, 1]$ random variable through this inverted CDF. For high dimensional distributions, the same CDF technique does not apply. However, the intuition of giving a higher weight to samples with a higher probability remains valid. For example, if we want to draw a sample from the orange dataset, it is almost certain that we want a spherical orange than a cubical orange.

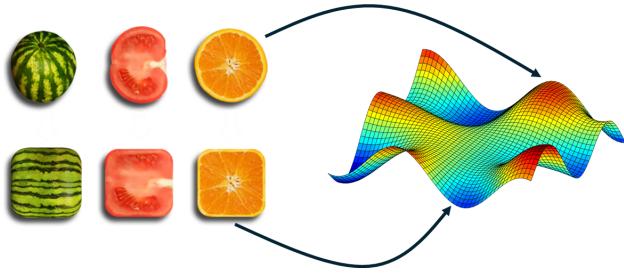


Figure 3.1: If our goal is to generate a fruit image, then we should expect the underlying distribution $p(\mathbf{x})$ will have a higher value for those “normal-looking” fruit images than those “weird-looking” images. Therefore, to sample from a distribution, it is more natural to pick a sample from a higher position of the distribution. The fruit image is taken from <https://cognitiveseo.com/blog/4224/unnatural-links-definition-examples/>

Therefore, in a non-rigorous way, we can argue that if we are given $p(\mathbf{x})$, we should aim to draw samples from a location where $p(\mathbf{x})$ has a high value. This idea of searching for a higher probability can be translated into an optimization

$$\mathbf{x}^* = \underset{\mathbf{x}}{\operatorname{argmax}} \log p(\mathbf{x}),$$

where the goal is to maximize the log-likelihood of the distribution $p(\mathbf{x})$. Certainly, this maximization does not provide any clue on how to draw a low probability sample which we will explain through the lens of Langevin equation. For now, we want to make a remark about the difference between the maximization here and maximum likelihood estimation. In maximum likelihood, the data point \mathbf{x} is fixed but the model parameters are changing. Here, the model parameters are fixed but the data point is changing. We are given a *fixed* model. Our goal is to draw the most likely sample from *this* model. The table below shows the difference between our sampling problem and maximum likelihood estimation.

Problem	Sampling	Maximum Likelihood
Optimization target	A sample \mathbf{x}	Model parameter θ
Formulation	$\mathbf{x}^* = \underset{\mathbf{x}}{\operatorname{argmax}} \log p(\mathbf{x}; \theta)$	$\theta^* = \underset{\theta}{\operatorname{argmax}} \log p(\mathbf{x}; \theta)$

Let’s continue our argument about the maximization. If $p(\mathbf{x})$ is a simple parametric model, the maximization will have analytic solutions. However, in general, optimizations in a high-dimensional space are ill-posed with many local minima. Therefore, there is no single algorithm that is globally converging in all situations. A reasonable trade-off between computational complexity, memory requirement, difficulty of

implementation, and solution quality is the gradient descent algorithm which is a first-order method. For our optimization where the objective function is $\log p(\mathbf{x})$, the gradient descent algorithm defines a sequence of iterates via the update rule

$$\mathbf{x}_{t+1} = \mathbf{x}_t + \tau \nabla_{\mathbf{x}} \log p(\mathbf{x}_t),$$

where $\nabla_{\mathbf{x}} \log p(\mathbf{x}_t)$ denotes the gradient of $\log p(\mathbf{x})$ evaluated at \mathbf{x}_t , and τ is the step size. Here we use “+” instead of the typical “−” because we are solving a maximization problem.

If you agree with the above approach, we can now provide an information introduction about the Langevin equation. Without worrying too much about its roots in physics, we can treat Langevin equation as an iterative procedure that allows us to draw samples.

Definition 3.1. The (discrete-time) **Langevin equation** for sampling from a known distribution $p(\mathbf{x})$ is an iterative procedure for $t = 1, \dots, T$:

$$\mathbf{x}_{t+1} = \mathbf{x}_t + \tau \nabla_{\mathbf{x}} \log p(\mathbf{x}_t) + \sqrt{2\tau} \mathbf{z}, \quad \mathbf{z} \sim \mathcal{N}(0, \mathbf{I}), \quad (3.1)$$

where τ is the step size which users can control, and \mathbf{x}_0 is white noise.

Example 3.1. Consider a Gaussian distribution $p(x) = \mathcal{N}(x | \mu, \sigma^2)$, we can show that the Langevin equation is

$$\begin{aligned} x_{t+1} &= x_t + \tau \cdot \nabla_x \log \left\{ \frac{1}{\sqrt{2\pi\sigma^2}} e^{-\frac{(x_t - \mu)^2}{2\sigma^2}} \right\} + \sqrt{2\tau} z \\ &= x_t - \tau \cdot \frac{x_t - \mu}{\sigma^2} + \sqrt{2\tau} z, \end{aligned} \quad z \sim \mathcal{N}(0, 1),$$

where the initial state can be set as $x_0 \sim \mathcal{N}(0, 1)$.

If we ignore the noise term $\sqrt{2\tau} \mathbf{z}$, the Langevin equation in Eqn (3.1) is exactly **gradient descent**, but for a particular function — the log likelihood of the random variable \mathbf{x} . Therefore, while gradient descent is a generic first-order optimization algorithm for any objective function, the Langevin equation focuses on the distribution if we use it in the context of generative models. The gradient descent algorithm plus a small noise perturbation gives us a simple summary.

In generative models,

Langevin equation = **gradient descent + noise**

applied to the log-likelihood function.

But why do we want gradient descent + noise instead of gradient descent? One interpretation is that we are not interested in solving the optimization problem. Instead, we are more interested in *sampling* from a distribution. By introducing the random noise to the gradient descent step, we randomly pick a sample that is following the objective function’s trajectory while not staying at where it is. If we are closer to the peak, we will move left and right slightly. If we are far from the peak, the gradient direction will pull us towards the peak. If the curvature around the peak is sharp, we will concentrate most of the steady state points \mathbf{x}_T there. If the curvature around the peak is flat, we will spread around. Therefore, by repeatedly initializing the gradient descent (plus noise) algorithm at a uniformly distributed location, we will eventually collect samples that will follow the distribution we designate.

A slightly more formal way to justify the Langevin equation is the Fokker-Planck equation. The Fokker-Planck equation is a fundamental result of stochastic processes. For any Markovian processes (e.g., Wiener process and Brownian motion), the dynamics of the solution \mathbf{x}_t is described by a stochastic differential equation (i.e., Langevin equation). However, since \mathbf{x}_t is a random variable at any time t , there is an underlying probability distribution $p(\mathbf{x}, t)$ associated with each \mathbf{x}_t . Fokker-Planck equation provides a mathematical statement about the distribution. The distribution must satisfy a partial differential equation. Roughly speaking, in the context of our problem, the Fokker-Planck equation can be described below.

Theorem 3.1. In the context of our problem, the solution \mathbf{x}_t of the Langevin equation will have a probability distribution $p(\mathbf{x}, t)$ at time t satisfying the **Fokker-Planck Equation**

$$\partial_t p(\mathbf{x}, t) = -\partial_{\mathbf{x}} \left\{ [\partial_{\mathbf{x}}(\log p(\mathbf{x}))] p(\mathbf{x}, t) \right\} + \partial_{\mathbf{x}}^2 p(\mathbf{x}, t), \quad (3.2)$$

where $\log p(\mathbf{x})$ is the log-likelihood of the ground truth distribution.

Deriving the Fokker-Planck equation will take a tremendous amount of effort. However, if we are given a candidate solution, verifying whether it satisfies the Fokker-Planck equation is not hard.

Verification of Theorem. Suppose that we have run Langevin equation for long enough that we have reached a converging solution \mathbf{x}_t as $t \rightarrow \infty$. We argue that this limiting distribution is $p(\mathbf{x})$. Indeed, we can show that

$$\partial_{\mathbf{x}} \left\{ \log p(\mathbf{x}) \right\} = \frac{\partial_{\mathbf{x}} p(\mathbf{x})}{p(\mathbf{x})}.$$

Then, substituting $p(\mathbf{x}, t)$ by $p(\mathbf{x})$ when $t \rightarrow \infty$, we can show that

$$\begin{aligned} -\partial_{\mathbf{x}} \left\{ [\partial_{\mathbf{x}}(\log p(\mathbf{x}))] p(\mathbf{x}) \right\} + \partial_{\mathbf{x}}^2 p(\mathbf{x}) &= \partial_{\mathbf{x}} \left\{ [-\partial_{\mathbf{x}}(\log p(\mathbf{x}))] p(\mathbf{x}) + \partial_{\mathbf{x}} p(\mathbf{x}) \right\} \\ &= \partial_{\mathbf{x}} \left\{ -\frac{\partial_{\mathbf{x}} p(\mathbf{x})}{p(\mathbf{x})} p(\mathbf{x}) + \partial_{\mathbf{x}} p(\mathbf{x}) \right\} \\ &= \partial_{\mathbf{x}} \left\{ -\partial_{\mathbf{x}} p(\mathbf{x}) + \partial_{\mathbf{x}} p(\mathbf{x}) \right\} = 0. \end{aligned}$$

On the other hand, when $t \rightarrow \infty$, it holds that $\partial_t p(\mathbf{x}) = 0$. Therefore, the Fokker-Planck equation is verified.

Example 3.2. Consider a Gaussian mixture $p(x) = \pi_1 \mathcal{N}(x | \mu_1, \sigma_1^2) + \pi_2 \mathcal{N}(x | \mu_2, \sigma_2^2)$. We can calculate the gradient $\nabla_x \log p(x)$ analytically or numerically. For demonstration, we choose $\pi_1 = 0.6$, $\mu_1 = 2$, $\sigma_1 = 0.5$, $\pi_2 = 0.4$, $\mu_2 = -2$, $\sigma_2 = 0.2$. We initialize $x_0 = 0$. We choose $\tau = 0.05$. We run the above gradient descent iteration for $T = 500$ times, and we plot the trajectory of the values $p(x_t)$ for $t = 1, \dots, T$. As we can see in the figure below, the sequence $\{x_1, x_2, \dots, x_T\}$ simply follows the shape of the Gaussian and climb to one of the peaks.

What is more interesting is when we add the noise term. Instead of landing at the peak, the sequence x_t moves around the peak and finishes somewhere near the peak. (Remark: To terminate the algorithm, we can gradually make τ smaller or we can early stop.)

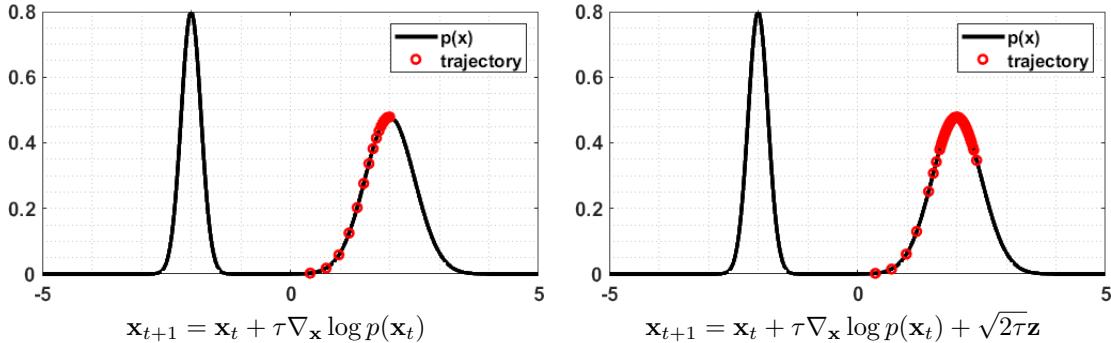


Figure 3.2: Deterministic algorithm aiming to pick a sample that maximizes the likelihood, versus a stochastic algorithm which adds noise at every iteration.

Figure 3.3 shows an interesting description of the sample trajectory. Starting with an arbitrary location, the data point \mathbf{x}_t will do a random walk according to the Langevin dynamics equation. The direction of the

random walk is not completely arbitrary. There is a certain amount of pre-defined drift while at every step there is some level of randomness. The drift is determined by $\nabla_{\mathbf{x}} \log p(\mathbf{x})$ whereas the randomness comes from \mathbf{z} .

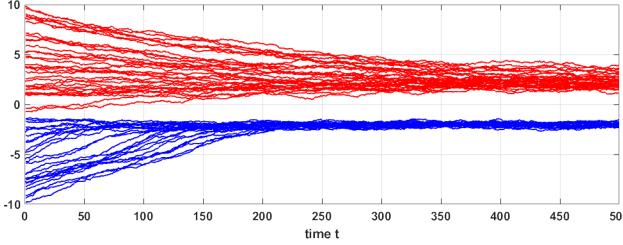


Figure 3.3: Trajectory of sample evolutions using the Langevin dynamics. We colored the two modes of the Gaussian mixture in different colors for better visualization. The setting here is identical to the example above, except that the step size is $\tau = 0.001$.

Example 3.3. Following the previous example we again consider a Gaussian mixture

$$p(x) = \pi_1 \mathcal{N}(x | \mu_1, \sigma_1^2) + \pi_2 \mathcal{N}(x | \mu_2, \sigma_2^2).$$

We choose $\pi_1 = 0.6$, $\mu_1 = 2$, $\sigma_1 = 0.5$, $\pi_2 = 0.4$, $\mu_2 = -2$, $\sigma_2 = 0.2$. Suppose we initialize $M = 10000$ uniformly distributed samples $x_0 \sim \text{Uniform}[-3, 3]$. We run Langevin updates for $t = 100$ steps. The histograms of generated samples are shown in the figures below.

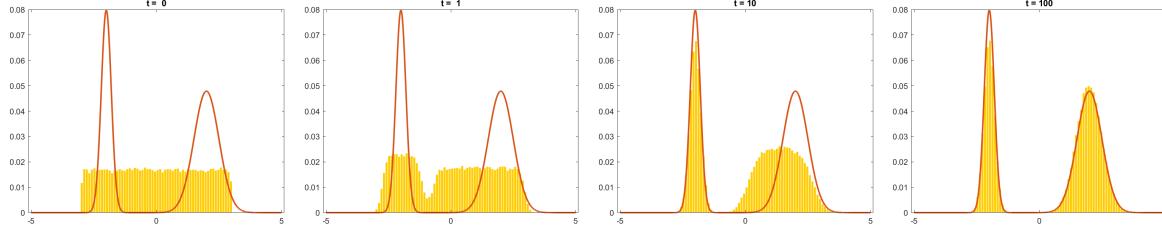


Figure 3.4: Samples generated by Langevin dynamics. Initially the samples are uniformly distributed. As time progresses, the distribution of the samples become the desired distribution.

Remark 1: Stochastic Gradient Langevin Dynamics. The dynamical behavior of \mathbf{x}_t governed by the Langevin equation is often known as the Langevin dynamics. Langevin dynamics uses gradient descent plus noise. This is not the same as stochastic gradient descent (SGD). SGD uses minibatches to approximate the full gradient. There is no noise. The randomness in SGD comes from the minibatch which are uniformly sampled from the training dataset. SGD can be paired with Langevin dynamics to make stochastic gradient Langevin dynamics as outlined in [46] which provided a clear comparison in the context of classical maximum-a-posteriori (MAP) estimation.

$$\text{Stochastic Gradient Descent} \quad \Delta \boldsymbol{\theta}_t = \frac{\epsilon_t}{2} \left(\nabla_{\boldsymbol{\theta}} \log p(\boldsymbol{\theta}_t) + \frac{N}{n} \sum_{i=1}^n \nabla \log p(\mathbf{y}_{t_i} | \boldsymbol{\theta}_t) \right)$$

$$\text{Langevin Dynamics} \quad \Delta \boldsymbol{\theta}_t = \frac{\epsilon}{2} \left(\nabla_{\boldsymbol{\theta}} \log p(\boldsymbol{\theta}_t) + \sum_{i=1}^N \nabla \log p(\mathbf{y}_i | \boldsymbol{\theta}_t) \right) + \boldsymbol{\eta}_t$$

$$\text{SG Langevin Dynamics} \quad \Delta \boldsymbol{\theta}_t = \frac{\epsilon_t}{2} \left(\nabla_{\boldsymbol{\theta}} \log p(\boldsymbol{\theta}_t) + \frac{N}{n} \sum_{i=1}^n \nabla \log p(\mathbf{y}_{t_i} | \boldsymbol{\theta}_t) \right) + \boldsymbol{\eta}_t,$$

where ϵ_t is a sequence of step sizes satisfying the properties that $\sum_{t=1}^{\infty} \epsilon_t = \infty$ and $\sum_{t=1}^{\infty} \epsilon_t^2 < \infty$, and $\eta_t \sim \mathcal{N}(0, \mathbf{I})$ is white noise. The set $\{\mathbf{y}_1, \dots, \mathbf{y}_N\}$ denotes the training set. The constants N and n denote the number of training samples in the full training set and in the minibatch, respectively.

3.2 (Stein's) Score Function

Putting aside the Langevin dynamics and the underlying Fokker-Planck equation, the key subject of the iterative procedure is the gradient of the log-likelihood function $\nabla_{\mathbf{x}} \log p(\mathbf{x})$. It has a formal name known as the **Stein's score function**, denoted by

$$\mathbf{s}_{\theta}(\mathbf{x}) \stackrel{\text{def}}{=} \nabla_{\mathbf{x}} \log p_{\theta}(\mathbf{x}). \quad (3.3)$$

We should be careful not to confuse Stein's score function with the **ordinary score function** which is defined as

$$\mathbf{s}_{\mathbf{x}}(\boldsymbol{\theta}) \stackrel{\text{def}}{=} \nabla_{\boldsymbol{\theta}} \log p_{\boldsymbol{\theta}}(\mathbf{x}). \quad (3.4)$$

The ordinary score function is the gradient (with respect to $\boldsymbol{\theta}$) of the log-likelihood. In contrast, Stein's score function is the gradient with respect to the data point \mathbf{x} . Maximum likelihood estimation uses the ordinary score function, whereas Langevin dynamics uses Stein's score function. However, since most people in the diffusion literature calls Stein's score function as the score function, we follow this culture.

Example 3.4. If $p(x)$ is a Gaussian with $p(x) = \frac{1}{\sqrt{2\pi\sigma^2}} e^{-\frac{(x-\mu)^2}{2\sigma^2}}$, then

$$s(x) = \nabla_x \log p(x) = -\frac{(x-\mu)}{\sigma^2}.$$

Example 3.5. If $p(x)$ is a Gaussian mixture with $p(x) = \sum_{i=1}^N \pi_i \frac{1}{\sqrt{2\pi\sigma_i^2}} e^{-\frac{(x-\mu_i)^2}{2\sigma_i^2}}$, then

$$s(x) = \nabla_x \log p(x) = -\frac{\sum_{j=1}^N \pi_j \frac{1}{\sqrt{2\pi\sigma_j^2}} e^{-\frac{(x-\mu_j)^2}{2\sigma_j^2}} \frac{(x-\mu_j)}{\sigma_j^2}}{\sum_{i=1}^N \pi_i \frac{1}{\sqrt{2\pi\sigma_i^2}} e^{-\frac{(x-\mu_i)^2}{2\sigma_i^2}}}.$$

The probability density function and the corresponding score function of the above two examples are shown in Figure 3.5.

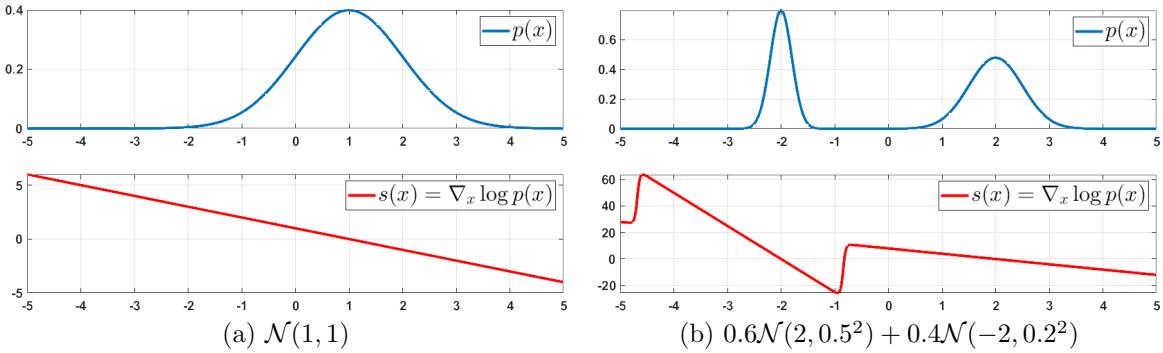


Figure 3.5: Examples of score functions

Geometric Interpretations of the Score Function. The way to understand the score function is to remember that it is the gradient with respect to the data \mathbf{x} . For any high-dimensional distribution $p(\mathbf{x})$, the gradient will give us a vector field. There are a few useful interpretations of the score functions:

- The magnitude of the vectors are the strongest at places where the change of $\log p(\mathbf{x})$ is the biggest. Therefore, in regions where $\log p(\mathbf{x})$ is close to the peak will be mostly very weak gradient.
- The vector field indicates *how* a data point should travel in the contour map. In Figure 3.6 we show the contour map of a Gaussian mixture (with two Gaussians). We draw arrows to indicate the vector field. Now if we consider a data point living the space, the Langevin dynamics equation will basically move the data points along the direction pointed by the vector field towards the basin.
- In physics, the score function is equivalent to the “*drift*”. This name suggests how the diffusion particles should flow to the lowest energy state.

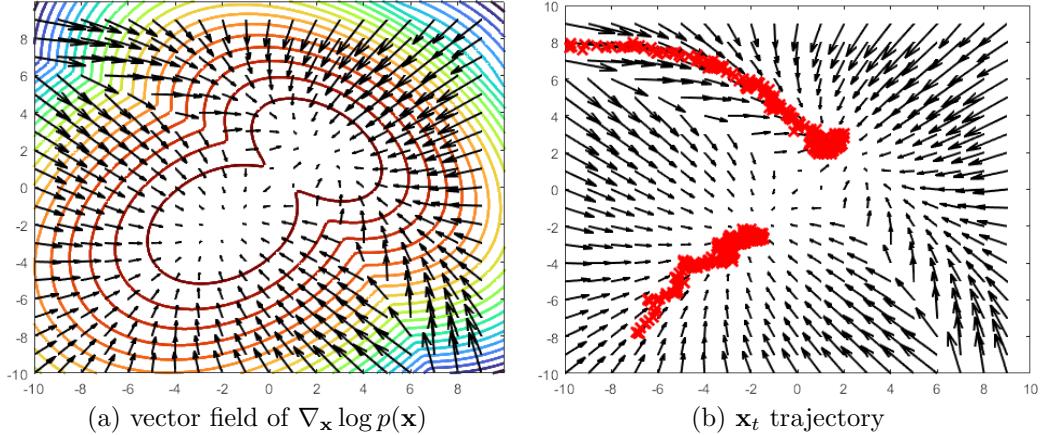


Figure 3.6: The contour map of the score function, and the corresponding trajectory of two samples.

3.3 Score Matching Techniques

The most difficult question in Langevin dynamics is how to obtain $\nabla_{\mathbf{x}} \log p(\mathbf{x})$ because we have no access to $p(\mathbf{x})$. In this section, we briefly discuss a few known techniques.

Explicit Score-Matching [43]. Suppose that we are given a dataset $\mathcal{X} = \{\mathbf{x}^{(1)}, \dots, \mathbf{x}^{(M)}\}$. The solution people came up with is to consider the classical kernel density estimation by defining a distribution

$$q_h(\mathbf{x}) = \frac{1}{M} \sum_{m=1}^M \frac{1}{h} K\left(\frac{\mathbf{x} - \mathbf{x}^{(m)}}{h}\right), \quad (3.5)$$

where h is just some hyperparameter for the kernel function $K(\cdot)$, and $\mathbf{x}^{(m)}$ is the m -th sample in the training set. Figure 3.7 illustrates the idea of kernel density estimation. In the cartoon figure shown on the left, we show multiple kernels $K(\cdot)$ centered at different data points $\mathbf{x}^{(m)}$. The sum of all these individual kernels gives us the overall kernel density estimate $q(\mathbf{x})$. On the right hand side we show a real histogram and the corresponding kernel density estimate. We remark that $q(\mathbf{x})$ is at best an approximation to the true data distribution $p(\mathbf{x})$ which is never known.

Since $q(\mathbf{x})$ is an approximation to $p(\mathbf{x})$ which is never accessible, we can learn $\mathbf{s}_{\theta}(\mathbf{x})$ based on $q(\mathbf{x})$. This leads to the following definition of a loss function which can be used to train a network.

Theorem 3.2. The explicit score matching loss is

$$\begin{aligned} J_{\text{ESM}}(\theta) &\stackrel{\text{def}}{=} \frac{1}{2} \mathbb{E}_{p(\mathbf{x})} \|\mathbf{s}_{\theta}(\mathbf{x}) - \nabla_{\mathbf{x}} \log p(\mathbf{x})\|^2 \\ &\approx \frac{1}{2} \mathbb{E}_{q_h(\mathbf{x})} \|\mathbf{s}_{\theta}(\mathbf{x}) - \nabla_{\mathbf{x}} \log q_h(\mathbf{x})\|^2. \end{aligned} \quad (3.6)$$

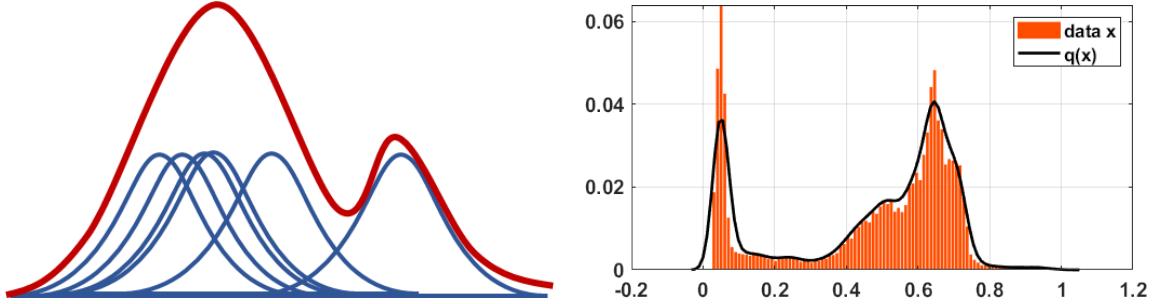


Figure 3.7: Illustration of kernel density estimation.

By substituting the kernel density estimation, we can show that the loss is

$$\begin{aligned}
 J_{\text{ESM}}(\boldsymbol{\theta}) &= \mathbb{E}_{q_h(\mathbf{x})} \|\mathbf{s}_{\boldsymbol{\theta}}(\mathbf{x}) - \nabla_{\mathbf{x}} \log q_h(\mathbf{x})\|^2 \\
 &= \int \|\mathbf{s}_{\boldsymbol{\theta}}(\mathbf{x}) - \nabla_{\mathbf{x}} \log q_h(\mathbf{x})\|^2 q_h(\mathbf{x}) d\mathbf{x} \\
 &\approx \frac{1}{M} \sum_{m=1}^M \int \|\mathbf{s}_{\boldsymbol{\theta}}(\mathbf{x}) - \nabla_{\mathbf{x}} \log q_h(\mathbf{x})\|^2 \frac{1}{h} K\left(\frac{\mathbf{x} - \mathbf{x}^{(m)}}{h}\right) d\mathbf{x}.
 \end{aligned} \tag{3.7}$$

So, we have derived a loss function that can be used to train the network. Once we train the network $\mathbf{s}_{\boldsymbol{\theta}}$, we can replace it in the Langevin dynamics equation to obtain the recursion:

$$\mathbf{x}_{t+1} = \mathbf{x}_t + \tau \mathbf{s}_{\boldsymbol{\theta}}(\mathbf{x}_t) + \sqrt{2\tau} \mathbf{z}. \tag{3.8}$$

The issue of explicit score matching is that the kernel density estimation is a fairly poor non-parameter estimation of the true distribution. Especially when we have a limited number of samples and the samples live in a high dimensional space, the kernel density estimation performance can be poor.

Implicit Score Matching [18]. In implicit score matching, the explicit score matching loss is replaced by an implicit one.

$$J_{\text{ISM}}(\boldsymbol{\theta}) \stackrel{\text{def}}{=} \mathbb{E}_{p(\mathbf{x})} \left[\text{Tr}(\nabla_{\mathbf{x}} \mathbf{s}_{\boldsymbol{\theta}}(\mathbf{x})) + \frac{1}{2} \|\mathbf{s}_{\boldsymbol{\theta}}(\mathbf{x})\|^2 \right], \tag{3.9}$$

where $\nabla_{\mathbf{x}} \mathbf{s}_{\boldsymbol{\theta}}(\mathbf{x})$ denotes the Jacobian of $\mathbf{s}_{\boldsymbol{\theta}}(\mathbf{x})$. The implicit score matching loss can be approximated by Monte Carlo

$$J_{\text{ISM}}(\boldsymbol{\theta}) \approx \frac{1}{M} \sum_{m=1}^M \sum_i \left(\partial_i \mathbf{s}_{\boldsymbol{\theta}}(\mathbf{x}^{(m)}) + \frac{1}{2} [\mathbf{s}_{\boldsymbol{\theta}}(\mathbf{x}^{(m)})]_i^2 \right),$$

where $\partial_i \mathbf{s}_{\boldsymbol{\theta}}(\mathbf{x}^{(m)}) = \frac{\partial}{\partial x_i} [\mathbf{s}_{\boldsymbol{\theta}}(\mathbf{x})]_i = \frac{\partial^2}{\partial x_i^2} \log p(\mathbf{x})$. If the model for the score function is realized by a deep neural network, the trace operator can be difficult to compute, hence making the implicit score matching not scalable [40].

Denoising Score Matching. Given the potential drawbacks of explicit and implicit score matching, we now introduce a more popular score matching known as the denoising score matching (DSM) by Vincent [43]. In DSM, the loss function is defined as follows.

$$J_{\text{DSM}}(\boldsymbol{\theta}) \stackrel{\text{def}}{=} \mathbb{E}_{q(\mathbf{x}, \mathbf{x}') \left[\frac{1}{2} \|\mathbf{s}_{\boldsymbol{\theta}}(\mathbf{x}) - \nabla_{\mathbf{x}} \log q(\mathbf{x} | \mathbf{x}')\|^2 \right]} \tag{3.10}$$

The key difference here is that we replace the distribution $q(\mathbf{x})$ by a conditional distribution $q(\mathbf{x} | \mathbf{x}')$. The former requires an approximation, e.g., via kernel density estimation, whereas the latter does not.

In the special case where $q(\mathbf{x}|\mathbf{x}') = \mathcal{N}(\mathbf{x} | \mathbf{x}', \sigma^2)$, we can let $\mathbf{x} = \mathbf{x}' + \sigma\mathbf{z}$. This will give us

$$\begin{aligned}\nabla_{\mathbf{x}} \log q(\mathbf{x}|\mathbf{x}') &= \nabla_{\mathbf{x}} \log \frac{1}{(\sqrt{2\pi\sigma^2})^d} \exp \left\{ -\frac{\|\mathbf{x} - \mathbf{x}'\|^2}{2\sigma^2} \right\} \\ &= \nabla_{\mathbf{x}} \left\{ -\frac{\|\mathbf{x} - \mathbf{x}'\|^2}{2\sigma^2} - \log(\sqrt{2\pi\sigma^2})^d \right\} \\ &= -\frac{\mathbf{x} - \mathbf{x}'}{\sigma^2} = -\frac{\mathbf{z}}{\sigma}.\end{aligned}$$

As a result, the loss function of the denoising score matching becomes

$$\begin{aligned}J_{\text{DSM}}(\boldsymbol{\theta}) &\stackrel{\text{def}}{=} \mathbb{E}_{q(\mathbf{x}, \mathbf{x}')} \left[\frac{1}{2} \|\mathbf{s}_{\boldsymbol{\theta}}(\mathbf{x}) - \nabla_{\mathbf{x}} \log q(\mathbf{x}|\mathbf{x}')\|^2 \right] \\ &= \mathbb{E}_{q(\mathbf{x}')} \left[\frac{1}{2} \left\| \mathbf{s}_{\boldsymbol{\theta}}(\mathbf{x}' + \sigma\mathbf{z}) + \frac{\mathbf{z}}{\sigma} \right\|^2 \right].\end{aligned}$$

If we replace the dummy variable \mathbf{x}' by \mathbf{x} , and we note that sampling from $q(\mathbf{x})$ can be replaced by sampling from $p(\mathbf{x})$ when we are given a training dataset, we can conclude the following.

Theorem 3.3. The Denoising Score Matching has a loss function defined as

$$J_{\text{DSM}}(\boldsymbol{\theta}) = \mathbb{E}_{p(\mathbf{x})} \left[\frac{1}{2} \left\| \mathbf{s}_{\boldsymbol{\theta}}(\mathbf{x} + \sigma\mathbf{z}) + \frac{\mathbf{z}}{\sigma} \right\|^2 \right] \quad (3.11)$$

The beauty about Eqn (3.11) is that it is highly interpretable. The quantity $\mathbf{x} + \sigma\mathbf{z}$ is effectively adding noise $\sigma\mathbf{z}$ to a clean image \mathbf{x} . The score function $\mathbf{s}_{\boldsymbol{\theta}}$ is supposed to take this noisy image and predict the noise $\frac{\mathbf{z}}{\sigma}$. Predicting noise is equivalent to denoising, because any denoised image plus the predicted noise will give us the noisy observation. Therefore, Eqn (3.11) is a *denoising* step.

The following theorem, proven by Vincent [43], establishes the equivalence between DSM and ESM. It is this equivalence that allows us to use DSM to estimate the score function.

Theorem 3.4. [Vincent [43]] For up to a constant C which is independent of the variable $\boldsymbol{\theta}$, it holds that

$$J_{\text{DSM}}(\boldsymbol{\theta}) = J_{\text{ESM}}(\boldsymbol{\theta}) + C. \quad (3.12)$$

Proof of Theorem 3.4 The proof here is based on [43]. We start with the explicit score matching loss function, which is given by

$$\begin{aligned}J_{\text{ESM}}(\boldsymbol{\theta}) &= \mathbb{E}_{q(\mathbf{x})} \left[\frac{1}{2} \|\mathbf{s}_{\boldsymbol{\theta}}(\mathbf{x}) - \nabla_{\mathbf{x}} \log q(\mathbf{x})\|^2 \right] \\ &= \mathbb{E}_{q(\mathbf{x})} \left[\frac{1}{2} \|\mathbf{s}_{\boldsymbol{\theta}}(\mathbf{x})\|^2 - \mathbf{s}_{\boldsymbol{\theta}}(\mathbf{x})^T \nabla_{\mathbf{x}} \log q(\mathbf{x}) + \underbrace{\frac{1}{2} \|\nabla_{\mathbf{x}} \log q(\mathbf{x})\|^2}_{\stackrel{\text{def}}{=} C_1, \text{independent of } \boldsymbol{\theta}} \right].\end{aligned}$$

Let's zoom into the second term. We can show that

$$\begin{aligned}\mathbb{E}_{q(\mathbf{x})} [\mathbf{s}_{\boldsymbol{\theta}}(\mathbf{x})^T \nabla_{\mathbf{x}} \log q(\mathbf{x})] &= \int (\mathbf{s}_{\boldsymbol{\theta}}(\mathbf{x})^T \nabla_{\mathbf{x}} \log q(\mathbf{x})) q(\mathbf{x}) d\mathbf{x}, \quad (\text{expectation}) \\ &= \int \left(\mathbf{s}_{\boldsymbol{\theta}}(\mathbf{x})^T \frac{\nabla_{\mathbf{x}} q(\mathbf{x})}{q(\mathbf{x})} \right) q(\mathbf{x}) d\mathbf{x}, \quad (\text{gradient}) \\ &= \int \mathbf{s}_{\boldsymbol{\theta}}(\mathbf{x})^T \nabla_{\mathbf{x}} q(\mathbf{x}) d\mathbf{x}.\end{aligned}$$

Next, we consider conditioning by recalling $q(\mathbf{x}) = \int q(\mathbf{x}')q(\mathbf{x}|\mathbf{x}')d\mathbf{x}'$. This will give us

$$\begin{aligned}
\int \mathbf{s}_\theta(\mathbf{x})^T \nabla_{\mathbf{x}} q(\mathbf{x}) d\mathbf{x} &= \int \mathbf{s}_\theta(\mathbf{x})^T \underbrace{\nabla_{\mathbf{x}} \left(\int q(\mathbf{x}') q(\mathbf{x}|\mathbf{x}') d\mathbf{x}' \right)}_{=q(\mathbf{x})} d\mathbf{x} && \text{(conditional)} \\
&= \int \mathbf{s}_\theta(\mathbf{x})^T \left(\int q(\mathbf{x}') \nabla_{\mathbf{x}} q(\mathbf{x}|\mathbf{x}') d\mathbf{x}' \right) d\mathbf{x} && \text{(move gradient)} \\
&= \int \mathbf{s}_\theta(\mathbf{x})^T \left(\int q(\mathbf{x}') \nabla_{\mathbf{x}} q(\mathbf{x}|\mathbf{x}') \times \frac{q(\mathbf{x}|\mathbf{x}')}{q(\mathbf{x}|\mathbf{x}')} d\mathbf{x}' \right) d\mathbf{x} && \text{(multiple and divide)} \\
&= \int \mathbf{s}_\theta(\mathbf{x})^T \int q(\mathbf{x}') \underbrace{\left(\frac{\nabla_{\mathbf{x}} q(\mathbf{x}|\mathbf{x}')}{q(\mathbf{x}|\mathbf{x}')}\right)}_{=\nabla_{\mathbf{x}} \log q(\mathbf{x}|\mathbf{x}')} q(\mathbf{x}|\mathbf{x}') d\mathbf{x}' d\mathbf{x} && \text{(rearrange terms)} \\
&= \int \mathbf{s}_\theta(\mathbf{x})^T \left(\int q(\mathbf{x}') \left(\nabla_{\mathbf{x}} \log q(\mathbf{x}|\mathbf{x}') \right) q(\mathbf{x}|\mathbf{x}') d\mathbf{x}' \right) d\mathbf{x} \\
&= \int \int \underbrace{q(\mathbf{x}|\mathbf{x}') q(\mathbf{x}')}_{=q(\mathbf{x}, \mathbf{x}')} \left(\mathbf{s}_\theta(\mathbf{x})^T \nabla_{\mathbf{x}} \log q(\mathbf{x}|\mathbf{x}') \right) d\mathbf{x}' d\mathbf{x} && \text{(move integration)} \\
&= \mathbb{E}_{q(\mathbf{x}, \mathbf{x}')} [\mathbf{s}_\theta(\mathbf{x})^T \nabla_{\mathbf{x}} \log q(\mathbf{x}|\mathbf{x}')].
\end{aligned}$$

So, if we substitute this result back to the definition of ESM, we can show that

$$J_{\text{ESM}}(\boldsymbol{\theta}) = \mathbb{E}_{q(\mathbf{x})} \left[\frac{1}{2} \|\mathbf{s}_\theta(\mathbf{x})\|^2 \right] - \mathbb{E}_{q(\mathbf{x}, \mathbf{x}')} [\mathbf{s}_\theta(\mathbf{x})^T \nabla_{\mathbf{x}} \log q(\mathbf{x}|\mathbf{x}')] + C_1.$$

Comparing this with the definition of DSM, we can observe that

$$\begin{aligned}
J_{\text{DSM}}(\boldsymbol{\theta}) &\stackrel{\text{def}}{=} \mathbb{E}_{q(\mathbf{x}, \mathbf{x}')} \left[\frac{1}{2} \|\mathbf{s}_\theta(\mathbf{x}) - \nabla_{\mathbf{x}} q(\mathbf{x}|\mathbf{x}')\|^2 \right] \\
&= \mathbb{E}_{q(\mathbf{x}, \mathbf{x}')} \left[\frac{1}{2} \|\mathbf{s}_\theta(\mathbf{x})\|^2 - \mathbf{s}_\theta(\mathbf{x})^T \nabla_{\mathbf{x}} \log q(\mathbf{x}|\mathbf{x}') + \underbrace{\frac{1}{2} \|\nabla_{\mathbf{x}} \log q(\mathbf{x}|\mathbf{x}')\|^2}_{\stackrel{\text{def}}{=} C_2, \text{independent of } \boldsymbol{\theta}} \right] \\
&= \mathbb{E}_{q(\mathbf{x})} \left[\frac{1}{2} \|\mathbf{s}_\theta(\mathbf{x})\|^2 \right] - \mathbb{E}_{q(\mathbf{x}, \mathbf{x}')} [\mathbf{s}_\theta(\mathbf{x})^T \nabla_{\mathbf{x}} \log q(\mathbf{x}|\mathbf{x}')] + C_2.
\end{aligned}$$

Therefore, we conclude that

$$J_{\text{DSM}}(\boldsymbol{\theta}) = J_{\text{ESM}}(\boldsymbol{\theta}) - C_1 + C_2.$$

The **training** procedure in a score matching model is typically done by minimizing the denoising score matching loss function. If we are given a training dataset $\{\mathbf{x}^{(m)}\}_{m=1}^M$, the optimization goal is to

$$\begin{aligned}
\boldsymbol{\theta}^* &= \operatorname{argmin}_{\boldsymbol{\theta}} \mathbb{E}_{p(\mathbf{x})} \left[\frac{1}{2} \left\| \mathbf{s}_\theta(\mathbf{x} + \sigma \mathbf{z}) + \frac{\mathbf{z}}{\sigma} \right\|^2 \right] \\
&\approx \operatorname{argmin}_{\boldsymbol{\theta}} \frac{1}{M} \sum_{m=1}^M \frac{1}{2} \left\| \mathbf{s}_\theta \left(\mathbf{x}^{(m)} + \sigma \mathbf{z}^{(m)} \right) + \frac{\mathbf{z}^{(m)}}{\sigma} \right\|^2, \quad \text{where } \mathbf{z}^{(m)} \sim \mathcal{N}(0, \mathbf{I}).
\end{aligned}$$

Figure 3.8 illustrates the training procedure of the score function $\mathbf{s}_\theta(\mathbf{x})$.

The above training procedure assumes a fixed noise level σ . Generalizing it to multiple noise levels is not difficult. The noise conditioned score network (NCSN) by Song and Ermon [40] argued that one can instead optimize the following loss

$$J_{\text{NCSN}}(\theta) = \frac{1}{L} \sum_{i=1}^L \lambda(\sigma_i) \ell(\boldsymbol{\theta}; \sigma_i), \tag{3.13}$$

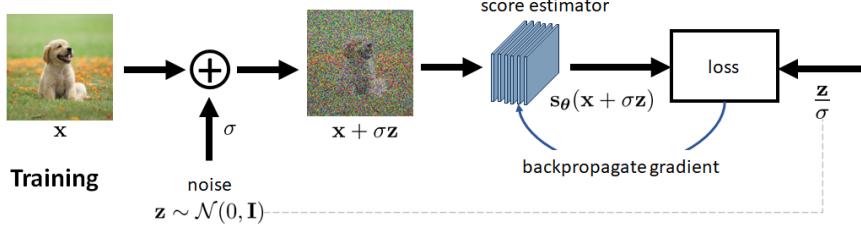


Figure 3.8: Training of s_θ for denoising score matching. The network s_θ is trained to estimate the noise.

where the individual loss function is defined according to the noise levels $\sigma_1, \dots, \sigma_L$:

$$\ell(\theta; \sigma) = \mathbb{E}_{p(\mathbf{x})} \left[\frac{1}{2} \left\| \mathbf{s}_\theta(\mathbf{x} + \sigma \mathbf{z}) + \frac{\mathbf{z}}{\sigma} \right\|^2 \right].$$

The coefficient function $\lambda(\sigma_i)$ is often chosen as $\lambda(\sigma) = \sigma^2$ based on empirical findings [40]. The noise level sequence often satisfies $\frac{\sigma_1}{\sigma_2} = \dots = \frac{\sigma_{L-1}}{\sigma_L} > 1$.

For **inference**, we assume that we have already trained the score estimator s_θ . To generate an image, we use the Langevin equation to iteratively draw samples by denoising the image. In case of NCSN, the corresponding Langevin equation can be implemented via an annealed importance sampling:

$$\mathbf{x}_{t+1} = \mathbf{x}_t + \frac{\alpha_i}{2} \mathbf{s}_\theta(\mathbf{x}_t, \sigma_i) + \sqrt{\alpha_i} \mathbf{z}_t, \quad \mathbf{z}_t \sim \mathcal{N}(0, \mathbf{I}),$$

where $\alpha_i = \sigma_i^2 / \sigma_L^2$ is the step size and $\mathbf{s}_\theta(\mathbf{x}_t, \sigma_i)$ denotes the score matching function for noise level σ_i . The iteration over t is repeated sequentially for each σ_i from $i = 1$ to L . For additional details of the implementation, we refer readers to Algorithm 1 of the original paper by Song and Ermon [40].

3.4 Concluding Remark

Additional readings about score-matching should start with Vincent's technical report [43]. A very popular paper in the recent literature is Song and Ermon [40], their follow up work [41], and [42]. In their papers, they brought up an important discussion that when training a score function, we need a noise schedule so that the score function is trained better.

Score matching has a wide range of applicability beyond generating images. They can be used in solving many important image restoration problems such as deblurring, denoising, super-resolution, etc. Kadkhodaie and Simoncellil [19] is among the earlier papers to explicitly employ the score function as part of the image reconstruction process. A similar concept is presented by Kawar et al. [21], where they sample from a posterior distribution which contains the forward image formation model and the prior. For problems with a better structured forward model, it has been shown that employing the ideas of proximal maps would improve the performance. This line of work can be built upon the operator splitting strategy such as the plug-and-play ADMM [8] by extending it to Plug-and-Play diffusion model, e.g., Zhu et al. [?] or the generative plug and play model by Bouman and Buzzard [5]. Recently, it was also observed that one can directly perform the regression to mean when we do not have access to the degradation process [11]. Various applications papers [9, 17, 37].

4 Stochastic Differential Equation (SDE)

In the previous two sections we studied the diffusion models via the DDPM and the SMLD perspectives. In this section, we will study diffusion models through the lens of differential equation. As we mentioned during our discussions about the Langevin equation, the steady state solution of the Langevin equation is a random variable whose probability distribution satisfies the Fokker-Planck equation. This unusual linkage between the iterative algorithm and a differential equation makes the topic remarkably interesting. The purpose of this section is to provide the basic principles of stochastic differential equations and how they are used to understand diffusion models.

4.1 From Iterative Algorithms to Ordinary Differential Equations

The first and the foremost question, at least to readers with little background in differential equations, is why an iterative algorithm can be related to a differential equation. Let's understand this through two examples.

Example 4.1. Simple First-Order ODE. Imagine that we are given a discrete-time algorithm with the iterations defined by the recursion:

$$\mathbf{x}_i = \left(1 - \frac{\beta\Delta t}{2}\right) \mathbf{x}_{i-1}, \quad \text{for } i = 1, 2, \dots, N, \quad (4.1)$$

for some hyperparameter β and a step-size parameter Δt . We can turn this iterative scheme into a continuous-time differential equation.

Suppose that there is a continuous time function $\mathbf{x}(t)$. We define a discretization scheme by letting $\mathbf{x}_i = \mathbf{x}(\frac{i}{N})$ for $i = 1, \dots, N$, and $\Delta t = \frac{1}{N}$, and $t \in \{0, \frac{1}{N}, \dots, \frac{N-1}{N}\}$. Then the above recursion can be written as

$$\mathbf{x}(t + \Delta t) = \left(1 - \frac{\beta\Delta t}{2}\right) \mathbf{x}(t).$$

Rearranging the terms will give us

$$\frac{\mathbf{x}(t + \Delta t) - \mathbf{x}(t)}{\Delta t} = -\frac{\beta}{2} \mathbf{x}(t),$$

where at the limit when $\Delta t \rightarrow 0$, we can write the discrete equation as an ordinary differential equation (ODE)

$$\frac{d\mathbf{x}(t)}{dt} = -\frac{\beta}{2} \mathbf{x}(t). \quad (4.2)$$

Not only that, we can solve for an analytic solution for the ODE where the solution is given by

$$\mathbf{x}(t) = e^{-\frac{\beta}{2}t}. \quad (4.3)$$

Verification of the solution can be done by substituting Eqn (4.3) into Eqn (4.2).

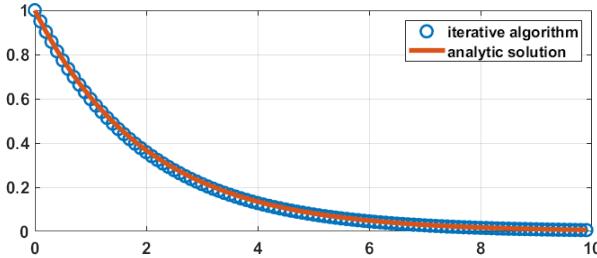


Figure 4.1: Analytic solution and the estimates produced by the numerical scheme.

The power of the ODE is that it offers us an *analytic* solution. Instead of resorting to the iterative scheme (which will take hundreds to thousands of iterations), the analytic solution tells us exactly the behavior of the solution at *any* time t . To illustrate this fact, we show in the figure above the trajectory of the solution $\mathbf{x}_1, \mathbf{x}_2, \dots, \mathbf{x}_i, \dots, \mathbf{x}_N$ defined by the algorithm. Here, we choose $\Delta t = 0.1$. In the same plot, we directly plot the continuous-time solution $\mathbf{x}(t) = \exp\{-\beta t/2\}$ for arbitrary t . As shown, the analytic solution is exactly the same as the trajectory predicted by the iterative scheme.

What we observe in this motivating example are two interesting facts:

- The discrete-time iterative scheme can be written as a continuous-time ODE. In fact, many discrete-time algorithms have their associating ODEs.
- For simple ODEs, we can write down the analytic solution in closed form. More complicated ODE would be hard to write an analytic solution. But we can still use ODE tools to analyze the behavior of the solution. We can also derive the limiting solution $t \rightarrow 0$.

Example 4.2. Gradient Descent. Recall that a gradient descent algorithm for a (well-behaved) convex function f is the following recursion. For $i = 1, 2, \dots, N$, do

$$\mathbf{x}_i = \mathbf{x}_{i-1} - \beta_{i-1} \nabla f(\mathbf{x}_{i-1}), \quad (4.4)$$

for step-size parameter β_i . Using the same discretization as we did in the previous example, we can show that (by letting $\beta_{i-1} = \beta(t)\Delta t$):

$$\begin{aligned} \mathbf{x}_i = \mathbf{x}_{i-1} - \beta_{i-1} \nabla f(\mathbf{x}_{i-1}) &\implies \mathbf{x}(t + \Delta t) = \mathbf{x}(t) - \beta(t)\Delta t \nabla f(\mathbf{x}(t)) \\ &\implies \frac{\mathbf{x}(t + \Delta t) - \mathbf{x}(t)}{\Delta t} = -\beta(t) \nabla f(\mathbf{x}(t)) \\ &\implies \frac{d\mathbf{x}(t)}{dt} = -\beta(t) \nabla f(\mathbf{x}(t)). \end{aligned} \quad (4.5)$$

The ordinary differential equation shown on the right has a solution trajectory $\mathbf{x}(t)$. This $\mathbf{x}(t)$ is known as the *gradient flow* of the function f .

For simplicity, we can make $\beta(t) = \beta$ for all t . Then there are two simple facts about this ODE. First, we can show that

$$\begin{aligned} \frac{d}{dt} f(\mathbf{x}(t)) &= \nabla f(\mathbf{x}(t))^T \frac{d\mathbf{x}(t)}{dt} && \text{(chain rule)} \\ &= \nabla f(\mathbf{x}(t))^T [-\beta \nabla f(\mathbf{x}(t))] && \text{(Eqn (4.5))} \\ &= -\beta \nabla f(\mathbf{x}(t))^T \nabla f(\mathbf{x}(t)) \\ &= -\beta \|\nabla f(\mathbf{x}(t))\|^2 \leq 0 && \text{(norm-squares).} \end{aligned}$$

Therefore, as we move from \mathbf{x}_{i-1} to \mathbf{x}_i , the objective value $f(\mathbf{x}(t))$ has to go down. This is consistent with our expectation because a gradient descent algorithm should bring the cost down as the iteration goes on. Second, at the limit when $t \rightarrow \infty$, we know that $\frac{d\mathbf{x}(t)}{dt} \rightarrow 0$. Hence, $\frac{d\mathbf{x}(t)}{dt} = -\beta \nabla f(\mathbf{x}(t))$ will imply that

$$\nabla f(\mathbf{x}(t)) \rightarrow 0, \quad \text{as } t \rightarrow \infty. \quad (4.6)$$

Therefore, the solution trajectory $\mathbf{x}(t)$ will approach the minimizer for the function f .

Forward and Backward Updates.

Let's use the gradient descent example to illustrate one more aspect of the ODE. Going back to Eqn (4.4), we recognize that the recursion can be written equivalently as (assuming $\beta(t) = \beta$):

$$\underbrace{\mathbf{x}_i - \mathbf{x}_{i-1}}_{\Delta \mathbf{x}} = -\underbrace{\beta_{i-1}}_{\beta \Delta t} \nabla f(\mathbf{x}_{i-1}) \Rightarrow d\mathbf{x} = -\beta \nabla f(\mathbf{x}) dt, \quad (4.7)$$

where the continuous equation holds when we set $\Delta t \rightarrow 0$ and $\Delta \mathbf{x} \rightarrow 0$. The interesting point about this equality is that it gives us a summary of the update $\Delta \mathbf{x}$ by writing it in terms of dt . It says that if we move the along the time axis by dt , then the solution \mathbf{x} will be updated by $d\mathbf{x}$.

Eqn (4.7) defines the relationship between *changes*. If we consider a sequence of iterates $i = 1, 2, \dots, N$, and if we are told that the progression of the iterates follows Eqn (4.7), then we can write

$$\begin{aligned} \text{(forward)} \quad \mathbf{x}_i &= \mathbf{x}_{i-1} + \Delta \mathbf{x}_{i-1} \approx \mathbf{x}_{i-1} + d\mathbf{x} \\ &= \mathbf{x}_{i-1} - \nabla f(\mathbf{x}_{i-1})\beta dt \\ &\approx \mathbf{x}_{i-1} - \beta_{i-1} \nabla f(\mathbf{x}_{i-1}). \end{aligned}$$

We call this as the **forward** equation because we update \mathbf{x} by $\mathbf{x} + \Delta \mathbf{x}$ assuming that $t \leftarrow t + \Delta t$.

Now, consider a sequence of iterates $i = N, N-1, \dots, 2, 1$. If we are told that the progression of the iterates follows Eqn (4.7), then the time-reversal iterates will be

$$\begin{aligned} \text{(reverse)} \quad \mathbf{x}_{i-1} &= \mathbf{x}_i - \Delta \mathbf{x}_i \approx \mathbf{x}_i - d\mathbf{x} \\ &= \mathbf{x}_i + \beta \nabla f(\mathbf{x}_i)dt \\ &\approx \mathbf{x}_i + \beta_i \nabla f(\mathbf{x}_i). \end{aligned}$$

Note the change in sign when reversing the progression direction. We call this the **reverse** equation.

4.2 What is an SDE?

Stochastic differential equation (SDE) can be regarded as an extension to the ordinary differential equation (ODE). In this subsection, we briefly introduce the notations and explain their meanings.

ODE. We consider a first-order ODE of the following form:

$$\frac{d\mathbf{x}(t)}{dt} = \mathbf{f}(t, \mathbf{x}),$$

where $\mathbf{f}(t, \mathbf{x})$ is some function of t and \mathbf{x} . Assuming that the initial condition is $\mathbf{x}(0) = \mathbf{x}_0$, the solution takes the form of

$$\mathbf{x}(t) = \mathbf{x}_0 + \int_0^t \mathbf{f}(s, \mathbf{x}(s))ds.$$

The integral form of the solution often comes with a short-hand notation known as the **differential form**, which can be written as

$$d\mathbf{x} = \mathbf{f}(t, \mathbf{x})dt. \quad (4.8)$$

SDE. In an SDE, in addition to a deterministic function $\mathbf{f}(t, \mathbf{x})$, we consider a stochastic perturbation. For example, the stochastic perturbation can take the following form:

$$\frac{d\mathbf{x}(t)}{dt} = \mathbf{f}(t, \mathbf{x}) + \mathbf{g}(t, \mathbf{x})\xi(t), \quad \text{where } \xi(t) \sim \mathcal{N}(0, \mathbf{I}),$$

where $\xi(t)$ is a noise function, e.g., white noise. We can define $d\mathbf{w} = \xi(t)dt$, where $d\mathbf{w}$ is often known as the differential form of the Brownian motion. Then, the differential form of this SDE can be written as

$$d\mathbf{x} = \mathbf{f}(t, \mathbf{x})dt + \mathbf{g}(t, \mathbf{x})d\mathbf{w}. \quad (4.9)$$

Because $\xi(t)$ is random, the solution to this differential equation is also random. To be explicit about the randomness of the solution, we should interpret the differential form via the integral equation

$$\mathbf{x}(t, \omega) = \mathbf{x}_0 + \int_0^t \mathbf{f}(s, \mathbf{x}(s, \omega))ds + \int_0^t \mathbf{g}(s, \mathbf{x}(s, \omega))d\mathbf{w}(s, \omega),$$

where ω denotes the index of the state of \mathbf{x} . Therefore, as we pick a particular state of the random process $\mathbf{w}(s, \omega)$, we solve a differential equation corresponding to this particular ω .

Example 4.3. Consider the stochastic differential equation

$$d\mathbf{x} = ad\mathbf{w},$$

for some constants a . Based on our discussions above, the solution trajectory will take the form

$$\mathbf{x}(t) = \mathbf{x}_0 + \int_0^t ad\mathbf{w}(s) = \mathbf{x}_0 + a \int_0^t \boldsymbol{\xi}(s)ds,$$

where the last equality uses the fact that $d\mathbf{w} = \boldsymbol{\xi}(t)dt$. We can visualize the solution trajectory by numerically implementing $d\mathbf{x} = ad\mathbf{w}$ via the discrete-time iteration:

$$\mathbf{x}_i - \mathbf{x}_{i-1} = a \underbrace{(\mathbf{w}_i - \mathbf{w}_{i-1})}_{\stackrel{\text{def}}{=} \mathbf{z}_{i-1} \sim \mathcal{N}(0, \mathbf{I})} \Rightarrow \mathbf{x}_i = \mathbf{x}_{i-1} + a\mathbf{z}_{i-1}.$$

For example, if $a = 0.05$, one possible trajectory of $\mathbf{x}(t)$ will behave as shown below. The initial point $\mathbf{x}_0 = 0$ is marked as in red to indicate that the process is moving forward in time.

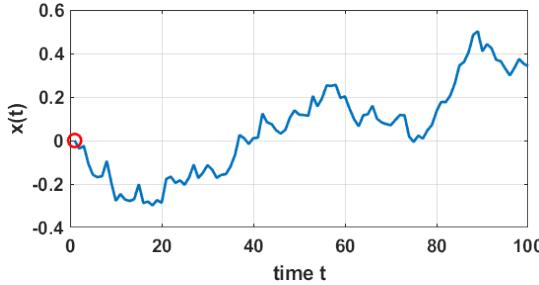


Figure 4.2: Trajectory of the process $d\mathbf{x} = ad\mathbf{w}$.

Example 4.4. Consider the stochastic differential equation

$$d\mathbf{x} = -\frac{\alpha}{2}\mathbf{x}dt + \beta d\mathbf{w}.$$

The solution to this problem is given by [Risken Eqn 3.7]

$$\mathbf{x}(t) = \mathbf{x}_0 e^{-\frac{\alpha}{2}t} + \beta \int_0^t e^{-\frac{\alpha}{2}(t-s)} \boldsymbol{\xi}(s)ds.$$

To visualize the trajectory, we consider $\alpha = 1$ and $\beta = 0.1$. The discretization will give us

$$\mathbf{x}_i - \mathbf{x}_{i-1} = -\frac{\alpha}{2}\mathbf{x}_{i-1} + \beta(\mathbf{w}_i - \mathbf{w}_{i-1}) \Rightarrow \mathbf{x}_i = \left(1 - \frac{\alpha}{2}\right)\mathbf{x}_{i-1} + \beta\mathbf{z}_{i-1}.$$

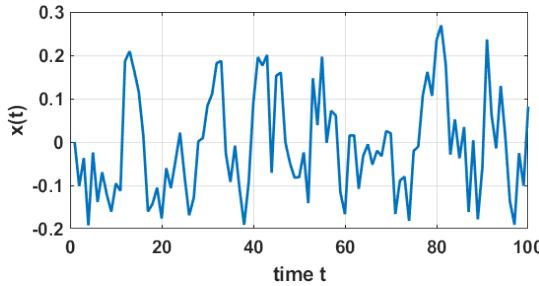


Figure 4.3: Trajectory of the process $d\mathbf{x} = -\frac{\alpha}{2}\mathbf{x}dt + \beta d\mathbf{w}$.

4.3 Stochastic Differential Equation for DDPM and SMLD

Without drilling into the physics of SDE (which we will do later), we want to connect SDE with the denoising diffusion probabilistic model (DDPM) and the score matching Langevin dynamics (SMLD) we studied in the previous sections.

Forward and Reverse Diffusion. Diffusion models involve a pair of equations: the forward diffusion process and the reverse diffusion process. Expressed in terms of derivatives, a forward diffusion process can be written as

$$\frac{d\mathbf{x}(t)}{dt} = \mathbf{f}(\mathbf{x}, t) + g(t)\xi(t), \quad \text{where } \xi(t) \sim \mathcal{N}(0, \mathbf{I}).$$

We emphasize that this is a particular diffusion process based on Brownian motion. The random perturbation ξ is assumed to be a random process with $\xi(t)$ being an i.i.d. Gaussian random variable at all t . Thus, the autocorrelation function is a delta function $\mathbb{E}[\xi(t)\xi(t')] = \delta(t - t')$. For this diffusion process, we can write it in terms of the differential:

Definition 4.1. Forward Diffusion.

$$d\mathbf{x} = \underbrace{\mathbf{f}(\mathbf{x}, t)}_{\text{drift}} dt + \underbrace{g(t)}_{\text{diffusion}} d\mathbf{w}. \quad (4.10)$$

Here, the differential is $d\mathbf{w} = \xi(t)dt$. This suggests that we can view $\xi(t)$ as some rate of change (over t) from which the integration of $\xi(t)dt$ will give us $d\mathbf{w}$.

The two terms $\mathbf{f}(\mathbf{x}, t)$ and $g(t)$ carry physical meanings. The drift coefficient is a vector-valued function $\mathbf{f}(\mathbf{x}, t)$ defining how molecules in a closed system would move in the absence of random effects. For the gradient descent algorithm, the drift is defined by the negative gradient of the objective function. That is, we want the solution trajectory to follow the gradient of the objective. The diffusion coefficient $g(t)$ is a scalar function describing how the molecules would randomly walk from one position to another. The function $g(t)$ determines how strong the random movement is.

The reverse direction of the diffusion equation is to move backward in time. The reverse-time SDE, according to Anderson [2], is given as follows.

Definition 4.2. Reverse Diffusion.

$$d\mathbf{x} = \underbrace{[\mathbf{f}(\mathbf{x}, t) - g(t)^2 \nabla_{\mathbf{x}} \log p_t(\mathbf{x})]}_{\text{drift}} dt + \underbrace{g(t)d\bar{\mathbf{w}}}_{\text{reverse-time diffusion}}, \quad (4.11)$$

where $p_t(\mathbf{x})$ is the probability distribution of \mathbf{x} at time t , and $\bar{\mathbf{w}}$ is the Wiener process when time flows backward.

Let's briefly talk about the reverse-time diffusion. The reverse-time diffusion is nothing but a random process that proceeds in the reverse time order. So while the forward diffusion defines $\mathbf{z}_i = \mathbf{w}_{i+1} - \mathbf{w}_i$, the reverse diffusion defines $\mathbf{z}_i = \mathbf{w}_{i-1} - \mathbf{w}_i$. The following is an example.

Example 4.5. Consider the reverse diffusion equation

$$d\mathbf{x} = a d\bar{\mathbf{w}}. \quad (4.12)$$

We can write the discrete-time recursion as follows. For $i = N, N-1, \dots, 1$, do

$$\mathbf{x}_{i-1} = \mathbf{x}_i + a \underbrace{(\mathbf{w}_{i-1} - \mathbf{w}_i)}_{=\mathbf{z}_i} = \mathbf{x}_i + a \mathbf{z}_i, \quad \mathbf{z}_i \sim \mathcal{N}(0, \mathbf{I}).$$

In the figure below we show the trajectory of this reverse-time process. Note that the initial point marked in red is at \mathbf{x}_N . The process is tracked backward to \mathbf{x}_0 .

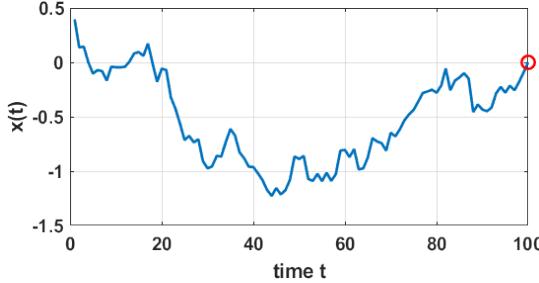


Figure 4.4: Trajectory of the reverse diffusion $d\mathbf{x} = ad\bar{\mathbf{w}}$.

Stochastic Differential Equation for DDPM. In order to draw the connection between DDPM and SDE, we consider the discrete-time DDPM forward iteration. For $i = 1, 2, \dots, N$:

$$\mathbf{x}_i = \sqrt{1 - \beta_i} \mathbf{x}_{i-1} + \sqrt{\beta_i} \mathbf{z}_{i-1}, \quad \mathbf{z}_{i-1} \sim \mathcal{N}(0, \mathbf{I}). \quad (4.13)$$

We can show that this equation can be derived from the forward SDE equation below.

Theorem 4.1. The forward sampling equation of **DDPM** can be written as an SDE via

$$d\mathbf{x} = \underbrace{-\frac{\beta(t)}{2} \mathbf{x}}_{=\mathbf{f}(\mathbf{x}, t)} dt + \underbrace{\sqrt{\beta(t)} d\mathbf{w}}_{=g(t)}. \quad (4.14)$$

Proof. We define a step size $\Delta t = \frac{1}{N}$, and consider an auxiliary noise level $\{\bar{\beta}_i\}_{i=1}^N$ where $\beta_i = \frac{\bar{\beta}_i}{N}$. Then

$$\beta_i = \underbrace{\beta\left(\frac{i}{N}\right)}_{\bar{\beta}_i} \cdot \frac{1}{N} = \beta(t + \Delta t) \Delta t,$$

where we assume that in the $N \rightarrow \infty$, $\bar{\beta}_i \rightarrow \beta(t)$ which is a continuous time function for $0 \leq t \leq 1$. Similarly, we define

$$\mathbf{x}_i = \mathbf{x}\left(\frac{i}{N}\right) = \mathbf{x}(t + \Delta t), \quad \mathbf{z}_i = \mathbf{z}\left(\frac{i}{N}\right) = \mathbf{z}(t + \Delta t).$$

Hence, we have

$$\begin{aligned} \mathbf{x}_i &= \sqrt{1 - \beta_i} \mathbf{x}_{i-1} + \sqrt{\beta_i} \mathbf{z}_{i-1} \\ \Rightarrow \quad \mathbf{x}_i &= \sqrt{1 - \frac{\bar{\beta}_i}{N}} \mathbf{x}_{i-1} + \sqrt{\frac{\bar{\beta}_i}{N}} \mathbf{z}_{i-1} \\ \Rightarrow \quad \mathbf{x}(t + \Delta t) &= \sqrt{1 - \beta(t + \Delta t) \cdot \Delta t} \mathbf{x}(t) + \sqrt{\beta(t + \Delta t) \cdot \Delta t} \mathbf{z}(t) \\ \Rightarrow \quad \mathbf{x}(t + \Delta t) &\approx \left(1 - \frac{1}{2}\beta(t + \Delta t) \cdot \Delta t\right) \mathbf{x}(t) + \sqrt{\beta(t + \Delta t) \cdot \Delta t} \mathbf{z}(t) \\ \Rightarrow \quad \mathbf{x}(t + \Delta t) &\approx \mathbf{x}(t) - \frac{1}{2}\beta(t)\Delta t \mathbf{x}(t) + \sqrt{\beta(t) \cdot \Delta t} \mathbf{z}(t). \end{aligned}$$

Thus, as $\Delta t \rightarrow 0$, we have

$$d\mathbf{x} = -\frac{1}{2}\beta(t)\mathbf{x}dt + \sqrt{\beta(t)} d\mathbf{w}. \quad (4.15)$$

Therefore, we showed that the DDPM forward update iteration can be equivalently written as an SDE.

Being able to write the DDPM forward update iteration as an SDE means that the DDPM estimates can be determined by solving the SDE. In other words, for an appropriately defined SDE solver, we can

throw the SDE into the solver. The solution returned by an appropriately chosen solver will be the DDPM estimate. Of course, we are not required to use the SDE solver because the DDPM iteration itself is solving the SDE. It may not be the best SDE solver because the DDPM iteration is only a first order method. Nevertheless, if we are not interested in using the SDE solver, we can still use the DDPM iteration to obtain a solution. Here is an example.

Example 4.6. Consider the DDPM forward equation with $\beta_i = 0.05$ for all $i = 0, \dots, N - 1$. We initialize the sample \mathbf{x}_0 by drawing it from a Gaussian mixture such that

$$\mathbf{x}_0 \sim \sum_{k=1}^K \pi_k \mathcal{N}(\mathbf{x}_0 | \boldsymbol{\mu}_k, \sigma_k^2 \mathbf{I}),$$

where $\pi_1 = \pi_2 = 0.5$, $\sigma_1 = \sigma_2 = 1$, $\boldsymbol{\mu}_1 = 3$ and $\boldsymbol{\mu}_2 = -3$. Then, using the equation

$$\mathbf{x}_i = \sqrt{1 - \beta_i} \mathbf{x}_{i-1} + \sqrt{\beta_i} \mathbf{z}_{i-1}, \quad \mathbf{z}_{i-1} \sim \mathcal{N}(0, \mathbf{I}),$$

we can plot the trajectory and the distribution as follows.

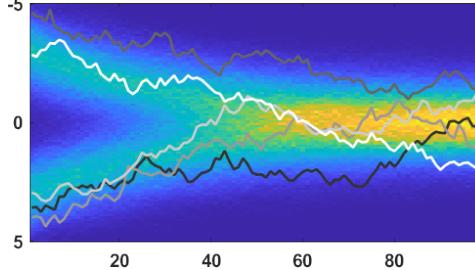


Figure 4.5: Realizations of the trajectories of \mathbf{x}_t , starting with a Gaussian mixture and ending with a single Gaussian.

The reverse diffusion equation follows from Eqn (4.11) by substituting the appropriate quantities: $\mathbf{f}(\mathbf{x}, t) = -\frac{\beta(t)}{2}$ and $g(t) = \sqrt{\beta(t)}$. This will give us

$$\begin{aligned} d\mathbf{x} &= [\mathbf{f}(\mathbf{x}, t) - g(t)^2 \nabla_{\mathbf{x}} \log p_t(\mathbf{x})] dt + g(t) d\bar{\mathbf{w}} \\ &= \left[-\frac{\beta(t)}{2} \mathbf{x} - \beta(t) \nabla_{\mathbf{x}} \log p_t(\mathbf{x}) \right] dt + \sqrt{\beta(t)} d\bar{\mathbf{w}}, \end{aligned}$$

which will give us the following equation:

Theorem 4.2. The reverse sampling equation of **DDPM** can be written as an SDE via

$$d\mathbf{x} = -\beta(t) \left[\frac{\mathbf{x}}{2} + \nabla_{\mathbf{x}} \log p_t(\mathbf{x}) \right] dt + \sqrt{\beta(t)} d\bar{\mathbf{w}}. \quad (4.16)$$

Proof. The iterative update scheme can be written by considering $d\mathbf{x} = \mathbf{x}(t) - \mathbf{x}(t - \Delta t)$, and $d\bar{\mathbf{w}} = \mathbf{w}(t - \Delta t) - \mathbf{w}(t) = -\mathbf{z}(t)$. Then, letting $dt = \Delta t$, we can show that

$$\begin{aligned} \mathbf{x}(t) - \mathbf{x}(t - \Delta t) &= -\beta(t) \Delta t \left[\frac{\mathbf{x}(t)}{2} + \nabla_{\mathbf{x}} \log p_t(\mathbf{x}(t)) \right] - \sqrt{\beta(t) \Delta t} \mathbf{z}(t) \\ \Rightarrow \quad \mathbf{x}(t - \Delta t) &= \mathbf{x}(t) + \beta(t) \Delta t \left[\frac{\mathbf{x}(t)}{2} + \nabla_{\mathbf{x}} \log p_t(\mathbf{x}(t)) \right] + \sqrt{\beta(t) \Delta t} \mathbf{z}(t). \end{aligned}$$

By grouping the terms, and assuming that $\beta(t)\Delta t \ll 1$, we recognize that

$$\begin{aligned}\mathbf{x}(t - \Delta t) &= \mathbf{x}(t) \left[1 + \frac{\beta(t)\Delta t}{2} \right] + \beta(t)\Delta t \nabla_{\mathbf{x}} \log p_t(\mathbf{x}(t)) + \sqrt{\beta(t)\Delta t} \mathbf{z}(t) \\ &\approx \mathbf{x}(t) \left[1 + \frac{\beta(t)\Delta t}{2} \right] + \beta(t)\Delta t \nabla_{\mathbf{x}} \log p_t(\mathbf{x}(t)) + \frac{(\beta(t)\Delta t)^2}{2} \nabla_{\mathbf{x}} \log p_t(\mathbf{x}(t)) + \sqrt{\beta(t)\Delta t} \mathbf{z}(t) \\ &= \left[1 + \frac{\beta(t)\Delta t}{2} \right] (\mathbf{x}(t) + \beta(t)\Delta t \nabla_{\mathbf{x}} \log p_t(\mathbf{x}(t))) + \sqrt{\beta(t)\Delta t} \mathbf{z}(t).\end{aligned}$$

Then, following the discretization scheme by letting $t \in \{0, \dots, \frac{N-1}{N}\}$, $\Delta t = 1/N$, $\mathbf{x}(t - \Delta t) = \mathbf{x}_{i-1}$, $\mathbf{x}(t) = \mathbf{x}_i$, and $\beta(t)\Delta t = \beta_i$, we can show that

$$\begin{aligned}\mathbf{x}_{i-1} &= (1 + \frac{\beta_i}{2}) \left[\mathbf{x}_i + \frac{\beta_i}{2} \nabla_{\mathbf{x}} \log p_i(\mathbf{x}_i) \right] + \sqrt{\beta_i} \mathbf{z}_i \\ &\approx \frac{1}{\sqrt{1-\beta_i}} \left[\mathbf{x}_i + \frac{\beta_i}{2} \nabla_{\mathbf{x}} \log p_i(\mathbf{x}_i) \right] + \sqrt{\beta_i} \mathbf{z}_i,\end{aligned}\tag{4.17}$$

where $p_i(\mathbf{x})$ is the probability density function of \mathbf{x} at time i . For practical implementation, we can replace $\nabla_{\mathbf{x}} \log p_i(\mathbf{x}_i)$ by the estimated score function $\mathbf{s}_{\theta}(\mathbf{x}_i)$.

So, we have recovered the DDPM iteration that is consistent with the one defined by Song and Ermon in [42]. This is an interesting result, because it allows us to connect DDPM's iteration using the score function. Song and Ermon [42] called the SDE an **variance preserving** (VP) SDE.

Example 4.7. Following from the previous example, we perform the reverse diffusion equation using

$$\mathbf{x}_{i-1} = \frac{1}{\sqrt{1-\beta_i}} \left[\mathbf{x}_i + \frac{\beta_i}{2} \nabla_{\mathbf{x}} \log p_i(\mathbf{x}_i) \right] + \sqrt{\beta_i} \mathbf{z}_i,$$

where $\mathbf{z}_i \sim \mathcal{N}(0, \mathbf{I})$. The trajectory of the iterates is shown below.

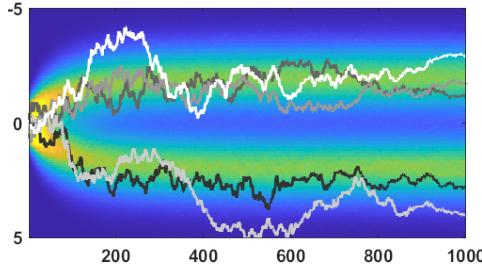


Figure 4.6: Realizations of the trajectories of \mathbf{x}_t , starting with a single Gaussian and ending with a Gaussian mixture.

Stochastic Differential Equation for SMLD. The score-matching Langevin Dynamics model can also be described by an SDE. To start with, we notice that in the SMLD setting, there isn't really a "forward diffusion step". However, we can roughly argue that if we divide the noise scale in the SMLD training into N levels, then the recursion should follow a Markov chain

$$\mathbf{x}_i = \mathbf{x}_{i-1} + \sqrt{\sigma_i^2 - \sigma_{i-1}^2} \mathbf{z}_{i-1}, \quad i = 1, 2, \dots, N.\tag{4.18}$$

This is not too hard to see. If we assume that the variance of \mathbf{x}_{i-1} is σ_{i-1}^2 , then we can show that

$$\begin{aligned}\text{Var}[\mathbf{x}_i] &= \text{Var}[\mathbf{x}_{i-1}] + (\sigma_i^2 - \sigma_{i-1}^2) \\ &= \sigma_{i-1}^2 + (\sigma_i^2 - \sigma_{i-1}^2) = \sigma_i^2.\end{aligned}$$

Therefore, given a sequence of noise levels, Eqn (4.18) will indeed generate estimates \mathbf{x}_i such that the noise statistics will satisfy the desired property.

If we agree Eqn (4.18), it is easy to derive the SDE associated with Eqn (4.18). We summarize our results as follows.

Theorem 4.3. The forward sampling equation of **SMLD** can be written as an SDE via

$$d\mathbf{x} = \sqrt{\frac{d[\sigma(t)^2]}{dt}} d\mathbf{w}. \quad (4.19)$$

Proof. Assuming that in the limit $\{\sigma_i\}_{i=1}^N$ becomes the continuous time $\sigma(t)$ for $0 \leq t \leq 1$, and $\{\mathbf{x}_i\}_{i=1}^N$ becomes $\mathbf{x}(t)$ where $\mathbf{x}_i = \mathbf{x}(\frac{i}{N})$ if we let $t \in \{0, \frac{1}{N}, \dots, \frac{N-1}{N}\}$. Then we have

$$\begin{aligned} \mathbf{x}(t + \Delta t) &= \mathbf{x}(t) + \sqrt{\sigma(t + \Delta t)^2 - \sigma(t)^2} \mathbf{z}(t) \\ &\approx \mathbf{x}(t) + \sqrt{\frac{d[\sigma(t)^2]}{dt} \Delta t} \mathbf{z}(t). \end{aligned}$$

At the limit when $\Delta t \rightarrow 0$, the equation converges to

$$d\mathbf{x} = \sqrt{\frac{d[\sigma(t)^2]}{dt}} d\mathbf{w}.$$

Mapping this to Eqn (4.11), we derive the reverse-time diffusion equation.

Theorem 4.4. The reverse sampling equation of **SMLD** can be written as an SDE via

$$d\mathbf{x} = -\left(\frac{d[\sigma(t)^2]}{dt} \nabla_{\mathbf{x}} \log p_t(\mathbf{x})\right) dt + \sqrt{\frac{d[\sigma(t)^2]}{dt}} d\bar{\mathbf{w}}. \quad (4.20)$$

Proof. We recognize that

$$\mathbf{f}(\mathbf{x}, t) = 0, \quad \text{and} \quad g(t) = \sqrt{\frac{d[\sigma(t)^2]}{dt}}.$$

As a result, if we write the reverse equation Eqn (4.11), we should have

$$\begin{aligned} d\mathbf{x} &= [\mathbf{f}(\mathbf{x}, t) - g(t)^2 \nabla_{\mathbf{x}} \log p_t(\mathbf{x})] dt + g(t) d\bar{\mathbf{w}} \\ &= -\left(\frac{d[\sigma(t)^2]}{dt} \nabla_{\mathbf{x}} \log p_t(\mathbf{x}(t))\right) dt + \sqrt{\frac{d[\sigma(t)^2]}{dt}} d\bar{\mathbf{w}}. \end{aligned}$$

For the discrete-time iterations, we first define $\alpha(t) = \frac{d[\sigma(t)^2]}{dt}$. Then, using the same set of discretization setups as the DDPM case, we can show that

$$\begin{aligned} \mathbf{x}(t + \Delta t) - \mathbf{x}(t) &= -\left(\alpha(t) \nabla_{\mathbf{x}} \log p_t(\mathbf{x})\right) \Delta t - \sqrt{\alpha(t) \Delta t} \mathbf{z}(t) \\ \Rightarrow \quad \mathbf{x}(t) &= \mathbf{x}(t + \Delta t) + \alpha(t) \Delta t \nabla_{\mathbf{x}} \log p_t(\mathbf{x}) + \sqrt{\alpha(t) \Delta t} \mathbf{z}(t) \\ \Rightarrow \quad \mathbf{x}_{i-1} &= \mathbf{x}_i + \alpha_i \nabla_{\mathbf{x}} \log p_i(\mathbf{x}_i) + \sqrt{\alpha_i} \mathbf{z}_i \\ \Rightarrow \quad \mathbf{x}_{i-1} &= \mathbf{x}_i + (\sigma_i^2 - \sigma_{i-1}^2) \nabla_{\mathbf{x}} \log p_i(\mathbf{x}_i) + \sqrt{(\sigma_i^2 - \sigma_{i-1}^2)} \mathbf{z}_i, \end{aligned} \quad (4.21)$$

which is identical to the SMLD reverse update equation. Song and Ermon [42] called the SDE an **variance exploding (VE)** SDE.

Equivalence between VP and VE's Inference. As we have just seen, DDPM and SMLD correspond to two variants of the stochastic differential equation: the variance exploding (VE) and the variance preserving (VP). An observation made by Kawar et al. [21] was that the *inference* process determined by VP and VE are equivalent. Therefore, if we want to use a diffusion model as part of another task such as image restoration, it does not matter if we use VE or VP. Note, however, that the specific choice of hyperparameters will still affect the *training* and hence the model itself.

4.4 Numerical Solvers for ODE and SDE

SDE and ODE are not easy to solve analytically. In many situations, these differential equations are solved numerically. In this subsection, we briefly highlight the overall idea of these methods.

To make our discussion slightly easier, we shall focus on a simple ODE:

$$\frac{d\mathbf{x}(t)}{dt} = \mathbf{f}(t, \mathbf{x}(t)). \quad (4.22)$$

Geometrically, this ODE means that the *derivative* of $\mathbf{x}(t)$ equals to the function $\mathbf{f}(t, \mathbf{x})$ evaluated at $\mathbf{x}(t)$. For this to happen, we need the slope of $\mathbf{x}(t)$ to match with the functional value. In the 1D case, the above geometric interpretation can be translated to the following equation:

$$\frac{x(t) - x(t_0)}{t - t_0} = f(t_0, x_0),$$

where t_0 is a time fairly close to t . By rearranging the terms, we see that this equation is equivalent to

$$x(t) = x(t_0) + f(t_0, x_0)(t - t_0).$$

So we have recovered something very similar to gradient descent. This is the Euler method.

Euler Method. The Euler method is a first-order numerical method for solving the ODE. Given $\frac{dx(t)}{dt} = f(t, x)$, and $x(t_0) = x_0$, Euler method solves the problem via an iterative scheme for $i = 0, 1, \dots, N-1$ such that

$$x_{i+1} = x_i + \alpha \cdot f(t_i, x_i), \quad i = 0, 1, \dots, N-1,$$

where α is the step size. Let's consider a simple example.

Example 4.8. [3, Example 2.2] Consider the following ODE

$$\frac{dx(t)}{dt} = \frac{x(t) + t^2 - 2}{t + 1}.$$

If we apply the Euler method with a step size α , then the iteration will take the form

$$x_{i+1} = x_i + \alpha \cdot f(t_i, x_i) = x_i + \alpha \cdot \frac{(x_i + t_i^2 - 2)}{t_i + 1}.$$

Runge-Kutta (RK) Method. Another popularly used ODE solver is the Runge-Kutta (RK) method. The classical RK-4 algorithm solves the ODE via the iteration

$$x_{i+1} = x_i + \frac{\alpha}{6} \cdot \left(k_1 + 2k_2 + 2k_3 + k_4 \right), \quad i = 1, 2, \dots, N,$$

where the quantities k_1, k_2, k_3 and k_4 are defined as

$$\begin{aligned} k_1 &= f(x_i, t_i), \\ k_2 &= f\left(t_i + \frac{\alpha}{2}, x_i + \alpha \frac{k_1}{2}\right), \\ k_3 &= f\left(t_i + \frac{\alpha}{2}, x_i + \alpha \frac{k_2}{2}\right), \\ k_4 &= f(t_i + \alpha, x_i + \alpha k_3). \end{aligned}$$

For more details, you can consult numerical methods textbooks such as [3].

Predictor-Corrector Algorithm [42]. Since different numerical solvers have different behavior in terms of the error of approximation, throwing the ODE (or SDE) into an off-the-shelf numerical solver will result in various degrees of error [20]. However, if we are specifically trying to solve the reverse diffusion equation, it is possible to use techniques other than numerical ODE/SDE solvers to make the appropriate corrections, as illustrated in Figure 4.7.

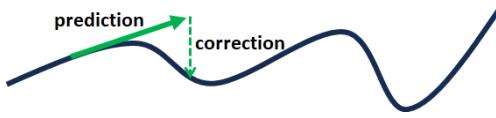


Figure 4.7: Prediction and correction algorithm.

Let's use DDPM as an example. In DDPM, the reverse diffusion equation is given by

$$\mathbf{x}_{i-1} = \frac{1}{\sqrt{1-\beta_i}} \left[\mathbf{x}_i + \frac{\beta_i}{2} \nabla_{\mathbf{x}} \log p_i(\mathbf{x}_i) \right] + \sqrt{\beta_i} \mathbf{z}_i.$$

We can consider it as an Euler method for the reverse diffusion. However, if we have already trained the score function $\mathbf{s}_{\theta}(\mathbf{x}_i, i)$, we can run the score-matching equation, i.e.,

$$\mathbf{x}_{i-1} = \mathbf{x}_i + \epsilon_i \mathbf{s}_{\theta}(\mathbf{x}_i, i) + \sqrt{2\epsilon_i} \mathbf{z}_i,$$

for M times to make the correction. The algorithm below summarizes the idea. (Note that we have replaced the score function by the estimate.)

Algorithm 1 Prediction Correction Algorithm for DDPM. [42]

```

 $\mathbf{x}_N = \mathcal{N}(0, \mathbf{I}).$ 
for  $i = N - 1, \dots, 0$  do
```

$$(\text{Prediction}) \quad \mathbf{x}_{i-1} = \frac{1}{\sqrt{1-\beta_i}} \left[\mathbf{x}_i + \frac{\beta_i}{2} \mathbf{s}_{\theta}(\mathbf{x}_i, i) \right] + \sqrt{\beta_i} \mathbf{z}_i. \quad (4.23)$$

```
    for  $m = 1, \dots, M$  do
```

$$(\text{Correction}) \quad \mathbf{x}_{i-1} = \mathbf{x}_i + \epsilon_i \mathbf{s}_{\theta}(\mathbf{x}_i, i) + \sqrt{2\epsilon_i} \mathbf{z}_i, \quad (4.24)$$

```
    end for
```

```
end for
```

For the SMLD algorithm, the two equations are:

$$\begin{aligned} \mathbf{x}_{i-1} &= \mathbf{x}_i + (\sigma_i^2 - \sigma_{i-1}^2) \mathbf{s}_{\theta}(\mathbf{x}_i, \sigma_i) + \sqrt{\sigma_i^2 - \sigma_{i-1}^2} \mathbf{z} && \text{Prediction,} \\ \mathbf{x}_{i-1} &= \mathbf{x}_i + \epsilon_i \nabla_{\mathbf{x}} \mathbf{s}_{\theta}(\mathbf{x}_i, \sigma_i) + \sqrt{\epsilon_i} \mathbf{z} && \text{Correction.} \end{aligned}$$

We can pair them up as in the case of DDPM's prediction-correction algorithm by repeating the correction iteration a few times.

4.5 Concluding Remark

The connection between the iterative algorithms with stochastic differential equations is not only a discovery that has its own scientific value, the finding has practical utilities as commented by some papers. By unifying multiple diffusion models to the same SDE framework, one can compare the algorithms. In some cases, one can improve the numerical scheme by borrowing ideas from the SDE literature as well as the probabilistic sampling literature. For example, the predictor-corrector scheme in [42] was a hybrid SDE solver coupled

with a Markov Chain Monte Carlo. Mapping the diffusion iterations to SDE, according to some papers such as [1], offers more design flexibility. Outside the context diffusion algorithms, in general stochastic gradient descent algorithms have corresponding SDE such as the Fokker-Planck equations. People have demonstrated how to theoretically analyze the limiting distribution of the estimates, in exact closed-form. This alleviates the difficulty of analyzing the random algorithm by means of analyzing a well-defined limiting distribution.

There is a growing interest in developing accelerated SDE solvers. Lu et al. [26] showed that the typical SDE we encounter in diffusion models can be decomposed into a semi-linear term and a nonlinear term. Since the semi-linear term has a closed-form solution, one just needs to use a specialized numerical solver to handle the nonlinear term. It was further showed that DDIM can be considered as an accelerated SDE solver. Other approaches to accelerate the SDE solver is by means of knowledge distillation [36, 28].

5 Langevin and Fokker-Planck Equations

Many recent papers on diffusion models are focusing on either improving the image generation quality (by generalizing it to more semantically meaningful images) or applying it to problems in new domains (e.g. medical data). To help beginner readers understand better the foundations of these applications, we want to go backward in time by discussing the *physics* of the stochastic differential equations associated with the diffusion models. By studying the fundamental principles of these SDEs, we want to gain a deeper understanding of where these equations are coming from.

More specifically, we are interested in studying two major sets of equations related to diffusion: the Langevin equation and the Fokker-Planck equation. There are several goals of this section. First, we want to discuss the origin of the Langevin equation, and explain why it is related to diffusion models. We want to explain the general properties of a Markovian process, and its associated differential equations. Finally, we want to show how the Fokker-Planck equation is derived, and discuss why it plays an important role in diffusion models.

5.1 Brownian Motion

Historical Perspective. In 1827, botanist Robert Brown observed a phenomenon that small pollen grains have irregular motion when placed in water [6]. The motion of the pollen grains is later named the *Brownian motion*. The explanation of the Brownian motion was made by Albert Einstein in 1905 [14] and independently by Marian Smoluchowski [44] around the same time. Einstein's main argument was that the motion of the pollen grains is caused by the impact of the water molecules. However, since there are trillions of molecules in the system and we never know the initial state of individual molecules, it is nearly impossible to use classical analysis to study the microscope states. Einstein showed that the mean squared displacement can be related to a diffusion coefficient, and so he introduced a probabilistic approach by considering the statistical behavior of the molecules⁴. Einstein's theoretical prediction was then empirically confirmed by Jean-Baptiste Perrin who later won the Nobel Prize for Physics in 1926. A few years later since Einstein's paper, in 1908, French physicist Paul Langevin constructed a random Markovian force to describe the collision and interactions of the particles. This then led to the development of a partial differential equation by Dutch physicist Adriaan Fokker in 1914 and German physicist Max Planck in 1917, which is now known as the Fokker-Planck equation. The Kramers-Moyal expansion was introduced by Hans Kramers in 1940 and José Enrique Moyal in 1949, which showed a Taylor expansion technique to describe the time evolution of a probability distribution.

Derivation of Brownian Motion. So, what is Brownian motion and how is it related to diffusion models? Assume that there is a particle suspended in fluid. Stoke's law states that the friction applied to the particle is given by

$$F(t) = -\alpha v(t), \quad (5.1)$$

where F is the friction, v is the velocity, and $\alpha = 6\pi\mu R$. Here, R is the radius of the particle, and μ is the viscosity of the fluid. By Newton's second law, we further know that $F(t) = m\dot{v}(t)$, where m is the mass of the particle. Equating the two equations

$$\begin{cases} F(t) = -\alpha v(t) \\ F(t) = m\frac{dv(t)}{dt}, \end{cases}$$

we will obtain the following differential equation

$$m\frac{dv(t)}{dt} + \alpha v(t) = 0.$$

By defining $\gamma \stackrel{\text{def}}{=} \frac{\alpha}{m}$, we can simplify the above equation as

$$\frac{dv(t)}{dt} + \gamma v(t) = 0. \quad (5.2)$$

⁴A remark here is that Andrey Kolmogorov's probability book wasn't published until 1933 [25]. So back in 1905, the notion of probability theory was in its infancy.

The above deterministic equation is accurate when the particle is significantly more massive than the fluid molecules. The reason is that, by conservation of momentum, the massive particle will gain little velocity when it collides with the fluid molecule. However, since pollen grains are so light, the bombardment of water molecules will cause them to accelerate in a small but non-negligible way. This will create a random fluctuation. The total force of the water molecule acting on the particle is then modified to

$$F(t) = -\alpha v(t) + F_f(t),$$

where $F_f(t)$ is a stochastic term. This will then give us a modified differential equation

$$m \frac{dv(t)}{dt} = -\alpha v(t) + F_f(t).$$

By defining $\Gamma(t) = F_f(t)/m$, we arrive at a new differential equation

$$\frac{dv(t)}{dt} + \gamma v(t) = \Gamma(t), \quad (5.3)$$

which can be written in terms of a short-hand notation

$$\dot{v} + \gamma v = \Gamma(t). \quad (5.4)$$

In Eqn (5.4), the random process $\Gamma(t)$ represents a stochastic force known as the **Langevin force**. It satisfies two properties:

- (i) $\mathbb{E}[\Gamma(t)] = 0$ for all t so that its mean function is a constant zero;
- (ii) $\mathbb{E}[\Gamma(t)\Gamma(t')] = q\delta(t-t')$ for all t and t' , i.e., the autocorrelation function is a delta function with an amplitude q .

Remark. Properties (i) and (ii) are special cases of a wide sense stationary process. A wide sense stationary process is a random process that has a constant mean function (the constant is not necessarily zero), and that the autocorrelation function $R(t, t') \stackrel{\text{def}}{=} \mathbb{E}[\Gamma(t)\Gamma(t')]$ is a function of the difference $t - t'$. This function is not necessarily a delta function. For example $R(t, t') = e^{-|t-t'|}$ can be a valid autocorrelation function of a wide sense stationary process.

A random process satisfying properties (i) and (ii) are sometime called a delta-correlated process in the statistical mechanics literature. There are different ways to construct a delta-correlated process. For example, we can assume $\Gamma(t) \sim \mathcal{N}(0, 1)$ for every t , or any other independent and identically distributions defined in the same way. Gaussian distributions are more often used because many physical phenomena can be described by a Gaussian, e.g., thermal noise. A Gaussian random process satisfying properties (i) and (ii) is called a Gaussian white noise.

For any wide sense stationary process, **Wiener-Khinchin Theorem** says that the *power spectral density* can be defined through the Fourier transform of the autocorrelation function. More specifically, if $R(\tau) = \mathbb{E}[\Gamma(t+\tau)\Gamma(t)]$ is the autocorrelation function (we can write $R(t, t')$ as $R(\tau)$ if $\Gamma(t)$ is a wide sense stationary process), Wiener-Khinchin Theorem states that the power spectral density is

$$S(\omega) = \int_{-\infty}^{\infty} R(\tau) e^{-j\omega\tau} d\tau. \quad (5.5)$$

So if $R(\tau)$ is a delta function, $S(\omega)$ will have a constant value for all ω .

Remark. The name “**Gaussian white noise**” comes from the fact that the power spectral density $S(\omega)$ is uniform for every frequency ω (so it contains all the colors in the visible spectrum). A white noise is defined as $\Gamma(t) \sim \mathcal{N}(0, \sigma^2)$ for all t . It is easy to show that such a $\Gamma(t)$ would satisfy the above two criteria.

Firstly, $\mathbb{E}[\Gamma(t)]$ is $\mathbb{E}[\Gamma(t)] = 0$ by construction (since $\Gamma(t) \sim \mathcal{N}(0, \sigma^2)$). Secondly, if $\Gamma(t) \sim \mathcal{N}(0, \sigma^2)$, it is necessary that $R(\tau) = \mathbb{E}[\Gamma(t+\tau)\Gamma(t)]$ is a delta function. Wiener-Khinchin Theorem then states that the power spectral density is flat because it is the Fourier transform of a delta function. The

figures below shows the random realizations of a white noise, and their autocorrelation functions.

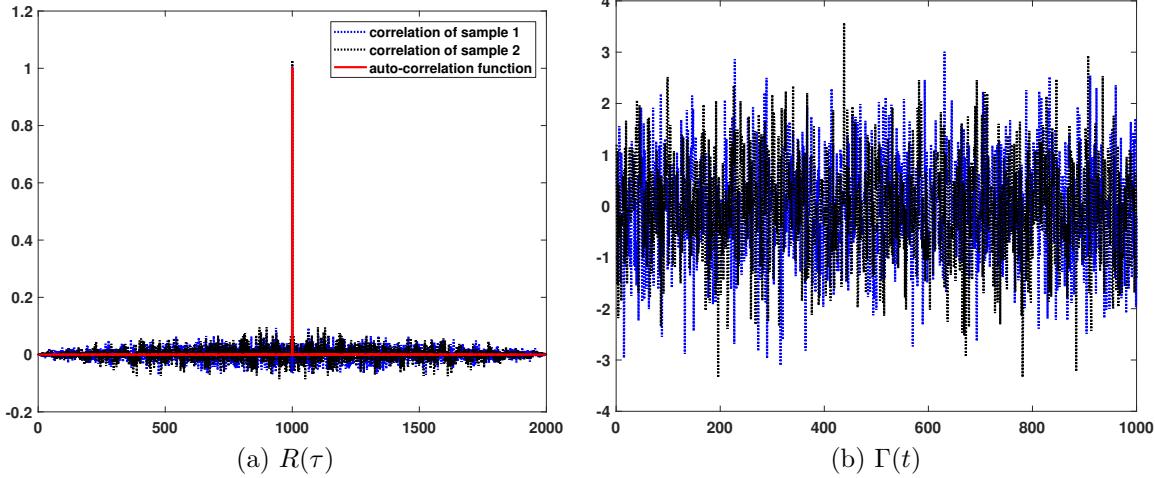


Figure 5.1: (a) Autocorrelation function $R(\tau)$ of a white Gaussian noise. (b) The random realization of the random process $\Gamma(t)$.

From Physics to Generative AI. Because of the randomness exhibited in $\Gamma(t)$, the differential equation given by Eqn (5.4) is a stochastic differential equation (SDE). The solution to this SDE is therefore a random process where the value $v(t)$ is a random variable at any time t . Brownian motion refers to the trajectory of this random process $v(t)$ as a function of time. The resulting SDE in Eqn (5.4) is a special case of the *Langevin equation*. We call it a linear Langevin equation with a δ -correlated Langevin force:

Definition 5.1. A **linear Langevin equation** with a δ -correlated Langevin force is a stochastic differential equation of the form

$$\dot{\xi} + \gamma\xi = \Gamma(t), \quad (5.6)$$

where $\Gamma(t)$ is a random process satisfying two properties that (i) $\mathbb{E}[\Gamma(t)] = 0$ for all t and (ii) $\mathbb{E}[\Gamma(t)\Gamma(t')] = q\delta(t - t')$ for all t and t' .

At this point we can connect the Langevin equation in Eqn (5.6) with a diffusion model, e.g., DDPM.

Example 5.1. Forward DDPM. Recall that a DDPM forward diffusion equation is given by

$$d\mathbf{x} = \underbrace{-\frac{\beta(t)}{2}\mathbf{x}}_{=f(t)} dt + \underbrace{\sqrt{\beta(t)}d\mathbf{w}}_{=g(t)}.$$

Expressing it in the Langevin equation form, we can write it as

$$\dot{\xi}(t) + f(t)\xi(t) = g(t)\Gamma(t), \quad \text{where } \Gamma(t) \sim \mathcal{N}(0, \mathbf{I}).$$

Example 5.2. Reverse DDPM. The reverse DDPM diffusion is given by Eqn (4.16)

$$d\mathbf{x} = \underbrace{-\beta(t) \left[\frac{\mathbf{x}}{2} + \nabla_{\mathbf{x}} \log p_t(\mathbf{x}) \right]}_{=\mathbf{f}(\xi, t)} dt + \underbrace{\sqrt{\beta(t)}d\mathbf{w}}_{=g(t)}.$$

Expressing it in the Langevin equation form, we can write it as

$$\dot{\xi}(t) = \mathbf{f}(\xi, t) + g(t)\Gamma(t), \quad \text{where } \Gamma(t) \sim \mathcal{N}(0, \mathbf{I}).$$

We can continue these examples for other diffusion models such as SMLD. We leave these as exercises for the readers. Our bottom message is that the diffusion equations we see in the previous chapters can all be formulated through the Langevin equation. Therefore, if we want to know the probability distributions of what these diffusion equations produce, we should look for tools in the literature of Langevin equations.

Solution to (Linear) Langevin Equation. The linear Langevin equation presented in Eqn (5.6) is a simple one. It is possible to analytically derive the solution $\xi(t)$ at any time t .

We start by considering the simpler problem where $\Gamma(t) = 0$. In this case, the differential equation is

$$\dot{\xi}(t) + \gamma\xi(t) = 0,$$

and it is called a first-order homogeneous differential equation. The solution of this differential equation is as follows.

Theorem 5.1. Consider the following differential equation

$$\dot{\xi}(t) + \gamma\xi(t) = 0,$$

with an initial condition $\xi(0) = \xi_0$. The solution is given by

$$\xi(t) = \xi_0 e^{-\gamma t}. \quad (5.7)$$

Proof. By rearranging the terms, we can show that

$$\frac{\dot{\xi}(t)}{\xi(t)} = -\gamma,$$

where we assume that $\xi(t) \neq 0$ for all t so that we can take $1/\xi(t)$. Integrating both sides will give us

$$\int_0^t \frac{\dot{\xi}(t')}{\xi(t')} dt' = - \int_0^t \gamma dt'.$$

The left hand side of the equation will give us $\log \xi(t) - \log \xi(0)$ where as the right hand side will give us $-\gamma t$. Equating them will give us

$$\log \xi(t) - \log \xi(0) = -\gamma t \implies \xi(t) = \xi_0 e^{-\gamma t}.$$

Now let's consider the case where $\Gamma(t)$ is present. The differential equation becomes

$$\dot{\xi} + \gamma\xi = \Gamma(t),$$

which is called a first-order non-homogeneous differential equation. To solve this differential equation, we employ a technique known as the *variation of parameter* or *variation of constant* [29, Theorem 1.2.3]. The idea can be summarized in two steps. We know from our previous derivation that the solution to a homogeneous equation is $\xi(t) = \xi_0 e^{-\gamma t}$. So let's make an educated guess about the solution of the non-homogenous case that the solution takes the form of $s(t) = A(t)e^{-\gamma t}$ for some $A(t)$. For notation simplicity we define $h(t) = e^{-\gamma t}$. If $s(t)$ is indeed the solution to the differential equation, then we can evaluate $\dot{s}(t) + \gamma s(t) = \Gamma(t)$. The left hand side of this equation is

$$\begin{aligned} \dot{s}(t) + \gamma s(t) &= [A(t)h(t)]' + \gamma[A(t)h(t)] \\ &= A'(t)h(t) + A(t)h'(t) + \gamma A(t)h(t) \\ &= A(t)[h'(t) + \gamma h(t)] + A'(t)h(t) \\ &= A'(t)h(t), \end{aligned}$$

where the last equality follows from the fact that $h(t) = e^{-\gamma t}$ is a solution to the homogeneous equation, hence $h'(t) + \gamma h(t) = 0$. Therefore, for $\dot{s}(t) + \gamma s(t) = \Gamma(t)$, it is necessary for $A'(t)h(t) = \Gamma(t)$ by finding an appropriate $A'(t)$. But this is not difficult. The equation $A'(t)h(t) = \Gamma(t)$ can be written as

$$A'(t)e^{-\gamma t} = \Gamma(t) \quad \Rightarrow \quad A'(t) = e^{\gamma t}\Gamma(t).$$

Integrating both side will give us

$$A(t) = \int_0^t e^{\gamma t'}\Gamma(t') dt',$$

and since $s(t) = A(t)e^{-\gamma t}$, we can show that

$$s(t) = e^{-\gamma t} \int_0^t e^{\gamma t'}\Gamma(t') dt' = \int_0^t e^{-\gamma(t-t')}\Gamma(t') dt'.$$

Therefore, the complete solution (which is the sum of the homogeneous part and the non-homogeneous part) is

$$\xi(t) = \xi_0 e^{-\gamma t} + \int_0^t e^{-\gamma(t-t')}\Gamma(t') dt'.$$

We summarize the result as follows.

Theorem 5.2. Consider the following differential equation

$$\dot{\xi}(t) + \gamma\xi(t) = \Gamma(t),$$

with an initial condition $\xi(0) = \xi_0$. The solution is given by

$$\xi(t) = \xi_0 e^{-\gamma t} + \int_0^t e^{-\gamma(t-t')}\Gamma(t') dt'. \quad (5.8)$$

Distribution at Equilibrium. The previous result shows that the solution $\xi(t)$ is a function of a random process $\Gamma(t)$. Since we do not know the particular realization of $\Gamma(t)$ every time we run the (Brownian motion) experiment, it is often more useful to characterize $\xi(t)$ by looking at the probability distribution of $\xi(t)$. In what follows, we follow Risken [33] to analyze the probability distribution at the equilibrium where $t \rightarrow \infty$ and $\xi(t) \rightarrow x$.

Theorem 5.3. Consider the Langevin equation in Definition 5.1 where

$$\dot{\xi} + \gamma\xi = \Gamma(t), \quad (5.9)$$

and $\Gamma(t)$ is a white Gaussian noise so that it satisfies the properties aforementioned. Let $\xi(t) = x$ be the solution at equilibrium to this SDE, and let $p(x)$ be the probability distribution of x . It holds that

$$p(x) = \frac{1}{\sqrt{2\pi\sigma^2}} e^{-\frac{x^2}{2\sigma^2}}, \quad (5.10)$$

where $\sigma = \sqrt{\frac{q}{2\gamma}}$. In other words, the solution $\xi(t) = x$ at equilibrium is a zero-mean Gaussian random variable.

Proof. Let $\xi_0 = \xi(0)$ be the initial condition. Then, the solution of the SDE takes the form

$$\xi(t) = \xi_0 e^{-\gamma t} + \int_0^t e^{-\gamma(t-t')}\Gamma(t') dt'. \quad (5.11)$$

At equilibrium when $t \rightarrow \infty$, we can drop $\xi_0 e^{-\gamma t}$. Moreover, by letting $\tau = t - t'$, we can write the

solution as

$$\xi(t) = \int_0^\infty e^{-\gamma(t-t')} \Gamma(t') dt' = \int_0^\infty e^{-\gamma\tau} \Gamma(t-\tau) d\tau.$$

The probability density function $p(\xi)$ can be determined by taking the inverse Fourier transform of the characteristic function. Recall that the characteristic function of a random variable $v(t)$ is

$$C(u) = \mathbb{E}[\exp\{iu \cdot \xi(t)\}] = 1 + \sum_{n=1}^{\infty} \frac{(iu)^n}{n!} \mathbb{E}[\xi(t)^n].$$

So, to find $C(u)$ we need to determine the moments $\mathbb{E}[\xi(t)^n]$. Using a result in Risken (Chapter 3 Eqns 3.26 and 3.27), we can show that

$$\begin{aligned} \mathbb{E}[\xi(t)^{2n+1}] &= 0 \\ \mathbb{E}[\xi(t)^{2n}] &= \frac{(2n)!}{2^n n!} \left[\int_0^\infty \int_0^\infty e^{-\gamma(\tau_1+\tau_2)} q \delta(\tau_1 - \tau_2) d\tau_1 d\tau_2 \right]^n \end{aligned} \quad (5.12)$$

$$= \frac{(2n)!}{2^n n!} \left[q \int_0^\infty e^{-2\gamma\tau_2} d\tau_2 \right]^n = \frac{(2n)!}{2^n n!} \left[\frac{q}{2\gamma} \right]^n. \quad (5.13)$$

Substituting this result into the characteristic function will give us

$$\begin{aligned} C(u) &= 1 + 0 + \frac{2!}{2} \left[\frac{q}{2\gamma} \right] + 0 + \frac{(2 \cdot 2)!}{2^2 2!} \left[\frac{q}{2\gamma} \right]^2 + \dots \\ &= \sum_{n=0}^{\infty} \frac{(iu)^{2n} \mathbb{E}[\xi(t)^{2n}]}{(2n)!} \\ &= \sum_{n=0}^{\infty} \frac{(iu)^{2n}}{(2n)!} \cdot \frac{(2n)!}{2^n n!} \left[\frac{q}{2\gamma} \right]^n \\ &= \sum_{n=0}^{\infty} \frac{1}{n!} \left(-\frac{u^2 q}{4\gamma} \right)^n = e^{-\frac{u^2 q}{4\gamma}}. \end{aligned} \quad (5.14)$$

Recognizing that this is the characteristic function of a Gaussian, we can use inverse Fourier transform to retrieve the probability density function

$$p(x) = \sqrt{\frac{\gamma}{\pi q}} e^{-\frac{\gamma x^2}{q}}.$$

Example 5.3. (Forward DDPM Distribution at Equilibrium) Let's do a sanity check by applying our result to the forward DDPM equation, and see what probability distribution will we obtain at the equilibrium state.

For simplicity let's assume a constant learning rate for the DDPM equation

$$d\mathbf{x} = -\frac{\beta}{2} \mathbf{x} dt + \sqrt{\beta} d\mathbf{w}.$$

The associate Langevin equation is

$$\dot{\xi}(t) + \frac{\beta}{2} \xi(t) = \sqrt{\beta} \Gamma(t).$$

As $t \rightarrow \infty$, our theorem above suggests that

$$p(x) = \sqrt{\frac{\gamma}{\pi q}} e^{-\frac{\gamma x^2}{q}} = \frac{1}{\sqrt{2\pi}} e^{-\frac{x^2}{2}},$$

where we substituted $\gamma = \beta/2$, and $q = \sqrt{\beta^2} = \beta$. Therefore, the probability distribution of $\xi(t)$ when $t \rightarrow \infty$ is $\mathcal{N}(0, 1)$. This is consistent with what we expect.

Wiener Process. In the special case where $\gamma = 0$, the linear Langevin equation is simplified to

$$\dot{\xi} = \Gamma(t).$$

By letting $\gamma = 0$ in Eqn (5.11), we can show that

$$\xi(t) = \xi_0 + \int_0^t \Gamma(t') dt',$$

which is also known as the Wiener process. The probability distribution of the solution of the Wiener process can be derived as follows.

Theorem 5.4. Wiener Process. Consider the Wiener process

$$\dot{\xi} = \Gamma(t), \quad (5.15)$$

where $\Gamma(t)$ is the Gaussian white noise with $\mathbb{E}[\Gamma(t)] = 0$ and $\mathbb{E}[\Gamma(t)\Gamma(t')] = q\delta(t-t')$. The probability distribution $p(x, t)$ of the solution $\xi(t)$ where $\xi(t) = x$ is

$$p(x, t) = \frac{1}{\sqrt{2\pi qt}} e^{-\frac{(x-\xi_0)^2}{2qt}}. \quad (5.16)$$

Proof. The main difference between this result and Theorem 5.3 is that here we are interested in the distribution at any time t . To do so, we notice that $\xi(t) = \xi_0 + \int_0^t \Gamma(t') dt'$. So, to eliminate the non-zero mean, we can consider $\xi(t) - \xi_0$ instead. Substituting this into Eqn (5.13), we can show that

$$\begin{aligned} \mathbb{E}[(\xi(t) - \xi_0)^{2n+1}] &= 0 \\ \mathbb{E}[(\xi(t) - \xi_0)^{2n}] &= \frac{(2n)!}{2^n n!} \left[q \int_0^t e^{0d\tau_2} \right]^n = \frac{(2n)!}{2^n n!} (qt)^n. \end{aligned}$$

This implies that the characteristic function of $\xi(t) - \xi_0$ is

$$C(u) = \sum_{n=0}^{\infty} \frac{1}{n!} \left(-\frac{u^2 qt}{2} \right)^n = e^{-\frac{u^2 qt}{2}}. \quad (5.17)$$

Taking the inverse Fourier transform will give us the probability distribution for $\xi(t)$:

$$p(x, t) = \frac{1}{\sqrt{2\pi qt}} e^{-\frac{(x-\xi_0)^2}{2qt}}. \quad (5.18)$$

To gain insights about this equation, let's assume $\xi_0 = 0$ and $q = 2k$ for some constant k . This will give us

$$p(x, t) = \frac{1}{\sqrt{4\pi kt}} e^{-\frac{x^2}{4kt}}.$$

An interesting observation of this result, which can be found in many thermal dynamics textbooks, is that the probability distribution $p(x, t)$ derived above is in fact the solution of the **heat equation**:

$$\frac{\partial}{\partial t} p(x, t) = k \frac{\partial^2}{\partial x^2} p(x, t), \quad (5.19)$$

assuming that the initial condition is $p(x, 0) = \delta(x)$. To see this, we just need to substitute the probability distribution into the heat equation. We can then see that

$$\begin{aligned}\frac{\partial}{\partial t} p(x, t) &= \frac{1}{2t} \cdot \left(\frac{x^2}{2kt} - 1 \right) \cdot \frac{1}{\sqrt{4\pi kt}} e^{-\frac{x^2}{4kt}} \\ \frac{\partial^2}{\partial x^2} p(x, t) &= \frac{1}{2kt} \cdot \left(\frac{x^2}{2kt} - 1 \right) \cdot \frac{1}{\sqrt{4\pi kt}} e^{-\frac{x^2}{4kt}}.\end{aligned}$$

The solution to the heat equation behaves like a Gaussian starting at the origin and expanding outward as time increases. The significance of this result is that while it is relatively difficult to know the exact trajectory of the random process $\xi(t)$ defined by the Langevin equation, the heat equation provides a full picture of the probability distribution. Figure 5.2 shows the random realization of a Wiener process $\xi(t)$, and its corresponding probability distribution $p(x, t)$.

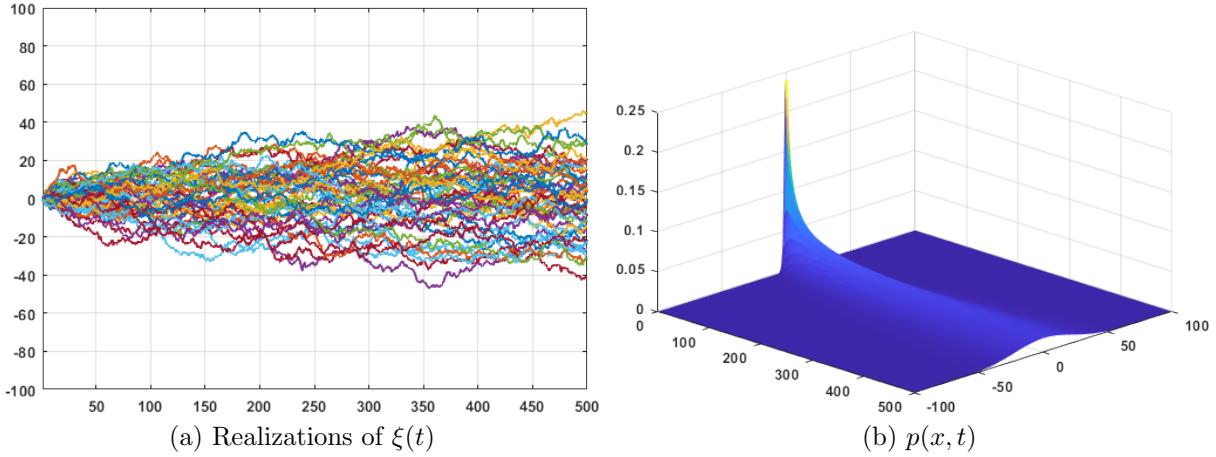


Figure 5.2: Realization of a Wiener process. (a) The random process follows the stochastic differential equation. We show a few realizations of the random process. (b) The underlying probability distribution $p(x, t)$. As t increases, the variance of the Gaussian also increases.

For more complicated Langevin equations (involving nonlinear terms), it seems natural to expect a similar partial equation characterizing the probability distribution. More specifically, it seems reasonable to expect that on one side of the equation, we will have $\partial/\partial t$. And on the other side of the equation, we will have $\partial^2/\partial x^2$. As we will show later, the Fokker-Planck equation will have a form similar to this. Indeed, one can derive the heat equation from the Fokker-Planck equation.

Remark: For a system of homogeneous differential equations of the form

$$\dot{\xi}_i + \sum_{j=1}^N \gamma_{ij} \xi_j = \Gamma_i(t), \quad i = 1, \dots, N,$$

with $\mathbb{E}[\Gamma_i(t)] = 0$ and $\mathbb{E}[\Gamma_i(t)\Gamma_j(t)] = q_{ij}\delta(t-t')$, and $q_{ij} = q_{ji}$, the corresponding random process is known as the **Ornstein-Uhlenbeck** process.

5.2 Masters Equation

Thus far we have been studying the linear Langevin equation $\dot{\xi}(t) + \gamma\xi(t) = \Gamma(t)$. This equation allows us to handle most of the *forward* diffusions where the goal is to add noise to the sample (i.e., convert the input distribution to a Gaussian.) For *reverse* diffusions such as the reverse DDPM equation and the reverse SMLD equation, we would need something more general. The equation we consider now is the nonlinear Langevin equation, expressed as follows.

Definition 5.2. A Nonlinear Langevin Equation takes the form of

$$\dot{\xi} = h(\xi, t) + g(\xi, t)\Gamma(t), \quad (5.20)$$

where $h(\xi, t)$ and $g(\xi, t)$ are functions denoting the drift and diffusion, respectively. Like before, we assume that $\Gamma(t)$ is a Gaussian white noise so that $\mathbb{E}[\Gamma(t)] = 0$ for all t , and $\mathbb{E}[\Gamma(t)\Gamma(t')] = 2\delta(t - t')$.

Readers can refer to Example 5.2 to see how the reverse DDPM would fit this equation.

The difficulty of analyzing the nonlinear Langevin equation is that there is no simple closed-form solution. Therefore, we need to develop some mathematical tools to help us understand the nonlinear Langevin equation.

Markov Property. Let's first define a Markov process. Suppose that $\xi(t)$ has a value $x_n = \xi(t_n)$ at time t_n , and let $t_1 \leq t_2 \dots \leq t_n$. We will use the notation $p(x_n, t_n)$ to describe the probability density of having $\xi(t_n) = x_n$. We also introduce the following short-hand notation

$$\mathbf{x}_n = [x_n, \dots, x_1], \quad \text{and} \quad \mathbf{t}_n = [t_n, \dots, t_1].$$

Therefore, $p(\mathbf{x}_n, \mathbf{t}_n) = p(x_n, t_n, \dots, x_1, t_1)$ is the joint distribution of $(\xi(t_n), \dots, \xi(x_1))$.

Let's define a Markov process. We say that a random process $\xi(t)$ is Markovian if the following memoryless condition is met.

Definition 5.3. A random process $\xi(t)$ is said to be a **Markov process** if

$$p(x_n, t_n | \mathbf{x}_{n-1}, \mathbf{t}_{n-1}) = p(x_n, t_n | x_{n-1}, t_{n-1}). \quad (5.21)$$

That is, the probability of getting state x_n at t_n given *all* the previous states is the same as if we are only conditioning on the immediate previous state x_{n-1} at t_{n-1} .

The random process $\xi(t)$ satisfying the nonlinear Langevin equation defined in Definition 5.2 is Markov, as long as $\Gamma(t)$ is δ -correlated. That means, the conditional probability at t_n only depends on that value at t_{n-1} . The reason was summarized by Risken [33]: (i) A first-order differential equation is uniquely determined by its initial value; (ii) A δ -correlated Langevin force $\Gamma(t)$ at a former time $t < t_{n-1}$ cannot change the conditional probability at a later time $t > t_{n-1}$. Risken further elaborates that the Markovian property is destroyed if $\Gamma(t)$ is no longer δ -correlated. For example, if $\Gamma(t)$ is such that $\mathbb{E}[\Gamma(t)\Gamma(t')] = \frac{q}{2\gamma}e^{-\gamma|t-t'|}$, then the process described by $\dot{\xi}(t) = h(\xi) + \Gamma(t)$ will be non-Markovian. From now on, we will focus only on the Markov processes.

Chapman-Kolmogorov Equation. Consider a Markov process $\xi(t)$. We can derive a useful result known as the Chapman-Kolmogorov equation. The Chapman-Kolmogorov equation states that the joint distribution at t_3 and t_1 can be found by integrating the conditional probabilities of t_3 given t_2 and then t_2 given t_1 . The two key arguments here are the Bayes Theorem plus the definition of marginalization, and the memoryless property of a Markov process.

Theorem 5.5. Chapman-Kolmogorov Equation. Let $\xi(t)$ be a Markov process, and let $x_n = \xi(t_n)$ be the state of $\xi(t)$ at time t_n . Then

$$p(x_3, t_3 | x_1, t_1) = \int p(x_3, t_3 | x_2, t_2)p(x_2, t_2 | x_1, t_1)dx_2, \quad (5.22)$$

assuming $t_1 \leq t_2 \leq t_3$.

Proof. For notational simplicity, let's denote $\mathbf{x}_n = \{x_n, \dots, x_1\}$ and $\mathbf{t}_n = \{t_n, \dots, t_1\}$. If $\xi(t)$ is Markov, then by the memoryless property of Markov, we have that

$$\begin{aligned} p(\mathbf{x}_n, \mathbf{t}_n) &= p(x_n, t_n | \mathbf{x}_{n-1}, \mathbf{t}_{n-1}) p(\mathbf{x}_{n-1}, \mathbf{t}_{n-1}) \\ &= p(x_n, t_n | x_{n-1}, t_{n-1}) p(\mathbf{x}_{n-1}, \mathbf{t}_{n-1}). \end{aligned}$$

Repeating the above argument will give us

$$p(\mathbf{x}_n, \mathbf{t}_n) = p(x_n, t_n | x_{n-1}, t_{n-1}) \cdots p(x_2, t_2 | x_1, t_1) \cdot p(x_1, t_1).$$

Consequently, by marginalizing $p(\mathbf{x}_3, \mathbf{t}_3)$ over x_2 , we can show that

$$\begin{aligned} p(x_3, t_3, x_1, t_1) &= \int p(\mathbf{x}_3, \mathbf{t}_3) dx_2 \\ &= \int p(x_3, t_2 | x_2, t_2) p(x_2, t_2 | x_1, t_1) p(x_1, t_1) dx_2. \end{aligned}$$

By writing $p(x_3, t_3, x_1, t_1) = p(x_3, t_3 | x_1, t_1) p(x_1, t_1)$, we can show that

$$p(x_3, t_3 | x_1, t_1) = \int p(x_3, t_2 | x_2, t_2) p(x_2, t_2 | x_1, t_1) dx_2. \quad (5.23)$$

This completes the proof.

As a corollary of the Chapman-Kolmogorov equation, we can let $x = x_3$ be the current state, $x_0 = x_1$ be the initial state, and $x' = x_2$ be the intermediate state. Then the equation can be written as

$$p(x, t | x_0, t_0) = \int p(x, t | x', t') p(x', t' | x_0, t_0) dx', \quad (5.24)$$

If we further drop the conditioning on x_0 , we can write

$$p(x, t) = \int p(x, t | x', t') p(x', t') dx'.$$

Masters Equation. Based on the Chapman-Kolmogorov equation, we can derive a fundamental equation for Markov processes. This equation is called the Masters Equation.

Theorem 5.6. Let $\xi(t)$ be a Markov process. The **Masters Equation** states that

$$\frac{\partial}{\partial t} p(x, t) = \int [W(x|x') p(x', t) - W(x'|x) p(x, t)] dx', \quad (5.25)$$

where $W(x|x')$ is the probability density function per unit time.

Proof. Recall the Chapman-Kolmogorov Equation

$$p(x_3, t_3 | x_1, t_1) = \int p(x_3, t_3 | x_2, t_2) p(x_2, t_2 | x_1, t_1) dx_2. \quad (5.26)$$

We consider the following mapping:

$$\begin{aligned} (x_1, t_1) &\longrightarrow (x_0, t_0) \\ (x_2, t_2) &\longrightarrow (x, t) \\ (x_3, t_3) &\longrightarrow (x, t + \Delta t) \end{aligned}$$

Then, the Chapman-Kolmogorov Equation can be written as

$$p(x, t + \Delta t | x_0, t_0) = \int p(x, t + \Delta t | x', t) p(x', t | x_0, t_0) dx'. \quad (5.27)$$

Since our goal is to obtain the partial derivative in time, we consider the time derivative of $p(x, t | x_0, t_0)$:

$$\begin{aligned} \frac{\partial}{\partial t} p(x, t | x_0, t_0) &= \lim_{\Delta t \rightarrow 0} \frac{p(x, t + \Delta t | x_0, t_0) - p(x, t | x_0, t_0)}{\Delta t} \\ &= \lim_{\Delta t \rightarrow 0} \frac{\int p(x, t + \Delta t | x', t) p(x', t | x_0, t_0) dx' - p(x, t | x_0, t_0)}{\Delta t}. \end{aligned}$$

We note that on the right-hand side of the equation above, there is an integration. If we switch the variables x and x' , we can use the following observation:

$$\int p(x', t + \Delta t | x, t) dx' = 1.$$

We can insert it into the above equation and obtain

$$\lim_{\Delta t \rightarrow 0} \frac{1}{\Delta t} \left[\int p(x, t + \Delta t | x', t) p(x', t | x_0, t_0) dx' - \int p(x', t + \Delta t | x, t) p(x, t | x_0, t_0) dx' \right]$$

Next, we can move the limits into the integration. Let's define

$$\begin{aligned} W(x, t | x', t) &= \lim_{\Delta t \rightarrow 0} \frac{1}{\Delta t} p(x, t + \Delta t | x', t) \\ W(x', t | x, t) &= \lim_{\Delta t \rightarrow 0} \frac{1}{\Delta t} p(x', t + \Delta t | x, t) \end{aligned}$$

So, we have

$$\frac{\partial}{\partial t} p(x, t | x_0, t_0) = \int [W(x, t | x', t) p(x', t | x_0, t_0) - W(x', t | x, t) p(x, t | x_0, t_0)] dx' \quad (5.28)$$

If we fix (x_0, t_0) , then we can drop the conditioning. This will give us

$$\frac{\partial}{\partial t} p(x, t) = \int [W(x, t | x', t) p(x', t) - W(x', t | x, t) p(x, t)] dx'. \quad (5.29)$$

In the derivation above, the terms $W(x, t | x', t)$ and $W(x', t | x, t)$ are known as the **transition rates**. They are the transition probability *per unit time*, with a unit [time⁻¹]. Thus, if we integrate them with respect to time, we will obtain

$$\begin{aligned} \int W(x, t | x', t) dt &= p(x, t | x', t) \\ \int W(x', t | x, t) dt &= p(x', t | x, t). \end{aligned}$$

One way to visualize the Masters Equation is to consider $\int W(x, t | x', t) p(x', t) dx'$ as the *in-flow* and $\int W(x', t | x, t) p(x, t) dx'$ as the *out-flow* of the transition probability from state x' to x (and from x to x'). So if we view the probability as the density of particles in a room, then the Masters Equation says that the rate of the change of the density is the difference between the in-flow and the out-flow of the particles:

$$\underbrace{\frac{\partial}{\partial t} p(x, t)}_{\text{rate of change}} = \underbrace{\int [W(x, t | x', t) p(x', t)] dx'}_{\text{in-flow of probability}} - \underbrace{\int [W(x', t | x, t) p(x, t)] dx'}_{\text{out-flow of probability}}. \quad (5.30)$$

Masters Equation is used widely in chemistry, biology, and many disciplines. The notion of in-flow and out-flow of particiles is particularly useful to study the dynamics of a system. Another important aspect of the Masters Equation is that it relates time ∂t with the state dx' . This will become prevalent in the Fokker-Planck equation.

One thing we can complain about the above proof is that although it is rigorous, it lacks the physics intuition during the proof. In what follows, we present an alternative proof which is more intuitive but less rigorous. The proof is based on a Lecture Note of Luca Donati [13].

Intuitive Proof. Consider a particle that can take only two states either 1 or 2. The probabilities of landing on a particular state are $p(x_1, t)$ and $p(x_2, t)$, such that $p(x_1, t) + p(x_2, t) = 1$ for any t . Now consider a small interval $(t, t + dt)$. In this interval, the particle can either stay at its current state or it can jump to the other state. This means that at the end of $t + dt$, we can write

$$\begin{aligned} p(x_1, t + dt) &= p(x_1, t)\mathbb{P}[\text{stay in } x_1] + p(x_2, t)\mathbb{P}[\text{move from } x_2 \text{ to } x_1] \\ &= p(x_1, t)\left(1 - \mathbb{P}[\text{move from } x_1 \text{ to } x_2]\right) + p(x_2, t)\mathbb{P}[\text{move from } x_2 \text{ to } x_1] \end{aligned}$$

Let define the rate $W(x_2|x_1)$ and $W(x_1|x_2)$ such that

$$\begin{aligned} \mathbb{P}[\text{move from } x_1 \text{ to } x_2] &= W(x_2|x_1)dt \\ \mathbb{P}[\text{move from } x_2 \text{ to } x_1] &= W(x_1|x_2)dt. \end{aligned}$$

Notice here we have implicitly assumed that the transition distribution is Markov so that the current state only depends on its previous state and not the entire history. Then the above equation can be written as

$$p(x_1, t + dt) = p(x_1, t)(1 - W(x_2|x_1)dt) + p(x_2, t)W(x_1|x_2)dt + \mathcal{O}(dt^2).$$

The high-order term is there to account for multiple jumps during $(t, t + dt)$, e.g., jump from x_1 to x_2 and then from x_2 to x_1 within the interval. However, this term will vanish if $dt \rightarrow 0$. By rearranging the terms, we can write

$$\frac{dp(x_1, t)}{dt} = -W(x_2|x_1)p(x_1, t) + W(x_1|x_2)p(x_2, t).$$

We can generalize this result to multiple states to and from x_1 . For example,

$$\frac{dp(x_1, t)}{dt} = \sum_{j \neq 1} [-W(x_j|x_1)p(x_1, t) + W(x_1|x_j)p(x_j, t)].$$

To make it even more general, we can consider a continuum of x_j . By rearranging the terms, we will obtain

$$\frac{dp(x, t)}{dt} = \int \left[W(x|x')p(x', t) - W(x'|x)p(x, t) \right] dx',$$

which is the Masters equation.

5.3 Kramers-Moyal Expansion

With the Masters Equation developed, we can now tackle the nonlinear Langevin Equation. Recall that the nonlinear Langevin Equation does not have a closed form solution, and hence we cannot write down the probability distribution of the solution analytically. Masters Equation allows us to write down the *conditions* for the probability distribution through a partial differential equation known as the Fokker-Planck Equation. The derivation of the Fokker-Planck Equation requires a mathematical result known as the Kramers-Moyal Expansion.

Theorem 5.7. Let $\xi(t)$ be a Markov process and let $p(x, t)$ be the probability distribution of $\xi(t)$ taking a value x at time t . The **Kramers-Moyal Expansion** states that

$$\frac{\partial}{\partial t} p(x, t) = \sum_{m=1}^{\infty} \left[-\frac{\partial^m}{\partial x^m} D^{(m)}(x, t) p(x, t) \right],$$

where the Kramers-Moyal expansion coefficients are defined as

$$D^{(m)}(x, t) = \frac{1}{m!} \lim_{\Delta t \rightarrow 0} \left[\frac{1}{\Delta t} \mathbb{E} [(\xi(t + \Delta t) - x)^m] \Big|_{\xi(t)=x} \right].$$

Proof. Let's start with the Masters Equation. The Masters Equation states that

$$\begin{aligned} \frac{\partial}{\partial t} p(x, t | x_0, t_0) &= \lim_{\Delta t \rightarrow 0} \frac{1}{\Delta t} \left[\int p(x, t + \Delta t | x', t) p(x', t | x_0, t_0) dx' \right. \\ &\quad \left. - \int p(x', t + \Delta t | x, t) p(x, t | x_0, t_0) dx' \right]. \end{aligned}$$

We can inject a test function $\varphi(x)$ such that

$$\begin{aligned} \frac{\partial}{\partial t} \int \varphi(x) p(x, t | x_0, t_0) dx &= \lim_{\Delta t \rightarrow 0} \frac{1}{\Delta t} \left\{ \int \varphi(x) \int p(x, t + \Delta t | x', t) p(x', t | x_0, t_0) dx' dx \right. \\ &\quad \left. - \int \varphi(x) \int p(x', t + \Delta t | x, t) p(x, t | x_0, t_0) dx' dx \right\}. \end{aligned}$$

Taylor expansion of the test function will give us an infinite series

$$\varphi(x) = \varphi(x') + \sum_{m=1}^{\infty} \frac{(x - x')^m}{m!} \frac{\partial^m}{\partial x^m} \varphi(x) \Big|_{x=x'}.$$

Substituting the expansion in the Masters Equation will give us

$$\begin{aligned} &\frac{\partial}{\partial t} \int \varphi(x) p(x, t | x_0, t_0) dx \\ &= \lim_{\Delta t \rightarrow 0} \frac{1}{\Delta t} \left[\iint \varphi(x') p(x, t + \Delta t | x', t) p(x', t | x_0, t_0) dx' dx \right. \\ &\quad + \iint \sum_{m=1}^{\infty} \frac{(x - x')^m}{m!} \frac{\partial^m}{\partial x^m} \varphi(x) \Big|_{x=x'} p(x, t + \Delta t | x', t) p(x', t | x_0, t_0) dx' dx \\ &\quad \left. - \iint \varphi(x) p(x', t + \Delta t | x, t) p(x, t | x_0, t_0) dx' dx \right] \tag{5.31} \end{aligned}$$

We notice that the last double integral in the equation above has dummy variables x' and x . We can switch the dummy variables, and write

$$\iint \varphi(x) p(x', t + \Delta t | x, t) p(x, t | x_0, t_0) dx' dx = \iint \varphi(x') p(x, t + \Delta t | x', t) p(x', t | x_0, t_0) dx' dx$$

Then, the first and the third double integrals in Eqn (5.31) can be canceled. This will leave us

$$\begin{aligned} & \frac{\partial}{\partial t} \int \varphi(x) p(x, t | x_0, t_0) dx \\ &= \lim_{\Delta t \rightarrow 0} \frac{1}{\Delta t} \iint \sum_{m=1}^{\infty} \frac{(x - x')^m}{m!} \frac{\partial^m}{\partial x^m} \varphi(x) \Big|_{x=x'} p(x, t + \Delta t | x', t) p(x', t | x_0, t_0) dx' dx \end{aligned} \quad (5.32)$$

Now let's define

$$D^{(m)}(x', t) = \frac{1}{m!} \lim_{\Delta t \rightarrow 0} \frac{1}{\Delta t} \int (x - x')^m p(x, t + \Delta t | x', t) dx.$$

Then, the Masters Equation becomes

$$\begin{aligned} & \frac{\partial}{\partial t} \int \varphi(x) p(x, t | x_0, t_0) dx \\ &= \int \sum_{m=1}^{\infty} D^{(m)}(x', t) \frac{\partial^m}{\partial x^m} \varphi(x) \Big|_{x=x'} p(x', t | x_0, t_0) dx' \quad \text{Substitute } D^{(m)}(x', t) \\ &= \sum_{m=1}^{\infty} \int D^{(m)}(x', t) p(x', t | x_0, t_0) \frac{\partial^m}{\partial x^m} \varphi(x) \Big|_{x=x'} dx' \quad \text{Switch summation and integration} \\ &= \sum_{m=1}^{\infty} (-1)^m \int \varphi(x') \frac{\partial^m}{\partial x^m} [D^{(m)}(x, t) p(x, t, x_0, t_0)] dx' \quad \text{Generalized integration by part,} \end{aligned}$$

where the last step is known as the generalized integration by part which states that for any continuously differentiable functions f and g ,

$$\int g \cdot \frac{\partial^m f}{\partial x^m} dx = (-1)^m \int f \frac{\partial^m g}{\partial x^m} dx.$$

Combining all these, and recognizing that the above result holds for any arbitrary $\varphi(x)$, it follows that

$$\frac{\partial}{\partial t} p(x, t | x_0, t_0) = \sum_{m=1}^{\infty} (-1)^m \frac{\partial^m}{\partial x^m} [D^{(m)}(x, t) p(x, t, x_0, t_0)]. \quad (5.33)$$

If we further drop the conditioning on (x_0, t_0) , we will obtain

$$\frac{\partial}{\partial t} p(x, t) = \sum_{m=1}^{\infty} \frac{1}{m!} (-1)^m \frac{\partial^m}{\partial x^m} [D^{(m)}(x, t) p(x, t)] \quad (5.34)$$

which completes the proof.

Kramers-Moyal expansion expresses the time-derivative ∂_t of the probability distribution of any Markov process (including the solution of the nonlinear Langevin Equation) through the spatial-derivative ∂_x . However, the expansion has infinitely many terms. An important question now is whether we allowed to truncate any of these terms. If so, how many terms can be truncated? Pawula Theorem provides an answer to this question: [30]

Theorem 5.8. Pawula Theorem. The Kramers-Moyal expansion may stop at one of the following three cases:

- $m = 1$: The resulting differential equation is known as the Liouville Equation which is a deterministic process.

- $m = 2$: The resulting differential equation is known as the Fokker-Planck Equation.
- $m = \infty$, i.e., the expansion cannot be truncated.

Proof. Recall that the Kramer's-Moyal coefficients are defined as

$$D^{(m)}(x, t) = \frac{1}{m!} \lim_{\Delta t \rightarrow 0} \left[\frac{1}{\Delta t} \mathbb{E} [(\xi(t + \Delta t) - x)^m] \Big|_{\xi(t)=x} \right].$$

Denote $x' = \xi(t + \Delta t)$, the subject of interest here is the m -th moment $\mathbb{E} [(x' - x)^m]$.

We apply Cauchy-Schwarz inequality which states that for any functions f and g , and random variables X and Y , it follows that

$$\mathbb{E}[f(X)g(Y)]^2 \leq \mathbb{E}[f(X)^2]\mathbb{E}[g(Y)^2].$$

We consider two possibilities:

- Suppose that $m \geq 3$ and m is odd, then

$$\mathbb{E}[(x' - x)^m]^2 = \mathbb{E} \left[(x' - x)^{\frac{m-1}{2}} (x' - x)^{\frac{m+1}{2}} \right]^2 \leq \mathbb{E} [(x' - x)^{m-1}] \mathbb{E} [(x' - x)^{m+1}].$$

- Suppose that $m \geq 4$ and m is even, then

$$\mathbb{E}[(x' - x)^m]^2 = \mathbb{E} \left[(x' - x)^{\frac{m-2}{2}} (x' - x)^{\frac{m+2}{2}} \right]^2 \leq \mathbb{E} [(x' - x)^{m-2}] \mathbb{E} [(x' - x)^{m+2}].$$

Note that we cannot apply the above arguments for $m = 0, 1, 2$ because they will give trivial equalities.

From these two relationship, and suppose that we denote $D_m = D^{(m)}(x, t)$, then the above two cases can be written as

$$\begin{aligned} D_m^2 &\leq D_{m-1}D_{m+1}, & m \text{ odd and } m \geq 3, \\ D_m^2 &\leq D_{m-2}D_{m+2}, & m \text{ even and } m \geq 4. \end{aligned}$$

Our goal now is to show that this recurring relationship will give us $D_m = 0$ for any $m \geq 3$.

Suppose first that $D_4 = 0$. Then $D_6 \leq D_4D_8$ implies that $D_6 = 0$. But if $D_6 = 0$ then $D_8 \leq D_6D_{10}$ implies that $D_8 = 0$. Repeating the process will give us $D_m = 0$ for $m = 4, 6, 8, 10, \dots$. Similarly, suppose that $D_6 = 0$. Then $D_4 \leq D_2D_6$ implies that $D_4 = 0$. But if $D_4 = 0$, we can go back to the first case and show that $D_m = 0$ for $m = 4, 6, 8, 10, \dots$. In general, all even $m \geq 4$ should be zero if any one of these even $m \geq 4$ is zero. For the odd m 's: If $D_4 = 0$, then $D_3 \leq D_2D_4$ implies that $D_3 = 0$. Similarly, if $D_6 = 0$, we will have $D_5 = 0$. So if $D_4 = D_6 = D_8 = \dots = 0$, then $D_3 = D_5 = D_7 = \dots = 0$. Therefore, if $D_m = 0$ for any even m such that $m \geq 4$, then $D_m = 0$ all integers $m \geq 3$.

The above analysis suggests that if Kramers-Moyal expansion is truncated up to $m = 3$ so that $D_3 \neq 0$ and $D_4 = D_5 = \dots = 0$, then $D_4 = 0$ will force $D_3 = 0$. So we will have $D_m = 0$ for all $m \geq 3$. Similarly, if the Kramers-Moyal expansion is truncated up to $m = 4$ so that $D_4 \neq 0$ and $D_5 = D_6 = \dots = 0$, then $D_6 = 0$ will force $D_4 = 0$. So we will have $D_m = 0$ for all $m \geq 3$ again.

By repeating the above argument for other $m \geq 3$, we see that it is impossible to have Kramers-Moyal expansion be truncated for any $m \geq 3$. In other words, we can either truncate the expansion for $m = 1, m = 2$, or we never truncate it.

Pawula Theorem does not say that the Fokker-Planck Equation (truncating Kramers-Moyal Expansion up to $m = 2$) is a good approximation to the underlying Masters Equation. It only says that we can either exactly approximate the Masters Equation using $m = 1$ or $m = 2$, or we cannot approximate at all.

5.4 Fokker-Planck Equation

We can now discuss the Fokker-Planck Equation. The Fokker-Planck Equation is the truncation of the Kramers-Moyal's Expansion using $m = 2$.

Definition 5.4. The **Fokker-Planck Equation** is obtained by truncating the Kramers-Moyal expansion to $m = 2$. That is, for any Markov process $\xi(t)$, the probability distribution $p(x, t)$ of $\xi(t) = x$ at time t will satisfy the following partial differential equation:

$$\frac{\partial}{\partial t} p(x, t) = -\frac{\partial}{\partial x} D^{(1)}(x, t)p(x, t) + \frac{\partial^2}{\partial x^2} D^{(2)}(x, t)p(x, t). \quad (5.35)$$

Fokker-Planck Equation is a general result for *any* Markov random processes because it is a consequence of the Chapman-Kolmogorov Equation and the Masters Equation. Processes we study in this tutorial, e.g., Langevin Equation, are special cases of this big family of random processes. Therefore, if we have a Langevin Equation, it is necessary that the solution $\xi(t)$ will have a probability distribution satisfying the Fokker-Planck Equation.

Nonlinear Langevin Equation. If we focus on the nonlinear Langevin equation

$$\dot{\xi} = h(\xi, t) + g(\xi, t)\Gamma(t),$$

we can evaluate the Kramers-Moyal coefficients $D^{(m)}(x, t)p(x, t)$. The following theorem summarizes the coefficients. We remark that during the proof of this theorem, it will become clear why $D^{(m)}(x, t)p(x, t)$ is only limited to $m = 1$ and $m = 2$.

Theorem 5.9. Fokker-Planck for nonlinear Langevin Equation. Consider the nonlinear Langevin equation

$$\dot{\xi} = h(\xi, t) + g(\xi, t)\Gamma(t),$$

for functions $h(\xi, t)$ and $g(\xi, t)$. The Fokker-Planck Equation for this nonlinear Langevin equation will have Kramers-Moyal coefficients:

$$(\text{Drift}) \quad D^{(1)}(x, t) = h(x, t) + g'(x, t)g(x, t) \quad (5.36)$$

$$(\text{Diffusion}) \quad D^{(2)}(x, t) = g^2(x, t). \quad (5.37)$$

Proof. Recall the definition of the Kramers-Moyal coefficient:

$$D^{(m)}(x, t) = \frac{1}{m!} \lim_{\tau \rightarrow 0} \frac{\mathbb{E}[(\xi(t + \tau) - x)^m]}{\tau} \Big|_{\xi(t)=x}.$$

The hard part is how to evaluate the moments $\mathbb{E}[(\xi(t + \tau) - x)^m]$.

We start by looking at the nonlinear Langevin equation:

$$\dot{\xi} = h(\xi, t) + g(\xi, t)\Gamma(t).$$

Expressing $\xi(t + \tau) - \xi(t) = \int_t^{t+\tau} \dot{\xi}(t') dt'$ for a small τ and let $\xi(t) = x$, we can write

$$\xi(t + \tau) - x = \int_t^{t+\tau} [h(\xi(t'), t') + g(\xi(t'), t')\Gamma(t')] dt'.$$

We assume that h and g can be expanded as

$$\begin{aligned} h(\xi(t'), t') &= h(x, t') + h'(x, t')(\xi(t') - x) + \dots \\ g(\xi(t'), t') &= g(x, t') + g'(x, t')(\xi(t') - x) + \dots \end{aligned}$$

This will give us

$$\begin{aligned}
\xi(t + \tau) - x &= \int_t^{t+\tau} \left[h(x, t') + h'(x, t')(\xi(t') - x) + \dots \right] dt' \\
&\quad + \int_t^{t+\tau} \left[g(x, t') + g'(x, t')(\xi(t') - x) + \dots \right] \Gamma(t') dt' \\
&= \int_t^{t+\tau} h(x, t') dt' + \int_t^{t+\tau} h'(x, t')(\xi(t') - x) dt' + \dots \\
&\quad + \int_t^{t+\tau} g(x, t') \Gamma(t') dt' + \int_t^{t+\tau} g'(x, t')(\xi(t') - x) \Gamma(t') dt' + \dots
\end{aligned}$$

Now, we iterate the above equation and replace $\xi(t') - x$ by the integrations. This will give us

$$\begin{aligned}
\xi(t + \tau) - x &= \int_t^{t+\tau} h(x, t') dt' + \int_t^{t+\tau} h'(x, t') \left[\int_t^{t'} h(x, t'') dt'' \right] dt' + \\
&\quad + \int_t^{t+\tau} h'(x, t') \left[\int_t^{t'} g(x, t'') \Gamma(t'') dt'' \right] dt' + \dots \\
&\quad + \int_t^{t+\tau} g(x, t') \Gamma(t') dt' + \int_t^{t+\tau} g'(x, t') \left[\int_t^{t'} h(x, t'') dt'' \right] \Gamma(t') dt' \\
&\quad + \int_t^{t+\tau} g'(x, t') \left[\int_t^{t'} g(x, t'') \Gamma(t'') dt'' \right] \Gamma(t') dt' + \dots
\end{aligned} \tag{5.38}$$

where we only write the terms involving h , g and Γ . Terms involving $\xi(t'') - x$ are not dropped.

Take expectation, and noticing that $\mathbb{E}[\Gamma(t)] = 0$, we can show that only the first two terms and the last term in Eqn (5.38) will survive. Thus, we have

$$\begin{aligned}
\mathbb{E}[\xi(t + \tau) - x] &= \int_t^{t+\tau} h(x, t') dt' + \int_t^{t+\tau} \int_t^{t'} h'(x, t') h(x, t'') dt'' dt' + \dots \\
&\quad + \int_t^{t+\tau} g'(x, t') \underbrace{\int_t^{t'} g(x, t'') 2\delta(t'' - t') dt'' dt'}_{=g(x, t')} + \dots
\end{aligned} \tag{5.39}$$

where we follow Risken's definition that $\int_t^{t'} 2\delta(t'' - t') dt'' = 1$ [33]. As $\tau \rightarrow 0$, it follows that the first and third terms of Eqn (5.39) are

$$\begin{aligned}
\lim_{\tau \rightarrow 0} \int_t^{t+\tau} h(x, t') dt' &= h(x, t), \\
\lim_{\tau \rightarrow 0} \int_t^{t+\tau} g'(x, t') g(x, t') dt' &= g'(x, t) g(x, t).
\end{aligned}$$

For the second term in Eqn (5.39), we can show that

$$\begin{aligned}
\lim_{\tau \rightarrow 0} \int_t^{t+\tau} \int_t^{t'} h'(x, t') h(x, t'') dt'' dt' &= \lim_{\tau \rightarrow 0} \int_t^{t+\tau} h'(x, t') (H(x, t') - H(x, t)) dt' \\
&= h'(x, t) (H(x, t) - H(x, t)) = 0,
\end{aligned}$$

where we assume $h(x, t)$ is integrable over t and so we can define $H(x, t) = \int_0^t h(x, t') dt'$. Therefore, we arrive at

$$D^{(1)}(x, t) = h(x, t) + g'(x, t)g(x, t).$$

The derivation of $D^{(2)}(x, t)$ follows essentially the same set of arguments. The key to note here is that when we take the square in $\mathbb{E}[(\xi(t+\tau) - x)^2]$, the integrals in Eqn (5.39) will give us contributions proportional to τ^2 . When $\tau \rightarrow 0$, all these terms will vanish because there is only one $1/\tau$ in the definition of $D^{(2)}(x, t)$. As a result, the only term that can survive is

$$\begin{aligned} D^{(2)}(x, t) &= \frac{1}{2} \lim_{\tau \rightarrow 0} \frac{1}{\tau} \int_t^{t+\tau} \int_t^{t+\tau} g(x, t')g(x, t'') 2\delta(t' - t'') dt' dt'' \\ &= g^2(x, t), \end{aligned}$$

which completes the proof.

Example 5.4. Consider the following Langevin Equation

$$\dot{\xi} = A(\xi)\xi + \sigma\Gamma(t).$$

Then, the probability distribution $p(x, t)$ for the solution $\xi(t)$ will satisfy the following Fokker-Planck equation

$$\begin{aligned} \frac{\partial}{\partial t} p(x, t) &= -\frac{\partial}{\partial x} [h(x, t) + g'(x, t)g(x, t)] p(x, t) + \frac{\partial^2}{\partial x^2} [g^2(x, t)p(x, t)] \\ &= -\frac{\partial}{\partial x} [(A(x) + 0 \cdot \sigma) p(x, t)] + \frac{\partial^2}{\partial x^2} [\sigma^2 p(x, t)] \\ &= -\frac{\partial}{\partial x} [A(x)p(x, t)] + \sigma^2 \frac{\partial^2}{\partial x^2} [p(x, t)]. \end{aligned}$$

Example 5.5. For the special case where $A(x) = 0$, the Langevin equation is simplified to a Wiener process:

$$\dot{\xi} = \sigma\Gamma(t).$$

The corresponding Fokker-Planck equation is

$$\frac{\partial}{\partial t} p(x, t) = \sigma^2 \frac{\partial^2}{\partial x^2} p(x, t).$$

This equation is known as the **heat equation** or the **diffusion equation**. If the initial condition is that $p(x, 0) = \delta(x)$, the solution is (See derivation below.)

$$p(x, t) = \frac{1}{\sqrt{4\pi\sigma^2 t}} e^{-\frac{x^2}{4\sigma^2 t}}.$$

Solution to Heat Equation. The heat equation can be solved using Fourier transforms. For notational simplicity we denote $u_t = \frac{\partial u}{\partial t}$ and $u_{xx} = \frac{\partial^2 u}{\partial x^2}$. Consider a generic heat equation:

$$u_t(x, t) = ku_{xx}(x, t), \quad x \in \mathbb{R}, t > 0,$$

with initial condition $u(x, 0) = \phi(x)$. We can take Fourier transform (to map between $x \leftrightarrow \omega$) on both

sides by defining

$$\begin{aligned}\widehat{u}_t(\omega, t) &= \mathcal{F}\{u_t(x, t)\} = \int_{-\infty}^{\infty} u_t(x, t)e^{j\omega x}dx \\ \widehat{u}_{xx}(\omega, t) &= \mathcal{F}\{u_{xx}(x, t)\} = \int_{-\infty}^{\infty} u_{xx}(x, t)e^{j\omega x}dx.\end{aligned}$$

This will give us

$$\widehat{u}_t(\omega, t) = k\widehat{u}_{xx}(\omega, t).$$

Using the differentiation property of Fourier transform, we can write the right hand side of the equation as

$$k\widehat{u}_{xx}(\omega, t) = k(j\omega)^2\widehat{u}(\omega, t) = -k\omega^2\widehat{u}(\omega, t),$$

which will give us an ordinary differential equation

$$\widehat{u}_t(\omega, t) = -k\omega^2\widehat{u}(\omega, t). \quad (5.40)$$

The solution to this differential equation (in t) is given by

$$\widehat{u}(\omega, t) = \widehat{\phi}(\omega)e^{-k\omega^2 t}. \quad (5.41)$$

(Remark: Eqn (5.40) is just a simple differential equation $f'(t) = af(t)$ whose solution can be found by integration.) Therefore, if we take the inverse Fourier transform on $\widehat{u}(\omega, t)$ (with respect to $\omega \leftrightarrow x$), we will have

$$u(x, t) = \mathcal{F}^{-1}\{\widehat{u}(\omega, t)\} = \mathcal{F}^{-1}\left\{\widehat{\phi}(\omega)e^{-k\omega^2 t}\right\}$$

which is the inverse Fourier transform on the product of $\widehat{\phi}(\omega)$ and $\widehat{f}(\omega) \stackrel{\text{def}}{=} e^{-k\omega^2 t}$. Since multiplication in the Fourier domain is convolution in the spatial domain, it follows that $u(x, t)$ is the convolution of $\phi(x)$ and $f(x) = \mathcal{F}^{-1}(e^{-k\omega^2 t})$. But $\mathcal{F}^{-1}(e^{-k\omega^2 t}) = \frac{1}{\sqrt{2kt}}e^{-x^2/(4kt)}$. Therefore, we can show that the solution is

$$\begin{aligned}u(x, t) &= \frac{1}{\sqrt{2\pi}} \int_{-\infty}^{\infty} \phi(x - x') f(x') dx' \\ &= \frac{1}{\sqrt{2\pi}} \int_{-\infty}^{\infty} \phi(x - x') \frac{1}{\sqrt{2kt}} e^{-(x')^2/(4kt)} dx'.\end{aligned}$$

So, if $\phi(x) = \delta(x)$ so that $\phi(x - x') = \delta(x - x')$, it follows that

$$u(x, t) = \frac{1}{\sqrt{4k\pi t}} e^{-\frac{x^2}{4kt}}. \quad (5.42)$$

Probability Current. The Fokker-Planck Equation has some interesting physics interpretations. Recall that the Fokker-Planck Equation is

$$\frac{\partial}{\partial t} p(x, t) = -\frac{\partial}{\partial x} D^{(1)}(x, t)p(x, t) + \frac{\partial^2}{\partial x^2} D^{(2)}(x, t)p(x, t). \quad (5.43)$$

Let's define a quantity

$$S(x, t) = \left[D^{(1)}(x, t) - \frac{\partial}{\partial x} D^{(2)}(x, t) \right] p(x, t). \quad (5.44)$$

Then, the Fokker-Planck Equation can be written as

$$\frac{\partial p}{\partial t} + \frac{\partial S}{\partial x} = 0. \quad (5.45)$$

One way to interpret Eqn (5.45) is that it is the **probability current**.

Intuitive Derivation of Eqn (5.45). Conversation of energy tells us that if some increases/decreases in a spatial region (e.g., particles or charges), the change in the amount should be equal to the change in its surface. So, if $p(x, t)$ represents some sort of density, then $p(x, t)dx$ will be the amount of particles sitting between $(x, x + dx)$ at time t . Here, $S(x, t)$ can be viewed as the current of the particle flowing per unit time across x . For a time interval $(t, t + dt)$ and a spatial interval $(x, x + dx)$, the change in amount of particles is

$$\text{number of particles increased/decreased} = [p(x, t + dt) - p(x, t)]dx.$$

Because of the conversation of energy, for this change to happen, there must be some flow of the current. The current over the interval is

$$\text{number of particles flowing in/out} = [S(x, t) - S(x + dx, t)]dt.$$

By equating the two, we will obtain

$$[p(x, t + dt) - p(x, t)]dx = [S(x, t) - S(x + dx, t)]dt.$$

Taylor approximation to the first order will give us $p(x, t + dt) - p(x, t) = \frac{\partial p}{\partial t} dt$ and $S(x, t) - S(x + dx, t) = \frac{\partial S}{\partial x} dx$. This will then give us

$$\frac{\partial p}{\partial t} + \frac{\partial S}{\partial x} = 0. \quad (5.46)$$

Therefore, the Fokker-Planck Equation can be regarded as one form of conversation of energy where the change in p (over time) should be equal to change in S (over space).

Equilibrium Solution. At equilibrium, the current vanishes and so we have $S = 0$. Consequently, we can show the following.

Theorem 5.10. At equilibrium, since the probability current vanishes, the probability distribution $p(x)$ satisfies

$$D^{(1)}(x, t)p(x) = \frac{\partial}{\partial x} [D^{(2)}(x, t)p(x)].$$

Example 5.6. For example, if $D^{(1)} = -\gamma x$ and $D^{(2)} = \frac{\gamma kT}{m}$, then $S = 0$ will give us

$$\left(-\gamma x - \frac{\gamma kT}{m} \frac{\partial}{\partial x} \right) p(x) = 0.$$

Then $-\gamma x p(x) = \frac{\gamma kT}{m} \frac{\partial}{\partial x} p(x)$. Solving this differential equation will give us

$$p(x) = \sqrt{\frac{m}{2\pi kT}} e^{-\frac{m_x^2}{2kT}}.$$

Example 5.7. Connection with SMLD Let's try to map our results with Eqn (3.1) defined in

Definition 3.1. We shall consider the 1D case. Consider the following Langevin equation

$$\underbrace{\frac{dx}{dt}}_{\dot{x}} = \underbrace{\tau \frac{\partial}{\partial x} \log p(x)}_{h(\xi, t)} + \underbrace{\sigma \Gamma(t)}_{g(\xi, t)}.$$

To avoid notational confusions, we let $W(x, t)$ be the probability distribution of the solution $x(t)$ for this Langevin equation. The Kramers-Moyal coefficients for this Langevin Equation are

$$\begin{aligned} D^{(1)}(x, t) &= h(x, t) + g'(x, t)g(x, t) = \tau \frac{\partial}{\partial x} \log p(x) \stackrel{\text{def}}{=} A(x) \\ D^{(2)}(x, t) &= g(x, t)^2 = \sigma^2. \end{aligned}$$

So, the corresponding Fokker-Planck Equation is

$$\begin{aligned} \frac{\partial}{\partial t} W(x, t) &= -\frac{\partial}{\partial x} [D^{(1)}(x, t)W(x, t)] + \frac{\partial^2}{\partial x^2} [D^{(2)}(x, t)W(x, t)] \\ &= -\frac{\partial}{\partial x} [A(x)W(x, t)] + \sigma^2 \frac{\partial^2}{\partial x^2} W(x, t). \end{aligned}$$

At equilibrium when $t \rightarrow \infty$, the probability distribution $W(x, t)$ can be written as $W(x)$. Since the probability current vanishes, it follows that

$$A(x)W(x) = \sigma^2 \frac{\partial}{\partial x} W(x).$$

Recall that $A(x) = \tau \frac{\partial}{\partial x} \log p(x)$, it follows that

$$\tau \frac{\partial}{\partial x} \log p(x) = \sigma^2 \frac{\partial}{\partial x} W(x).$$

Since we have the freedom to choose σ , we will just make it $\sigma = \sqrt{\tau}$. Then the above equation is simplified to $\frac{\partial}{\partial x} \log p(x) = \frac{\partial}{\partial x} W(x)$. Integrating both sides with respect to x will give us

$$\log p(x) = W(x) + C, \quad (5.47)$$

for some constant C . Let $U(x) = e^{W(x)}$ be a probability distribution so that $W(x) = \log U(x)$ is the log-likelihood, Eqn (5.47) will give us $p(x) = U(x)e^C$. Since $p(x)$ and $U(x)$ are probability distributions, we must have $\int p(x)dx = 1$ and $\int U(x)dx = 1$. Thus, we can show that $C = 0$.

Therefore, we conclude that if we run the Langevin equation until convergence, the probability distribution $W(x)$ of the solution is exactly the ground truth distribution $p(x)$. Moreover, the noise level σ and the step size τ is related by $\sigma = \sqrt{\tau}$.

5.5 Concluding Remark

In this section we discussed the physics behind Brownian motion, Langevin equation, and Fokker-Planck equation, and demonstrated several classical theorems in the statistical physics literature. Many of our results are general, as they can be applied to any Markov random process. Going beyond the Markov processes can be done by limiting the spatial/temporal correlation to a small interval.

There are a plethora of references on this topic, many of which originated from physics. Risken's textbook [33] is a classic reference on this subject which contains essentially all the ingredients. For readers looking for something slightly more general, Reichl's statistical physics book [31] would serve the purpose.

6 Conclusion

This tutorial covers a few basic concepts underpinning the diffusion-based generative models in the recent literature. We find it particularly important to go deeper into these fundamental principles rather than staying at the surface of Python programming.

As we write this tutorial, a few lessons we learned are worth sharing. The development of DDPM is in many senses an extension of the VAE, both in terms of the structure, the usage of the evidence lower bound, and the re-parameterization trick. While DDPM is still one of the state-of-the-art methods, it would be more ideal if future models can avoid any iteration. Some recent papers are beginning to explore the feasibility of using knowledge distillation to reduce the number of iterations. Some are investigating acceleration methods by borrowing ideas from the differential equation literature.

For readers who are interested in imaging, the score matching Langevin dynamics would likely continue to play a fundamental role in solving inverse problems. The training of the score-matching function is nearly identical to training an image denoiser, and the application of a score-matching step is identical to a denoising step. Therefore, as soon as we know how to split the inverse problem into the forward model and prior distribution, we will be able to leverage score matching to perform the posterior sampling. Recent approaches have also begun to explore different forms of Bayesian diffusion models to connect the physical model and the data-driven model.

The SDE and Fokker-Planck equations offer great theoretical intuitions as to why the SMLD equation was derived in a certain way. Historically speaking, the development of SMLD does not appear to start with the SDE but the realization today helps us understand the behavior of the solution statistics.

Looking into the future, the biggest challenges of diffusion models are the consistency with the physical world, let alone their high computational complexity. Class-specific learning will continue to be influential. E.g., one can train a customized diffusion model using the gallery images on a cell phone. Temporal consistency would demand for larger models with more memory to store the frames. New architectures are needed to leverage the increasing number of temporal inputs to the model. Another open question is how to bring language and semantics into image generation. Should images continue to be represented as an array of pixels (or pixels of features), or are there new ways to describe the scene using a few words without losing information? Finally, information forensics is going to be the biggest challenge for the next decades until we can develop effective countermeasures (or policies).

Acknowledgement

This work is supported, in part, by the National Science Foundation under the awards 2030570, 2134209, and 2133032, as well as by SRC JUMP 2.0 Center, and research awards from Samsung Research America. Since this draft was posted on the internet in March 2024, we received numerous constructive feedback from readers from all over the world. Thank you all for your input. Thanks also to many of the graduate students at Purdue who shared good thoughts about the content of the tutorial. We want to give a special thanks to William Chi-Kin Yau who worked tirelessly with us on the section about Langevin and Fokker-Planck equations.

References

- [1] Michael S. Albergo, Nicholas M. Boffi, and Eric Vanden-Eijnden. Stochastic interpolants: A unifying framework for flows and diffusions. <https://arxiv.org/abs/2303.08797>.
- [2] Brian Anderson. Reverse-time diffusion equation models. *Stochastic Process. Appl.*, 12(3):313–326, May 1982. <https://www.sciencedirect.com/science/article/pii/0304414982900515>.
- [3] Kendall Atkinson, Weimin Han, and David Stewart. *Numerical solution of ordinary differential equations*. Wiley, 2009. https://homepage.math.uiowa.edu/~atkinson/papers/NAODE_Book.pdf.
- [4] Christopher Bishop. *Pattern Recognition and Machine Learning*. Springer, 2006.
- [5] Charles A. Bouman and Gregory T. Buzzard. Generative plug and play: Posterior sampling for inverse problems. In *2023 59th Annual Allerton Conference on Communication, Control, and Computing (Allerton)*, pages 1–7, 2023. <https://arxiv.org/abs/2306.07233>.
- [6] Robert Brown. A brief account of microscopical observations on the particles contained in the pollen of plants and the general existence of active molecules in organic and inorganic bodies. *Edinburgh New Philosophical Journal*, pages 358–371, 1828.
- [7] Stanley H. Chan. *Introduction to Probability for Data Science*. Michigan Publishing, 2021. <https://probability4datascience.com/>.
- [8] Stanley H. Chan, Xiran Wang, and Omar Elgendy. Plug-and-Play ADMM for image restoration: Fixed point convergence and applications. *IEEE Trans. Computational Imaging*, 3(5):84–98, Mar 2017. <https://arxiv.org/abs/1605.01710>.
- [9] Hyungjin Chung and Jong Chul Ye. Score-based diffusion models for accelerated mri. *Medical Image Analysis*, 80:102479, 2022. <https://arxiv.org/abs/2110.05243>.
- [10] Peter Dayan, Geoffrey E. Hinton, Radford M. Neal, and Richard S. Zemel. The Helmholtz machine. *Neural Computation*, 7(5):889–904, 1995.
- [11] Mauricio Delbracio and Peyman Milanfar. Inversion by direct iteration: An alternative to denoising diffusion for image restoration. *Transactions on Machine Learning Research (TMLR)*, 2023. <https://openreview.net/forum?id=VmyFF51L3F>.
- [12] Carl Doersch. Tutorial on variational autoencoders, 2016. <https://arxiv.org/abs/1606.05908>.
- [13] Luca Donati. From Chapman-Kolmogorov equation to Master equation and Fokker-Planck equation. https://www.zib.de/userpage/donati/stochastics2023/03/lecture_notes/L03_dCKeq.pdf.
- [14] Albert Einstein. On the movement of small particles suspended in stationary liquids required by the molecular-kinetic theory of heat. *Annalen der Physik*, pages 549–560, 1905.
- [15] Ian J. Goodfellow, Jean Pouget-Abadie, Mehdi Mirza, Bing Xu, David Warde-Farley, Sherjil Ozair, Aaron Courville, and Yoshua Bengio. Generative adversarial nets. In *Advances in Neural Information Processing Systems (NeurIPS)*, volume 27, 2014. <https://arxiv.org/abs/1406.2661>.
- [16] Jonathan Ho, Ajay Jain, and Pieter Abbeel. Denoising diffusion probabilistic models. In *Advances in Neural Information Processing Systems (NeurIPS)*, 2020. <https://arxiv.org/abs/2006.11239>.
- [17] Jason Hu, Bowen Song, Xiaojian Xu, Liyue Shen, and Jeffrey A. Fessler. Learning image priors through patch-based diffusion models for solving inverse problems, 2024. <https://arxiv.org/abs/2406.02462>.
- [18] Aapo Hyvärinen. Estimation of non-normalized statistical models by score matching. *Journal of Machine Learning Research (JMLR)*, 6(24):695–709, 2005. <https://jmlr.org/papers/volume6/hyvarinen05a/hyvarinen05a.pdf>.

- [19] Zahra Kadkhodaie and Eero P. Simoncelli. Solving linear inverse problems using the prior implicit in a denoiser, 2020. <https://arxiv.org/abs/2007.13640>.
- [20] Tero Karras, Miika Aittala, Timo Aila, and Samuli Laine. Elucidating the design space of diffusion-based generative models. In *Advances in Neural Information Processing Systems (NeurIPS)*, 2022. <https://arxiv.org/abs/2206.00364>.
- [21] Bahjat Kawar, Michael Elad, Stefano Ermon, and Jiaming Song. Denoising diffusion restoration models. In *Advances in Neural Information Processing Systems (NeurIPS)*, 2022. <https://arxiv.org/abs/2201.11793>.
- [22] Diederik P. Kingma, Tim Salimans, Ben Poole, and Jonathan Ho. Variational diffusion models. In *Advances in Neural Information Processing Systems (NeurIPS)*, 2021. <https://arxiv.org/abs/2107.00630>.
- [23] Diederik P Kingma and Max Welling. Auto-encoding variational Bayes. In *International Conference on Learning Representations (ICLR)*, 2014. <https://openreview.net/forum?id=33X9fd2-9FyZd>.
- [24] Diederik P. Kingma and Max Welling. An introduction to variational autoencoders. *Foundations and Trends in Machine Learning*, 12(4):307–392, 2019. <https://arxiv.org/abs/1906.02691>.
- [25] Andrey Kolmogorov. *Foundations of the Theory of Probability*. Dover, 2018. The original version was published in 1933 in German. <https://dn790007.ca.archive.org/0/items/foundationsofthe00kolm/foundationsofthe00kolm.pdf>.
- [26] Cheng Lu, Yuhao Zhou, Fan Bao, Jianfei Chen, Chongxuan Li, and Jun Zhu. DPM-Solver: A fast ODE solver for diffusion probabilistic model sampling in around 10 steps. In *Advances in Neural Information Processing Systems (NeurIPS)*, 2022. <https://arxiv.org/abs/2206.00927>.
- [27] Calvin Luo. Understanding diffusion models: A unified perspective, 2022. <https://arxiv.org/abs/2208.11970>.
- [28] Chenlin Meng, Robin Rombach, Ruiqi Gao, Diederik Kingma, Stefano Ermon, Jonathan Ho, and Tim Salimans. On distillation of guided diffusion models. In *IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, pages 14297–14306, 2023. <https://arxiv.org/abs/2210.03142>.
- [29] Gabriel Nagy. MTH 235 differential equations, 2024. <https://users.math.msu.edu/users/gnagy/teaching/ade.pdf>.
- [30] R. Pawula. Generalizations and extensions of the Fokker-Planck-Kolmogorov equations. *IEEE Transactions on Information Theory*, 13(1):33–41, 1967.
- [31] L. E. Reichl. *A Modern Course in Statistical Physics*. John Wiley and Sons, Inc, 2 edition, 1998.
- [32] Danilo Rezende and Shakir Mohamed. Variational inference with normalizing flows. In *Proceedings of International Conference on Machine Learning (ICML)*, pages 1530–1538, 2015. <https://arxiv.org/abs/1505.05770>.
- [33] Hannes Risken. *The Fokker-Planck Equations: Methods of solutions and applications*. Springer, 2 edition, 1989.
- [34] Robin Rombach, Andreas Blattmann, Dominik Lorenz, Patrick Esser, and Björn Ommer. High-resolution image synthesis with latent diffusion models. In *IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, pages 10684–10695, 2022. <https://arxiv.org/abs/2112.10752>.
- [35] Chitwan Saharia, William Chan, Saurabh Saxena, Lala Li, Jay Whang, Emily L Denton, Kamyar Ghasemipour, Raphael Gontijo Lopes, Burcu Karagol Ayan, Tim Salimans, Jonathan Ho, David J Fleet, and Mohammad Norouzi. Photorealistic text-to-image diffusion models with deep language understanding. In *Advances in Neural Information Processing Systems (NeurIPS)*, volume 35, pages 36479–36494, 2022. <https://arxiv.org/abs/2205.11487>.

- [36] Tim Salimans and Jonathan Ho. Progressive distillation for fast sampling of diffusion models. In *International Conference on Learning Representations (ICLR)*, 2022. <https://arxiv.org/abs/2202.00512>.
- [37] Yash Sanghvi, Yiheng Chi, and Stanley H. Chan. Kernel diffusion: An alternate approach to blind deconvolution. In *European Conference on Computer Vision (ECCV)*, 2024. <https://arxiv.org/abs/2312.02319>.
- [38] Jascha Sohl-Dickstein, Eric Weiss, Niru Maheswaranathan, and Surya Ganguli. Deep unsupervised learning using nonequilibrium thermodynamics. In *Proceedings of International Conference on Machine Learning (ICML)*, volume 37, pages 2256–2265, 2015. <https://arxiv.org/abs/1503.03585>.
- [39] Jiaming Song, Chenlin Meng, and Stefano Ermon. Denoising diffusion implicit models. In *International Conference on Learning Representations (ICLR)*, 2023. <https://openreview.net/forum?id=St1giarCHLP>.
- [40] Yang Song and Stefano Ermon. Generative modeling by estimating gradients of the data distribution. In *Advances in Neural Information Processing Systems (NeurIPS)*, 2019. <https://arxiv.org/abs/1907.05600>.
- [41] Yang Song and Stefano Ermon. Improved techniques for training score-based generative models. In *Advances in Neural Information Processing Systems (NeurIPS)*, 2020. <https://arxiv.org/abs/2006.09011>.
- [42] Yang Song, Jascha Sohl-Dickstein, Diederik P. Kingma, Abhishek Kumar, Stefano Ermon, and Ben Poole. Score-based generative modeling through stochastic differential equations. In *International Conference on Learning Representations (ICLR)*, 2021. <https://openreview.net/forum?id=PxTIG12RRHS>.
- [43] Pascal Vincent. A connection between score matching and denoising autoencoders. *Neural Computation*, 23(7):1661–1674, 2011. https://www.iro.umontreal.ca/~vincentp/Publications/smdae_techreport.pdf.
- [44] M. von Smoluchowski. Zur kinetischen theorie der brownschen molekularbewegung und der suspensio-nen. *Annalen der Physik*, pages 756–780, 1906.
- [45] Martin J. Wainwright and Michael I. Jordan. Graphical models, exponential families, and variational inference. *Foundations and Trends in Machine Learning*, 1(1-2):1–305, 2008. <https://cba.mit.edu/events/03.11.ASE/docs/Wainwright.1.pdf>.
- [46] Max Welling and Yee Whye Teh. Bayesian learning via stochastic gradient Langevin dynamics. In *Proceedings of International Conference on Machine Learning (ICML)*, pages 681–686, 6 2011. <https://www.stats.ox.ac.uk/~teh/research/compstats/WelTeh2011a.pdf>.