第 8 章 深度生成模型

8.1 引言

上一章最后一节我们讨论了语言生成模型，本章则主要探讨图像生成模型。本章所介绍的基本原理与模型并不仅限于图像生成，还可以扩展到视频、语音，甚至分子与蛋白质结构等数据的生成。尽管这些对象在物理意义与数据维度上存在差异，但从机器学习的角度来看，它们都可以统一表示为张量形式的数据。

图像生成模型的目标是学习真实图像的数据分布。具体而言，给定来自某一目标分布的观测数据 $\boldsymbol{x}$，生成模型的学习目标是显式地估计或隐式地逼近这些观测数据所服从的概率分布 $p(\boldsymbol{x})$。一旦模型完成学习，我们便可以通过从该分布 $p(\boldsymbol{x})$ 中采样（Sampling）来生成新的图像。在实际建模中，由于图像数据一般是高维的，我们通常并不直接从分布 $p(\boldsymbol{x})$ 中进行采样，而是构造一个可计算的生成过程，使得模型所生成的样本服从分布 $p(\boldsymbol{x})$，从而相当于从该分布中采样出了一个新样本。以生成猫的图像为例。现实世界中有各种各样的猫，因此也存在大量不同猫的图像。在机器学习中，我们通常将这些猫的图像视作来自某一概率分布 $p_{\text{cat}}(\boldsymbol{x})$。直观地说，该分布能够刻画不同猫图像出现的可能性：符合猫的常见视觉特征的图像通常具有较高的概率密度，而与猫的视觉特征明显不符的图像通常具有较低的概率密度。通过学习这一概率分布，模型便能够掌握猫图像的主要视觉特征及其变化规律。猫图像的生成任务则可以表述为：利用已有的猫图像学习一个近似于真实数据分布 $p_{\text{cat}}(\boldsymbol{x})$ 的概率模型，再从该模型中进行采样，从而生成符合猫的视觉特征的新图像。

在实际应用中，我们往往希望能够指定或控制生成图像的内容。例如，我们希望生成一只带有黑白斑点的小猫。这种以自然语言或其他形式给出的约束性要求，在机器学习中称为生成条件，通常记为 c。在这种情况下，原本学习无条件数据分布 $p(\boldsymbol{x})$ 的问题就转化为学习条件概率分布 $p(\boldsymbol{x}|c)$。条件生成赋予了模型更高的灵活性与实用价值。通过改变条件 c 的内容，我们可以对最终生成图像的语义和内容进行控制。例如，我们可以将条件修改为“生成一只有大大眼睛、圆圆脑袋的花狸猫”，从而引导模型生成符合该描述的图像。理想情况下，条件生成模型能够接受任意给定的条件 c，并生成与该条件一致且视觉上逼真的图像。

8.2 变分自编码器

在探讨变分自编码器之前，我们先介绍**自编码器**（Autoencoder）。自编码器的基本原理较为直观。如图8.1 所示，模型首先通过一个编码器 f，将原始数据空间中的输入图像 $\boldsymbol{x}$ 映射到表示空间中的隐表示 $\boldsymbol{z}$；随后，再通过一个解码器 g，从隐表示 $\boldsymbol{z}$ 重构出与原始图像尽可能相似的重构结果 $\boldsymbol{x}'$。当重构图像 $\boldsymbol{x}'$ 与原始输入 $\boldsymbol{x}$ 存在差异时，模型会利用基于二者差异定义的重构损失函数对编码器与解码器进行联合训练，从而使重构结果尽可能接近原始输入。值得注意的是，

自编码器名称中的“自”，并非“自动”的含义，而是强调“自我”，即模型以输入数据本身作为学习目标：输入图像既是模型的输入，也是模型希望重构的输出。由于训练过程中不需要人工标注，因此自编码器属于一种典型的**自监督学习**（Self-supervised Learning）方法。模型通过这种“自我重构”的过程学习数据的有效隐表示（即编码）。

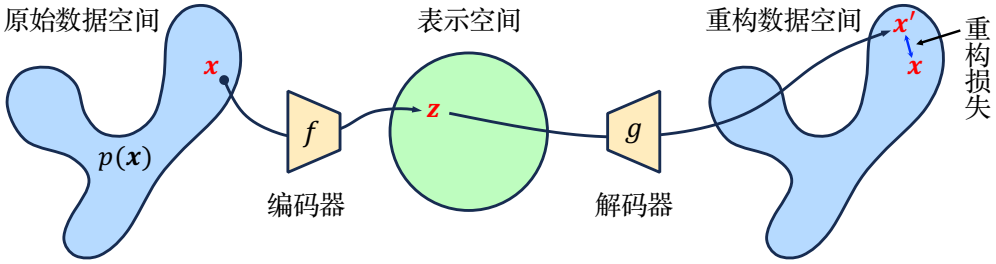


图 8.1: 自编码器示意图。输入图像 $\boldsymbol{x}$ 经编码器 f 映射为隐表示 $\boldsymbol{z}$，再通过解码器 g 从 $\boldsymbol{z}$ 重构出图像 $\boldsymbol{x}'$。模型通过最小化 $\boldsymbol{x}$ 与 $\boldsymbol{x}'$ 之间的重构损失对编码器和解码器进行端到端的联合训练。

良好的隐表示（Latent Representation）通常具有较低的维度（相对于高维的图像输入空间而言），能够以较为紧凑的形式表示图像中的重要信息。这是因为真实世界中的图像往往包含大量冗余信息，有时还夹杂着随机噪声，而学习隐表示的重要目标之一，就是在尽可能保留有用信息的同时，减少不必要的冗余与噪声，从而提取能够表征图像主要语义内容的关键特征。这些语义特征通常具有一定的不变性，即不容易因光照条件、拍摄角度、背景环境等因素的变化而发生本质改变。不过，这并不意味着隐表示会完全去除这些信息。在图像生成与重建等任务中，隐表示通常仍需要保留与光照、角度和背景等因素有关的信息，以便准确刻画和生成图像。这种紧凑的隐表示不仅有助于提取数据中的重要特征，还可以在采用压缩表示进行存储和传输时降低相应开销，并可能通过减少与任务无关的信息，在一定程度上提升模型的泛化能力。

自编码器中的编码器通常可以采用第7.3节所介绍的卷积神经网络将输入图像逐步压缩为指定维度的隐表示，而解码器则需要从低维隐表示逐步重构出高维图像，其本质上是一个升维的过程。一种常用的升维方法是采用**转置卷积**（Transposed Convolution）。转置卷积通过一个 $k \times k$ 的卷积核，将输入特征图中的每一个元素映射为一个 $k \times k$ 的局部区域，从而实现空间维度的扩张。具体而言，它将输入特征图中每一位置的元素与卷积核相乘，从而在输出特征图中的相应位置生成一个 $k \times k$ 的局部矩阵。相邻或相近输入元素所产生的局部区域在输出特征图中可能发生重叠，此时重叠位置的数值将进行累加。如图8.2所示，一个 3×3 的卷积核在步长（Stride）为 1 时，可以将 2×2 的输入特征图扩展为 4×4 的输出特征图；当步长设为 2 时，则可以生成 5×5 的输出特征图。这里的步长表示相邻输入元素在输出特征图中对应局部区域之间的位移。需要注意的是，转置卷积并不是普通卷积操作的逆运算，而是一种带有可学习参数的上采样（Upsampling）方法，因而被广泛应用于图像生成任务中。

基于转置卷积可以构建如图8.3所示的卷积生成网络 [59]。该网络首先利用一个全连接层（线性层）将 100 维的隐表示 $\boldsymbol{z}$ 映射为 $16,384$ 维的向量，并将其重塑（Reshape）为 1024 幅 4×4 的特征图；然后网络连续使用 4 个多通道转置卷积层，每个转置卷积层均采用 5×5 的卷积核和

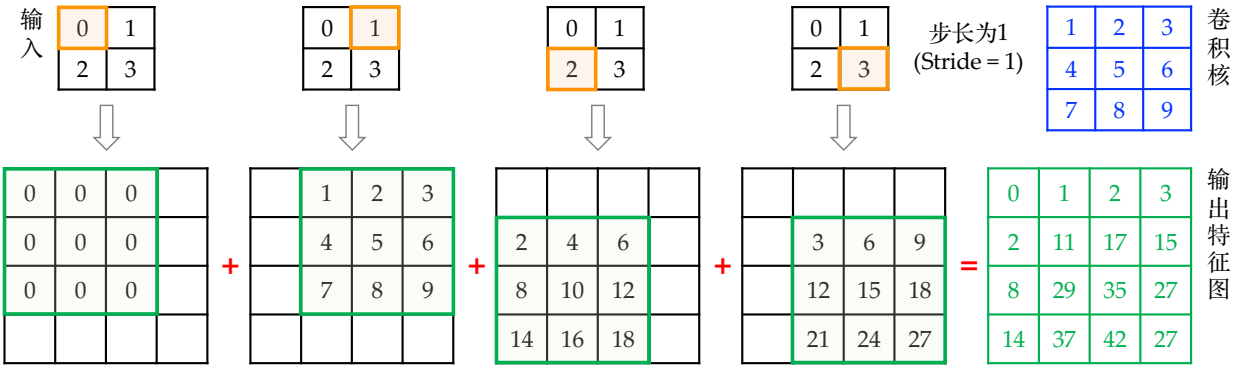


图 8.2: 转置卷积示意图。对于一个 2×2 的输入特征图，转置卷积首先将其中每个元素与一个 3×3 的卷积核相乘，从而生成对应的 3×3 局部区域；随后，根据步长将这些局部区域放置到输出特征图中的相应位置，并对重叠位置的数值进行累加，最终得到输出特征图。其中，步长表示相邻输入元素在输出特征图中对应局部区域之间的位移。

步长（Stride）为 2 的上采样设置，从而使特征图的空间分辨率逐步翻倍。在前 3 个转置卷积层之后，网络依次进行批量归一化和基于 ReLU 激活函数的非线性变换；而在最后一个转置卷积层之后，则直接通过 Tanh 激活函数进行非线性映射，最终生成一幅分辨率为 64×64（含 3 个通道）的图像。在实际应用中，输入的隐表示维度、输出图像的分辨率，以及各转置卷积层的通道数与空间尺寸等，都可以根据具体任务需求进行调整与优化。

分别为编码器和解码器选定适合的神经网络结构后，自编码器的参数训练过程便比较直观。给定一个包含 N 张图像的训练数据集 $\mathcal{D} = \{\boldsymbol{x}_i\}_{i=1}^{N}$，其中 $\boldsymbol{x}_i$ 表示第 i 张图像对应的向量表示。在训练过程中，样本 $\boldsymbol{x}_i$ 首先通过编码器 f 进行特征压缩与降维，生成相应的隐表示 $\boldsymbol{z}_i = f(\boldsymbol{x}_i)$；随后，解码器 g 以该隐表征作为输入进行重构，输出恢复后的图像 $\boldsymbol{x}_i' = g(\boldsymbol{z}_i)$。为了使重构结果尽可能接近原始输入，自编码器通常采用所有原始样本 $\boldsymbol{x}_i$ 与其重构结果 $\boldsymbol{x}_i'$ 之间的均方重构损失（即 ℓ_2 范数的平方损失）作为损失函数：

$$\mathcal{L} = \frac{1}{N} \sum_{i=1}^{N} \|\boldsymbol{x}_i' - \boldsymbol{x}_i\|^2 \tag{8.1}$$

训练通常以小批量的方式进行（如图8.3所示的卷积生成网络中也需要使用批量归一化），并以最小化损失函数为目标，采用随机梯度下降法对编码器与解码器的参数进行端到端的联合优化。

为何如公式8.1所示的均方重构损失适合作为自编码器的训练目标？从重构的角度来看，这一问题似乎比较直观，即其目标是在欧几里得表示空间中尽可能最小化原始样本与重构结果之间的距离。然而，若从更深层的统计学视角来看，生成模型的底层逻辑通常是先对数据分布 $p(\boldsymbol{x})$ 进行建模，然后通过最大化对数似然来优化模型参数。在这一理论框架下，我们首先需要为高维图像数据选择合适的概率分布模型。现实世界中的图像数据结构复杂、变化丰富，其真实数据分布 $p(\boldsymbol{x})$ 通常表现为复杂的多峰（Multi-modal）特性，难以使用简单的单峰分布直接进行建模。如图8.4(a) 所示，一种经典的统计学方法是采用在第2.5节介绍的混合高斯模型（Mixture of

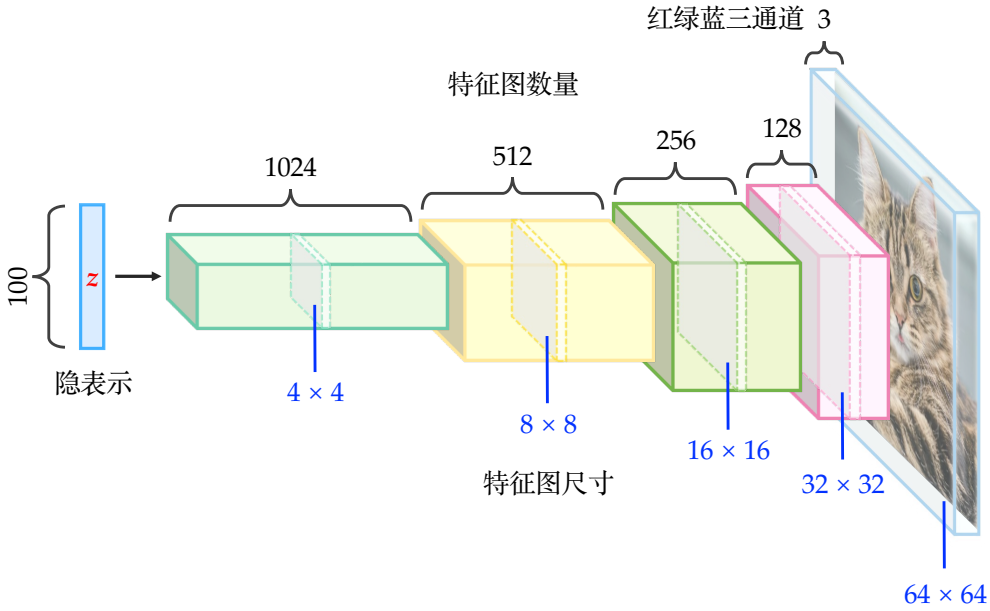


图 8.3: 基于转置卷积的卷积生成网络。网络先通过一个全连接层将 100 维隐表示转换成维度为 16,384 的高维向量，并将其重塑为 1024 幅 4×4 的特征图；随后，连续使用 4 个多通道转置卷积层，逐步提升特征图的空间分辨率，最终生成一幅分辨率为 64×64（含 3 个通道）的图像。

Gaussians）。混合高斯模型将复杂分布表示为 K 个离散高斯子成分的线性叠加：

$$p_\psi(\boldsymbol{x}) = \sum_{k=1}^{K} \pi_k \mathcal{N}(\boldsymbol{x}|\boldsymbol{\mu}_k, \boldsymbol{\Sigma}_k) \tag{8.2}$$

其中 $\boldsymbol{\psi} = \{\pi_k, \boldsymbol{\mu}_k, \boldsymbol{\Sigma}_k\}_{k=1}^{K}$。每个 $\mathcal{N}(\boldsymbol{x}|\boldsymbol{\mu}_k, \boldsymbol{\Sigma}_k)$ 称为混合高斯模型的子成分，π_k 为混合系数，需要满足以下条件：

$$\sum_{k=1}^{K} \pi_k = 1, \quad 0 \leq \pi_k \leq 1 \tag{8.3}$$

理论上，通过组合足够多的子成分并调整它们的均值、协方差及混合系数，离散混合高斯模型几乎可以以任意精度逼近任何复杂的分布。然而，对于现实世界中的图像数据，我们通常无法事先确定需要多少个高斯子成分，即难以确定 K 应取何值才能充分刻画真实数据分布。因此，自编码器将离散的有限混合扩展为连续混合高斯模型（Continuous Mixture of Gaussians），从而在理论上可以容纳无限多个高斯子成分（即让 $K \to \infty$）。随之而来的问题在于：我们无法直接学习无限多个高斯子成分对应的均值与协方差参数。解决方法是将这些高斯子成分的参数表示为关于连续隐变量 $\boldsymbol{z}$ 的非线性函数。在自编码器框架中，这一非线性映射是由解码器 g_θ 实现的（$\boldsymbol{\theta}$ 表示其参数）。如图8.4(b) 所示，解码器 g_θ 以连续隐变量 $\boldsymbol{z}$ 为输入，输出该隐变量所对应的高斯子成分的均值 $g_\theta^\mu(\boldsymbol{z})$ 与协方差 $g_\theta^\Sigma(\boldsymbol{z})$。由此，数据的生成过程可以建模为：首先给定一个隐变量 $\boldsymbol{z}$，然后从以解码器输出为参数的条件高斯分布中采样生成数据 $\boldsymbol{x}$，即：

$$p_\theta(\boldsymbol{x}|\boldsymbol{z}) = \mathcal{N}(\boldsymbol{x}|g_\theta^\mu(\boldsymbol{z}), g_\theta^\Sigma(\boldsymbol{z})) \tag{8.4}$$

在选择采用连续混合高斯模型来建模数据分布之后，我们可以通过最大化分布 $p(\boldsymbol{x})$ 的对数

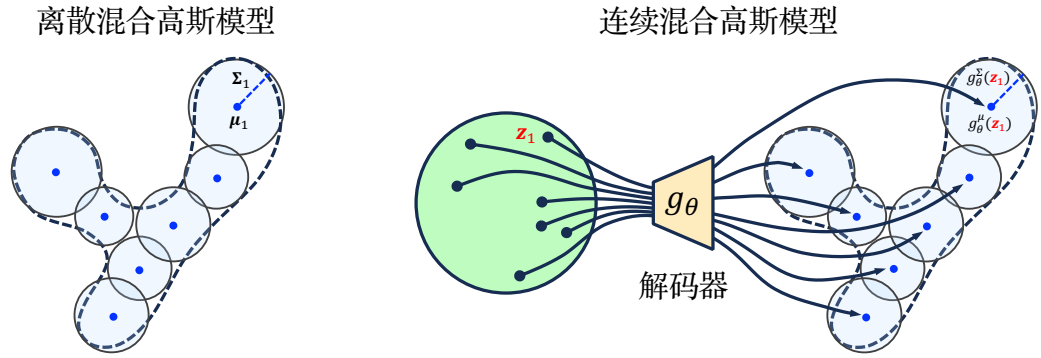


图 8.4: 离散混合高斯模型与连续混合高斯模型。(a) 离散混合高斯模型：通过有限个（共 K 个）高斯子成分的加权叠加来逼近复杂数据分布。(b) 连续混合高斯模型：自编码器将离散混合扩展到连续隐空间，从而在理论上能够容纳无限多个高斯子成分。核心思想是将高斯子成分的参数（均值与协方差）表示为关于连续隐变量 $\boldsymbol{z}$ 的非线性函数（即解码器），从而将数据的生成过程建模为对条件高斯分布 $p_\theta(\boldsymbol{x}|\boldsymbol{z}) = \mathcal{N}(\boldsymbol{x}|g_\theta^\mu(\boldsymbol{z}), g_\theta^\Sigma(\boldsymbol{z}))$ 的采样过程。此例子改编自文献 [76]。

似然来推导出模型的优化目标。在上述分析中，我们已经获得了数据的条件概率分布 $p_\theta(\boldsymbol{x}|\boldsymbol{z})$。为了得到观测数据的边缘分布 $p_\theta(\boldsymbol{x})$，我们需要将该条件概率对隐变量的先验分布 $p(\boldsymbol{z})$ 求期望，这等价于在整个连续隐空间上对隐变量进行积分：

$$p_\theta(\boldsymbol{x}) = \int_{\boldsymbol{z}} p(\boldsymbol{z}) p_\theta(\boldsymbol{x}|\boldsymbol{z})\, d\boldsymbol{z} = \int_{\boldsymbol{z}} p(\boldsymbol{z}) \mathcal{N}(\boldsymbol{x}|g_\theta^\mu(\boldsymbol{z}), g_\theta^\Sigma(\boldsymbol{z}))\, d\boldsymbol{z} \tag{8.5}$$

为了简化后续的推导，我们需要对条件概率分布引入一些合理的简化假设。在图像生成任务中，我们通常假设在给定隐变量 $\boldsymbol{z}$ 的条件下，生成的图像数据 $\boldsymbol{x}$ 服从一个各向同性的多维高斯分布，其协方差矩阵为 $\sigma^2\mathbf{I}$，其中 σ^2 是一个常数超参数，而 $\mathbf{I}$ 为单位矩阵。这一假设意味着，在给定隐表示 $\boldsymbol{z}$ 的条件下，生成图像 $\boldsymbol{x}$ 的各个像素点（或特征的各个维度）之间的噪声被认为是相互独立且同分布的，并且都服从均值为 0、方差为 σ^2 的高斯分布。此外，该条件高斯分布的均值由解码器直接预测，即 $g_\theta^\mu(\boldsymbol{z}) = \boldsymbol{x}'$。结合上述假设与分析，数据的边缘分布公式8.5可改写为：

$$p_\theta(\boldsymbol{x}) = \int_{\boldsymbol{z}} p(\boldsymbol{z}) p_\theta(\boldsymbol{x}|\boldsymbol{z})\, d\boldsymbol{z} = \int_{\boldsymbol{z}} p(\boldsymbol{z}) \mathcal{N}(\boldsymbol{x}|\boldsymbol{x}', \sigma^2\mathbf{I})\, d\boldsymbol{z} \tag{8.6}$$

由于传统自编码器并未对隐表示 $\boldsymbol{z}$ 的分布施加任何约束，其编码器 f 针对某一特定样本给出的隐表示 $\boldsymbol{z}_0$ 只能被视为隐空间中的一个确定性点。从统计学的角度来看，这等价于假设隐变量的分布服从狄拉克 δ 函数（Dirac Delta Function），即 $p(\boldsymbol{z}) = \delta(\boldsymbol{z} - \boldsymbol{z}_0)$。狄拉克 δ 函数是一种广义的概率密度函数，其全部概率质量都集中在 $\boldsymbol{z}_0$ 这一单点上。具体而言，它在除 $\boldsymbol{z} = \boldsymbol{z}_0$ 以外所有点上的值都等于零，且在整个定义域上的积分为 1。因此，在计算条件概率 $p_\theta(\boldsymbol{x}|\boldsymbol{z})$ 关于隐变量分布 $p(\boldsymbol{z})$ 的期望时，由于 δ 函数会将积分退化为在 $\boldsymbol{z}_0$ 一点处的取值，最终积分结果等价于

该点处的条件概率本身：

$$p_\theta(\boldsymbol{x}) = \int_{\boldsymbol{z}} p(\boldsymbol{z}) p_\theta(\boldsymbol{x}|\boldsymbol{z})\, d\boldsymbol{z} = \int_{\boldsymbol{z}} \delta(\boldsymbol{z}-\boldsymbol{z}_0) p_\theta(\boldsymbol{x}|\boldsymbol{z})\, d\boldsymbol{z} = p_\theta(\boldsymbol{x}|\boldsymbol{z}_0) \tag{8.7}$$

综上所述，数据的对数似然可以推导如下（假设输入数据维度为 D）：

$$\begin{aligned}
\log p_\theta(\boldsymbol{x}) &= \log p_\theta(\boldsymbol{x}|\boldsymbol{z}_0) \\
&= \log \mathcal{N}(\boldsymbol{x}|\boldsymbol{x}', \sigma^2\mathbf{I}) \quad (g_\theta^\mu(\boldsymbol{z}_0) = \boldsymbol{x}') \\
&= \log \frac{1}{(2\pi)^{D/2}|\sigma^2\mathbf{I}|^{1/2}} \exp\left(-\frac{1}{2}(\boldsymbol{x}-\boldsymbol{x}')^\top(\sigma^2\mathbf{I})^{-1}(\boldsymbol{x}-\boldsymbol{x}')\right) \quad (\text{多维高斯分布定义}) \\
&= \log \frac{1}{(2\pi\sigma^2)^{D/2}} \exp\left(-\frac{1}{2}(\boldsymbol{x}-\boldsymbol{x}')^\top(\sigma^2\mathbf{I})^{-1}(\boldsymbol{x}-\boldsymbol{x}')\right) \quad (|\sigma^2\mathbf{I}| = \sigma^{2D}) \\
&= \log \frac{1}{(2\pi\sigma^2)^{D/2}} \exp\left(-\frac{\|\boldsymbol{x}-\boldsymbol{x}'\|^2}{2\sigma^2}\right) \quad ((\sigma^2\mathbf{I})^{-1} = \frac{1}{\sigma^2}\mathbf{I}) \\
&= \underbrace{-\frac{D}{2}\log(2\pi\sigma^2)}_{\text{常数项}} - \underbrace{\frac{1}{2\sigma^2}}_{\text{常数项}} \|\boldsymbol{x}-\boldsymbol{x}'\|^2 \quad (\text{假设噪声方差 } \sigma^2 \text{ 为常数超参数})
\end{aligned}$$

从上述推导的最终结果可以看出：在假设数据的条件概率分布服从多维高斯分布且噪声方差 σ^2 为常数超参数的前提下，最大化数据的对数似然等价于最小化原始样本 $\boldsymbol{x}$ 与重构样本 $\boldsymbol{x}'$ 之间的欧几里得均方重构误差（即公式8.1）。

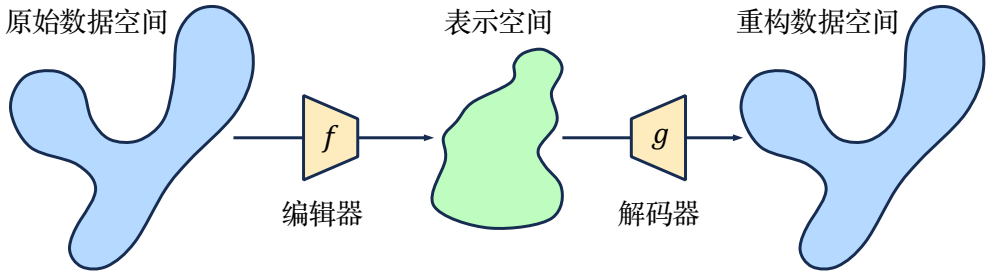


图 8.5: 自编码器所学习到的表示空间可能并不适用于采样生成。在训练过程中，仅依赖均方重构损失进行优化并不能保证习得的隐表示空间具有良好的连续性与紧致性，其复杂度可能与原数据空间相当，因而难以通过直接对其采样来生成新的图像。

在本节开篇中，我们提到“生成即采样”的思想，即生成本质上是从某种学习到的概率分布中随机采样，再根据采样结果生成新图像的过程。在自编码器的训练过程中，仅依赖均方重构损失只能保证模型能够对训练样本进行准确重构，却没有对隐表示空间的整体结构施加有效的约束。这可能导致学习得到的隐空间呈现出高度的不连续性，并存在大量未被训练样本覆盖的“空白区域”。如图8.5所示，仅通过均方重构损失学习到的隐表示空间，虽然在一定程度上实现了对原始数据的降维，但其结构复杂性仍可能与原始数据空间相当。这意味着：尽管模型能够较好地重构已有样本，但从隐空间中随机采样得到的点很可能落入空白区域，从而无法通过解码器生成合理的新图像。

基于上述分析，为了在生成过程中能够在隐表示空间中进行有效采样，我们需要对隐空间

的结构施加约束，即假设隐表示服从某种结构简单且易于采样的已知分布。通常，我们会规定隐表示的先验分布 $p(\boldsymbol{z})$ 服从均值为 $\mathbf{0}$、协方差矩阵为单位矩阵 $\mathbf{I}$ 的多元标准高斯分布，即：

$$p(\boldsymbol{z}) = \mathcal{N}(\boldsymbol{z}|\mathbf{0}, \mathbf{I}) \tag{8.8}$$

在这一假设下，隐变量 $\boldsymbol{z}$ 不再是隐空间中确定的一个单点，而是一个随机变量。因此，它将与观测数据 $\boldsymbol{x}$ 共同构成联合分布 $p(\boldsymbol{x}, \boldsymbol{z}) = p(\boldsymbol{z})p_\theta(\boldsymbol{x}|\boldsymbol{z})$（其中 $\boldsymbol{\theta}$ 是解码器的参数）。为了获得观测数据的边缘分布 $p(\boldsymbol{x})$，我们需要在整个隐空间上对联合分布进行积分：

$$p(\boldsymbol{x}) = \int p(\boldsymbol{x}, \boldsymbol{z})\, d\boldsymbol{z} = \int p(\boldsymbol{z})p_\theta(\boldsymbol{x}|\boldsymbol{z})\, d\boldsymbol{z} \tag{8.9}$$

然而，上述积分通常是不可行的（Intractable）。这是因为由深度神经网络建模的条件分布 $p_\theta(\boldsymbol{x}|\boldsymbol{z})$ 具有高度复杂的非线性结构，导致该积分无法获得解析解。

获得数据边缘分布 $p(\boldsymbol{x})$ 的另一种思路是借助概率的链式法则（Chain Rule of Probability）：

$$p(\boldsymbol{x}) = \frac{p(\boldsymbol{x}, \boldsymbol{z})}{p(\boldsymbol{z}|\boldsymbol{x})} \tag{8.10}$$

上式中的 $p(\boldsymbol{z}|\boldsymbol{x})$ 描述了给定观测数据时隐变量的真实后验分布，然而，这一真实后验分布通常是未知且难以直接计算的。为此，我们需要引入**变分推断**（Variational Inference）对其进行近似。变分推断的核心思想是：引入一个具有可学习参数 $\boldsymbol{\phi}$ 且形式相对简单的分布 $q_\phi(\boldsymbol{z}|\boldsymbol{x})$ 来尽可能地逼近未知的真实后验分布 $p(\boldsymbol{z}|\boldsymbol{x})$。在**变分自编码器**（Variational Autoencoder）中，分布 $q_\phi(\boldsymbol{z}|\boldsymbol{x})$ 正是由变分编码器（Variational Encoder）来建模的，其作用是根据输入样本 $\boldsymbol{x}$ 预测对应隐变量 $\boldsymbol{z}$ 的概率分布，而不再仅仅输出一个确定性的隐表示。

我们可以通过以下对数似然的推导过程来理解引入变分推断后对优化目标的影响：

$$\begin{aligned}
\log p(\boldsymbol{x}) &= \int q_\phi(\boldsymbol{z}|\boldsymbol{x}) \log p(\boldsymbol{x})\, d\boldsymbol{z} && \left(\text{因为} \int q_\phi(\boldsymbol{z}|\boldsymbol{x})\, d\boldsymbol{z} = 1\right) \\
&= \mathbb{E}_{q_\phi(\boldsymbol{z}|\boldsymbol{x})}[\log p(\boldsymbol{x})] && \text{（连续随机变量的期望定义）} \\
&= \mathbb{E}_{q_\phi(\boldsymbol{z}|\boldsymbol{x})}\left[\log \frac{p(\boldsymbol{x}, \boldsymbol{z})}{p(\boldsymbol{z}|\boldsymbol{x})}\right] && \text{（概率链式法则公式8.10）} \\
&= \mathbb{E}_{q_\phi(\boldsymbol{z}|\boldsymbol{x})}\left[\log \frac{p(\boldsymbol{x}, \boldsymbol{z})q_\phi(\boldsymbol{z}|\boldsymbol{x})}{p(\boldsymbol{z}|\boldsymbol{x})q_\phi(\boldsymbol{z}|\boldsymbol{x})}\right] && \text{（分子和分母同乘 } q_\phi(\boldsymbol{z}|\boldsymbol{x})\text{）} \\
&= \mathbb{E}_{q_\phi(\boldsymbol{z}|\boldsymbol{x})}\left[\log \frac{p(\boldsymbol{x}, \boldsymbol{z})}{q_\phi(\boldsymbol{z}|\boldsymbol{x})}\right] + \mathbb{E}_{q_\phi(\boldsymbol{z}|\boldsymbol{x})}\left[\log \frac{q_\phi(\boldsymbol{z}|\boldsymbol{x})}{p(\boldsymbol{z}|\boldsymbol{x})}\right] \\
&= \mathbb{E}_{q_\phi(\boldsymbol{z}|\boldsymbol{x})}\left[\log \frac{p(\boldsymbol{x}, \boldsymbol{z})}{q_\phi(\boldsymbol{z}|\boldsymbol{x})}\right] + D_{\mathrm{KL}}(q_\phi(\boldsymbol{z}|\boldsymbol{x}) \| p(\boldsymbol{z}|\boldsymbol{x})) && \text{（根据 KL 散度的定义）}
\end{aligned}$$

在第2.4节中，我们已知 KL 散度（Kullback–Leibler Divergence）可用于衡量两个概率分布之间

的距离，并且其取值始终非负。因此，我们有：

$$\log p(\boldsymbol{x}) = \mathbb{E}_{q_\phi(\boldsymbol{z}|\boldsymbol{x})}\left[\log \frac{p(\boldsymbol{x},\boldsymbol{z})}{q_\phi(\boldsymbol{z}|\boldsymbol{x})}\right] + \underbrace{D_{\mathrm{KL}}(q_\phi(\boldsymbol{z}|\boldsymbol{x})\|p(\boldsymbol{z}|\boldsymbol{x}))}_{\geq 0}$$

$$\geq \underbrace{\mathbb{E}_{q_\phi(\boldsymbol{z}|\boldsymbol{x})}\left[\log \frac{p(\boldsymbol{x},\boldsymbol{z})}{q_\phi(\boldsymbol{z}|\boldsymbol{x})}\right]}_{\text{证据下界（ELBO）}}$$

由于 KL 散度始终非负，因此观测数据的对数似然 $\log p(\boldsymbol{x})$ 总是大于或等于上式中的**证据下界**（Evidence Lower Bound, ELBO）。通过调整模型参数以最大化 ELBO，可以带来两个重要且期望的结果。一方面，由于 ELBO 是对数似然 $\log p(\boldsymbol{x})$ 的下界，因此最大化 ELBO 将间接推动数据对数似然的提升，从而提高模型对数据分布的拟合能力。另一方面，在对数似然 $\log p(\boldsymbol{x})$ 固定的情况下，最大化 ELBO 等价于最小化变分分布 $q_\phi(\boldsymbol{z}|\boldsymbol{x})$ 与真实后验分布 $p(\boldsymbol{z}|\boldsymbol{x})$ 之间的 KL 散度，从而使变分编码器所学习到的近似后验尽可能逼近真实后验分布。

从上述分析可以看出，通过引入变分推断，使得最大化数据的对数似然转化为了最大化其证据下界。为了推导出具体的损失函数，我们对证据下界 ELBO 做进一步展开：

$$\begin{aligned}
\mathbb{E}_{q_\phi(\boldsymbol{z}|\boldsymbol{x})}\left[\log \frac{p(\boldsymbol{x},\boldsymbol{z})}{q_\phi(\boldsymbol{z}|\boldsymbol{x})}\right] &= \mathbb{E}_{q_\phi(\boldsymbol{z}|\boldsymbol{x})}\left[\log \frac{p_\theta(\boldsymbol{x}|\boldsymbol{z})p(\boldsymbol{z})}{q_\phi(\boldsymbol{z}|\boldsymbol{x})}\right] && \text{（概率链式法则）} \\
&= \mathbb{E}_{q_\phi(\boldsymbol{z}|\boldsymbol{x})}\left[\log p_\theta(\boldsymbol{x}|\boldsymbol{z})\right] + \mathbb{E}_{q_\phi(\boldsymbol{z}|\boldsymbol{x})}\left[\log \frac{p(\boldsymbol{z})}{q_\phi(\boldsymbol{z}|\boldsymbol{x})}\right] \\
&= \underbrace{\mathbb{E}_{q_\phi(\boldsymbol{z}|\boldsymbol{x})}\left[\log p_\theta(\boldsymbol{x}|\boldsymbol{z})\right]}_{\text{重构项}} - \underbrace{D_{\mathrm{KL}}(q_\phi(\boldsymbol{z}|\boldsymbol{x})\|p(\boldsymbol{z}))}_{\text{先验约束项}} && \text{（KL 散度的定义）}
\end{aligned}$$

因此，最大化证据下界等价于同时最大化重构项并最小化先验约束项。上述两项都具有明确且直观的意义。第一个重构项保证了解码器 $p_\theta(\boldsymbol{x}|\boldsymbol{z})$ 能够从变分后验分布 $q_\phi(\boldsymbol{z}|\boldsymbol{x})$ 采样得到的隐变量 $\boldsymbol{z}$ 中尽可能重构出原始观测样本 $\boldsymbol{x}$。第二个先验约束项则迫使变分后验分布 $q_\phi(\boldsymbol{z}|\boldsymbol{x})$ 尽可能与我们预设的先验分布 $p(\boldsymbol{z})$ 相似，从而让编码器 $q_\phi(\boldsymbol{z}|\boldsymbol{x})$ 能够学习一个连续的概率分布，而不是退化为一个单点的狄拉克 δ 函数，从而保证隐空间具有良好的连续性与可采样性。

变分推断需要为近似后验分布 $q_\phi(\boldsymbol{z}|\boldsymbol{x})$ 选择形式相对简单的概率分布。我们通常采用协方差矩阵为对角形式的多元高斯分布：

$$q_\phi(\boldsymbol{z}|\boldsymbol{x}) = \mathcal{N}(\boldsymbol{z}|\boldsymbol{\mu}_\phi(\boldsymbol{x}), \mathrm{diag}(\boldsymbol{\sigma}_\phi^2(\boldsymbol{x}))) \tag{8.11}$$

其中 $\boldsymbol{\mu}_\phi(\boldsymbol{x})$ 和 $\boldsymbol{\sigma}_\phi^2(\boldsymbol{x})$ 分别表示由深度神经网络编码器输出的均值向量与方差向量，$\mathrm{diag}(\cdot)$ 表示以该向量元素构成对角线的对角协方差矩阵。由于隐变量的先验分布 $p(\boldsymbol{z})$ 被设定为多元标准高斯分布，而两个多元高斯分布之间的 KL 散度存在如下通用的解析形式：

$$D_{\mathrm{KL}}(\mathcal{N}(\boldsymbol{\mu}_1,\boldsymbol{\Sigma}_1)\|\mathcal{N}(\boldsymbol{\mu}_2,\boldsymbol{\Sigma}_2)) = \frac{1}{2}\left(\mathrm{tr}(\boldsymbol{\Sigma}_2^{-1}\boldsymbol{\Sigma}_1) + (\boldsymbol{\mu}_2-\boldsymbol{\mu}_1)^\top\boldsymbol{\Sigma}_2^{-1}(\boldsymbol{\mu}_2-\boldsymbol{\mu}_1) - D + \log\frac{|\boldsymbol{\Sigma}_2|}{|\boldsymbol{\Sigma}_1|}\right)$$

其中 D 为高斯分布的维度，$\mathrm{tr}(\cdot)$ 表示矩阵的迹，而 $|\cdot|$ 表示矩阵的行列式。

得益于对近似后验分布与隐变量先验分布的特定选择，证据下界中的先验约束项（即 KL 散度）存在闭式解，而证据下界中的重构项通常无法直接计算，需要借助蒙特卡洛估计（Monte

Carlo Estimation）进行近似。蒙特卡洛估计（详见第11章）的核心思想是：通过从目标分布中随机采样，并利用样本均值来近似难以直接计算的期望值。根据大数定律，当采样数量足够大时，样本均值会逐渐收敛到真实的期望值。因此，变分自编码器的代价函数可以写成如下形式：

$$\begin{aligned}\mathcal{J}(\boldsymbol{\phi},\boldsymbol{\theta}) &= -\mathbb{E}_{q_\phi(\boldsymbol{z}|\boldsymbol{x})}\left[\log p_\theta(\boldsymbol{x}|\boldsymbol{z})\right] + D_{\mathrm{KL}}(q_\phi(\boldsymbol{z}|\boldsymbol{x})\|p(\boldsymbol{z})) \\ &\approx \underbrace{-\frac{1}{L}\sum_{l=1}^{L}\log p_\theta(\boldsymbol{x}|\boldsymbol{z}^{(l)})}_{\text{重构项}} + \underbrace{D_{\mathrm{KL}}(q_\phi(\boldsymbol{z}|\boldsymbol{x})\|p(\boldsymbol{z}))}_{\text{先验约束项}}\end{aligned} \tag{8.12}$$

其中 $\{\boldsymbol{z}^{(l)}\}_{l=1}^{L}$ 表示对于给定观测样本 $\boldsymbol{x}$，从变分后验分布 $q_\phi(\boldsymbol{z}|\boldsymbol{x})$ 中进行独立同分布采样得到的 L 个隐表示，用于对重构项中的期望进行蒙特卡洛近似。变分自编码器的编码器与解码器通常均由深度神经网络构建，并采用随机梯度下降法及其变体进行优化。然而，蒙特卡洛估计所需的随机采样过程是不可导的，这会导致梯度在反向传播过程中因采样而中断。为了解决这一问题，需要引入**重参数化**（Reparameterization）技巧。其核心思想是：将待采样的随机变量表示为确定性函数与随机噪声变量的组合，从而将随机性与模型参数分离，使模型参数仍然能够通过反向传播进行优化。以一维高斯分布为例，假设随机变量 x 服从均值为 μ、方差为 σ^2 的高斯分布 $\mathcal{N}(x|\mu,\sigma^2)$。通过重参数化，样本 x 可以改写为：

$$x = \mu + \epsilon\sigma \tag{8.13}$$

其中 $\epsilon \sim \mathcal{N}(\epsilon|0,1)$ 为服从标准高斯分布的随机噪声。换言之，任意高斯分布都可以看作是由标准高斯分布经过线性变换后得到：即先对标准高斯噪声进行缩放（乘以目标标准差 σ），再进行平移（加上目标均值 μ）。因此，从任意高斯分布中的采样过程，可以等价地转化为：先从标准高斯分布中采样得到 ϵ，再对其进行确定性的线性变换。如此，随机性便被转移到了与模型参数无关的噪声变量 ϵ 中，而变量 x 则成为关于 μ 与 σ 的可导函数。因此，损失函数关于 x 的梯度仍然能够反向传播到 μ 和 σ。同理，从如公式8.11所示的多元高斯近似后验分布 $q_\phi(\boldsymbol{z}|\boldsymbol{x})$ 中采样得到的隐变量 $\boldsymbol{z}$ 也可以写为：

$$\boldsymbol{z} = \boldsymbol{\mu}_\phi(\boldsymbol{x}) + \boldsymbol{\epsilon}\odot\boldsymbol{\sigma}_\phi(\boldsymbol{x}) \tag{8.14}$$

其中 $\boldsymbol{\epsilon} \sim \mathcal{N}(\boldsymbol{\epsilon}|\mathbf{0},\mathbf{I})$ 为多元标准高斯分布，符号 $\odot$ 表示向量的逐元素相乘，而 $\boldsymbol{\mu}_\phi(\boldsymbol{x})$ 与 $\boldsymbol{\sigma}^2_\phi(\boldsymbol{x})$ 分别为编码器神经网络输出的均值向量 $f^\mu_\phi(\boldsymbol{x})$ 与方差向量 $f^\Sigma_\phi(\boldsymbol{x})$（协方差矩阵为对角矩阵）。

引入上述重参数化技巧后，变分自编码器重构项的前向计算如图8.6所示（隐空间中分布的均值向量与方差向量分别简写为 $\boldsymbol{\mu}_{\boldsymbol{z}}$ 和 $\boldsymbol{\sigma}^2_{\boldsymbol{z}}$）。其具体的计算步骤如下：

- 首先从训练集中随机采样一个原始观测样本 $\boldsymbol{x}$，并利用编码器网络 f_ϕ 预测该样本在隐空间中所对应高斯分布的均值向量 $\boldsymbol{\mu}_{\boldsymbol{z}} = f^\mu_\phi(\boldsymbol{x})$ 和协方差矩阵的对角线元素 $\boldsymbol{\sigma}^2_{\boldsymbol{z}} = f^\Sigma_\phi(\boldsymbol{x})$；
- 然后利用重参数化公式8.14，结合从标准高斯分布中采样得到的随机噪声 $\boldsymbol{\epsilon}$，生成对应的隐变量表示 $\boldsymbol{z} = \boldsymbol{\mu}_{\boldsymbol{z}} + \boldsymbol{\epsilon}\odot\boldsymbol{\sigma}_{\boldsymbol{z}}$（为了表示简洁，图8.6中省略了逐元素相乘符号 $\odot$）；
- 接着将 $\boldsymbol{z}$ 输入解码器网络 g_θ，得到重构样本 $\boldsymbol{x}'$；
- 最后以最小化原始样本与重构样本之间的均方误差 $\|\boldsymbol{x}'-\boldsymbol{x}\|^2$ 来联合优化编码器参数 $\boldsymbol{\phi}$ 与

解码器参数 $\boldsymbol{\theta}$。

正如前文所述，在假设观测数据的条件概率分布服从多元高斯分布且噪声方差为常数超参数的前提下，最大化数据的对数似然完全等价于最小化原始样本 $\boldsymbol{x}$ 与重构样本 $\boldsymbol{x}'$ 之间的欧几里得均方误差。另一方面，代价函数中的先验约束项在后验分布与先验分布均为高斯分布时，可以直接利用 KL 散度的解析式进行计算与优化。在变分自编码器的参数训练过程中，重构项与先验约束项共同作用、相互制衡。重构项迫使模型捕捉输入数据的关键特征，将不同观测样本的隐表示映射到隐空间中的独特位置，以确保解码器能够准确重构；而先验约束项则可以视为一种正则化，它通过 KL 散度将每个样本对应的条件后验分布 $q_{\phi}(\boldsymbol{z}|\boldsymbol{x})$ 约束到预设的先验分布 $p(\boldsymbol{z})$。这两部分目标的共同作用，使得隐空间既能够保留数据的关键信息，又具有良好的连续性与平滑性。也就是说，隐空间中彼此接近的隐变量通常对应着语义上相似的数据样本。当模型训练正常收敛后，我们可以直接从标准高斯先验分布 $p(\boldsymbol{z})$ 中进行随机采样，并将采样得到的隐变量送入解码器网络，从而生成训练集中未曾出现过、但仍符合真实数据分布特征的新样本。

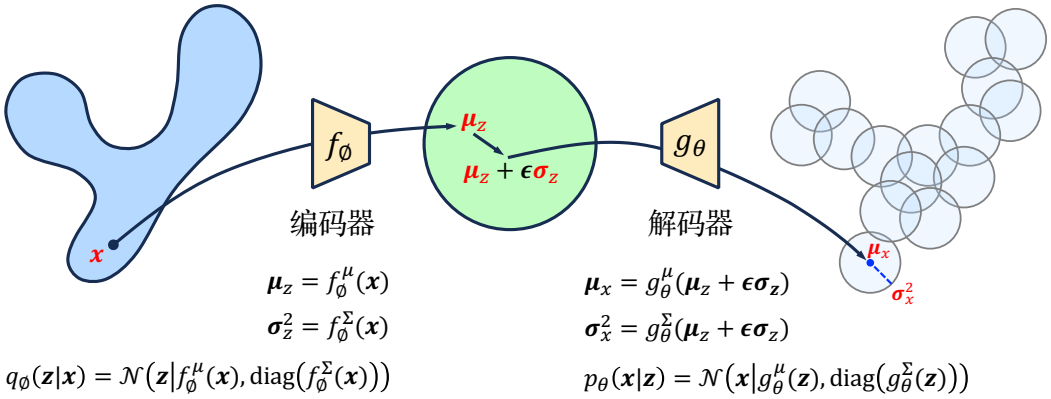


图 8.6: 变分自编码器的编码与解码流程示意图。输入样本首先通过编码器计算出其在隐空间中对应的多元高斯分布的均值与方差。然后从该分布中随机采样出一个隐表示，并将其输入解码器以重构原始输入样本。最终通过比较原始样本与重构结果之间的差异来计算重构损失。

根据神经网络的通用近似定理，只要网络具备足够的深度或宽度，理论上它就能够以任意精度逼近紧集上的复杂连续非线性函数。这为解码器网络学习复杂的非线性映射提供了理论基础，使其有可能从简单的多元标准高斯先验分布出发，构建能够逼近复杂真实数据分布 $p(\boldsymbol{x})$ 的生成模型[1]。在构建出良好的隐空间之后，我们不仅能够进行无条件随机采样，还可以对其进行操控或者引入条件约束以实现定向的控制生成。以“文本引导”的条件生成为例。在训练阶段，首先利用预训练语言模型（如第7.7节介绍的 BERT 或 GPT 模型）作为文本编码器，将图像的语义文字描述转化为文本特征向量；随后将该文本特征向量与隐空间中采样得到的隐表示进行拼接（Concatenation），共同作为解码器网络的输入进行训练。在生成（或推理）阶段，采取完全相同的流程，即将期望生成图像的文本描述输入相同的文本编码器，提取其文本特征向量，并将其与从标准高斯分布中随机采样得到的隐表示向量进行拼接。解码器在接收到这一包含了语

[1]严格地说，通用近似定理保证的是连续函数的逼近能力，而生成模型对概率分布的逼近能力则还取决于隐变量维度、解码器的条件分布形式以及其他模型假设。

义引导信息的向量后，便能定向生成与文本描述相符的图像。值得注意的是：由于随机采样得到的隐变量不同，即使输入相同的文本描述，模型也能够生成内容细节各异但语义一致的多种结果。这种机制使得生成模型不仅能够“理解”文本语义，还能够在满足语义约束的前提下生成丰富、多样的内容。

8.3 生成对抗网络

生成模型的核心目标是使生成或合成的数据与真实数据尽可能的相似。此前所介绍的变分自编码器主要采用重构的思想来显式建模观测数据的分布 $p(\boldsymbol{x})$，从而使其解码器（即生成器）能够生成与真实数据相似的图像。然而，我们是否有一种方法能够直接衡量生成器所生成图像与真实图像的相似程度呢？**生成对抗网络**（Generative Adversarial Network, GAN）提供了一种全新且直观的思路：如果一个训练良好的判别器（通常为分类器）完全无法区分合成图像和真实图像，那么便可以认为这些合成数据已经与真实数据足够相似 [30]。

具体而言，生成对抗网络同时训练两个神经网络：生成器（Generator）负责将随机噪声映射为合成图像；而判别器（Discriminator）负责判断输入图像究竟来自真实数据集，还是由生成器合成得到。在训练过程中，生成器与判别器处于一个典型的双人零和博弈（Zero-sum Game）中：生成器的目标是最大化判别器的误判概率，即让其生成的图像能够“骗”过判别器；而判别器的目标则是尽可能准确地将生成图像与真实图像区分开来。两者相互对抗，并在对抗中不断提升各自的能力。在理论上理想的情况下，当模型达到博弈的纳什均衡（Nash Equilibrium）时，生成器将能够完美地捕获真实数据的分布，实现“以假乱真”；而此时的判别器由于完全无法从概率上分辨图像的真伪，其预测概率将稳定在 0.5，即与随机猜测无异。需要指出的是，这里的“判别器预测概率为 0.5”并不意味着判别器性能退化，而是意味着生成图像与真实图像在统计意义上已经无法区分。这表明生成器成功学习到了真实数据的分布。在实际应用中，人们通常更加关注训练完成后的生成器，因为它将承担生成新图像的任务；而判别器主要作为训练阶段的辅助模块，为生成器提供学习信号和优化方向。

正如文献 [30] 中所采用的经典比喻，生成对抗网络的训练过程可以看作一场持续进行的“猫鼠游戏”。在这一框架下，生成器类似于一伙假钞制造者（鼠），其核心目标是不断改进工艺，试图制造并使用假币而不被发现；而判别器则类似于负责打击假币的警方（猫），努力提高侦查技术以精准识别这些伪造的货币。在这场零和博弈中，竞争压力促使双方交替升级各自的方法与技术。这种针锋相对的对抗与改进过程会不断演进，直到假钞的逼真程度与真品在统计上变得难以区分。

如图8.7所示，生成对抗网络中的判别器通常采用卷积神经网络等架构。它以真实图像或合成图像作为输入，并输出一个表示输入属于真实图像的概率值。假设真实图像服从数据分布 $p(\boldsymbol{x})$，并用 $D_\phi(\boldsymbol{x})$ 表示判别器将输入样本 $\boldsymbol{x}$ 判定为真实图像（而非生成图像）的概率，其中 $\boldsymbol{\phi}$ 是判别器网络的参数。为了使判别器能够正确识别真实图像，希望其对于来自真实数据分布的

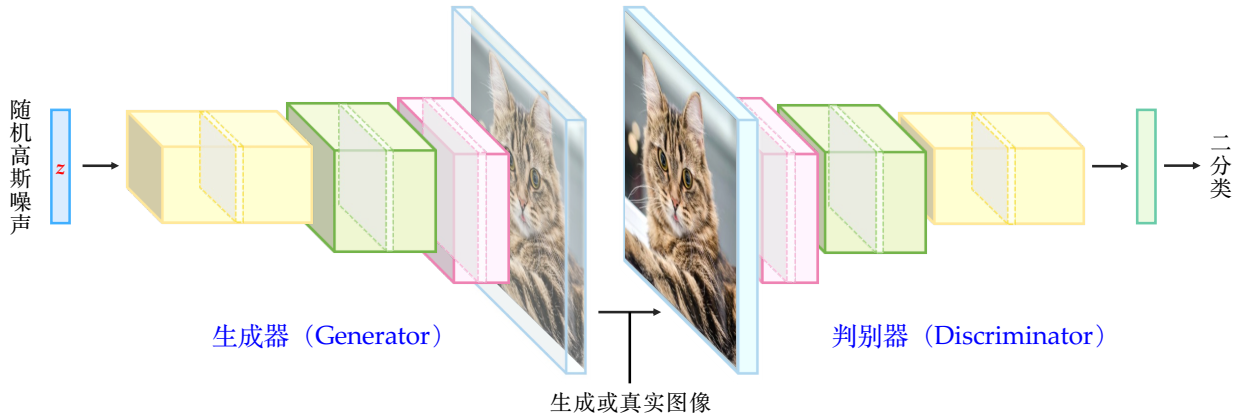


图 8.7: 生成对抗网络示意图。生成器以随机噪声向量为输入，并生成相应的合成图像；判别器以真实图像或生成图像为输入，输出输入样本属于真实图像的概率。训练过程中，生成器试图生成更加逼真的图像以欺骗判别器，而判别器则尽力区分真实图像与生成图像。两者通过对抗形成双人零和博弈（Two-player Minimax Game）。经过多轮对生成器与判别器的交替优化之后，生成分布逐渐逼近真实数据分布，并最终达到纳什均衡。

样本输出尽可能接近于 1 的概率。因此，对应的优化目标可以写为：

$$\mathbb{E}_{\boldsymbol{x}\sim p(\boldsymbol{x})}\left[\log D_\phi(\boldsymbol{x})\right] \tag{8.15}$$

生成器 $G_\theta(\boldsymbol{z})$ 则可以采用如图8.3所示的基于转置卷积的卷积生成网络，它以随机噪声向量 $\boldsymbol{z}$ 作为输入，并输出相应的合成图像。生成器的训练目标是最大化判别器的误判概率，即尽可能让判别器将合成图像误判为真实图像。对于生成样本 $G_\theta(\boldsymbol{z})$，判别器希望其输出概率 $D_\phi(G_\theta(\boldsymbol{z}))$ 接近于 0，即判定其为生成图像；而生成器则希望判别器的输出尽可能接近于 1，即使判别器将生成图像误认为真实图像。因此，生成器通过优化其参数 $\boldsymbol{\theta}$ 来最小化如下期望：

$$\mathbb{E}_{\boldsymbol{z}\sim p(\boldsymbol{z})}\left[\log(1-D_\phi(G_\theta(\boldsymbol{z})))\right] \tag{8.16}$$

其中噪声分布 $p(\boldsymbol{z})$ 通常设为多元标准高斯分布，即 $\boldsymbol{z}\sim\mathcal{N}(\boldsymbol{z}|\mathbf{0},\mathbf{I})$。综合判别器与生成器各自的优化目标（分别对应公式8.15和公式8.16），生成对抗网络的双人零和博弈（Two-player Minimax Game）的目标函数（亦称价值函数）可以表示为如下的极小极大优化问题：

$$\min_\theta \max_\phi \{\mathbb{E}_{\boldsymbol{x}\sim p(\boldsymbol{x})}\left[\log D_\phi(\boldsymbol{x})\right] + \mathbb{E}_{\boldsymbol{z}\sim p(\boldsymbol{z})}\left[\log(1-D_\phi(G_\theta(\boldsymbol{z})))\right]\} \tag{8.17}$$

在实际训练过程中，为了使生成器与判别器这两者形成有效的对抗，通常采用交替迭代的训练策略，即轮流更新判别器和生成器的参数，直至满足预设的停止条件。当更新判别器时，除了使用真实图像作为正样本外，还需要利用当前生成器生成的合成图像作为负样本，以训练判别器区分真实图像与生成图像的能力。具体而言，在判别器的每次更新过程中，分别从真实数据分布和噪声分布中随机采样 B 个小批量样本，并利用蒙特卡洛方法对前述期望（即公式8.15）进行近似。因此，判别器的经验优化目标可以表示为最大化如下对数似然：

$$\frac{1}{B}\sum_{j=1}^{B}\left(\log D_\phi(\boldsymbol{x}_j) + \log(1-D_\phi(G_\theta(\boldsymbol{z}_j)))\right) \tag{8.18}$$

其中 $\boldsymbol{x}_j$ 表示小批量中的第 j 个真实图像，$\boldsymbol{z}_j$ 表示输入生成器的第 j 个随机噪声向量。目标函数中的第一项旨在鼓励判别器为真实图像输出更高的置信概率，第二项则旨在引导判别器对生成的图像输出更低的概率值。通过同时优化这两项，判别器能够逐步精确构建区分真实图像与合成图像的决策边界。生成对抗网络的训练过程如算法8.1所示，而训练的演进过程如图8.8所示。

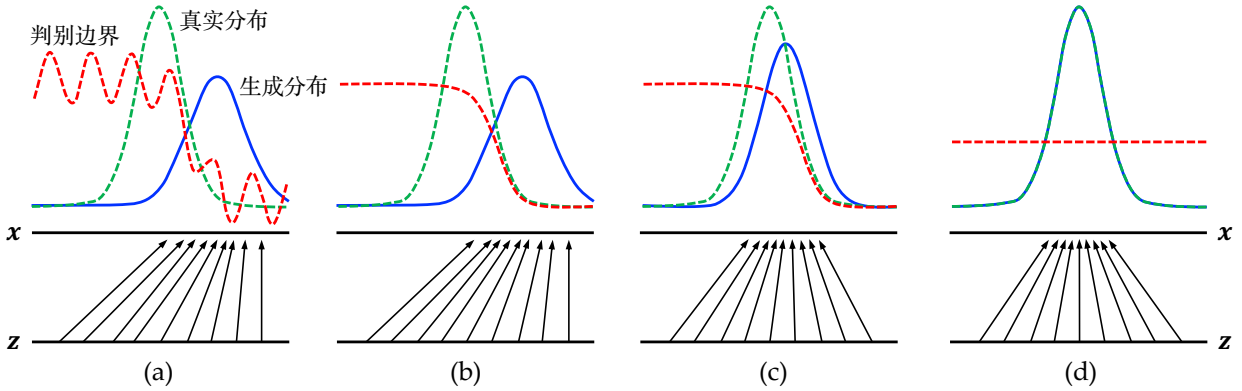


图 8.8: 生成对抗网络训练演进示意图。在训练过程中，交替更新判别器 $D_\phi(\boldsymbol{x})$ 的参数和生成器 $G_\theta(\boldsymbol{z})$ 的参数。图中红色虚线表示判别器的输出曲线，蓝色实线表示生成器产生的数据分布。通过判别器与生成器参数的交替优化，在理想情况下，生成数据分布逐步向真实数据分布逼近。图中的绿色虚线表示真实数据分布 $p(\boldsymbol{x})$。最下方水平线表示隐变量 $\boldsymbol{z}$ 的取值空间，而上方水平线表示数据变量 $\boldsymbol{x}$ 的取值空间，向上的箭头表示生成器所学习到的映射关系 $\boldsymbol{x} = G_\theta(\boldsymbol{z})$。(a) 生成分布与真实数据分布仍存在一定差异，判别器能够在一定程度上区分真实样本与生成样本。(b) 固定生成器参数，更新判别器参数，使判别器能够更准确地区分真实样本与生成样本。(c) 固定判别器参数，更新生成器参数，使生成分布向真实数据分布进一步靠近，从而提高判别器的判别难度。(d) 经过多轮交替优化后，生成分布逐渐逼近真实数据分布。在理想条件下，最终达到纳什均衡。此时，判别器无法有效区分真实样本与生成样本，其预测概率在全空间稳定为一条直线。此例来自文献 [30]。

算法 8.1: 生成对抗网络训练算法

输入: 真实数据集 $\mathcal{D} = \{\boldsymbol{x}_i\}_{i=1}^N$、批量大小 B 和最大迭代次数 H。

初始化: 随机初始化生成器参数 $\boldsymbol{\theta}$ 与判别器参数 $\boldsymbol{\phi}$，并设当前迭代次数 $h = 1$；

while $h \leq H$ **do**

  从真实数据集 $\mathcal{D}$ 中随机采样 B 个真实样本 $\{\boldsymbol{x}_1, \ldots, \boldsymbol{x}_B\}$；

  从多元标准高斯分布中随机采样 B 个噪声向量 $\{\boldsymbol{z}_1, \ldots, \boldsymbol{z}_B\}$；

  最大化以下经验目标函数（即公式8.18）来更新判别器参数 $\boldsymbol{\phi}$:

    $\frac{1}{B}\sum_{j=1}^{B}(\log D_\phi(\boldsymbol{x}_j) + \log(1 - D_\phi(G_\theta(\boldsymbol{z}_j))))$

  最小化以下经验目标函数（即公式8.16）来更新生成器参数 $\boldsymbol{\theta}$:

    $\frac{1}{B}\sum_{j=1}^{B}\log(1 - D_\phi(G_\theta(\boldsymbol{z}_j)))$

  $h \leftarrow h + 1$；

输出: 经训练后得到的生成器参数 $\boldsymbol{\theta}$ 和判别器参数 $\boldsymbol{\phi}$。

在实际训练中，判别器本质上执行的是二分类任务，相较于复杂的图像生成任务，其拟合与收敛速度通常较快。因此，在生成对抗网络训练的早期阶段，当生成器尚未具备良好的图像

生成能力时，判别器可以轻而易举地分辨出真实图像与合成图像。此时，对于生成样本 $G_\theta(\boldsymbol{z})$，判别器输出的概率 $D_\phi(G_\theta(\boldsymbol{z}))$ 往往接近于 0。在这种情况下，生成器从判别器获得的梯度信号会显著减弱，从而导致训练过程出现严重的梯度消失问题。为了缓解这一问题，并在训练初期为生成器提供更强的学习信号，通常会将生成器的优化目标调整为最大化如下经验目标函数：

$$\frac{1}{B}\sum_{j=1}^{B}\log D_\phi(G_\theta(\boldsymbol{z}_j)) \tag{8.19}$$

其中 B 是批量大小。上述对生成器优化目标的修正并不会改变在理想条件下的目标平衡点，即生成数据分布与真实数据分布相一致时的平衡状态，但它能够在判别器过于强势的训练初期，为生成器提供较大的梯度，从而加速模型的训练。

生成对抗网络同样可以扩展为条件生成对抗网络（Conditional GAN），以实现受控的条件生成。仍以“文本引导”的条件生成为例 [62]，其核心思想是将条件信息（文本）同时引入生成器和判别器，从而引导模型学习文本语义与图像内容之间的对应关系。对于生成器，首先利用预训练语言模型（如第7.7节介绍的 BERT 或 GPT）将输入的文本描述转换为文本特征向量，然后将该文本特征向量与随机噪声向量进行拼接，共同作为生成器的输入，以引导图像的定向生成。对于判别器，其任务不再仅仅是判断输入图像是真实图像还是生成图像，还需要评估图像内容是否与给定文本描述匹配。因此，在判别器提取图像特征的过程中，通常会将相同的文本特征向量与图像特征进行融合，并基于融合后的联合表征完成最终判别。换言之，判别器不仅需要识别图像的真实性，还需要判断图像与文本之间的语义一致性。当模型训练完成之后，在定向生成阶段，只需要将文本描述转换为对应的文本特征向量，再与随机噪声向量进行拼接并输入生成器，即可生成符合文本语义描述的新图像。

8.4 降噪扩散模型

训练完成后的变分自编码器在生成新图像时，首先从多元标准高斯分布中随机采样得到隐变量，然后将其输入解码器网络生成图像。根据神经网络的通用近似定理，只要解码器具有足够的表达能力，理论上就能够逼近复杂的非线性映射。这为变分自编码器从简单的多元标准高斯分布出发，学习并生成服从复杂数据分布 $p(\boldsymbol{x})$ 的图像提供了理论基础。然而，变分自编码器试图通过一次映射，直接将随机采样得到的隐变量转换为高维且复杂的真实图像。这种“一步生成”的方式往往难以兼顾图像从宏观结构到微观纹理的多尺度特征。此外，在采用均方重构损失进行训练时，如果同一隐表示对应多种可能的图像，解码器往往倾向于输出这些图像的条件均值，而不是其中某一种具体结果。这种对不同可能结果的平均效应，容易使重构或生成的图像出现边缘模糊、纹理不清和细节缺失等问题。如图8.9(a) 所示，变分自编码器的核心机制可简化为：首先利用编码器 $q(\boldsymbol{z}|\boldsymbol{x})$ 将原始图像 $\boldsymbol{x}$ 映射到对应的隐变量 $\boldsymbol{z}$，再通过解码器 $p(\boldsymbol{x}|\boldsymbol{z})$ 实现从隐变量到原图像的重构。为了克服单步生成的局限性，并能够生成更清晰且细节更丰富的图像，一种自然的改进方案是采用“分而治之”的思想，即将单步生成扩展为包含多个中间隐变量的多步渐进式生成。**降噪扩散模型**（Denoising Diffusion Model）的核心思想正是通过引入分

层结构（即构建多个连续的隐层空间），将传统的单层变分自编码器扩展为深度的分级变分自编码器（如图8.9(b) 所示）。从这一视角来看，降噪扩散模型在本质上可以理解为一种具有多层次隐变量（多级抽象）的生成模型 [51]。

多步生成同样始于从多元标准高斯分布中随机采样得到的隐变量。然而，与变分自编码器的一步生成不同，逐步生成机制能够将复杂的生成任务分解为一系列相对简单的步骤，从而有助于不同尺度信息的逐步形成与细化。通常情况下，在生成过程的早期阶段，模型倾向于首先确定图像的整体布局和全局结构，例如物体的大致轮廓及其空间关系；中间阶段会逐渐形成更加丰富的语义信息，例如颜色分布、形状特征以及组成部分之间的关系；而在生成过程的后期阶段，模型则进一步补充和完善局部细节，例如材质纹理、边缘结构以及其他高频信息。如果将图像生成过程比作绘画，那么传统自编码器更像是一位初学者，试图通过一次落笔来完成整幅写实作品。但由于任务过于复杂，往往只能得到大致的轮廓以及较为模糊的细节。相比之下，扩散模型更像是一位经验丰富的画家：首先用简洁的线条勾勒出整体构图（高噪声阶段），随后不断调整和丰富画面的主体结构（中噪声阶段），最后再用细致的笔触逐步描绘纹理、边缘和材质等细节（低噪声阶段）。正是这种由粗到细、逐步细化的生成方式，使得扩散模型能够更好地兼顾图像的全局结构与局部细节，从而生成更加清晰和逼真的图像。

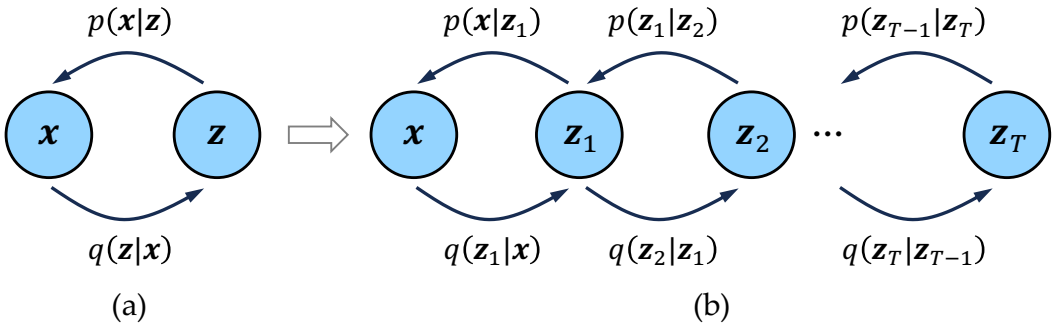


图 8.9: 变分自编码器与降噪扩散模型对比示意图。降噪扩散模型在本质上可以理解为将传统“一步生成”的变分自编码器扩展为具有深度分层结构的多步渐进式生成模型。

尽管引入多层隐变量能够实现“分而治之”的生成策略，将复杂的生成任务分解为多个相对简单的步骤。与此同时，也带来了如何设计和学习中间隐变量分布的问题。如果不对这些中间隐变量的分布及其状态转移路径施加适当的约束，模型将面临巨大的学习空间，需要同时学习复杂的中间隐表示以及它们之间的转移规律，从而显著增加训练难度。为了解决这一关键问题，降噪扩散模型引入了以下三个关键约束：

- 所有中间隐变量 $\boldsymbol{z}_t$ 的维度与真实数据 $\boldsymbol{x}$ 的维度相同。
- 每个时间步的前向转移分布 $q(\boldsymbol{z}_t|\boldsymbol{z}_{t-1})$ 无需学习，而预定为线性高斯模型。换言之，当前时刻隐变量服从一个以前一时刻隐变量的线性函数为均值、协方差矩阵元素为预设时间相关常数的多元高斯分布。
- 在最终时间步 T 时，隐变量 $\boldsymbol{z}_T$ 的分布为一个多元标准高斯分布，即服从 $\mathcal{N}(\boldsymbol{z}_T|\mathbf{0},\mathbf{I})$。

上述约束实际上规定了从真实数据 $\boldsymbol{x}$ 到最终时间步的隐变量 $\boldsymbol{z}_T$（纯高斯噪声）的演化路程（即

前向扩散过程）。该过程通常通过逐步加入高斯噪声来实现，从而在多步变换后将真实数据逐渐转化为标准高斯分布。由这些约束所确定的前向扩散轨迹，为学习其反向生成过程 $p(\boldsymbol{z}_{t-1}|\boldsymbol{z}_t)$ 提供了明确的渐进去噪方向。在模型训练完成后，其生成过程从多元标准高斯噪声出发，通过逐步反向去噪恢复出符合真实数据分布的图像。降噪扩散模型亦因此而得名。

为了符合降噪扩散模型的符号使用习惯，如图8.10所示，我们将时刻（或步骤）t 对应的隐变量由原先的 $\boldsymbol{z}_t$ 统一改写为 $\boldsymbol{x}_t$。在此符号体系下，整个扩散过程采用了统一的变量表示：真实数据（如真实图像）记为 $\boldsymbol{x}_0$，中间特定时间步 t 对应的隐变量记为 $\boldsymbol{x}_t$，而最终时间步 T 对应的纯高斯噪声则记为 $\boldsymbol{x}_T$。

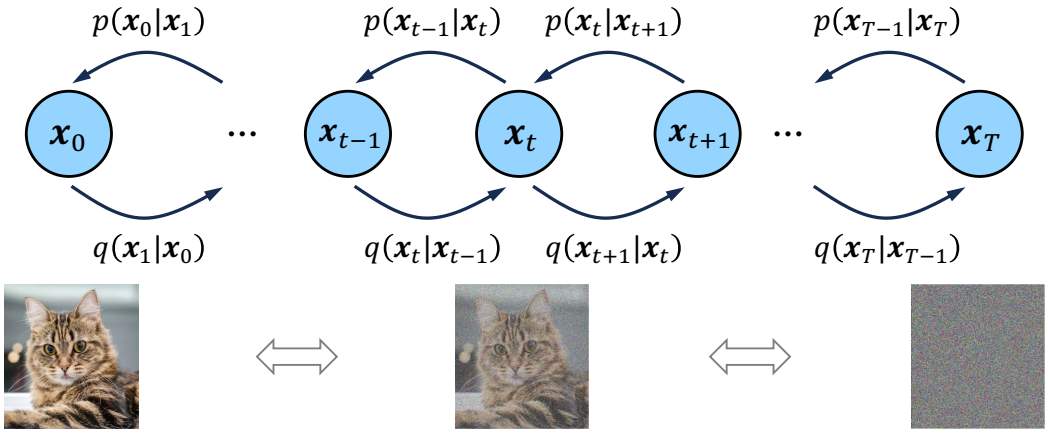


图 8.10: 降噪扩散模型原理与状态转移示意图。其中 $\boldsymbol{x}_0$ 表示真实观测数据（如自然图像），$\boldsymbol{x}_T$ 表示纯高斯噪声，而 $\boldsymbol{x}_t$ 表示经过 t 步加噪后得到的中间隐变量。每个条件分布 $q(\boldsymbol{x}_t|\boldsymbol{x}_{t-1})$ 均被建模为多元高斯分布，并满足马尔科夫性质，即当前状态 $\boldsymbol{x}_t$ 仅依赖于前一步状态 $\boldsymbol{x}_{t-1}$。

不同于变分自编码器，降噪扩散模型的前向编码器 $q(\boldsymbol{x}_t|\boldsymbol{x}_{t-1})$ 并不包含可学习的参数，其核心作用在于通过预先定义的加噪机制构造前向扩散过程，为训练反向生成过程中所使用的解码器 $p(\boldsymbol{x}_{t-1}|\boldsymbol{x}_t)$ 提供明确的扩散轨迹。另一个需要特别注意的是，降噪扩散模型并非为每个特定的时间步 t 单独训练一个独立的解码器，而是在所有时间步中共享同一个解码器网络。训练完成之后，该解码器网络将在生成过程中被反复多次调用，以逐步完成从噪声到数据的去噪过程。为了使同一个网络能够适应不同时间步对应的去噪任务，模型通常会将时间步 t 作为额外条件输入到解码器中。为了使整个前向扩散过程具有良好的统计性质，并便于在不同时间步共享同一个降噪网络，降噪扩散模型通常采用精心设计的加噪机制。其中一个重要的设计思想是方差保持（Variance-preserving），即通过适当控制每一步的信号缩放与噪声强度，使隐变量的总体方差保持在相对稳定的范围内。例如，在去噪扩散概率模型（Denoising Diffusion Probabilistic Models, DDPM）中所采用的方差保持的前向扩散过程中，若初始数据的协方差矩阵为单位矩阵，即 $\mathrm{Cov}(\boldsymbol{x}_0) = \mathbf{I}$，那么各时间步的隐变量也满足：

$$\mathrm{Cov}[\boldsymbol{x}_t] = \mathrm{Cov}[\boldsymbol{x}_{t-1}] = \mathbf{I},\ t \in \{1, \ldots, T\} \tag{8.20}$$

这种方差保持特性使得不同时间步的隐变量始终处于相近的数值尺度，从而避免信号幅度在扩散过程中出现过快衰减或增长。同时，它也使得同一个解码器网络能够在所有时间步上处理具

有相似统计特性的输入与输出，从而提高模型训练与推理过程的稳定性。因此，前向扩散过程通常被预定义为：

$$q(\boldsymbol{x}_t|\boldsymbol{x}_{t-1}) = \mathcal{N}(\boldsymbol{x}_t|\sqrt{\alpha_t}\boldsymbol{x}_{t-1}, (1-\alpha_t)\mathbf{I}) \tag{8.21}$$

其中超参数 α_t 会按照预先设定的方差调度策略（如线性或余弦调度）随时间步推进逐步递减，以确保最终时刻隐变量的边际分布 $p(\boldsymbol{x}_T)$ 收敛到多元标准高斯分布。以下证明为何公式8.21所定义的前向扩散过程能够保持方差的稳定性。假设：

$$\text{Cov}[\boldsymbol{x}_{t-1}] = \mathbf{I} \tag{8.22}$$

根据前向加噪的定义，当前时刻的隐变量可表示为：

$$\boldsymbol{x}_t = \sqrt{\alpha_t}\boldsymbol{x}_{t-1} + \sqrt{(1-\alpha_t)}\boldsymbol{\epsilon} \tag{8.23}$$

其中 $\boldsymbol{\epsilon} \sim \mathcal{N}(\boldsymbol{\epsilon}|\mathbf{0}, \mathbf{I})$ 为当前步引入的多元标准高斯噪声。由于 $\boldsymbol{x}_{t-1}$ 与当前噪声 $\boldsymbol{\epsilon}$ 相互独立，则隐变量 $\boldsymbol{x}_t$ 的协方差矩阵可以计算为：

$$\text{Cov}[\boldsymbol{x}_t] = \alpha_t\text{Cov}[\boldsymbol{x}_{t-1}] + (1-\alpha_t)\text{Cov}[\boldsymbol{\epsilon}] = \alpha_t\mathbf{I} + (1-\alpha_t)\mathbf{I} = \mathbf{I} \tag{8.24}$$

因此，若某一时刻隐变量的协方差矩阵为单位矩阵，则经过一步扩散后仍保持单位矩阵。由此可见，前向扩散过程在逐步注入噪声的同时，能够保持隐变量的方差尺度不变。此外，前向扩散过程通常假设满足马尔科夫性，即当前时刻的隐变量仅依赖于前一时刻的隐变量，而与更早时刻的隐变量无关。因此，给定真实数据 $\boldsymbol{x}_0$ 后，前向扩散过程的联合条件概率分布可以写为：

$$q(\boldsymbol{x}_{1:T}|\boldsymbol{x}_0) = \prod_{t=1}^{T} q(\boldsymbol{x}_t|\boldsymbol{x}_{t-1}) \tag{8.25}$$

相应地，反向生成过程同样可建模为一个马尔科夫链，其联合分布可表示为：

$$p(\boldsymbol{x}_{0:T}) = p(\boldsymbol{x}_T)\prod_{t=1}^{T} p_\theta(\boldsymbol{x}_{t-1}|\boldsymbol{x}_t) \tag{8.26}$$

其中 $p(\boldsymbol{x}_T) = \mathcal{N}(\boldsymbol{x}_T|\mathbf{0}, \mathbf{I})$，而 $\boldsymbol{\theta}$ 表示解码器的参数。与变分自编码器类似，降噪扩散模型同样

可以通过最大化证据下界（ELBO）来优化解码器的参数：

$$
\begin{aligned}
\log p(\boldsymbol{x}) &= \log \int p(\boldsymbol{x}_{0:T})\, d\boldsymbol{x}_{1:T} \quad （\boldsymbol{x} \text{ 即为 } \boldsymbol{x}_0）\\
&= \log \int \frac{p(\boldsymbol{x}_{0:T}) q(\boldsymbol{x}_{1:T}|\boldsymbol{x}_0)}{q(\boldsymbol{x}_{1:T}|\boldsymbol{x}_0)}\, d\boldsymbol{x}_{1:T} \quad （分子和分母同乘 q(\boldsymbol{x}_{1:T}|\boldsymbol{x}_0)）\\
&= \log \mathbb{E}_{q(\boldsymbol{x}_{1:T}|\boldsymbol{x}_0)} \left[\frac{p(\boldsymbol{x}_{0:T})}{q(\boldsymbol{x}_{1:T}|\boldsymbol{x}_0)} \right] \quad （根据期望的定义）\\
&\geqslant \underbrace{\mathbb{E}_{q(\boldsymbol{x}_{1:T}|\boldsymbol{x}_0)} \left[\log \frac{p(\boldsymbol{x}_{0:T})}{q(\boldsymbol{x}_{1:T}|\boldsymbol{x}_0)} \right]}_{\text{证据下界（ELBO）}} \quad （根据琴生不等式）\\
&= \mathbb{E}_{q(\boldsymbol{x}_{1:T}|\boldsymbol{x}_0)} \left[\log \frac{p(\boldsymbol{x}_T) \prod_{t=1}^{T} p_\theta(\boldsymbol{x}_{t-1}|\boldsymbol{x}_t)}{\prod_{t=1}^{T} q(\boldsymbol{x}_t|\boldsymbol{x}_{t-1})} \right] \quad （代入公式8.25与公式8.26）\\
&= \mathbb{E}_{q(\boldsymbol{x}_{1:T}|\boldsymbol{x}_0)} \left[\log \frac{p(\boldsymbol{x}_T) p_\theta(\boldsymbol{x}_0|\boldsymbol{x}_1)}{q(\boldsymbol{x}_T|\boldsymbol{x}_{T-1})} \right] + \mathbb{E}_{q(\boldsymbol{x}_{1:T}|\boldsymbol{x}_0)} \left[\log \prod_{t=1}^{T-1} \frac{p_\theta(\boldsymbol{x}_t|\boldsymbol{x}_{t+1})}{q(\boldsymbol{x}_t|\boldsymbol{x}_{t-1})} \right]\\
&= \mathbb{E}_{q(\boldsymbol{x}_{1:T}|\boldsymbol{x}_0)} \left[\log \frac{p(\boldsymbol{x}_T) p_\theta(\boldsymbol{x}_0|\boldsymbol{x}_1)}{q(\boldsymbol{x}_T|\boldsymbol{x}_{T-1})} \right] + \sum_{t=1}^{T-1} \mathbb{E}_{q(\boldsymbol{x}_{1:T}|\boldsymbol{x}_0)} \left[\log \frac{p_\theta(\boldsymbol{x}_t|\boldsymbol{x}_{t+1})}{q(\boldsymbol{x}_t|\boldsymbol{x}_{t-1})} \right]\\
&= \mathbb{E}_{q(\boldsymbol{x}_1|\boldsymbol{x}_0)} \left[\log p_\theta(\boldsymbol{x}_0|\boldsymbol{x}_1) \right] + \mathbb{E}_{q(\boldsymbol{x}_{T-1}, \boldsymbol{x}_T|\boldsymbol{x}_0)} \left[\log \frac{p(\boldsymbol{x}_T)}{q(\boldsymbol{x}_T|\boldsymbol{x}_{T-1})} \right]\\
&\quad + \sum_{t=1}^{T-1} \mathbb{E}_{q(\boldsymbol{x}_{t-1}, \boldsymbol{x}_t, \boldsymbol{x}_{t+1}|\boldsymbol{x}_0)} \left[\log \frac{p_\theta(\boldsymbol{x}_t|\boldsymbol{x}_{t+1})}{q(\boldsymbol{x}_t|\boldsymbol{x}_{t-1})} \right] \quad （根据马尔科夫性）\\
&= \underbrace{\mathbb{E}_{q(\boldsymbol{x}_1|\boldsymbol{x}_0)} \left[\log p_\theta(\boldsymbol{x}_0|\boldsymbol{x}_1) \right]}_{\text{重构项}} - \underbrace{\mathbb{E}_{q(\boldsymbol{x}_{T-1}|\boldsymbol{x}_0)} \left[D_{\mathrm{KL}}(q(\boldsymbol{x}_T|\boldsymbol{x}_{T-1}) \| p(\boldsymbol{x}_T)) \right]}_{\text{先验约束项}}\\
&\quad - \sum_{t=1}^{T-1} \underbrace{\mathbb{E}_{q(\boldsymbol{x}_{t-1}, \boldsymbol{x}_{t+1}|\boldsymbol{x}_0)} \left[D_{\mathrm{KL}}(q(\boldsymbol{x}_t|\boldsymbol{x}_{t-1}) \| p_\theta(\boldsymbol{x}_t|\boldsymbol{x}_{t+1})) \right]}_{\text{对齐项}}
\end{aligned}
$$

上式中的三项分别具有明确的意义：

- 重构项利用第一步隐变量 $\boldsymbol{x}_1$ 来重构原始数据 $\boldsymbol{x}_0$。这一项与变分自编码器中的重构项具有相似的作用，其目标是确保最终生成的数据能够准确恢复真实数据分布。
- 先验约束项要求最终时间步的隐变量分布与多元标准高斯先验 $p(\boldsymbol{x}_T)$ 一致。由于前向扩散过程被设计为逐步将数据分布转化为标准高斯分布，因此该项并不包含可学习参数。当扩散步数 T 足够大时，隐变量 $\boldsymbol{x}_T$ 的分布将逐渐逼近多元标准高斯分布。
- 对齐项用于约束反向生成过程与前向扩散过程保持一致。具体而言，对于每一个时间步 t，模型学习得到的反向去噪分布 $p_\theta(\boldsymbol{x}_t|\boldsymbol{x}_{t+1})$ 应尽可能逼近由前向扩散过程所诱导出的加噪分布 $q(\boldsymbol{x}_t|\boldsymbol{x}_{t-1})$。该项通过最小化这两个分布之间的 KL 散度来实现。

上述重构项与对齐项均以期望的形式出现，因此可以采用蒙特卡洛方法进行近似估计。然而，对于对齐项而言，其期望的计算同时涉及随机变量 $\boldsymbol{x}_{t-1}$ 和 $\boldsymbol{x}_{t+1}$ 的采样，其对应的蒙特卡洛估计通常具有较大的方差，进而不利于模型训练的稳定性。接下来，我们将推导出与原目标

等价、但期望形式仅依赖于单个随机变量的优化目标，从而降低蒙特卡洛估计的方差。我们先将前向扩散过程的状态转移分布 $q(\boldsymbol{x}_t|\boldsymbol{x}_{t-1})$ 写成 $q(\boldsymbol{x}_t|\boldsymbol{x}_{t-1},\boldsymbol{x}_0)$ 的形式。该改写是成立的，因为前向扩散过程满足马尔科夫性质，即在给定 $\boldsymbol{x}_{t-1}$ 的条件下，$\boldsymbol{x}_t$ 与更早时刻的状态 $\boldsymbol{x}_0$ 条件独立。因此，在条件概率中添加 $\boldsymbol{x}_0$ 项并不会改变分布本身。利用这一改写形式，并结合贝叶斯公式，可以将每一步编码器的状态转移分布重写为：

$$q(\boldsymbol{x}_t|\boldsymbol{x}_{t-1},\boldsymbol{x}_0)=\frac{q(\boldsymbol{x}_{t-1}|\boldsymbol{x}_t,\boldsymbol{x}_0)q(\boldsymbol{x}_t|\boldsymbol{x}_0)}{q(\boldsymbol{x}_{t-1}|\boldsymbol{x}_0)} \tag{8.27}$$

利用上述形式的前向扩散过程状态转移分布，我们可以重新推导降噪扩散模型的证据下界如下：

$$\begin{aligned}
\log p(\boldsymbol{x}) &\geq \mathbb{E}_{q(\boldsymbol{x}_{1:T}|\boldsymbol{x}_0)}\left[\log\frac{p(\boldsymbol{x}_{0:T})}{q(\boldsymbol{x}_{1:T}|\boldsymbol{x}_0)}\right] \\
&= \mathbb{E}_{q(\boldsymbol{x}_{1:T}|\boldsymbol{x}_0)}\left[\log\frac{p(\boldsymbol{x}_T)\prod_{t=1}^{T}p_\theta(\boldsymbol{x}_{t-1}|\boldsymbol{x}_t)}{\prod_{t=1}^{T}q(\boldsymbol{x}_t|\boldsymbol{x}_{t-1})}\right] \quad \text{（代入公式8.25与公式8.26）} \\
&= \mathbb{E}_{q(\boldsymbol{x}_{1:T}|\boldsymbol{x}_0)}\left[\log\frac{p(\boldsymbol{x}_T)p_\theta(\boldsymbol{x}_0|\boldsymbol{x}_1)\prod_{t=2}^{T}p_\theta(\boldsymbol{x}_{t-1}|\boldsymbol{x}_t)}{q(\boldsymbol{x}_1|\boldsymbol{x}_0)\prod_{t=2}^{T}q(\boldsymbol{x}_t|\boldsymbol{x}_{t-1},\boldsymbol{x}_0)}\right] \quad \text{（采用前向状态转移的改写形式）} \\
&= \mathbb{E}_{q(\boldsymbol{x}_{1:T}|\boldsymbol{x}_0)}\left[\log\frac{p(\boldsymbol{x}_T)p_\theta(\boldsymbol{x}_0|\boldsymbol{x}_1)}{q(\boldsymbol{x}_1|\boldsymbol{x}_0)}+\log\prod_{t=2}^{T}\frac{p_\theta(\boldsymbol{x}_{t-1}|\boldsymbol{x}_t)}{q(\boldsymbol{x}_t|\boldsymbol{x}_{t-1},\boldsymbol{x}_0)}\right] \\
&= \mathbb{E}_{q(\boldsymbol{x}_{1:T}|\boldsymbol{x}_0)}\left[\log\frac{p(\boldsymbol{x}_T)p_\theta(\boldsymbol{x}_0|\boldsymbol{x}_1)}{q(\boldsymbol{x}_1|\boldsymbol{x}_0)}+\log\prod_{t=2}^{T}\frac{p_\theta(\boldsymbol{x}_{t-1}|\boldsymbol{x}_t)}{\frac{q(\boldsymbol{x}_{t-1}|\boldsymbol{x}_t,\boldsymbol{x}_0)q(\boldsymbol{x}_t|\boldsymbol{x}_0)}{q(\boldsymbol{x}_{t-1}|\boldsymbol{x}_0)}}\right] \quad \text{（代入公式8.27）} \\
&= \mathbb{E}_{q(\boldsymbol{x}_{1:T}|\boldsymbol{x}_0)}\left[\log\frac{p(\boldsymbol{x}_T)p_\theta(\boldsymbol{x}_0|\boldsymbol{x}_1)}{q(\boldsymbol{x}_1|\boldsymbol{x}_0)}+\log\frac{q(\boldsymbol{x}_1|\boldsymbol{x}_0)}{q(\boldsymbol{x}_T|\boldsymbol{x}_0)}+\log\prod_{t=2}^{T}\frac{p_\theta(\boldsymbol{x}_{t-1}|\boldsymbol{x}_t)}{q(\boldsymbol{x}_{t-1}|\boldsymbol{x}_t,\boldsymbol{x}_0)}\right] \\
&= \mathbb{E}_{q(\boldsymbol{x}_{1:T}|\boldsymbol{x}_0)}\left[\log\frac{p(\boldsymbol{x}_T)p_\theta(\boldsymbol{x}_0|\boldsymbol{x}_1)}{q(\boldsymbol{x}_T|\boldsymbol{x}_0)}+\sum_{t=2}^{T}\log\frac{p_\theta(\boldsymbol{x}_{t-1}|\boldsymbol{x}_t)}{q(\boldsymbol{x}_{t-1}|\boldsymbol{x}_t,\boldsymbol{x}_0)}\right] \quad \text{（连乘相同分子分母相互抵消）} \\
&= \mathbb{E}_{q(\boldsymbol{x}_1|\boldsymbol{x}_0)}\left[\log p_\theta(\boldsymbol{x}_0|\boldsymbol{x}_1)\right]+\mathbb{E}_{q(\boldsymbol{x}_T|\boldsymbol{x}_0)}\left[\log\frac{p(\boldsymbol{x}_T)}{q(\boldsymbol{x}_T|\boldsymbol{x}_0)}\right]+\sum_{t=2}^{T}\mathbb{E}_{q(\boldsymbol{x}_t,\boldsymbol{x}_{t-1}|\boldsymbol{x}_0)}\left[\log\frac{p_\theta(\boldsymbol{x}_{t-1}|\boldsymbol{x}_t)}{q(\boldsymbol{x}_{t-1}|\boldsymbol{x}_t,\boldsymbol{x}_0)}\right] \\
&= \underbrace{\mathbb{E}_{q(\boldsymbol{x}_1|\boldsymbol{x}_0)}\left[\log p_\theta(\boldsymbol{x}_0|\boldsymbol{x}_1)\right]}_{\text{重构项}}-\underbrace{\mathbb{E}_{q(\boldsymbol{x}_T|\boldsymbol{x}_0)}\left[D_{\mathrm{KL}}(q(\boldsymbol{x}_T|\boldsymbol{x}_0)\|p(\boldsymbol{x}_T))\right]}_{\text{先验约束项}} \\
&\quad -\sum_{t=2}^{T}\underbrace{\mathbb{E}_{q(\boldsymbol{x}_t|\boldsymbol{x}_0)}\left[D_{\mathrm{KL}}(q(\boldsymbol{x}_{t-1}|\boldsymbol{x}_t,\boldsymbol{x}_0)\|p_\theta(\boldsymbol{x}_{t-1}|\boldsymbol{x}_t))\right]}_{\text{降噪匹配项}}
\end{aligned}$$

上式中的降噪匹配项的推导过程如下：

$$\begin{aligned}
&\mathbb{E}_{q(\boldsymbol{x}_t,\boldsymbol{x}_{t-1}|\boldsymbol{x}_0)}\left[\log\frac{p_\theta(\boldsymbol{x}_{t-1}|\boldsymbol{x}_t)}{q(\boldsymbol{x}_{t-1}|\boldsymbol{x}_t,\boldsymbol{x}_0)}\right] \\
=&\mathbb{E}_{q(\boldsymbol{x}_t|\boldsymbol{x}_0)}\left[\mathbb{E}_{q(\boldsymbol{x}_{t-1}|\boldsymbol{x}_t,\boldsymbol{x}_0)}\left[\log\frac{p_\theta(\boldsymbol{x}_{t-1}|\boldsymbol{x}_t)}{q(\boldsymbol{x}_{t-1}|\boldsymbol{x}_t,\boldsymbol{x}_0)}\right]\right] \quad (q(\boldsymbol{x}_t,\boldsymbol{x}_{t-1}|\boldsymbol{x}_0)=q(\boldsymbol{x}_t|\boldsymbol{x}_0)q(\boldsymbol{x}_{t-1}|\boldsymbol{x}_t,\boldsymbol{x}_0)) \\
=&\mathbb{E}_{q(\boldsymbol{x}_t|\boldsymbol{x}_0)}\left[-D_{\mathrm{KL}}(q(\boldsymbol{x}_{t-1}|\boldsymbol{x}_t,\boldsymbol{x}_0)\|p_\theta(\boldsymbol{x}_{t-1}|\boldsymbol{x}_t))\right] \quad \text{（根据 KL 散度的定义）}
\end{aligned}$$

降噪匹配项的优化目标是通过调整参数 $\boldsymbol{\theta}$ 使解码器分布 $p_\theta(\boldsymbol{x}_{t-1}|\boldsymbol{x}_t)$（通过深度网络来实现）

尽可能逼近真实后验分布 $q(\boldsymbol{x}_{t-1}|\boldsymbol{x}_t, \boldsymbol{x}_0)$，即最小化两者之间的 KL 散度。值得注意的是，该项的期望仅依赖于随机变量 $\boldsymbol{x}_t$。接下来需要解决的问题是如何计算降噪匹配项中的 KL 散度。为此，我们首先来推导真实后验分布 $q(\boldsymbol{x}_{t-1}|\boldsymbol{x}_t, \boldsymbol{x}_0)$ 的解析表达式。根据贝叶斯公式，我们可以将其改写为：

$$q(\boldsymbol{x}_{t-1}|\boldsymbol{x}_t, \boldsymbol{x}_0) = \frac{q(\boldsymbol{x}_t|\boldsymbol{x}_{t-1}, \boldsymbol{x}_0)q(\boldsymbol{x}_{t-1}|\boldsymbol{x}_0)}{q(\boldsymbol{x}_t|\boldsymbol{x}_0)} \tag{8.28}$$

根据公式8.21，前向扩散过程可被预定义为：

$$q(\boldsymbol{x}_t|\boldsymbol{x}_{t-1}, \boldsymbol{x}_0) = q(\boldsymbol{x}_t|\boldsymbol{x}_{t-1}) = \mathcal{N}(\boldsymbol{x}_t|\sqrt{\alpha_t}\boldsymbol{x}_{t-1}, (1-\alpha_t)\mathbf{I}) \tag{8.29}$$

我们还需要推导出公式8.28中 $q(\boldsymbol{x}_{t-1}|\boldsymbol{x}_0)$ 和 $q(\boldsymbol{x}_t|\boldsymbol{x}_0)$ 的解析表达式。借助在变分自编码器中介绍过的重参数化技巧，从分布 $q(\boldsymbol{x}_t|\boldsymbol{x}_{t-1})$ 中采样得到的随机变量 $\boldsymbol{x}_t$ 可以表示为：

$$\boldsymbol{x}_t = \sqrt{\alpha_t}\boldsymbol{x}_{t-1} + \sqrt{1-\alpha_t}\boldsymbol{\epsilon}, \quad \boldsymbol{\epsilon} \sim \mathcal{N}(\boldsymbol{\epsilon}|\mathbf{0}, \mathbf{I}) \tag{8.30}$$

同理，随机变量 $\boldsymbol{x}_{t-1}$ 也可以表示为：

$$\boldsymbol{x}_{t-1} = \sqrt{\alpha_{t-1}}\boldsymbol{x}_{t-2} + \sqrt{1-\alpha_{t-1}}\boldsymbol{\epsilon}, \quad \boldsymbol{\epsilon} \sim \mathcal{N}(\boldsymbol{\epsilon}|\mathbf{0}, \mathbf{I}) \tag{8.31}$$

因此，通过迭代应用重参数化技巧，可以推导出分布 $q(\boldsymbol{x}_t|\boldsymbol{x}_0)$ 的解析形式：

$$\begin{aligned}
\boldsymbol{x}_t &= \sqrt{\alpha_t}\boldsymbol{x}_{t-1} + \sqrt{1-\alpha_t}\boldsymbol{\epsilon} \\
&= \sqrt{\alpha_t}\left(\sqrt{\alpha_{t-1}}\boldsymbol{x}_{t-2} + \sqrt{1-\alpha_{t-1}}\boldsymbol{\epsilon}\right) + \sqrt{1-\alpha_t}\boldsymbol{\epsilon} \\
&= \sqrt{\alpha_t\alpha_{t-1}}\boldsymbol{x}_{t-2} + \sqrt{\sqrt{\alpha_t - \alpha_t\alpha_{t-1}}^2 + \sqrt{1-\alpha_t}^2}\boldsymbol{\epsilon} \quad \text{（两个相互独立的高斯随机变量之和仍然服从高斯分布，其均值为两者均值之和，方差为两者方差之和）} \\
&= \sqrt{\alpha_t\alpha_{t-1}}\boldsymbol{x}_{t-2} + \sqrt{1-\alpha_t\alpha_{t-1}}\boldsymbol{\epsilon} \\
&= \dots \\
&= \sqrt{\prod_{i=1}^{t}\alpha_i}\boldsymbol{x}_0 + \sqrt{1-\prod_{i=1}^{t}\alpha_i}\boldsymbol{\epsilon} \\
&= \sqrt{\bar{\alpha}_t}\boldsymbol{x}_0 + \sqrt{1-\bar{\alpha}_t}\boldsymbol{\epsilon} \quad \left(\text{记 } \bar{\alpha}_t = \prod_{i=1}^{t}\alpha_i\right) \qquad (8.32) \\
&\sim \mathcal{N}(\boldsymbol{x}_t|\sqrt{\bar{\alpha}_t}\boldsymbol{x}_0, (1-\bar{\alpha}_t)\mathbf{I}) \qquad (8.33)
\end{aligned}$$

由上式可知，条件分布 $q(\boldsymbol{x}_t|\boldsymbol{x}_0)$ 服从均值为 $\sqrt{\bar{\alpha}_t}\boldsymbol{x}_0$、协方差矩阵为 $(1-\bar{\alpha}_t)\mathbf{I}$ 的多元高斯分布。

据此，可以进一步推导出真实后验分布 $q(\boldsymbol{x}_{t-1}|\boldsymbol{x}_t,\boldsymbol{x}_0)$ 的解析形式：

$$
\begin{aligned}
q(\boldsymbol{x}_{t-1}|\boldsymbol{x}_t,\boldsymbol{x}_0) &= \frac{q(\boldsymbol{x}_t|\boldsymbol{x}_{t-1},\boldsymbol{x}_0)q(\boldsymbol{x}_{t-1}|\boldsymbol{x}_0)}{q(\boldsymbol{x}_t|\boldsymbol{x}_0)} \\
&= \frac{\mathcal{N}(\boldsymbol{x}_t|\sqrt{\alpha_t}\boldsymbol{x}_{t-1},(1-\alpha_t)\mathbf{I})\,\mathcal{N}(\boldsymbol{x}_{t-1}|\sqrt{\bar{\alpha}_{t-1}}\boldsymbol{x}_0,(1-\bar{\alpha}_{t-1})\mathbf{I})}{\mathcal{N}(\boldsymbol{x}_t|\sqrt{\bar{\alpha}_t}\boldsymbol{x}_0,(1-\bar{\alpha}_t)\mathbf{I})} \quad \text{（代入相应的高斯分布）} \\
&\propto \exp\left\{-\left[\frac{(\boldsymbol{x}_t-\sqrt{\alpha_t}\boldsymbol{x}_{t-1})^2}{2(1-\alpha_t)}+\frac{(\boldsymbol{x}_{t-1}-\sqrt{\bar{\alpha}_{t-1}}\boldsymbol{x}_0)^2}{2(1-\bar{\alpha}_{t-1})}-\frac{(\boldsymbol{x}_t-\sqrt{\bar{\alpha}_t}\boldsymbol{x}_0)^2}{2(1-\bar{\alpha}_t)}\right]\right\} \\
&\propto \exp\left\{-\frac{1}{2}\left(\frac{1}{\frac{(1-\alpha_t)(1-\bar{\alpha}_{t-1})}{1-\bar{\alpha}_t}}\right)\left[\boldsymbol{x}_{t-1}^2-2\frac{\sqrt{\alpha_t}(1-\bar{\alpha}_{t-1})\boldsymbol{x}_t+\sqrt{\bar{\alpha}_{t-1}}(1-\alpha_t)\boldsymbol{x}_0}{1-\bar{\alpha}_t}\boldsymbol{x}_{t-1}\right]\right\} \\
&\propto \mathcal{N}\left(\boldsymbol{x}_{t-1}\middle|\underbrace{\frac{\sqrt{\alpha_t}(1-\bar{\alpha}_{t-1})\boldsymbol{x}_t+\sqrt{\bar{\alpha}_{t-1}}(1-\alpha_t)\boldsymbol{x}_0}{1-\bar{\alpha}_t}}_{\boldsymbol{\mu}(\boldsymbol{x}_t,\boldsymbol{x}_0)},\underbrace{\frac{(1-\alpha_t)(1-\bar{\alpha}_{t-1})}{1-\bar{\alpha}_t}\mathbf{I}}_{\boldsymbol{\Sigma}(t)}\right)
\end{aligned}
$$

由上式可知，真实后验分布 $q(\boldsymbol{x}_{t-1}|\boldsymbol{x}_t,\boldsymbol{x}_0)$ 服从均值为 $\boldsymbol{\mu}(\boldsymbol{x}_t,\boldsymbol{x}_0)$、协方差矩阵为 $\boldsymbol{\Sigma}(t)$ 的多元高斯分布。注意到该分布的协方差矩阵仅由预先设定的超参数 α_t 决定，因此无需通过数据进行学习。为了简化模型的学习过程，由深度神经网络近似的反向解码器分布 $p_\theta(\boldsymbol{x}_{t-1}|\boldsymbol{x}_t)$ 通常也假设服从多元高斯分布，并采用与真实后验分布 $q(\boldsymbol{x}_{t-1}|\boldsymbol{x}_t,\boldsymbol{x}_0)$ 相同的协方差矩阵，而仅利用深度神经网络对均值进行建模：

$$
\boldsymbol{\mu}_\theta(\boldsymbol{x}_t,t)=\frac{\sqrt{\alpha_t}(1-\bar{\alpha}_{t-1})\boldsymbol{x}_t+\sqrt{\bar{\alpha}_{t-1}}(1-\alpha_t)\boldsymbol{x}_\theta(\boldsymbol{x}_t,t)}{1-\bar{\alpha}_t} \tag{8.34}
$$

其中 $\boldsymbol{x}_\theta(\boldsymbol{x}_t,t)$ 由深度神经网络参数化并进行预测。将上式与真实后验分布的形式对比可以看出，该网络的学习目标是在给定噪声图像 $\boldsymbol{x}_t$ 及时间步 t 的条件下，对原始图像 $\boldsymbol{x}_0$ 进行预测。因此，其训练过程可等价地表述为最小化如下均方重构损失：

$$
\mathcal{L}=\|\boldsymbol{x}_\theta(\boldsymbol{x}_t,t)-\boldsymbol{x}_0\|^2 \tag{8.35}
$$

通过最小化上述均方重构损失可以减小真实后验分布 $q(\boldsymbol{x}_{t-1}|\boldsymbol{x}_t,\boldsymbol{x}_0)$ 与解码器分布 $p_\theta(\boldsymbol{x}_{t-1}|\boldsymbol{x}_t)$ 之间的 KL 散度，从而等价于最小化降噪匹配项。然而，不同时间步对应的隐变量 $\boldsymbol{x}_t$ 在分布上存在显著差异：当 t 接近于 0 时，$\boldsymbol{x}_t$ 与原始图像 $\boldsymbol{x}_0$ 较为接近，因此对 $\boldsymbol{x}_0$ 的预测相对容易；而当 t 接近于 T 时，则 $\boldsymbol{x}_t$ 接近于纯高斯噪声，此时从中恢复 $\boldsymbol{x}_0$ 的难度显著增加。因此，直接使用公式8.35来优化网络参数 $\boldsymbol{\theta}$ 会导致不同时间步上的重构误差在尺度上不一致，从而可能引发训练过程的不稳定性。

为了获得在不同时间步上误差尺度更为一致的等价学习目标，我们先利用重参数化技巧得到的以下等式（公式8.32）：

$$
\boldsymbol{x}_t=\sqrt{\bar{\alpha}_t}\boldsymbol{x}_0+\sqrt{1-\bar{\alpha}_t}\boldsymbol{\epsilon} \tag{8.36}
$$

将原始图像 $\boldsymbol{x}_0$ 表示为：

$$
\boldsymbol{x}_0=\frac{\boldsymbol{x}_t-\sqrt{1-\bar{\alpha}_t}\boldsymbol{\epsilon}}{\sqrt{\bar{\alpha}_t}} \tag{8.37}
$$

利用以上表示，我们可以将真实后验分布 $q(\boldsymbol{x}_{t-1}|\boldsymbol{x}_t,\boldsymbol{x}_0)$ 的均值 $\boldsymbol{\mu}(\boldsymbol{x}_t,\boldsymbol{x}_0)$ 重写为仅依赖于 $\boldsymbol{x}_t$ 与

噪声 $\boldsymbol{\epsilon}$ 的形式，进而使得原本通过参数化深度神经网络预测原始图像 $\boldsymbol{x}_0$ 来最小化真实后验分布 $q(\boldsymbol{x}_{t-1}|\boldsymbol{x}_t,\boldsymbol{x}_0)$ 与解码器分布 $p_\theta(\boldsymbol{x}_{t-1}|\boldsymbol{x}_t)$ 之间 KL 散度的学习目标等价地转化为对噪声 $\boldsymbol{\epsilon}$ 的预测。

将公式8.37代入真实后验分布均值 $\boldsymbol{\mu}(\boldsymbol{x}_t,\boldsymbol{x}_0)$ 的表达式中，可以得到如下推导：

$$\begin{aligned}\boldsymbol{\mu}(\boldsymbol{x}_t,\boldsymbol{x}_0) &= \frac{\sqrt{\alpha_t}(1-\bar{\alpha}_{t-1})\boldsymbol{x}_t + \sqrt{\bar{\alpha}_{t-1}}(1-\alpha_t)\boldsymbol{x}_0}{1-\bar{\alpha}_t} \\ &= \frac{\sqrt{\alpha_t}(1-\bar{\alpha}_{t-1})\boldsymbol{x}_t + \sqrt{\bar{\alpha}_{t-1}}(1-\alpha_t)\frac{\boldsymbol{x}_t-\sqrt{1-\bar{\alpha}_t}\boldsymbol{\epsilon}}{\sqrt{\bar{\alpha}_t}}}{1-\bar{\alpha}_t} \\ &= \frac{1}{\sqrt{\alpha_t}}\boldsymbol{x}_t - \frac{1-\alpha_t}{\sqrt{1-\bar{\alpha}_t}\sqrt{\alpha_t}}\boldsymbol{\epsilon}\end{aligned} \tag{8.38}$$

同样地，如公式8.34所示的解码器分布均值也可以表示为关于 $\boldsymbol{x}_t$ 和噪声 $\boldsymbol{\epsilon}$ 的形式：

$$\boldsymbol{\mu}_\theta(\boldsymbol{x}_t,t) = \frac{1}{\sqrt{\alpha_t}}\boldsymbol{x}_t - \frac{1-\alpha_t}{\sqrt{1-\bar{\alpha}_t}\sqrt{\alpha_t}}\boldsymbol{\epsilon}_\theta(\boldsymbol{x}_t,t) \tag{8.39}$$

其中 $\boldsymbol{\epsilon}_\theta(\boldsymbol{x}_t,t)$ 为由深度神经网络预测的噪声估计值。对比公式8.38和公式8.39可以发现，降噪扩散模型的学习目标可以由原始图像重构等价地转化为噪声预测：

$$\mathcal{L} = \|\boldsymbol{\epsilon} - \boldsymbol{\epsilon}_\theta(\boldsymbol{x}_t,t)\|^2 \tag{8.40}$$

其中噪声 $\boldsymbol{\epsilon}$ 服从多元标准高斯分布 $\mathcal{N}(\boldsymbol{\epsilon}|\mathbf{0},\mathbf{I})$，而带噪样本 $\boldsymbol{x}_t$ 则由公式8.36计算生成。与直接预测原始图像 $\boldsymbol{x}_0$ 相比，噪声 $\boldsymbol{\epsilon}$ 在不同时间步上具有更加稳定且一致的统计特性，因此该目标能够在不同时间步之间保持更为均衡的误差尺度，从而提高模型训练的稳定性。降噪扩散模型的训练过程与生成过程分别如算法8.2和算法8.3所示。

算法 8.2: 降噪扩散模型训练算法

输入: 真实数据集 $\mathcal{D}=\{\boldsymbol{x}_i\}_{i=1}^N$ 以及最大迭代次数 H。

初始化: 随机初始化解码器参数 $\boldsymbol{\theta}$，并设当前迭代次数 $h=1$;

while $h \le H$ **do**

  从数据集 $\mathcal{D}$ 中随机采样样本 $\boldsymbol{x}_0$;

  从均匀分布中随机采样时间步 t;

  从高斯分布 $\mathcal{N}(\boldsymbol{\epsilon}|\mathbf{0},\mathbf{I})$ 中采样噪声 $\boldsymbol{\epsilon}$;

  按以下公式计算目标函数的梯度，并利用梯度下降法更新解码器参数：$\nabla_\theta\|\boldsymbol{\epsilon}-\boldsymbol{\epsilon}_\theta(\sqrt{\bar{\alpha}_t}\boldsymbol{x}_0+\sqrt{1-\bar{\alpha}_t}\boldsymbol{\epsilon},t)\|^2$;

  $h \leftarrow h+1$;

输出: 训练完成后得到的解码器参数 $\boldsymbol{\theta}$。

算法 8.3: 降噪扩散模型生成算法

输入: 深度神经网络解码器的参数 $\boldsymbol{\theta}$ 以及生成总步数 T。

初始化: 从多元标准高斯分布 $\mathcal{N}(\boldsymbol{x}_T|\mathbf{0},\mathbf{I})$ 中随机采样初始噪声 $\boldsymbol{x}_T$;

for $t \leftarrow T$ **to** 1 **do**

  $\boldsymbol{\epsilon} \leftarrow \mathbf{0}$;

  if $(t>1)$ **then**

    从多元标准高斯分布 $\mathcal{N}(\boldsymbol{\epsilon}|\mathbf{0},\mathbf{I})$ 中采样随机噪声 $\boldsymbol{\epsilon}$;

  $\boldsymbol{x}_{t-1} = \frac{1}{\sqrt{\alpha_t}}\boldsymbol{x}_t - \frac{1-\alpha_t}{\sqrt{1-\bar{\alpha}_t}\sqrt{\alpha_t}}\boldsymbol{\epsilon}_\theta(\boldsymbol{x}_t,t) + \sqrt{\boldsymbol{\Sigma}(t)}\boldsymbol{\epsilon}$;

输出: 生成的图像 $\boldsymbol{x}_0$。

在降噪扩散模型的生成算法8.3中，第 6 步关于 $\boldsymbol{x}_{t-1}$ 的均值依据公式8.39计算，其协方差矩阵 $\boldsymbol{\Sigma}(t)$ 则取自真实后验分布 $q(\boldsymbol{x}_{t-1}|\boldsymbol{x}_t,\boldsymbol{x}_0)$ 的解析形式，其表达式为：

$$\boldsymbol{\Sigma}(t) = \frac{(1-\alpha_t)(1-\bar{\alpha}_{t-1})}{1-\bar{\alpha}_t}\mathbf{I} \tag{8.41}$$

在生成算法的第 6 步中，不仅需要在上一步隐变量 $\boldsymbol{x}_t$ 的基础上减去由模型预测的修正噪声，还

需要额外叠加一项 $\sqrt{\boldsymbol{\Sigma}}(t)\boldsymbol{\epsilon}$（其中 $\boldsymbol{\epsilon}$ 服从多元标准高斯分布）。这一设计的原因与变分自编码器中的采样机制类似，即隐变量 $\boldsymbol{x}_{t-1}$ 的后验分布并非一个确定的点，而是一个条件高斯分布。因此，在生成过程中需要通过随机采样来反映该分布的不确定性。如果去除该随机噪声项，则除初始噪声 $\boldsymbol{x}_T$ 外，整个反向生成过程将退化为确定性过程，从而在一定程度上降低了生成结果的多样性。此外，在每个生成时间步中引入适当的随机扰动，能够为采样轨迹提供一种“容错与校正”机制，使得多步生成过程中可能产生的累积误差有机会得以修正，从而提升生成样本的整体质量与稳定性。

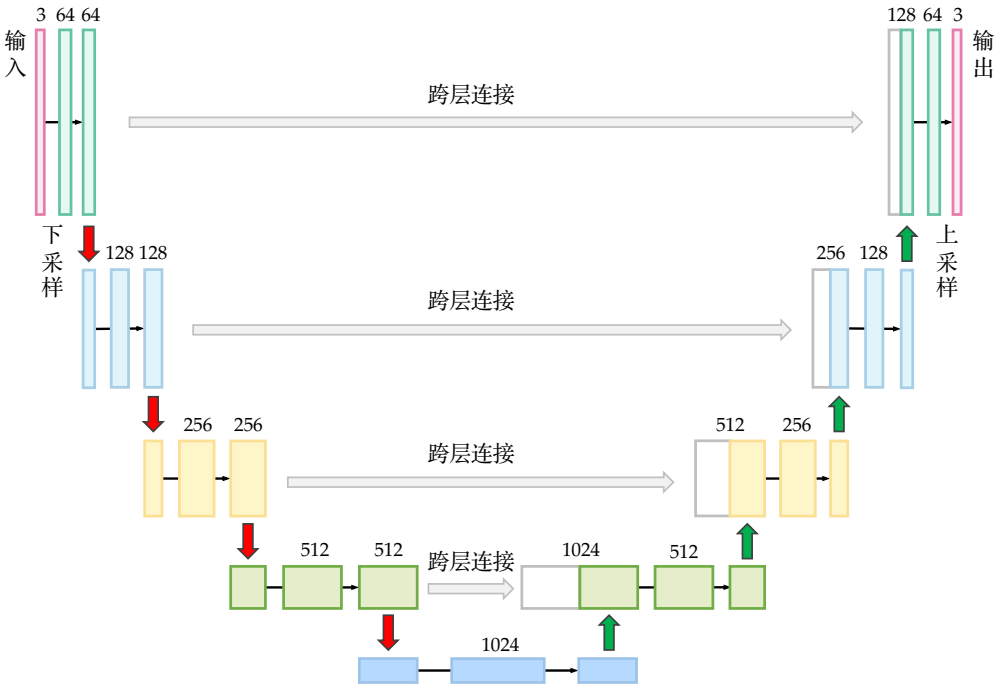


图 8.11: U-Net 网络架构示意图。每个长方形表示一个多通道特征图（顶部的数字表示通道数）。长方形的高度表示特征图的空间分辨率，宽度示意特征图的通道数量。跨层连接将下采样过程中所产生的特征图复制并拼接到上采样过程中对应的层级，用以特征融合与复用。

降噪扩散模型中作为解码器的深度神经网络，无论用于预测原始图像还是噪声，其输入与输出一般具有相同的空间分辨率。以图像生成为例，网络的输入与输出通常为具有相同空间尺寸（高度与宽度）和通道数（通常为三通道）的张量。U-Net[64] 是降噪扩散模型及许多生成模型中常用的主干网络结构之一。它首先通过卷积神经网络对输入进行逐步下采样，获得低分辨率的特征表示；随后再通过上采样（如转置卷积）逐步恢复至原始分辨率。由于其下采样与上采样路径在结构上呈现高度对称性，整体形状类似英文字母“U”，因此而得名（如图8.11所示）。U-Net 的另一重要特点是引入跨层连接（Skip Connections）机制。在上采样过程中，会将当前输入的特征图与下采样对应层级的特征图在通道维度上进行拼接，从而实现特征的高效融合与复用。另一种在扩散模型等生成任务中被广泛使用的结构是视觉 Transformer（Vision Transformer, ViT）[22]，ViT 将原本用于自然语言处理的 Transformer 架构扩展到图像处理任务，其核心思想是将图像划分为若干固定大小的图像块（Patch），并将每个图像块展平后通过线性映射转换为

向量。随后，在这些特征向量中叠加反映空间位置信息的位置编码，并将得到的向量序列并行输入 Transformer，以利用自注意力机制建模图像中的全局依赖关系。在生成过程中，模型也以并行方式输出与所有图像块对应的预测结果（如噪声或像素估计值），而非以逐块自回归的方式生成图像内容。在条件生成的情况下，模型在训练与生成阶段需将条件信息（如文本特征向量）注入到 U-Net 或 ViT 等网络的特征提取与融合过程中，从而实现可控生成。

8.5 流匹配模型

前几节的讨论表明，所谓生成，本质上是从真实数据分布 $p(\boldsymbol{x})$ 中采样的过程（Generation as Sampling）。生成模型则旨在学习一种映射机制，通过单步或多步变换，将来自简单已知分布（通常为多元标准高斯分布）的随机样本转换为服从或近似服从真实数据分布 $p(\boldsymbol{x})$ 的样本。

降噪扩散模型将上述多步变换定义为前向加噪的逆向过程。由于前向扩散过程是通过连续注入高斯噪声来实现的，其在概率空间中的演化轨迹在几何上呈现出非线性的弯曲（详见后文分析）。这意味着在逆向生成时，若离散化步长过大，采样轨迹极易偏离理论轨道。因此，为了保证生成质量，逆向过程通常需要采用较小的步长进行多步迭代（往往需要数百乃至数千步）。这使得扩散模型在获得高质量样本的同时，也面临较高的计算复杂度和较低的生成效率。本节我们将讨论生成效率更高的流匹配模型（Flow Matching）。

在对上述现象进行理论分析之前，我们先将这种基于多步变换的生成过程统一建模为**常微分方程**（Ordinary Differential Equations）或**随机微分方程**（Stochastic Differential Equations）的数值求解过程 [72]。在讨论该统一建模框架之前，有必要先对后续使用的数学符号及其含义进行说明。在介绍扩散模型时，我们使用带时间下标的向量 $\boldsymbol{x}_t$ 表示离散时间步 t 处的随机变量或其具体的采样样本。这种表示方法在机器学习文献中十分常见，但从严格的概率论角度来看，它属于一种符号上的简化。在经典的概率论中，通常用大写字母 X 表示随机变量本身，而用小写字母 $\boldsymbol{x}$ 表示该随机变量的具体取值或观测样本。因此，在扩散模型中常见的条件概率转移 $p(\boldsymbol{x}_{t-1}|\boldsymbol{x}_t)$，其严格的数学表达式应当为 $p_{X_{t-1}|X_t}(\boldsymbol{x}_{t-1}|\boldsymbol{x}_t)$。然而，当我们将扩散模型从离散时间的马尔可夫链推广到连续时间动力学框架时，需要严格区分随机变量及其具体实现值。在本节后续内容中，我们用大写字母（如 X_t）表示随机变量或随机过程，而用小写字母（如 $\boldsymbol{x}_t$）表示其具体取值或采样样本。这样的符号约定与概率论、随机过程以及微分方程等领域的经典文献保持一致，也有助于对常微分方程和随机微分方程进行统一描述。

我们先从常微分方程入手，常微分方程的解可以表示为一条轨迹（Trajectory），即关于时间 t 的函数（此处将 X_t 视为状态变量）：

$$X : [0,1] \mapsto \mathbb{R}^D, \quad t \mapsto X_t \tag{8.42}$$

它将时间 $t \in [0,1]$ 映射到 D 维空间中的某个位置 $X_t \in \mathbb{R}^D$。每个常微分方程都由一个向量场（Vector Field）所定义，即：

$$u : \mathbb{R}^D \times [0,1] \to \mathbb{R}^D, \quad (\boldsymbol{x}, t) \mapsto u_t(\boldsymbol{x}) \tag{8.43}$$

也就是说，对于任意时刻 t 和空间中的位置 $\boldsymbol{x}$，向量场都会给出一个速度向量 $u_t(\boldsymbol{x}) \in \mathbb{R}^D$，用于描述该位置在该时刻的运动速度与方向。常微分方程还对轨迹施加了一个约束：即希望找到一条从初始点 $\boldsymbol{x}_0$ 出发，并始终按照向量场 u_t 所指定方向演化的轨迹 X。这种轨迹可以形式化地定义为以下微分方程的解：

$$\frac{\mathrm{d}}{\mathrm{d}t}X_t = u_t(X_t) \qquad \text{（常微分方程）} \tag{8.44}$$

$$X_0 = \boldsymbol{x}_0 \qquad \text{（初始条件）} \tag{8.45}$$

公式8.44要求轨迹在时刻 t 的变化率由向量场 $u_t(\boldsymbol{x})$ 所决定，即轨迹的切线方向始终与向量场保持一致。公式8.45则规定轨迹在初始时刻 $t = 0$ 从点 $\boldsymbol{x}_0$ 出发。由此，我们可以提出一个核心问题：如果轨迹在 $t = 0$ 时位于 $X_0 = \boldsymbol{x}_0$，那么在任意时刻 t，它将到达什么位置？即如何求得 X_t。这一问题可以通过**流**（Flow）函数来回答。流函数描述了向量场所诱导的运动过程，它将初始状态映射为任意指定时刻的状态：

$$\psi : \mathbb{R}^D \times [0,1] \mapsto \mathbb{R}^D,\ (\boldsymbol{x}_0, t) \mapsto \psi_t(\boldsymbol{x}_0) \tag{8.46}$$

$$\frac{\mathrm{d}}{\mathrm{d}t}\psi_t(\boldsymbol{x}_0) = u_t(\psi_t(\boldsymbol{x}_0)) \tag{8.47}$$

$$\psi_0(\boldsymbol{x}_0) = \boldsymbol{x}_0 \tag{8.48}$$

对于给定的初始条件 $X_0 = \boldsymbol{x}_0$，常微分方程的一条解轨迹可表示为 $X_t = \psi_t(X_0)$。从直观上看，向量场、常微分方程与流可以视为同一动力学系统的三种不同描述：向量场定义了常微分方程，而其解即为流函数。对于图像生成任务，解轨迹对应于从初始噪声到目标图像的生成过程，而最终生成的图像可表示为 $\psi_1(\boldsymbol{x}_0)$，其中初始状态 $\boldsymbol{x}_0$ 通常是从简单分布（如多元标准高斯分布）中采样得到。需要注意的是，此处的时序定义与传统的降噪扩散模型正好相反，我们用 $\boldsymbol{x}_0$ 表示初始状态（即噪声），用 $\boldsymbol{x}_1$ 表示最终生成的图像，其中 $t \in [0,1]$ 为连续时间变量。

通常情况下，当向量场 $u_t(X_t)$ 不是简单的线性形式时，流映射 ψ_t 往往不存在解析解。因此需要借助数值方法对常微分方程进行近似求解。常微分方程的数值解法是一个经典且研究比较充分的课题，已经发展出大量成熟的算法 [40]。其中最简单且直观的方法之一是**欧拉方法**（Euler Method）。在欧拉方法中，我们从初始状态 $X_0 = \boldsymbol{x}_0$ 出发，将时间区间 $[0,1]$ 等间隔离散为 $T \in \mathbb{N}$ 个步长，每一步的步长为 $h = \frac{1}{T}$。然后通过如下递推公式近似求解轨迹：

$$X_{t+h} = X_t + h u_t(X_t) \tag{8.49}$$

将上式与降噪扩散模型生成算法8.3中的迭代更新公式进行对比：

$$\boldsymbol{x}_{t-1} = \frac{1}{\sqrt{\alpha_t}}\boldsymbol{x}_t - \underbrace{\frac{1-\alpha_t}{\sqrt{1-\bar{\alpha}_t}\sqrt{\alpha_t}}\boldsymbol{\epsilon}_\theta(\boldsymbol{x}_t, t)}_{\text{确定项}} + \underbrace{\sqrt{\boldsymbol{\Sigma}(t)}\boldsymbol{\epsilon}}_{\text{随机项}} \tag{8.50}$$

我们可以看出，降噪扩散模型的多步迭代生成在形式上同样包含一个由神经网络预测的确定项（对应于常微分方程中的向量场驱动）。但在此基础上，它还额外引入了一个由标准高斯噪声 $\boldsymbol{\epsilon}$ 决定的随机扰动项。因此，与仅包含确定性演化的常微分方程数值求解过程不同，扩散模型的迭代生成本质上对应于随机微分方程的数值求解形式。

随机微分方程是在常微分方程所描述的确定性动态系统基础上引入随机扰动而得到的数学模型，因此能够刻画随机过程随时间的演化规律。我们可以将随机微分方程表示为：

$$\mathrm{d}X_t = u_t(X_t)\mathrm{d}t + \sigma_t \mathrm{d}W_t \qquad \text{（随机微分方程）} \tag{8.51}$$

$$X_0 = \boldsymbol{x}_0 \qquad \text{（初始条件）} \tag{8.52}$$

其中 $u_t(X_t)$ 被称为漂移项（Drift Term），用于描述系统的确定性演化趋势，$\sigma_t \geq 0$ 称为扩散系数（Diffusion Coefficient），用于控制随机扰动的强度，而 $W = \{W_t\}_{0\leq t\leq 1}$ 表示**布朗运动**（Brownian Motion），又称**维纳过程**（Wiener Process），是一类连续时间随机过程。布朗运动的轨迹 $t \mapsto W_t$ 几乎必然处处连续，且 $W_0 = 0$，并满足以下两个重要性质：

- 高斯增量：对于任意 $0 \leq s < t$，增量 $(W_t - W_s)$ 服从均值为 $\mathbf{0}$、协方差为 $(t-s)\mathbf{I}$ 的多元高斯分布。增量的方差随时间间隔 $(t-s)$ 线性增长。
- 独立增量：对于任意给定的时间序列 $0 \leq t_0 < t_1 < \cdots < t_T = 1$，不同时段上的增量 $(W_{t_1} - W_{t_0}), \ldots, (W_{t_T} - W_{t_{T-1}})$ 相互独立。

随机微分方程通常难以获得解析解，因此也往往需要借助数值离散化方法进行近似求解。最基础且最常用的数值方法之一是**欧拉–丸山方法**（Euler-Maruyama Method），该方法可以看作是常微分方程中欧拉方法在随机微分方程上的推广：

$$X_{t+h} = X_t + hu_t(X_t) + \sigma_t(W_{t+h} - W_t) \tag{8.53}$$

$$= X_t + \underbrace{hu_t(X_t)}_{\text{确定项}} + \underbrace{\sigma_t\sqrt{h}\boldsymbol{\epsilon}}_{\text{随机项}} \tag{8.54}$$

其中 $\boldsymbol{\epsilon} \sim \mathcal{N}(\boldsymbol{\epsilon}|\mathbf{0}, \mathbf{I})$ 为与当前状态独立的多元标准高斯随机变量。上式中的第一项描述由漂移项引起的确定性变化，而第二项描述由布朗运动增量引起的随机扰动。上述递推关系可与降噪扩散模型生成过程中的迭代更新公式8.50相对应。降噪扩散模型的前向过程通常通过逐步加噪，将数据分布逐渐转化为标准高斯分布，从而构造出一条从真实数据分布到噪声分布的概率演化路径。在这一过程中，状态演化既包含使真实数据信息逐渐衰减的确定性漂移，也包含了注入高斯噪声的随机扩散。这种确定性漂移与随机扩散的组合，在数学上可以由随机微分方程描述。传统降噪扩散模型的前向加噪过程可表示为（即公式8.36）：

$$\boldsymbol{x}_t = \sqrt{\bar{\alpha}_t}\boldsymbol{x}_0 + \sqrt{1-\bar{\alpha}_t}\boldsymbol{\epsilon} \tag{8.55}$$

对时间 t 求导可得到对应的条件速度向量[2]（即条件轨迹的切向量）：

$$u_t(\boldsymbol{x}_t|\boldsymbol{x}_0) = \frac{\mathrm{d}\boldsymbol{x}_t}{\mathrm{d}t} = \frac{\mathrm{d}\sqrt{\bar{\alpha}_t}}{\mathrm{d}t}\boldsymbol{x}_0 + \frac{\mathrm{d}\sqrt{1-\bar{\alpha}_t}}{\mathrm{d}t}\boldsymbol{\epsilon} \tag{8.56}$$

从上式可以看出，该速度向量同时受到数据项和噪声项的影响，其方向和幅值均会随着时间持续变化。因此，降噪扩散模型对应的生成轨迹通常具有较高的曲率，速度向量也会随着时间发生显著变化。为了准确逼近这种连续时间动力学过程，防止生成轨迹偏离理论路径，往往需要采用较小的时间步长和较多的迭代步数。相比之下，流匹配模型将生成过程建模为常微分方程，

[2]严格地说，这里构造的确定性插值轨迹在各时间步具有与去噪扩散概率模型前向扩散过程相同的条件边缘分布，但其跨时间步的联合分布一般不同。以下导数描述的是该插值轨迹的变化速度，而非原随机扩散过程的样本轨迹导数。

其轨迹仅由确定性向量场驱动，在数值求解过程中不再包含随机扰动项。在最常见的线性插值流匹配模型中（详见后文介绍），设 $\boldsymbol{x}_0$ 表示噪声样本，而 $\boldsymbol{x}_1$ 表示数据样本，则其条件速度向量可以表示为：

$$u_t(\boldsymbol{x}_t|\boldsymbol{x}_0,\boldsymbol{x}_1)=\boldsymbol{x}_1-\boldsymbol{x}_0 \tag{8.57}$$

由此可以看出，该条件速度向量与时间 t 无关，因而对应的条件轨迹为连接 $\boldsymbol{x}_0$ 与 $\boldsymbol{x}_1$ 的直线。由于轨迹方向在整个运动过程中保持不变，速度向量随时间的变化更加平缓，从而允许采用更大的时间步长进行数值求解。这使得流匹配模型能够显著减少生成过程中的迭代步数（通常仅需数十步），从而有效提升生成效率。需要注意的是，线性条件路径并不意味着模型实际生成的轨迹一定是直线。这是因为生成时使用的速度场综合了不同条件路径的速度信息，其对应的生成轨迹仍可能发生弯曲。图8.12展示了扩散模型与流匹配模型在连续状态空间中的生成轨迹对比。

需要说明的是，上述将扩散模型与随机微分方程联系起来，而将流匹配模型与常微分方程联系起来，主要是为了便于直观理解。事实上，在连续时间扩散模型中，既可以通过随机的反向扩散过程（反向随机微分方程）进行采样，也可以通过确定性的概率流常微分方程进行采样。而流匹配本质上是一种学习速度场的训练方法，其采用的概率路径并不局限于线性插值路径，也可以采用扩散形式的概率路径。二者并不能简单地按照 “随机过程” 与 “确定性过程” 进行区分。

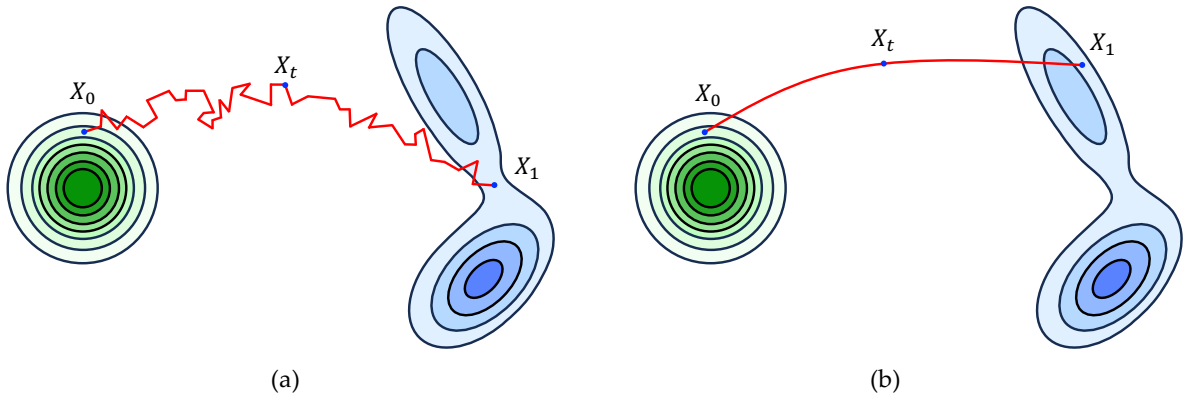


图 8.12: 扩散与流的对比示意图。(a) 连续状态空间中的随机扩散轨迹（由随机微分方程描述）。(b) 连续状态空间中的确定性流轨迹（由常微分方程描述）。此例改编自文献 [49]。

从上述讨论可知，多步生成过程可以统一建模为常微分方程，而常微分方程的解轨迹则对应于样本从初始噪声逐渐演化为目标图像的生成过程。采用欧拉方法及其变体，可以对常微分方程的解轨迹进行数值近似求解。根据欧拉方法的递推公式8.49，在求解过程中唯一需要确定的是对应的向量场 $u_t(X_t)$。因此，**流匹配**（Flow Matching）模型的学习目标可以概括为：利用深度神经网络拟合一个参数化向量场 $u_t^{\theta}(X_t)$（其中 $\boldsymbol{\theta}$ 为网络参数），使其产生的流 ψ_t 能够将初始噪声分布逐步变换为真实图像分布。具体而言，该流满足 $X_t=\psi_t(X_0)$，且当时间 t 从 0 演化到 1 时，初始状态和最终状态分别满足 $X_0\sim p_0$（高斯噪声分布）和 $X_1\sim p_1$（真实图像分布）。为了实现这一学习目标，通常需要遵循以下两个步骤。首先，设计一条在时间上连续连接初始噪声分布 p_0 与目标分布 p_1 的概率路径 p_t；然后，通过回归训练深度神经网络 $u_t^{\theta}(X_t)$，使其逼

近能够生成该概率路径 p_t 的目标向量场 $u_t(X_t)$。当学习得到的向量场能够准确生成预先设计的概率路径时，样本便可沿着对应的流从噪声分布逐步演化到真实图像分布，从而完成生成过程。

我们先对连接初始分布与目标分布的概率路径 p_t 进行设计。设初始分布为 $p_0 = \mathcal{N}(\boldsymbol{x}|\mathbf{0}, \mathbf{I})$，概率路径 p_t 可以看作是一组条件概率路径 $p_t(\boldsymbol{x}|\boldsymbol{x}_1)$ 的连续混合（即边缘化结果），其中每一条条件概率路径以数据集中的一个真实样本 $X_1 = \boldsymbol{x}_1$ 作为条件。因此，概率路径 p_t 可以表示为：

$$p_t(\boldsymbol{x}) = \int p_t(\boldsymbol{x}|\boldsymbol{x}_1)p_1(\boldsymbol{x}_1)\, d\boldsymbol{x}_1 \tag{8.58}$$

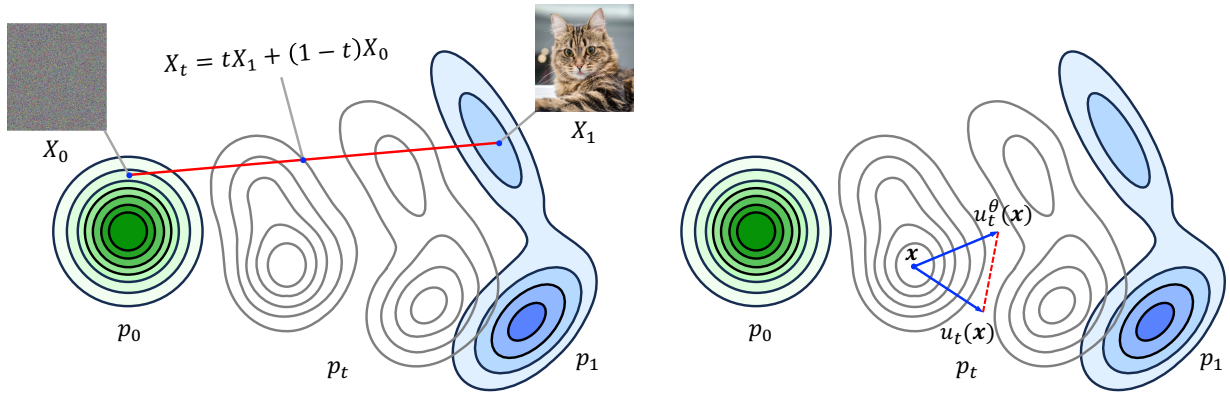


图 8.13: 流匹配模型的概率路径与训练目标。(a) 对于给定的初始状态 X_0 和真实样本 X_1，条件最优传输路径将时刻 t 的状态 X_t 表示为二者之间的线性插值。(b) 在位置 $\boldsymbol{x}$ 处，$u_t(\boldsymbol{x})$ 表示目标速度向量，$u_t^\theta(\boldsymbol{x})$ 表示由神经网络预测的速度向量。模型训练的目标是最小化两者之间的误差，即流匹配损失（图中红色虚线所示）。

对于最常用的线性插值路径，我们将条件概率路径定义为：

$$p_t(\boldsymbol{x}|\boldsymbol{x}_1) = \mathcal{N}(\boldsymbol{x}|t\boldsymbol{x}_1, (1-t)^2\mathbf{I}) \tag{8.59}$$

这样的概率路径被称为**条件最优传输路径**（Conditional Optimal Transport Path）或者线性路径（Linear Path）。基于该概率路径，设 $X_0 \sim p_0$ 为从初始噪声分布采样得到的随机变量，而 $X_1 \sim p_1$ 为从目标分布采样得到的随机变量，则任意时刻 t 对应的状态 $X_t \sim p_t$ 可表示为如下线性插值形式（如图8.13(a) 所示）：

$$X_t = tX_1 + (1-t)X_0 \tag{8.60}$$

我们可以证明，在给定 $X_1 = \boldsymbol{x}_1$ 的条件下，上式所对应的条件分布即为公式8.59所定义的多元高斯分布 $\mathcal{N}(\boldsymbol{x}|t\boldsymbol{x}_1, (1-t)^2\mathbf{I})$。为此，只需计算其条件均值和协方差矩阵：

$$\mathbb{E}[X_t|X_1 = \boldsymbol{x}_1] = \mathbb{E}[t\boldsymbol{x}_1 + (1-t)X_0] = t\boldsymbol{x}_1 + (1-t)\underbrace{\mathbb{E}[X_0]}_{=\mathbf{0}} = t\boldsymbol{x}_1$$

$$\mathrm{Cov}[X_t|X_1 = \boldsymbol{x}_1] = \mathrm{Cov}[t\boldsymbol{x}_1 + (1-t)X_0] = \underbrace{\mathrm{Cov}[t\boldsymbol{x}_1]}_{=\mathbf{0}} + (1-t)^2\mathrm{Cov}[X_0] = (1-t)^2\mathbf{I}$$

在给定 $X_1 = \boldsymbol{x}_1$ 的条件下，$t\boldsymbol{x}_1$ 为常数，因此其协方差矩阵为零矩阵。

如图8.13(b) 所示，在确定概率路径 p_t 之后，我们可以通过最小化如下均方误差损失函数来

训练一个深度神经网络 $u_t^\theta(X_t)$，使其拟合目标向量场 $u_t(X_t)$：

$$\mathcal{L}(\boldsymbol{\theta}) = \mathbb{E}_{t\sim \mathrm{U}(t|0,1), X_t\sim p_t}\left[\|u_t^\theta(X_t) - u_t(X_t)\|^2\right] \tag{8.61}$$

其中 $t \sim \mathrm{U}(t|0,1)$ 表示在时间区间 $[0,1]$ 上进行均匀采样。该损失函数旨在使参数化的神经网络逼近能够生成所定义概率路径 p_t 的目标向量场 $u_t(X_t)$。对于条件最优传输路径（公式8.60），在给定样本对 (X_0, X_1) 的条件下，其样本路径关于时间 t 的导数为：

$$\frac{\mathrm{d}}{\mathrm{d}t}X_t = \frac{\mathrm{d}}{\mathrm{d}t}\left(tX_1 + (1-t)X_0\right) = X_1 - X_0 \tag{8.62}$$

需要注意的是，$(X_1 - X_0)$ 表示单条样本轨迹对应的速度向量，而目标向量场 $u_t(X_t)$ 则是定义在状态空间上的确定性函数。条件流匹配定理表明，利用样本轨迹速度 $(X_1 - X_0)$ 作为监督信号进行回归，与直接拟合目标向量场 $u_t(X_t)$ 具有相同的全局最优解。因此，在实际训练过程中，可以采用如下等价的损失函数：

$$\mathcal{L}(\boldsymbol{\theta}) = \mathbb{E}_{t\sim \mathrm{U}(t|0,1), X_0\sim p_0, X_1\sim p_1}\left[\|u_t^\theta(X_t) - (X_1 - X_0)\|^2\right] \tag{8.63}$$

该形式仅依赖于从初始分布 p_0 和目标分布 p_1 采样得到的样本对，而无需显式计算目标向量场 $u_t(X_t)$。因此，流匹配将向量场学习问题转化为一个简单的回归问题，从而显著降低模型训练的复杂度。流匹配模型的训练过程与生成过程分别如算法8.4和算法8.5所示。在算法的描述中，用于预测速度向量场的神经网络记作 $u_\theta(\boldsymbol{x}_t, t)$，以强调时间 t 与状态 $\boldsymbol{x}_t$ 共同作为该网络的输入。

算法 8.4: 流匹配模型训练算法

输入: 真实数据集 $\mathcal{D} = \{\boldsymbol{x}_i\}_{i=1}^N$ 以及最大迭代次数 K。

初始化: 随机初始化向量场神经网络参数 $\boldsymbol{\theta}$，并设当前迭代次数 $k = 1$；

while $k \le K$ **do**

- 从数据集 $\mathcal{D}$ 中随机采样样本 $\boldsymbol{x}_1$；
- 从连续均匀分布中随机采样时间 t；
- 从多元标准高斯分布中采样噪声 $\boldsymbol{x}_0$；
- 线性插值：$\boldsymbol{x}_t = t\boldsymbol{x}_1 + (1-t)\boldsymbol{x}_0$；
- 按损失函数计算梯度并更新网络参数：$\nabla_\theta \|u_\theta(\boldsymbol{x}_t, t) - (\boldsymbol{x}_1 - \boldsymbol{x}_0)\|^2$；
- $k \leftarrow k + 1$；

输出: 训练完成后得到的网络参数 $\boldsymbol{\theta}$。

算法 8.5: 流匹配模型生成算法

输入: 预测速度向量神经网络参数 $\boldsymbol{\theta}$ 以及常微分方程求解步数 T。

初始化: 从多元标准高斯分布 $\mathcal{N}(\boldsymbol{x}_0|\mathbf{0}, \mathbf{I})$ 中随机采样初始噪声 $\boldsymbol{x}_0$；

计算步长：$h = \frac{1}{T}$；

for $k \leftarrow 0$ **to** $T-1$ **do**

- 计算当前对应的时刻：$t \leftarrow h \times k$；
- 利用神经网络预测当前速度向量：$\boldsymbol{v}_t = u_\theta(\boldsymbol{x}_t, t)$；
- 利用欧拉方法沿向量场向前推进：$\boldsymbol{x}_{t+h} = \boldsymbol{x}_t + h\boldsymbol{v}_t$；

输出: 生成的样本或图像 $\boldsymbol{x}_1$。

流匹配模型的训练算法在每次迭代中，首先随机采样一对样本 $(\boldsymbol{x}_0, \boldsymbol{x}_1)$，其中 $\boldsymbol{x}_0$ 为从多元高斯分布 $\mathcal{N}(\mathbf{0}, \mathbf{I})$ 中采样得到的噪声样本，而 $\boldsymbol{x}_1$ 为真实数据样本（如图像）；随后从区间 $[0,1]$ 上按照均匀分布 $\mathrm{U}(t|0,1)$ 随机采样时刻 t，并根据 $\boldsymbol{x}_0$、$\boldsymbol{x}_1$ 和 t 通过线性插值构造中间状态 $\boldsymbol{x}_t$；接着将 $\boldsymbol{x}_t$ 和 t 输入深度神经网络，预测对应的速度向量 $u_\theta(\boldsymbol{x}_t, t)$，并通过计算与目标速度向量 $(\boldsymbol{x}_1 - \boldsymbol{x}_0)$ 之间的均方误差来求得损失函数关于模型参数的梯度，从而完成参数更新（通常以小批量方式进行）。流匹配模型的生成过程本质上是对所学习的常微分方程进行数值求解。最简单的做法是采用欧拉方法对常微分方程进行离散积分，从初始噪声逐步生成目标样本。在实际应用中，还可以采用休恩方法（Heun's Method）等更高阶的数值解法 [40]。

主要符号表

数学符号

$\boldsymbol{x}$	向量
y	标量
D	向量维度
x_j	向量的第 j 个分量（向量分量为标量）
f	输入到输出的某种映射
$\mathbf{X}^\top$	矩阵转置
$\mathbf{X}^{-1}$	逆矩阵
$\|\mathbf{X}\|$	矩阵的行列式
$\mathrm{tr}(\mathbf{X})$	矩阵的迹（矩阵所有特征值之和，也等于主对角线元素的总和）
$\ln$	以自然常数 e 为底的对数
e^x 或 $\exp(x)$	自然常数 e 的指数函数
$f'(x)$	一阶导数
$f''(x)$	二阶导数
$\nabla f(\boldsymbol{x})$	一阶梯度（梯度向量）
$\nabla^2 f(\boldsymbol{x})$	二阶梯度（海森矩阵）
$\frac{\partial f(x)}{\partial x}$	一阶偏导
$\frac{\partial^2 f(x)}{\partial x^2}$	二阶偏导

机器学习

$\mathcal{D}$	数据集
N	样本数量
$\boldsymbol{x}_i$	第 i 个训练样本
y_i	第 i 个训练样本对应的输出
$\kappa(\boldsymbol{x}, \boldsymbol{x}')$	核函数
$\mathcal{L}(\cdot)$	损失函数
$\mathcal{J}(\cdot)$	目标函数或代价函数

概率统计

$\mathbb{E}[x]$	均值或期望
$\text{Var}[x]$	方差
$\text{Cov}[x, z]$	随机变量 x 和 z 之间的协方差
$\Gamma(z)$	伽玛函数
$\mathcal{N}(x\|\mu, \sigma^2)$	高斯分布，其中 μ 为均值，σ 为标准差
$\mathcal{N}(x\|\mu, \lambda^{-1})$	高斯分布，其中 μ 为均值，λ 为精度
$\text{Bern}(x\|\mu)$	伯努利分布
$\text{Bin}(k\|\mu, N)$	二项分布
$\text{Beta}(\mu\|\alpha, \beta)$	贝塔分布
$\text{Poisson}(k\|\lambda)$	泊松分布
$\text{Exp}(\lambda\|\beta)$	指数分布
$\text{Gamma}(\lambda\|\alpha, \beta)$	伽玛分布，其中 α 为形状参数，β 为逆尺度参数
$\text{Mult}(\boldsymbol{m}\|\boldsymbol{\mu}, N)$	多项分布
$\text{Dir}(\boldsymbol{\mu}\|\boldsymbol{\alpha})$	狄利克雷分布
$\mathcal{N}(\boldsymbol{x}\|\boldsymbol{\mu}, \boldsymbol{\Sigma})$	多维高斯分布，其中 $\boldsymbol{\mu}$ 为均值向量，$\boldsymbol{\Sigma}$ 为协方差矩阵
$\mathcal{N}(\boldsymbol{x}\|\boldsymbol{\mu}, \boldsymbol{\Lambda})$	多维高斯分布，其中 $\boldsymbol{\mu}$ 为均值向量，$\boldsymbol{\Lambda}$ 为精度矩阵
$\mathcal{W}(\boldsymbol{\Lambda}\|\mathbf{W}, \eta)$	威沙特分布，其中 $\mathbf{W}$ 为尺度矩阵，η 为自由度
$\text{NormalGamma}(\mu, \lambda\|\upsilon, \nu, \alpha, \beta)$	高斯-伽玛分布
$\text{NormalWishart}(\boldsymbol{\mu}, \boldsymbol{\Lambda}\|\boldsymbol{\upsilon}, \nu, \mathbf{W}, \eta)$	高斯-威沙特分布
$\text{St}(x\|\mu, \tau, \eta)$	学生氏分布，其中 μ 为均值，τ 为精度，η 为自由度
$\text{St}(\boldsymbol{x}\|\boldsymbol{\mu}, \boldsymbol{\Lambda}, \eta)$	多维学生氏分布，其中 $\boldsymbol{\mu}$ 为均值向量，$\boldsymbol{\Lambda}$ 为精度矩阵
$\text{Laplace}(x\|\mu, \tau)$	拉普拉斯分布，其中 μ 为位置参数，τ 为尺度参数
$\chi^2(x\|\eta)$	卡方分布，其中 η 为自由度
$\text{U}(x\|a, b)$	连续型均匀分布，其中 a 为下界、b 为上界 $(a < b)$
$\text{H}[x]$	信息熵，其中 x 为随机变量
$\text{H}[y\|x]$	条件熵，其中 x 和 y 为随机变量
$\text{H}[p, q]$	分布 p 和 q 之间的交叉熵
$D_{\text{KL}}(p\|\|q)$	分布 p 和 q 之间的 KL 散度

参考文献

[1] Daniel J. Amit, Hanoch Gutfreund, and Haim Sompolinsky. “Storing infinite numbers of patterns in a spin-glass model of neural networks”. In: *Physical Review Letters* 55.14 (1985), pp. 1530–1533.

[2] Mikel Artetxe et al. “On the role of bidirectionality in language model pre-training”. In: *Findings of the Conference on Empirical Methods in Natural Language Processing* (2022), pp. 3973–3985.

[3] Jimmy Lei Ba, Jamie Ryan Kiros, and Geoffrey E Hinton. “Layer normalization”. In: *arXiv: 1607.06450* (2016).

[4] Yoshua Bengio, Patrice Simard, and Paolo Frasconi. “Learning long-term dependencies with gradient descent is difficult”. In: *IEEE Transactions on Neural Networks* 5.2 (1994), pp. 157–166.

[5] Jame O. Berger. *Statistical decision theory and Bayesian analysis (second edition)*. Springer, 1980.

[6] James Bergstra and Yoshua Bengio. “Random search for hyper-parameter optimization”. In: *Journal of Machine Learning Research* 13.2 (2012).

[7] Jeremy Bernstein and Laker Newhouse. “Old optimizer, new norm: An anthology”. In: *Proceedings of the 16th Annual Workshop on Optimization for Machine Learning* (2024), pp. 1–19.

[8] J Besag. “On spatio-temporal models and Markov fields”. In: *Transactions of the 7th Prague Conference on Information Theory, Statistical Decision Functions and Random Processes* (1974), pp. 47–75.

[9] Julian Besag. “On the statistical analysis of dirty pictures”. In: *Journal of the Royal Statistical Society Series B: Statistical Methodology* 48.3 (1986), pp. 259–279.

[10] Christopher M. Bishop. *Pattern recognition and machine learning*. Springer, 2006.

[11] Bernhard E. Boser, Isabelle M. Guyon, and Vladimir N. Vapnik. “A training algorithm for optimal margin classifiers”. In: *Proceedings of the Fifth Annual Workshop on Computational Learning Theory*. 1992, pp. 144–152.

[12] George E.P. Box and Norman Richard Draper. *Empirical model-building and response surfaces*. John Wiley & Sons, 1987.

[13] Stephen Boyd and Lieven Vandenberghe. *Convex optimization*. Cambridge University Press, 2004.

[14] Tom Brown et al. “Language models are few-shot learners”. In: *Proceedings of the Conference on Neural Information Processing Systems* 33 (2020), pp. 1877–1901.

[15] Kyunghyun Cho et al. “Learning phrase representations using RNN encoder–decoder for statistical machine translation”. In: *Proceedings of the 2014 Conference on Empirical Methods in Natural Language Processing* (2014), pp. 1724–1734.

[16] Peter Clifford. “Markov random fields in statistics”. In: *Disorder in Physical Systems: A Volume in Honour of John M. Hammersley* (1990), pp. 19–32.

[17] Thomas H. Cormen et al. *Introduction to algorithms (third edition)*. The MIT Press, 2022.

[18] Thomas M. Cover and Joy A. Thomas. *Elements of information theory*. Wiley, 1991.

[19] George Cybenko. “Approximation by superpositions of a sigmoidal function”. In: *Mathematics of control, signals and systems* 2.4 (1989), pp. 303–314.

[20] Morris H. DeGroot and Mark J. Schevish. *Probability and statistics (forth edition)*. China Machine Press, 2012.

[21] Jacob Devlin et al. “BERT: Pre-training of deep bidirectional transformers for language understanding”. In: *Proceedings of the 2019 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies* (2019), pp. 4171–4186.

[22] Alexey Dosovitskiy et al. “An image is worth 16×16 words: Transformers for image recognition at scale”. In: *Proceedings of the International Conference on Learning Representations* (2021).

[23] John Duchi, Elad Hazan, and Yoram Singer. “Adaptive subgradient methods for online learning and stochastic optimization”. In: *Journal of Machine Learning Research* 12.7 (2011).

[24] Walter D. Fisher. “On grouping for maximum homogeneity”. In: *Journal of the American Statistical Association* 53.284 (1958), pp. 789–798.

[25] Roger Fletcher. *Practical methods of optimization (second edition)*. Wiley, 1987.

[26] Yoav Freund and Robert E. Schapire. “A decision-theoretic generalization of on-line learning and an application to boosting”. In: *Journal of Computer and System Sciences* 55.1 (1997), pp. 119–139.

[27] Stuart Geman and Donald Geman. “Stochastic relaxation, Gibbs distributions, and the Bayesian restoration of images”. In: *IEEE Transactions on pattern analysis and machine intelligence* 6 (1984), pp. 721–741.

[28] Felix A. Gers, Nicol N. Schraudolph, and Jürgen Schmidhuber. “Learning precise timing with LSTM recurrent networks”. In: *Journal of Machine Learning Research* 3 (2002), pp. 115–143.

[29] Xavier Glorot and Yoshua Bengio. “Understanding the difficulty of training deep feedforward neural networks”. In: *Proceedings of the Thirteenth International Conference on Artificial Intelligence and Statistics* (2010), pp. 249–256.

[30] Ian J Goodfellow et al. “Generative adversarial nets”. In: *Proceedings of the Conference on Neural Information Processing Systems* 27 (2014).

[31] Trevor Hastie, Robert Tibshirani, and Jerome Friedman. *The elements of statistical learning: data mining, inference, and prediction (second edition)*. Springer, 2009.

[32] Kaiming He and Jian Sun. “Convolutional neural networks at constrained time cost”. In: *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition* (2015), pp. 5353–5360.

[33] Kaiming He et al. “Deep residual learning for image recognition”. In: *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition* (2016), pp. 770–778.

[34] Kaiming He et al. “Delving deep into rectifiers: Surpassing human-level performance on imagenet classification”. In: *Proceedings of the IEEE International Conference on Computer Vision* (2015), pp. 1026–1034.

[35] Nicholas J. Higham. *Functions of matrices: Theory and computation*. SIAM, 2008.

[36] Geoffrey Hinton, Oriol Vinyals, and Jeff Dean. “Distilling the knowledge in a neural network”. In: *arXiv: 1503.02531* (2015).

[37] Geoffrey E. Hinton. “Training products of experts by minimizing contrastive divergence”. In: *Neural Computation* 14.8 (2002), pp. 1771–1800.

[38] John J. Hopfield. “Neural networks and physical systems with emergent collective computational abilities”. In: *Proceedings of the National Academy of Sciences* 79.8 (1982), pp. 2554–2558.

[39] Sergey Ioffe and Christian Szegedy. “Batch normalization: Accelerating deep network training by reducing internal covariate shift”. In: *Proceedings of the International Conference on Machine Learning* (2015), pp. 448–456.

[40] Arieh Iserles. *A first course in the numerical analysis of differential equations*. 44. Cambridge university press, 2009.

[41] Max Jaderberg et al. “Population based training of neural networks”. In: *arXiv: 1711.09846* (2017).

[42] Kevin Jamieson and Ameet Talwalkar. “Non-stochastic best arm identification and hyperparameter optimization”. In: *Proceedings of the 19th International Conference on Artificial Intelligence and Statistics* (2016), pp. 240–248.

[43] Keller Jordan et al. *Muon: An optimizer for hidden layers in neural networks*. 2024. URL: https://kellerjordan.github.io/posts/muon/.

[44] Diederik P. Kingma and Jimmy L. Ba. “Adam: A method for stochastic optimization”. In: *Proceedings of the International Conference on Learning Representations* (2015).

[45] Josef Kittler and Janos Föglein. “Contextual classification of multispectral pixel data”. In: *Image and vision computing* 2.1 (1984), pp. 13–29.

[46] Solomon Kullback and Richard A. Leibler. “On information and sufficiency”. In: *The Annals of Mathematical Statistics* 22.1 (1951), pp. 79–86.

[47] Yann LeCun et al. “Gradient-based learning applied to document recognition”. In: *Proceedings of the IEEE* 86.11 (2002), pp. 2278–2324.

[48] Mike Lewis et al. “BART: Denoising sequence-to-sequence pre-training for natural language generation, translation, and comprehension”. In: *Proceedings of the 58th Annual Meeting of the Association for Computational Linguistics* (2020), pp. 7871–7880.

[49] Yaron Lipman et al. “Flow matching guide and code”. In: *arXiv: 2412.06264* (2024).

[50] Ilya Loshchilov and Frank Hutter. “Decoupled weight decay regularization”. In: *arXiv: 1711.05101* (2017).

[51] Calvin Luo. “Understanding diffusion models: A unified perspective”. In: *arXiv: 2208.11970* (2022).

[52] Marvin Minsky and Seymour Papert. *Perceptrons: An introduction to computational geometry*. The MIT Press, 1969.

[53] Kevin P. Murphy. *Machine learning: A probabilistic perspective*. The MIT Press, 2012.

[54] Brady Neal et al. “A modern take on the bias-variance tradeoff in neural networks”. In: *arXiv: 1810.08591* (2018).

[55] Long Ouyang et al. “Training language models to follow instructions with human feedback”. In: *Proceedings of the Conference on Neural Information Processing Systems* 35 (2022), pp. 27730–27744.

[56] Judea Pearl. *Probabilistic reasoning in intelligent systems*. Morgan kaufmann, 2014.

[57] Matthew E. Peters et al. “Deep Contextualized Word Representations”. In: *Proceedings of the 2018 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies* (2018), pp. 2227–2237.

[58] John C. Platt. “Fast training of support vector machines using sequential minimal optimization”. In: *Advances in Kernel Methods* (1999), pp. 185–208.

[59] Alec Radford, Luke Metz, and Soumith Chintala. “Unsupervised representation learning with deep convolutional generative adversarial networks”. In: *Proceedings of the International Conference on Learning Representations* (2016).

[60] Alec Radford et al. “Improving language understanding by generative pre-training”. In: *OpenAI’s Technical Report* (2018).

[61] Colin Raffel et al. “Exploring the limits of transfer learning with a unified text-to-text transformer”. In: *Journal of Machine Learning Research* 21.140 (2020), pp. 1–67.

[62] Scott Reed et al. “Generative adversarial text to image synthesis”. In: *Proceedings of the International Conference on Machine Learning* (2016), pp. 1060–1069.

[63] Edmund T. Rolls. *Cerebral cortex: principles of operation*. Oxford University Press, 2016.

[64] Olaf Ronneberger, Philipp Fischer, and Thomas Brox. “U-Net: Convolutional networks for biomedical image segmentation”. In: *Proceedings of the International Conference on Medical Image Computing and Computer-Assisted Intervention* (2015), pp. 234–241.

[65] David E. Rumelhart, Geoffrey E. Hinton, and Ronald J. Williams. “Learning representations by back-propagating errors”. In: *Nature* 323.6088 (1986), pp. 533–536.

[66] Leonard J. Savage. *The subjective basis of statistical practice*. University of Michigan, 1961.

[67] Andrew M Saxe, James L McClelland, and Surya Ganguli. “Exact solutions to the nonlinear dynamics of learning in deep linear neural networks”. In: *Proceedings of the International Conference on Learning Representations* (2014).

[68] Robert E. Schapire. “The strength of weak learnability”. In: *Machine learning* 5.2 (1990), pp. 197–227.

[69] Rico Sennrich, Barry Haddow, and Alexandra Birch. “Neural machine translation of rare words with subword units”. In: *Proceedings of the 54th Annual Meeting of the Association for Computational Linguistics* (2016), pp. 1715–1725.

[70] Claude Elwood Shannon. “A mathematical theory of communication”. In: *The Bell System Technical Journal* 27.3 (1948), pp. 379–423.

[71] Jasper Snoek, Hugo Larochelle, and Ryan P Adams. “Practical Bayesian optimization of machine learning algorithms”. In: *Proceedings of the Conference on Neural Information Processing Systems* 25 (2012).

[72] Yang Song et al. “Score-based generative modeling through stochastic differential equations”. In: *Proceedings of the International Conference on Learning Representations* (2021).

[73] Gilbert Strang. *Introduction to linear algebra*. Wellesley-Cambridge Press, 2016.

[74] Jianlin Su et al. “Roformer: Enhanced transformer with rotary position embedding”. In: *Neurocomputing* 568 (2024), p. 127063.

[75] Tijmen Tieleman and Geoffrey E. Hinton. “Lecture 6.5-RMSprop: Divide the gradient by a running average of its recent magnitude”. In: *COURSERA: Neural Networks for Machine Learning* 4.2 (2012), pp. 26–26.

[76] Antonio Torralba, Phillip Isola, and William Freeman. *Foundations of computer vision*. The MIT Press, 2024.

[77] Ashish Vaswani et al. “Attention is all you need”. In: *Proceedings of the Conference on Neural Information Processing Systems* 30 (2017).

[78] Xiaoqing Zheng, Hanyang Chen, and Tianyu Xu. “Deep learning for Chinese word segmentation and POS tagging”. In: *Proceedings of the 2013 Conference on Empirical Methods in Natural Language Processing* (2013), pp. 647–657.

[79] Ji Zhu et al. “Multi-class AdaBoost”. In: *Statistics and its Interface* 2.3 (2009), pp. 349–360.

[80] Hui Zou and Trevor Hastie. “Regularization and variable selection via the elastic net”. In: *Journal of the Royal Statistical Society Series B: Statistical Methodology* 67.2 (2005), pp. 301–320.