An Introduction to Generative Modeling with Flow-based and Diffusion-based Models

Jae Hyun Lim

v0.2 (Draft)

Part 2 Appendix

Chapter 1

Introduction

In machine learning, *generative modeling* refers to the task of learning the underlying statistical properties of data distributions from observed samples. To achieve this, a fundamental principle in modern machine learning is to view generative modeling from a probabilistic perspective.

From this perspective, *data* (or *data points*) are regarded as the outcome of some data generation procedure, and the procedure itself is treated as a *random variable*–a mathematical experiment (or object) that embodies randomness. The random variable produces an outcome known as a *sample*, the collection of which becomes our data, and the underlying rule governing this random experiment is the *distribution* of the random variable.[1]

In this sense, generative modeling is the activity of defining a new random variable—referred to as a *model*—and adjusting its distribution to match that of the data based solely on the observed samples. The model in this context is called a *generative model*, and the primary focus of research in generative modeling is to address the challenge of aligning the model's distribution with the data's distribution.[2]

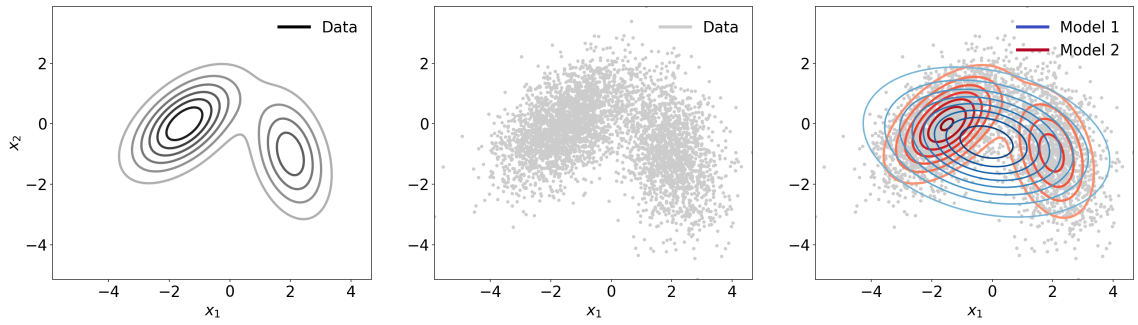


Figure 1.1: Generative modeling (continuous random variable). (Left) probability density function (pdf) governs the behavior (distribution) of a random variable. (Middle) Samples generated from the data's distribution. (Right) Two different models fitted to the data samples.

Building on this probabilistic view, modern generative models extend these ideas in powerful ways. However, the mathematical machinery behind many of these methods is becoming increasingly complex. To properly understand them, it has become increasingly important to grasp the following definitions in a rigorous way:

(i) The definition of a random variable and its distribution.
(ii) The relationship between samples and their distributions.
(iii) Various characterizations of distributions.
(iv) The precise meaning of a model 'learning' from data.

[1] In some contexts, data samples or data points are simply referred to as data, and the distribution of the data is also termed as data.
[2] Of course, generative modeling also encompasses other types of problems beyond determining a distribution from data samples, which we will cover later in this tutorial.

Yet, developing a rigorous understanding of these concepts can be challenging, and it often creates a significant barrier for newcomers.

The aim of this tutorial is therefore to lower the entry barrier for students and researchers who may not have a strong mathematical background, by connecting key concepts with the underlying mathematics and helping readers develop familiarity with the required machinery.

In addition, this tutorial introduces the essential ideas for unfamiliar readers and clarifies how the development of modern diffusion and flow-based models can be motivated in terms of fundamental probabilistic concepts and key design criteria in generative modeling. Since research papers are usually intended for readers who share a similar background with the authors, these underlying motivations are often assumed and left unarticulated, which can raise the entry barrier for newcomers to the field. We take a deliberate step back to examine these concepts in depth. By doing so, we seek to illuminate the underlying motivations of recent generative models and provide structured training materials for assessing their theoretical alignment.

To effectively convey the ideas presented above, it is important to recognize that there are multiple levels of explanation and a variety of ways to present the material. The remainder of this chapter will therefore lay out several key pieces of meta-information to help readers make the most of this tutorial:

(i) The intended audience and their assumed background knowledge.
(ii) Important considerations when engaging with this content.
(iii) Specific learning objectives we hope readers will achieve.
(iv) Notational conventions.

By clarifying these aspects up front, we aim to provide readers with a roadmap that frames the rest of the chapter and enables a more purposeful and informed learning experience.

Target readers With these overarching goals in mind, it is helpful to be explicit about who this tutorial is intended for and the level of background knowledge it assumes. This tutorial is written for readers who already possess a basic understanding of probabilistic graphical models and undergraduate-level probability theory, along with a general familiarity with core concepts in deep learning—while a thorough reading of texts such as *Deep Learning* by Goodfellow et al. (2016) or *Pattern Recognition and Machine Learning* by Bishop & Nasrabadi (2006) may not be expected, it is assumed that readers have are reasonably familiar with these ideas from their coursework or research experience.

In particular, we aim to support those who are interested in developing a more rigorous understanding of advanced generative modeling but feel constrained by gaps in their mathematical background, such as students from engineering disciplines. However, even for such motivated learners, the recent rapid influx of resources and publications in this field can make it difficult to determine what to study and where to begin. This tutorial aims to address that challenge by presenting a curated and conceptually grounded pathway into the subject, which we hope will serve as a useful starting point for those beginning graduate-level research.

Notes on scope and perspective Before proceeding, a few notes on how to approach this tutorial may be helpful. Much of the material presented here is adapted from my submitted PhD thesis. As mentioned earlier, the discussion focuses on relatively recent generative models, examined primarily from three perspectives that I consider fundamental: expressivity, bias, and training stability. From the viewpoint of non-mathematicians, the content may appear overly theoretical and perhaps impractical; conversely, from a mathematician's standpoint, it may lack full rigor or foundational completeness. Nevertheless, writing from the perspective of a non-mathematician, I have chosen to concentrate on the mathematical motivations underlying each method. I believe that once these theoretical motivations are well understood, the importance of the corresponding techniques for each model will become much clearer.

In light of this focus, the tutorial may not cover certain important techniques in depth, such as network architecture design, optimization methods, or engineering tricks. While these elements are often just as important as theory—and deeply intertwined with it in deep generative models—it is not always possible to treat them entirely independently. Even so, I will make a conscious effort to highlight them only where they are directly relevant to the theoretical

discussion. The overarching aim is to provide a broad conceptual understanding of each topic, so I encourage readers to treat this tutorial as a gateway into the field. If a particular area sparks deeper interest, the references provided should serve as a useful starting point for further study.

Finally, it is worth noting that the topics covered in each chapter often originate from different subfields, and as a result, the original papers may use a wide variety of terminology. In this tutorial, I have opted for consistency by following terminology commonly used in mathematics and statistics, as these fields have the longest-standing traditions and the most precise references. My hope is that this choice will make it easier for readers to look up additional details on their own.

Before you begin A solid grasp of the four key building blocks introduced above will make the discussions throughout this tutorial more coherent and meaningful. To achieve this, we will describe these building blocks from a probabilistic point of view, specifically adopting the measure-theoretic perspective. This perspective will also serve as the foundation for the material in the rest of the tutorial. Probability theory—when understood in this formal way—provides a consistent language that will greatly aid our understanding of many generative models to come.

As noted earlier, this tutorial is written for readers with a science or engineering background at the college level (or early graduate level), but not necessarily with formal training in mathematics. For such readers, the measure-theoretic formulation of probability may feel unfamiliar. However, expecting them to work through a full textbook in advanced mathematics would be an unreasonable demand. To bridge this gap, we provide a concise overview of probability theory from the measure-theoretic perspective. To preserve the flow, this material is deferred to Appendix A. I recommend reading it around Chapters 3 and 4, when it will be most relevant. Many concepts that may already be familiar—including integration and expectation—are redefined there in a way that generalizes to broader contexts. These generalizations can feel less intuitive at first, but they are correspondingly more versatile. For readers encountering this perspective for the first time, I strongly encourage taking the time to read that section carefully—the effort will be well rewarded.

Conventions and notation In this tutorial, the first-person plural pronoun *we* will be used instead of *I*, reflecting the collaborative nature of the work presented in this tutorial. Random variables will be denoted by uppercase letters such as X, Y, or Z, while lowercase letters, such as x, will represent deterministic values. Accordingly, the deterministic value of a random variable X will typically be written in lowercase, for example x for X, and similarly for other variables. An exception is the lowercase epsilon ε, which will be used to denote random noise.

We will occasionally use blackboard-bold symbols, such as $\mathbb{R}$ and $\mathbb{S}$, to represent spaces or topological sets. In general, however, uppercase letters such as H will be used to denote spaces like Hilbert spaces. Uppercase calligraphic letters, such as $\mathcal{H}$, $\mathcal{X}$, or $\mathcal{F}$, will be used to denote σ-algebras. In some contexts, blackboard-bold symbols such as $\mathbb{P}$ and $\mathbb{Q}$ will be used to represent path measures of stochastic processes.

We will not use bold lowercase letters to distinguish scalars from vectors; instead, when explicit distinction is needed, tuples will be used. For example, an element $x \in \mathbb{R}^d$ will be written as $x = (x_1, x_2, \ldots, x_d)$. Mappings, including functions and operators, will generally be expressed explicitly together with their arguments—for instance, for an input x, the output of a function f will be written as $f(x)$. When the argument is unspecified, the notation $f(\cdot)$ will be used.

When introducing a new term for the first time, it will be written in *italic* to highlight its definition. Additional notations will be introduced as needed throughout the tutorial.

As this tutorial is intended to serve as a learning resource, feedback from readers will be especially valuable. In particular, if you notice places where citations are missing, could be improved, or where alternative references would provide greater clarity, I warmly encourage you to share those suggestions.

References

Bishop, C. M. and Nasrabadi, N. M. *Pattern recognition and machine learning*, volume 4. Springer, 2006.

Goodfellow, I., Bengio, Y., and Courville, A. *Deep learning*. MIT press, 2016.

Part 1

Euclidean

Chapter 2

Modeling continuous random variables

2.1 Introduction

In Chapter 1, we described generative modeling as the process of defining a model random variable and adjusting its distribution to align with that of the data, using only observed samples as guidance. Building on this working definition, we now turn to a more fundamental question: what does it truly mean to model a random variable, and what choices do we have in doing so? Rather than focusing narrowly on the mechanics of a few popular approaches, this chapter aims to develop a broader perspective—one that helps us see how different modeling strategies, each with their own distinctive characteristics, shape the ability of a model to reproduce the data distribution. This perspective will serve as a foundation for understanding and comparing the more specific methods introduced later in the tutorial.

To pursue this, we first examine the key components that define generative models in Section 2.2, including how random variables are represented, how they can be trained, and how their sample spaces relate to those of the data. In particular, we emphasize the importance of aligning the model's sample space with the data distribution; in practice, this alignment is often overlooked, leading to inevitable errors. Next, we provide an overview of major families of continuous generative models in Section 2.3, briefly analyzing them with respect to these criteria. We then explore several distance metrics and divergences in Section 2.4, which formalize the notion of closeness between model and data distributions. Finally, in Section 2.5, we will discuss the broader design criteria and highlight the key factors—expressivity, bias, and training stability—that have driven much of the development in modern generative modeling. These criteria will clarify the motivations behind generative models' the distinctive features and the motivations behind them.

To address these topics more effectively, we assume that readers are somewhat familiar with the terminology of probability theory. For those less comfortable, we have included a dedicated summary in Appendix A, which introduces core concepts such as random variables and distributions in a way that extends naturally to more general spaces. While these definitions may feel less intuitive at first, they provide a versatile foundation, and we encourage unfamiliar readers to review this material carefully.

2.2 Key components of generative models

In this section, we briefly review several core components of generative models, with a focus on three aspects: how random variables are represented, how such models can be trained, and how their sample spaces align with those of the data. Each of these aspects will be illustrated through examples.

The most important component in generative modeling is how the model is defined as it determines how its random variable is represented and what statistical properties we can access. The choice of representation, in turn, constrains the training methods available: depending on what information about the distribution is tractable, some approaches are feasible while others are not. Finally, the alignment between the model's sample space and the data's sample space is

a subtle but crucial consideration. Although technically part of model specification, this issue is often overlooked in practice. It is especially relevant for diffusion- and flow-based models—the primary focus of this tutorial—where the choice of sample space strongly influences the mathematical machinery that defines these models.

2.2.1 Random variable representations

To motivate the discussion, let us start with an example that highlights two complementary ways of defining random variables. This will help clarify why the choice of representation plays such an important role in practice.

Example 2.1. [Explicit density models vs. implicit density models] For the first example, assume we would like to model a random variable $X \sim N(m, C)$, where $m \in \mathbb{R}^d$ and C is a symmetric and positive semi-definite matrix with the size $d \times d$. The random variable X can also be written by

$$X = m + LZ \coloneqq f_\theta(Z),$$

where L is a $d \times d$-matrix satisfying $LL^\top = C$, Z is and a $\mathbb{R}^d$-random variable following the standard normal distribution, and a function $f_\theta : \mathbb{R}^d \to \mathbb{R}^d$ is a linear map with parameters $\theta = \{m, C\}$. In this case, we can **write its density explicitly**

$$p(x) = \frac{1}{\sqrt{(2\pi)^d \det(C)}} \exp\big(-(X-m)^\top C^{-1}(X-m)\big).$$

We may be able to update the learnable parameters by maximizing log-likelihoods for data observations. Similarly, there are several approaches to define various distributions, but with the intention to evaluate their probability density functions in mind. We refer to such a family of random variable models as *explicit density models*.

For the second example, let Z be a $\mathbb{R}^d$-random variable that follows the standard normal distribution, denoting $Z \sim N(0, I)$. Assume we have a ϕ-parameterized function $g_\phi : \mathbb{R}^d \to \mathbb{R}^d$. Define Y by

$$Y = g_\phi(Z).$$

While we define Y simply by specifying its generation process, the resulting Y is a valid $\mathbb{R}^d$-valued random variable (under mild assumptions on g_ϕ), and the generation process induces its distribution. Likewise, we can compose many complex processes to define a generation process. In general, however, we **don't know what their densities would look like**; hence, we name such models by *implicit density models*. One can construct this family of models very easily, but it isn't easy to analyze their statistical properties. Such models are often called *implicit density models*.

Example 2.1 shows two different strategies to define random variables. Someone may need densities so that they can compare two observations by evaluating the log-likelihoods under the same model. On the contrary, one would like to follow a specific generation process in order to preserve the process's statistical properties. In addition to the above two approaches, there are numerous ways to define random variables. We will discuss this perspective of generative modeling in Section 2.3.

2.2.2 Training methods

The next example describes two different training methods for the same random variable model. This will help us understand the trade-offs of different training strategies.

Example 2.2. [Likelihoods vs. variational divergence] Suppose we want to model a random variable X, following a data distribution $N(m_{\text{data}}, C_{\text{data}})$, where $m_{\text{data}} \in \mathbb{R}^d$ is its mean and C_{data} is its $d \times d$-covariance matrix. Here, we write p_{data} for its probability density function. We denote n-number of i.i.d. observations as $x^{(i)} \overset{\text{i.i.d.}}{\sim} p_{\text{data}}$ for $i = 1, \ldots, n$.

Let Y be a model random variable defined by $N(m, C)$, where m, C are its mean and covariance, respectively. Here, m and C are learnable parameters and we write $\theta = \{m, C\}$. Moreover, we denote its density as p_θ.

One strategy to train Y is **minimizing the Kullback–Leibler (KL) divergence** between two distributions with respect to θ. The KL is written by

$$\begin{aligned} D_{\text{KL}}\left(p_{\text{data}} \| p_\theta\right) &= \mathbb{E}[\log p_{\text{data}}(X) - \log p_\theta(X)] \\ &= \frac{1}{n}\sum_{i}^{n}\Big[\underbrace{\log p_{\text{data}}(x^{(i)})}_{\theta\text{-independent}} - \underbrace{\log p_\theta(x^{(i)})}_{\text{log-likelihood}}\Big]. \end{aligned}$$

It is well known that if D_{KL} is zero, then the statistical properties of X and Y are equal (more precisely, they are equal in distribution). Moreover, since the first term in the summation is θ-independent, minimizing this KL-divergence is equivalent to maximizing the log-likelihoods, which we are more familiar with.

On the contrary, Y can also learn X **by minimizing a different discrepancy measure**. For example, we can use the Wasserstein-1 distance is an integral probability metric, and it can be written as

$$\begin{aligned} W_1(p_{\text{data}} \| p_\theta) &= \sup_{f \in \mathcal{F}_{\texttt{1-Lips}}} \left|\mathbb{E}[f(X)] - \mathbb{E}[f(Y)]\right| \\ &= \sup_{f \in \mathcal{F}_{\texttt{1-Lips}}} \left|\frac{1}{n}\sum_{i=1}^{n} f(x^{(i)}) - \frac{1}{n}\sum_{i=1}^{n} f(y^{(i)})\right|, \end{aligned}$$

where $\mathcal{F}_{\texttt{1-Lips}}$ is the set of 1-Lipschitz functions, and $y^{(i)} \overset{\text{i.i.d.}}{\sim} p_\theta$ for $i = 1, \ldots, n$ are n samples of Y drawn i.i.d. Similarly to the KL-divergence, X and Y are equal in distribution if the distance is zero.

Unlike the KL-divergence, however, the Wasserstein-1 distance in the above form has a supremum. For any given θ, the proper value of this distance can be obtained only when $\sup$ is satisfied. Thus, for updating θ to the direction of minimizing the distance, we need to optimize the inner loop first. Assume we have $f_\phi : \mathbb{R}^d \to \mathbb{R}$ such that f_θ is always 1-Lipschitz. Then, for θ fixed, we solve the following first

$$\max_{\phi} \left|\frac{1}{n}\sum_{i=1}^{n} f_\phi(x^{(i)}) - \frac{1}{n}\sum_{i=1}^{n} f_\phi(y^{(i)})\right|.$$

Once we obtain a sufficiently good ϕ, we update θ by minimizing the distance for ϕ fixed. After the update, we probably need to solve the inner loop again for the new θ. During performing the inner loop, maintaining f_ϕ to be 1-Lipschitz may not be trivial; however, numerous works have addressed this topic and they can be found in Arjovsky et al. (2017); Gulrajani et al. (2017); Yoshida & Miyato (2017); Roth et al. (2017); Miyato et al. (2018).

Example 2.2 demonstrates that multiple ways exist to train the same model. Minimizing the KL, *a.k.a.* maximizing log-likelihoods, seems more straightforward in the example. On the contrary, minimizing Wasserstein-1 distance only needs samples and doesn't even require any information about the model density p_θ. In addition to this, those two discrepancy measures have more distinguishable characteristics; for example, they have different loss landscapes around the optimal θ^*. In addition to KL and Wasserstein-1, there exist various discrepancy measures that are worth exploring. Someone may prefer one measure over another depending on chosen random variable representations, preferred learning behaviors, and so on. We will continue to discuss this topic in Section 2.4.

2.2.3 Sample spaces

The third example depicts a case when the domain of a model and the data's domain are different. The example will help us understand how such disparity causes unwanted outcomes.

Example 2.3. [Uniform vs. Gaussian] Let X be a real-valued random variable following a uniform distribution from 0 to 1, denoting $X \sim U([0,1])$. We would like to learn it by a normal distribution $N(m,c)$ for $m \in \mathbb{R}$ and $c > 0$. We write its density as p_θ by letting $\theta = \{m, c\}$ be learnable parameters. Assume we train the model by maximizing log-likelihoods, then we have θ^*

$$\theta^* = \arg\max_\theta \frac{1}{n} \sum_{i=1}^{n} \log p_\theta(x^{(i)}),$$

where $x^{(i)} \overset{\text{i.i.d.}}{\sim} U([0,1])$ for $i = 1, \ldots, n$. In this case, even though $\theta^* = \{m^*, c^*\}$ is optimal, there exists non-zero probability $Y \sim p_{\theta^*}$ will lie outside the range $[0,1]$.

Example 2.3 illustrates a scenario where the models fail to comply with the constraints of the data's space, resulting in **the generation of incorrect samples that would never occur in the actual data**. Therefore, adhering to the data's space is crucial for constructing generative models.

However, in general, defining complex models like diffusion-based models in constrained sample spaces is more challenging than it may initially seem. This is because, in certain topological spaces, common calculus rules—that are valid in Euclidean spaces—may not be directly applicable. We will gradually learn about these more challenging cases step by step throughout the tutorial.

So far, we have explored three examples related to the building blocks of generative models. Keeping these in mind, we will examine examples of two key components; in particular, we will focus on (continuous) random variable models and comparing distributions. Note that the early part of this tutorial will focus on modeling *continuous* random variables, such as those on the n-dimensional Euclidean space $\mathbb{R}^n$. Here, the term 'continuous' indicates in those spaces properties like *closeness* and *limits* between its elements are well defined via so-called distances. Later in the tutorial, we will discuss how those definitions are consistently applicable to non-Euclidean spaces and thus extend generative models on those spaces, including discrete states.

2.3 Continuous random variable models

In the previous section, we discussed the fundamental building blocks for defining a generative model. Among them, the first key aspect was defining the behavior of a random variable in some form. As we saw in Chapter A, if we can specify any one of the law, distribution function, or probability density function (pdf), we can fully define a random variable. Moreover, a random variable can be constructed by applying a mapping or a composition of mappings to another random variable.

But how are these definitions actually used in practice? Understanding this process is crucial because the statistical properties we can control depend on how generative models define their random variables. In this section, we will overview how some popular generative models represent random variables. Figure 2.1 illustrates the generative model families that will be described in this section, and they are arranged depending on their random variable representations.

2.3.1 Autoregressive models

Around the time before flow-based models and GAN-based models had been proposed, using pdfs of well-known continuous random variables was prevalent in modeling data distributions. For Euclidean spaces, popular choices are normal, Laplace, student's t-distributions, or their mixture models. While their statistical properties are well studied,

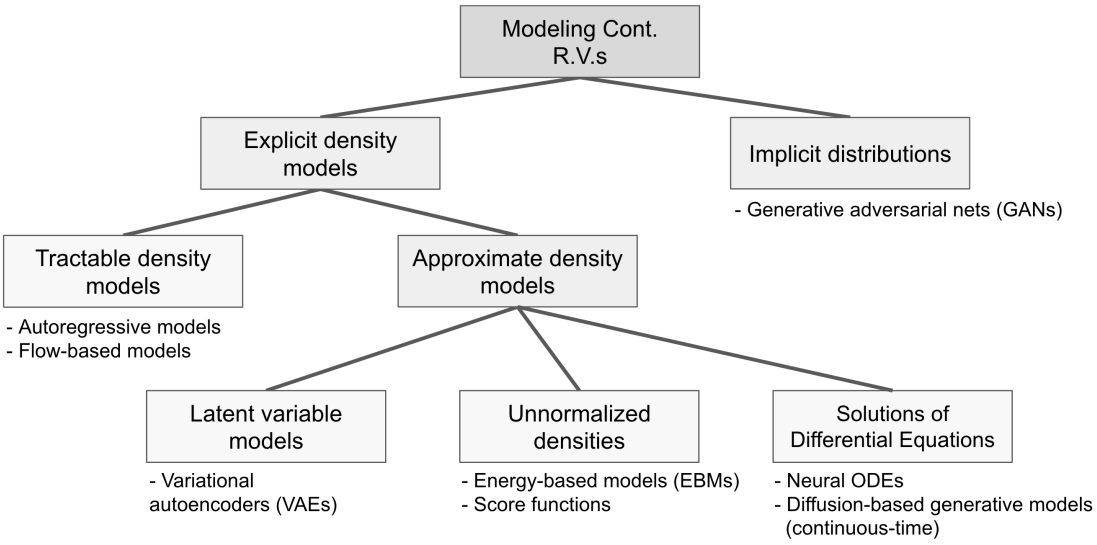


Figure 2.1: Continuous random variable models

available parameters to control their behaviors are limited to a finite number of moments, such as mean, variance, skewness, and so on. Thus, there exist apparent limitations to representing arbitrarily complex distributions by using a few moments. In particular, for modeling n-dimensional data, n-th order statistics requires d^n-number of computations, and it exacerbates the practicality of using such well-known densities.

In this context, one has proposed to factorize the density and model the pdf of each dimension (Frey, 1998; Larochelle & Murray, 2011). Let $X = (X_1, \ldots, X_d)$ be a $\mathbb{R}^d$-valued random variable that follows a pdf p, which is defined as

$$p(x) = p(x_1, x_2, \ldots, x_n) = p(x_1|x_2, \ldots, x_d)p(x_2|x_3, \ldots, x_d) \cdots p(x_d). \tag{2.1}$$

This family of models is called *autoregressive (AR) models*, whose name originated from statistics, econometrics, and signal processing. Note that by treating i-th dimension of X as a i-th time step, predicting X_i conditions on its own history $X_1, \ldots, X_{i-1}$, hence the name. Even though the pdf of each dimension may not be a universal pdf approximater, the expressivity of the overall pdf has significantly improved.

The AR models define density explicitly, and we can compute their likelihoods exactly. Hence, it is often categorized as *explicit density models* as in Figure 2.1. In exchange for improving the expressivity, the AR models trade off sampling efficiency. In order to sample X, one needs to sample X_1 first, then X_2 conditioning on the sample of X_1, and so on.

2.3.2 Variational autoencoders

In the density modeling literature, a central research focus has been developing more effective models for representing arbitrary continuous random variables. In this context, variational autoencoders (VAEs) quickly became one of the most popular approaches following their introduction by Kingma & Welling (2014); Rezende et al. (2014).

One reason for this is that VAEs can be viewed as a form of infinite Gaussian mixture models (IGMMs), which, in theory, are capable of representing any arbitrary continuous probability density function (pdf)[1]

VAEs are latent variable models (LVMs), and they explicitly specify the pdfs. For a VAE, a pdf for an input x is

[1]We often call such models s universal pdf approximators, and we will explore this aspect further in Section 2.5.1.

defined by

$$p(x) = \int p(x|z)p(z)\,\mathrm{d}z, \tag{2.2}$$

where z is called a *latent variable*. The common VAE models parameterize the likelihood $p(x|z)$, often called *decoder*, to be isotropic Gaussian distributions[2], and the prior $p(z)$ to be simple standard Gaussian distributions with no trainable parameters. This model can be understood as an infinite Gaussian mixture model (IGMM) as we have uncountably many different Gaussian modes, each of which is defined as $p(x|z)$. Moreover, as IGMMs are universal approximators (see Section 2.5.1), the model defined by Equation 2.2 can approximate any continuous pdf arbitrarily closely.

While the VAEs are universal pdf approximators, their training requires some caution. In order to compute log-likelihood, computing the expectation in Equation 2.2 is intractable, and thus, one may need estimators for the integral. We will revisit and delve deeper into the VAEs in Section 3.

2.3.3 Energy-based models

A *energy-based model* (*EBM*, Li, 2001; Winkler, 2002; Teh et al., 2003; He et al., 2004; Carreira-Perpinan & Hinton, 2005) is a Boltzmann distribution, and it is defined by a energy function $f(x)$; that is,

$$p_\theta(x) = \frac{\exp\left(-f(x)\right)}{Z}, \tag{2.3}$$

where $Z = \int \exp\left(-f(x)\right) dx$ and is called a *normalizing constant*. As long as f can approximate any continuous function, the Boltzmann density in Equation 2.3 can represent arbitrary density. It is well known that deep neural networks can model any continuous function, and thus, EBMs are powerful.

However, its density contains the normalizing constant Z, and its computation is intractable in general. Thus, computing log-likelihoods of EBMs is extremely challenging. Moreover, it is not straightforward to generate samples from the EBMs. In particular, without knowing the normalizing constant, one cannot use a common sampling technique that feeds uniform distribution to inverse cdfs. We won't cover this topic in this tutorial. However, we will revisit the training of EBMs and relevant topics in Section 5.

2.3.4 Flow-based models

In the context of density modeling, several universal pdf approximators, such as VAEs and EBMs, have been proposed. However, the difficulties in training them have been major obstacles to unlocking their full potential. Regarding to seek for more powerful density models while maintaining exact likelihood computations, Kingma et al. (2016); Rezende & Mohamed (2015); Dinh et al. (2016) propose a novel method, called *normalizing flows (NFs)*, that uses bijections to transform a simple distribution to represent a complex distribution.

More specifically, suppose a function $\phi : \mathbb{R}^d \to \mathbb{R}^d$ is *invertible*, *a.k.a. bijective*. Let Z be a random variable that follows a density $p(z)$, and finally we define $X = \phi(Z)$. Then, due to *change-of-variable formula*, the likelihood $p(x)$ is written as

$$\begin{aligned} p(x) &= p(\phi^{-1}(x)) \left| \frac{\mathrm{d}\phi^{-1}}{\mathrm{d}x} \right| \\ \Rightarrow \quad \log p(x) &= \log p(\phi^{-1}(x)) + \log \left| \frac{\mathrm{d}\phi^{-1}}{\mathrm{d}x} \right| \end{aligned} \tag{2.4}$$

While it requires a rigorous proof, Equation 2.4 can be interpreted in a simpler fashion. With slight abuse of the notation,

[2] In a VAE, $p(x|z)$ is referred to as the likelihood function or model. However, to avoid confusion with the general term "likelihood," we will refer to it as the "decoder" throughout this tutorial.

the equation is rewritten as

$$p(x) = p(z) \left| \frac{\mathrm{d}z}{\mathrm{d}x} \right|_{z=\phi^{-1}(x)} \quad \Rightarrow \quad \frac{\mathrm{d}\mu}{\mathrm{d}x} = \frac{\mathrm{d}\mu}{\mathrm{d}z} \left| \frac{\mathrm{d}z}{\mathrm{d}x} \right|_{z=\phi^{-1}(x)},$$

where X and Z follow the same measure, as X is defined with an one-to-one mapping from Z. Thus, both are on the same probability space $(\mathbb{R}^d, \mathcal{B}(\mathbb{R}^d), \mu)$, but have different representations (see the definition of random variables in Section A.2). Moreover, a pdf is the derivative of a measure with respect to the uniform volume. The change-of-variable formula states that a density with respect to a new volume is the density of the original measure weighted by the rate of the volumetric change defined by the bijection ϕ.

The most important premise in the above argument is that ϕ is bijection. Under this assumption, it is possible to form powerful density models with exact likelihood computations. On the other hand, however, it is not trivial to design a multivariate bijection whose log-determinant of Jacobian is easily computed. We will further discuss about normalizing flows in Section 4.

2.3.5 Neural ODEs

Chen et al. (2018) proposes a new family of flow-based generative models, called *neural ordinary differential equations (neural ODEs)*. A neural ODE defines an invertible transformation as a solution of an *ordinary differential equation (ODE)*, in which a deep neural network model substitutes the first-order derivative in the differential equation. The resulting transformation can be treated as an infinite-depth normalizing flow, significantly enhancing its expressivity compared to finite-depth normalizing flows.

To formally describe neural ODEs, let us consider an initial value problem of an ODE

$$\mathrm{d}x_t = f(x_t, t)\,\mathrm{d}t \qquad \text{for } t \in [0, 1], \tag{2.5}$$

where the initial value x_0 and a function $f : \mathbb{R}^d \to \mathbb{R}^d$ are given. Then, its solution is written as

$$x_t = x_0 + \int_0^t f(x_s, s)\,\mathrm{d}s =: \phi(x_0, t). \tag{2.6}$$

Under mild assumption on f, the solution x_t of Equation 2.6 exists and is unique (Chen et al., 2018). Thanks to the uniqueness, $\phi(\cdot, t)$ is invertible with respect to the first argument.

To define a generative model by using the above, we define the initial condition, *i.e.* the distribution of X_0, and f for all t. From the resulting stochastic process $\{X_t\}_{t\in[0,1]}$, where $X_t = \phi(X_0, t)$, X_1 becomes the model's random variable, which will learn the data distribution. Since $\phi(\cdot, t)$ are invertible and defined by the integration, they can be treated as infinite-depth normalizing flows.

To compute the model's likelihood, however, one may need extra caution since the direct computation of a $\phi(\cdot, t)$'s Jacobian is intractable in general. Instead, Chen et al. (2018) proposes *instantaneous change of variable formula* that computes the rate of change of the log-densities,

$$\frac{\partial \log p(x_t)}{\partial t} = -\mathrm{tr}\left(\frac{\mathrm{d}f}{\mathrm{d}x_t}\right). \tag{2.7}$$

Therefore, we can compute likelihood of the model for a given x_1 by solving the above differential equation and its

solution is written as

$$\begin{aligned}\log p_\theta(x_1) &= \log p(x_0) - \int_0^1 \mathrm{tr}\left(\frac{\mathrm{d}f(x_s, s)}{\mathrm{d}x_s}\right) ds \\ &= \log p(x_0) + \int_1^0 \mathrm{tr}\left(\frac{\mathrm{d}f(x_s, s)}{\mathrm{d}x_s}\right) ds. \end{aligned} \tag{2.8}$$

The integral in RHS can be computed by using a black-box differential equation solver starting from x_1.

Compared to the previous normalizing flows, neural ODEs significantly advance strategies to construct expressive bijections, since constraints on f is minimal . On the contrary, their log-likelihoods can not be computed exactly. They can only be approximated by solving the ODEs, whose number of discretization steps can be increased to a few hundred for good solutions. Moreover, computing the trace of the f's Jacobian is computationally expensive for arbitrary f, especially for high-dimensional data. We will continue the discussion on neural ODEs, including its training and other advanced topics, in Section 4.

2.3.6 Implicit distributions

Density models have been one of the dominant approaches in generative models since the maximum likelihood estimation (MLE) has been studied throughout the long history of random variable modeling. The MLE inevitably requires access to the likelihoods. However, by sacrificing the accessibility of the likelihoods, we can construct more expressive random variables. For example, let Z be a random variable that follows a density $p(z)$. By defining a data generation process $\phi : \mathbb{R}^d \to \mathbb{R}^d$, we can define a random variable $X = \phi(Z)$. Here, ϕ doesn't need to be bijective. This random variable induces a distribution, but we commonly don't have access to its density. Such a model is called a *implicit density model*, and the induced distribution is called the *implicit distribution* (Mohamed & Lakshminarayanan, 2016).

For implicit density models, we can fully exploit the expressive power of the deep neural networks to construct very complex distributions. One of the most popular examples of implicit density models is *generative adversarial nets (GANs, Goodfellow et al., 2014a)*. Since its initial work, GAN-based models have been the most powerful family of models in high-resolution image generation benchmark datasets Karras et al. (2019). However, since we cannot evaluate the likelihoods of implicit distributions, we need other ways to train the models. The training of implicit distributions is deferred to Section 2.4 since this section focuses on the representation of random variables.

2.3.7 Wrapping-up

In this section, we discussed how various generative models represent random variables. We could either directly define probability densities or transform simple random variables by using some mappings. These mappings could be bijective or not. Various problems emerged depending on the representation of random variables. In particular, due to the choices to improve the expressivity of random variables, some models had to approximate likelihoods, while others couldn't compute likelihoods at all.

In addition to the methods mentioned here, there are numerous other methods to represent random variables. We omitted explanations of score functions and diffusion-based models here. Score functions will be discussed in Section 5.2, while diffusion-based generative models will be covered in Section 6.3. In the next section, we will discuss the training of random variable models.

2.4 Comparing distributions

The generative modeling aims to train a random variable model to mimic the random variable of data. In other words, we are seeking for those two random variables' internal representations, *i.e.* their distributions, to become

equal. To accomplish this goal, the generative modeling performs minimizing distance-like quantities between two random variables, often called *statistical distances*. In particular, we would like to achieve this minimization under the assumption that only data samples are available without knowing their internal information.

In this regard, it is important to understand how we can determine the closeness of those two distributions. Interestingly, the choice of statistical distances can change the properties of trained models even if the same models are used. On the other hand, the choices of the models may restrict available distances or such quantities. Therefore, understanding statistical distances is very important for generative modeling in general. In this section, we briefly introduce two major categories of the statistical distances and list up a few popular examples.

2.4.1 Metrics vs. divergences

Let $(\Omega, \mathcal{F})$ is a measurable space. Assume we write $\mathcal{P}$ for the the space of probability measures on $(\Omega, \mathcal{F})$. A function $d : \mathcal{P} \times \mathcal{P} \to \mathbb{R}_{\geq 0}$ is called *metric* (also called *distance function* or simply *distance*) if it satisfies the following conditions for all $\mu_x, \mu_y, \mu_z \in \mathcal{P}$:

(i) $d(\mu_x, \mu_y) \geq 0$ (non-negativity).

(ii) $d(\mu_x, \mu_y) = 0$ if and only if $x = y$.

(iii) $d(\mu_x, \mu_y) = d(\mu_y, \mu_x)$ (symmetry).

(iv) $d(\mu_x, \mu_z) \leq d(\mu_x, \mu_y) + d(\mu_y, \mu_z)$ (subadditivity / triangle inequality).

On the other hand, statistical distances that satisfy only (i) and (ii) are referred to as *divergences*.

2.4.2 Kullback–Leibler divergence

The Kullback–Leibler divergence is one of the most popular statistical distances in the context of generative modeling. Again, let $(\Omega, \mathcal{F})$ be a measurable space. Suppose X and Y be random variables on $(\Omega, \mathcal{F})$, and assume their distributions are denoted by $\mathbb{P}$ and $\mathbb{Q}$, respectively. When X and Y are $\mathbb{R}^d$-valued, then they have densities; for their pdfs, we write $p(x)$ and $q(y)$, respectively.

Then *Kullback–Leibler (KL) divergence* is defined by that for $\mathbb{Q} \ll \mathbb{P}$,

$$D_{\mathrm{KL}}(\mathbb{P}\|\mathbb{Q}) \coloneqq \int \log\left(\frac{\mathrm{d}\mathbb{P}}{\mathrm{d}\mathbb{Q}}\right) \mathrm{d}\mathbb{P}. \tag{2.9}$$

We may also use the following equivalent notations for the RHS:

$$\int \log\left(\frac{\mathrm{d}\mathbb{P}}{\mathrm{d}\mathbb{Q}}\right) \mathrm{d}\mathbb{P} = \int \log\left(\frac{\mathrm{d}\mathbb{P}}{\mathrm{d}\mathbb{Q}}(x)\right) \mathrm{d}\mathbb{P}(x) = \mathop{\mathbb{E}}_{X\sim\mathbb{P}}\left[\log\left(\frac{\mathrm{d}\mathbb{P}}{\mathrm{d}\mathbb{Q}}(X)\right)\right].$$

Moreover, X and Y are equal in distribution if and only if $D_{\mathrm{KL}}(\mathbb{P}\|\mathbb{Q}) = 0$. When X and Y are $\mathbb{R}^d$-valued, we have

$$d\mathbb{P} = \mathbb{P}(\mathrm{d}x) = \frac{\mathrm{d}\mathbb{P}}{\mathrm{d}x}\,\mathrm{d}x = p\,\mathrm{d}x \ \left(= p(x)\,\mathrm{d}x\right) \quad \text{and} \quad \frac{\mathrm{d}\mathbb{P}}{\mathrm{d}\mathbb{Q}} = \frac{\mathrm{d}\mathbb{P}/\,\mathrm{d}x}{\mathrm{d}\mathbb{Q}/\,\mathrm{d}x} = \frac{p}{q} \ \left(= \frac{p(x)}{q(x)}\right).$$

For that, we will also write

$$D_{\mathrm{KL}}(\mathbb{P}\|\mathbb{Q}) = \int \log\left(\frac{p(x)}{q(x)}\right) p(x)\,\mathrm{d}x = \mathop{\mathbb{E}}_{X\sim p}\Big[\log p(X) - \log q(X)\Big] =: D_{\mathrm{KL}}(p(x)\|q(x)). \tag{2.10}$$

We will often use $D_{\mathrm{KL}}(p\|q)$ instead of $D_{\mathrm{KL}}(p(x)\|q(x))$.

Note that when $\mathbb{Q}$ is a model distribution, and $\mathbb{P}$ is the data distribution of interest, the KL divergence can be estimated via the Monte Carlo (MC) estimates. Let $x^{(i)} \overset{\text{i.i.d.}}{\sim} \mathbb{P}$ for $i = 1, \ldots, n$. The MC estimate of the KL is written

as

$$\int \log\left(\frac{\mathrm{d}\mathbb{P}}{\mathrm{d}\mathbb{Q}}\right)\,\mathrm{d}\mathbb{P} = \frac{1}{n}\sum_{i=1}^{n}\log\left(\frac{\mathrm{d}\mathbb{P}}{\mathrm{d}\mathbb{Q}}(x^{(i)})\right). \tag{2.11}$$

Moreover, when the densities are available, we have

$$\int \log\left(\frac{p(x)}{q(x)}\right)p(x)\,\mathrm{d}x = \frac{1}{n}\sum_{i}^{n}\Big[\underbrace{\log p(x^{(i)})}_{q\text{-independent}} - \log q(x^{(i)})\Big] \tag{2.12}$$

In particular, minimizing the RHS with respect to q doesn't require to compute $\log p(x)$.

As we can infer from its name—not being a metric—the KL divergence is not symmetric. In this regard, we often use different names depending on the use case. For the previous case where $\mathbb{Q}$ is a model distribution and $\mathbb{P}$ is a data distribution, we call it the *forward-KL*. On the contrary, we can consider the opposite. When $\mathbb{P}$ is a model and $\mathbb{Q}$ is a data, we are required to sample from the model, and we often call it *reverse-KL*; in this case, we are required to evaluate $\log p(x)$.

MLE is to minimize forward KL In Section 2.3, we introduced models classified as density models, such as autoregressive models, VAEs, energy-based models, and flow-based models. These models are commonly trained by using a method known as maximum likelihood estimation (MLE).

To do that, let us denote the data's probability density function (pdf) as p and our model density as q. Given an input x, its likelihood (under the model) is represented as $q(x)$. The log-likelihood of x under q is then written as $\log q(x)$. The MLE objective is to maximize the expectation of the log-likelihoods with respect to the data distribution, which is defined as $\mathcal{L}_{\mathrm{MLE}}$:

$$\mathcal{L}_{\mathrm{MLE}}(p) \coloneqq \mathop{\mathbb{E}}_{X\sim p}\left[\log q(X)\right].$$

Minimizing Equation 2.10, after removing the q-independent term, is equivalent to maximizing the expected log-likelihoods. In this sense, training with MLE is equivalent to minimizing the forward KL divergence.

2.4.3 Fisher divergence

In the context of machine learning, the Fisher divergence is first introduced in the context of score matching by Hyvärinen & Dayan (2005). This divergence has also been widely used in the information theory (Cover, 1999; Johnson, 2004; DasGupta, 2008).

Assume $\mathbb{Q} \ll \mathbb{P}$ and $\frac{d\mathbb{P}}{d\mathbb{Q}}(x)$ is continuous and differentiable at all $x \in \Omega$. Then, the *Fisher divergence* of $\mathbb{Q}$ with respect to $\mathbb{P}$ is defined by

$$D_{\textit{Fisher}}\left(\mathbb{P}\|\mathbb{Q}\right) \coloneqq \int \left\|\nabla \log\left(\frac{\mathrm{d}\mathbb{P}}{\mathrm{d}\mathbb{Q}}\right)\right\|^2 \mathrm{d}\mathbb{P}. \tag{2.13}$$

In particular, when the densities are available, we write

$$\begin{aligned} D_{\textit{Fisher}}\left(\mathbb{P}\|\mathbb{Q}\right) &= \int \left\|\nabla \log\left(\frac{\mathrm{d}\mathbb{P}}{\mathrm{d}x}\Big/\frac{\mathrm{d}\mathbb{Q}}{\mathrm{d}x}\right)\right\|^2 \mathrm{d}\mathbb{P} \\ &= \int \left\|\nabla \log p(x) - \nabla \log q(x)\right\|^2 p(x)\,\mathrm{d}x. \end{aligned} \tag{2.14}$$

Similarly to the KL divergence, when $\mathbb{P}$ is the data distribution of interest, we can perform the Monte Carlo (MC) estimates. However, unlike the KL divergence, this requires evaluating the log densities of both data and model. In

order to address this issue, several unbiased estimators have been proposed in the context of score matching methods. The development of such unbiased estimators also plays an important role in training diffusion-based generative models (and others). We will continue the detailed discussions on several score matching methods in Section 5.

2.4.4 Integral probability metrics and f-divergences

The previous divergences require the models' densities. However, we cannot explicitly evaluate the densities of some generative models, such as GANs. Which statistical distances are used for such models?

One of the common statistical distance in the context of the GANs are the integral probability metrics. Let $\mathcal{T}$ be a set of real-valued functions on Ω. In particular, assume all $g \in \mathcal{T}$ is $\mathbb{P}$ and $\mathbb{Q}$-integrable. Then, the *integral probability metrics (IPMs)* are defined by

$$D_{\mathcal{T}}\left(\mathbb{P} \| \mathbb{Q}\right) = \sup_{g \in \mathcal{T}} \left| \int g(x) \, \mathrm{d}\mathbb{P} - \int g(y) \, \mathrm{d}\mathbb{Q} \right|. \tag{2.15}$$

Interestingly, depending on the choice of $\mathcal{T}$, the IPM corresponds to a statistical distance. For example, when $\mathcal{T}$ is the set of 1-Lipschitz functions on Ω, then the IPM becomes the dual representation of the *Wasserstein-1 distance*–also called *earth mover distance* (Wikipedia, 2024). When the densities are available, we write

$$D_{\mathcal{T}}\left(\mathbb{P} \| \mathbb{Q}\right) = \sup_{g \in \mathcal{T}} \left| \int g(x) p(x) \, \mathrm{d}x - \int g(y) q(y) \, \mathrm{d}y \right|. \tag{2.16}$$

The MC estimates of the IPM is written as

$$\sup_{g \in \mathcal{T}} \left| \int g(x) \, \mathrm{d}\mathbb{P} - \int g(y) \, \mathrm{d}\mathbb{Q} \right| = \sup_{g \in \mathcal{T}} \left| \frac{1}{n} \sum_{i=1}^{n} g(x^{(i)}) - \frac{1}{m} \sum_{j=1}^{m} g(y^{(j)}) \right|, \tag{2.17}$$

where $x^{(i)} \overset{\text{i.i.d.}}{\sim} \mathbb{P}$ for $i = 1, \ldots, n$ and $y^{(j)} \overset{\text{i.i.d.}}{\sim} \mathbb{Q}$ for $j = 1, \ldots, m$.

In practice, one defines a trainable model for g and constraints it to be in $\mathcal{T}$. Thus, for every update step of the $\mathbb{Q}$-generative models, one needs to solve the internal optimization for $x^{(i)}$ and $y^{(j)}$s.

Interestingly, the KL-divergence is one of the f-divergences, and the f-divergences can also be represented similarly to the IPMs, more specifically, their variational forms. The *f-divergences* are defined by that for $\mathbb{Q} \ll \mathbb{P}$, we write

$$D_f\left(\mathbb{P} \| \mathbb{Q}\right) \coloneqq \int f\left(\frac{\mathrm{d}\mathbb{P}}{\mathrm{d}\mathbb{Q}}\right) \mathrm{d}\mathbb{P}, \tag{2.18}$$

where f is a lower-semicontinuous function satisfying $f(1) = 0$ and $\frac{\mathrm{d}\mathbb{P}}{\mathrm{d}\mathbb{Q}}(x)$ is continuous and differentiable at all $x \in \Omega$. When $f(x) = x \log x$, the f-divergence becomes the KL.

The variational form of a f-divergence in Equation 2.18 is written as

$$D_f\left(\mathbb{P} \| \mathbb{Q}\right) = \sup_{g \in \mathcal{T}} \int g(x) \, \mathrm{d}\mathbb{P} - \int f^*(g(y)) \, \mathrm{d}\mathbb{Q}, \tag{2.19}$$

where f^* is the convex conjugate of the convex function f and $\mathcal{T}$ is an arbitrary set of functions $g : \Omega \to \mathbb{R}$ (Nowozin et al., 2016). Thus, when the densities are available

$$D_f\left(\mathbb{P} \| \mathbb{Q}\right) = \sup_{g \in \mathcal{T}} \int g(x) p(x) \, \mathrm{d}x - \int f^*(g(y)) q(y) \, \mathrm{d}y. \tag{2.20}$$

2.4.5 Wrapping up

In this section, we have briefly introduced a few statistical distances that are closely related to the articles presented in the tutorial. Besides the distances mentioned here, there are other important statistical distances, such as the total variation distance. In addition, there are numerous discussions about differences between the statistical distances as well as their trade-offs. For example, the rate at which the models converge can vary depending on the choice of the distances. The errors or the variances of estimators can also vary with the choice of a distance, even for the same number of samples. However, this tutorial will not cover these aspects.

2.5 Key design factors for generative modeling

So far in the previous sections, we learned about the components that constitute generative models. The outcomes of generative modeling can vary depending on how these components are chosen. Then, on what basis should we choose these components? While there could be numerous reasons for each choice, this tutorial highlights the following factors as the most important ones:

(i) Expressivity.

(ii) Training bias.

(iii) Training stability.

While we concentrate on these three factors, a broader range of design considerations remains highly significant, including computational efficiency (*i.e.*, accelerating sampling at test time) (Song et al., 2021; Salimans & Ho, 2022), adversarial robustness (Szegedy et al., 2013; Goodfellow et al., 2014b), and equivariance & invariance (Köhler et al., 2020; Satorras et al., 2021). Nevertheless, our focus on expressivity, bias, and dynamics is motivated by the fact that these criteria are often amenable to analytical and theoretical assessment, making them especially suitable for the perspective developed in this chapter.

2.5.1 Expressivity

One of the most fundamental questions in designing generative models is whether the chosen model class is expressive enough to represent the data distribution accurately. Training in generative modeling typically relies only on data samples, without access to the true distribution. If the model family is inherently too limited, it may fail to capture the target distribution even with unlimited data and perfect optimization. In such cases, the failure is not due to insufficient training but to the lack of expressivity of the model itself. In short, expressivity refers to whether a model class is capable, at least in principle, of representing the target data distribution.

In this subsection, we first look at a motivating example that highlights the limitations of simple models. We then introduce the concept of universal density approximators, illustrated by infinite Gaussian mixture models, and conclude with a brief reflection on why expressivity alone is not sufficient without effective training methods.

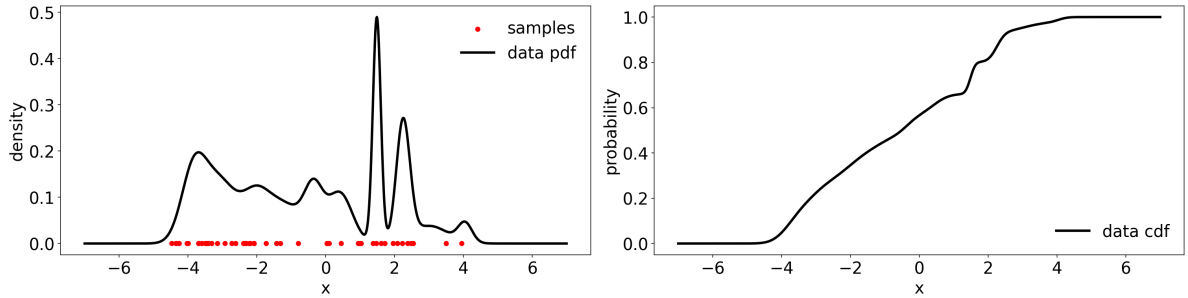


Figure 2.2: Example of one-dimensional data distribution. (Left) probability density function (pdf) and samples from the distribution. (Right) cumulative density function (cdf).

Example problem Consider a one-dimensional data distribution with a highly intricate density (Figure 2.2). Suppose we attempt to model this distribution using the Gaussian family. Even after optimization, the fitted Gaussian may capture the mean and variance but fails to reproduce the complex structure of the data. Figure 2.3 shows this mismatch by comparing the probability density function (pdf) and the cumulative density function (cdf) of the data with those of the fitted model. This mismatch illustrates a fundamental limitation: the Gaussian family lacks the expressivity needed to approximate the data distribution. No amount of additional data or training effort can overcome this gap.

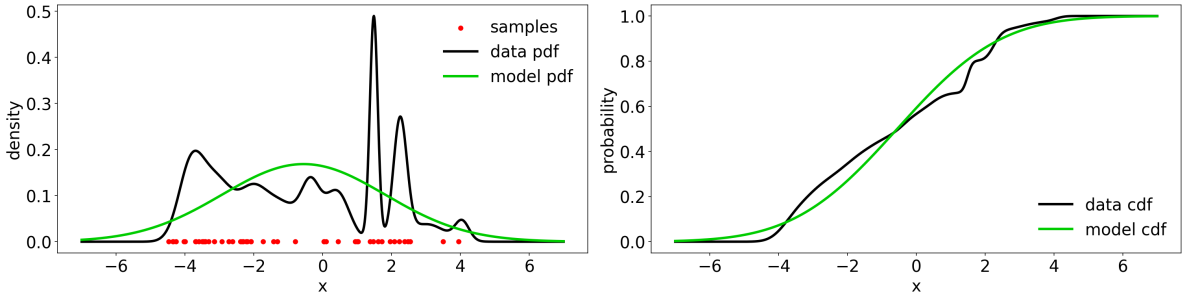


Figure 2.3: Normal distribution fitted to the samples from the example data data distribution. The data samples and the pdfs are plotted in the left, and the cdfs are plotted in the right.

This example highlights the importance of expressivity: the ability of a model to approximate complex distributions. More expressive models are often informally described as more powerful. If a model lacks sufficient expressivity, it risks oversimplifying the data, leading to underfitting regardless of the sample size or training procedure. Developing expressive models is therefore not merely a matter of technical sophistication but a prerequisite for advancing generative modeling.

Universal function approximators Moving beyond intuition, how can we rigorously characterize the expressivity of a model class? Unlike bias or variance, expressivity is not easily quantified by a single numerical measure. Instead, a more precise formulation is to ask whether a model class can, in theory, approximate any target distribution under certain conditions. If so, it is referred to as a *universal density approximators* in the context of density modeling.[3]

In Euclidean space, a canonical example of a universal density approximator is the infinite Gaussian mixture model (IGMM). A Gaussian mixture model (GMM) represents a distribution as a mixture of Gaussian components, while an IGMM extends this idea by allowing the number of components to be infinite (Figure 2.4).

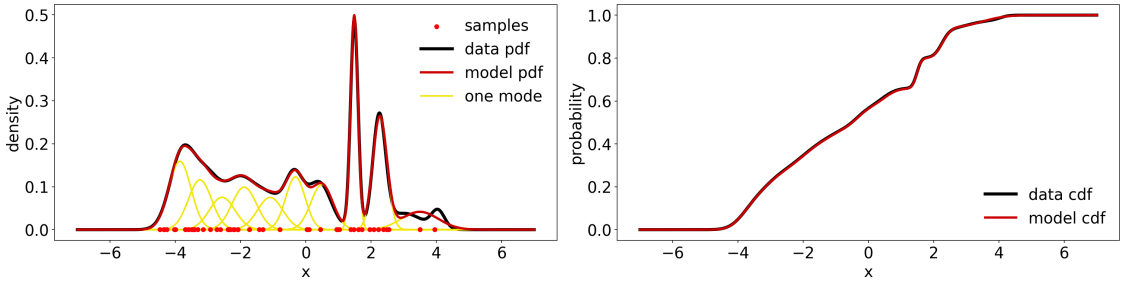


Figure 2.4: Gaussian mixture model (GMM) fitted to the samples from the example data data distribution (# of mixtures $= 10$). The data samples and the pdfs are plotted in the left, and the cdfs are plotted in the right.

By combining sufficiently many Gaussian components, an IGMM can approximate any continuous distribution on the real line to arbitrary accuracy. Figure 2.5 shows how we can approximate a cdf via the cdf of the Gaussian mixture models. Formally, we can state the following proposition.

[3] In a broader context, if a model class can approximate any target function, it is referred to as a *universal function approximator. Informally, such models are often simply called universal approximators.*

Proposition 2.1. [IGMMs are universal density approximators] *Let $f : \mathbb{R} \to [0, 1]$ be a cumulative density function (cdf) of a random variable X on $\mathbb{R}$. Thus, f is a non-decreasing function such that $\lim_{x\to-\infty} f(x) = 0$ and $\lim_{x\to\infty} f(x) = 1$. We further assume that f is continuous and L-Lipschitz (for some $L > 0$).*

Suppose $S_n : \mathbb{R} \to [0, 1]$ be the cdf of a mixture of n-Gaussians; that is,

$$S_n(x) = \sum_{i=1}^{n} w_i \, \Phi\left(\frac{1}{\sigma_i}(x - \mu_i)\right), \tag{2.21}$$

where $\{(\mu_i, \sigma_i)\}_{i=1,\dots,n}$ are the mean and variance pairs of the Gaussian components, $\sum_i w_i = 1$, and $\Phi(x)$ is the cdf of the standard normal distribution defined by

$$\Phi(x) = \frac{1}{2}\left[1 + \operatorname{erf}\left(\frac{x}{\sqrt{2}}\right)\right] \quad \textit{and} \quad \operatorname{erf}(x) = \frac{2}{\sqrt{\pi}} \int_0^x e^{-t^2} \, \mathrm{d}t.$$

Then, for any $\varepsilon > 0$, there exists n and $\{w_i, b_i, c_i\}_{i=1\dots n}$ such that $|f(x) - S_n(x)| < \varepsilon$ for all $x \in (-\infty, \infty)$.

This proposition shows that S_n approximates f arbitrarily closely for a sufficiently large n. As $n \to \infty$, the Gaussian mixtures model reduces to an infinite mixture, which we refer to as an IGMM; thus, we conclude that IGMMs are universal density approximators. One could make this intuition even more precise by rewriting the approximation bound explicitly as a function of n, thereby showing the error vanishes as n grows, but we will not pursue that here. The following is the proof of the proposition.

Proof. We adapt the proof of the universality of neural sigmoidal flows from Huang et al. (2018).[4] We would like to show that for any $\varepsilon > 0$, we can find $n \in \mathbb{N}$ such that

$$|f(x) - S_n(x)| \;\leq\; \varepsilon \quad \text{for } x \in \mathbb{R}.$$

The proof can be sketched as follows. We first show that any non-decreasing function can be approximated by a sum of step functions $S_n^*(x)$. We then show that the mixture of step functions, in turn, can be approximated by a mixture of sigmoidal functions; here, the sigmoidal corresponds to a Gaussian cdf in our problem of interest. We then use the triangle inequality:

$$|f(x) - S_n(x)| \;\leq\; |f(x) - S_n^*(x)| + |S_n^*(x) - S_n(x)|.$$

By doing so, we will prove that by combining enough Gaussian cdfs, the resulting mixture cdf converges to the target distribution's cdf arbitrary closely. This establishes IGMMs as universal density approximators.

[4] Similar proofs of universality can be found in various references; see, for instance, this lecture note.

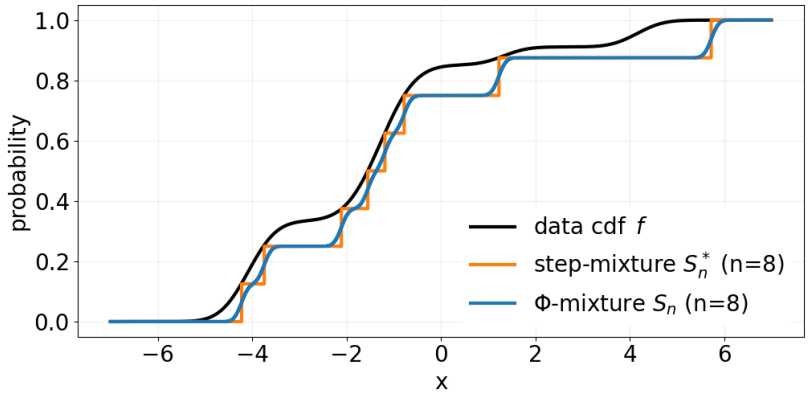


Figure 2.5: Approximating a cdf f by a mixture of step functions S_n^* and a mixture of sigmoidal functions S_n ($n = 8$). The approximation accuracy increase as n increases.

Step 1: Approximate f by a mixture of steps S_n^* We first define a staircase function (step function) as in

$$s(x) \coloneqq \mathbb{1}_{\{x \geq 0\}}(x) = \begin{cases} 1 & \text{if } x \geq 0, \\ 0 & \text{otherwise.} \end{cases}$$

Fix $n \in \mathbb{N}$, and let $y_0, y_1, \ldots, y_{n-1}, y_n$ be evenly spaced between $[0, 1]$: that is,

$$y_i = \frac{i}{n} \quad \text{for } i = 1, \ldots, n,$$

and $y_0 = 0$. We then define t_i for $i = 1, \ldots, n$ as

$$t_i \coloneqq \inf\{x \in \mathbb{R} : f(x) \geq y_i\} \quad \text{for } i = 1, 2, \ldots, n.$$

With these, we can now define a mixture of step functions as in

$$S_n^*(x) = \sum_{i=1}^{n} w_i \, s(x - t_i),$$

where $w_i = 1/n$ (note that $\sum_i w_i = 1$), so the function jumps at t_is.

Then, we can show that the pointwise error between f and S^*n is bounded by $1/n$: that is

$$\begin{aligned} |f(x) - S_n^*(x)| &= \left| f(x) - \sum_i w_i \, s(x - t_i) \right| \\ &= \left| y_{i-1} - f(x) \right| \quad \text{for } t_{i-1} < x \leq t_i, \;\; i = 1, \ldots, n \\ &\leq \left| y_i - y_{i-1} \right| \quad \text{for } t_{i-1} < x \leq t_i, \;\; i = 1, \ldots, n \\ &= \frac{1}{n} \quad \text{for } x \in \mathbb{R}. \end{aligned}$$

Step 2: Approximate S_n^* by a mixture of the erf's S_n For the simplicity of the proof, we set w_i's and t_i's of S_n to be the same with S_n^*'s, and we further assume that $\sigma_i = \sigma$ for all $i = 1, \ldots, n$ (for some $\sigma > 0$).

Fix parameter $r > 0$ and $\sigma > 0$. For $x \in \mathbb{R}$, we define an index set A_x as in

$$A_x \coloneqq \{i \in \{1, \ldots, n\} : |x - t_i| \leq r\}.$$

With this, we can bound the pointwise error between a step function and the Φ, which will be used to bound the error

between S_n and S_n^*.

For $i \notin A_x$, we have

$$\left| s(x - t_i) - \Phi\left(\frac{x - t_i}{\sigma}\right)\right| \;\le\; \delta(\sigma, r) \quad \text{with} \quad \delta(\sigma, r) \coloneqq \Phi\left(-\frac{r}{\sigma}\right),$$

since either $x \le t_i - r$ (close to zero) or $x \ge t_i + r$ (close to 1).

For $i \in A_x$, the worst-case pointwise cap between a step function and the Φ is at most $1/2$ per step; that is

$$\left| s(x - t_i) - \Phi\left(\frac{x - t_i}{\sigma}\right)\right| \;\le\; \frac{1}{2}.$$

Therefore, with weights $w_i = 1/n$,

$$|S_n^*(x) - S_n(x)| \;\le\; \frac{1}{n}\sum_{i \in A_x} \frac{1}{2} + \frac{1}{n}\sum_{i \notin A_x} \delta(\sigma, r) \;\le\; \frac{1}{2n}|A_x| + \delta(\sigma, r), \tag{2.22}$$

where $|A_x| = \sum_{i \in A_x} 1$ is the cardinality of the set A_x. Here, the second inequality comes from

$$\frac{1}{n}\sum_{i \notin A_x} \delta(\sigma, r) \le \delta(\sigma, r),$$

since the number of t_is that are not in A_x are less than equal to n.

In addition, $|A_x|$ can be further bounded by using Lipschitzness of f. Since each t_i increases f by exactly $1/n$, the number of t_is in $[x - r, x + r]$ can be computed through dividing $f(x + r) - f(x - r)$ by $1/n$ with at most one additional point if the jumps are located at $x - r$ and $x + r$. Thus, we write

$$|A_x| \;\le\; n\underbrace{\big(f(x + r) - f(x - r)\big)}_{\le 2rL} + 1 \;\le\; n2rL + 1.$$

Plugging this into Equation 2.22, we get

$$|S_n^*(x) - S_n(x)| \;\le\; \frac{n2rL + 1}{2n} + \delta(\sigma, r) \;=\; Lr + \frac{1}{2n} + \delta(\sigma, r). \tag{2.23}$$

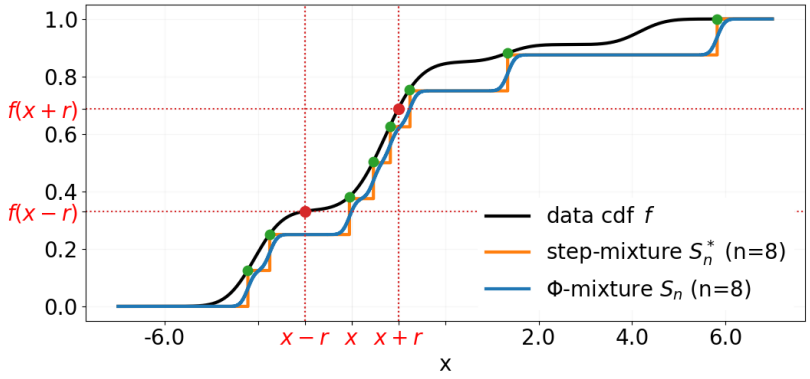


Figure 2.6: Bounding $|A_x|$ with Lipschitzness of f. The number of indices $|A_x|$ is the number of green dots insides the box defined by the two red dots.

Step 3: Combine and choose parameters Given $\varepsilon > 0$, we set

$$n \geq \left\lceil \frac{9}{2\varepsilon} \right\rceil, \quad r = \frac{\varepsilon}{3L}, \quad \delta \coloneqq \frac{\varepsilon}{3},$$

and then choose σ such that $\Phi(-r/\sigma) \leq \delta$ (equivalently, $r/\sigma \geq \Phi^{-1}(1-\delta)$). With this choice, we get

$$|f(x) - S_n(x)| \;\leq\; |f(x) - S_n^*(x)| + |S_n^*(x) - S_n(x)| \;\leq\; \frac{3}{2n} + Lr + \delta(\sigma, r) \;\leq\; \frac{\varepsilon}{3} + \frac{\varepsilon}{3} + \frac{\varepsilon}{3} = \varepsilon.$$

This completes the proof.[5] □

This example illustrates how one can rigorously establish the expressive power of a model class. However, IGMMs still contain a practical limitation: training an IGMM requires optimizing infinitely many means and variances, which is infeasible in practice. Thus, their universality provides only a theoretical guarantee, and its practical realization depends on training considerations—a limitation that applies, in various forms, to most other expressive model classes as well.

Wrapping up In this section, we explored the notion of expressivity. Expressivity captures whether a model class has the capacity, at least in principle, to represent arbitrary distributions. We examined a proof sketch showing that IGMMs are universal density approximators. Beyond this example, however, we will see throughout the tutorial that many advances in generative modeling have been driven by the pursuit of greater expressivity.

While this property is essential, it alone does not guarantee successful generative modeling. At its core, generative modeling requires adjusting the model's parameters—*i.e.*, training—using data samples. As illustrated in the IGMMs' case, training of such expressive models can be far from straightforward.

This observation naturally leads to the next two design factors. Even when a model is expressive enough, its practical effectiveness depends on whether the training objective introduces bias (Section 2.5.2) and whether optimization remains stable under finite samples (Section 2.5.3). Together, these considerations determine whether a model's expressive potential can be realized in practice.

2.5.2 Bias in training objectives

In generative modeling, training is often formulated as minimizing a statistical distance between the model and data distributions.[6] However, the analytical solution for this distance function is frequently intractable or even entirely inaccessible.

As a result, it is common to employ auxiliary losses instead of optimizing the original objective function or express the original in a variational form.[7] These approximations inherently mean that the model is not trained directly on the exact quantity we seek to optimize. Consequently, they often lead to suboptimal outcomes—*i.e.*, the original target distance may not converge to zero, even if the model class is universal.

In this section, we will delve into this topic through two representative example scenarios from VAEs and GANs. We begin by introducing an example of training VAEs, where its typical maximum-likelihood objective is often intractable and alternative objectives are required. We then briefly discuss the implications of training with one alternative objective, called evidence lower bound (ELBO), with a particular focus on the training's suboptimality. Here, we provide only a high-level overview—specific challenges and potential solutions will be covered in detail in Section 3. We then turn to GAN training, which similarly relies on an inaccessible distance function, highlighting that training suboptimality is not unique to VAEs. To start, we first consider the case of variational autoencoders (VAEs) and the challenge of training them by maximizing their likelihood.

[5] One may also choose the r and σ for $\delta(\sigma, r)$ in Equation 2.23 in terms of n, and shows that the error goes zero as $n \to \infty$.

[6] Some may describe this as optimizing (minimizing or maximizing) an objective function, depending on how the training problem is formulated.

[7] Note that these approximations are orthogonal to the numerical approximations like Monte Carlo estimate of the integrations, and we will talk about this aspect in Section 2.5.3.

Optimizing auxiliary objective in VAEs Let $(\mathbb{R}, \mathcal{B}(\mathbb{R}))$ denote a measurable space, and let p_{data} represent the data density on this space. Additionally, let p be the density on $\mathbb{R}$ of a variational autoencoder (VAE) model, as discussed in Section 2.3.2. For $x \in \mathbb{R}$, $p(x)$ is expressed as:

$$p(x) = \int p(x|z)p(z)\,\mathrm{d}z. \tag{2.24}$$

To train the model p, the maximum likelihood estimate (MLE) is a common approach. This is equivalent to minimizing the forward KL divergence, as previously discussed in Section 2.4.2. Under this approach, the model is trained to maximize the following quantity:

$$\mathcal{L}_{\mathrm{MLE}}(p) \coloneqq \mathbb{E}\left[\log p(X)\right]. \tag{2.25}$$

The term $\log p(x)$ is referred to as the log-likelihood of x under the model p.

However, maximizing $\log p(x)$ for a given x (and thereby $\mathcal{L}_{\mathrm{MLE}}(p)$) is not a trivial task, as the integration in the log-likelihood is often analytically intractable. To overcome this difficulty, an alternative objective—sometimes referred to as an *auxiliary objective function*—is constructed and optimized instead. In the case of VAEs, the *Evidence Lower Bound* (*ELBO*), denoted by $\mathcal{E}$, serves as this auxiliary objective. The ELBO is defined as follows:

$$\begin{aligned}
\log p(x) &= \log \int p(x|z)p(z)dz \\
&= \log \int \frac{q(z|x)}{q(z|x)} p(x|z)p(z)\,\mathrm{d}z \\
&\overset{(1)}{\geq} \int \log\left(\frac{p(x|z)p(z)}{q(z|x)}\right) q(z|x)\,\mathrm{d}z \\
&= \mathop{\mathbb{E}}_{q(z|x)}\left[\log p(x|Z) + \log p(Z) - \log q(Z|x)\right] \\
&=: \mathcal{E}(x; p, q),
\end{aligned} \tag{2.26}$$

where (1) follows from Jensen's inequality, and $q(z|x)$ is a proposal distribution specific to x. Importantly, q can be freely chosen as long as $q(z|x) \ll p(z)$, and $q(z|x)$ is not necessarily conditional on x in all cases.

Consequently, the objective that is optimized during training becomes the expected ELBO:

$$\mathcal{L}_{\mathrm{ELBO}}(p, q) \coloneqq \mathop{\mathbb{E}}_{X \sim p_{\mathrm{data}}}\left[\mathcal{E}(X; p, q)\right]. \tag{2.27}$$

The ELBO $\mathcal{E}$ serves as a lower bound on the original log-likelihood. Thus, maximizing $\mathcal{E}$ for each x (and its expected value) guarantees improvement in the original target. This property makes the ELBO the default choice for auxiliary objectives when training VAEs.

However, the p that maximizes the ELBO does not necessarily guarantee to be the maximum of the (expected) log-likelihood. The difference between between $\log p(x)$ and $\mathcal{E}(x; p, q)$ can be written by

$$\log p(x) - \mathcal{E}(x; p, q) = D_{\mathrm{KL}}\big(q(z|x) \big\| p(z|x)\big). \tag{2.28}$$

The difference is often called *variational gap*, and its nonzero unless $q(z|x)$ is exactly equal to the true posterior $p(z|x)$. Consequently, the model optimized with the ELBO is not directly trained on the exact quantity we aim to optimize.

What are the implications of a nonzero gap? This discrepancy raises important questions about the effectiveness of the training process and the limitations it imposes on the model's performance. Let us delve deeper into this issue in the following discussion.

Suboptimality of optimizing ELBOs Let us continue our discussion on VAE training from the preceding section. In the ideal case, *i.e.* with an infinite amount of data and under some mild assumptions, maximizing $\mathcal{L}_{\mathrm{MLE}}(p)$ ensures that the learned model distribution p converges to the true data distribution. This is because a VAE can be viewed as an infinite Gaussian mixture model, which serves as a universal density approximator. Moreover, MLE corresponds to minimizing the KL divergence between the data distribution and the model distribution. Since the model is universal, the optimal solution minimizes the KL divergence to zero, thereby making the model distribution being equivalent to the data distribution.

However, the situation changes when we consider maximizing the expected ELBO. When the gap $\log p(x) - \mathcal{E}(x; p, q)$ is nonzero, the optimal solution for maximizing the expected ELBO does not guarantee the KL divergence being zero.

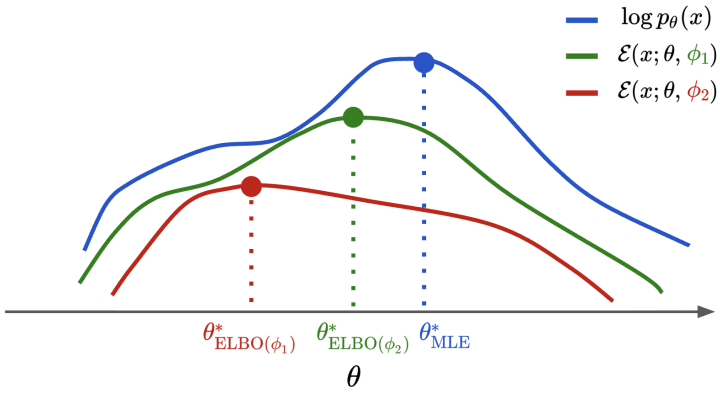


Figure 2.7: Imaginary objective landscape illustrating the variational gap.

Figure 2.7 draws variational gaps along different model parameters θ for fixed posterior parameters ϕ_1 and ϕ_2. The figure illustrates that even if a model is a universal approximator, the learned model via maximizing ELBOs can deviate from the optimal θ^*_{MLE}. This discrepancy occurs particularly when the auxiliary function used for training is not well aligned with the original objective function.

Then, how should we address this issue? There have been important research efforts to address this challenges in the context of VAEs. In Chapter 3, we will delve deeper into this problem and discuss strategies to resolve it.

Interestingly, suboptimality of optimizing the auxiliary objectives are not exclusive to VAEs. Similar challenges also manifest in other generative models, such as GANs. Before ending this subsection, let us turn our attention to this another case.

Optimizing auxiliary objective in GANs In the preceding discussion of VAEs, we saw that training often relies on optimizing an auxiliary objective—the ELBO—rather than the true likelihood, which introduces bias and may lead to suboptimal solutions. A similar situation arises in the training of Generative Adversarial Networks (GANs), though the form of the auxiliary objective differs and, in some respects, is even less well aligned with the original target.

To better understand this, let us first formalize the setup of GANs. Let $(\mathbb{R}, \mathcal{B}(\mathbb{R}))$ be a measurable space, and let p_{data} denote the density of the data distribution on this space. As a generative model, consider an implicit distribution induced by a random variable $X = \varphi(Z)$, where $Z \sim p(z)$ is a latent random variable and $\varphi : \mathbb{R}^d \to \mathbb{R}^d$ is a deterministic transformation parameterized by the generator. We denote the resulting model density by $p(x)$, although in practice it cannot be evaluated explicitly.

In principle, the model should be trained by minimizing a statistical distance between $p(x)$ and $p_{\mathrm{data}}(x)$. For example, one might aim to minimize the forward KL divergence:

$$D_{\mathrm{KL}}(p_{\mathrm{data}} \,\|\, p) = \mathop{\mathbb{E}}_{X \sim p_{\mathrm{data}}} \left[\log \frac{p_{\mathrm{data}}(X)}{p(X)} \right]. \tag{2.29}$$

However, since the likelihood $p(x)$ is intractable for implicit models, this divergence cannot be computed directly. Instead, GANs exploit *variational representations* of divergences, also known as dual forms.[8] For a broad class of f-divergences, one can show:

$$D_f(p_{\text{data}} \,\|\, p) = \int f\left(\frac{p_{\text{data}}(x)}{p(x)}\right) p(x)\,\mathrm{d}x = \sup_{f \in \mathcal{F}} \Big(\mathop{\mathbb{E}}_{X \sim p_{\text{data}}} \big[f(X)\big] - \mathop{\mathbb{E}}_{Y \sim p} \big[f^*(f(Y))\big] \Big), \tag{2.30}$$

where $f : \mathbb{R}_+ \to \mathbb{R}$ is convex with $f(1) = 0$ and f^* is the convex conjugate of f. The supremum is taken over a function class $\mathcal{F}$. When choosing $f(u) = u \log u$, the expression in Equation 2.30 recovers the forward KL divergence discussed earlier (Nowozin et al., 2016). In this case, the convex conjugate is given by $f^*(t) = e^{\,t-1}$.

In practice, the discriminator in GANs serves as the function f, and its parameters are trained to optimize for achieving the supremum of Equation 2.30. Viewed from another angle, the discriminator's role is to distinguish real samples from generated ones, typically by optimizing a loss based on cross-entropy. This connection explicitly characterizes the discriminator's objective under the KL setting—more generally f-divergences—and highlights the auxiliary nature of the GAN objective. Unless the discriminator achieves its optimal solution at every step (*i.e.*, realizes the supremum), the variational form only approximates the true divergence. As a result, the generator is not trained on the exact quantity of interest but rather on a biased surrogate.

In addition, unlike the ELBO in VAEs, this surrogate is not guaranteed to be a valid bound on the target divergence, since the generator minimizes Equation 2.30, whose supremum is rarely attainable in practice. The difficulty in achieving this supremum stems from two sources of error. The first is approximation error: when the discriminator's function class $\mathcal{F}$ is not sufficiently expressive, the variational objective cannot properly match the true divergence. The second is estimation error: even with a rich discriminator class, the discriminator is seldom optimized perfectly, and thus the objective further deviates from the target. These two factors together explain why GAN training is often unstable and why the learned distribution may diverge from the data distribution, even when the generator itself is expressive enough.

Despite these challenges, GANs have proven remarkably effective in practice, particularly for high-dimensional tasks such as image generation. A large body of research has therefore focused on mitigating the bias of their auxiliary objectives and stabilizing the adversarial training process. While these strategies are beyond the scope of this tutorial, they remain an essential theme in the ongoing development of GAN-based generative models.

2.5.3 Instability of training objectives under finite samples

The final topic in this section introduces another crucial criterion for evaluating training methods: **training stability**. In particular, we focus on the errors that arise when expected values—frequently appearing as objective functions or as integral components thereof—are estimated from a finite number of samples.

In Section 2.5.2, we examined how auxiliary objectives may misalign with the true optimization goals. Yet even when the objective is well chosen, we still face the challenge of computing expectations that are rarely tractable in closed form. As a result, we rely on estimators such as Monte Carlo (MC) estimates, which inevitably introduce randomness and variability into the optimization process.

The quality of an estimator can be assessed in several ways. Some estimators are unbiased but differ in their variance, while others may be biased altogether, requiring additional caution in their interpretation. In this subsection, we will first focus on the more common case of unbiased estimators with varying variance, before turning to a contrasting example involving biased estimators.

High-variance estimators can lead to noisy gradient signals, often resulting in unstable training dynamics and suboptimal convergence. In contrast, low-variance estimators enable smoother optimization trajectories and faster convergence. By analyzing the variance of estimators in the context of generative modeling, we aim to clarify how this factor influences the overall performance of a model.

[8] Informally, the term "variational representation" refers to expressing a given quantity in the form of an optimization problem.

To begin, we will examine a simple variance-reduction technique. We then discuss the benefits of such methods and their implications for improving training stability.

Example problem Let $(\Omega, \mathcal{F})$ be a measurable space and X be a random variable on it, denoting its distribution by $\mathbb{P}$. Let $h : \Omega \to \mathbb{R}$ be an integrable function, meaning $\int |h(x)| \, \mathrm{d}\mathbb{P} < \infty$. Consider the problem of estimating a quantity Z, which is written by the expected value of h with respect to $\mathbb{P}$:[9]

$$Z = \mathop{\mathbb{E}}_{X \sim \mathbb{P}} [h(X)] \, . \tag{2.31}$$

In this context, one may call the RHS as an estimator.

In general, the computing the above expectation is intractable, and thus, we will approximate the numerical value by the Monte Carlo (MC) approach:

$$\hat{Z}_n = \frac{1}{n} \sum_{i=1}^{n} h(x^{(i)}), \tag{2.32}$$

where the n-number of i.i.d. observations of X are denoted by $x^{(i)}$ for $i = 1, \ldots, n$. It is well known that $\hat{Z}_n \to Z$ as $n \to \infty$ thanks to the Law of Large Numbers (LLN).

The quality of the MC estimate will depends of the variance of the random variable $h(X)$, where $X \sim \mathbb{P}$, *i.e.* the lower the variance is, the lower number of samples is required to get closer to the ground truth Z. The variance $h(X)$ is written by

$$\mathrm{Var}(h(X)) = \mathop{\mathbb{E}}_{X \sim \mathbb{P}} \left[h(X)^2 \right] - \mathop{\mathbb{E}}_{X \sim \mathbb{P}} [h(X)]^2 = \mathop{\mathbb{E}}_{X \sim \mathbb{P}} \left[h(X)^2 \right] - Z^2.$$

Variance reduction by importance sampling To improve the estimation of Z with lower variance, we often use the importance sampling estimator. For that, suppose we have another distribution $\mathbb{Q}$ satisfying $\mathbb{P} \ll \mathbb{Q}$. Then, instead of Equation 2.31, we take another expectation to estimate Z: that is

$$Z = \mathop{\mathbb{E}}_{X \sim \mathbb{P}} [h(X)] = \int h(x) p(x) dx = \int h(x) \frac{p(x)}{q(x)} q(x) dx = \mathop{\mathbb{E}}_{Y \sim q} \Big[\underbrace{h(Y) \frac{p(Y)}{q(Y)}}_{=: \, h^*(Y)} \Big], \tag{2.33}$$

and we often call the expectation in RHS as the *importance sampling estimator*. The variance of $h^*(Y)$ is written by

$$\mathrm{Var}\big(h^*(Y)\big) = \mathop{\mathbb{E}}_{Y \sim q} \left[h^*(Y)^2 \right] - Z^2.$$

Given this, one of the important goal in the importance sampling is to choose q such that the variance $\mathrm{Var}\big(h^*(Y)\big)$ is lower than the original.

To understand how the importance sampling can lower the variance, we can observe the difference between the variance of the above two random variables:

$$\mathrm{Var}\big(h(X)\big) - \mathrm{Var}\big(h^*(Y)\big) = \mathop{\mathbb{E}}_{X \sim \mathbb{P}} \left[h(X)^2 \right] - \mathop{\mathbb{E}}_{Y \sim \mathbb{Q}} \left[h^*(Y)^2 \right] = \int \left(1 - \frac{p(x)}{q(x)} \right) h(x)^2 p(x) dx.$$

When the difference is positive, we can expect that the MC estimate of the importance sampling estimator converges to the ground truth faster than the original estimator. Intuitively speaking, the difference will be positive if q is chosen such that

(i) $\frac{p(x)}{q(x)} < 1$ when $h(x)^2 p(x)$ is large and

[9] This example is copied and modified from the lecture note [link].

(ii) $\frac{p(x)}{q(x)} > 1$ when $h(x)^2 p(x)$ is small.

Note that it is necessary to consider the both cases, since satisfying the condition $\frac{p(x)}{q(x)} < 1$ for all $x \in \Omega$ is implausible: it contradicts the assumption that $\int q(x)dx = 1$.

As we are free to choose q, suppose that

$$q(x) = \frac{h(x)p(x)}{Z}. \tag{2.34}$$

By using this choice, we can show that the variance $\mathrm{Var}\big(h^*(Y)\big)$ becomes zero;

$$\mathrm{Var}\big(h^*(Y)\big) = \mathop{\mathbb{E}}_{Y\sim q}\left[h(Y)^2 \frac{p(Y)^2}{q(Y)^2}\right] - Z^2 = Z^2 - Z^2 = 0.$$

This implies that if we apply the Monte Carlo method using this specific choice of q, a single sample would be sufficient, as would be identical to Z with probability one.

Notes on choosing q The choice of q in Equation 2.34 ultimately depends on knowledge of the target value Z, making it impractical in most real-world scenarios. Instead, a more feasible approach is to train q in a way that minimizes the sample variance, which is a common strategy in practice (Robert et al., 1999).

At first glance, importance sampling may not seem directly related to generative modeling. However, similar challenges frequently arise in this field. For example, energy-based models require normalizing constant estimation, where we only have access to p but not able to sample from it, sharing conceptual parallels with importance sampling. The example presented above serves as a representative illustration of these broader connections.

Intuition behind the benefits Building on the previous example, let us now turn to the optimization problem of minimizing (or maximizing) the quantity Z with respect to p. This setting highlights how the variance of an estimator directly affects the stability of training.

Suppose we apply stochastic gradient descent (SGD). Figure 2.8 illustrates the optimization trajectories obtained with two different estimators. Under the same conditions—such as the same model, the same number of samples, and identical update rules—a low-variance estimator yields smoother gradient signals that consistently point toward the optimum. In contrast, a high-variance estimator produces noisy updates that wander and oscillate around the landscape. Consequently, training with low-variance estimators typically converges more rapidly and reliably to the optimal solution.

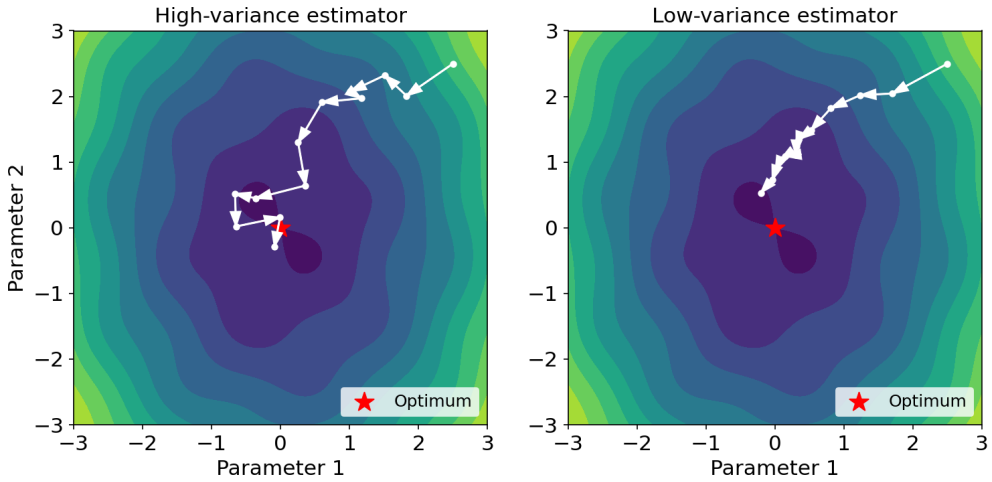


Figure 2.8: Effect of estimator variance on optimization. High-variance estimators (left) produce noisy and unstable updates, while low-variance estimators (right) yield smoother trajectories and faster convergence toward the optimum.

Instability of GAN training under finite samples In the preceding discussion of VAEs, we examined how finite-sample estimation of the training objective can affect the learning process. A similar phenomenon arises in the training of Generative Adversarial Networks (GANs). As discussed earlier, GANs are built upon implicit distributions, for which likelihoods are not directly accessible. Instead, training relies on variational formulations of statistical divergences estimated from finite samples. In this framework, the discriminator is trained to separate real samples from generated ones, while the generator is updated to fool the discriminator by minimizing the estimated divergence. Alternating between these two steps constitutes the core training loop of GANs.

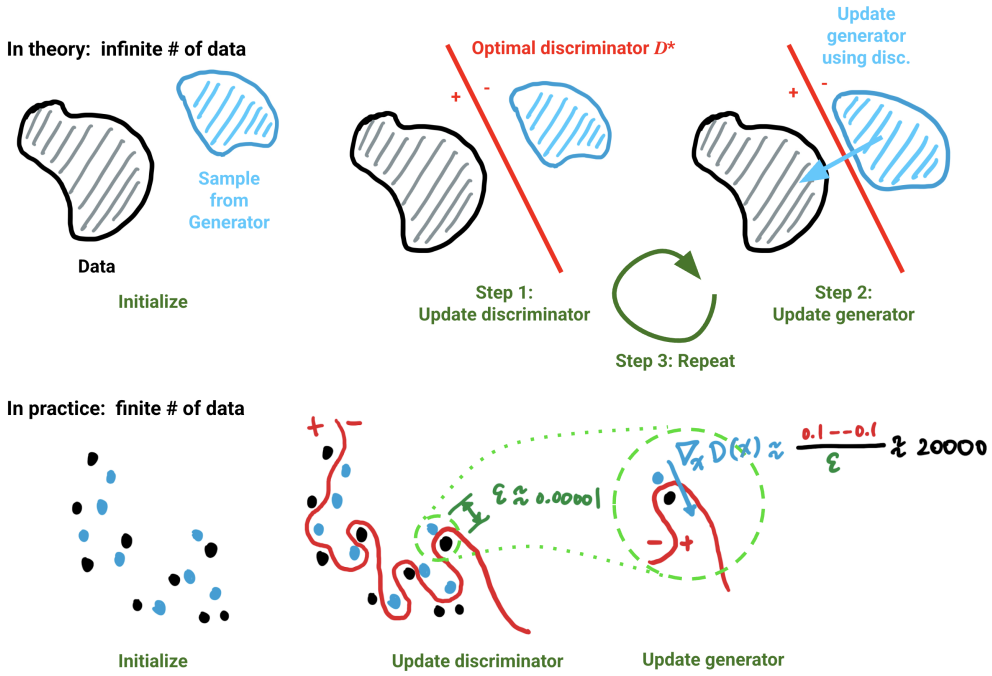


Figure 2.9: Training instability of GAN objective

In principle, with infinite data, the adversarial game converges as the generator distribution approaches the data

distribution. The top of Figure 2.9 illustrates this scenario. In practice, however, we are restricted to finite samples, which introduces a fundamental source of instability. Even when the generator's distribution is already close to the data distribution, a sufficiently expressive discriminator can construct sharp decision boundaries around individual data points, as shown in the bottom of Figure 2.9. These boundaries produce disproportionately large gradient signals for the generator, causing updates to overshoot the target direction and leading to oscillations or divergence. This explains why early GANs frequently exhibited severe instability in their training.

Addressing this instability has been a central theme in GAN research. The Wasserstein GAN (Arjovsky et al., 2017) reformulated the training objective as the Wasserstein-1 distance under the constraint that the discriminator $f \in \mathcal{F}$ be 1-Lipschitz, thereby controlling the scale of gradients. Practical implementations, such as gradient penalty (Gulrajani et al., 2017) and spectral normalization (Miyato et al., 2018), further enforce this constraint efficiently, regularizing the discriminator's gradients to achieve more stable training dynamics.

Thus, the instability of GANs under finite samples exemplifies how noisy and unreliable gradient signals can undermine optimization, even when the model class is expressive enough to represent the data distribution. This insight underscores a recurring theme in generative modeling: beyond the expressivity of the model or the alignment of the objective, the stability of training is ultimately determined by the quality and reliability of the gradient estimates available in practice.

2.6 Conclusion

We have primarily explored the fundamental components of random variable models and the key properties that influence their selection. Moving forward, we will delve deeper into various generative models, starting with VAEs, and examine the challenges associated with their training and application.

References

Arjovsky, M., Chintala, S., and Bottou, L. Wasserstein generative adversarial networks. *ICML*, 2017.

Carreira-Perpinan, M. A. and Hinton, G. On contrastive divergence learning. *AISTATS*, 2005.

Chen, R. T., Rubanova, Y., Bettencourt, J., and Duvenaud, D. K. Neural ordinary differential equations. *NeurIPS*, 2018.

Cover, T. M. *Elements of information theory*. John Wiley & Sons, 1999.

DasGupta, A. *Asymptotic theory of statistics and probability*, volume 180. Springer, 2008.

Dinh, L., Sohl-Dickstein, J., and Bengio, S. Density estimation using Real NVP. *ICLR*, 2016.

Frey, B. J. *Graphical models for machine learning and digital communication*. MIT press, 1998.

Goodfellow, I., Pouget-Abadie, J., Mirza, M., Xu, B., Warde-Farley, D., Ozair, S., Courville, A., and Bengio, Y. Generative adversarial nets. *NeurIPS*, 2014a.

Goodfellow, I. J., Shlens, J., and Szegedy, C. Explaining and harnessing adversarial examples. *arXiv preprint arXiv:1412.6572*, 2014b.

Gulrajani, I., Ahmed, F., Arjovsky, M., Dumoulin, V., and Courville, A. C. Improved training of Wasserstein gans. *NeurIPS*, 2017.

He, X., Zemel, R. S., and Carreira-Perpinán, M. A. Multiscale conditional random fields for image labeling. *CVPR*, 2004.

Huang, C.-W., Krueger, D., Lacoste, A., and Courville, A. Neural autoregressive flows. *ICML*, 2018.

Hyvärinen, A. and Dayan, P. Estimation of non-normalized statistical models by score matching. *Journal of Machine Learning Research*, 6(4), 2005.

Johnson, O. *Information theory and the central limit theorem*. World Scientific, 2004.

Karras, T., Laine, S., and Aila, T. A style-based generator architecture for generative adversarial networks. *CVPR*, 2019.

Kingma, D. P. and Welling, M. Auto-encoding variational bayes. *ICLR*, 2014.

Kingma, D. P., Salimans, T., Jozefowicz, R., Chen, X., Sutskever, I., and Welling, M. Improved variational inference with inverse autoregressive flow. *NeurIPS*, 2016.

Köhler, J., Klein, L., and Noé, F. Equivariant flows: exact likelihood generative learning for symmetric densities. 2020.

Larochelle, H. and Murray, I. The neural autoregressive distribution estimator. *AISTATS*, 2011.

Li, S. Z. Markov random field modeling in image analysis. 2001.

Miyato, T., Kataoka, T., Koyama, M., and Yoshida, Y. Spectral normalization for generative adversarial networks. *ICLR*, 2018.

Mohamed, S. and Lakshminarayanan, B. Learning in implicit generative models. *arXiv preprint arXiv:1610.03483*, 2016.

Nowozin, S., Cseke, B., and Tomioka, R. f-GAN: Training generative neural samplers using variational divergence minimization. *NeurIPS*, 2016.

Rezende, D. J. and Mohamed, S. Variational inference with normalizing flows. *ICML*, 2015.

Rezende, D. J., Mohamed, S., and Wierstra, D. Stochastic backpropagation and approximate inference in deep generative models. *ICML*, 2014.

Robert, C. P., Casella, G., and Casella, G. *Monte Carlo statistical methods*, volume 2. Springer, 1999.

Roth, K., Lucchi, A., Nowozin, S., and Hofmann, T. Stabilizing training of generative adversarial networks through regularization. *NeurIPS*, 2017.

Salimans, T. and Ho, J. Progressive distillation for fast sampling of diffusion models. *arXiv preprint arXiv:2202.00512*, 2022.

Satorras, V. G., Hoogeboom, E., and Welling, M. E(n) equivariant graph neural networks. *ICML*, 2021.

Song, J., Meng, C., and Ermon, S. Denoising diffusion implicit models. *ICLR*, 2021.

Szegedy, C., Zaremba, W., Sutskever, I., Bruna, J., Erhan, D., Goodfellow, I., and Fergus, R. Intriguing properties of neural networks. *arXiv preprint arXiv:1312.6199*, 2013.

Teh, Y. W., Welling, M., Osindero, S., and Hinton, G. E. Energy-based models for sparse overcomplete representations. *Journal of Machine Learning Research*, 4(Dec):1235–1260, 2003.

Wikipedia. Wasserstein metric — Wikipedia, the free encyclopedia. `http://en.wikipedia.org/w/index.php?title=Wasserstein%20metric&oldid=1233567381`, 2024. [Online; accessed July-2024].

Winkler, G. *Image Analysis, Random Fields and Markov chain Monte Carlo Methods*. Springer-Verlag, second edition, 2002.

Yoshida, Y. and Miyato, T. Spectral norm regularization for improving the generalizability of deep learning. *arXiv preprint arXiv:1705.10941*, 2017.

Chapter 3

Variational Autoencoders

3.1 Introduction

In Section 2.3.2, we briefly introduced Variational Autoencoders (VAEs) as an example of latent variable models. In this chapter, we take a closer look. VAEs are not only influential in practice but also particularly valuable from a pedagogical perspective: they embody many of the challenges and ideas that define modern generative modeling.

At their core, VAEs highlight two recurring difficulties. First, capturing complex data distributions with model class with limited capacities. VAEs—more precisely, latent variable models—demonstrate how latent representations can be introduced to address this issue—by making infinite mixture. Second, they reveal the fundamental obstacle of inference: the posterior distribution is typically intractable, defined only up to an unnormalized density. To address this challenge, VAEs rely on approximate inference, variational formulations, and stochastic estimation.

Because of this, VAEs serve as more than just a single model family. They provide a concrete, accessible framework through which we can study broader questions: How do we approximate distributions we cannot compute exactly? How do we train models reliably when direct likelihood evaluation is infeasible? By answering these questions in the context of VAEs, we prepare ourselves to confront them in the wider landscape of generative modeling.

In the remainder of this chapter, we will first revisit the formulation of variational autoencoders in Section 3.2, consolidating the ideas introduced earlier. We will then turn to one of the central research directions surrounding VAEs: reducing the bias of learning signals, which will be discussed in Section 3.3 along with related approaches.

3.2 Latent variable models and variational autoencoders

Variational autoencoders (VAEs, Kingma & Welling, 2014; Rezende et al., 2014) are among the most popular and widely-used latent variable models (LVMs, Bishop, 1998) for modeling continuous random variables. The model explicitly defines the (log) likelihood of an input x as follows:

$$\log p(x) = \log \int p(x|z)p(z)dz, \tag{3.1}$$

where z is the latent random variable governed by a pdf $p(z)$, often referred to as the *prior*, and the conditional pdf $p(x|z)$ is known as the *decoder*. In practice, decoders are commonly modeled using isotropic Gaussian distributions—assuming we are modeling continuous random variables—while the priors are typically assumed to follow standard normal distributions without learnable parameters. However, learnable priors can also be employed in some cases.

The reason VAEs are so widely adopted for modeling continuous random variables is due to their powerful expressiveness despite their simplicity. Notably, a VAE can be interpreted as an infinite Gaussian mixture model since

uncountably many z-conditional normal distributions $p(x|z)$ are mixed with the weights $p(z)$ for each z. It is well established that infinite Gaussian mixture models are universal density approximators (Dalal & Hall, 1983; Goodfellow et al., 2016), and thus, VAEs inherit this property, making them universal density approximators (also see Section 2.5.1 for the relevant discussions).

3.2.1 Maximum likelihood estimation

Since the log-likelihood $\log p(x)$ is explicitly defined, it is natural to train p via maximizing the expected values of the log-likelihoods: that is

$$\mathcal{L}_{\mathrm{MLE}}(p) \coloneqq \mathbb{E}\left[\log p(X)\right], \tag{3.2}$$

where $X \sim p_{\mathrm{data}}$ and p_{data} is the pdf of a data distribution. Note that *maximum likelihood estimation (MLE)* is equivalent to minimizing (forward) KL since

$$D_{\mathrm{KL}}\left(p_{\mathrm{data}} \| p\right) = \underbrace{\mathbb{E}\left[\log p_{\mathrm{data}}(X)\right]}_{=\ p\text{-independent}} - \mathbb{E}\left[\log p(X)\right]$$

where $X \sim p_{\mathrm{data}}$ in both expectations.

3.2.2 Evidence lower bound

Training the model p is not straightforward as it seems, since the integrals often don't have analytical solutions; thus, the log-likelihood $\log p(x)$ is intractable in general. Moreover, simple Monte Carlo (MC) estimate of the log-likelihood Equation 3.1 by taking the expectation of $\log p(x|z)$ over $p(z)$ suffer from high-variance, especially when the dimension of z is large.

Instead of directly maximizing the log-likelihood $\log p(x)$, we can use a useful quantity, derived from importance sampling techniques, called *Evidence Lower Bound (ELBO)*;

$$\begin{aligned}
\log p(x) &= \log \int p(x|z)p(z)dz \\
&= \log \int \frac{q(z|x)}{q(z|x)} p(x|z)p(z)dz \\
&\overset{(1)}{\geq} \int \log\left(\frac{p(x|z)p(z)}{q(z|x)}\right) q(z|x)dz \\
&= \mathop{\mathbb{E}}_{Z\sim q(\cdot|x)}\left[\log p(x|Z) + \log p(Z) - \log q(Z|x)\right] \\
&=: \mathcal{E}(x; p, q),
\end{aligned} \tag{3.3}$$

where (1) is due to Jensen's inequality and $q(z|x)$ is x-specific proposal distribution. Note that $q(z|x)$ doesn't need to be x-conditional in general. In other words, rather than maximizing the intractable $\log p(x)$, we instead maximize its tractable surrogate, the ELBO $E(x; p, q)$—and consequently maximizing the expected value of the ELBO. This reformulation turns the training objective into a quantity that can be efficiently estimated and optimized in practice.

3.2.3 Variational inference and variational gap

Previously, we stated that instead of maximizing the intractable log-likelihood $\log(p)$, we instead maximize the ELBO $E(x; p, q)$ with respect to p for some q. This naturally raises the question: what constitutes a good choice of q? As the ELBO $E(x; p, q)$ lower bounds the log-likelihood $\log p(x)$ regardless of the choice of q—even when it is estimated via

MC method—maximizing it will result in maximizing the original objective. More precisely, we will maximize the maximum of $\mathcal{E}(x; p, q)$ with respect to q; that is,

$$\log p(x) \geq \max_q \mathcal{E}(x; p, q) \tag{3.4}$$

We will optimize the above quantity with respect to p.

In the pre-deep learning era, the mean-field methods optimize $q(z|x)$ for each x independently, often as Gaussian (Jordan et al., 1999; Wainwright et al., 2008; Blei et al., 2017); in order words, it learns the posterior's mean and variance per sample.[1] This optimization is often performed during test time when x is observed.[2]

In contrast, Kingma & Welling (2014); Rezende et al. (2014) proposes to define a x-conditional model (often a single model) to represent all posteriors—that is the pdf function $q(z|x)$ of a VAE, called the *encoder*.[3] As a result, instead of maximizing $\mathcal{L}_{\mathrm{MLE}}$ of Equation 3.2, we maximize the expected values of the ELBO over the data distribution: that is,

$$\begin{aligned} &\max_{p,q} \mathcal{L}_{\mathrm{ELBO}}(p, q), \\ \text{where} \quad &\mathcal{L}_{\mathrm{ELBO}}(p, q) \coloneqq \mathop{\mathbb{E}}_{X \sim p_{\mathrm{data}}} \left[\mathcal{E}(X; p, q)\right]. \end{aligned} \tag{3.5}$$

This choice benefits to further reducing variance thanks to sharing parameters across different x observation during training.[4]

We commonly maximize the $\mathcal{L}_{\mathrm{ELBO}}$ w.r.t. both p and q simultaneously as stated in Equation 3.2.3. However, it is also valid to alternate between training one and another; such process is often called the *expectation maximization (EM)*.

Variational gap To better understand the effect of maximizing the ELBO with respect to $q(z|x)$, we examine the difference between the log-likelihood and the ELBO. This difference, often referred to as the *variational gap*, is given by

$$\log p(x) - \mathcal{E}(x; p, q) = \mathop{\mathbb{E}}_{Z \sim q(\cdot|x)} \big[\log q(Z|x) - \log p(Z|x)\big] = D'_{\mathrm{KL}}\big(q(z|x) \big\| p(z|x)\big). \tag{3.6}$$

Here, $p(z|x)$ represents the posterior distribution of z given x under p, which is generally intractable. We use the notation D'_{KL} instead of D_{KL} to emphasize that the expectation is taken with respect to $q(z|x)$, while x is fixed.[5]

Acknowledging that the ELBO can not be larger than the log-likelihood, we can concludes that maximizing the ELBO w.r.t. q for a fixed p is minimizing the variational gap; the learning aims at modeling the true posterior distribution. Thus, if the a family of densities that $q(z|x)$ can represent is sufficiently large enough, this training will give us $q(z|x) = p(z|x)$, resulting in the gap being zero. In the optimal scenario, maximizing $\mathcal{L}_{\mathrm{ELBO}}$ w.r.t. p becomes equivalent to maximizing $\mathcal{L}_{\mathrm{MLE}}$.

However, achieving a zero-gap scenario is practically impossible. As a consequence, maximizing the ELBO may not always align with the original goal of maximum likelihood, particularly when the gap is nonzero (as we discussed in Section 2.5.2). Let us take a closer look at this issue.

[1] The mean-field approximation has its roots in statistical physics (Parisi & Shankar, 1988) and has become the most widely used variational family in probabilistic modeling.

[2] The formulation in Equation 3.4 expresses the quantity of interest as an optimization problem, and the learning of q in this setting is referred to as *variational inference* (Blei et al., 2017).

[3] The terminology **autoencoder** arises because the model is composed of two complementary components: the encoder $q(z|x)$ and the decoder $p(x|z)$. Combined with the term **variational**, this gives rise to the name **variational autoencoder**.

[4] Interestingly, this approach achieves the same goal as the mean-field method, but it shifts the inference time cost from test time to training. In the machine learning literature, this strategy is often referred to as *amortized inference*, emphasizing the idea of (in a sense) borrowing the time required for inference and paying it during training.

[5] By common convention, $D_{\mathrm{KL}}(q(z|x) \| p(z|x)) = \mathbb{E}[D'_{\mathrm{KL}}(q(z|X) \| p(z|X))]$.

Implication of the non-zero variational gap What, then, are the concrete issues that arise when the variational gap is nonzero? While we already touched on this in Section 2.5.2, it is worth revisiting in the present context. To make this point concrete, let us consider two models: one obtained by maximizing the true log-likelihood (*i.e.*, MLE), and another obtained by maximizing its lower bound, the ELBO. For these, we write

$$p^*_{\mathrm{MLE}} = \arg\max_p \mathcal{L}_{\mathrm{MLE}}(x;p) \quad \text{and} \quad p^*_{\mathrm{ELBO}} = \arg\max_p \mathcal{L}_{\mathrm{ELBO}}(p,q).$$

While maximizing the ELBOs has been being a very important approach in VAEs, it is important to note that p^*_{MLE} and p^*_{ELBO} may not be equivalent.

In other words, even if our model class is universal pdf approximators, the learned model p^*_{ELBO} may be very far from the target data distribution p_{data}. This suboptimality arises because, in general, $\mathcal{L}_{\mathrm{ELBO}} \neq \mathcal{L}_{\mathrm{MLE}}$: the variational gap is strictly positive. It is helpful to remember that the ELBO provides a biased training signal for p in general. Consequently, the gradient of $\mathcal{L}_{\mathrm{ELBO}}$ with respect to p does not align with the gradient of $\mathcal{L}_{\mathrm{MLE}}$, meaning that maximizing the ELBO may guide the model toward a different solution than true maximum likelihood. This distinction underscores why closing the variational gap is a central theme in the development of more powerful VAEs (also see Section 2.5.2 for the relevant discussions).

3.3 Reducing variational gap

One of the most important topics in recent VAEs is minimizing the variational gap. There are several approaches in this topic. However, it can be broadly divided into two big branches in the context of the variational inference. One is to make the proposal distributions (posterior distributions) more expressive. The other is to use a different estimator other than ELBO. In this section, we will first discuss methods for creating powerful posterior distributions. Then, we will briefly discuss how to perform MLE using a different estimator other than ELBO.

3.3.1 Expressive posterior

When the posterior $q(z|x)$ is not powerful enough, e.g. isotropic Gaussians, it is fundamentally restricted to achieve zero-variational gap. Such not-so-expressive posteriors inevitably produce biased learning signals for training the likelihoods p, resulting in suboptimal solutions.

Hierarchical VAE One of the simplest ways to improve the expressivity of the posteriors is *auxiliary hierarchical method* or simply *hierarchical method* (Maaløe et al., 2016). It introduces an auxiliary latent variable Y in addition to a vanilla VAE in Equation 3.1. More specifically, it aims to build *infinite mixture models* for Z. Thus, its ELBO is written as

$$\mathcal{E}(x;p,q) = \mathbb{E}\left[\log\left(\frac{p(x|Z)\,p(Z)\,p(Y|x,Z)}{q(Y|x)\,q(Z|Y,x)}\right)\right],$$

where $Y \sim q(y|x)$ and $Z \sim q(z|y,x)$. The variational gap for a given input x is

$$\log p(x) - \mathcal{E}(x;p,q) = D_{\mathrm{KL}}\left(q(y,z|x)\|p(y,z|x)\right).$$

Note that the gap may not become zero as long as $p(y|x,z)$, $q(y|x)$, and $q(z|y,x)$ are isotropic Gaussians. Nevertheless, it has been well studied that adding auxiliary latent variables reduces the variational gap in general in comparison to the vanilla VAEs, even when the probability densities are restricted as isotropic Gaussians. Therefore, there have been several works to improve the performances of hierarchical VAEs (Sønderby et al., 2016). For instance, combined with several stabilization techniques for training, Vahdat & Kautz (2020) proposes one of the first VAE-based models that performs comparable to GAN-based models.

Normalizing flow Another popular approach to improve the posterior distributions is using *normalizing flows* (*NFs*, Rezende & Mohamed, 2015; Kingma et al., 2016). As discussed in Section 2.3, a normalizing flow defines a random variable with another simple random variable to which a complex bijection is applied. The bijection is constructed by composing a sequence of simple bijections, *a.k.a.* invertible layers. With carefully designed invertible layers, we are able to compute the exact log densities of zs given x—by using a change-of-variable formula—and generate samples at the same time. In particular, suppose the for $i = 1, \dots, L$, functions $f^{(i)} : \mathbb{R}^d \times \mathbb{R}^d \to \mathbb{R}^d$ are bijective with respect to the first argument. To define $q(z|x)$, we construct a sequence of latent variables $z_0, \dots, z_L = z$ by using $f^{(i)}$s; that is,

$$z_i = f^{(i)}(z_{i\text{-}1}, x)$$

for $i = 1, \dots, L$, where $z_0 \sim N(0, I)$. By denoting z_i's pdf for a given x as $q_i(\cdot|x)$, we have

$$\begin{aligned}
\log q(z|x) &= \log q_{L\text{-}1}(z_{L\text{-}1}|x) + \log |dz_{L\text{-}1}/dz| \\
&= \log q_{L\text{-}2}(z_{L\text{-}2}|x) + \log |dz_{L\text{-}2}/dz_{L\text{-}1}| + \log |dz_{L\text{-}1}/dz| \\
&\dots \\
&= \log q_0(z_0|x) + \sum_{i=1}^{L} \log |dz_{i\text{-}1}/dz_i|,
\end{aligned}$$

where $q_0(\cdot|x) = q_0(\cdot) = N(0, I)$. Moreover, since $\log |dz_{i\text{-}1}/dz_i| = -\log |dz_i/dz_{i\text{-}1}|$, we can compute the log-det-Jacobian of $f^{(i)}(z_{i\text{-}1}, x)$ on the way to sample z starting from z_0.

Likelihood-free inference Third approaches is *likelihood-free inference in VAEs*, and it exploits that maximizing the ELBO with respect to θ doesn't require to evaluate the entropy of $\log q(z|x)$. More specifically, when we take a look at the ELBO $\mathcal{E}(x; \theta, \phi)$

$$\mathcal{E}(x; \theta, \phi) = \mathop{\mathbb{E}}_{q(\cdot|x)} [\, \underbrace{\log p_\theta(x|Z) + \log p_\theta(Z)}_{=\theta\text{-dependent}} - \log q(Z|x) \,],$$

where the last term in the expectation doesn't depend on ϕ. This implies that in order to learn the generative models p_θ, we don't need to evaluate $\log q(z|x)$, but we only need to sample the latent variable Z. Therefore, instead of the isotropic Gaussians, one can use implicit distributions for $q(z|x)$. For example, Mescheder et al. (2017) proposes to use implicit distributions for $q(z|x)$ and train it via adversarial training, with which $D_{\mathrm{KL}}\left(q(z|x) \| p_\theta(z)\right)$. Lim et al. (2020) also proposes to use implicit variational posteriors but training the posteriors via approximating the pathwise gradients of the entropy term of $q(z|x)$.

3.3.2 Importance weighting

Instead of improving the expressivity of the variational posteriors, one can improve the quality of likelihood estimators. In particular, Burda et al. (2015) proposes an improved estimator, called *Importance Weighted Autoencoder (IWAE)*, lower-bounding the likelihood, but whose variational gap is tighter than the vanilla VAEs. More specifically, the paper proposes the IWAE-ELBO $\mathcal{E}_k$, which uses k-number of i.i.d. samples of the variational posterior, and it is defined as

$$\log p_\theta(x) = \log \mathbb{E}\left[\frac{1}{k}\sum_{i=1}^{k} \frac{p_\theta(x, Z_i)}{q_\theta(Z_i|x)}\right] \geq \mathbb{E}\left[\log \frac{1}{k}\sum_{i=1}^{k} \frac{p_\theta(x, Z_i)}{q_\theta(Z_i|x)}\right] =: \mathcal{E}_k(x;\, \theta, \phi), \tag{3.7}$$

where $Z_i \overset{\text{i.i.d.}}{\sim} q(\cdot|x)$ for $i = 1, \dots, k$. In addition, the authors emphasize the following properties of the estimator;

(i) $\log p_\theta(x) \geq \mathcal{E}_{k+1} \geq \mathcal{E}_k \geq \mathcal{E}_1 = \mathcal{E}$, where $\mathcal{E}$ is the ELBO.

(ii) $\lim_{k\to\infty} \mathcal{E}_k = \log p_\theta(x)$.

Note that (i) implies that with an increasing number of k, we may get tighter lower bounds. This conjecture has been empirically demonstrated in the paper and further supported by an in-depth analysis of the estimator from Cremer et al. (2017). Thus, even with simple posterior distributions, one may reach out to the optimal solutions for p_θ by maximizing the IWAE-ELBO. However, this approach has a trade-off that in exchange for reducing the bias for training p_θ, the encoder $q(z|x)$ becomes ineffective to predict Z from X (Alemi et al., 2018).

3.3.3 Alternatives to variational inference

Markov chain Monte Carlo (MCMC) methods offer a well-established alternative to variational methods. Traditionally, unlike variational approaches, MCMC does not involve learning. It instead to aim at sampling from the true posterior $p(z|x)$. Key techniques in this domain include the following:

(i) Metropolis–Hastings algorithm

(ii) (Unadjusted) Langevin algorithm (ULA), also known as Langevin dynamics, and Metropolis-adjusted Langevin algorithm (MALA)

(iii) Hamiltonian Monte Carlo (HMC)

(iv) Annealed importance sampling (AIS)

(v) Sequential Monte Carlo (SMC).

MCMC methods have a longer history than deep learning-based variational techniques and are widely applied in areas such as inference, sampling from unnormalized densities, Bayesian learning, and more. In recent years, studies that combine variational inference with MCMC-based methods have gained popularity. Notably, in Section 6.6, we briefly introduced diffusion-based sampling methods, and Chapter **??** will discuss on this topics more closely. Despite their relevance to the variational inference, this tutorial will not delve into MCMC-based methods. However, they remain an exciting area for further exploration.

References

Alemi, A., Poole, B., Fischer, I., Dillon, J., Saurus, R. A., and Murphy, K. An information-theoretic analysis of deep latent-variable models. 2018.

Bishop, C. M. Latent variable models. In *Learning in graphical models*, pp. 371–403. Springer, 1998.

Blei, D. M., Kucukelbir, A., and McAuliffe, J. D. Variational inference: A review for statisticians. *Journal of the American statistical Association*, 112(518):859–877, 2017.

Burda, Y., Grosse, R., and Salakhutdinov, R. Importance weighted autoencoders. *arXiv preprint arXiv:1509.00519*, 2015.

Cremer, C., Morris, Q., and Duvenaud, D. Reinterpreting importance-weighted autoencoders. *arXiv preprint arXiv:1704.02916*, 2017.

Dalal, S. and Hall, W. Approximating priors by mixtures of natural conjugate priors. *Journal of the Royal Statistical Society: Series B (Methodological)*, 45(2):278–286, 1983.

Goodfellow, I., Bengio, Y., and Courville, A. *Deep learning*. MIT press, 2016.

Jordan, M. I., Ghahramani, Z., Jaakkola, T. S., and Saul, L. K. An introduction to variational methods for graphical models. *Machine learning*, 37(2):183–233, 1999.

Kingma, D. P. and Welling, M. Auto-encoding variational bayes. *ICLR*, 2014.

Kingma, D. P., Salimans, T., Jozefowicz, R., Chen, X., Sutskever, I., and Welling, M. Improved variational inference with inverse autoregressive flow. *NeurIPS*, 2016.

Lim, J. H., Courville, A., Pal, C., and Huang, C.-W. AR-DAE: towards unbiased neural entropy gradient estimation. *ICML*, 2020.

Maaløe, L., Sønderby, C. K., Sønderby, S. K., and Winther, O. Auxiliary deep generative models. *ICML*, 2016.

Mescheder, L., Nowozin, S., and Geiger, A. Adversarial variational bayes: Unifying variational autoencoders and generative adversarial networks. *ICML*, 2017.

Parisi, G. and Shankar, R. Statistical field theory. 1988.

Rezende, D. J. and Mohamed, S. Variational inference with normalizing flows. *ICML*, 2015.

Rezende, D. J., Mohamed, S., and Wierstra, D. Stochastic backpropagation and approximate inference in deep generative models. *ICML*, 2014.

Sønderby, C. K., Raiko, T., Maaløe, L., Sønderby, S. K., and Winther, O. Ladder variational autoencoders. *NeurIPS*, 29, 2016.

Vahdat, A. and Kautz, J. Nvae: A deep hierarchical variational autoencoder. *NeurIPS*, 2020.

Wainwright, M. J., Jordan, M. I., et al. Graphical models, exponential families, and variational inference. *Foundations and Trends® in Machine Learning*, 1(1–2):1–305, 2008.

Chapter 4

Normalizing flows and neural ODEs

This chapter is under preparation and will be updated in a future version.

References

Chapter 5

Score matching and denoising autoencoders

5.1 Introduction

In Section 2.4.3, we briefly introduced the Fisher divergence as an alternative to the KL divergence in the context of statistical distances. One of the most influential methods that directly builds on this idea is score matching (SM), which minimizes the Fisher divergence.

Although originally proposed as a technique for training unnormalized statistical models, score matching has since taken on a much broader role in modern generative modeling. In particular, SM has become a key building block for diffusion-based and related generative models, where the notion of a score function plays a central role.

Because of its recurring importance across diverse modeling paradigms, SM merits a dedicated chapter. Here we will review the core formulation and highlight its extensions—including its link to denoising autoencoders—that have enabled the training of powerful modern generative models at scale.

5.2 Score matching families

5.2.1 (Explicit) score matching

Score matching and its variants have been proposed to train energy-based models. To understand the motivation behind score matching, we briefly introduce the energy-based models. As we briefly discussed in Equation 5.1, the *energy-based models (EBMs)* define its energy functions $f_\theta(x)$ in order to define the densities of the Boltzmann distributions,

$$p_\theta(x) = \frac{\exp\left(-f_\theta(x)\right)}{Z}, \tag{5.1}$$

where Z is a *normalizing constant*, and it is defined as $Z = \int \exp\left(-f_\theta(x)\right)\, \mathrm{d}x$. Thanks to the fact that deep neural networks can approximate any continuous function, f_θ can model any continuous function, and thus, the density in Equation 5.1 is a universal pdf approximator.

A notable downside of EBMs is that the computation of the normalizing constant is intractable in general. To circumvent this issue, minimizing Fisher divergence is proposed instead of maximizing likelihoods (Hyvärinen & Dayan, 2005). As described in Equation 2.14, Fisher divergence is

$$D_{\textit{Fisher}}\left(p(x) \| p_\theta(x)\right) \coloneqq \int \left\| \nabla \log p(x) - \nabla \log p_\theta(x) \right\|^2 p(x)\, \mathrm{d}x,$$

where $p(x)$ is the pdf of a target distribution to model. The derivative of log density with respect to input $\nabla \log p(x)$ is called *score function*.

The thing to note here is that $\nabla \log p_\theta(x)$ doesn't require computing the normalizing constant since

$$\nabla \log p_\theta(x) = -\nabla f_\theta(x).$$

Thus, minimizing Fisher divergence is called *explicit score matching (ESM)* since the model's score function learns to match the target distribution's score, and we write the objective function as

$$\mathcal{L}_{\mathrm{ESM}}(p_\theta) \coloneqq D_{\textit{Fisher}}\left(p \| p_\theta\right) = \mathbb{E}\Big[\left\|\nabla \log p(X) - \nabla \log p_\theta(X)\right\|^2\Big], \tag{5.2}$$

where $X \sim p$. One may not need to use EBM and instead model the score function directly.

$$\nabla \log p_\theta(x) = s_\theta(x). \tag{5.3}$$

Such parameterization is often called as *score models*.

While ESM doesn't require to compute the normalizing constant of the models, it has a trade-off. For common application scenarios in generative modeling, we don't have access to $\nabla \log p(x)$, and we rather have samples of X instead. Thus, it is difficult to minimize ESM directly in practice. In the following sections, we discuss unbiased estimators, which don't require a ground truth score.

5.2.2 Implicit score matching

As we discussed before, it is difficult to optimize the ESM objective in practice. However, there exists an unbiased estimator of the Fisher divergence when s_θ satisfies

$$\lim_{x \to \pm\infty} s_\theta(x)\, p(x) = 0.$$

Then, we have

$$\begin{aligned}
\mathbb{E}\left[\left\|\nabla \log p(X) - s_\theta(X)\right\|^2\right] &\overset{*}{=} \mathbb{E}\left[\frac{1}{2}\left\|s_\theta(X)\right\|^2 + \nabla \cdot s_\theta(X)\right] \\
&=: \mathcal{L}_{\mathrm{ISM}}\left(\theta\right),
\end{aligned} \tag{5.4}$$

where $(*)$ is due to the following identity

$$\begin{aligned}
\mathbb{E}\left[s_\theta(X)^\top \nabla \log p(X)\right] &= \int s_\theta(x)^\top \nabla \log p(x)\, p(x)\, \mathrm{d}x \\
&= \int s_\theta(x)^\top \nabla p(x)\, \mathrm{d}x \\
&\overset{\dagger}{=} \left[s_\theta(x)\, p(x)\right]_{-\infty}^{\infty} - \int \nabla \cdot s_\theta(x)\, p(x)\, \mathrm{d}x \\
&\overset{\ddagger}{=} -\,\mathbb{E}\left[\nabla \cdot s_\theta(X)\right],
\end{aligned}$$

where $(\dagger)$ is due to the integration by parts and $(\ddagger)$ is due to the assumptions on $s_\theta(x)$. We refer to the RHS estimator in Equation 5.4 as *implicit score matching (ISM)* estimator.

5.2.3 Sliced score matching

The ISM objective requires computing the divergence, whose computation is often expensive in modern machine learning frameworks, especially when the dimensionality of data is high. To address this issue, Song et al. (2020) proposes to use Hutchinson's trick (Hutchinson, 1989; Adams et al., 2018) to estimate the second order term in

Equation 5.4;

$$\mathbb{E}\left[\frac{1}{2}\left\|s_\theta(X)\right\|^2 + \nabla \cdot s_\theta(X)\right] \overset{*}{=} \mathbb{E}\left[\frac{1}{2}\left\|s_\theta(X)\right\|^2 + V^\top \nabla \left(V^\top s_\theta(X)\right)\right] \\ =: \mathcal{L}_{\mathrm{SSM}}(\theta), \tag{5.5}$$

where a random variable $V = (V_1, V_2, \ldots, V_d)$ is a d-dimensional random variable satisfying the following conditions

(i) $V_1, V_2, \ldots, V_d$ are i.i.d. and zero mean. Thus, $\mathbb{E}[V_i V_j] = 0$ for all $i \neq j$.

(ii) $\mathbb{E}\left[V_i^2\right] = 1$ for all $i = 1, \ldots, d$.

. Thus, $(*)$ holds because for any x fixed we have

$$\mathbb{E}\left[V^\top \nabla \left(V^\top s_\theta(x)\right)\right] = \mathbb{E}\left[\sum_{j,i} V_j V_i \frac{\partial s_{\theta,i}(x)}{\partial x_j}\right] \overset{\dagger}{=} \sum_i \frac{\partial s_{\theta,i}(x)}{\partial x_i} = \nabla \cdot s_\theta(x),$$

where $(\dagger)$ is due to the assumptions on V. We refer this estimator as *sliced score matching (SSM)* estimator.

The quality of the SSM estimator depends on the choice of V. The two most common choices of V are the independent normal distribution and the Rademacher distribution. A random variable V follows Rademacher distribution and is defined as

$$V = \begin{cases} 1 & \text{with probability } 0.5 \\ -1 & \text{with probability } 0.5, \end{cases} \tag{5.6}$$

It has been known that the SSM estimator with Rademacher distribution has lower variance compared to the one with normal distribution (Song et al., 2020).

5.2.4 Denoising score matching

While there exist a few variance reduction techniques for SSM estimators, it still suffers from high variance issues, especially for high-dimensional data. However, when the target random variable X constitutes a certain construction, a low-variance estimator exists that is more suitable for high-dimensional data. When X is defined as

$$X = \alpha X_0 + \sigma \varepsilon, \tag{5.7}$$

where $X_0 \sim p_0$, $\varepsilon \sim N(0, I)$, $\alpha \in \mathbb{R}$, and $\sigma > 0$. We denote its pdf as $p(x)$. Then, we have that

$$\mathbb{E}\left[\left\|\nabla \log p(X) - s_\theta(X)\right\|^2\right] \overset{*}{=} \mathbb{E}\left[\left\|\nabla \log p(X|X_0) - s_\theta(X)\right\|^2\right] + C, \tag{5.8}$$

where C is θ-independent term, and it is defined as

$$C = \mathbb{E}\left[\left\|\nabla \log p(X)\right\|^2\right] - \mathbb{E}\left[\left\|\nabla \log p(X|X_0)\right\|^2\right].$$

Here, $(*)$ uses the following identity

$$\begin{aligned}
\int s_\theta(x)^\top \nabla \log p(x) p(x)\,\mathrm{d}x &= \int s_\theta(x)^\top \frac{\nabla p(x)}{\cancel{p(x)}}\cancel{p(x)}\,\mathrm{d}x \\
&= \int s_\theta(x)^\top \nabla \left(\int p(x|x_0) p_0(x_0) dx_0 \right) \mathrm{d}x \\
&\overset{\dagger}{=} \int \int s_\theta(x)^\top \nabla p(x|x_0) p_0(x_0)\,\mathrm{d}x_0\,\mathrm{d}x \\
&= \int \int s_\theta(x)^\top \nabla \log p(x|x_0) p(x|x_0) p_0(x_0)\,\mathrm{d}x_0\,\mathrm{d}x \\
&= \mathbb{E}\left[s_\theta(X)^\top \nabla \log p(X|X_0) \right] \qquad (5.9)
\end{aligned}$$

$(\dagger)$ is due to the Leibniz integral rule.

Since C in Equation 5.8 is θ-independent, minimizing the first term in the RHS is equivalent to minimizing the Fisher divergence. Thus, we define *denoising score matching (DSM, Vincent (2011))* objective as

$$\mathcal{L}_{\mathrm{DSM}}(s_\theta) \coloneqq \mathbb{E}\left[\left\| \nabla \log p(X|X_0) - s_\theta(X) \right\|^2 \right]. \qquad (5.10)$$

While DSM can be used under a limited condition, it has been demonstrated through a large number of works that minimizing DSM is better compared to ISM and SSM when it is applicable.

5.3 Denoising autoencoders and its connection to score matching

5.3.1 Denoising autoencoders

DSM has "denoising" in its name, but its connection to denoising is unclear. Interestingly, DSM has obtained its name through a study on the connection between denoising autoencoder's training and score matching (Vincent, 2011). Here, we first introduce *denoising autoencoder (DAE, Vincent et al. (2008))*, and then draw its connection to DSM.

Let us consider X in Equation 5.7 again, but here we assume $\alpha = 1$ for simplicity.

$$X = X_0 + \sigma\varepsilon, \qquad (5.11)$$

where $X_0 \sim p_0$, $\varepsilon \sim N(0, I)$, and $\sigma > 0$. As a result, we have $X \mid X_0 \sim N(X_0, \sigma^2)$.

Ultimately, DAE is solving the inverse problem above; for a given observation X, one needs to recover X_0. Specifically, DAE learns a model by minimizing mean squared error (MSE),

$$\mathcal{L}_{\mathrm{DAE}}(\mu_\theta) \coloneqq \mathbb{E}\left[\left\| X_0 - \mu_\theta(X) \right\|^2 \right]. \qquad (5.12)$$

Minimizing the MSE is equivalent to maximum-likelihood training of a Normal distribution while its variance is fixed to 1.

5.3.2 Bayesian inverse problems

We can consider the above problem as a Bayesian inverse problem. For such a problem, the optimal mean μ^* of is known to be the Bayes estimator, and it is written as

$$\mu^*(x) = \mathbb{E}\left[X_0 | X = x \right]. \qquad (5.13)$$

In particular, for the Bayesian inverse problem with Gaussian perturbation, we can rewrite $\mathbb{E}\left[X_0 | X = x \right]$ in an interesting form by using Tweedie's formula.

Theorem 5.1. [Tweedie's formula (Robbins, 1992; Efron, 2011)]

$$\mathbb{E}\left[X_0|X=x\right] = x + \sigma^2 \nabla \log p(x) \tag{5.14}$$

where $p(x)$ is X's pdf and ∇ acts on x.

Proof. Note that $p(x_0|x)$ is the density of a multivariate normal distribution. Then, we can re-write the Bayes estimator as in

$$\begin{aligned}
\mathbb{E}\left[X_0|X=x\right] = \int x_0\, p(x_0|x)\, dx_0 = \int x_0\, \frac{p(x|x_0)p(x_0)}{p(x)}\, \mathrm{d}x_0 \\
&\overset{\dagger}{=} \frac{xp(x) + \nabla p(x)}{p(x)} \\
&= x + \sigma^2 \nabla \log p(x),
\end{aligned} \tag{5.15}$$

where †-identity is due to

$$\begin{aligned}
\nabla p(x) &= \nabla \int p(x|x_0)p(x_0)\, \mathrm{d}x_0 \\
&= \int p(x_0) \nabla \frac{1}{\sqrt{2\pi\sigma^2}} \exp\left(-\frac{1}{2\sigma^2}\left\|x - x_0\right\|^2\right) \mathrm{d}x_0 \\
&= \int \frac{1}{\sigma^2}(x_0 - x)\, p(x_0) \frac{1}{\sqrt{2\pi\sigma^2}} \exp\left(-\frac{1}{2\sigma^2}\left\|x - x_0\right\|^2\right) \mathrm{d}x_0 \\
&= \frac{1}{\sigma^2}\left(\int x_0\, p(x|x_0)p(x_0)\, \mathrm{d}x_0 - x\, p(x)\right).
\end{aligned}$$

□

Following the solution's form, we can consider efficient parameterizations of μ_θ. For example, consider the following parameterization,

$$\mu_\theta(x) = x + \sigma^2 s_\theta(x).$$

In this case, we can show that the DAE objective is equivalent to the DSM in a straightforward manner.

$$\begin{aligned}
\mathcal{L}_{\mathrm{DAE}}(\mu_\theta) = \mathbb{E}\left[\left\|X_0 - \mu_\theta(X)\right\|^2\right] &= \mathbb{E}\left[\left\|X_0 - \left(X + \sigma^2 s_\theta(X)\right)\right\|^2\right] \\
&= \mathbb{E}\left[\left\|X_0 - \left(X_0 + \sigma\varepsilon + \sigma^2 s_\theta(X)\right)\right\|^2\right] \\
&= \sigma^4\, \mathbb{E}\left[\left\|\frac{\varepsilon}{\sigma} + s_\theta(X)\right\|^2\right] \\
&\overset{*}{=} \sigma^4 \underbrace{\mathbb{E}\left[\left\|\nabla \log p(X|X_0) - s_\theta(X)\right\|^2\right]}_{\mathcal{L}_{\mathrm{DSM}}(\theta)},
\end{aligned} \tag{5.16}$$

where $(*)$ is due to

$$\nabla \log p(X|X_0) = -\frac{X - X_0}{\sigma^2} = -\frac{\cancel{X_0} + \sigma\varepsilon - \cancel{X_0}}{\sigma^2} = -\frac{\varepsilon}{\sigma}.$$

Therefore, we can conclude that learn-to-denoise Gaussian noise is essentially performing score matching.

References

Adams, R. P., Pennington, J., Johnson, M. J., Smith, J., Ovadia, Y., Patton, B., and Saunderson, J. Estimating the spectral density of large implicit matrices. *arXiv preprint arXiv:1802.03451*, 2018.

Efron, B. Tweedie's formula and selection bias. *Journal of the American Statistical Association*, 106(496):1602–1614, 2011.

Hutchinson, M. F. A stochastic estimator of the trace of the influence matrix for Laplacian smoothing splines. *Communications in Statistics-Simulation and Computation*, 18(3):1059–1076, 1989.

Hyvärinen, A. and Dayan, P. Estimation of non-normalized statistical models by score matching. *Journal of Machine Learning Research*, 6(4), 2005.

Robbins, H. E. An empirical Bayes approach to statistics. In *Breakthroughs in Statistics: Foundations and basic theory*, pp. 388–394. Springer, 1992.

Song, Y., Garg, S., Shi, J., and Ermon, S. Sliced score matching: A scalable approach to density and score estimation. *UAI*, 2020.

Vincent, P. A connection between score matching and denoising autoencoders. *Neural computation*, 23(7):1661–1674, 2011.

Vincent, P., Larochelle, H., Bengio, Y., and Manzagol, P.-A. Extracting and composing robust features with denoising autoencoders. *ICML*, 2008.

Chapter 6

Diffusion-based generative models

6.1 Introduction

The most common approach for generative modeling is to build a stack of non-linear transformations that convert a simple distribution to a complex distribution. The advance in deep learning enables us to build arbitrary complex maps that are capable of representing almost any distribution. The flow-based models, including neural ordinary differential equations (neural ODEs) and GANs, are notable families before the diffusion-based generative models. However, training such complicated maps is extremely difficult, especially for high-dimensional data. The recent success of the diffusion-based generative models is a result of their efficient training method, while they are extremely deep generative models.

Extending from an initial work of Sohl-Dickstein et al. (2015), Ho et al. (2020) proposes denoising diffusion probability models (DDPMs), improving the diffusion-based generative models via efficient model parameterization and the utilization of UNet network architectures (Ronneberger et al., 2015). This becomes one of the first likelihood-based models to outperform GAN-based models at large-scale image generation benchmark datasets. Prior to DDPMs, Song & Ermon (2019) proposes score-based generative models based on annealed Langevin dynamics. Song et al. (2020) generalizes it to continuous-time diffusion-based generative models and establishes its connection to the denoising diffusions.

Since then, several influential practical techniques have been introduced; for instance, noise scheduling design to improve sampling qualities (Nichol & Dhariwal, 2021; Kingma et al., 2021; Karras et al., 2022) and more effective model parameterizations (Salimans & Ho, 2022). In addition to the image generation, such techniques have further improved the performance of the diffusion-based generative models in other applications, such as computer vision (Meng et al., 2022; Ho et al., 2022b; Saharia et al., 2022; Rombach et al., 2022), video generation (Ho et al., 2022c,a; Blattmann et al., 2023), and audio generation (Kong et al., 2021; Chen et al., 2021).

In this chapter, instead of following the chronological order of the history of the diffusion-based generative models, we will start from the definition of Itô diffusions and their properties in Section 6.2 in terms of stochastic differential equations. This will enable in-depth discussions on advanced topics on diffusion-based models. Then, we move on to the diffusion-based generative models in Section 6.3. In this, we will first explore the continuous-time score-based generative models in Section 6.3.1. In particular, we will discuss the theorem of the existence of time-reversed diffusions, which initiated the original motivation of the very first work of Sohl-Dickstein et al. (2015). Then, we will cover the diffusion-based models' other perspectives: the denoising diffusion perspectives in Section 6.3.2 and the annealed Langevin dynamics views in Section 6.3.3.

6.2 Stochastic differential equations and diffusions

In this section, we will briefly review the key concepts of stochastic differential equations, particularly in the context of diffusion-based generative models. We begin by exploring the foundation of continuous stochastic processes, which describe systems that evolve over some index (*e.g.* time) with inherent randomness. An essential example of this is the Wiener process, which plays a crucial role in stochastic differential equations. Building upon this, we delve into the formal definition of stochastic differential equations (SDEs), which govern the dynamics of systems influenced by both deterministic and random forces. To gain a deeper understanding of SDEs, we will examine them through their discretizations. Lastly, we introduce the Fokker-Planck equations, which describe the time evolution of probability distributions associated with the solutions of SDEs, offering insights into the probabilistic behavior of the system.

6.2.1 Stochastic process

In recent generative models, one of the most important approaches is to define a model's random variable as an outcome of the chains of transformations from a simple distribution. Applying such transformations produces a collection of random variables starting from a prior distribution,*i.e.* initial distribution, to a terminal distribution, which will model the data distribution. In order to characterize their behavior, we formally describe the definition of such a collection of random variables.

A *($\mathbb{R}^d$-valued) stochastic process* is a t-parametrized collection of random variables

$$\{X_t\}_{t\in\mathcal{T}}$$

defined on a probability space $(\Omega, \mathcal{F}, P)$ and each X_t's value in $\mathbb{R}^d$. The parameter space $\mathcal{T}$ can be any set. For example, it can be a metric space, such as the real line $\mathbb{R}$, the half line $[0, \infty)$, or an interval $[a, b]$. In addition, it can be the non-negative integers or subsets of $\mathbb{R}^n$ for $n \geq 1$.

For each $t \in \mathcal{T}$ fixed, we have a random variable

$$\omega \mapsto X_t(\omega) \quad \text{for } \omega \in \Omega.$$

On the other hand, for each $\omega \in \Omega$ fixed, we obtain the function of t

$$t \mapsto X_t(\omega) \quad \text{for } t \in \mathcal{T},$$

which is called a *(sample) path* of X_t. It is important to note that the single event $\omega \in \Omega$ governs the outcome of X_t's at all $t \in \mathcal{T}$. Thus, we will sometime write $X(t, \omega)$ instead of $X_t(\omega)$.

In SDEs, we will focus on when $\mathcal{T}$ is a metric space such that a path of X_t is continuous with respect to t. We refer to such process as *continuous stochastic process*.

6.2.2 Wiener process

The stochastic differential equations (SDEs) are studies on a family of stochastic processes, such as $\{X_t\}_{t\in[0,T]}$, whose joint distributions are characterized by the time-evolution of X_t's. Specifically, X_t's randomness is represented by another random process called the Wiener process. To understand SDEs properly, it is important to understand the definition of the Wiener processes and its properties.

A standard one-dimensional *Wiener process*, also called *Brownian motion*, is a stochastic process $\{W_t\}_{t\geq 0}$ indexed by nonnegative real numbers t with the following properties

(i) $W_0 = 0$.

(ii) With probability 1, the function $t \mapsto W_t$ is continuous in t.

(iii) The process $\{W_t\}_{t\geq 0}$ has *stationary, independent increments.*

(iv) The increment $W_{t+s} - W_t$ follows the normal distribution with the variance t, *i.e.* $W_{t+s} - W_s \sim N(0, t)$.

Here, *independent increments* means that for any choice of nonnegative real numbers $0 < t_1 < t_2 < \cdots < t_n < \infty$.

$$W_{t_1} - W_0,\, W_{t_2} - W_{t_1},\, W_{t_3} - W_{t_2},\, \ldots,\, W_{t_n} - W_{t_{n-1}}, \tag{6.1}$$

are jointly independent. The *stationary increment* means that for any $0 \leq s < t < \infty$, the distribution $W_t - W_s$ follows the same distribution of $W_{t-s} - W_0 = W_{t-s}$.

A standard d-dimensional Wiener process is a $\mathbb{R}^d$-valued stochastic process

$$W_t = \left(W_t^{(1)}, W_t^{(2)}, \ldots, W_t^{(d)}\right),$$

whose component $W_t^{(i)}$ are independent, standard one-dimensional Wiener process.

6.2.3 Stochastic differential equations

As we mentioned, the *stochastic differential equations (SDEs)* are studies of a stochastic process, *e.g.* $\{X_t\}_{t\in[0,T]}$, especially when their joint distributions are represented by the time-evolution of X_t's. More specifically, in SDEs, the X_t's time-evolution is defined by certain types of differentiation and integration rules applied with respect to t, as its name SDEs stands for. In general, the choice of calculus rules for SDEs changes the characteristics of the stochastic processes. In this chapter, we will focus on Itô processes, which apply Itô calculus to define the behaviors of the stochastic processes.

An *Itô process*, also called *Itô diffusions*, is a stochastic process $\{X_t\}_{t\in[0,T]}$ on $(\Omega, \mathcal{F}, P)$ of the form

$$X_t = X_0 + \int_0^t f(X_s, s)\,\mathrm{d}s + \int_0^t g(X_s, s)\,\mathrm{d}W_s \tag{6.2}$$

for $t \in [0, T]$, where $\{W_t\}_{t\in[0,T]}$ is a Wiener process, $\int_0^t g(X_s, s)\,\mathrm{d}W_s$ is the Itô's integral, and $X_0 = x$ for some initial value $x \in \mathbb{R}^d$. Here, we can also assume that X_0 follows a distribution $\mathbb{P}_0$, instead of a value, and the X_0's distribution is often called as *initial distribution*. We will sometimes omit the curly bracket of a stochastic process $\{X_t\}$ for brevity.

The Itô process is more commonly written in the differential form. We say a stochastic process X_t solves the following SDEs,

$$dX_t = \underbrace{f(X_t, t)}_{=\text{ diffusion term}}\mathrm{d}t + \underbrace{g(X_t, t)}_{=\text{ drift term}}\mathrm{d}W_t \tag{6.3}$$

for $t \in [0, T]$, where W_t is a Wiener process and X_0 has an initial value or follows an initial distribution. The first term in the right-hand side $f(X_t, t)$ is often called *drift term*, and the second term $g(X_t, t)$ is called *diffusion term*.

In order to fully understand the solution process X_t, we need to discuss how to solve the two terms, $\int_0^t f(X_s, s)\,\mathrm{d}s$ and $\int_0^t g(X_s, s)\mathrm{d}W_s$. To do that, we need to introduce the Itô's integral, specifically to discuss the second term. In addition, solving the former term $\int_0^t f(X_s, s)ds$ is not straightforward as well since the integrand contains X_s, which is not deterministic.

In this chapter, however, we will avoid discussing general approaches to the solutions. It is because that in the context of generative models, the most interesting part of SDEs is its time-evolution part, which can essentially be considered as stochastic maps. In addition, in diffusion-based generative models, we are interested in transporting an initial distribution X_0 to a simple distribution, and this can be achieved by the linear SDEs. Thus, instead of general solutions, we will discuss how the solutions would look like when X_t is discretized, *i.e.* the joint distribution for some choice of nonnegative real numbers $0 < t_1 < t_2 < \cdots < t_{n-1} < t_n = T$

$$X_0,\, X_{t_1},\, X_{t_2},\, \ldots,\, X_T. \tag{6.4}$$

Moreover, we will introduce the general solution of linear SDEs when $f(X_t, t)$ is linear in X_t and $g(X_t, t)$ is independent of X_t, as such solution is a normal distribution. We refer to Oksendal (2013) for more details on the motivations, definitions, and properties of general solutions of SDEs.

6.2.4 Understand SDEs via discretization

Consider a discretized version of a stochastic process X_t that solves Equation 6.3, meaning that we will consider a subset of X_ts at time ts. Specifically, for $T > 0$, let L be the number of discretization steps, and we define a step size $\Delta t \coloneqq T/L$. We have

$$X_0, X_{\Delta t}, X_{2\Delta t}, \ldots, X_{L\Delta t}. \tag{6.5}$$

As they follows the SDE in Equation 6.3, for sufficiently small Δt, we have

$$\mathrm{d}X_t \approx X_{i+\Delta t} - X_i \tag{6.6}$$

$$f(X_t, t)\,\mathrm{d}t \approx f(X_i, i)\,\Delta t \tag{6.7}$$

$$g(X_t, t)\,\mathrm{d}W_t \approx g(X_i, i)\,(W_{i+\Delta t} - W_i) \tag{6.8}$$

for $i = 0, \ldots, (L-1)\Delta t$. Since W_t is stationary independent increment and follows normal distributions, we have

$$W_{i+\Delta t} - W_i \sim N(0, \Delta t). \tag{6.9}$$

Thus, we have $\{X_i\}_{i\in\{0,\Delta t\ldots,T\}}$ that follows

$$X_{i+\Delta t} = X_i + f(X_i, i)\,\Delta t + g(X_i, i)\,\sqrt{\Delta t}\,\varepsilon_i \tag{6.10}$$

for $i = 0, \ldots, (L-1)\Delta t$, where $\varepsilon_i \sim N(0, I)$ i.i.d. and X_0 follows an initial distribution. This can be also written as

$$X_{i+\Delta t} \sim N\left(X_i + f(X_i, i)\Delta t,\, g(X_i, i)^2\Delta t\right). \tag{6.11}$$

Equation 6.10 tells us that at each time step, we move the random variable by $f(X_i, i)\Delta t$ amount and add Gaussian noise with the amount of $g(X_i, i)^2\Delta t$. From this, we can assume that Itô diffusion is an iterative procedure that transports a distribution, where some uncertainty defined by Gaussian noises is involved. In the literature of the SDEs, the discretization in Equation 6.10 is called *Euler-Maruyama* discretization (EM-discretization).

6.2.5 Solutions of linear SDEs

To approximate the distribution of X_t, one may try to apply the update rule of Equation 6.10 iteratively from the initial value. Whilst its simplicity of the update rule Equation 6.10, even applying it twice is not straightforward, and it is intractable in general. However, one can imagine that we may be able to get closed-form solutions for X_t when the drift term and the diffusion term are simple enough. Let us consider a d-dimensional stochastic process X_t that solves

$$dX_t = (f(t)X_t + h(t))\,\mathrm{d}t + g(t)\,\mathrm{d}W_t \tag{6.12}$$

for $t \in [0, T]$, where $X_0 = x_0 \in \mathbb{R}^n$ and we have

$$f : [0, T] \to \mathbb{R}, \quad a : [0, T] \to \mathbb{R}, \quad g : [0, T] \to \mathbb{R}.$$

Then, under mild assumptions on f, a, and g, we can show that X_t is a normal distribution for a given X_0. More specifically, we have

$$X_t \,|\, X_0 \sim N\left(\mu(X_0, t), \sigma^2(t)\right), \tag{6.13}$$

where the mean function $\mu : \mathbb{R}^d \times [0, T] \to \mathbb{R}$ and variance function $\sigma^2 : [0, T] \to \mathbb{R}$ are defined by

$$\mu(X_0, t) = \alpha(t)X_0 + \alpha(t) \int_0^t \alpha(s)^{-1} h(s)\, \mathrm{d}s \tag{6.14}$$

$$\sigma^2(t) = \alpha^2(t) \int_0^t \alpha(s)^{-2} g^2(s)\, \mathrm{d}s, \tag{6.15}$$

and $\alpha(t) = \exp\left(\int_0^t f(s)ds\right)$.Note that there exist the closed-form solutions for linear SDEs when f, a and g are multivariate functions. However, we will stick to the above cases for simplicity.

One can view that $h(t)$ modulates the mean of sample paths, and if $h(t) = 0$ for all t, we have

$$\mu(X_0, t) = \alpha(t)X_0. \tag{6.16}$$

This implies that with the linear SDEs, X_t is simply adding an accumulated Gaussian noise with the amount of $\sigma^2(t)$ to a random variable X_0. From now on, we will omit $h(t)$ when we refer to the linear SDEs.

Example 6.1. Let X_t be a stochastic process that solves a SDE of Equation 6.12 for $t \in [0, T]$, and assume $X_0 = x_0 \in \mathbb{R}^n$. If $h(t) = 0$, $f(t) = -\frac{1}{2}$, and $g(t) = 1$ for $\forall t \in [0, \infty]$, then

$$\alpha(t) = \exp\left(-0.5t\right) \quad \text{and} \quad \sigma^2(t) = 1 - \exp(-t).$$

Moreover, regardless of the initial condition X_0, we get $\lim_{t\to\infty} \alpha(t) = 0$ and $\lim_{t\to\infty} \sigma^2(t) = 1$, and thus $\lim_{t\to\infty} X_t \sim N(0, I)$.

This example shows that the solution X_t of linear SDEs (with appropriate drift and diffusion terms) converges to a normal distribution regardless of the initial distribution. This can also be interpreted as the solution of the linear SDE being a stochastic map that transports an arbitrary continuous random variable to a simple distribution, such as the standard normal distribution.

6.2.6 Fokker-Planck equation

In the previous sections, we have discussed how an SDE characterizes a stochastic process X_t. It is useful to know that the probability density functions of X_t are also a solution to a differential equation. A diffusion process X_t that solves an SDE induces a family of densities $p(\cdot, t)$ for $t \in [0, T]$. And the density functions follow *Fokker-Planck equation*, also called *Kolmogorov forward equation*,

$$\begin{aligned}
\frac{\partial p(x,t)}{\partial t} &= \underbrace{-\nabla \cdot \left(f(x,t)\, p(x,t)\right)}_{\text{first-order}} + \underbrace{\frac{1}{2} \sum_{i,j} \frac{\partial^2}{\partial x_i \partial x_j} \left(G(x,t)_{ij} p(x,t)\right)}_{\text{second-order}} \\
&\overset{\dagger}{=} -\nabla \cdot \left(f(x,t)\, p(x,t)\right) + \frac{1}{2} \nabla \cdot \left[\left\{\nabla \cdot G(x,t) + G(x,t) \nabla \log p(x,t)\right\} p(x,t)\right],
\end{aligned} \tag{6.17}$$

where $G(x,t) \coloneqq g(x,t)g(x,t)^\top$ and the †-identity for the second-order term is due to

$$\begin{aligned}\frac{1}{2}\sum_{i,j}\frac{\partial^2}{\partial x_j \partial x_i}(G(x,t)_{ij}p(x,t)) &= \frac{1}{2}\sum_j \frac{\partial}{\partial x_j}\left\{p(x,t)\sum_i \frac{\partial}{\partial x_i}G(x,t)_{ij} + \sum_i G(x,t)_{ij}\frac{\partial}{\partial x_i}p(x,t)\right\} \\ &= \frac{1}{2}\sum_j \frac{\partial}{\partial x_j}\{p(x,t)[\nabla\cdot G(x,t)]_j + p(x,t)[G(x,t)\nabla \log p(x,t)]_j\} \\ &= \frac{1}{2}\nabla\cdot\left[\{\nabla\cdot G(x,t) + G(x,t)\nabla\log p(x,t)\}\, p(x,t)\right].\end{aligned}$$

For the following sections, we may write $p_t(x)$ instead of $p(x,t)$.

6.3 Diffusion-based generative models

So far, we have discussed important building blocks to diffusion-based generative models. One key component to their success is that they provide efficient training methods for extremely deep generative models. In this subsection, we will explore in-depth discussions of diffusion-based generative models, covering their motivation, definition, training methods, and other ways to define the same models.

6.3.1 Continuous-time score-based generative models

In this chapter, we will first discuss the diffusion-based generative models from the perspective of continuous-time score-based generative models (Song et al., 2020) (see Figure 6.1a). Interestingly, this definition is much closer to the original motivation of the very first diffusion-based generative models, *i.e.* diffusion probabilistic models (DPM, Sohl-Dickstein et al., 2015), compared to other perspectives. After that, we will cover how the continuous-time score-based models are equivalent to denoising diffusions (Figure 6.1b) in Section 6.3.2 and the annealed Langevin dynamics-based models (Figure 6.1c) in Section 6.3.3.

6.3.1.1 Forward and reverse diffusions

In order to discuss the key observations highlighted by Sohl-Dickstein et al. (2015), we will discuss the following theorem first. The theorem states that even with a simple linear SDE, we can map any (data) distribution to normal distributions. Moreover, there exists another diffusion that reverts back to the above mapping, running from the normal distribution to the data distribution. Interestingly, the goal of generative modeling is to build such mappings, which the latter diffusions perform.

Theorem 6.1. [Existence of time-reversed diffusions in Euclidean spaces] *Let X_t be a d-dimensional stochastic process that solves a SDE for $t \in [0,T]$,*

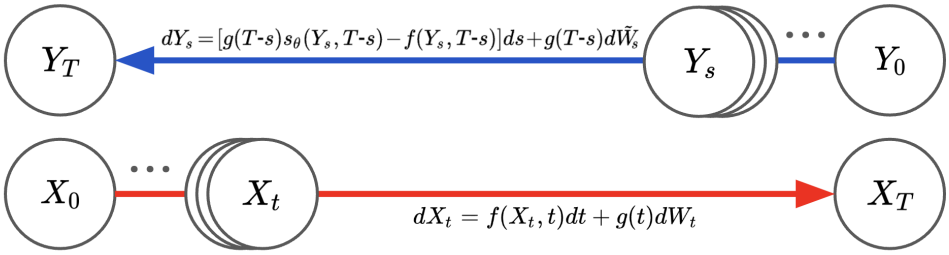
$$dX_t = f(X_t,t)\,\mathrm{d}t + g(t)\,\mathrm{d}W_t, \tag{6.18}$$

where W_t is a Wiener process and $X_0 \sim q_0$. The pdfs of X_t's are written by $q(x,t)$.

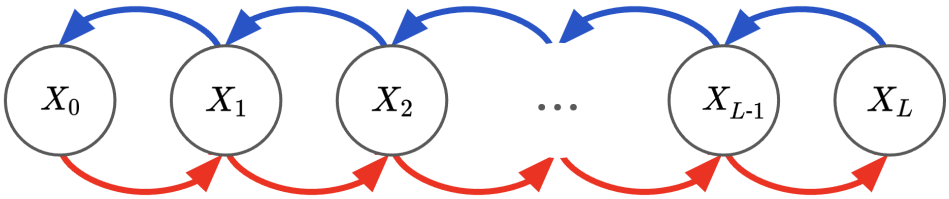
Then, under some mild assumptions on f and g, there exists a stochastic process Y_s indexed by $s = T - t$, which solves the following SDE for $s \in [0,T]$.

$$dY_s = \left[g(T\text{-}s)^2\nabla\log q(Y_s, T\text{-}s) - f(Y_s, T\text{-}s)\right]\,\mathrm{d}s + g(T\text{-}s)\,\mathrm{d}\tilde{W}_s, \tag{6.19}$$

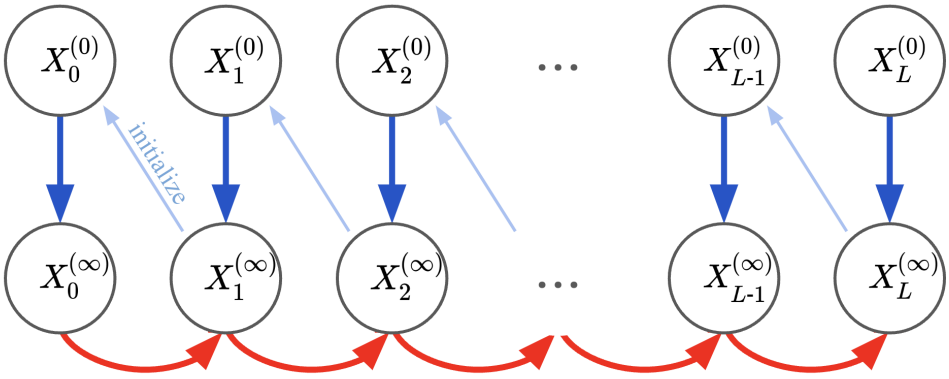
where $\tilde{W}_s$ is a Wiener process that runs in backward (s-increasing direction), and $q_{T\text{-}s}$ is the pdf of $X_{t=T\text{-}s}$. And Y_s equals to $X_{t=T\text{-}s}$ in distribution for all $s \in [0,T]$.



(a) Continuous-time score-based generative models.



(b) Denoising diffusion-based generative models.



(c) Annealed Langevin dynamics-based generative models.

Figure 6.1: Various perspectives of diffusion-based generative models. The **blue** directed acyclic graphs illustrate the reverse/denoising processes. The **red** graphs describe the forward/perturbation processes.

Proof. Assume we have a stochastic process $\{Y_s\}_{s\in[0,T]}$ follows probability density functions $p(y,s)$ for $s \in [0,T]$. We further assume that $p(y,s) = q(y, T\text{-}s)$ for all $s \in [0,T]$. Then, we have

$$\begin{aligned}
\frac{\partial p(y,s)}{\partial s} &= (-1)\frac{\partial q(x,t)}{\partial t}\Big|_{t=T\text{-}s} \\
&\overset{*}{=} -\nabla\cdot(-f(x,t)q(x,t)) - \frac{1}{2}g(t)^2\,\nabla\cdot\Big[\{\nabla\log q(x,t)\}q(x,t)\Big]\Big|_{t=T\text{-}s} \\
&= -\nabla\cdot\left[\left\{g(t)^2\nabla\log q(x,t) - f(x,t)\right\}q(x,t)\right] \\
&\qquad\qquad + \frac{1}{2}\nabla\cdot\left[\left\{g(t)^2\nabla\log q(x,t)\right\}q(x,t)\right]\Big|_{t=T\text{-}s} \\
&= -\nabla\cdot\Big[\left\{g(T\text{-}s)^2\nabla\log q(x,T\text{-}s) - f(x,t)\right\}q(x,t)\Big] \\
&\qquad\qquad + \frac{1}{2}\nabla\cdot\left[\left\{g(T\text{-}s)^2\nabla\log q(x,t)\right\}q(x,t)\right],
\end{aligned}$$

where $(*)$ uses the Fokker-Planck equation for q. Then, $p(y,s)$ is a Fokker-Planck equation for a stochastic process that solves a SDE

$$dY_s = \left[g(T\text{-}s)^2\nabla\log q(Y_s, T\text{-}s) - f(Y_s, T\text{-}s)\right]\,\mathrm{d}t + g(T\text{-}s)\,\mathrm{d}\tilde{W}_s,$$

where $\tilde{W}_s$ is a Wiener process. □

Note that we use a position-independent diffusion term $g(x)$ for the brevity of the theorem and proof. The above theorem can be extended to more general forward diffusions, for example, when the diffusion term in Equation 6.18 is $g(X_t, t)$ instead of $g(x)$.

Theorem 6.1 holds for any initial distribution p_0 on Euclidean spaces. Therefore, the theorem provides the two key insights to construct the diffusion-based generative models as discussed by Sohl-Dickstein et al. (2015). First, even with the forward diffusion in Equation 6.18 is linear, it can transport arbitrarily complex data distribution to normal distributions (See Example 6.1). Second, if we can approximate the reverse diffusion in Equation 6.19 well, we are able to generate the original data distribution by solving the SDE to get Y_T, starting from $Y_0(= Y_T)$; Y_0 is a normal distribution, when Equation 6.18 is linear.

In this theorem, Equation 6.18 and Equation 6.19 are the essential components to define the diffusion-based generative models, and we refer to them as *forward* and *reverse diffusions*, respectively. In the following section, we will discuss the definition of continuous-time score-based generative models.

6.3.1.2 Score-based generative models

Let X_t be a d-dimensional forward diffusion that solves a SDE

$$\mathrm{d}X_t = f(t)X_t\,\mathrm{d}t + g(t)\,\mathrm{d}W_t \tag{6.20}$$

for $t \in [0,T]$, where W_t is a Wiener process and X_0 follows a data distribution q_0. In particular, $f(t)$ is defined to satisfy $\alpha(T) \approx 0$, where $\alpha(t) = \exp\left(\int_0^t f(s)ds\right)$ appeared in Equation 6.14. The pdfs of X_t at t are written as $q(x,t)$.

Next, we define a stochastic process Y_s indexed by $s = T - t$, which solves the following SDE for $s \in [0,T]$

$$\mathrm{d}Y_s = \left[g(T\text{-}s)^2 s_\theta(Y_s, T\text{-}s) - f(T\text{-}s)Y_s\right]\,\mathrm{d}s + g(T\text{-}s)\,\mathrm{d}\tilde{W}_s, \tag{6.21}$$

where $\tilde{W}_s$ is a Wiener process that runs in s-increasing direction, and Y_0 assumes to follow a normal distribution. We denote its pdfs by $p(y,s)$ for all $s \in [0,T]$.

Then, we refer to the solution Y_T for Equation 6.21 as the *diffusion-based generative models* or *(continuous-time) score-based generative models* (Song et al., 2020). To train the generative model, one can consider to train s_θ to

approximate the score of the density p's at all times. We will discuss the training methods further in the following sections.

6.3.1.3 Learning by score matching

As generative modeling, our goal is to make the generative diffusion in Equation 6.21 to approximate the ground truth reverse diffusion for a given forward diffusion and data distribution. As we briefly mentioned in the previous section, the most straightforward way is to approximate $\nabla \log q(x,t)$ by $s_\theta(x,t)$ for all $t \in [0,T]$. To do that, we can use score matching objectives.

$$\mathcal{L}_{\mathrm{SM}}(s_\theta) = \int_0^T \mathbb{E}\Big[\|\nabla \log q(X_t,t) - s_\theta(X_t,t)\|_2^2\Big]\ \mathrm{d}t, \tag{6.22}$$

where X_0 follows the data distribution and X_t is the solution of the forward diffusion defined in Equation 6.20.

Remind that X_t solves a linear SDE in Equation 6.20, and thus, $X_t|X_0$ follows a normal distribution (see Equation 6.13); equivalently we write

$$X_t = \alpha_t X_0 + \sigma_t \varepsilon,$$

where $\varepsilon \sim N(0,I)$, $\alpha_t = \alpha(t) = e^{\int_0^t f(\tau)d\tau}$, and $\sigma_t^2 = \sigma^2(t) = \alpha_t^2 \int_0^t \alpha_\tau^{-2} g^2(\tau) d\tau$ (see Equations 6.14 and 6.15). Importantly, $X_t|X_0$ being a normal distribution implies that denoising score matching (DSM) can be used. Thus, the resulting objective function is the DSM objective accumulated in $t \in [0,T]$ and it is written as

$$\mathcal{L}_{\mathrm{DSM}}(s_\theta) = \int_0^T \mathbb{E}\Big[\|\nabla \log q(X_t|X_0) - s_\theta(X_t,t)\|_2^2\Big]\ \mathrm{d}t \tag{6.23}$$

$$= \int_0^T \mathbb{E}\left[\left\|-\frac{1}{\sigma_t^2}\big(X_t - \alpha_t X_0\big) - s_\theta(X_t,t)\right\|_2^2\right]\ \mathrm{d}t \tag{6.24}$$

$$= \int_0^T \mathbb{E}\left[\left\|\frac{1}{\sigma_t}\varepsilon + s_\theta(X_t,t)\right\|_2^2\right]\ \mathrm{d}t. \tag{6.25}$$

Here, it is possible to define each model at t (or s) independently; however, it is impractical due to memory cost. Typically, time-conditional models are used (Ho et al., 2020; Song & Ermon, 2019; Song et al., 2020).

6.3.1.4 Learning by maximum likelihood

The above training method is to use the approximate scores as drop-in replacements to the ground truth in Equation 6.19. This plug-in reverse diffusion will generate the data distribution if s_θ perfectly models the ground truth for all ts. However, the objective does not estimate the discrepancy between the data distribution and the model X_T, which is the solution of the plug-in reverse diffusion. Thus, it is difficult to tell how close X_T is, especially when s_θ is not perfectly fit to the ground truth.

Huang et al. (2021); Kingma et al. (2021); Song et al. (2021b) address this problem and show that for a given data x, the ELBO of the plugin-reverse diffusion models is the weighted score matching objective. Here, we follow Huang et al. (2021) to get the ELBO.

Theorem 6.2. [Continuous-time ELBO (Huang et al., 2021)] *Let X_t and Y_s be the solutions of the SDEs defined by Equation 6.20 and Equation 6.21, respectively. We denote the path measure of X_t by $\mathbb{Q}$ and the one with the initial value x, i.e. $X_0 = x$, denoted by $\mathbb{Q}_{|0}$.[a] Moreover, we write $\mathbb{P}$ for the Y_s's path measure. For the densities of X_t and Y_s, we write $q(x,t)$ and $p(y,s)$—or $q_t(x)$ and $p_s(y)$—for all $t, s \in [0,T]$, respectively. Note that $t = T$- s for $s \in [0,T]$. Then, the log-likelihood of a data x of the model Y_T, i.e. $\log p(x,T)$, is lower-bounded by the ELBO*

$\mathcal{E}(x;\mathbb{P},\mathbb{Q}_{|0})$ and it is defined by

$$\log p(x,T) \geq \mathbb{E}\Bigg[\log p_0(X_T) - \frac{1}{2}\int_0^T \left\|g(t)^2 s_\theta(X_t,t)\right\|_2^2 \,\mathrm{d}t - \int_0^T \nabla\cdot\Big(g(t)^2 s_\theta(X_t,t) - f(t)X_t\Big)\,\mathrm{d}t \,\Bigg|\, X_0 = x\Bigg] =: \mathcal{E}(x;\mathbb{P},\mathbb{Q}_{|0}),$$

where the expectation is taken with respect to $\mathbb{Q}_{|0}$.

[a] Thus, $\mathbb{Q}$ is the product measure of $\mathbb{Q}_{|0}$ and the data distribution.

Interestingly, the expected value of the ELBO is written in a much simpler form. Moreover, this form shows a strong connection between the MLE and the above score matching-based training. The following corollary states that maximizing the expected ELBO is equivalent to minimizing the weighted score matching objective.

Corollary 6.3. *The expectation of the ELBO over the data distribution is the weighted score matching objective and θ-independent terms.*

$$\begin{aligned}
\mathbb{E}[\mathcal{E}(X_0)] &= \mathbb{E}\left[\log p_0(X_T) - \frac{1}{2}\int_0^T \left\|g(t)^2 s_\theta(X_t,t)\right\|_2^2 \,\mathrm{d}t - \int_0^T \nabla\cdot\Big(g(t)^2 s_\theta(X_t,t) - f(t)X_t\Big)\,\mathrm{d}t\right] \\
&= \mathbb{E}\Bigg[\overbrace{\log p_0(X_T) + \int_0^T \nabla\cdot f(t)X_t\,\mathrm{d}t + \frac{1}{2}\int_0^T g(t)^2\left\|\nabla\log q(X_t,t)\right\|^2\mathrm{d}t}^{=\ \theta\text{-independent}} \\
&\qquad\qquad - \int_0^T g(t)^2\left\|s_\theta(X_t,t) - \nabla\log q(X_t,t)\right\|_2^2\,\mathrm{d}t\Bigg], \\
&= -\mathbb{E}\left[\int_0^T g(t)^2\left\|s_\theta(X_t,t) - \nabla\log q(X_t,t)\right\|_2^2\,\mathrm{d}t\right] + (\textit{θ-independent term}).
\end{aligned}$$

Here, we omitted the path measure arguments of $\mathcal{E}$; thus, $\mathcal{E}(x) = \mathcal{E}(x;\mathbb{P},\mathbb{Q}_{|0})$.

Proof of Theorem 6.2 and Corollary 6.3. The proof follows Huang et al. (2021). Consider the model's diffusion Y_s. Here, we will denote its drift term by μ as in

$$\mathrm{d}Y_s = \underbrace{\big[g(T\text{-}s)^2 s_\theta(Y_s, T\text{-}s) - f(T\text{-}s)Y_s\big]}_{=:\,\mu(Y_s,T\text{-}s)}\mathrm{d}s + g(T\text{-}s)\,\mathrm{d}\tilde{W}_s.$$

Let X'_t be a diffusion that runs in the t-increasing direction. In particular, it is the solution of

$$\mathrm{d}X'_t = -\mu(X'_t,t)\,\mathrm{d}t + g(t)\,\mathrm{d}W'_t.$$

By using Feynman-Kac formula, the Y_s's density for an input x, *i.e.* $= p(x, T\text{-}t)$, can be written by

$$p(x,T) = \mathbb{E}\left[p(X'_T,0)\exp\left(-\int_0^T \nabla\cdot\mu(X'_t,t)\,\mathrm{d}t\right)\,\middle|\, X'_0 = x\right],$$

where the expectation is taken under X'_t with the initial condition $X'_0 = x$.

Moreover, the expectation can be computed by using X_t instead of X'_t. For that, we get

$$p(x,T) = \mathbb{E}\left[p(X_T,0)\exp\left(-\int_0^T \nabla\cdot\mu(X_t,t)\,\mathrm{d}t\right)\frac{d\mathbb{Q}'_{|0}}{d\mathbb{Q}_{|0}}\,\middle|\, X_0 = x\right],$$

where $d\mathbb{Q}'_{|0}/d\mathbb{Q}_{|0}$ is the RN-derivative of X'_t's path measure with respect to the X_t's of Equation 6.20; here, the RN-derivative is a function of a sample path with the initial value x. By using the Girsanov, we write $d\mathbb{Q}'_{|0}/d\mathbb{Q}_{|0}$ as

$$\frac{d\mathbb{Q}'_{|0}}{d\mathbb{Q}_{|0}} = \exp\left(-\int_0^T \frac{1}{g}\left(g^2 s_\theta(X_t,t)\right)\mathrm{d}W_t - \frac{1}{2}\int_0^T \frac{1}{g^2}\left\|g^2 s_\theta(X_t,t)\right\|^2\mathrm{d}t\right).$$

In particular, by using Jensen's inequality, $\log p(x,T)$ can be lower bounded as in

$$\begin{aligned}\log p(x,T) \;\geq\; \mathbb{E}\Bigg[\log p_0(Y_T) - \frac{1}{2}\int_0^T \left\|g(t)^2 s_\theta(X_t,t)\right\|_2^2\,\mathrm{d}t \\ - \int_0^T \nabla\cdot\Big(g(t)^2 s_\theta(X_t,t) - f(t)X_t\Big)\,\mathrm{d}t\,\Bigg|\, X_0 = x\Bigg] \;=:\; \mathcal{E}(x).\end{aligned}$$

Again, we omitted the path measure arguments of $\mathcal{E}$; thus, $\mathcal{E}(x) = \mathcal{E}(x;\mathbb{P},\mathbb{Q}_{|0})$.

Finally, for the integrand to have the divergence operator, we apply Fubini's theorem to change the order of the integration and the expectation, and then apply the integration-by-parts to the divergence term, we get the rest for the Corollary 6.3. □

Theorem 6.2 and Corollary 6.3 state that maximizing the expected ELBO is equivalent to minimizing the weighted score matching objective. In addition, we can observe that the optimal solution will become the true scores regardless of the weights, since for the optimal solution $\|s_\theta(X_t,t) - \nabla\log q(X_t,t)\|_2^2 = 0$ for all ts. In this perspective, we may be able to modify the weights in terms of some criteria, such as reducing the variance of the MC estimators. Some recent works study the effects of such weighting of the score matching objectives, and for the in-depth discussions on this topic, we would like to refer to Nichol & Dhariwal (2021); Karras et al. (2022); Kingma et al. (2021); Kingma & Gao (2024).

6.3.1.5 Generation

The score function s_θ of the score-based generative models in Equation 6.21 can be trained using either score matching or maximum likelihood. To sample Y_T from the trained model, it is necessary to solve the SDE in Equation 6.21 starting from $Y_0 \sim N(0,I)$. Generally, to achieve this, the Euler-Maruyama method, as described in Equation 6.10, is used to obtain numerical solutions. To get the solution for Equation 6.21, consider a subset of $\{Y_s\}_{s\in[0,T]}$; that is

$$Y_0, Y_{\Delta s}, Y_{2\Delta s}, \ldots, Y_{L\Delta s},$$

where L is the number of discretization steps over the interval $[0,T]$ and $\Delta s = T/L$ is the step size. Then, starting from the initial samples $y_0 \sim N(0,I)$, repeating the following update rule will give us the samples of Y_T;

$$y_{i+\Delta s} = y_i + \left[g(T\text{-}s)^2 s_\theta(y_i, T\text{-}s) - f(T\text{-}s)y_i\right]\,\Delta s + g(T\text{-}s)\,\sqrt{\Delta s}\,\varepsilon_i \tag{6.26}$$

for $i = 0,\ldots,(L-1)\Delta s$, where $\varepsilon_i \overset{\text{i.i.d.}}{\sim} N(0,I)$.

6.3.1.6 Probability flow ODE

Theorem 6.1 establishes that the marginal distributions of the forward diffusion in Equation 6.20 and the reverse diffusion in Equation 6.21 are identical for all ts. However, beyond this reverse diffusion, there exists a family of stochastic processes that is marginally equivalent to the forward diffusion. Here, the term 'marginally equivalent' indicates that the marginal distributions of the two processes are identical for all ts.[1]

Importantly, there exists a marginally equivalent stochastic process that can be represented as an ODE rather than an SDE. Along with introducing continuous-time score-based generative models, Song et al. (2020) introduced this deterministic process, which they termed *probability flow ODEs*. This process is define as follows. For $t \in [0, T]$, let x_t solve the following ODEs

$$\frac{\mathrm{d}x_t}{\mathrm{d}t} = f(t)\, x_t - \frac{g^2(t)}{2} \nabla \log q(x_t, t), \tag{6.27}$$

where x_0 is the initial value and a sample from the data distribution. By reversing the direction of time, this ODE can also be expressed as follows;

$$\frac{\mathrm{d}y_s}{\mathrm{d}s} = -f(T\text{-}s)\, y_s + \frac{g^2(T\text{-}s)}{2} \nabla \log q(y_s, T\text{-}s) \quad \text{for } s \in [0, T]. \tag{6.28}$$

Similar to score-based generative models, samples of a probability flow ODE can be generated by solving the ODE, whose $\nabla \log q(x_t, t)$ term is substituted by the learned score. Note that in order to use Equation 6.27, one needs to solve the equation from $t = T$ to $t = 0$.

A notable distinction from reverse SDEs is that probability flow ODEs offer the advantage of using ODE solvers that have been extensively employed in the context of Neural ODEs. For instance, Song et al. (2020) utilizes the general-purpose Runge-Kutta methods (Dormand & Prince, 1980).

To demonstrate how probability flow ODEs are marginally equivalent to the forward process, we can use the Fokker-Planck equation again, as in the proof of Theorem 6.1. For the proof, let's utilize a slightly generalized version of the approach by Huang et al. (2021). For $\lambda \in [0, 1]$, let a stochastic processes X_t be the solution of a SDE

$$\mathrm{d}X_t = \underbrace{\big[\, f(X_t, t) - \frac{\lambda}{2} g(t)^2 \nabla \log q(X_t, t) \,\big]}_{=: f_\lambda(x,t)} \mathrm{d}t + \underbrace{\sqrt{1-\lambda}\, g(t)}_{=: g_\lambda(t)} \mathrm{d}W_t \quad \text{for } t \in [0, T],$$

where W_t is a standard Wiener process. We can show that for all $\lambda \in [0, 1]$, the above processes are marginally equivalent to the forward diffusion in Equation 6.20, since the process shares the same Fokker-Planck equation as in

$$\begin{aligned}
\frac{\partial q(x,t)}{\partial t} &= -\nabla \cdot (f(x,t)\, q(x,t)) + \frac{1}{2}\, g(t)^2\, \nabla \cdot \big[\nabla \log q(x,t)\, q(x,t)\big] \\
&= -\nabla \cdot \Big(\overbrace{\big[f(x,t) - \frac{\lambda}{2} g(t)^2 \nabla \log q(x,t)\big]}^{= f_\lambda(x,t)}\, q(x,t) \Big) \\
&\qquad + \frac{1}{2} \underbrace{(1-\lambda) g(t)^2}_{= g_\lambda^2(t)} \nabla \cdot \big[\nabla \log q(x,t)\, q(x,t)\big].
\end{aligned}$$

In particular, $\lambda = 1$ corresponds to the probability flow ODE in Equation 6.27.

[1] See Huang et al. (2021) for the definition of marginally equivalent processes.

6.3.2 Denoising diffusion perspective

To this point, we have focused on the reverse diffusion perspective of the diffusion-based generative models, *i.e.* continuous-time score-based generative models. However, pioneering works by Sohl-Dickstein et al. (2015); Ho et al. (2020) have adopted different approaches. While they are motivated by Theorem 6.1, they rather focus on the operational aspects of the reverse diffusions, specifically on how the reverse diffusion reverts back X_t at each time step. Briefly speaking, one can say the reverse diffusion denoises its input by a small amount at a time, reverting back the perturbations performed by the forward diffusion. In this section, we will discuss the perturbation-denoising perspective of the forward-reverse diffusions (see Figure 6.2b). Especially, this interpretation can be understood easily when we discretize the stochastic processes.

6.3.2.1 Discrete-time ELBO

Consider a subset of a forward diffusion X_t that solves Equation 6.20 for $t \in [0, T]$. Let $L \in \mathbb{N}$ and $\Delta t \coloneqq T/L$ be a step size. For some choice of nonnegative real numbers

$$0 < \Delta t < 2\Delta t < \cdots < (L-1)\Delta t < L\Delta t = T,$$

we have X_0, $X_{\Delta t}$, $X_{2\Delta t}$, …, $X_{L\Delta t}$. As we discussed in Equation 6.10, for sufficiently small Δt the differential equation can be approximated as

$$\begin{aligned} X_{i+\Delta t} &= X_i + f(i)\,\Delta t X_i + g(i)\,\sqrt{\Delta t}\,\varepsilon_i \\ &= (1 + f(i)\,\Delta t)\,X_i + g(i)\,\sqrt{\Delta t}\,\varepsilon_i \end{aligned}$$

for $i = 0, \ldots, (L-1)\Delta t$, where $\varepsilon_i \overset{\text{i.i.d.}}{\sim} N(0, I)$ and X_0 follows an initial distribution. For the brevity of the notations, we use $l \coloneqq i/\Delta t$ for $i = 0, \Delta t, \ldots, L\Delta t$. Then, we write

$$\begin{aligned} &X_0,\, X_1,\, X_2,\, \ldots,\, X_L, \text{ and} \\ &X_{l+1} = a_{l+1}X_l + b_{l+1}\varepsilon_{l+1} \quad \text{for } l = \{0, \ldots, L\text{-}1\}, \end{aligned} \tag{6.29}$$

where $\varepsilon_{l+1} \overset{\text{i.i.d.}}{\sim} N(0, I)$, $a_{l+1} \coloneqq 1 + f(l\Delta t)\Delta t$, and $b_{l+1} \coloneqq g(l\Delta t)\sqrt{\Delta t}$. Here, for the sake of discussing the connection to continuous-time, we choose to define a_{l+1} and b_{l+1} as values corresponding to the EM-discretization. However, the derivations below remain valid even for arbitrary $a_{l+1} > 0$ and $b_{l+1} > 0$.

Here, our goal is to build a model that reverts back to the above disruptions. Theorem 6.1 tells us that there exists a diffusion, whose marginal distribution is equal in-distribution to the forward at all times. If we focus on the existence of such diffusion regardless of how the ground truth would look like, we can start with defining a diffusion that runs from a prior in t-decreasing direction. Moreover, we know that a diffusion can be approximated in discrete-time by a finite-step Markov chain, whose transition probability is the normal distribution. Acknowledging this, we define our model as

$$p_\theta(x_0, x_1, \ldots, x_L) = p(x_L) \prod_{l=0}^{L-1} p_\theta(x_l | x_{l+1}) \tag{6.30}$$

$$p_\theta(x_l | x_{l+1}) = N\left(x_l; \mu_\theta(x_{l+1}, l{+}1), \sigma_\theta(l{+}1)^2 I\right), \tag{6.31}$$

where $p(x_L)$ is a prior's density such as the standard normal distribution, and X_0 follows the data distribution. Then,

we write the ELBO for a given data observation x by

$$\begin{aligned}\log p_\theta(x) &\geq \mathbb{E}\left[\log\frac{p_\theta(X_0, X_1, \ldots, X_L)}{q(X_1, \ldots, X_L|X_0)}\middle| X_0 = x\right] \\ &= \mathbb{E}\left[\sum_{i=0}^{L-1}\log\frac{p_\theta(X_l|X_{l+1})}{q(X_{l+1}|X_l)} + \log p(X_L)\middle| X_0 = x\right] =: \mathcal{E}^L(x; p_\theta, q),\end{aligned} \tag{6.32}$$

where the expectation is computed on $q(x_1, \ldots, x_L|x)$.

6.3.2.2 A Bayesian inverse problem at each time step

Having that the forward diffusion adds a Gaussian noise to its input at each time step, learning the normal distribution $p_\theta(x_l|x_{l+1})$ looks similar to solving the inverse problem of DAE's in Equation 5.12. A learned model will eventually be diffusions, which is a chain of Guassians; in particular, they denoise the perturbations defined by the forward diffusion. Hence, we call the models denoising diffusions.

To explore this perspective, consider the l-th log probability ratio element of the summation term in Equation 6.32, *i.e.* $\log p_\theta(x_l|x_{l+1}) - \log q(x_{l+1}|x_l)$. Note that $\log q(x_{l+1}|x_l)$ is θ-independent. Thus, for d-dimensional case, we will focus on

$$\log p_\theta(x_l|x_{l+1}) = -\frac{d}{2}\log 2\pi - \log\det(\sigma_\theta(l{+}1)) - \frac{1}{2\sigma_\theta(l{+}1)^2}\|x_l - \mu_\theta(x_{l+1}, l{+}1)\|^2.$$

Thus, we write its expected value for $l = 0, \ldots, L$-1 as in

$$\mathbb{E}[\log p_\theta(X_l|X_{l+1})] = -\mathbb{E}\left[\log\det(\sigma_\theta(l{+}1)) + \frac{1}{2\sigma_\theta(l{+}1)^2}\|X_l - \mu_\theta(x_{l+1}, l{+}1)\|^2\right] - \frac{d}{2}\log 2\pi,$$

where the expectation is taken with respect to the joint distribution $q(x_l, x_{l+1})$.

As a result, maximizing the discrete-time ELBO becomes training a collection of denoising autoencoders, equivalently solving a collection of Bayesian inverse problems. Note that we omit to treat the effect of σ_θ in the training to simplify the discussion in this section.

Since the entire training becomes solving a collection of Bayes inverse problems, we can parameterize $\mu_\theta(x_{l+1}, l{+}1)$ in more efficient manners by mirroring the Bayes estimator's form. In particular, using Tweedie's formula as described in Theorem 5.1, the models can be parameterized as in

$$\mu_\theta(x_{l+1}, l{+}1) \coloneqq \frac{1}{a_{l+1}}\Big(x_{l+1} + b_{l+1}^2 s_\theta(x_{l+1}, l{+}1)\Big). \tag{6.33}$$

Moreover, when we write the update rule defined by $p_\theta(x_l|x_{l+1})$ with the new parameterization, it is written by

$$X_l = \frac{1}{a_{l+1}}\Big(X_{l+1} + b_{l+1}^2 s_\theta(X_{l+1}, l{+}1)\Big) + \sigma_\theta(l{+}1)\tilde{\varepsilon}_l \tag{6.34}$$

for $l = 0, \ldots, L-1$, where $\tilde{\varepsilon}_l \overset{\text{i.i.d.}}{\sim} N(0, I)$ and X_L is from the prior. Note that this resembles the discretization of Equation 6.21, which is Equation 6.26; however, our update is written in the t-decreasing direction.

6.3.2.3 Denoising diffusion probabilistic models

On top of the initial denoising diffusion models of Sohl-Dickstein et al. (2015), Ho et al. (2020) have made substantial contributions to the development of diffusion-based models, which have played a key role in establishing the current prominence of this model family. Among their numerous contributions, two aspects are particularly important for a deeper theoretical understanding of diffusion-based models:

(i) "variance reduction by conditioning" for the ELBO estimations, and

(ii) the ε-parameterization of $\mu_\theta(x_{l+1}, l{+}1)$.

A detailed examination of these aspects will follow.

Variance reduction by conditioning The first key recipe from their works is to compute the ELBO from Equation 6.32 with reduced variance, which ultimately accelerates the entire training procedure. This low variance estimator has significantly contributed to the scalability of denoising diffusion models. To do that, they first re-arrange the ELBO in Equation 6.32, which is written by

$$\mathcal{E}^L(x_0; p_\theta, q) = \mathbb{E}\left[\underbrace{\log p_\theta(X_0|X_1)}_{\text{Reconstruction loss}} + \underbrace{\sum_{i=1}^{L-1} \log \frac{p_\theta(X_l|X_{l+1})}{q(X_l|X_{l+1}, X_0)}}_{\text{Diffusion loss}} + \underbrace{\log \frac{p(X_L)}{q(X_L|X_0)}}_{\text{Prior loss}} \,\middle|\, X_0 = x_0 \right]. \tag{6.35}$$

For computing the ELBO, Sohl-Dickstein et al. (2015) points out that $q(x_l|x_{l+1}, x_0)$ is a normal distribution. Combining this with the fact that the expected value of each l-th log probability ratio in the diffusion loss is a negative KL, we get the closed-form solution of the KL between two Gaussians. Consequentially, the variance of the resulting ELBO estimator will be reduced compared to the original one since the randomness of X_l is integrated out when x_{l+1} and x_0 are given.

To show $q(x_l|x_{l+1}, x_0)$ is a Gaussian, recall that

$$q(x_l|x_{l+1}, x_0) \propto \underbrace{q(x_{l+1}|x_l, x_0)}_{= q(x_{l+1}|x_l)} q(x_l|x_0). \tag{6.36}$$

The first term is a normal distribution by definition. For the second term, we can show that $q(x_l|x_0)$ is also a normal distribution by iterating the update rule in Equation 6.29; that is,

$$X_l = \alpha_l X_0 + \sigma_l \varepsilon'_l, \tag{6.37}$$

$$\alpha_l \coloneqq \prod_{i=1}^{l} a_i\,, \quad \text{and} \quad \sigma_l^2 \coloneqq \sum_{i=1}^{l} b_i \Big(\prod_{j=i+1}^{l} a_j^2 \Big) = b_l^2 + a_l^2 \sigma_{l\text{-}1}^2, \tag{6.38}$$

where $\varepsilon' \overset{\text{i.i.d.}}{\sim} N(0, I)$ for all l's. Moreover, $q(x_{l+1}|x_l)$'s mean is $a_{l+1}x_l$, we can rewrite the exponent of the pdf as

$$\exp\Big(-\frac{1}{2b_{l+1}^2} \left\| x_{l+1} - a_{l+1} x_l \right\|^2 \Big) = \exp\Big(-\frac{1}{2b_{l+1}^2/a_{l+1}^2} \Big\| \frac{1}{a_{l+1}} x_{l+1} - x_l \Big\|^2 \Big).$$

Considering the RHS as x_l's unnormalized density, which is a normal distribution. Consequentially, $q(x_l|x_{l+1}, x_0)$ in Equation 6.36 becomes a product of two Gaussian densities, and thus it is again a Gaussian; for that, we write

$$q(x_l|x_{l+1}, x_0) = N\big(x_l;\, \mu_{l|l+1}(x_{l+1}, x_0), \sigma_{l|l+1}^2 I\big), \tag{6.39}$$

where its mean and variance are given by

$$\begin{aligned} \mu_{l|l+1}(x_{l+1}, x_0) &= \frac{\sigma_l^2 a_{l+1}\, x_{l+1} + b_{l+1}^2 \alpha_l\, x_0}{\sigma_{l+1}^2} \quad \text{and} \quad \sigma_{l|l+1}^2 = \frac{b_{l+1}^2\, \sigma_l^2}{\sigma_{l+1}^2}. \\ &= \frac{1}{a_{l+1}} \left(x_{l+1} - \frac{b_{l+1}^2}{\sigma_{l+1}} \varepsilon'_{l+1} \right) \end{aligned}$$

For the second expression of the mean, we first rearrange Equation 6.37 to express x_0 in terms of all the others; that is

$$x_0 = \frac{1}{\alpha_{l+1}} \left(x_{l+1} - \sigma_{l+1} \varepsilon'_{l+1} \right). \tag{6.40}$$

Plugging it into the above mean we get the result.

Finally, having that $q(x_l|x_{l+1}, x_0)$ is a Gaussian, we can show that the l-th log probability ratio in the diffusion loss is written by the KL between two Gaussians. We write $\mathcal{L}_L(X_0; p_\theta, q)$ for the diffusion loss in the ELBO $\mathcal{E}^L$, and it is written by

$$\begin{aligned} \mathcal{L}_L(X_0; p_\theta, q) &= \mathbb{E}\left[\sum_{i=1}^{L-1} \log \frac{p_\theta(X_l|X_{l+1})}{q(X_l|X_{l+1}, X_0)} \middle| X_0 \right] \\ &= \mathbb{E}\left[\sum_{i=1}^{L-1} \underbrace{\mathbb{E}\left[\log \frac{p_\theta(X_l|X_{l+1})}{q(X_l|X_{l+1}, X_0)} \middle| X_{l+1}, X_0 \right]}_{-D_{\mathrm{KL}}\left(q(x_l|x_{l+1},x_0) \,\middle\|\, p_\theta(x_l|x_{l+1})\right)} \middle| X_0 \right], \end{aligned} \tag{6.41}$$

where the KL between $q(x_l|x_{l+1}, x_0)$ and $p_\theta(x_l|x_{l+1})$ is written by

$$\begin{aligned} &D_{\mathrm{KL}}\big(q(x_l|x_{l+1}, x_0) \,\big\|\, p_\theta(x_l|x_{l+1})\big) \\ &\quad = \frac{\left\| \mu_{l|l+1}(x_{l+1}, x_0) - \mu_\theta(x_{l+1}, l{+}1) \right\|^2}{2\sigma_\theta(l{+}1)^2} + \frac{d}{2}\left(\log \frac{\sigma_\theta(l{+}1)}{\sigma_{l|l+1}} + \frac{\sigma^2_{l|l+1}}{\sigma_\theta(l{+}1)^2} - 1 \right). \end{aligned} \tag{6.42}$$

The expectation in the RHS is taken with respect to $X_{l+1}|X_0$ and has no dependency of X_l. Intuitively, one can interpret that the variance reduction of the RHS is due to the elimination of the randomness of X_l, unlike the original expectation in Equation 6.42. In the statistics literature, such variation reduction scheme is called *variance reduction by conditioning*.

ε-parameterization The second key contribution of Ho et al. (2020) is that it proposes an efficient parameterization of $p_\theta(x_l|x_{l+1})$. In this tutorial, to enhance the coherence of the overall discussion, we describe the parameterization based on its recent variant shown in Kingma et al. (2021). Fundamentally, these efficient parameterizations use the conditional reverse pdf $q(x_l|x_{l+1}, x_0)$ in Equation 6.39 for models. The resulting models are written by

$$p_\theta(x_l|x_{l+1}) = q(x_l|x_{l+1},\, x_0{=}x_\theta(x_{l+1}, l{+}1)), \tag{6.43}$$

where we define $x_\theta(x_{l+1}, l{+}1)$ as

$$x_\theta(x_{l+1}, l{+}1) = \frac{1}{\alpha_{l+1}} \left(x_{l+1} - \sigma_{l+1} \varepsilon_\theta(x_{l+1}, l{+}1) \right). \tag{6.44}$$

Then, the mean of $q(x_l|x_{l+1}, x_0)$ becomes

$$\mu_\theta(x_{l+1}, l{+}1) = \frac{\sigma_l^2 a_{l+1}\, x_{l+1} + b^2_{l+1} \alpha_l\, x_\theta(x_{l+1}, l{+}1)}{\sigma^2_{l+1}} \tag{6.45}$$

$$= \frac{1}{a_{l+1}} \Big(x_{l+1} - \frac{b^2_{l+1}}{\sigma_{l+1}} \varepsilon_\theta(x_{l+1}, l{+}1) \Big), \tag{6.46}$$

and the variance becomes $\sigma^2_\theta(l{+}1) = \sigma^2_{l|l+1} = b^2_{l+1}\, \sigma_l^2 / \sigma^2_{l+1}$. Here, $\varepsilon_\theta(x_{l+1}, l{+}1)$ is called the *epsilon*-parameterization. If one may directly parameterize $x_\theta(x_{l+1}, l{+}1)$ instead of using ε_θ, such form is called *denoiser*-parameterization—also *denoising model* or simply *denoiser*. The denoiser directly predicts x_0 for the given x_{l+1}, while the epsilon model infers the noise part in Equation 6.40 and then predicts x_0.

Interestingly, the epsilon parameterization resembles Equation 6.33. Naturally, we can convert from ε_θ to s_θ—and

vice versa—by

$$-\frac{1}{\sigma_{l+1}}\varepsilon_\theta(x_{l+1}, l{+}1) = s_\theta(x_{l+1}, l{+}1). \tag{6.47}$$

In the literature, this is called the *score*-parameterization.

Using the above parameterizations, the MSE of the KL in Equation 6.42 becomes

$$\frac{1}{2\sigma_\theta(l{+}1)^2}\left\|\mu_{l|l+1}(x_{l+1}, x_0) - \mu_\theta(x_{l+1}, l{+}1)\right\|^2 = \frac{b_{l+1}^2\alpha_l^2}{2\sigma_l^2\,\sigma_{l+1}^2}\left\|x_0 - x_\theta(x_{l+1}, l{+}1)\right\|^2 \tag{6.48}$$

$$= \frac{b_{l+1}^2}{2a_{l+1}^2\,\sigma_l^2}\left\|\varepsilon'_{l+1} - \varepsilon_\theta(x_{l+1}, l{+}1)\right\|^2 \tag{6.49}$$

$$= \frac{b_{l+1}^2\sigma_{l+1}^2}{2a_{l+1}^2\,\sigma_l^2}\left\|\frac{1}{\sigma_{l+1}}\varepsilon'_{l+1} + s_\theta(x_{l+1}, l{+}1)\right\|^2. \tag{6.50}$$

Interestingly, the MSE in Equation 6.50 and the DSM's MSE in Equation 6.25 are the same except the scalar coefficient outside the MSE. Thus, the denoising diffusions are equal to the (continuous-time) score-based generative models in both the model perspective and their training objectives.

6.3.2.4 Connection to continuous-time models

In the previous section, we learned the reasoning behind the name "denoising diffusions"; in particular, we show that the models are a collection of denoising autoencoders. We also discussed efficient ELBO estimation, which will improve the training of the denoising diffusion models. Moreover, we learned that while the DDPMs' parameterizations are obtained from the denoising diffusion perspective, it is equal (up to scalar) to the score-parameterization of the score-based generative models. This relationship implies that we may be able to draw a more explicit connection between the denoising diffusion models, which are discrete-time models, and continuous-time score-based generative models. The following theorem provides the conclusion.

Theorem 6.4. [Consistency (Theorem 5 in Huang et al., 2021)] *Let functions f and g be of the forward SDE defined in Equation 6.20. Let μ_θ be defined as the mean function in Equation 6.30, and the mean is parameterized by σ_θ as in Equation 6.33. If g, g^{-2}, μ_θ, s_θ, and their derivatives up to the fourth order are all bounded and continuous, and that g is non-singular. Moreover, we write $\mathcal{E}^L$ for the discrete-time ELBO in Equation 6.32 and $\mathcal{E}$ for the continuous-time ELBO in Equation 6.2. Then, $\mathcal{E}^L \to \mathcal{E}$ as $L \to \infty$.*

For the proof of the consistency theorem, we refer to Huang et al. (2021). The theorem states that the denoising diffusion model will converge to the continuous-time score-based generative models as the discretization step goes zero. In conclusion, the score-based generative models and denoising diffusion models belong to the same family of models.

6.3.2.5 SNR-based expressions

As previously mentioned, Kingma et al. (2021) is one of the studies that introduced the variational lower bound for continuous-time diffusion-based generative models. Unlike Theorem 6.2, their approach extends discrete-time denoising diffusion models to a continuous-time framework. To do that, they introduced the signal-to-noise ratio (SNR) of the diffused data, a function that remains highly useful to this day in various applications, and proposed a method to express the variational lower bound through this SNR. They named this approach as *variational diffusion models (VDMs)*. This novel SNR-based characterization of diffusions enhances our understanding of this class of models. One of the byproducts is enabling a flexible discretization scheme of continuous-time models, resembling the DDPMs, which provides better performance in general compared to the EM methods for the same number of discretization steps. In the following, we will briefly introduce this study in relation to our discussion.

Before we dive into the details, let us first review the properties of the forward diffusion process in Equation 6.18; what is written by

$$\mathrm{d}X_t = f(X_t, t)\,\mathrm{d}t + g(t)\,\mathrm{d}W_t,$$

where W_t is a Wiener process. The solution X_t for an initial value $X_0 = x_0$ is written as a normal distribution, whose pdf is

$$q(x_t|x_0) = N(x_t; \alpha_t x_0,\, \sigma_t^2 I),$$

where α_t and σ_t^2 are defined in Equations 6.14 and 6.15; for them, we write

$$\alpha_t = \alpha(t) = e^{\int_0^t f(\tau)\,\mathrm{d}\tau}$$
$$\sigma_t^2 = \sigma^2(t) = \alpha_t^2 \int_0^t \alpha_\tau^{-2} g^2(\tau)\,\mathrm{d}\tau.$$

This definition is useful and holds for arbitrary functions f and g (under mild assumptions).

However, in the context of generative models, such a general solution may not be necessary. Instead, the subset of forward processes that are of interest in generative modeling can be fully characterized by the parameters α_t and σ_t alone–without defining SDEs (Kingma et al., 2021). For that, we let a stochastic process $\{X_t\}_{t\in[0,T]}$ satisfy that $X_t|X_0$ for $s \in (0, T]$ follow the pdf $q(x_t|x_0)$ defined by

$$q(x_t|x_0) = N(x_t; \alpha_t x_0,\, \sigma_t^2 I), \tag{6.51}$$

and $X_0 \sim q(x_0)$. In addition, we further assume the followings:

(i) The process is Markov such that for $0 \le s < t \le T$,

$$q(x_t|x_s, x_0) = q(x_t|x_s) = N(\alpha_{t|s}, \sigma_{t|s}^2 I), \tag{6.52}$$

where $\alpha_{t|s}$ and $\sigma_{t|s}$ are some positive number.

(ii) Both α_t and σ_t^2 are differentiable in t.

(iii) α_t^2 is monotonically non-increasing in t and $\lim_{t\to 0} \alpha_t = 1$.

(iv) σ_t^2 is monotonically increasing in t and $\lim_{t\to 0} \sigma_t^2 = 0$ (thus $\sigma_t > 0$ for $t > 0$.

For (i), we can show that $\alpha_{t|s} = \alpha_t/\alpha_s$, $\sigma_{t|s}^2 = \sigma_t^2 - \alpha_{t|s}^2 \sigma_s^2$. Interestingly, for such a process, there exists a unique stochastic process, solving a linear SDE whose joint distribution is equivalent to the one of $\{X_t\}_{t\in[0,T]}$. In particular, if $\alpha_t^2 + \sigma_t^2 = 1$ for all t, then it is a *variance preserving forward process*. If $\alpha_t^2 = 1$, then it corresponds to *variance exploding forward process*.[2]

In order to systematically describe the aforementioned definition, Kingma et al. (2021) introduces the *signal-to-noise ratio (SNR)* function of diffusion; that is

$$\mathrm{SNR}(t) \coloneqq \frac{\alpha_t^2}{\sigma_t^2}. \tag{6.53}$$

Instead of the assumptions (iii) and (iv) above, if $\mathrm{SNR}(t)$ is monotonically decreasing in t, $\lim_{t\to 0} \sigma_t^2 = 0$, and $\lim_{t\to 0} \alpha_t = 1$, then there exists a unique stochastic process, solving the following linear SDE (forward) (see Kingma

[2]For more details of the variance preserving and variance exploding processes, we refer to Song et al. (2020).

et al. (2021) for the proof). Moreover, f and g of the linear SDE as in Equation 6.20 are written as

$$f(t) = \frac{\mathrm{d}\log\alpha_t}{\mathrm{d}t} \quad \text{and} \quad g^2(t) = -2\sigma_t^2\,\frac{\mathrm{d}\log(\alpha_t/\sigma_t)}{\mathrm{d}t}. \tag{6.54}$$

Let us now define the discrete-time denoising diffusion process using the forward process characterized by α_t and σ_t, and we proceed to derive the lower bound of this model. We will begin by writing the one-step denoising diffusion in Equation 6.43, following the DDPMs. However, unlike DDPMs, where the coefficients for each step are predefined, the continuous-time model expressed in terms of α_t and σ_t allows for the convenient expression of the one-step denoising diffusion from time t to s for any arbitrary $0 \le s < t \le T$; that is,

$$x_s = \frac{\sigma_s^2\alpha_{t|s}\,x_t + \sigma_{t|s}^2\alpha_s\,x_0}{\sigma_t^2} + \frac{\sigma_{t|s}\,\sigma_s}{\sigma_t}\varepsilon,$$

where $\varepsilon \sim N(0, I)$. Then, we define the model to be $p_\theta(x_s|x_t) = q(x_s|x_t,\, x_0{=}x_\theta(x_t,t))$, and its mean and variance are written by

$$\begin{aligned}\mu_\theta(x_t,t,s) &= \frac{\sigma_s^2\alpha_{t|s}\,x_t + \sigma_{t|s}^2\alpha_s\,x_\theta(x_t,t)}{\sigma_t^2} \quad \text{and} \quad \sigma_{s|t,\,\theta}^2(t,s) = \frac{\sigma_{t|s}^2\,\sigma_s^2}{\sigma_t^2}, \\ &= \frac{1}{\alpha_t}\Big(x_t - \frac{\sigma_{t|s}^2}{\sigma_t}\varepsilon_\theta(x_t,t)\Big)\end{aligned} \tag{6.55}$$

where the epsilon parameterization is obtained via $x_\theta(x_t,t){=}\frac{1}{\alpha_t}\,(x_t - \sigma_t\varepsilon_\theta(x_t,t))$ from Equation 6.44. Having this expression in mind, we now discretize the continuous-time diffusion and then compute the discretized model' ELBO, which will be written as Equation 6.35. For that, consider a $L+1$-length sequence over the interval $[0,T]$ that satisfies

$$0 = \kappa(0) < \kappa(1) < \kappa(2) < \cdots < \kappa(i) < \cdots < \kappa(L) = T$$

for $i = 0, \ldots, L$. Here, a function κ maps from the index i to a time value on the interval; for example, $\kappa(i) \coloneqq i\,T/L$. Using this sequence as a discretization, we compute the ELBO for a subset $\{X_{\kappa(i)}\}_{i=0,\ldots,L}$ of $\{X_t\}_{t\in[0,T]}$. For an input x from the data, the diffusion loss $\mathcal{L}_L(x;p_\theta,q)$ of the discrete-time model now becomes

$$\mathcal{L}_L(x;p_\theta,q) = \frac{1}{2}\,\mathbb{E}\left[\sum_{i=2}^{L}\left(\frac{\alpha_{\kappa(i\text{-}1)}^2}{\sigma_{\kappa(i\text{-}1)}^2} - \frac{\alpha_{\kappa(i)}^2}{\sigma_{\kappa(i)}^2}\right)\|x - x_\theta(x_{\kappa(i)},\kappa(i))\|_2^2\right] \tag{6.56}$$

$$= \frac{1}{2}\,\mathbb{E}\left[\sum_{i=2}^{L}\left(\mathrm{SNR}(\kappa(i)) - \mathrm{SNR}(\kappa(i\text{-}1))\right)\|x - x_\theta(x_{\kappa(i)},\kappa(i))\|_2^2\right]. \tag{6.57}$$

Here, the denoiser-parameterizations are used to emphasize the brevity of the ELBO expression by using the SNR function (or α_t and σ_t).

Interestingly, thanks to the $\mathrm{SNR}(t)$-based expressions, the ELBO of the continuous-time denoising diffusions can also be easily computed and written in concise forms. For that, choose the uniform discretization over the interval $[0,T]$—$\kappa(i) \coloneqq i\,T/L$ for $i = 0, \ldots, L$, and then the summation in Equation 6.56 can be re-written as

$$\sum_{i=2}^{L}\big(\mathrm{SNR}(\kappa(i\text{-}1)) - \mathrm{SNR}(\kappa(i))\big)\|x - x_\theta(x_{\kappa(i)},\kappa(i))\|_2^2 = \sum_{i=2}^{L}\frac{\Delta\mathrm{SNR}(\kappa(i))}{\Delta t}\|x - x_\theta(x_{\kappa(i)},\kappa(i))\|_2^2,$$

where $\Delta\mathrm{SNR}(\kappa(i)) = \mathrm{SNR}(\kappa(i)) - \mathrm{SNR}(\kappa(i\text{-}1))$ and $\Delta t = \kappa(i) - \kappa(i\text{-}1) = T/L$. Due to the assumptions on the SNR function, the limit of the summation exists becomes an (improper) integral as $L \to \infty$. In addition, the prior loss

and the reconstruction loss in Equation 6.35 become zero for the limit. Thus, for $L \to \infty$, we get

$$\mathcal{E}^{\infty}(x) = \mathcal{L}_{\infty}(x) = \frac{1}{2}\,\mathbb{E}\left[\lim_{s\to 0+}\int_{s}^{T} \mathrm{SNR}'(t)\|x - x_\theta(x_t, t)\|_2^2\,\mathrm{d}t\right], \tag{6.58}$$

where $\mathrm{SNR}'(t)$ denotes the first derivative of the SNR function with respect to t.

Building upon the above $\mathrm{SNR}(t)$-based ELBO expression, Kingma et al. (2021) introduces more simplified expressions of the ELBO, and by using those expressions, they show that the ELBO is invariant with respect to the SNR functions. Moreover, Leveraging the invariance property, they propose a technique to improve the ELBO estimations. To do that, they first introduce a function $\gamma_t = \gamma(t) \coloneqq -\log \mathrm{SNR}(t)$. The SNR function is monotonic decreasing in t, and thus $\gamma(t)$ is monotonic increasing; moreover, it is invertible. For the inverse of the $\gamma(t)$, we write $\tau = \gamma^{-1}$. By using them, the ELBO in Equation 6.58 is rewritten as

$$\begin{aligned}
\mathcal{E}^{\infty}(x) &= \frac{1}{2}\,\mathbb{E}\left[\lim_{s\to 0+}\int_{s}^{T} \mathrm{SNR}'(t)\|x - x_\theta(x_t, t)\|_2^2\,\mathrm{d}t\right] \\
&\overset{\dagger}{=} \frac{1}{2}\,\mathbb{E}\left[\lim_{s\to 0+}\int_{s}^{T} \gamma'(t)\|\varepsilon_t - \varepsilon_\theta(x_t, t)\|_2^2\,\mathrm{d}t\right] &(6.59)\\
&\overset{\ddagger}{=} \frac{1}{2}\,\mathbb{E}\left[\lim_{s\to 0+}\int_{\gamma_s}^{\gamma_T} \|\varepsilon_\tau(\gamma) - \varepsilon_\theta(x_{\tau(\gamma)}, \tau(\gamma))\|_2^2\,\mathrm{d}\gamma\right], &(6.60)
\end{aligned}$$

where for ‡-identity, we first convert x_θ to the epsilon-parameterization and then rewrite the coefficient in terms of γ_t. For ‡-identity, we change the variable of the integration by $t = \tau(\gamma)$. Equation 6.60 implies that as long as the initial γ_s and terminal γ_T are unchanged, the ELBO doesn't change regardless of how the function $\mathrm{SNR}(t)$ is shaped, *i.e.* noise scheduling. However, Kingma et al. (2021) points out that the ELBO estimators and their qualities will still be affected by the $\mathrm{SNR}(t)$ function. Acknowledging this, they propose to minimize the variance of the ELBOs with respect to $\mathrm{SNR}(t)$, which will improve the training of the generative model ε_θ—also x_θ or s_θ, depending on the choice of parameterizations. See Kingma et al. (2021) for the details of the techniques.

Moreover, SNR-based expressions (or those based on α_t and σ_t) offer a more effective discretization approach for sampling continuous-time models. Remind that a continuous-time denoising diffusion is also a continuous-time score-based generative model, whose f and g are defined by Equation 6.54. For sampling the latter, we often use the EM-discretization in Equation 6.26. However, thanks to α_t and σ_t-based expressions of the continuous-time models, we can discretize them more flexibly and use the generation step in Equation 6.43 for the generation. For the same continuous-time score-based generative models and the number of function evaluations, the sampling with the DDPM-discretization has proven to be more accurate; in particular, such improvement is more noticeable for the small number of discretization steps.

6.3.3 Annealed Langevin dynamics perspective

So far, regarding the diffusion-based generative models, we have discussed the continuous-time score-based generative models defined by the forward-reverse diffusions and the denoising diffusion models formed by the perturbation-denoising relationship. However, the (discrete-time) score-based generative models are originally proposed to perform annealed Langevin dynamics (Song & Ermon, 2019). In this section, we briefly discuss Langevin dynamics, and move to how Song & Ermon (2019) motivate the annealed version of it. Then, we learn discrete-time score-based generative models, and conclude this section with drawing a link between the discrete-time and continuous-time models (see Figure 6.1c).

6.3.3.1 Langevin dynamics

As mentioned in Section 3.3.3, Langevin dynamics is a type of MCMC method. It has been widely used in tasks, such as estimating normalizing constants or performing inference, that require sampling from an unnormalized density. In the sampling problem, the goal is to sample from a probability density function π on $\mathbb{R}^d$. This differs from generative modeling, where we typically have access to samples rather than the target density.

One popular approach to achieve this is to use (over-damped) Langevin diffusion (Rossky et al., 1978; Parisi, 1981). That is an Itô diffusion X_t, which is the solution of the SDE defined by

$$\mathrm{d}X_t = \frac{1}{2}\nabla \log \pi(X_t)\,\mathrm{d}t + \mathrm{d}W_t \quad \text{for } t \geq 0, \tag{6.61}$$

where W_t is the d-dimensional Wiener process, and X_0 follows a prior distribution, such as the standard normal distribution. Here, we assume that π is continuous and differentiable, allowing access to $\nabla \log \pi(x)$. Under minor assumptions on π, X_t's distribution converges to the target distribution π as $t \to \infty$ (Roberts & Tweedie, 1996).

In practice, to solve the above SDE for sampling from π, the Euler-Maruyama discretization of Equation 6.61 is commonly used. This algorithm is referred to as *Langevin dynamics (LD)*, also known as the *unadjusted Langevin algorithm (ULA)* or *Langevin Monte Carlo (LMC)* in the MCMC literature.

$$X_{i+1} = X_i + \frac{1}{2}\gamma^2 \nabla \log \pi(X_i) + \gamma\,\varepsilon_{i+1}, \tag{6.62}$$

where $\varepsilon_{i+1} \overset{\text{i.i.d.}}{\sim} N(0, I)$ for all $i = 0, 1, \ldots, \infty$, and $\gamma > 0$ is the step size. Similar to continuous-time Langevin diffusions, under mild assumptions on g_i and π, LD also converges to the target distribution (Dalalyan, 2017; Durmus & Moulines, 2017).

Due to its simplicity and effectiveness, it has been widely adopted in the MCMC community. In the context of machine learning, Welling & Teh (2011) introduced an LD-based inference method known as stochastic gradient Langevin dynamics (SGLD), which has influenced the development of Bayesian inference techniques in machine learning.

6.3.3.2 Annealed Langevin dynamics

Contrary to the previous use case of the Langevin dynamics, our goal is generative modeling. Thus, we want to learn to sample the target density π by observing its data, specifically under the assumption that we don't have access to $\nabla \log \pi(x)$.

To do that, one may consider learning $\nabla \log \pi(x)$ by score matching and performing the LD with the learned score—a substitute for the true score. However, Song & Ermon (2019) points out that such a naive extension doesn't work well in high-dimensional data for several reasons. First of all, the inherent downside of the LD still remains, *i.e.* whose mixing is slow in high-dimensional data. In addition, accurately learning $\nabla \log \pi(x)$ from data is extremely difficult in high-dimensional cases.

To overcome these issues, Song & Ermon (2019) first proposes the annealed version of the Langevin dynamics, called the *annealed Langevin dynamics (ALD)*. That is, instead of sampling π directly by running Equation 6.62, to perform a cascade of sampling a sequence of noise-injected version of πs, whose noise levels converge to zero. To describe the algorithm more precisely, let $X^{(0)}$ be a random variable that follows the data distribution, and assume we have the following random variables

$$\begin{aligned} &X^{(0)}, X^{(1)}, \ldots, X^{(L)}, \text{ and} \\ &\quad X^{(l)} \coloneqq X^{(0)} + \beta_l \varepsilon'_l \quad \text{for } l = 1, \ldots, L, \end{aligned}$$

where $\varepsilon'_l \overset{\text{i.i.d.}}{\sim} N(0, I)$ and $\beta_1 > \beta_{l\text{-}1}$. Here, $X^{(l)}$'s densities are denoted by q_l. Then, we start sampling $X^{(L)}$ by running

the LD with $\nabla \log q_l(x)$; in particular, the dynamics is initialized with samples from a fixed prior distribution, which is often chosen to the perturbation noise's distribution. Next, we continue to sample $X^{(L\text{-}1)}$ by running the LD again but initialized with the samples of $X^{(L)}$. Finally, we repeat the previous step—sampling $X^{(l)}$ by running the LD but initialized with $X^{(l+1)}$'s samples—until we sample $X^{(0)}$.

As a generative model, Song & Ermon (2019) proposes to perform the ALD with the learned scores and name it the *score-based generative modeling*. In particular, to learn $\nabla \log q_l(x)$ for $l = 1, \ldots, L$, they proposes to use denoising score matching, since $X^{(l)}$s are constructed by adding Gaussian noises to the data $X^{(0)}$. In addition, the authors present a few essential techniques. For instance, instead of training multiple models to learn different noise level scores separately, they propose to use a single noise-conditional model to predict all levels of scores in an amortized fashion. The paper also introduces to selection of the step size γ_l of the LD to sample $X^{(l)}$ in proportion to its perturbation variance β_l^2.

6.3.3.3 Connection to continuous-time models

In the previous section, we learn that the original score-based generative models represent a subset of the solution of the forward diffusion in Equation 6.20; in particular, the forward diffusion of $f(s) = 0$; therefore, the former is a discretized version of the continuous-time one. In addition, the denoising score matching also serves an important role in the discretized version.

However, their samplings—the ALD and solving SDEs via discretizations—are, in essence, different. The latter focuses on transporting samples from one distribution to another. On the contrary, the ALD emphasizes performing the LD at each noise level but accelerates the LD's convergence by initializing it with the samples from the previous LD. To bridge the gap between the two, Song et al. (2020) propose to combine them, called the *predictor-corrector* samplers. That is, whenever we perform the update step for solving an SDE, *i.e.* prediction, we run the LD with the score at the corresponding time step, *i.e.* correction.

6.3.4 Wrapping up

In Section 6.3, we have discussed various perspectives of diffusion-based generative models and how they represent different sides of the same model class: the forward-reverse diffusions, perturbation-denoising diffusions, and annealed Langevin dynamics.

Interestingly such relationships only established in Euclidean spaces can also be extended to other non-trivial spaces. Suppose one aims to model a random variable on a non-trivial space, such as bounded space. In that case, the above identity implies that as long as we can define a generative model by using one approach, we can convert the model into other perspectives. We may cover these in the later chapters in this tutorial.

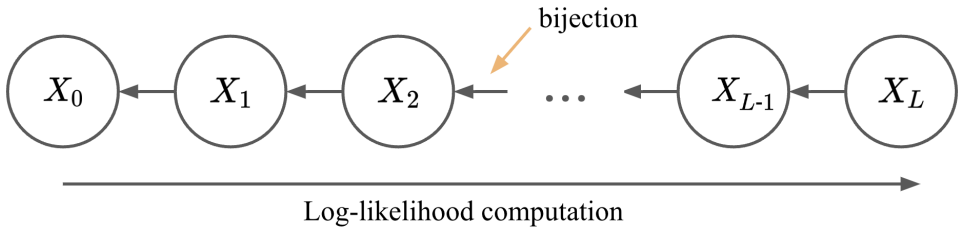
In the next sections, we cover the core intuitions (Section 6.4) and practical components (Section 6.5) that contribute to the effectiveness of diffusion-based models

6.4 Training of diffusion-models is divide-and-conquer

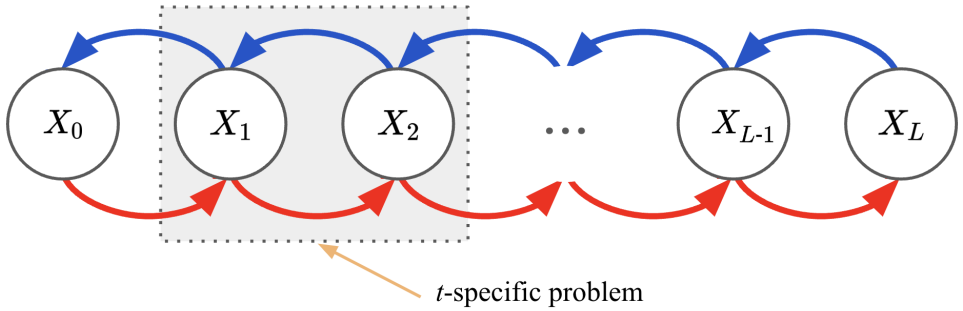
So far, we have discussed the fundamentals of diffusion-based generative models, specifically the theoretical sides. However, such theoretical components can't fully explain their successes in various applications. In particular, compared to other powerful generative models, such as GANs, the diffusion-based generative models have been demonstrating not only outstanding performance, but also training stability. This section provides a concise perspective on the factors that have contributed to the distinctive performance of diffusion-based generative models.

The primary distinction between diffusion-based generative models (DBGMs) and other generative models lies in the fact that the diffusion-based generative models operate in a manner analogous to **divide-and-conquer** algorithms. It is well known that the training of GANs or normalizing flows[3] updates the generative transformations in one go.

[3] In particular, the maximum likelihood training of normalizing flows



(a) Likelihood computation of normalizing flows.



(b) Training of denoising diffusion models at t.

Figure 6.2: Comparison between the MLE-based training of normalizing flows and the training of denoising diffusion models. The **black** arrows depict deterministic processes. The **blue** and **red** arrows illustrate probabilistic dependencies of graphical models, and they describe the generative and perturbation processes, respectively.

However, the DBGMs first decompose a complicated transform, *i.e.* solutions of the reverse diffusions, into (infinitely) many small problems, each of which is one step update, *e.g.* Equation 6.10, and then they learn the t-specific models for each t separately.

To help visualize such difference, see Figure 6.2. For training of a normalizing flow, one needs to compute the log-likelihood of a datapoint under the flow. The log-likelihood is computed by accumulating the log determinant Jacobians of all (invertible) transformations from the prior to the data. Updating the normalizing flow to maximize the log-likelihood implies updating all transformations of the graph at once, *i.e.* updates all graphs at once. On the contrary, the denoising diffusions learn to denoise for each time t separately, while the perturbation process is fixed.

6.5 Important techniques in practice

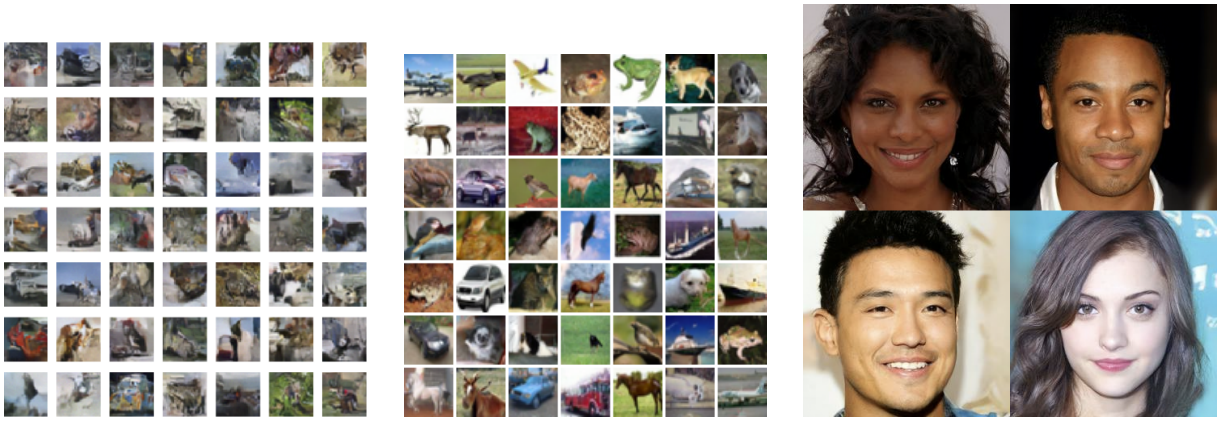
It is apparent that the success of diffusion-based models can be attributed to their 'divide-and-conquer'-like aspect. However, this success also relies on numerous critical deep learning techniques applied to the implementation of diffusion-based models. Without these techniques, the diffusion models would be prone to underfitting and would fail to achieve the performance levels we have come to expect. Among the many advancements developed in recent years, the two following are the crucial building blocks for diffusion-based models.

(i) Time-conditional UNet.

(ii) Noise schedule.

These two components are critical; without proper settings, diffusion models fail to function entirely. Thus, ensuring these elements are correctly implemented is fundamental to the diffusion models' operation.

Moreover, there are key components that, while not strictly essential, play a crucial role in achieving outstanding performance of the diffusion-based models. These components extend beyond the conventional techniques used to enhance the performance of general deep learning models, such as improved network architectures, and are specific considerations that must be addressed in the context of diffusion-based models. These components are outlined below.



(a) CIFAR-10 (DPM) **(b)** CIFAR-10 (DDPM) **(c)** CelebA-HQ 256×256 (DDPM)

Figure 6.3: Generated samples of the diffusion probabilistic model (DPM, Sohl-Dickstein et al., 2015) and the denoising diffusion probabilistic model (DDPM, Ho et al., 2020). The images are taken from the references.

(i) Parameterization and pre-conditioning.

(ii) Accelerated sampling.

6.5.1 Time-conditional UNet

Initially, diffusion-based generative models were first introduced as the diffusion probabilistic models (DPMs, Sohl-Dickstein et al., 2015) in the context of image modeling. However, when it was published, its performance was unremarkable compared to other generative model families and did not receive much attention. Later Ho et al. (2020) proposed denoising diffusion probabilistic models (DDPMs), which built upon the DPMs. The performance of DDPMs attracted significant attention to diffusion-based models.

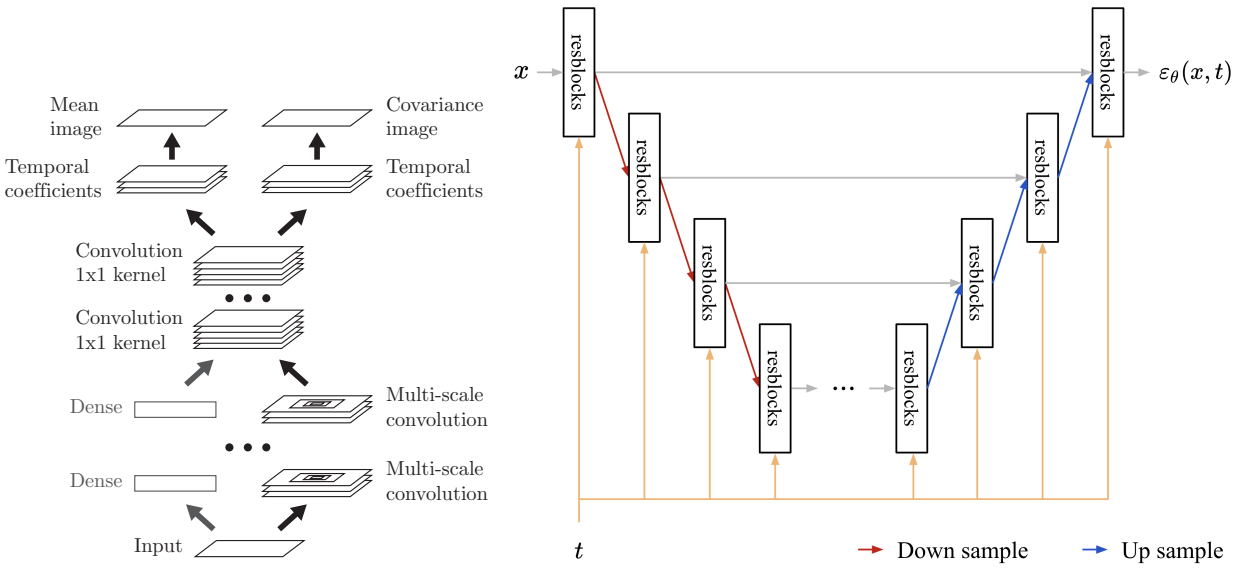
Although theoretical components of DDPMs and DPMs are essentially identical, they show a significant difference in practical performance, as illustrated in Figure 6.3. These improvements are primarily rooted in improved deep learning techniques. One of the most notable updates is the network architecture designs. In particular, the time-conditional UNets employed by DDPMs play an essential role. To elaborate more, not only was the network architecture changed from a convnet to a UNet, but the design of the time-dependent models was significantly improved, as seen in Figure 6.4.

For DPMs, Sohl-Dickstein et al. (2015) proposed a time-dependent convnet architecture to avoid defining separate convnets for each time step. Specifically, they add per-pixel bump functions after a convnet, which is shared across all times. The per-pixel bump functions shift up the output of the convnet depending on t to represent t-th mean of the denoising diffusions, as described in Figure 6.4a. For more details, see Sohl-Dickstein et al. (2015).

On the contrary, Ho et al. (2020) utilized the UNet network architecture in their approach. The UNet architecture is built using a deep hierarchical structure composed of residual modules; each module is essentially a small ResNet. This design, combined with extensive skip connections, has demonstrated exceptional performance in image-related tasks. Furthermore, Ho et al. (2020) introduced a novel feature: the feature-wise transformation technique (Dumoulin et al., 2018) to control the feature activations of every residual module based on the time step, as shown in Figure 6.4b. This allows the model's output to evolve in a highly complex yet smooth manner over time, unlike DPMs, where modulation is restricted to simpler feature levels.[4]

Overall, there is broad consensus that both the UNet architecture and the time-conditioning method have had a substantial impact. However, discerning which of these two contributes more significantly to performance improvement is challenging. Notably, the absence of either component leads to a marked decline in performance. Typically, attempts

[4] In a similar vein, Song & Ermon (2019) also proposed time-conditional UNets for the discrete-time score-based generative models, further supporting the effectiveness of the extensive feature-wise modulations.



(a) DPM **(b)** DDPM

Figure 6.4: Network architectures of the diffusion probabilistic model (DPM, Sohl-Dickstein et al., 2015) and the denoising diffusion probabilistic model (DDPM, Ho et al., 2020). **(a)** is taken from the reference. **(b)** Each 'resblocks' consists of a combination of small ResNets and attentions layers.

to use either a 'UNet without per-module time-conditioning' or a 'convnet/resnet with per-module time-conditioning' result in performance degradation to the point where the model is barely functional.

Indeed, based on this foundational combination, the refinement of network architectures in diffusion-based models continues to advance. Notable examples include research on enhanced time-conditioning (Nichol & Dhariwal, 2021) and the substitution of UNet with visual transformers (Peebles & Xie, 2023). These are valuable references that should be considered.

6.5.2 Noise schedule

As shown in Corollary 6.3, and similarly demonstrated in the analysis using denoising diffusion in Equation 6.60, the optimal solution of the diffusion-based generative models remains the same regardless of the choice of noise schedule or the weights in the score matching loss. However, both the noise scheduling and the weights can significantly affect the quality of the estimators of the training objectives, thereby influencing the speed and quality of training (Huang et al., 2021; Kingma et al., 2021). Furthermore, even for the same model, the granularity of discretization at different noise levels during the generation process can greatly impact the quality of the generated samples, even with the same number of function evaluations (Nichol & Dhariwal, 2021; Karras et al., 2022). Noise scheduling has become a crucial factor in training high-dimensional models end-to-end (Hoogeboom et al., 2023). These are key hyperparameters in the practical application of diffusion-based models. For those interested in exploring this topic further, the references above provide valuable insights.

6.5.3 Parameterization and pre-conditioning

Throughout this section, we have defined the diffusion-based generative models through various perspectives, with different model parameterizations being proposed depending on the context. However, since they are ultimately the same underlying model, conversions between different parameterizations are possible. The preferred parameterization tends to vary depending on applications, and different parameterizations appear to exhibit varying levels of variance and approximation errors. Generally, the epsilon parameterization is preferred. However, no definitive conclusion has

Table 6.1: Conversion table between popular parameterizations. The model described on top can be used to compute the values on the left.

	ε_θ	s_θ	x_θ
ε_θ	-	$-\sigma_t s_\theta(x_t, t)$	$-\frac{1}{\sigma_t}(x_\theta(x_t, t) - x_t)$
s_θ	$-\frac{1}{\sigma_t}\varepsilon_\theta(x_t, t)$	-	$\frac{1}{\sigma_t^2}(x_\theta(x_t, t) - x_t)$
x_θ	$\frac{1}{\alpha_t}(x_t - \sigma_t \varepsilon_\theta(x_t, t))$	$\frac{1}{\alpha_t}\big(x_t + \sigma_t^2 s_\theta(x_t, t)\big)$	-

been drawn on this matter, and the choice is typically determined through practical experience. Currently, the most commonly recognized parameterizations are as follows:

(i) Score parameterization s_θ (Song & Ermon, 2019; Song et al., 2020).

(ii) Epsilon parameterization ε_θ (Ho et al., 2020).

(iii) Denoising models x_θ (Kingma et al., 2021).

(iv) $\mathbf{v}$-prediction $\mathbf{v}_\theta$ (Salimans & Ho, 2022).

In addition to these, Karras et al. (2022) introduced a novel parameterization that applies what is known as *preconditioning* to the denoising parameterization. Preconditioning refers informally to initializing an optimization problem with a 'good' solution, which can significantly enhance learning performance (Karras et al., 2022; Williams et al., 2024).

Table 6.1 outlines the equations for converting between the most widely used parameterizations, excluding $\mathbf{v}$-parameterization. The model described above can be used to compute the values on the left side.

6.5.4 Accelerated sampling

The primary advantage of diffusion-based generative models lies in their ability to decompose a complex transformation problem into a series of smaller subproblems, ranging from a few dozen to nearly a thousand or even an infinite number. Notably, the finer the decomposition of the transformation problem, the more refined the resulting model becomes (see Theorem 6.4); hence the better performance. However, as a consequence, if carelessly implemented, the generation process typically requires as many repetitions of the update rule (such as Equation 6.34) as there are subproblems. For instance, when solving Equation 6.21 using score-based generative models with EM-discretization, smaller discretization steps generally yield superior results. Early studies often employed 1,000 discretization steps. Similarly, even when utilizing probability flow ODEs for generation, a smaller discretization step in the ODE solver—and consequently, a higher number of evaluations—tends to produce better results. As a result, numerous subsequent studies have investigated various methods to accelerate the generation process. In this tutorial, I will provide a brief overview of SDE solvers and several advanced ODE solvers in the context of generations of diffusion-based generative models. For approaches involving acceleration through learnable models, such as distillation, we refer to external references, including Salimans & Ho (2022) and related works.

Solving SDEs In the generation of continuous-time diffusion-based models, the numerical solution of the reverse SDE described in Equation 6.19 is typically used. This process involves first obtaining initial samples from the prior distribution and repeatedly applying the EM method outlined in Equation 6.26 for updates. However, beyond this straightforward discretization approach, as discussed in the concluding section of Section 6.3.2, one may alternatively employ a discretization scheme utilizing α_t and σ_t as described in Equation 6.55. As previously noted, VDM-style discretization tends to yield better results compared to EM discretization for the same number of discretization steps (Kingma et al., 2021). Furthermore, hybrid methods, such as the predictor-corrector samplers proposed by Song & Ermon (2019), can be employed as well (refer to Section 6.3.3 for further details).

Denoising diffusion implicit models *Denoising diffusion implicit models* (*DDIM*, Song et al., 2021a) represent one of the earliest works on the acceleration of diffusion model sampling. The primary motivation behind DDIM was to

redesign the forward and reverse diffusion processes (in continuous time) in a way that enables acceleration. In particular, it proposed a method that utilizes arbitrary discretization, akin to the one-step reverse process of VDM described in Equation 6.55. However, unlike VDM, DDIM introduces non-Markovian perturbations $q(x_t|x_s, x_0) \neq q(x_t|x_s)$ for $0 \leq s < t \leq T$, which deviates from the common assumption of forward diffusion.

More specifically, instead of defining the forward process and then inducing the joint distribution of its sample paths, they design the joint distribution of (arbitrary) discretizations of sample paths based on the followings

(i) $q(x_t|x_0) = N(x_t; \alpha_t x_0,\ (1-\alpha_t^2)I)$ for all ts, where α_t is monotonically decreasing in t.

(ii) For all $s < t$, the one-step reverse $q(x_s|x_t, x_0)$ is defined to satisfy

$$q(x_s|x_0) = \int q(x_s|x_t, x_0) q(x_t|x_0)\, \mathrm{d}x_t = N(x_s; \alpha_s x_0,\ (1-\alpha_s^2)I). \tag{6.63}$$

Thus, for some discretization $\{\kappa(i)\}_{i=0,\ldots,L}$ of the interval $[0, T]$, where $\kappa(i) \coloneqq i\,T/L$, we write the joint pdf of the forward by

$$q(x_{\kappa(1)}, \ldots, x_{\kappa(L)}|x_{\kappa(0)}) = q(x_{\kappa(L)}|x_{\kappa(0)}) \prod_{i=1}^{L} q(x_{\kappa(i-1)}|x_{\kappa(i)}, x_{\kappa(0)}). \tag{6.64}$$

In particular, the one-step reverse chosen by Song et al. (2021a) is

$$\begin{aligned} q(x_s|x_t, x_0) &= N(x_s|\mu_s(x_t, x_0), \sigma_{s|t}^2 I) \\ \mu_s(x_t, x_0) &= \alpha_s x_0 + \frac{\sqrt{1-\alpha_s^2-\sigma_{s|t}^2}}{\sqrt{1-\alpha_t^2}}\big(x_t - \alpha_t x_0\big), \end{aligned} \tag{6.65}$$

where $\sigma_{s|t}^2 \geq 0$. Note that we may not be able to get the analytical form of the forward transition $q(x_t|x_s)$ as well as the x_0-conditioned forward transition $q(x_t|x_s, x_0)$. Interestingly, with Equation 6.65 and the assumptions above, we can show the condition described in Equation 6.63.[5]

Finally, for parametric models in the generative process, DDIMs choose the plug-in one-step reverse with the denoising model; that is,

$$p_\theta(x_s|x_t) = q(x_s|x_t,\ x_0 = x_\theta(x_t, t)) \quad \text{and} \quad x_\theta(x_t, t) = \frac{1}{\alpha_t}(x_t - \sqrt{1-\alpha_t^2}\,\varepsilon_\theta(x_t, t)). \tag{6.66}$$

Here, one can directly parameterize $x_\theta(x_t, t)$ instead of using the epsilon parameterization. Then, the model is trained by minimizing the KL between the forward's joint distribution of Equation 6.64 and the reverse's defined by

$$p(x_{\kappa(0)}, x_{\kappa(1)}, \ldots, x_{\kappa(L)}) = \prod_{i=1}^{L} p(x_{\kappa(i-1)}|x_{\kappa(i)}). \tag{6.67}$$

Interestingly, DDIM encapsulates the DDPM as a special case. By setting $\sigma_{s|t}^2 = (1-\alpha_t^2/\alpha_s^2)(1-\alpha_s^2)/(1-\alpha_t^2)$, the above process becomes Markovian, and it becomes a variance preserving DDPM. Moreover, by $\sigma_{s|t} = 0$, we can define a fully deterministic DDIM, whose one-step update is written by

$$\begin{aligned} x_s &= \frac{\sqrt{1-\alpha_s^2}}{\sqrt{1-\alpha_t^2}}\, x_t + \left(\alpha_s - \alpha_t \frac{\sqrt{1-\alpha_s^2}}{\sqrt{1-\alpha_t^2}}\right) x_\theta(x_t, t) \\ &= \frac{\alpha_s}{\alpha_t} x_t - \alpha_s \left(\frac{\sqrt{1-\alpha_t^2}}{\alpha_t} - \frac{\sqrt{1-\alpha_s^2}}{\alpha_s}\right) \varepsilon_\theta(x_t, t). \end{aligned} \tag{6.68}$$

[5] For the proof, we can easily show that $\mathbb{E}[\,\mathbb{E}[X_s|X_t, X_0]\,|X_0 = x_0] = \alpha_s x_0$. The variance can be obtained similarly.

In addition, the fully deterministic DDIM sampler turns out to be a special numerical solution of the probability flow ODE, and the following discussion will cover this perspective.

Exponential integrators Probability flow ODEs, as described in Equation 6.27, enabled the use of ODE solvers rather than SDE solutions for sample generation in pre-trained diffusion-based models. The initial proposal by Song et al. (2020) employed general-purpose Runge-Kutta methods (Dormand & Prince, 1980) for solving these ODEs. However, even with such general-purpose ODE solvers, achieving state-of-the-art performance required a number of function evaluations (NFEs) in the range of a few hundred. Compared to SDE solvers, which require hundreds of discretization steps to achieve strong performance, the general-purpose ODE solver did not offer a significant reduction in NFEs. Consequently, various strategies have been pursued to overcome these limitations and reduce the required NFEs.

In the context of developing advanced ODE solvers, DPM-Solver (Lu et al., 2022) and Diffusion Exponential Integrator Sampler (DEIS, Zhang & Chen, 2023) focused on the observation that the probability flow ODEs in diffusion-based generative models are not only semi-linear but also characterized by a non-linear term that is exponentially weighted. Building on this insight, they proposed more efficient ODE solvers specifically tailored to solve probability flow ODEs, in contrast to general-purpose ODE solvers.[6]

To get a sample, we solve from T to 0 of the probability flow ODEs defined by Equation 6.27. In particular, the solution x_t at time s with the initial value x_t at time t, where $0 \leq s < t \leq T$, is written by

$$\begin{aligned} x_s &= e^{\int_t^s f(\tau)\,\mathrm{d}\tau} x_t + e^{\int_t^s f(\tau)\,\mathrm{d}\tau} \int_t^s \left[e^{-\int_t^\tau f(r)\,\mathrm{d}r} \frac{g^2(\tau)}{2\,\sigma_\tau} \varepsilon_\theta(x_\tau, \tau) \right] \mathrm{d}\tau \\ &= \frac{\alpha_s}{\alpha_t} x_t - \alpha_s \int_t^s \frac{\sigma_\tau}{\alpha_\tau} \left(\frac{\mathrm{d}\log(\alpha_\tau/\sigma_\tau)}{\mathrm{d}\tau} \right) \varepsilon_\theta(x_\tau, \tau)\,\mathrm{d}\tau \\ &= \frac{\alpha_s}{\alpha_t} x_t - \alpha_s \int_{\lambda_t}^{\lambda_s} e^{-\lambda} \varepsilon_\theta(x_{\tau(\lambda)}, \tau(\lambda))\,\mathrm{d}\lambda, \end{aligned} \tag{6.69}$$

where $\lambda_s \coloneqq \log(\alpha_s/\sigma_s)$, $\sigma_s/\alpha_s = \exp(-\lambda_s)$, and the inverse of $\lambda(\cdot)$ is denoted by τ, *i.e.* $\tau(\cdot) \coloneqq \lambda^{-1}(\cdot)$. Since the integration of the above equation is intractable, we use the polynomial approximation of ε_θ; in particular on λ-space. For the brevity of equations, we use $x_\lambda = x_{\tau(\lambda)}$ and $\varepsilon_\theta(x_\lambda, \lambda) = \varepsilon_\theta(x_{\tau(\lambda)}, \tau(\lambda))$. Then, we get

$$x_s = \frac{\alpha_s}{\alpha_t} x_t - \alpha_s \sum_{n=0}^{k-1} \varepsilon_\theta^{(n)}(x_{\lambda_t}, \lambda_t) \underbrace{\int_{\lambda_t}^{\lambda_s} e^{-\lambda} \frac{(\lambda - \lambda_t)^n}{n!}\,\mathrm{d}\lambda}_{(*)}, \tag{6.70}$$

where $\varepsilon_\theta^{(n)}$ is n-th derivative with respect to λ. The integration of the term $(*)$ can be computed analytically (by using integration-by-parts for $n > 0$). For instance, the first-order approximate solution from s to t is written by

$$\begin{aligned} x_s &= \frac{\alpha_s}{\alpha_t} x_t - \alpha_s \left(e^{-\lambda_t} - e^{-\lambda_s} \right) \varepsilon_\theta(x_{\lambda_t}, \lambda_t) \\ &= \frac{\alpha_s}{\alpha_t} x_t - \alpha_s \left(\frac{\sigma_t}{\alpha_t} - \frac{\sigma_s}{\alpha_s} \right) \varepsilon_\theta(x_t, t). \end{aligned} \tag{6.71}$$

Interestingly, this is equivalent to the deterministic DDIM, when $\sigma_t^2 = 1 - \alpha_t^2$. In addition, the DPM-Solvers also provide higher-order approximations for Equation 6.70, and such high-order solutions produce high-quality samples in only a few tens of iterations, such as 10 or 20, which is far fewer than the hundreds of iterations required by previous methods.

[6] Here, we follow Lu et al. (2022)'s derivation. While Zhang & Chen (2023)'s DEIS is driven from a slightly different perspective, it is essentially equivalent to the DPM-solver.

Higher-order numerical solvers In addition to methods like the DPM-Solver above, other higher-order numerical solution methods have also been developed. For instance, Karras et al. (2022) employed Heun's 2nd-order method. Moreover, beyond advancing ODE solvers, training additional neural networks to distill the results of probability flow ODE solutions of the pre-trained models (Salimans & Ho, 2022; Song et al., 2023). Additionally, Daras et al. (2024) proposes a martingale-based regularization to improve models' training, enhancing the generation qualities of the diffusion-based models.[7]

6.6 Other topics on diffusion-based models

Non-Euclidean spaces In this section, we have explored the fundamentals of the diffusion-based generative models. In particular, the discussions target the models on Euclidean spaces. In Chapter **??**, we will discuss function spaces and other essential building blocks to define diffusion-based generative models on function spaces. Besides Euclidean and function spaces, several studies have worked on other interesting domains, on which they propose diffusion-based generative models; for instances, Riemannian manifolds (De Bortoli et al., 2022; Huang et al., 2022; Lou et al., 2023) or bounded domains (Fishman et al., 2023; Lou & Ermon, 2023).

Diffusion-based samplers In addition to modeling random variables from data, the expressive power of diffusion-based models is also a huge merit to other important problems in the context of the generative models, such as variational inference (for latent variable models) or sampling from unnormalized densities. Unlike generative modeling, applying the diffusion-based models to those tasks introduces extra challenges. In particular, there are no data samples from the target densities, and thus, we cannot perform the score matchings. Nevertheless, several works have successfully addressed the inaccessibility of the target distribution and proposed diffusion-based sampling methods either by incorporating other reference path measures (Zhang & Chen, 2022; Vargas et al., 2023), using a REINFORCE-style estimator (Berner et al., 2022), using an AIS-based method (Doucet et al., 2022), or more (Richter & Berner, 2024; Akhound-Sadegh et al., 2024).

Optimal transport and bridge matching In the context of the optimal transport (OT, Peyré et al., 2019), particularly for the Schrödinger bridge problem, diffusion-based models have also been studied (De Bortoli et al., 2021; Shi et al., 2024). Briefly speaking, the Schrödinger bridge problem assume to have two distributions, which sampled independently, and find a joint distribution that is optimal with respect to a predefined cost function. Here, the joint distribution can be represented as one distribution and a conditional distribution of the other from one, and often, the problem looks for the optimal transport map between the two distributions. These studies further motivate studies on learning maps between coupled distributions, valuable to the style transfer task (Zhou et al., 2024; De Bortoli et al., 2023).

In addition, the diffusion-based Schrödinger bridge methods have triggered to explore using the Brownian bridge in generative modeling, resulting in the introduction of the bridge matching method (Liu et al., 2022), which learns diffusion-based mapping from one distribution to another from their samples. Interestingly, the bridge matching can be considered as a stochastic version of the flow matching (Lipman et al., 2023), which is a popular family of ODE-based generative models.[8]

References

Akhound-Sadegh, T., Rector-Brooks, J., Bose, A. J., Mittal, S., Lemos, P., Liu, C.-H., Sendera, M., Ravanbakhsh, S., Gidel, G., Bengio, Y., et al. Iterated denoising energy matching for sampling from Boltzmann densities. *arXiv preprint arXiv:2402.06121*, 2024.

[7] Lai et al. (2023) shows the connection between the consistency models introduced by Song et al. (2023) and the martingale-based regularization of Daras et al. (2024). See the reference for the details.

[8] The flow matching model is a neural ODE, and it performs to minimize what-so-called the flow matching objective. The training of this family also resembles to the divide-and-conquer algorithms like the training of the diffusion-based generative models.

Berner, J., Richter, L., and Ullrich, K. An optimal control perspective on diffusion-based generative modeling. *arXiv preprint arXiv:2211.01364*, 2022.

Blattmann, A., Dockhorn, T., Kulal, S., Mendelevitch, D., Kilian, M., Lorenz, D., Levi, Y., English, Z., Voleti, V., Letts, A., et al. Stable video diffusion: Scaling latent video diffusion models to large datasets. *arXiv preprint arXiv:2311.15127*, 2023.

Chen, N., Zhang, Y., Zen, H., Weiss, R. J., Norouzi, M., and Chan, W. WaveGrad: Estimating gradients for waveform generation. *ICLR*, 2021.

Dalalyan, A. S. Theoretical guarantees for approximate sampling from smooth and log-concave densities. *Journal of the Royal Statistical Society Series B: Statistical Methodology*, 79(3):651–676, 2017.

Daras, G., Dagan, Y., Dimakis, A., and Daskalakis, C. Consistent diffusion models: Mitigating sampling drift by learning to be consistent. *NeurIPS*, 2024.

De Bortoli, V., Thornton, J., Heng, J., and Doucet, A. Diffusion Schrödinger bridge with applications to score-based generative modeling. *NeurIPS*, 2021.

De Bortoli, V., Mathieu, E., Hutchinson, M., Thornton, J., Teh, Y. W., and Doucet, A. Riemannian score-based generative modelling. *NeurIPS*, 2022.

De Bortoli, V., Liu, G.-H., Chen, T., Theodorou, E. A., and Nie, W. Augmented bridge matching. *arXiv preprint arXiv:2311.06978*, 2023.

Dormand, J. R. and Prince, P. J. A family of embedded Runge-Kutta formulae. *Journal of computational and applied mathematics*, 6(1):19–26, 1980.

Doucet, A., Grathwohl, W., Matthews, A. G., and Strathmann, H. Score-based diffusion meets annealed importance sampling. *NeurIPS*, 2022.

Dumoulin, V., Perez, E., Schucher, N., Strub, F., Vries, H. d., Courville, A., and Bengio, Y. Feature-wise transformations. *Distill*, 3(7):e11, 2018.

Durmus, A. and Moulines, É. Nonasymptotic convergence analysis for the unadjusted Langevin algorithm. *The Annals of Applied Probability*, 27(3):1551–1587, 2017.

Fishman, N., Klarner, L., De Bortoli, V., Mathieu, E., and Hutchinson, M. J. Diffusion models for constrained domains. *Transactions on Machine Learning Research*, 2023.

Ho, J., Jain, A., and Abbeel, P. Denoising diffusion probabilistic models. *NeurIPS*, 2020.

Ho, J., Chan, W., Saharia, C., Whang, J., Gao, R., Gritsenko, A., Kingma, D. P., Poole, B., Norouzi, M., Fleet, D. J., et al. Imagen video: High definition video generation with diffusion models. *arXiv preprint arXiv:2210.02303*, 2022a.

Ho, J., Saharia, C., Chan, W., Fleet, D. J., Norouzi, M., and Salimans, T. Cascaded diffusion models for high fidelity image generation. *Journal of Machine Learning Research*, 23(47):1–33, 2022b.

Ho, J., Salimans, T., Gritsenko, A., Chan, W., Norouzi, M., and Fleet, D. J. Video diffusion models. *NeurIPS*, 35, 2022c.

Hoogeboom, E., Heek, J., and Salimans, T. Simple diffusion: End-to-end diffusion for high resolution images. *ICML*, 2023.

Huang, C.-W., Lim, J. H., and Courville, A. C. A variational perspective on diffusion-based generative models and score matching. *NeurIPS*, 2021.

Huang, C.-W., Aghajohari, M., Bose, J., Panangaden, P., and Courville, A. C. Riemannian diffusion models. *NeurIPS*, 2022.

Karras, T., Aittala, M., Aila, T., and Laine, S. Elucidating the design space of diffusion-based generative models. *NeurIPS*, 2022.

Kingma, D. P. and Gao, R. Understanding diffusion objectives as the ELBO with simple data augmentation. *NeurIPS*, 2024.

Kingma, D. P., Salimans, T., Poole, B., and Ho, J. Variational diffusion models. *NeurIPS*, 2021.

Kong, Z., Ping, W., Huang, J., Zhao, K., and Catanzaro, B. DiffWave: A versatile diffusion model for audio synthesis. *ICLR*, 2021.

Lai, C.-H., Takida, Y., Uesaka, T., Murata, N., Mitsufuji, Y., and Ermon, S. On the equivalence of consistency-type models: Consistency models, consistent diffusion models, and Fokker-Planck regularization. *arXiv preprint arXiv:2306.00367*, 2023.

Lipman, Y., Chen, R. T., Ben-Hamu, H., Nickel, M., and Le, M. Flow matching for generative modeling. *ICLR*, 2023.

Liu, X., Wu, L., Ye, M., et al. Let us build bridges: Understanding and extending diffusion generative models. *NeurIPS 2022 Workshop on Score-Based Methods*, 2022.

Lou, A. and Ermon, S. Reflected diffusion models. *ICML*, 2023.

Lou, A., Xu, M., Farris, A., and Ermon, S. Scaling Riemannian diffusion models. *NeurIPS*, 2023.

Lu, C., Zhou, Y., Bao, F., Chen, J., Li, C., and Zhu, J. DPM-solver: A fast ode solver for diffusion probabilistic model sampling in around 10 steps. *NeurIPS*, 2022.

Meng, C., He, Y., Song, Y., Song, J., Wu, J., Zhu, J.-Y., and Ermon, S. SDEdit: Guided image synthesis and editing with stochastic differential equations. *ICLR*, 2022.

Nichol, A. Q. and Dhariwal, P. Improved denoising diffusion probabilistic models. *ICML*, 2021.

Oksendal, B. *Stochastic differential equations: an introduction with applications*. Springer Science & Business Media, 2013.

Parisi, G. Correlation functions and computer simulations. *Nuclear Physics B*, 180(3):378–384, 1981.

Peebles, W. and Xie, S. Scalable diffusion models with transformers. *ICCV*, 2023.

Peyré, G., Cuturi, M., et al. Computational optimal transport: With applications to data science. *Foundations and Trends® in Machine Learning*, 11(5-6):355–607, 2019.

Richter, L. and Berner, J. Improved sampling via learned diffusions. *ICLR*, 2024.

Roberts, G. O. and Tweedie, R. L. Exponential convergence of Langevin distributions and their discrete approximations. *Bernoulli*, pp. 341–363, 1996.

Rombach, R., Blattmann, A., Lorenz, D., Esser, P., and Ommer, B. High-resolution image synthesis with latent diffusion models. *CVPR*, 2022.

Ronneberger, O., Fischer, P., and Brox, T. U-Net: Convolutional networks for biomedical image segmentation. In *Medical Image Computing and Computer-Assisted Intervention–MICCAI 2015: 18th International Conference, Munich, Germany, October 5-9, 2015, Proceedings, Part III 18*, pp. 234–241. Springer, 2015.

Rossky, P. J., Doll, J. D., and Friedman, H. L. Brownian dynamics as smart Monte Carlo simulation. *The Journal of Chemical Physics*, 69(10):4628–4633, 1978.

Saharia, C., Chan, W., Chang, H., Lee, C., Ho, J., Salimans, T., Fleet, D., and Norouzi, M. Palette: Image-to-image diffusion models. In *ACM SIGGRAPH 2022 conference proceedings*, pp. 1–10, 2022.

Salimans, T. and Ho, J. Progressive distillation for fast sampling of diffusion models. *arXiv preprint arXiv:2202.00512*, 2022.

Shi, Y., De Bortoli, V., Campbell, A., and Doucet, A. Diffusion Schrödinger bridge matching. *NeurIPS*, 2024.

Sohl-Dickstein, J., Weiss, E., Maheswaranathan, N., and Ganguli, S. Deep unsupervised learning using nonequilibrium thermodynamics. *ICML*, 2015.

Song, J., Meng, C., and Ermon, S. Denoising diffusion implicit models. *ICLR*, 2021a.

Song, Y. and Ermon, S. Generative modeling by estimating gradients of the data distribution. *NeurIPS*, 2019.

Song, Y., Sohl-Dickstein, J., Kingma, D. P., Kumar, A., Ermon, S., and Poole, B. Score-based generative modeling through stochastic differential equations. *ICLR*, 2020.

Song, Y., Durkan, C., Murray, I., and Ermon, S. Maximum likelihood training of score-based diffusion models. *NeurIPS*, 2021b.

Song, Y., Dhariwal, P., Chen, M., and Sutskever, I. Consistency models. *ICML*, 2023.

Vargas, F., Grathwohl, W. S., and Doucet, A. Denoising diffusion samplers. *ICLR*, 2023.

Welling, M. and Teh, Y. W. Bayesian learning via stochastic gradient Langevin dynamics. *ICML*, 2011.

Williams, C., Falck, F., Deligiannidis, G., Holmes, C. C., Doucet, A., and Syed, S. A unified framework for U-Net design and analysis. *NeurIPS*, 2024.

Zhang, Q. and Chen, Y. Path integral sampler: A stochastic control approach for sampling. *ICLR*, 2022.

Zhang, Q. and Chen, Y. Fast sampling of diffusion models with exponential integrator. *ICLR*, 2023.

Zhou, L., Lou, A., Khanna, S., and Ermon, S. Denoising diffusion bridge models. *ICLR*, 2024.

Part 2

Appendix

Appendix A

Probability theory

A.1 Introduction

As noted in Chapter 1, throughout this tutorial we adopt a probabilistic perspective on generative modeling, viewing it as the task of modeling random variables. Under this view, generative modeling is the activity of defining a new random variable—referred to as a model—and adjusting its distribution to match that of the data based solely on observed samples. To express this idea precisely, it is essential to understand the following:

(i) The definitions of a random variable and its distribution.
(ii) The relationship between samples and their distributions.
(iii) Various characterizations of distributions.
(iv) The precise meaning of a model 'learning' from data.

One of the primary goals of this chapter is to provide formal definitions for these key concepts—particularly for readers who may not be familiar with them. To do so, we present a concise overview of probability theory from the measure-theoretic perspective.

This exercise ultimately leads us to revisit familiar concepts in a more general and extensible form. For example, many readers may already be comfortable with generative models defined on Euclidean spaces, where notions such as probability densities and distances are well established. Yet extending these models beyond Euclidean domains is not always straightforward. The measure-theoretic framework offers a principled way to generalize such concepts to broader spaces without altering the core definition of probability.

A useful practice when reading this material is to reflect on how each formal definition connects back to the more intuitive notions you may already know. By making these connections explicit, you will find it easier to grow comfortable with the general definitions and to apply them flexibly in new contexts.

A.2 Probability spaces

A classic example used to introduce the idea of probability is rolling a die. Let us consider this simple yet illustrative case. We would like to describe the process of rolling a die—a process that involves inherent randomness—in a mathematically precise way. But how can we represent such random procedure as a mathematical object?

To describe such procedures formally, we turn to *probability theory*, in which they are referred to as *random experiments*. This section provides a brief overview of how to define the random experiments, which will serve as the foundation for the more advanced notions to come.

A.2.1 Sample space and event space

To describe probabilistic events as mathematical objects, we begin by introducing two fundamental sets: a sample space Ω and an even space $\mathcal{F}$.

The *sample space* Ω is the set of all possible outcomes for a given random experiment. The *event space* $\mathcal{F}$ is a collection of subsets of Ω, each of which corresponds to an event. That is, any event $A \in \mathcal{F}$ is a subset of the sample space, *i.e.* $A \subseteq \Omega$.

These two sets form the foundation for describing a random experiment, even before we knowing which outcomes are more probable or not. Formally, the pair $(\Omega, \mathcal{F})$ is called a *measurable space*. Once defined, we can associate a *measure* with this space—a function that assigns values to elements of $\mathcal{F}$.[1]

Let us revisit the earlier example of rolling a die, now expressed using this terminology:

- Random experiment : rolling a die
- Sample space: $\{1, 2, 3, 4, 5, 6\}$
- An outcome : 1, 2, and so on.
- An event : the die shows an odd number = $\{1, 3, 5\}$
- Event space: the collection of all such subsets (events)
- Measure : a function assigning values to events—*e.g.*, for a fair die, $\mu(\{1\}) = 1/6$ and so on (we will return to this shortly)

A.2.2 Event space is a σ-algebra

To be more precise in the discussions that follow, it is important to note that the event space $\mathcal{F}$ is a *σ-algebra* (also called a *σ-field*), and we will often use the latter terms. A σ-algebra of a set Ω is a nonempty collection of subsets of Ω that satisfies the following properties:

(i) If $A \in \mathcal{F}$, then $A^c \in \mathcal{F}$, where A^c is the complement of A.
(ii) If $A_i \in \mathcal{F}$ is a countable sequence of sets, then $\cup_i A_i \in \mathcal{F}$.

As a consequence, a σ-algebra is also closed under countable intersections, since

$$\cap_i A_i = (\cup_i A_i^c)^c \,.$$

For readers unfamiliar with some of the terminology above, the implications of these properties may not be immediately clear. Let us clarify two points:

- **Countability**: The term *countable* refers either to a finite set or to an infinite set that can be placed in one-to-one correspondence with the natural numbers. For example, the collection of all real numbers in a closed interval $[a, b]$ is uncountable.
- **Closure properties**: The fact that $\mathcal{F}$ must be closed under complement and countable union means, in effect, that every event in $\mathcal{F}$ can be built from other events using unions (intuitively, set plus) and complements (intuitively, set minus). Moreover, these closure rules apply to countably many operations—that is, to countable sequence of such constructions–not to arbitrary collections.

The restriction to countable number of operations can feel somewhat subtle. For instance, recall the sample space $\Omega = [a, b]$. The whole set $[a, b]$ cannot be expressed as a countable union of singletons $\{x\}$ (for $x \in [a, b]$), since that would be an uncountable union. Yet, long before introducing σ-algebras, other terms, and the probability theory based on them, we have already studied well about continuous random variables on $[a, b]$ without problems. How, then, can we represent the event set (including singletons) of the closed interval in the language of σ-algebras?

[1] We will elaborate on the meaning of measure and measurable shortly. The naming might feel abstract at first, but will become intuitive once we understand the role of the measure as a function.

One way is to define elementary events not as individual points $x \in \Omega$, but as open intervals:[2]

$$E = (a_i, b_i) \quad \text{for} \quad a_i < b_i \text{ and } a_i, b_i \in [a, b].$$

For these, we can recover singletons as countable intersections of open sets:

$$\{x\} = \bigcap_k \Big(\big(x - \varepsilon(k),\, x + \varepsilon(k)\big) \bigcap \Omega \Big),$$

where $\varepsilon(k)$ is a sequence of strictly positive numbers such that $\lim_{k\to\infty} \varepsilon(k) \downarrow 0$, such as $\varepsilon(k) = \frac{1}{k}$.

In this way, we can define a σ-algebra on the sample space $[a, b]$. What remains is to assign probabilities to every element (event) of this σ-algebra. Before doing so, however, it is worth noting that we do not need to define a new σ-algebra for every sample space individually. In fact, the spaces we typically care about almost always come equipped with a unique, canonical σ-algebra. These will be sufficient to describe probabilistic behavior in the settings we consider, as we will see in the next section.

A.2.3 Borel σ-algebra

Among many σ-algebras, there is one very important σ-algebra for our discussions. When we discuss the probabilities on a (topological) space, the set of all events is unique. Such σ-algebra is called *Borel σ-algebra* (or *Borel algebra*) of the set (or space). In other words, for a given sample space Ω, the σ-algebra consists of all events that we can normally think of is called the Borel algebra. Thus, as long as the sample space of interest is a (topological) space, there exists its Borel σ-algebra and is unique. This says that as long as the same space are used in our random experiments, their possible events are the identical. Consequently, once the event space is fixed by the Borel σ-algebra, different random experiments are distinguished only by the probabilities assigned to those events. In the next section, we introduce measures, *a.k.a.* set functions, which provide the formal mechanism for assigning probabilities within the σ-algebra.

Note that the Borel σ-algebra is a fundamental concept that is required to understand the probability theory in general, but we omit the proper definition of the Borel algebras in this tutorial for the accessibility of the material. From now on, we will denote the Borel algebra on a topological space Ω as $\mathcal{B}(\Omega)$, and we will often write $\mathcal{B}$ instead of $\mathcal{B}(\mathbb{R})$.

A.2.4 Measure and probability space

Having a measurable space $(\Omega, \mathcal{F})$, we would like to assign probabilities to events, and for that we define a measure. A *measure* is a **non-negative, countably-additive set function**; that is, a function $\mu : \mathcal{F} \to [0, \infty]$ satisfies the following conditions:

(i) (Non-negative) $\mu(A) \geq \mu(\varnothing) = 0$ for all $A \in \mathcal{F}$.

(ii) (Countably additive) for any countable sequence of disjoint sets in $\mathcal{F}$ with $\cup_i A_i \in \mathcal{F}$, denoted by $\{A_i \in \mathcal{F} \colon i \in \mathbb{N}\}$, we get

$$\mu\left(\cup_i A_i\right) = \sum_i \mu\left(A_i\right). \tag{A.1}$$

Here, we omit to describe the usual properties of measure μ on $(\Omega, \mathcal{F})$, such as *monotonicity*, *subadditivity*, and *continuity*, and those can be found in standard textbooks (Williams, 1991; Durrett, 2019).

Combining the measurable space $(\Omega, \mathcal{F})$ and a measure μ defined on it, we obtain $(\Omega, \mathcal{F}, \mu)$, which is called *measure space*.

To see where the terminology comes from: in mathematics, a *space* typically refers to a set of objects together with some additional structures that govern how they relate to one another. A measurable space is such a set endowed with a

[2] This construction is adapted from Example 1.11 in Williams (1991). Similar arguments work if one starts with closed or half-open intervals instead.

σ-algebra, which organizes the subsets of Ω in a way that allows to use to reason about them consistently. It is called measurable because a measure can be defined on this structure. Once we actually assign a measure to it, we obtain a measure space—hence the name.

Probability measure and probability space Let $(\Omega, \mathcal{F})$ be a measurable space and μ be a measure on them. If $\mu(\Omega) = 1$, we have $\mu : \mathcal{F} \to [0, 1]$, then we call μ a *probability measure*. In this case, we call $(\Omega, \mathcal{F}, \mu)$ as a *probability triple* or *probability space*.

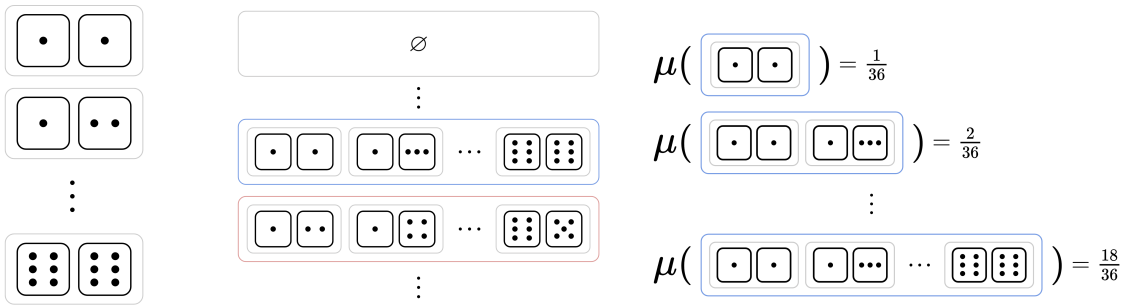
μ-null event and μ-almost everywhere Let $(\Omega, \mathcal{F}, \mu)$ be a probability space. An event A of $\mathcal{F}$ is called *μ-null* if $\mu(A) = 0$. A statement $\mathcal{S}$ about points $s \in S$ is said to hold *almost everywhere* (*a.e.*) if the negation of the statement $\mathcal{S}$ on S is false; that is,

$$\mu\left(\{s \in \Omega : \mathcal{S}(s) \text{ is false}\}\right) = 0. \tag{A.2}$$

For example, let φ and ψ be functions on Ω. We say $\varphi \geq \psi$ μ-almost everywhere (or $\varphi \geq \psi$ *μ-a.e.*) if

$$\mu\left(\{s \in \Omega : \varphi(s) < \psi(s)\}\right) = 0.$$

Examples of probability spaces Next, we will show two different examples of probability spaces: a discrete random variable and a continuous one. These examples will help us understand the difference between a sample space and an event space and familiarize us with measure functions.



(a) Sample space Ω **(b)** σ-algebra (or event space) $\mathcal{F}$ **(c)** Measure μ

Figure A.1: Example of probability space. (a) Sample space: all possible outcomes from two dice. (b) σ-algebra (or event space): all possible events we can construct with two dice. (c) Measure function: the probabilities assigned by a measure function μ to an each event.

Example A.1. [Two dice] Consider experiments to roll two dice. All possible outcomes establish its sample space Ω as shown in Figure A.1a. By considering all subsets from Ω, we can construct its event space (or σ-algebra) $\mathcal{F}$ as described in Figure A.1b. In particular, amongst many events, we illustrate events belonging to two specific experiments. One is when the sum of two dice is even (blue rectangle in Figure A.1b), and the other is when the sum is odd (red in Figure A.1b).

Figure A.1c illustrates how a measure function μ of one experiment (the sum of two dice is even) works. The measure specifies the probability of all events. In particular, under this measure, both dice are considered to be independent and fair, *i.e.* each face has the equal probability of $\frac{1}{6}$. As a result, the measure function μ assigns the probability of $\frac{1}{36}$ for each sample in each event.

Note that the probability of each event is fully governed by the measure function, not the sample space and the event space. For the sample measurable space $(\Omega, \mathcal{F})$, a different measure may assign probabilities to each sample

by different values. For example, imagine an experiment where one dice always faces the opposite side of the other, and such modification can be represented by a new measure.

Example A.2. [Real space $\mathbb{R}$] Consider a measurable space $(\mathbb{R}, \mathcal{B}(\mathbb{R}))$. The sample space of interest is $\mathbb{R}$, and each sample $x \in \mathbb{R}$ is a real value. Any event ω is an element of $\mathcal{B}(\mathbb{R})$, and some examples are

(i) A set consists of a single real value.
(ii) An open interval (a, b) for $a, b \in \mathbb{R}$ and $a < b$.
(iii) A closed interval (a, b) for $a, b \in \mathbb{R}$ and $a < b$.
(iv) and more.

For the measurable space $(\mathbb{R}, \mathcal{B}(\mathbb{R}))$, we can define a measure function. Note that a measure function is a set function, and it is different from a probability density function. It assigns a value to a set, not to each element in $\mathbb{R}$. This is very important for the latter discussions related to the probability theory in non-Euclidean spaces.

A.3 Random variables, distributions, and densities

At first glance, one might think that a probability space itself already captures a random experiment, and therefore there is no real distinction from a random variable. In many situations this identification works well enough, but for clarity it is important to distinguish between the two. A random variable should be regarded as a representation of a random experiment rather than the experiment itself.

Consider the following example. Suppose we have a single die, but we present its outcomes in two different ways. To person A, the outcome is displayed as a picture of a plant; to person B, it is displayed as a number. Both observers are witnessing the same underlying experiment, yet the random variables they perceive are different representations of it.

In this section, we will examine such distinctions more carefully, clarifying the role of random variables as representations of random experiments and setting the stage for their formal definition.

A.3.1 Measurable map and pushforward

Let $(\Omega, \mathcal{F})$ and (S, Σ) be measurable spaces. Suppose we have a function $X : \Omega \to S$, and we define a map $X^{-1} : \Sigma \to \mathcal{F}$ as in

$$X^{-1}(A) \coloneqq \{\omega \in \Omega : X(\omega) \in A\} \quad \text{for } A \subseteq S.$$

We say that X is a *measurable map* (or *measurable function*) from $(\Omega, \mathcal{F})$ to (S, Σ) if for all $A \in \Sigma$, $X^{-1}(A) \in \mathcal{F}$. Therefore, X and X^{-1} can be illustrated as

$$\begin{aligned} \Omega &\xrightarrow{X} S \\ \mathcal{F} &\xleftarrow{X^{-1}} \Sigma. \end{aligned}$$

In addition to two measurable spaces $(\Omega, \mathcal{F})$ and (S, Σ), assume we have a measure μ on $(\Omega, \mathcal{F})$. Suppose we have a measurable map X from $(\Omega, \mathcal{F})$ to (S, Σ). Then, X induces a measure ν on (S, Σ), and it is written as $\nu = \mu \circ X^{-1}$; that is for all $A \in \Sigma$,

$$\nu(A) = \mu \circ X^{-1}(A) = \mu \left\{\omega \in \Omega : X(\omega) \in A\right\}.$$

We refer to ν as the *pushforward measure* or *pushforward* of μ, and we write $\mu \circ f^{-1}$ or $f_{\#}\mu$. If μ is a probability measure then ν is also a probability measure.

A.3.2 Random variable

A *random variable* is a measurable map associated with a probability space. More specifically, let $(\Omega, \mathcal{F}, \mu)$ be a probability space and (S, Σ) be a measurable space. A measurable map $X : \Omega \to S$ from $(\Omega, \mathcal{F})$ to (S, Σ) is called a (S, Σ)-valued or S-valued random variable. For example, when $S = \mathbb{R}^d$, X is a random variable defined on d-dimensional Euclidean space (or $\mathbb{R}^d$-valued random variable). In here, X maps $\omega \in \Omega$ to arithmetical quantity, which is a d-dimensional real vector. Consider another example, such as rock-paper-scissors. Its sample space consists of abstract concepts; rock, paper, and scissors. On such element, applying some arithmetic may not be straightforward. However, once we assign some values on them, *i.e.* defining a random variable, we are now able to use some mathematical operations. While the above definition looks convoluted, this enables us to apply more general rules, such as even integration and differentiation on abstract mathematical objects (if they exist).

A.3.3 Law and distribution

To describe a random variable, we often use the *law* (or *distribution*) $\mathcal{L}_X$ of X; that is defined by

$$\mathcal{L}_X \coloneqq \mu \circ X^{-1}, \qquad \mathcal{L}_X : S \to [0, 1].$$

Note that $\mathcal{L}_X$ is a probability measure on (S, Σ). In effect, the random variable X serves as a bridge from the original probability space $(\Omega, \mathcal{F}, \mu)$ to the measurable space (S, Σ). Once we specify its distribution $\mathcal{L}_X$, we have fully characterized the probabilistic behavior of the random experiment.

Overall, the relationships between a probability space, a random variable, and its law can be visualized as in

$$\begin{array}{ccccc} & & \Omega & \xrightarrow{X} & S \\ & & & & \\ [0,1] & \xleftarrow{\mu} & \mathcal{F} & \xleftarrow{X^{-1}} & \Sigma \\ & & & & \\ [0,1] & & \xleftarrow{\qquad\qquad\mathcal{L}_X\qquad\qquad} & & \Sigma \end{array}$$

If X is real-valued random variable, we sometime use a *distribution function* F_X in order to describe a random variable X and it is defined as

$$F_X(x) \coloneqq \mathcal{L}_X(-\infty, x] = \mu(X \leq x) =: \mu\left\{\omega \in \Omega : X(\omega) \leq x\right\}.$$

A distribution function of a random variable X is sometimes called as a *cumulative distribution function (cdf)*.

Even though a random variable is a mapping, we often say a random variable X is "on" a probability space $(\Omega, \mathcal{F}, \mu)$ when X is from a probability space $(\Omega, \mathcal{F}, \mu)$. Moreover, we often refer to X as a Ω-valued random variable. More precisely, in this case, X is an identity map, and a measure μ becomes the law of X.

If X is distributed from a probability measure μ (or its law), we say X follows μ, and we write

$$X \sim \mu.$$

For more in-depth discussions in the context of the probability theory, it is important to distinguish a probability measure μ, a random variable X, its law $\mathcal{L}_X$, and the distribution function F_X. In this tutorial, however, we will use the terms—measure, law, distribution function—interchangeably, since a distribution function induces a law, and thus a probability measure, and vice versa.

A.3.4 Equal vs. equal in distribution

How can we say that two random experiments are "the same"? As mentioned earlier, it is possible to present a single experiment in different forms (*e.g.*, showing plant images versus showing numbers). This naturally raises a related

question: what does it really mean for two random variables to be equal?

Let us revisit the earlier example. Imagine a single die, but with its outcomes displayed differently: to person A as plant images, and to person B as numbers. Both observers are looking at the same experiment and seeing the same outcomes, yet the random variables are considered distinct, since they are different representations. Importantly, these two random variables always produce identical results, differing only in their form of expression.

Now consider a different situation. Suppose we hand out two separately manufactured dice, one to each person. The outcomes they observe will not always coincide, but the random behavior of the two dice is governed by the same probabilistic law.

This distinction is central:

- In the first case, if two random variables X and Y always yield the exact same outcome, we write $X = Y$.
- In the second case, if they induce the same law (*i.e.*, the same probability distribution), we say that X and Y are *equal in distribution*, denoted
$$X \overset{d}{=} Y.$$

Equality in distribution also implies that X and Y share the same probability measure or distribution function, even if their individual outcomes differ.

In fact, much of our discussion in generative modeling will be concerned not with strict equality $X = Y$, but with equality in distribution $X \overset{d}{=} Y$. When evaluating training methods, the central question is whether the proposed approach achieves this equality-in-distribution—that is, whether the learned model reproduces the same distribution as the data.

A.3.5 Composition

Let X be a random variable on a probability space $(\Omega, \mathcal{F}, \mu)$ and (S, Σ) be a another measurable space. Suppose we have a measurable map f from $(\Omega, \mathcal{F})$ to (S, Σ). Thus, a measurable map $Y := f \circ X$ is again a random variable, since f induces a pushforward probability measure $\nu = \mu \circ Y^{-1} = \mu \circ X^{-1} \circ f^{-1}$.

Similarly, we can keep create another random variable by transforming another random variable. However, one may need to be careful about designing new mapping, since not every mapping from Ω to S is measurable.

A.3.6 Absolutely continuity and Radon-Nikodym derivative

Suppose that μ and ν are probability measures on a measurable space $(\Omega, \mathcal{F})$. We say a measure μ is *absolutely continuous* with respect to ν, if

$$\nu(A) = 0 \quad \text{implies that} \quad \mu(A) = 0 \quad \text{for all } \ A \in \mathcal{F},$$

and we write $\mu \ll \nu$. If $\mu \ll \nu$ and $\nu \ll \mu$, then μ and ν are said to be *equivalent*.

If μ and ν are concentrated on disjoint sets, *i.e.* for all $A \in \mathcal{F}$

$$\mu(A) > 0 \ \Rightarrow \ \nu(A) = 0 \quad \text{and} \quad \nu(A) > 0 \ \Rightarrow \ \mu(A) = 0,$$

then they are called *singular*. In this case we write $\mu \perp \nu$.

The absolutely continuity of a measure relative to another leads us an important quantity called the Radon-Nikodym derivative.

Theorem A.1. [Radon-Nikodym theorem (Durrett, 2019)] *Suppose that μ and ν are probability measures on a measurable space $(\Omega, \mathcal{F})$. If $\nu \ll \mu$, there exists a measurable map f from $(\Omega, \mathcal{F})$ to $(\mathbb{R}_{\geq 0}, \mathcal{B}(\mathbb{R}_{\geq 0}))$ such that for*

all $A \in \mathcal{F}$

$$\int_A f \, \mathrm{d}\mu = \nu(A), \tag{A.3}$$

where the integral is the Lebesgue integral; moreover, the converse holds. Then, we write

$$\frac{\mathrm{d}\nu}{\mathrm{d}\mu} = f, \tag{A.4}$$

and it is called the Radon-Nikodym derivative (RN-derivative) of ν *relative to* μ *on* $(\Omega, \mathcal{F})$*. It is also called a density of* ν *relative to* μ*.*

RN-derivatives are very important tools in general. For example, it generalizes probability density functions, which we will introduce in the following section. Moreover, the RN-derivative allows to represent non-trivial distributions by using it combined with some known measures even on abstract spaces.

Many tools in machine learning are developed in the setting of Euclidean spaces, where probability densities are often sufficient to describe the behavior of $\mathbb{R}^d$-valued random variables. In this context, the Lebesgue measure plays a central role: it provides a natural way to treat volume uniformly across the real space, and probability densities can be understood as Radon–Nikodym derivatives with respect to this measure.

A.3.7 Lebesgue measure and probability density

Many tools in machine learning are developed in the setting of Euclidean spaces, where probability density functions are often sufficient to describe the behaviors of $\mathbb{R}^d$-valued random variables. Here, we aim at describing probability densities as Radon–Nikodym derivatives with respect to some reference measures, and this eventually enables facilitating the machine learning tools on other non-Euclidean spaces.

In order to do that we need to understand a special measure called Lebesgue measure.

Theorem A.2. [Theorem 1.1.4 Durrett (2019)] *A function* $F : \mathbb{R} \to \mathbb{R}$ *is called a Stieltjes (measure) function if* F *is non-decreasing and right continuous. For a Sieltjes function* F*, there is a unique measure* μ *on* $(\mathbb{R}, \mathcal{B})$ *with*

$$\mu((a, b]) = f(b) - f(a).$$

In particular, when $F(x) = x$*, the resulting measure is called Lebesgue measure, and we write* $\mathtt{Leb}(\mathbb{R})$*. In a similar way, we can construct Lebesgue measure on* $(\mathbb{R}^d, \mathcal{B}(\mathbb{R}^d))$*.*

The Lebesgue measure assigns provability uniformly on real space, and thus it is often called as n*-dimensional volume measure* or *the volume measure*. For this reason, its differential $d\mathtt{Leb}(\mathbb{R})$ is often written as dx.

Having the definition of the volume measure, we are now able to define the probability density functions. Let μ be a probability measure on $(\mathbb{R}, \mathcal{B})$. If μ is absolutely continuous w.r.t. the Lebesgue measure, then there exists a non-negative measurable map p such that

$$p = \frac{\mathrm{d}\mu}{\mathrm{d}\mathtt{Leb}(\mathbb{R})} = \frac{\mathrm{d}\mu}{\mathrm{d}x},$$

and it is called a *probability density function (pdf)* or a *density*. Thus, for a random variable X on $(\mathbb{R}, \mathcal{B}, \mu)$, we can

write

$$\mu(X \in A) = \int_A p(x)\,\mathrm{d}x.$$

When two random variables X and Y on $(\mathbb{R}^d, \mathcal{B}(\mathbb{R}^d))$ have the same density, they are necessarily equal in distribution. In other words, the density function alone is enough to characterize the random behavior of a variable. This perspective is especially useful in practice: it tells us that knowing the density fully determines the probabilistic law of the random variable.

For this reason, rather than always referring explicitly to the law induced by a density p, it is common to adopt a simpler notation. We say that an $\mathbb{R}^d$-valued random variable X **follows** the density p, and we write

$$X \sim p(x) \quad \text{or simply} \quad X \sim p.$$

This shorthand will be used frequently throughout the tutorial. It reflects the fact that, in many cases of interest, specifying a probability density is equivalent to specifying the entire distribution of the random variable.

A.4 Integration

This section is under preparation and will be updated in a future version.

A.5 Expectation

This section is under preparation and will be updated in a future version.

References

Durrett, R. *Probability: theory and examples*, volume 49. Cambridge university press, 2019.

Williams, D. *Probability with martingales*. Cambridge university press, 1991.