

生成式 AI 学习笔记

Author: 张哲源

Email:

Date: 2026-02-10

1 变分自编码器 (Variational Autoencoder, VAE)

符号说明 (Notation)

符号	含义
\mathcal{D}	数据集, $\mathcal{D} = \{\mathbf{x}^{(1)}, \dots, \mathbf{x}^{(n)}\}$
$\mathbf{x}^{(i)}$	第 i 个数据样本, 维度为 d
\mathbf{z}	瓶颈层学习到的压缩隐变量 (Latent Code)
$g_\phi(\cdot)$	编码器函数, 参数为 ϕ
$f_\theta(\cdot)$	解码器函数, 参数为 θ
$q_\phi(\mathbf{z} \mathbf{x})$	推断网络 (近似后验), 概率编码器
$p_\theta(\mathbf{x} \mathbf{z})$	生成网络 (似然), 概率解码器

1.1 前置知识：自编码器及其变体 (Autoencoders)

Definition 1.1 (自动编码器). 自动编码器 (Autoencoder, AE) 是一种无监督学习模型, 其核心目标是学习一个恒等函数 (Identity Function), 即输出 $\mathbf{x}' \approx \mathbf{x}$ 。它通过一个具有窄瓶颈层 (Bottleneck) 的神经网络将高维输入压缩为低维编码 (降维), 再从该编码重构出原始输入。

1.1.1 基本自动编码器 (Autoencoder)

自动编码器通常包含两个部分:

- **编码器 (Encoder)** $g_\phi(\cdot)$: 将高维输入 \mathbf{x} 映射为低维隐变量 $\mathbf{z} = g_\phi(\mathbf{x})$ 。
- **解码器 (Decoder)** $f_\theta(\cdot)$: 将隐变量 \mathbf{z} 映射回重构数据 $\mathbf{x}' = f_\theta(\mathbf{z})$ 。

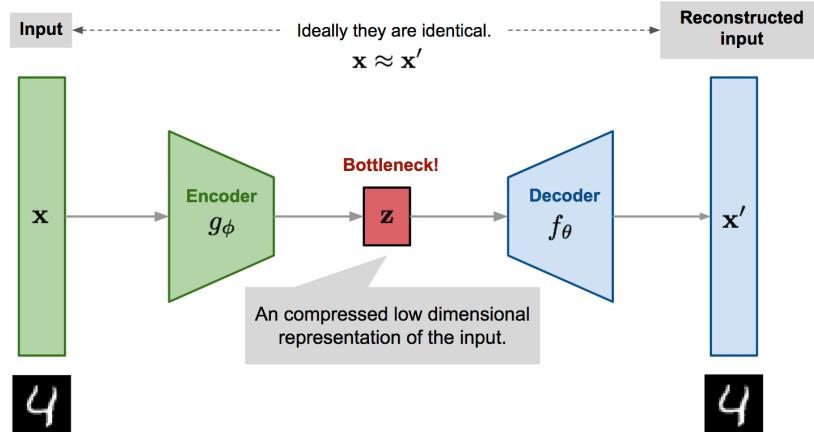


图 1-1 自动编码器模型架构示意图

训练目标是最小化重构误差，例如均方误差 (MSE):

$$L_{AE}(\theta, \phi) = \frac{1}{n} \sum_{i=1}^n (\mathbf{x}^{(i)} - f_\theta(g_\phi(\mathbf{x}^{(i)})))^2 \quad (1)$$

1.1.2 去噪自动编码器 (Denoising Autoencoder, DAE)

为了避免过拟合（即单纯地记忆输入数据），去噪自动编码器引入了“破坏”机制。

Note 1.1 (核心思想). DAE 在输入 \mathbf{x} 中主动加入噪声得到 $\tilde{\mathbf{x}}$ (例如将部分像素置零, 类似 Dropout)，然后训练模型从损坏的 $\tilde{\mathbf{x}}$ 中恢复出原始的无噪数据 \mathbf{x} 。这迫使模型学习数据维度间的依赖关系，从而提取更鲁棒的特征。

损失函数调整为：

$$L_{DAE}(\theta, \phi) = \frac{1}{n} \sum_{i=1}^n (\mathbf{x}^{(i)} - f_\theta(g_\phi(\tilde{\mathbf{x}}^{(i)})))^2 \quad (2)$$

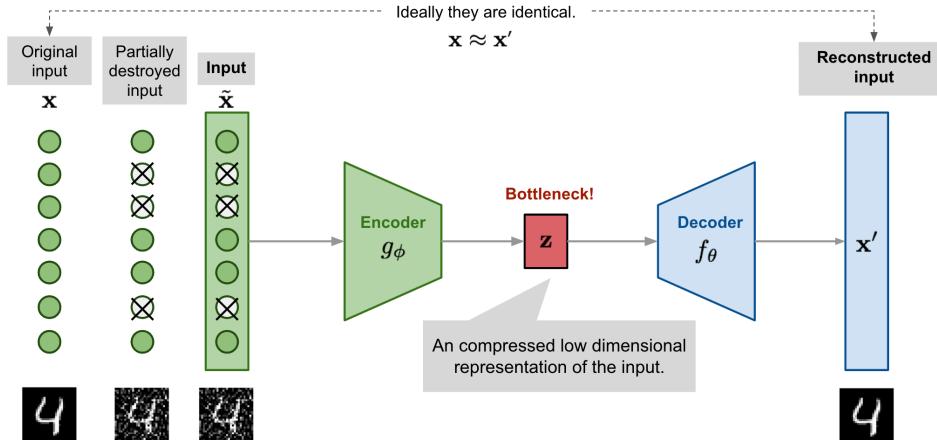


图 1-2 去噪自动编码器模型架构示意图

1.1.3 稀疏自动编码器 (Sparse Autoencoder, SAE)

稀疏自动编码器通过在损失函数中增加稀疏约束，强制隐层神经元在大部分时间处于“未激活”状态。假设隐藏层神经元 j 的平均激活度为 $\hat{\rho}_j$ ，我们希望它接近一个很小的稀疏参数 ρ （如 0.05）。通常使用 KL 散度作为惩罚项：

$$L_{\text{SAE}}(\theta) = L(\theta) + \beta \sum_{j=1}^{s_l} D_{\text{KL}}(\rho \parallel \hat{\rho}_j) \quad (3)$$

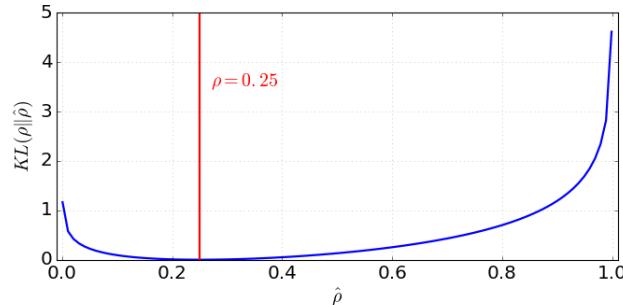


图 1-3 伯努利分布均值为 $\rho = 0.25$ 和均值为 $0 \leq \hat{\rho} \leq 1$ 的伯努利分布之间的 KL 散度

此外，还有 k -稀疏自动编码器 (k -Sparse Autoencoder)。其前向传播步骤如下：

1. 运行编码器得到潜在代码 $\mathbf{z} = g(\mathbf{x})$ 。
2. 对 \mathbf{z} 进行排序，仅保留前 k 个最大的激活值，将其余值置零，得到 $\mathbf{z}' = \text{Sparsify}(\mathbf{z})$ 。
3. 计算重构损失 $L = \|\mathbf{x} - f(\mathbf{z}')\|_2^2$ 。

注意：反向传播时，梯度仅通过这前 k 个激活的神经元传递。

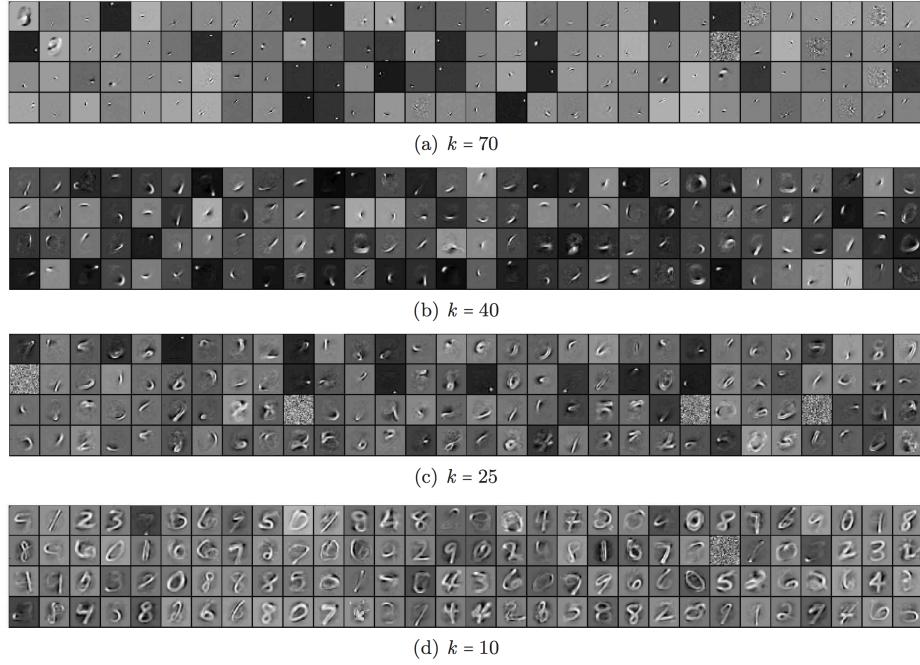


图 1-4 对不同稀疏级别 k 的 MNIST 数据集学习的 k -稀疏自动编码器的滤波器

1.1.4 收缩自动编码器 (Contracting Autoencoder, CAE)

收缩自动编码器旨在使学习到的表示对输入的微小扰动具有鲁棒性。它在损失函数中加入雅可比矩阵 (Jacobian Matrix) 的 Frobenius 范数作为惩罚项，抑制编码器激活值对输入的敏感度：

$$\|J_f(\mathbf{x})\|_F^2 = \sum_{ij} \left(\frac{\partial h_j(\mathbf{x})}{\partial x_i} \right)^2 \quad (4)$$

这促使模型学习到的特征分布在低维流形上，而对流形正交方向的变化保持不变。

1.2 VAE 原理详解 (The Principle of VAE)

Definition 1.2. 变分自编码器 (Variational Autoencoder, VAE) 是一种强大的深度生成模型，它巧妙地将深度学习的表示能力与贝叶斯推断的概率框架相结合。与传统的自编码器 (Autoencoder) 不同，VAE 并非学习一个从输入到编码的确定性映射，而是学习一个从输入到隐变量后验概率分布的映射。通过从这个学习到的分布中采样，VAE 能够生成与训练数据相似但全新的数据样本。

关键词：生成模型，变分自编码器，贝叶斯推断，证据下界，重参数化技巧

1.2.1 引言 (Introduction)

生成模型的根本目标是学习观测数据 X 的真实概率分布 $P_{data}(X)$ 。一旦这个分布被成功建模，我们便可以从中采样，生成新的数据。早期的生成模型如受限玻尔兹曼机 挑战

(RBM) 面临训练困难的问题，而传统的自编码器 (AE) 虽能学习数据的有效压缩表示，但其隐空间 (latent space) 缺乏良好的结构性，无法直接用于生成新样本。

Note 1.2 (VAE 的核心思想). 为了解决这一问题，Kingma 和 Welling 在 2013 年提出了变分自编码器。VAE 的核心思想是，不再将输入数据 X 编码为一个确定的隐向量 z ，而是将其编码为一个概率分布 (通常是高斯分布)。具体来说，模型学习这个分布的参数 (例如均值 μ 和方差 σ^2)。生成过程则变为：

1. 首先从这个分布中采样一个隐向量 z 。
2. 然后解码器 (Decoder) 基于这个 z 来重构出数据 \hat{X} 。

通过这种方式，VAE 不仅学习了如何重构数据，更重要的是，它学习到了一个结构化、连续的隐空间，使得我们可以在这个空间中进行采样和插值，从而生成多样化的新数据。

1.2.2 模型架构与概率框架 (Model Architecture & Probabilistic Framework)

从概率的视角来看，VAE 假设所有的数据样本 X 是由一个我们无法直接观测到的隐变量 z 生成的。这个生成过程包含以下步骤：

1. 从一个简单的先验分布 $p(z)$ 中采样一个隐变量 z 。通常，我们选择标准正态分布，即 $z \sim \mathcal{N}(0, I)$ 。
2. 根据隐变量 z ，通过一个以神经网络 (解码器) 参数化的条件概率分布 $p_\theta(X|z)$ 来生成数据 X 。

我们的最终目标是最大化观测数据的边际似然 $p(X)$ ，即所有可能隐变量 z 生成 X 的概率的积分：

$$p(X) = \int p_\theta(X|z)p(z)dz \quad (5)$$

然而，这个积分通常是难以计算的 (intractable)，因为它需要在整个高维的隐空间上进行。这就引出了变分推断 (Variational Inference) 的必要性。

VAE 模型架构 VAE 由两个核心部分组成，通常由深度神经网络实现：

- **编码器 (Encoder)，或称推断网络 (Inference Network):**
 - 输入：数据样本 X 。
 - 输出：隐变量 z 的后验分布 $p(z|X)$ 的近似分布 $q_\phi(z|X)$ 的参数。这里， ϕ 是编码器网络的参数。具体地，编码器输出近似后验分布的参数，例如高斯分布的均值 μ_X 和对数方差 $\log(\sigma_X^2)$ 。即 $q_\phi(z|X) = \mathcal{N}(z; \mu_X, \sigma_X^2 I)$ 。
- **解码器 (Decoder)，或称生成网络 (Generative Network):**

- 输入：从分布 $q_\phi(z|X)$ 中采样的隐向量 z 。
- 输出：重构的数据 \hat{X} ，或者更准确地说，是条件概率分布 $p_\theta(X|z)$ 的参数。

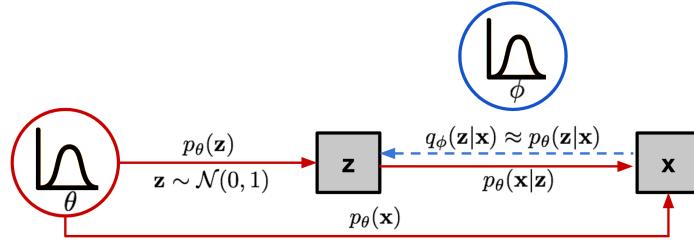


图 1-5 变分自动编码器中涉及的概率图模型

1.2.3 核心推导：证据下界 (Derivation of the Evidence Lower Bound, ELBO)

由于直接最大化 $p(X)$ 不可行，我们转而最大化它的一个下界，即证据下界 (**ELBO**)。我们从对数边际似然 $\log p(X)$ 开始，引入近似后验 $q_\phi(z|X)$ ，并根据詹森不等式 (Jensen's Inequality)，得到：

$$\begin{aligned} \log p(X) &\geq \int q_\phi(z|X) \log \frac{p_\theta(X|z)p(z)}{q_\phi(z|X)} dz \\ &= \mathbb{E}_{z \sim q_\phi(z|X)} \left[\log \frac{p_\theta(X|z)p(z)}{q_\phi(z|X)} \right] \end{aligned} \quad (6)$$

不等式的右侧就是证据下界 (ELBO)，记为 $\mathcal{L}(\phi, \theta; X)$ 。

Note 1.3 (ELBO 的分解).

$$\begin{aligned} \mathcal{L}(\phi, \theta; X) &= \mathbb{E}_{z \sim q_\phi(z|X)} [\log p_\theta(X|z) + \log p(z) - \log q_\phi(z|X)] \\ &= \underbrace{\mathbb{E}_{z \sim q_\phi(z|X)} [\log p_\theta(X|z)]}_{\text{重构项}} - \underbrace{D_{KL}(q_\phi(z|X)||p(z))}_{\text{KL 散度正则化项}} \end{aligned} \quad (7)$$

Note 1.4 (为什么使用反向 KL 散度？). 我们通过最小化 $D_{KL}(q_\phi(z|X)||p_\theta(z|X))$ 来逼近真实后验，这被称为反向 **KL 散度** (Reverse KL)。

- 前向 **KL** ($D_{KL}(P||Q)$): 要求近似分布 Q 覆盖真实分布 P 的所有非零区域。
- 反向 **KL** ($D_{KL}(Q||P)$): 倾向于将近似分布 Q “挤压”在真实分布 P 的峰值下方。这使得 q_ϕ 能够捕捉到 p_θ 的主要模式 (Mode-seeking behavior)，对于生成任务更为稳健。

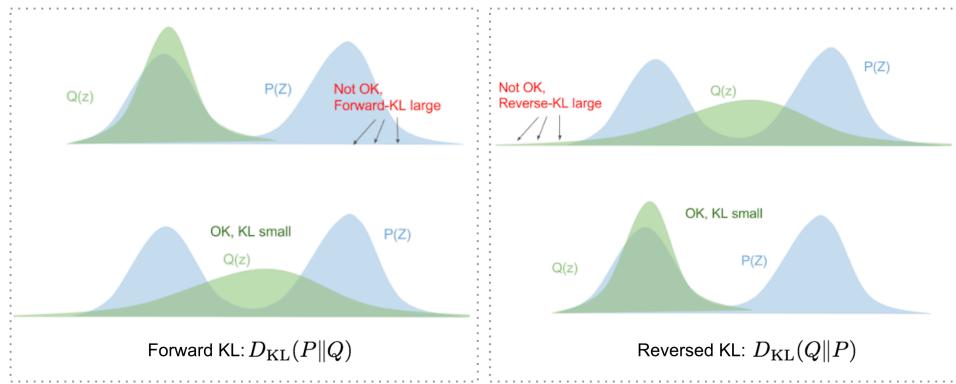


图 1-6 前向和反向 KL 散度对匹配两个分布有不同的要求

Note 1.5 (目标函数解读). VAE 的训练目标是最大化 ELBO，它由两部分构成：

- **重构项 (Reconstruction Term):** 该项最大化在给定隐变量 z 的情况下，重构出原始数据 X 的对数似然。这驱动解码器学习如何精确地从 z 恢复 X 。在实践中，常使用均方误差 (MSE) 或二元交叉熵 (BCE) 损失的负值。
- **KL 散度正则化项 (KL Divergence Regularization Term):** 该项衡量近似后验分布 $q_\phi(z|X)$ 与先验分布 $p(z)$ (通常是 $\mathcal{N}(0, I)$) 之间的差异。最小化 KL 散度，相当于对编码器施加约束，使其产生的隐变量分布接近标准正态分布，从而使隐空间变得平滑和连续。

因此，训练目标可以概括为：最大化重构质量的同时，保持隐空间分布的规整性。

1.2.4 关键技术：重参数化技巧 (The Reparameterization Trick)

Definition 1.3 (重参数化技巧). 在目标函数中，从 $q_\phi(z|X)$ 中采样 z 的过程是随机的，不可微分，这会阻碍梯度的反向传播。重参数化技巧通过将随机性与模型参数分离来解决此问题。对于高斯分布 $z \sim \mathcal{N}(z; \mu_X, \sigma_X^2 I)$ ，其采样过程可重写为：

1. 从一个固定的标准正态分布中采样噪声 $\epsilon \sim \mathcal{N}(0, I)$ 。
2. 通过确定性变换生成 z : $z = \mu_X + \sigma_X \odot \epsilon$ 。

这样，随机采样被移出计算图，梯度可以顺利地从损失函数回传到编码器的参数 ϕ (即 μ_X 和 σ_X)。

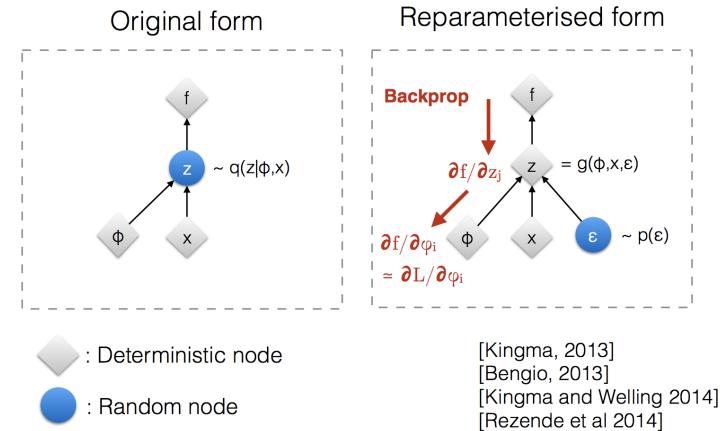


图 1-7 重参数化技巧示意图

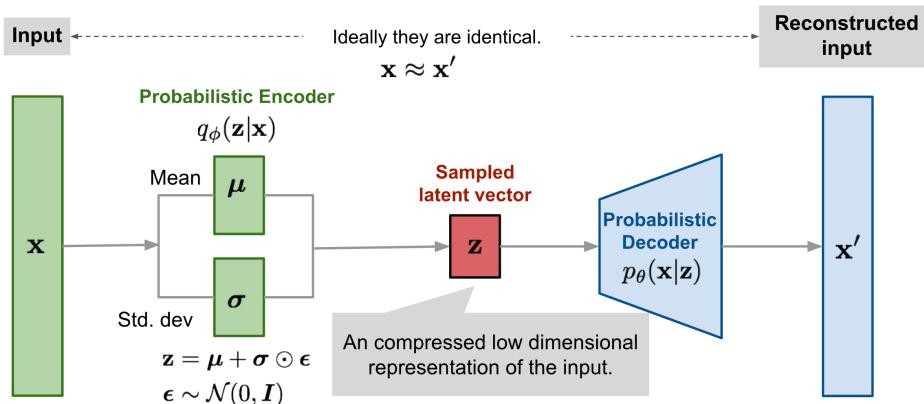


图 1-8 假设多元高斯分布的变分自动编码器模型示意图

1.2.5 结论与讨论 (Conclusion and Discussion)

变分自编码器 (VAE) 通过最大化证据下界 (ELBO) 成功地将深度神经网络与概率图模型结合起来。其损失函数由重构损失和 KL 散度正则化项构成，分别保证了模型的重构能力和隐空间的良好结构。关键的重参数化技巧使得模型能够通过标准的梯度下降算法进行端到端的训练。

Note 1.6 (VAE vs. GAN). 与 GAN 相比，VAE 的训练过程更稳定，能够显式地学习数据的概率密度。然而，VAE 生成的样本通常比 GAN 生成的样本模糊，这是因为其损失函数倾向于覆盖数据的所有模式，导致在细节上有所妥协。

尽管如此，VAE 作为一种 foundational 的深度生成模型，其思想，特别是对隐空间的概率建模，对后续的生成模型（包括扩散模型）产生了深远的影响。理解 VAE 是掌握现代生成式 AI 版图不可或缺的一步。

1.3 进阶变分模型 (Advanced VAE Models)

1.3.1 Beta-VAE (β -VAE)

Definition 1.4 (解耦表示 (Disentangled Representation)). 如果隐变量 \mathbf{z} 中的每个维度仅对数据的一个生成因子敏感 (例如一个维度控制 “发色”，另一个控制 “肤色”)，且对其他因子保持不变，则称这种表示为解耦的。解耦表示具有极佳的可解释性。

β -VAE 旨在通过调整 ELBO 中的 KL 散度项权重来发现解耦的潜在因子。无论是从直觉上增加 KL 惩罚，还是从约束优化的角度来看，都有坚实的理论基础。我们希望最大化生成概率，同时约束后验分布与先验分布的距离小于 δ ：

$$\max_{\phi, \theta} \mathbb{E}_{\mathbf{x} \sim \mathcal{D}} [\mathbb{E}_{\mathbf{z} \sim q_\phi(\mathbf{z}|\mathbf{x})} \log p_\theta(\mathbf{x}|\mathbf{z})] \quad \text{subject to } D_{\text{KL}}(q_\phi(\mathbf{z}|\mathbf{x}) \| p_\theta(\mathbf{z})) < \delta \quad (8)$$

使用拉格朗日乘数法，这等价于最大化：

$$\mathcal{F}(\theta, \phi, \beta) = \mathbb{E}_{\mathbf{z} \sim q_\phi(\mathbf{z}|\mathbf{x})} \log p_\theta(\mathbf{x}|\mathbf{z}) - \beta(D_{\text{KL}}(q_\phi(\mathbf{z}|\mathbf{x}) \| p_\theta(\mathbf{z})) - \delta) \quad (9)$$

去掉常数项，得到 β -VAE 的损失函数：

$$L_{\beta\text{-VAE}} = -\mathbb{E}_{\mathbf{z} \sim q_\phi(\mathbf{z}|\mathbf{x})} [\log p_\theta(\mathbf{x}|\mathbf{z})] + \beta D_{\text{KL}}(q_\phi(\mathbf{z}|\mathbf{x}) \| p_\theta(\mathbf{z})) \quad (10)$$

Note 1.7 (β 的作用).

- 当 $\beta = 1$ 时，模型退化为标准的 VAE。

- 当 $\beta > 1$ 时，模型对隐变量施加了更强的独立性约束，鼓励学习更有效的、解耦的潜在编码。但这通常会带来重构质量与解耦程度之间的权衡 (Trade-off)。

1.3.2 VQ-VAE (Vector Quantized VAE)

传统的 VAE 使用连续的隐变量分布 (如高斯分布)，这可能不适合某些模态 (如语言、推理)。VQ-VAE 引入了离散潜在变量。

Note 1.8 (向量量化 (Vector Quantization)). VQ-VAE 维护一个“代码簿” (Codebook)，包含 K 个嵌入向量 \mathbf{e}_i 。编码器的输出 $\mathbf{z}_e(\mathbf{x})$ 不再直接传给解码器，而是通过最近邻搜索映射到代码簿中最接近的嵌入向量 \mathbf{e}_k ：

$$\mathbf{z}_q(\mathbf{x}) = \mathbf{e}_k, \quad \text{where } k = \arg \min_i \|\mathbf{z}_e(\mathbf{x}) - \mathbf{e}_i\|_2 \quad (11)$$

由于 $\arg \min$ 操作不可导，VQ-VAE 使用直通估计器 (Straight-Through Estimator) 将解码器的梯度直接复制回编码器 (即 $\nabla_z L \approx \nabla_{z_q} L$)。其总损失函数如下 (其中 $\text{sg}[\cdot]$ 表示这个梯度的截断算子 stop gradient)：

$$L = \underbrace{\|\mathbf{x} - D(\mathbf{e}_k)\|_2^2}_{\text{Reconstruction Loss}} + \underbrace{\|\text{sg}[E(\mathbf{x})] - \mathbf{e}_k\|_2^2}_{\text{VQ Loss}} + \underbrace{\beta \|E(\mathbf{x}) - \text{sg}[\mathbf{e}_k]\|_2^2}_{\text{Commitment Loss}} \quad (12)$$

代码簿更新 (EMA): 为了更稳定地更新嵌入向量，可以使用指数移动平均 (Exponential Moving Average)：

$$N_i^{(t)} = \gamma N_i^{(t-1)} + (1 - \gamma)n_i^{(t)}, \quad \mathbf{m}_i^{(t)} = \gamma \mathbf{m}_i^{(t-1)} + (1 - \gamma) \sum_j \mathbf{z}_{i,j}^{(t)}, \quad \mathbf{e}_i^{(t)} = \frac{\mathbf{m}_i^{(t)}}{N_i^{(t)}} \quad (13)$$

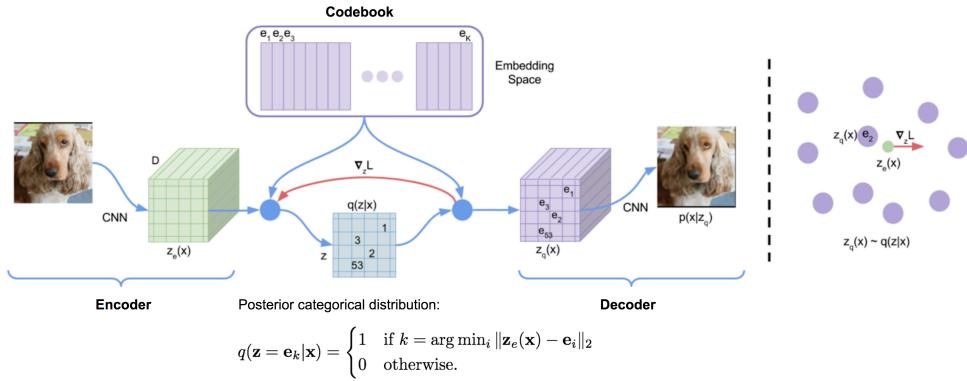


图 1-9 VQ-VAE 的架构

VQ-VAE-2 进一步引入了层级结构 (Hierarchical VQ-VAE)，分离局部纹理和全局形状信息，并结合自回归模型（如 PixelCNN）来学习离散编码的先验分布，生成了极高质量的图像。

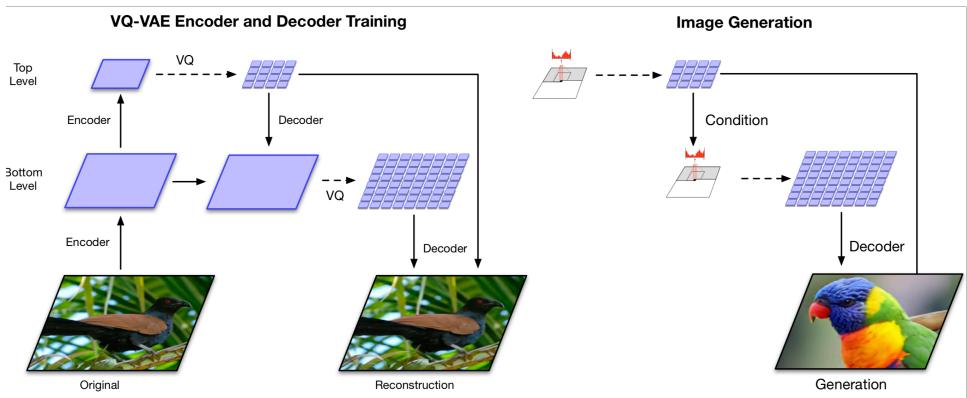


图 1-10 层次 VQ-VAE 和多阶段图像生成的架构

1.3.3 TD-VAE (Temporal Difference VAE)

TD-VAE 专为序列数据设计，旨在解决传统序列模型推断效率低的问题。它基于三个核心思想：

- 状态空间模型 (State Space Model):** 假设观测序列由一系列不可见的隐状态控制。
- 信念状态 (Belief State):** 智能体应学会将过去的所有信息编码为一个信念状态 b_t ，用于推理未来。

3. 跳跃预测 (Jump Prediction): 模型应具备根据当前信息直接预测遥远未来的能力，而无需一步步模拟中间过程。

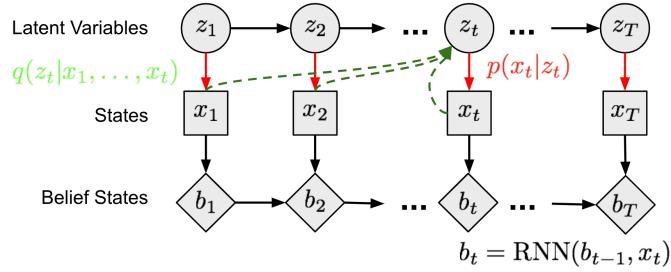


图 1-11 作为马尔可夫链模型的状态空间模型

TD-VAE 需要学习四种分布：

- 解码器分布 $p_D(x_t|z_t)$: 从隐状态生成观测。
- 转移分布 $p_T(z_t|z_{t-1})$: 捕捉隐变量的序列依赖。
- 信念分布 $p_B(z_t|b_t)$: 从信念状态推断隐变量。
- 平滑分布 $p_S(z_{t-1}|z_t, b_{t-1}, b_t)$: 回溯推断。

TD-VAE 的最终目标函数是在两个时间戳 $t_1 < t_2$ 之间最大化以下期望 (Jump Prediction) :

$$J_{t_1, t_2} = \mathbb{E}[\log p_D(x_{t_2}|z_{t_2}) + \log p_B(z_{t_1}|b_{t_1}) + \log p_T(z_{t_2}|z_{t_1}) - \log p_B(z_{t_2}|b_{t_2}) - \log p_S(z_{t_1}|z_{t_2}, b_{t_1}, b_{t_2})] \quad (14)$$

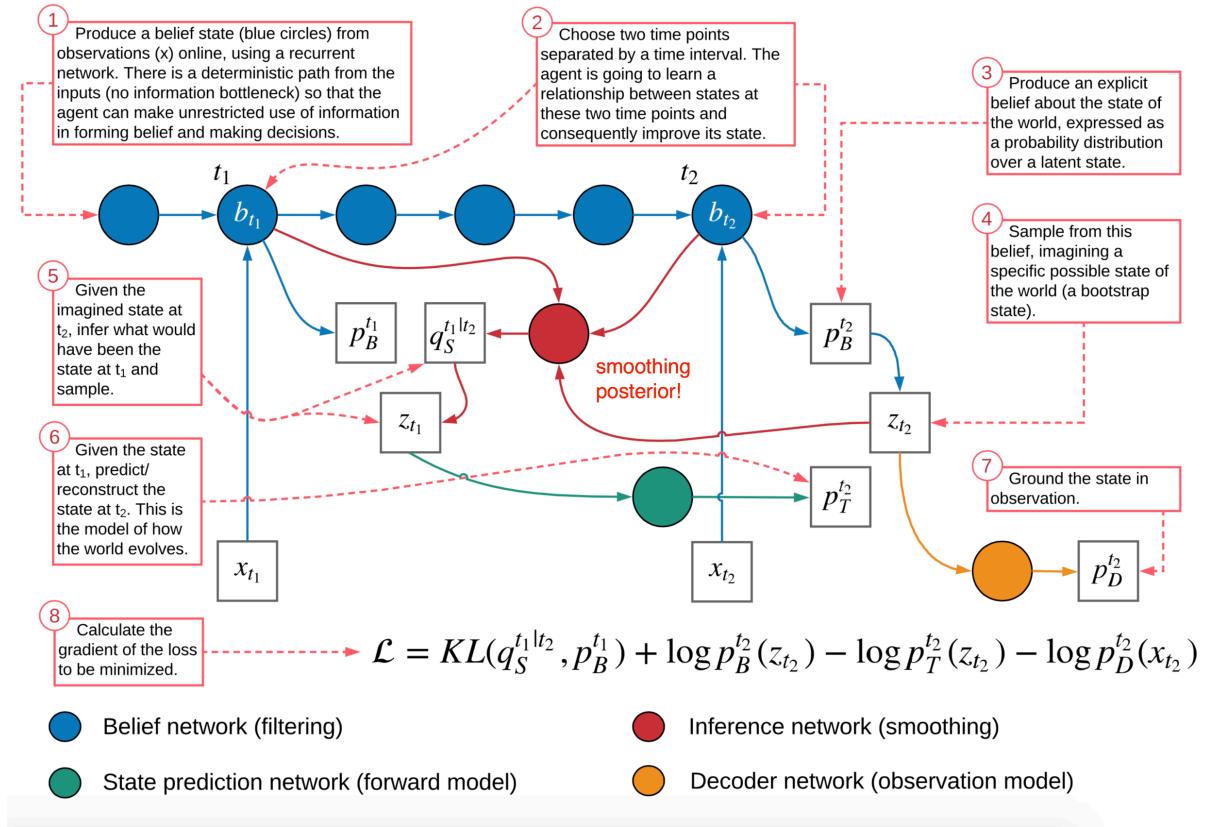


图 1-12 TD-VAE 架构详细概览

2 扩散模型 (Diffusion Models)

符号说明 (Notation)

符号	含义
\mathbf{x}_0	真实数据样本
\mathbf{x}_t	时间步 t 的噪声样本
T	总扩散步数
β_t	方差调度表, 控制加噪量
α_t	$1 - \beta_t$
$\bar{\alpha}_t$	$\prod_{i=1}^t \alpha_i$, 累积乘积
ϵ	标准高斯噪声, $\mathcal{N}(\mathbf{0}, \mathbf{I})$
ϵ_θ	噪声预测网络 (或分数网络)
$q(\mathbf{x}_t \mathbf{x}_{t-1})$	前向扩散过程 (加噪)
$p_\theta(\mathbf{x}_{t-1} \mathbf{x}_t)$	逆向扩散过程 (去噪)
L_{simple}	简化的均方误差损失

迄今为止, 必须要提到的生成模型有三类: GAN、VAE 和基于流的模型 (Flow-based models)。它们在生成高质量样本方面取得了巨大成功, 但各有限制。GAN 因其对抗性

训练性质，训练可能不稳定且生成多样性较差；VAE 依赖于代理损失；流模型则必须使用专门的架构来构建可逆变换。

扩散模型受非平衡热力学启发。它们定义了一个扩散步骤的马尔可夫链，通过缓慢地向数据添加随机噪声，然后学习逆向扩散过程，从噪声中构建所需的数据样本。与 VAE 或流模型不同，扩散模型是通过固定的过程学习的，并且隐变量具有高维度（与原始数据相同）。

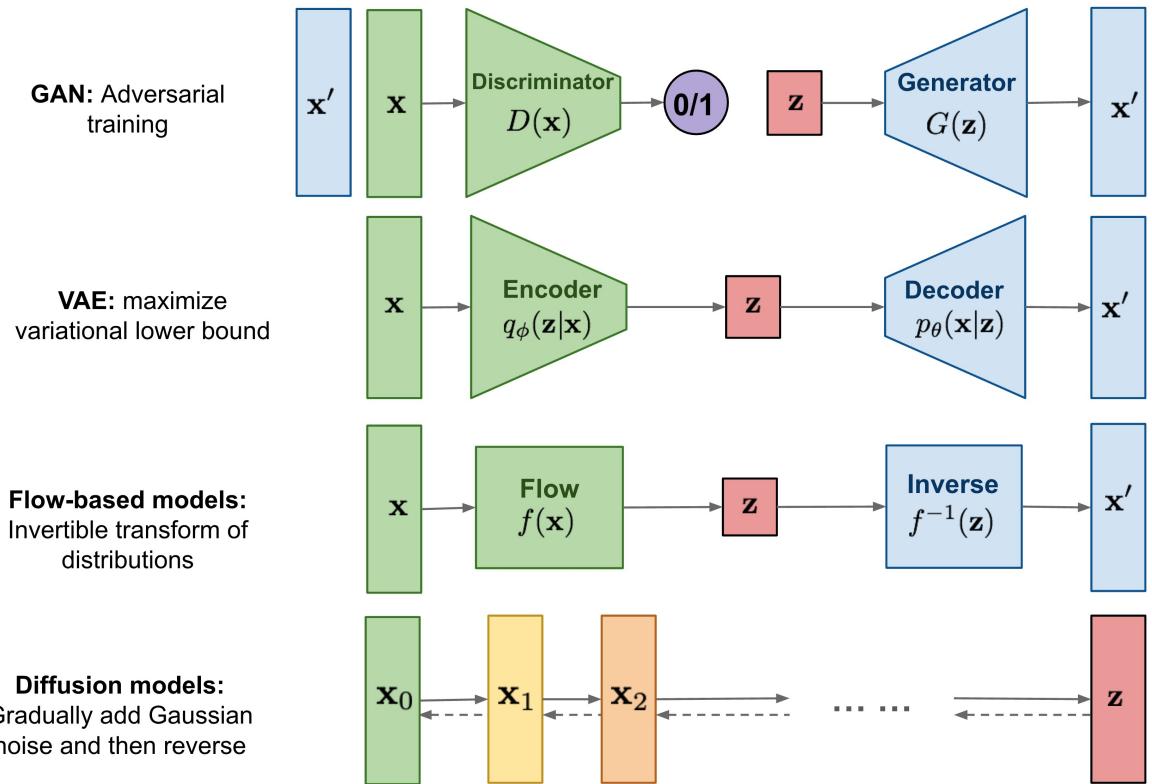


图 2-1 不同类型生成模型的概览：GAN、VAE、Flow-based 和 Diffusion

2.1 基本原理

包括扩散概率模型 (Sohl-Dickstein et al., 2015)、噪声条件分数网络 (NCSN; Yang & Ermon, 2019) 和去噪扩散概率模型 (DDPM; Ho et al. 2020) 在内的多种基于扩散的生成模型，其底层思想是相似的。

2.1.1 前向扩散过程 (Forward Diffusion Process)

给定从真实数据分布中采样的数据点 $\mathbf{x}_0 \sim q(\mathbf{x})$ ，我们定义一个前向扩散过程，在 T 个步骤中向样本添加少量高斯噪声，产生一系列噪声样本 $\mathbf{x}_1, \dots, \mathbf{x}_T$ 。步长由方差调度 $\{\beta_t \in (0, 1)\}$ 控制：

$$q(\mathbf{x}_t | \mathbf{x}_{t-1}) = \mathcal{N}(\mathbf{x}_t; \sqrt{1 - \beta_t} \mathbf{x}_{t-1}, \beta_t \mathbf{I}) \quad q(\mathbf{x}_{1:T} | \mathbf{x}_0) = \prod_{t=1}^T q(\mathbf{x}_t | \mathbf{x}_{t-1}) \quad (15)$$

随着步长 t 变大，数据样本 \mathbf{x}_0 逐渐失去其可辨别的特征。最终当 $T \rightarrow \infty$ 时， \mathbf{x}_T 等价于各向同性高斯分布。

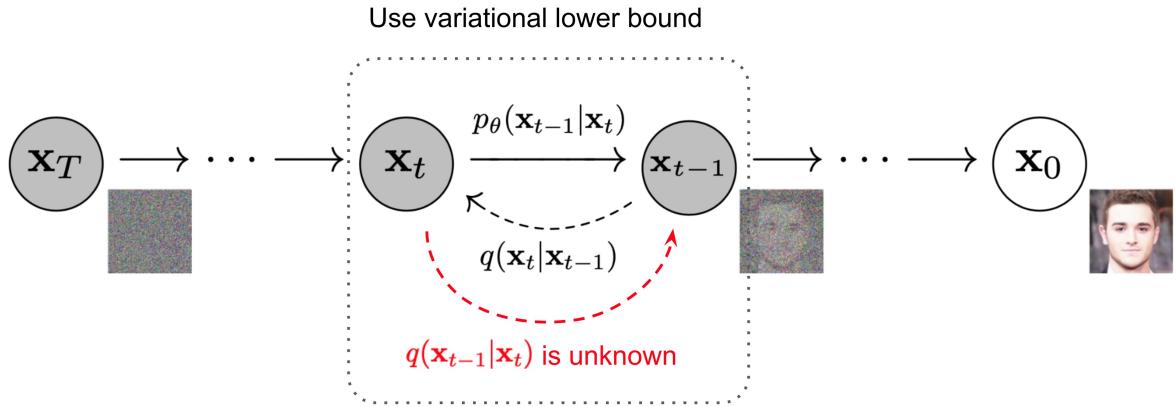


图 2-2 前向（逆向）扩散过程的马尔可夫链，通过缓慢添加（去除）噪声来生成样本

上述过程的一个优良特性是，我们可以使用重参数化技巧以闭式解在任意时间步 t 对 \mathbf{x}_t 进行采样。令 $\alpha_t = 1 - \beta_t$ 且 $\bar{\alpha}_t = \prod_{i=1}^t \alpha_i$ ：

$$\begin{aligned}\mathbf{x}_t &= \sqrt{\alpha_t} \mathbf{x}_{t-1} + \sqrt{1 - \alpha_t} \boldsymbol{\epsilon}_{t-1} && ; \text{where } \boldsymbol{\epsilon}_{t-1}, \boldsymbol{\epsilon}_{t-2}, \dots \sim \mathcal{N}(\mathbf{0}, \mathbf{I}) \\ &= \sqrt{\alpha_t \alpha_{t-1}} \mathbf{x}_{t-2} + \sqrt{1 - \alpha_t \alpha_{t-1}} \bar{\boldsymbol{\epsilon}}_{t-2} && ; \text{where } \bar{\boldsymbol{\epsilon}}_{t-2} \text{ merges two Gaussians (*).} \\ &= \dots \\ &= \sqrt{\bar{\alpha}_t} \mathbf{x}_0 + \sqrt{1 - \bar{\alpha}_t} \boldsymbol{\epsilon}\end{aligned}\tag{16}$$

也就是：

$$q(\mathbf{x}_t|\mathbf{x}_0) = \mathcal{N}(\mathbf{x}_t; \sqrt{\bar{\alpha}_t} \mathbf{x}_0, (1 - \bar{\alpha}_t) \mathbf{I})\tag{17}$$

(*) 提示：当我们合并两个具有不同方差的高斯分布 $\mathcal{N}(\mathbf{0}, \sigma_1^2 \mathbf{I})$ 和 $\mathcal{N}(\mathbf{0}, \sigma_2^2 \mathbf{I})$ 时，新的分布是 $\mathcal{N}(\mathbf{0}, (\sigma_1^2 + \sigma_2^2) \mathbf{I})$ 。在这里合并的标准差是 $\sqrt{(1 - \alpha_t) + \alpha_t(1 - \alpha_{t-1})} = \sqrt{1 - \alpha_t \alpha_{t-1}}$ 。通常，当样本变得更嘈杂时，我们可以承受更大的更新步长，所以 $\beta_1 < \beta_2 < \dots < \beta_T$ ，因此 $\bar{\alpha}_1 > \dots > \bar{\alpha}_T$ 。

2.1.2 逆向扩散过程 (Reverse Diffusion Process)

如果我们能够逆转上述过程并从 $q(\mathbf{x}_{t-1}|\mathbf{x}_t)$ 中采样，我们就能够从高斯噪声输入 $\mathbf{x}_T \sim \mathcal{N}(\mathbf{0}, \mathbf{I})$ 重建真实样本。注意，如果 β_t 足够小， $q(\mathbf{x}_{t-1}|\mathbf{x}_t)$ 也将是高斯分布。遗憾的是，我们无法轻易估计 $q(\mathbf{x}_{t-1}|\mathbf{x}_t)$ ，因为它需要使用整个数据集，因此我们需要学习一个模型 p_θ 来近似这些条件概率，以便运行逆向扩散过程。

$$p_\theta(\mathbf{x}_{0:T}) = p(\mathbf{x}_T) \prod_{t=1}^T p_\theta(\mathbf{x}_{t-1}|\mathbf{x}_t) \quad p_\theta(\mathbf{x}_{t-1}|\mathbf{x}_t) = \mathcal{N}(\mathbf{x}_{t-1}; \boldsymbol{\mu}_\theta(\mathbf{x}_t, t), \boldsymbol{\Sigma}_\theta(\mathbf{x}_t, t))\tag{18}$$

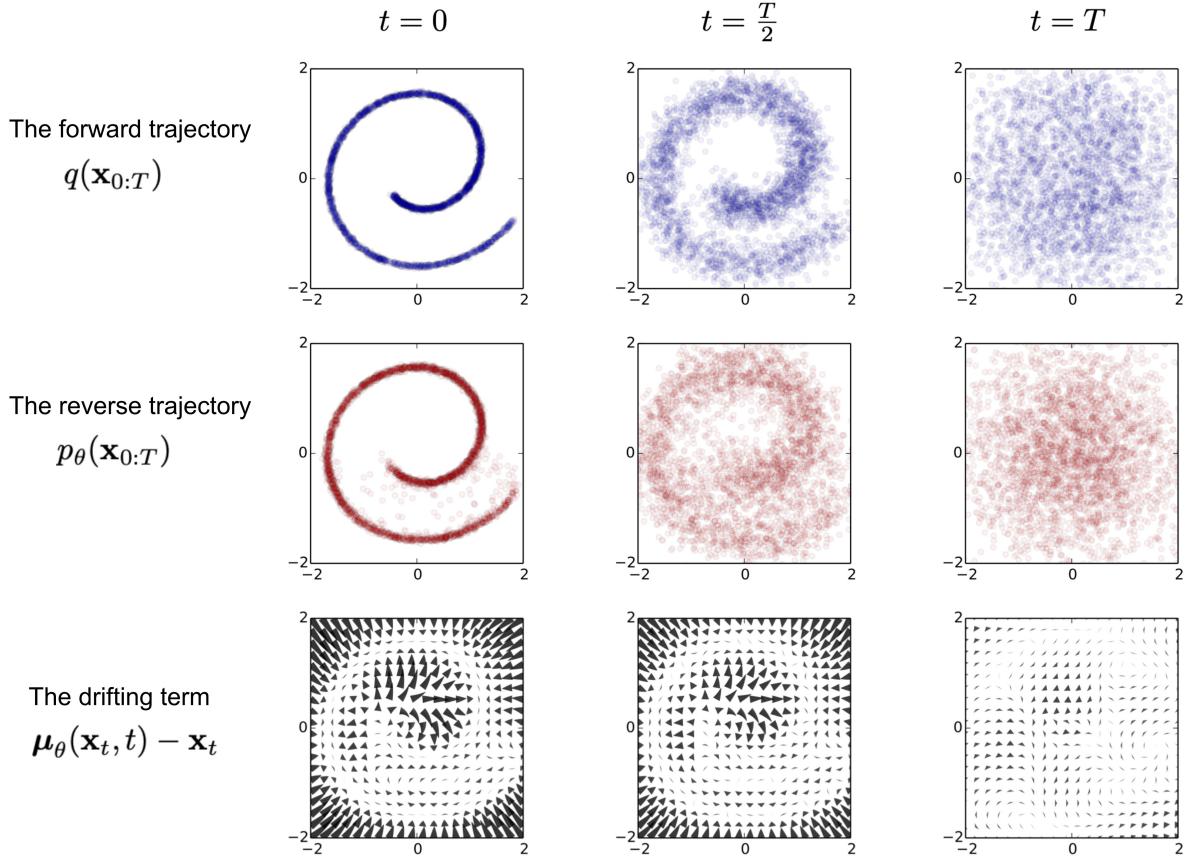


图 2-3 训练扩散模型以建模 2D 瑞士卷数据的示例

值得注意的是，当以 \mathbf{x}_0 为条件时，逆向条件概率是可计算的：

$$q(\mathbf{x}_{t-1}|\mathbf{x}_t, \mathbf{x}_0) = \mathcal{N}(\mathbf{x}_{t-1}; \tilde{\mu}(\mathbf{x}_t, \mathbf{x}_0), \tilde{\beta}_t \mathbf{I}) \quad (19)$$

使用贝叶斯规则：

$$\begin{aligned} q(\mathbf{x}_{t-1}|\mathbf{x}_t, \mathbf{x}_0) &= q(\mathbf{x}_t|\mathbf{x}_{t-1}, \mathbf{x}_0) \frac{q(\mathbf{x}_{t-1}|\mathbf{x}_0)}{q(\mathbf{x}_t|\mathbf{x}_0)} \\ &\propto \exp\left(-\frac{1}{2}\left(\frac{(\mathbf{x}_t - \sqrt{\alpha_t}\mathbf{x}_{t-1})^2}{\beta_t} + \frac{(\mathbf{x}_{t-1} - \sqrt{\bar{\alpha}_{t-1}}\mathbf{x}_0)^2}{1-\bar{\alpha}_{t-1}} - \frac{(\mathbf{x}_t - \sqrt{\bar{\alpha}_t}\mathbf{x}_0)^2}{1-\bar{\alpha}_t}\right)\right) \\ &= \exp\left(-\frac{1}{2}\left(\frac{\mathbf{x}_t^2 - 2\sqrt{\alpha_t}\mathbf{x}_t\mathbf{x}_{t-1} + \alpha_t\mathbf{x}_{t-1}^2}{\beta_t} + \frac{\mathbf{x}_{t-1}^2 - 2\sqrt{\bar{\alpha}_{t-1}}\mathbf{x}_0\mathbf{x}_{t-1} + \bar{\alpha}_{t-1}\mathbf{x}_0^2}{1-\bar{\alpha}_{t-1}} - C(\mathbf{x}_t, \mathbf{x}_0)\right)\right) \\ &= \exp\left(-\frac{1}{2}\left(\left(\frac{\alpha_t}{\beta_t} + \frac{1}{1-\bar{\alpha}_{t-1}}\right)\mathbf{x}_{t-1}^2 - \left(\frac{2\sqrt{\alpha_t}}{\beta_t}\mathbf{x}_t + \frac{2\sqrt{\bar{\alpha}_{t-1}}}{1-\bar{\alpha}_{t-1}}\mathbf{x}_0\right)\mathbf{x}_{t-1} + C'(\mathbf{x}_t, \mathbf{x}_0)\right)\right) \end{aligned} \quad (20)$$

其中 $C(\mathbf{x}_t, \mathbf{x}_0)$ 和 $C'(\mathbf{x}_t, \mathbf{x}_0)$ 是不包含 \mathbf{x}_{t-1} 的项。

根据标准高斯密度函数，我们可以得到均值和方差的参数化形式：

$$\tilde{\beta}_t = 1/\left(\frac{\alpha_t}{\beta_t} + \frac{1}{1-\bar{\alpha}_{t-1}}\right) = \frac{1-\bar{\alpha}_{t-1}}{1-\bar{\alpha}_t} \cdot \beta_t \quad (21)$$

以及：

$$\begin{aligned}
\tilde{\mu}_t(\mathbf{x}_t, \mathbf{x}_0) &= \left(\frac{\sqrt{\alpha_t}}{\beta_t} \mathbf{x}_t + \frac{\sqrt{\bar{\alpha}_{t-1}}}{1 - \bar{\alpha}_{t-1}} \mathbf{x}_0 \right) / \left(\frac{\alpha_t}{\beta_t} + \frac{1}{1 - \bar{\alpha}_{t-1}} \right) \\
&= \left(\frac{\sqrt{\alpha_t}}{\beta_t} \mathbf{x}_t + \frac{\sqrt{\bar{\alpha}_{t-1}}}{1 - \bar{\alpha}_{t-1}} \mathbf{x}_0 \right) \frac{1 - \bar{\alpha}_{t-1}}{1 - \bar{\alpha}_t} \cdot \beta_t \\
&= \frac{\sqrt{\alpha_t}(1 - \bar{\alpha}_{t-1})}{1 - \bar{\alpha}_t} \mathbf{x}_t + \frac{\sqrt{\bar{\alpha}_{t-1}}\beta_t}{1 - \bar{\alpha}_t} \mathbf{x}_0
\end{aligned} \tag{22}$$

利用 $\mathbf{x}_0 = \frac{1}{\sqrt{\bar{\alpha}_t}} (\mathbf{x}_t - \sqrt{1 - \bar{\alpha}_t} \epsilon_t)$, 我们可以将均值表示为仅依赖于 \mathbf{x}_t 和噪声 ϵ_t 的形式：

$$\tilde{\mu}_t = \frac{1}{\sqrt{\alpha_t}} \left(\mathbf{x}_t - \frac{1 - \alpha_t}{\sqrt{1 - \bar{\alpha}_t}} \epsilon_t \right) \tag{23}$$

2.1.3 训练目标

这与 VAE 非常相似, 因此我们可以使用变分下界 (VLLB) 来优化负对数似然:

$$\begin{aligned}
-\log p_\theta(\mathbf{x}_0) &\leq -\log p_\theta(\mathbf{x}_0) + D_{\text{KL}}(q(\mathbf{x}_{1:T}|\mathbf{x}_0) \| p_\theta(\mathbf{x}_{1:T}|\mathbf{x}_0)) \\
&= -\log p_\theta(\mathbf{x}_0) + \mathbb{E}_{\mathbf{x}_{1:T} \sim q(\mathbf{x}_{1:T}|\mathbf{x}_0)} \left[\log \frac{q(\mathbf{x}_{1:T}|\mathbf{x}_0)}{p_\theta(\mathbf{x}_{0:T})/p_\theta(\mathbf{x}_0)} \right] \\
&= -\log p_\theta(\mathbf{x}_0) + \mathbb{E}_q \left[\log \frac{q(\mathbf{x}_{1:T}|\mathbf{x}_0)}{p_\theta(\mathbf{x}_{0:T})} + \log p_\theta(\mathbf{x}_0) \right] \\
&= \mathbb{E}_q \left[\log \frac{q(\mathbf{x}_{1:T}|\mathbf{x}_0)}{p_\theta(\mathbf{x}_{0:T})} \right] = L_{\text{VLLB}}
\end{aligned} \tag{24}$$

可以将 L_{VLLB} 进一步重写为多个 KL 散度项的组合:

$$\begin{aligned}
L_{\text{VLB}} &= \mathbb{E}_{q(\mathbf{x}_{0:T})} \left[\log \frac{q(\mathbf{x}_{1:T} | \mathbf{x}_0)}{p_\theta(\mathbf{x}_{0:T})} \right] \\
&= \mathbb{E}_q \left[\log \frac{\prod_{t=1}^T q(\mathbf{x}_t | \mathbf{x}_{t-1})}{p_\theta(\mathbf{x}_T) \prod_{t=1}^T p_\theta(\mathbf{x}_{t-1} | \mathbf{x}_t)} \right] \\
&= \mathbb{E}_q \left[-\log p_\theta(\mathbf{x}_T) + \sum_{t=1}^T \log \frac{q(\mathbf{x}_t | \mathbf{x}_{t-1})}{p_\theta(\mathbf{x}_{t-1} | \mathbf{x}_t)} \right] \\
&= \mathbb{E}_q \left[-\log p_\theta(\mathbf{x}_T) + \sum_{t=2}^T \log \frac{q(\mathbf{x}_t | \mathbf{x}_{t-1})}{p_\theta(\mathbf{x}_{t-1} | \mathbf{x}_t)} + \log \frac{q(\mathbf{x}_1 | \mathbf{x}_0)}{p_\theta(\mathbf{x}_0 | \mathbf{x}_1)} \right] \\
&= \mathbb{E}_q \left[-\log p_\theta(\mathbf{x}_T) + \sum_{t=2}^T \log \left(\frac{q(\mathbf{x}_{t-1} | \mathbf{x}_t, \mathbf{x}_0)}{p_\theta(\mathbf{x}_{t-1} | \mathbf{x}_t)} \cdot \frac{q(\mathbf{x}_t | \mathbf{x}_0)}{q(\mathbf{x}_{t-1} | \mathbf{x}_0)} \right) + \log \frac{q(\mathbf{x}_1 | \mathbf{x}_0)}{p_\theta(\mathbf{x}_0 | \mathbf{x}_1)} \right] \\
&= \mathbb{E}_q \left[-\log p_\theta(\mathbf{x}_T) + \sum_{t=2}^T \log \frac{q(\mathbf{x}_{t-1} | \mathbf{x}_t, \mathbf{x}_0)}{p_\theta(\mathbf{x}_{t-1} | \mathbf{x}_t)} + \sum_{t=2}^T \log \frac{q(\mathbf{x}_t | \mathbf{x}_0)}{q(\mathbf{x}_{t-1} | \mathbf{x}_0)} + \log \frac{q(\mathbf{x}_1 | \mathbf{x}_0)}{p_\theta(\mathbf{x}_0 | \mathbf{x}_1)} \right] \\
&= \mathbb{E}_q \left[-\log p_\theta(\mathbf{x}_T) + \sum_{t=2}^T \log \frac{q(\mathbf{x}_{t-1} | \mathbf{x}_t, \mathbf{x}_0)}{p_\theta(\mathbf{x}_{t-1} | \mathbf{x}_t)} + \log \frac{q(\mathbf{x}_T | \mathbf{x}_0)}{q(\mathbf{x}_1 | \mathbf{x}_0)} + \log \frac{q(\mathbf{x}_1 | \mathbf{x}_0)}{p_\theta(\mathbf{x}_0 | \mathbf{x}_1)} \right] \\
&= \mathbb{E}_q \left[\log \frac{q(\mathbf{x}_T | \mathbf{x}_0)}{p_\theta(\mathbf{x}_T)} + \sum_{t=2}^T \log \frac{q(\mathbf{x}_{t-1} | \mathbf{x}_t, \mathbf{x}_0)}{p_\theta(\mathbf{x}_{t-1} | \mathbf{x}_t)} - \log p_\theta(\mathbf{x}_0 | \mathbf{x}_1) \right] \\
&= \mathbb{E}_q \left[\underbrace{D_{\text{KL}}(q(\mathbf{x}_T | \mathbf{x}_0) \| p_\theta(\mathbf{x}_T))}_{L_T} + \underbrace{\sum_{t=2}^T D_{\text{KL}}(q(\mathbf{x}_{t-1} | \mathbf{x}_t, \mathbf{x}_0) \| p_\theta(\mathbf{x}_{t-1} | \mathbf{x}_t))}_{L_{T-1+\dots+L_1}} - \underbrace{\log p_\theta(\mathbf{x}_0 | \mathbf{x}_1)}_{L_0} \right]
\end{aligned} \tag{25}$$

每一项标记如下：

$$L_{\text{VLB}} = L_T + L_{T-1} + \dots + L_0 \tag{26}$$

其中：

- L_T 是常数（因为 q 无参数且 \mathbf{x}_T 是高斯噪声），训练时可忽略。
- L_t 是两个高斯分布之间的 KL 散度，可有闭式解。
- L_0 通常使用独立的离散解码器来建模。

2.1.4 训练损失 L_t 的参数化

我们需要学习一个神经网络来近似逆向扩散过程中的条件概率分布 $p_\theta(\mathbf{x}_{t-1} | \mathbf{x}_t) = \mathcal{N}(\mathbf{x}_{t-1}; \boldsymbol{\mu}_\theta(\mathbf{x}_t, t), \boldsymbol{\Sigma}_\theta(\mathbf{x}_t, t))$ 。我们希望训练 $\boldsymbol{\mu}_\theta$ 来预测 $\tilde{\boldsymbol{\mu}}_t = \frac{1}{\sqrt{\alpha_t}}(\mathbf{x}_t - \frac{1-\alpha_t}{\sqrt{1-\tilde{\alpha}_t}}\boldsymbol{\epsilon}_t)$ 。因为 \mathbf{x}_t 在训练时是作为输入可用的，我们可以改为重参数化高斯噪声项，使其根据输入 \mathbf{x}_t 在时间步 t 预测 $\boldsymbol{\epsilon}_t$ ：

$$\mu_\theta(\mathbf{x}_t, t) = \frac{1}{\sqrt{\alpha_t}} \left(\mathbf{x}_t - \frac{1 - \alpha_t}{\sqrt{1 - \bar{\alpha}_t}} \epsilon_\theta(\mathbf{x}_t, t) \right) \quad (27)$$

因此 \mathbf{x}_{t-1} 的采样过程变为：

$$\mathbf{x}_{t-1} = \frac{1}{\sqrt{\alpha_t}} \left(\mathbf{x}_t - \frac{1 - \alpha_t}{\sqrt{1 - \bar{\alpha}_t}} \epsilon_\theta(\mathbf{x}_t, t) \right) + \Sigma_\theta(\mathbf{x}_t, t)^{1/2} \mathbf{z} \quad (28)$$

损失项 L_t 被参数化为最小化与 $\tilde{\mu}_t$ 的差异：

$$\begin{aligned} L_t &= \mathbb{E}_{\mathbf{x}_0, \epsilon} \left[\frac{1}{2 \|\Sigma_\theta(\mathbf{x}_t, t)\|_2^2} \|\tilde{\mu}_t(\mathbf{x}_t, \mathbf{x}_0) - \mu_\theta(\mathbf{x}_t, t)\|^2 \right] \\ &= \mathbb{E}_{\mathbf{x}_0, \epsilon} \left[\frac{1}{2 \|\Sigma_\theta\|_2^2} \left\| \frac{1}{\sqrt{\alpha_t}} \left(\mathbf{x}_t - \frac{1 - \alpha_t}{\sqrt{1 - \bar{\alpha}_t}} \epsilon_t \right) - \frac{1}{\sqrt{\alpha_t}} \left(\mathbf{x}_t - \frac{1 - \alpha_t}{\sqrt{1 - \bar{\alpha}_t}} \epsilon_\theta(\mathbf{x}_t, t) \right) \right\|^2 \right] \\ &= \mathbb{E}_{\mathbf{x}_0, \epsilon} \left[\frac{(1 - \alpha_t)^2}{2\alpha_t(1 - \bar{\alpha}_t) \|\Sigma_\theta\|_2^2} \|\epsilon_t - \epsilon_\theta(\mathbf{x}_t, t)\|^2 \right] \\ &= \mathbb{E}_{\mathbf{x}_0, \epsilon} \left[\frac{(1 - \alpha_t)^2}{2\alpha_t(1 - \bar{\alpha}_t) \|\Sigma_\theta\|_2^2} \left\| \epsilon_t - \epsilon_\theta(\sqrt{\bar{\alpha}_t} \mathbf{x}_0 + \sqrt{1 - \bar{\alpha}_t} \epsilon_t, t) \right\|^2 \right] \end{aligned} \quad (29)$$

2.1.5 简化 (Simplification)

Ho et al. (2020) 发现，使用忽略加权项的简化目标进行扩散模型训练效果更好：

$$\begin{aligned} L_t^{\text{simple}} &= \mathbb{E}_{t \sim [1, T], \mathbf{x}_0, \epsilon_t} [\|\epsilon_t - \epsilon_\theta(\mathbf{x}_t, t)\|^2] \\ &= \mathbb{E}_{t \sim [1, T], \mathbf{x}_0, \epsilon_t} \left[\left\| \epsilon_t - \epsilon_\theta(\sqrt{\bar{\alpha}_t} \mathbf{x}_0 + \sqrt{1 - \bar{\alpha}_t} \epsilon_t, t) \right\|^2 \right] \end{aligned} \quad (30)$$

最终的简单目标函数是：

$$L_{\text{simple}} = L_t^{\text{simple}} + C \quad (31)$$

其中 C 是一个不依赖于 θ 的常数。

Algorithm 1 Training

- 1: **repeat**
 - 2: $\mathbf{x}_0 \sim q(\mathbf{x}_0)$
 - 3: $t \sim \text{Uniform}(\{1, \dots, T\})$
 - 4: $\epsilon \sim \mathcal{N}(\mathbf{0}, \mathbf{I})$
 - 5: Take gradient descent step on
 $\nabla_\theta \|\epsilon - \epsilon_\theta(\sqrt{\bar{\alpha}_t} \mathbf{x}_0 + \sqrt{1 - \bar{\alpha}_t} \epsilon, t)\|^2$
 - 6: **until** converged
-

Algorithm 2 Sampling

- 1: $\mathbf{x}_T \sim \mathcal{N}(\mathbf{0}, \mathbf{I})$
 - 2: **for** $t = T, \dots, 1$ **do**
 - 3: $\mathbf{z} \sim \mathcal{N}(\mathbf{0}, \mathbf{I})$ if $t > 1$, else $\mathbf{z} = \mathbf{0}$
 - 4: $\mathbf{x}_{t-1} = \frac{1}{\sqrt{\alpha_t}} \left(\mathbf{x}_t - \frac{1 - \alpha_t}{\sqrt{1 - \bar{\alpha}_t}} \epsilon_\theta(\mathbf{x}_t, t) \right) + \sigma_t \mathbf{z}$
 - 5: **end for**
 - 6: **return** \mathbf{x}_0
-

图 2-4 DDPM 中的训练和采样算法

2.2 理论联系与参数化改进

2.2.1 与随机梯度朗之万动力学 (Langevin Dynamics) 的联系

朗之万动力学是物理学中的一个概念，用于统计模拟分子系统。结合随机梯度下降，随机梯度朗之万动力学 (Welling & Teh 2011) 可以仅使用梯度 $\nabla_{\mathbf{x}} \log p(\mathbf{x})$ 在马尔可夫更新链中从概率密度 $p(\mathbf{x})$ 生成样本：

$$\mathbf{x}_t = \mathbf{x}_{t-1} + \frac{\delta}{2} \nabla_{\mathbf{x}} \log p(\mathbf{x}_{t-1}) + \sqrt{\delta} \epsilon_t, \quad \text{where } \epsilon_t \sim \mathcal{N}(\mathbf{0}, \mathbf{I}) \quad (32)$$

其中 δ 是步长。当 $T \rightarrow \infty, \epsilon \rightarrow 0$ 时， \mathbf{x}_T 等价于真实概率密度 $p(\mathbf{x})$ 。相比标准 SGD，它通过注入高斯噪声来避免陷入局部极小值。

2.2.2 与噪声条件分数网络 (NCSN) 的联系

Song & Ermon (2019) 提出了基于分数的生成模型。该模型通过分数匹配 (Score Matching) 估计数据分布的梯度 (即分数 $\nabla_{\mathbf{x}} \log q(\mathbf{x})$)，并利用朗之万动力学生成样本。为了解决高维数据在低维流形上分布导致的分数估计不准确问题，他们提出添加不同级别的高斯噪声来扰动数据，并训练一个噪声条件分数网络 $s_{\theta}(\mathbf{x}, t) \approx \nabla_{\mathbf{x}} \log q(\mathbf{x})$ 。

如果我们使用扩散过程的标记，分数近似为 $s_{\theta}(\mathbf{x}_t, t) \approx \nabla_{\mathbf{x}_t} \log q(\mathbf{x}_t)$ 。回忆 $q(\mathbf{x}_t | \mathbf{x}_0) \sim \mathcal{N}(\sqrt{\bar{\alpha}_t} \mathbf{x}_0, (1 - \bar{\alpha}_t) \mathbf{I})$ ，因此：

$$s_{\theta}(\mathbf{x}_t, t) \approx -\frac{\epsilon_{\theta}(\mathbf{x}_t, t)}{\sqrt{1 - \bar{\alpha}_t}} \quad (33)$$

这表明预测噪声 ϵ_{θ} 本质上是在预测分数的缩放版本。

2.2.3 β_t 的参数化 (Parameterization of β_t)

Ho et al. (2020) 使用线性增加的常数序列作为方差调度，从 $\beta_1 = 10^{-4}$ 到 $\beta_T = 0.02$ 。Nichol & Dhariwal (2021) 提出了一种基于余弦的方差调度，以获得更好的对数似然 (NLL)。这种调度函数在训练过程中间提供近乎线性的下降，而在 $t = 0$ 和 $t = T$ 附近变化微妙。

$$\beta_t = \text{clip}\left(1 - \frac{\bar{\alpha}_t}{\bar{\alpha}_{t-1}}, 0.999\right), \quad \bar{\alpha}_t = \frac{f(t)}{f(0)}, \quad f(t) = \cos\left(\frac{t/T + s}{1 + s} \cdot \frac{\pi}{2}\right)^2 \quad (34)$$

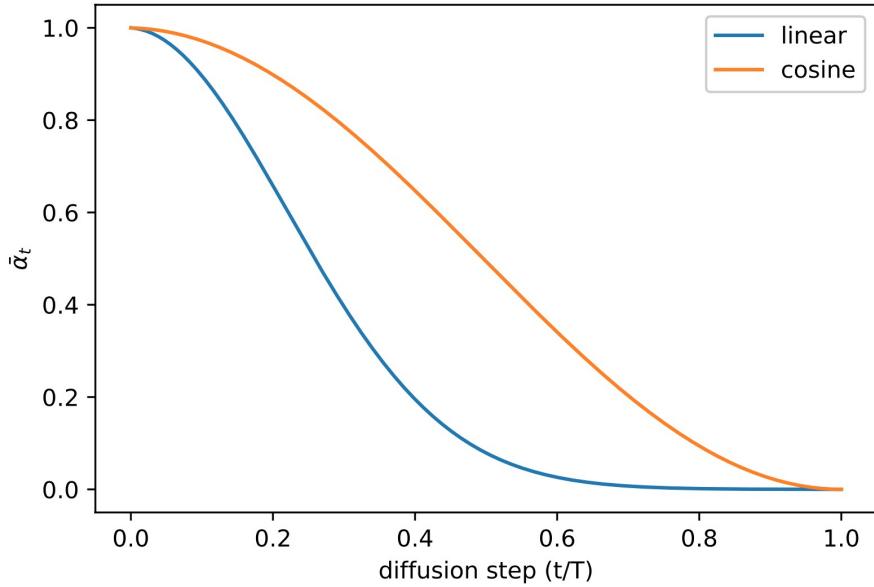


图 2-5 训练期间 β_t 的线性调度与余弦调度的比较

2.2.4 逆向方差 Σ_θ 的参数化

Ho et al. (2020) 将 β_t 设为常数，并设定 $\Sigma_\theta(\mathbf{x}_t, t) = \sigma_t^2 \mathbf{I}$ ，其中 σ_t 未学习。Nichol & Dhariwal (2021) 提出将 $\Sigma_\theta(\mathbf{x}_t, t)$ 学习为 β_t 和 $\tilde{\beta}_t$ 之间的插值，模型输出一个混合向量 v ：

$$\Sigma_\theta(\mathbf{x}_t, t) = \exp(v \log \beta_t + (1 - v) \log \tilde{\beta}_t) \quad (35)$$

这需要引入混合目标函数 $L_{\text{hybrid}} = L_{\text{simple}} + \lambda L_{\text{VLB}}$ 来指导 Σ_θ 的学习。

Model	ImageNet	CIFAR
Glow (Kingma & Dhariwal, 2018)	3.81	3.35
Flow++ (Ho et al., 2019)	3.69	3.08
PixelCNN (van den Oord et al., 2016c)	3.57	3.14
SPN (Menick & Kalchbrenner, 2018)	3.52	-
NVAE (Vahdat & Kautz, 2020)	-	2.91
Very Deep VAE (Child, 2020)	3.52	2.87
PixelSNAIL (Chen et al., 2018)	3.52	2.85
Image Transformer (Parmar et al., 2018)	3.48	2.90
Sparse Transformer (Child et al., 2019)	3.44	2.80
Routing Transformer (Roy et al., 2020)	3.43	-
DDPM (Ho et al., 2020)	3.77	3.70
DDPM (cont flow) (Song et al., 2020b)	-	2.99
Improved DDPM (ours)	3.53	2.94

图 2-6 改进版 DDPM 与其他生成模型的负对数似然 (NLL) 比较

2.3 条件生成 (Conditioned Generation)

虽然在 ImageNet 数据集等带有条件信息的图像上训练生成模型，但通常会根据类别标签或一段描述性文本来生成样本。

2.3.1 分类器引导扩散 (Classifier Guided Diffusion)

为了将类别信息显式地合并到扩散过程中, Dhariwal & Nichol (2021) 在噪声图像 \mathbf{x}_t 上训练了一个分类器 $f_\phi(y|\mathbf{x}_t, t)$, 并使用梯度 $\nabla_{\mathbf{x}_t} \log f_\phi(y|\mathbf{x}_t)$ 来引导扩散采样过程朝向条件信息 y (例如目标类别标签) 移动。回顾 $\nabla_{\mathbf{x}_t} \log q(\mathbf{x}_t) = -\frac{1}{\sqrt{1-\bar{\alpha}_t}}\epsilon_\theta(\mathbf{x}_t, t)$, 我们可以将联合分布 $q(\mathbf{x}_t, y)$ 的分数函数写为:

$$\begin{aligned}\nabla_{\mathbf{x}_t} \log q(\mathbf{x}_t, y) &= \nabla_{\mathbf{x}_t} \log q(\mathbf{x}_t) + \nabla_{\mathbf{x}_t} \log q(y|\mathbf{x}_t) \\ &\approx -\frac{1}{\sqrt{1-\bar{\alpha}_t}}\epsilon_\theta(\mathbf{x}_t, t) + \nabla_{\mathbf{x}_t} \log f_\phi(y|\mathbf{x}_t) \\ &= -\frac{1}{\sqrt{1-\bar{\alpha}_t}}\left(\epsilon_\theta(\mathbf{x}_t, t) - \sqrt{1-\bar{\alpha}_t}\nabla_{\mathbf{x}_t} \log f_\phi(y|\mathbf{x}_t)\right)\end{aligned}\quad (36)$$

由此产生的扰动预测器 $\bar{\epsilon}_\theta$ 形式如下:

$$\bar{\epsilon}_\theta(\mathbf{x}_t, t) = \epsilon_\theta(\mathbf{x}_t, t) - \sqrt{1-\bar{\alpha}_t}\nabla_{\mathbf{x}_t} \log f_\phi(y|\mathbf{x}_t) \quad (37)$$

为了控制分类器引导的强度, 我们可以向 delta 部分添加权重 w :

$$\bar{\epsilon}_\theta(\mathbf{x}_t, t) = \epsilon_\theta(\mathbf{x}_t, t) - \sqrt{1-\bar{\alpha}_t}w\nabla_{\mathbf{x}_t} \log f_\phi(y|\mathbf{x}_t) \quad (38)$$

Algorithm 1 Classifier guided diffusion sampling, given a diffusion model $(\mu_\theta(x_t), \Sigma_\theta(x_t))$, classifier $f_\phi(y|x_t)$, and gradient scale s .

```
Input: class label  $y$ , gradient scale  $s$ 
 $x_T \leftarrow$  sample from  $\mathcal{N}(0, \mathbf{I})$ 
for all  $t$  from  $T$  to 1 do
     $\mu, \Sigma \leftarrow \mu_\theta(x_t), \Sigma_\theta(x_t)$ 
     $x_{t-1} \leftarrow$  sample from  $\mathcal{N}(\mu + s\Sigma \nabla_{x_t} \log f_\phi(y|x_t), \Sigma)$ 
end for
return  $x_0$ 
```

Algorithm 2 Classifier guided DDIM sampling, given a diffusion model $\epsilon_\theta(x_t)$, classifier $f_\phi(y|x_t)$, and gradient scale s .

```
Input: class label  $y$ , gradient scale  $s$ 
 $x_T \leftarrow$  sample from  $\mathcal{N}(0, \mathbf{I})$ 
for all  $t$  from  $T$  to 1 do
     $\hat{\epsilon} \leftarrow \epsilon_\theta(x_t) - \sqrt{1-\bar{\alpha}_t}\nabla_{x_t} \log f_\phi(y|x_t)$ 
     $x_{t-1} \leftarrow \sqrt{\bar{\alpha}_{t-1}}\left(\frac{x_t - \sqrt{1-\bar{\alpha}_t}\hat{\epsilon}}{\sqrt{\bar{\alpha}_t}}\right) + \sqrt{1-\bar{\alpha}_{t-1}}\hat{\epsilon}$ 
end for
return  $x_0$ 
```

图 2-7 使用分类器引导来进行条件生成的算法 (DDPM 和 DDIM)

2.3.2 无分类器引导 (Classifier-Free Guidance)

即使没有独立的分类器 f_ϕ , 也可以通过结合条件和无条件扩散模型的分数来运行条件扩散步骤 (Ho & Salimans, 2021)。具体来说, 我们在成对数据 (\mathbf{x}, y) 上训练条件扩

散模型 $p_\theta(\mathbf{x}|y)$, 其中条件信息 y 会随机被丢弃, 以便模型也学会无条件地生成图像, 即 $\epsilon_\theta(\mathbf{x}_t, t) = \epsilon_\theta(\mathbf{x}_t, t, y = \emptyset)$ 。

隐式分类器的梯度可以用条件和无条件分数估计器来表示:

$$\begin{aligned}\nabla_{\mathbf{x}_t} \log p(y|\mathbf{x}_t) &= \nabla_{\mathbf{x}_t} \log p(\mathbf{x}_t|y) - \nabla_{\mathbf{x}_t} \log p(\mathbf{x}_t) \\ &= -\frac{1}{\sqrt{1-\bar{\alpha}_t}} (\epsilon_\theta(\mathbf{x}_t, t, y) - \epsilon_\theta(\mathbf{x}_t, t))\end{aligned}\quad (39)$$

将其代入分类器引导的分数公式中, 分数便不再依赖单独的分类器:

$$\begin{aligned}\bar{\epsilon}_\theta(\mathbf{x}_t, t, y) &= \epsilon_\theta(\mathbf{x}_t, t, y) - \sqrt{1-\bar{\alpha}_t} w \nabla_{\mathbf{x}_t} \log p(y|\mathbf{x}_t) \\ &= \epsilon_\theta(\mathbf{x}_t, t, y) + w (\epsilon_\theta(\mathbf{x}_t, t, y) - \epsilon_\theta(\mathbf{x}_t, t)) \\ &= (w+1)\epsilon_\theta(\mathbf{x}_t, t, y) - w\epsilon_\theta(\mathbf{x}_t, t)\end{aligned}\quad (40)$$

实验表明, 无分类器引导可以在 FID (图像保真度) 和 IS (多样性) 之间取得良好的平衡。GLIDE (Nichol, Dhariwal & Ramesh, et al. 2022) 发现无分类器引导优于 CLIP 引导。

2.4 加速扩散模型 (Speed up Diffusion Models)

从 DDPM 生成样本非常慢, 因为 T 可能高达 1000 步。一重简单的加速方法是运行跨步采样 (Strided Sampling), 例如每隔 $[T/S]$ 步进行一次更新。

另一种方法重写 $q_\sigma(\mathbf{x}_{t-1}|\mathbf{x}_t, \mathbf{x}_0)$, 使其由所需的标准差 σ_t 参数化:

$$\begin{aligned}\mathbf{x}_{t-1} &= \sqrt{\bar{\alpha}_{t-1}} \mathbf{x}_0 + \sqrt{1-\bar{\alpha}_{t-1}} \epsilon_{t-1} \\ &= \sqrt{\bar{\alpha}_{t-1}} \mathbf{x}_0 + \sqrt{1-\bar{\alpha}_{t-1}-\sigma_t^2} \epsilon_t + \sigma_t \epsilon \\ &= \sqrt{\bar{\alpha}_{t-1}} \left(\frac{\mathbf{x}_t - \sqrt{1-\bar{\alpha}_t} \epsilon_\theta^{(t)}(\mathbf{x}_t)}{\sqrt{\bar{\alpha}_t}} \right) + \sqrt{1-\bar{\alpha}_{t-1}-\sigma_t^2} \epsilon_\theta^{(t)}(\mathbf{x}_t) + \sigma_t \epsilon\end{aligned}\quad (41)$$

这样我们得到:

$$q_\sigma(\mathbf{x}_{t-1}|\mathbf{x}_t, \mathbf{x}_0) = \mathcal{N} \left(\mathbf{x}_{t-1}; \sqrt{\bar{\alpha}_{t-1}} \left(\frac{\mathbf{x}_t - \sqrt{1-\bar{\alpha}_t} \epsilon_\theta^{(t)}(\mathbf{x}_t)}{\sqrt{\bar{\alpha}_t}} \right) + \sqrt{1-\bar{\alpha}_{t-1}-\sigma_t^2} \epsilon_\theta^{(t)}(\mathbf{x}_t), \sigma_t^2 \mathbf{I} \right) \quad (42)$$

回忆 $\tilde{\beta}_t = \frac{1-\bar{\alpha}_{t-1}}{1-\bar{\alpha}_t} \cdot \beta_t$, 令 $\sigma_t^2 = \eta \cdot \tilde{\beta}_t$, 我们可以通过超参数 $\eta \in \mathbb{R}^+$ 控制采样随机性。

2.4.1 DDIM (Denoising Diffusion Implicit Models)

当 $\eta = 0$ 时, 采样过程变得确定性, 此时模型称为去噪扩散隐模型 (DDIM)。DDIM 更新步骤 (针对加速轨迹中的 $s < t$):

$$q_{\sigma, s < t}(\mathbf{x}_s|\mathbf{x}_t, \mathbf{x}_0) = \mathcal{N} \left(\mathbf{x}_s; \sqrt{\bar{\alpha}_s} \left(\frac{\mathbf{x}_t - \sqrt{1-\bar{\alpha}_t} \epsilon_\theta^{(t)}(\mathbf{x}_t)}{\sqrt{\bar{\alpha}_t}} \right) + \sqrt{1-\bar{\alpha}_s-\sigma_t^2} \epsilon_\theta^{(t)}(\mathbf{x}_t), \sigma_t^2 \mathbf{I} \right) \quad (43)$$

DDIM 具有一致性属性, 即以相同的隐变量为条件的多个样本应具有相似的高级特征。这也是因为 DDIM 可以在隐变量中进行语义上有意义的插值。

2.4.2 渐进式蒸馏 (Progressive Distillation)

Salimans & Ho (2022) 提出了一种将训练好的确定性采样器蒸馏成步数减半的新模型的方法。在每次蒸馏迭代中，我们可以将采样步骤减半。学生模型初始化自教师模型，并朝着匹配 2 个教师 DDIM 步骤的目标进行去噪（即算法中的 $z_{t''}$ ），而不是使用原始样本 \mathbf{x}_0 作为去噪目标。

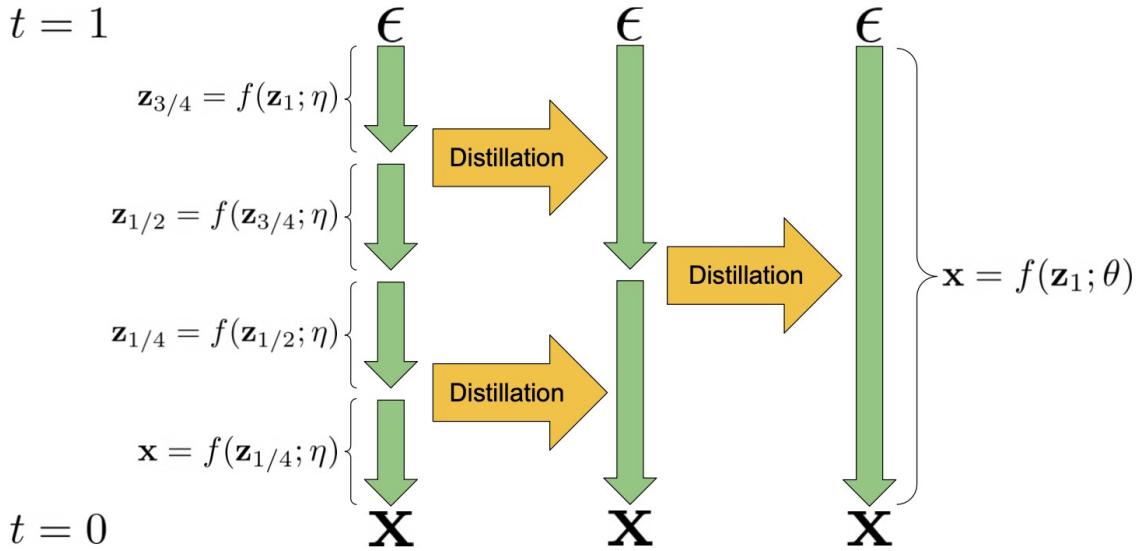


图 2-8 渐进式蒸馏可以在每次迭代中将扩散采样步骤减半

2.4.3 一致性模型 (Consistency Models)

Song et al. (2023) 提出的一致性模型旨在学习将扩散采样轨迹上的任何中间噪声数据点 \mathbf{x}_t 直接映射回其原点 \mathbf{x}_0 。如名称所示，它具有自一致性。

一致性模型定义为 $f_\theta : (\mathbf{x}_t, t) \mapsto \mathbf{x}_\epsilon$ ，使得对同一轨迹上的所有点输出相同。它可以被参数化为：

$$f_\theta(\mathbf{x}, t) = c_{\text{skip}}(t)\mathbf{x} + c_{\text{out}}(t)F_\theta(\mathbf{x}, t) \quad (44)$$

该论文介绍了两种训练一致性模型的方法：

1. **一致性蒸馏 (Consistency Distillation, CD):** 将预训练的扩散模型蒸馏为一致性模型。损失函数为：

$$\mathcal{L}_{CD}^N(\theta, \theta^-; \phi) = \mathbb{E}[\lambda(t_n)d(f_\theta(\mathbf{x}_{t_{n+1}}, t_{n+1}), f_{\theta^-}(\mathbf{x}_{t_n}^\phi, t_n))] \quad (45)$$

其中 θ^- 是 θ 的 EMA 版本。

2. **一致性训练 (Consistency Training, CT):** 独立训练一致性模型。损失函数为：

$$\mathcal{L}_{CT}^N(\theta, \theta^-; \phi) = \mathbb{E}[\lambda(t_n)d(f_\theta(\mathbf{x} + t_{n+1}\mathbf{z}, t_{n+1}), f_{\theta^-}(\mathbf{x} + t_n\mathbf{z}, t_n))] \quad (46)$$

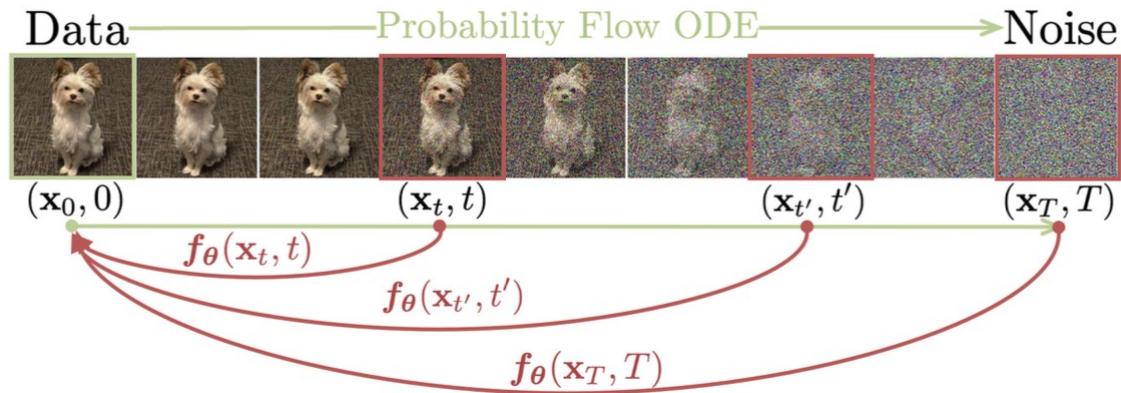


图 2-9 一致性模型学习将轨迹上的任何数据点映射回其原点

2.5 隐变量空间 (Latent Variable Space)

潜在扩散模型 (Latent Diffusion Model, LDM; Rombach et al. 2022) 在潜在空间而不是像素空间中运行扩散过程，从而降低了训练成本并加快了推理速度。其动机是观察到图像的大部分比特贡献了感知细节，而语义和概念构成在激进压缩后仍然保留。

LDM 将感知压缩（使用自动编码器去除像素级冗余）和语义压缩（在学习到的隐变量上使用扩散过程生成语义概念）松散地分解开来。

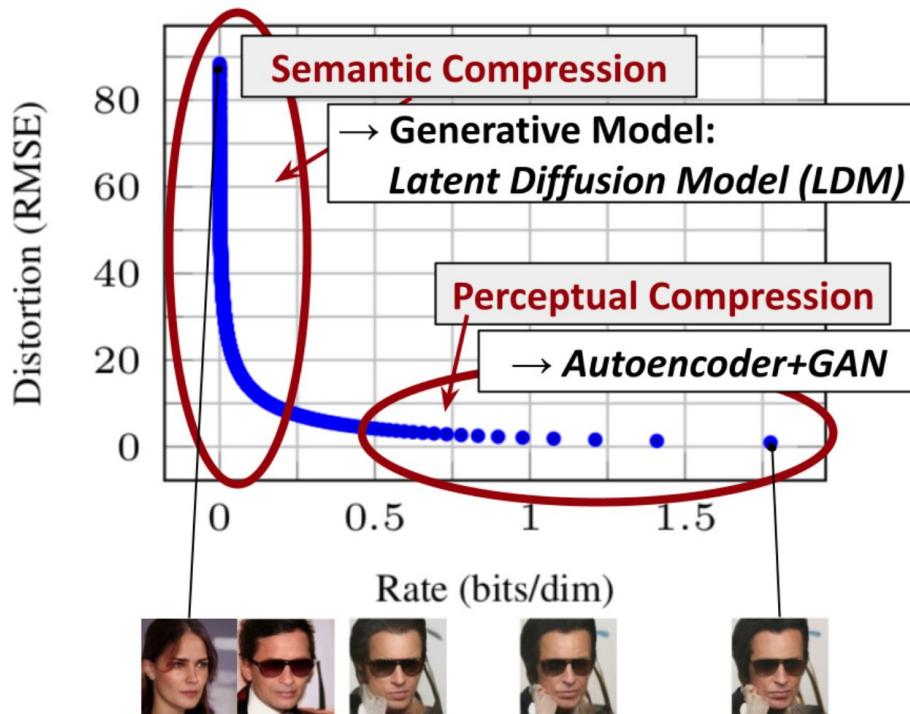


图 2-10 压缩率与失真之间的权衡，说明了两阶段压缩：感知压缩和语义压缩

感知压缩依赖于一个自编码器模型。扩散和去噪过程发生在隐向量 z 上。去噪模型

是一个时间条件的 U-Net，增强了交叉注意力机制（Cross-Attention）以处理灵活的条件信息（如类别标签、语义图等）。条件输入 y 被投影到中间表示 $\tau_\theta(y) \in \mathbb{R}^{M \times d_\tau}$ ，然后映射到交叉注意力组件：

$$\text{Attention}(Q, K, V) = \text{softmax}\left(\frac{QK^\top}{\sqrt{d}}\right) \cdot V \quad (47)$$

其中：

$$Q = W_Q^{(i)} \cdot \varphi_i(z_i), \quad K = W_K^{(i)} \cdot \tau_\theta(y), \quad V = W_V^{(i)} \cdot \tau_\theta(y) \quad (48)$$

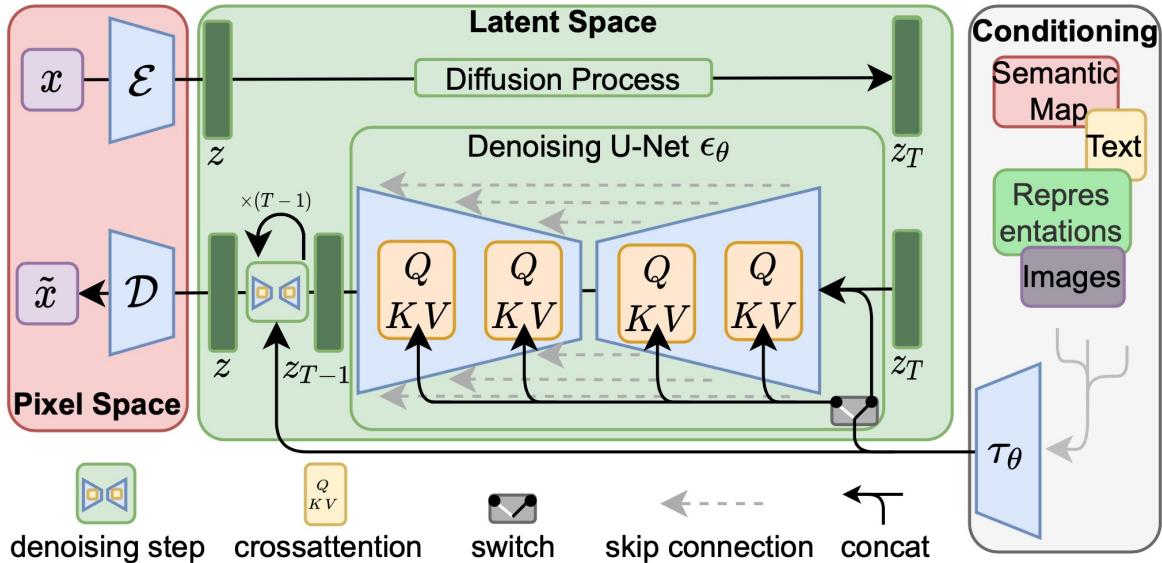


图 2-11 潜在扩散模型 (LDM) 的架构

2.6 扩大生成分辨率与质量

为了生成高分辨率的高质量图像，Ho et al. (2021) 提出使用多个扩散模型的级联管道。管道模型之间的噪声条件增强（Noise Conditioning Augmentation）对于最终图像质量至关重要。

2.6.1 级联扩散模型 (Cascaded Diffusion Models)

级联管道包括一个生成低分辨率图像的基础模型，以及随后的一个或多个超分辨率（Super-Resolution）扩散模型。研究发现，最有效的调节噪声是在低分辨率下施加高斯噪声，在高分辨率下施加高斯模糊。

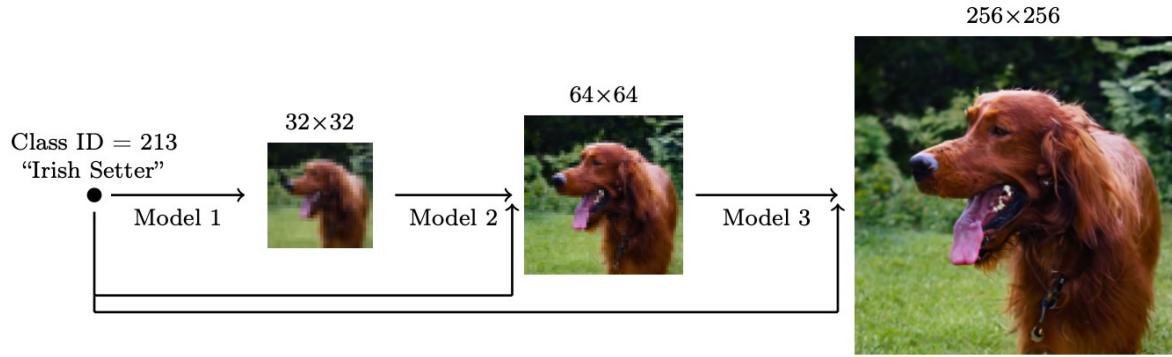


图 2-12 多个扩散模型在增加分辨率时的级联管道

2.6.2 unCLIP (DALL·E 2)

unCLIP (Ramesh et al. 2022) 是一种利用 CLIP 文本编码器的两阶段扩散模型。它并行学习两个模型：

1. 先验模型 (Prior) $P(c^i|y)$: 给定文本 y , 输出 CLIP 图像嵌入 c^i 。
2. 解码器 (Decoder) $P(x|c^i, [y])$: 给定 CLIP 图像嵌入 c^i (以及可选的原始文本 y), 生成图像 x 。

这两个模型通过贝叶斯规则实现条件生成：

$$P(x|y) = \underbrace{P(x, c^i|y)}_{c^i \text{ is deterministic given } x} = P(x|c^i, y)P(c^i|y) \quad (49)$$

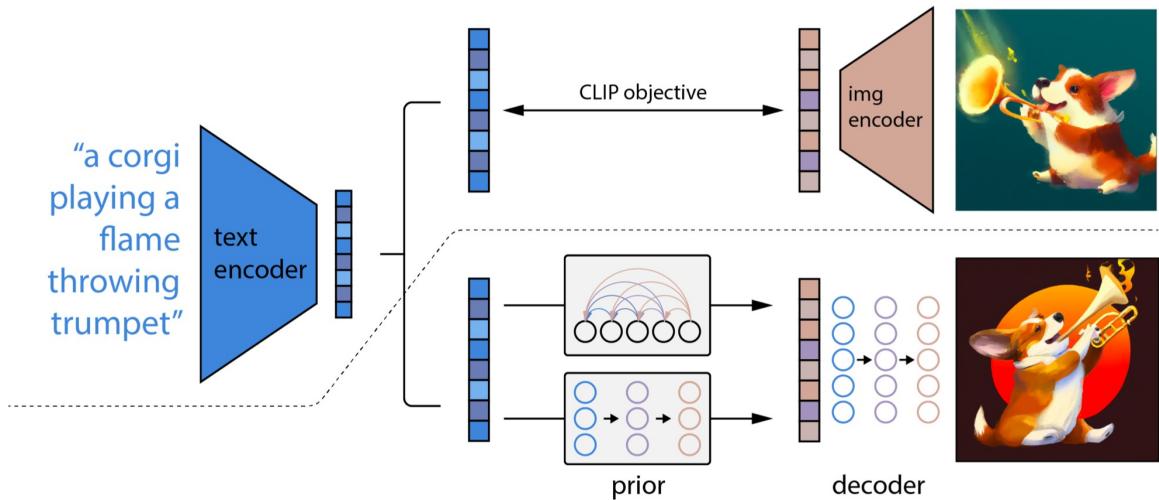


图 2-13 unCLIP 的架构

2.6.3 Imagen

Imagen (Saharia et al. 2022) 使用预训练的大型语言模型（即冻结的 T5-XXL 文本编码器）对文本进行编码。研究发现，扩大文本编码器的规模比扩大 U-Net 的规模对图像质量更重要。

2.7 模型架构 (Model Architecture)

扩散模型有两个常见的骨干架构选择：U-Net 和 Transformer。

2.7.1 U-Net

U-Net (Ronneberger, et al. 2015) 由下采样堆栈和上采样堆栈组成，中间通过快捷连接 (Shortcut Connections) 相连，提供高分辨率特征。

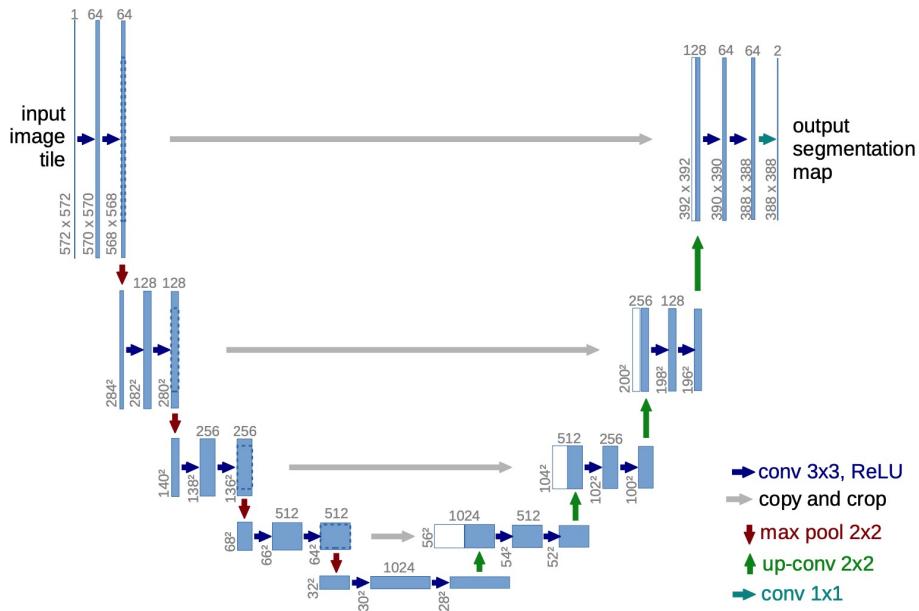


图 2-14 U-Net 架构

2.7.2 ControlNet

为了通过额外的图像（如 Canny 边缘、Hough 线、人体骨架图等）对图像生成进行条件控制，ControlNet (Zhang et al. 2023) 引入了架构更改。它主要包括两步：锁定原始模型参数，并创建一个可训练的副本；通过“零卷积”层连接这两个块。零卷积层的权重和偏置初始化为零，从而在训练初始阶段保护骨干网络不受随机噪声梯度的影响。对于神经网络块 $F_\theta(\cdot)$ ，ControlNet 首先冻结 θ ，然后克隆它得到带有参数 θ_c 的副本。零卷积层 $Z_{\theta_{z1}}$ 和 $Z_{\theta_{z2}}$ 连接这两个块：

$$y_c = F_\theta(\mathbf{x}) + Z_{\theta_{z2}}(F_{\theta_c}(\mathbf{x} + Z_{\theta_{z1}}(\mathbf{c}))) \quad (50)$$

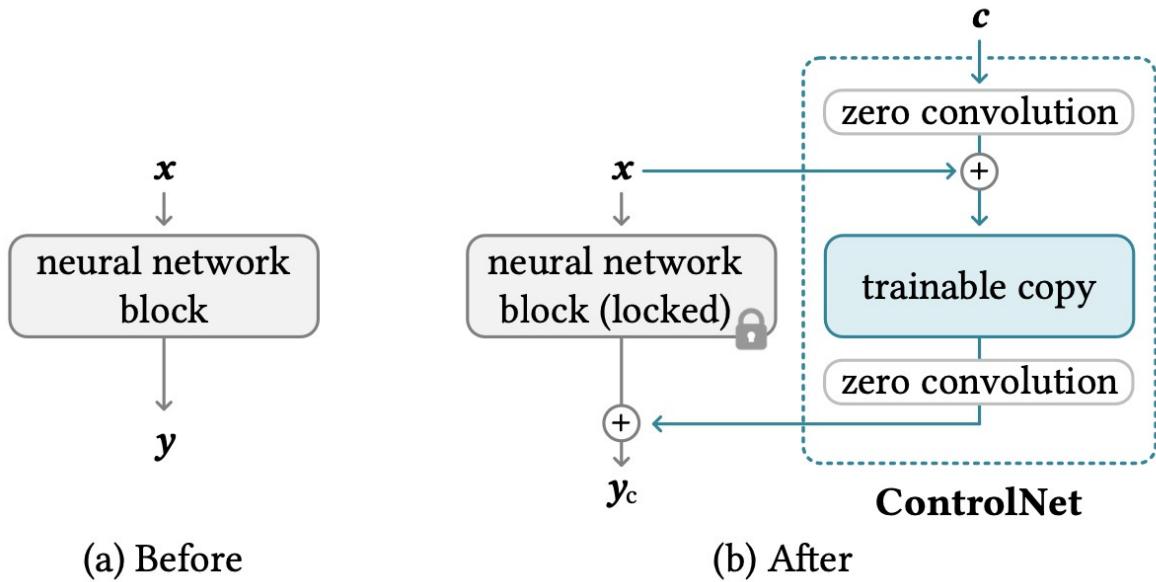


图 2-15 ControlNet 架构

2.7.3 Diffusion Transformer (DiT)

Diffusion Transformer (DiT; Peebles & Xie, 2023) 在潜在图块（Latent Patches）上运行。其步骤如下：1. 将输入 z 的潜在表示作为输入。2. 将噪声潜在表示“Patchify”为序列。3. 输入 Transformer 块。DiT 使用 adaLN-Zero（自适应层归一化）机制来注入条件信息（如时间步 t 和类别 c ）。DiT 的一大优势是其性能随着计算量的增加和模型规模的扩大而有效扩展。

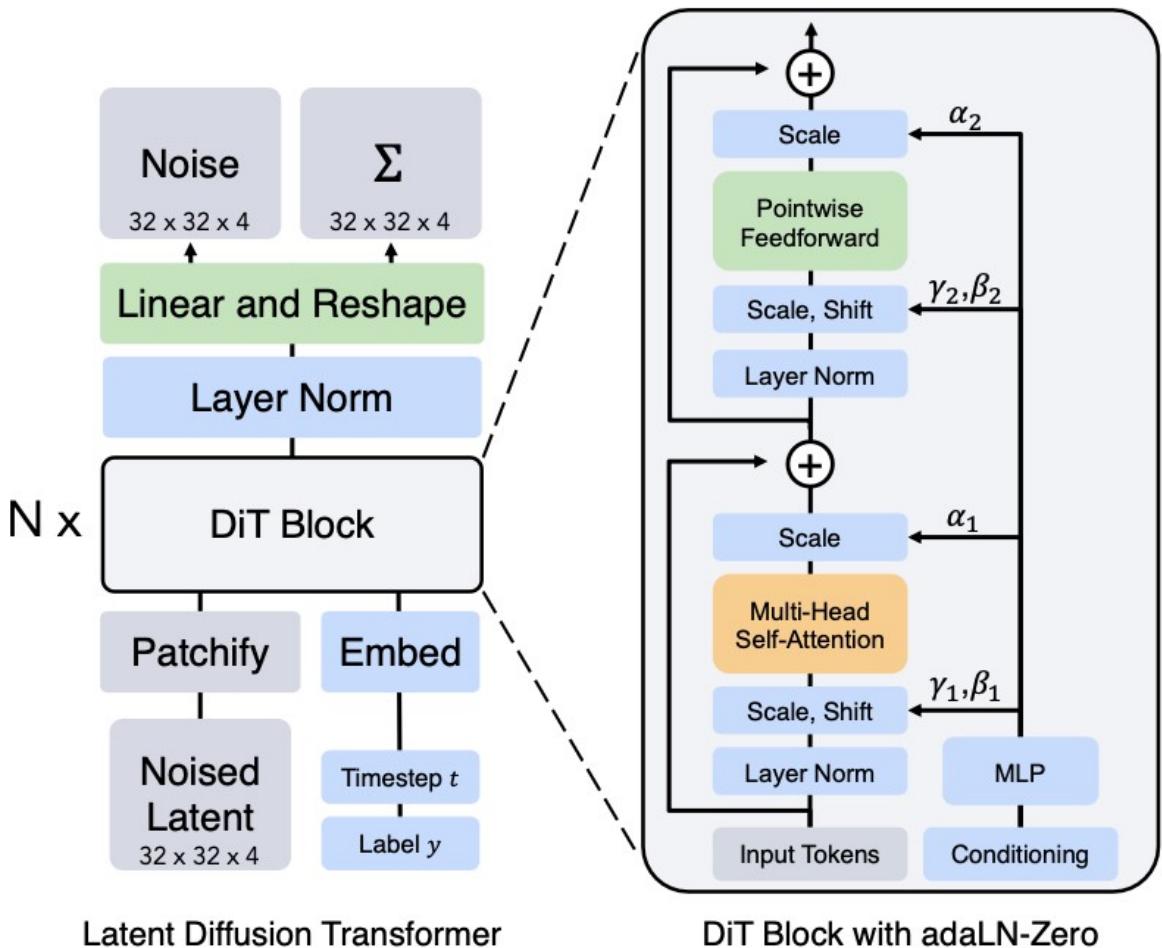


图 2-16 Diffusion Transformer (DiT) 架构

参考文献

- [1] WENG L. From autoencoder to beta-vae[J/OL]. <https://lilianweng.github.io/posts/2018-08-12-vae/>.
- [2] SOHL-DICKSTEIN J, WEISS E, MAHESWARANATHAN N, et al. Deep unsupervised learning using nonequilibrium thermodynamics[J]. International Conference on Machine Learning, 2015.
- [3] SONG Y, ERMON S. Generative modeling by estimating gradients of the data distribution[J]. Advances in Neural Information Processing Systems, 2019.
- [4] HO J, JAIN A, ABBEEL P. Denoising diffusion probabilistic models[J]. Advances in Neural Information Processing Systems, 2020.
- [5] SONG J, MENG C, ERMON S. Denoising diffusion implicit models[A]. 2020.
- [6] DHARIWAL P, NICHOL A. Diffusion models beat gans on image synthesis[J]. Advances in Neural Information Processing Systems, 2021.

- [7] HO J, SALIMANS T. Classifier-free diffusion guidance[A]. 2022.
- [8] NICHOL A, DHARIWAL P, RAMESH A, et al. Glide: Towards photorealistic image generation and editing with text-guided diffusion models[A]. 2021.
- [9] ROMBACH R, BLATTMANN A, LORENZ D, et al. High-resolution image synthesis with latent diffusion models[J]. Proceedings of the IEEE/CVF conference on computer vision and pattern recognition, 2022.
- [10] RAMESH A, DHARIWAL P, NICHOL A, et al. Hierarchical text-conditional image generation with clip latents[A]. 2022.
- [11] SAHARIA C, CHAN W, SAXENA S, et al. Photorealistic text-to-image diffusion models with deep language understanding[J]. Advances in Neural Information Processing Systems, 2022.
- [12] RONNEBERGER O, FISCHER P, BROX T. U-net: Convolutional networks for biomedical image segmentation[J]. Medical image computing and computer-assisted intervention–MICCAI 2015, 2015.
- [13] ZHANG L, RAO A, AGRAWALA M. Adding conditional control to text-to-image diffusion models [A]. 2023.
- [14] PEEBLES W, XIE S. Scalable diffusion models with transformers[J]. Proceedings of the IEEE/CVF International Conference on Computer Vision, 2023.
- [15] SALIMANS T, HO J. Progressive distillation for fast sampling of diffusion models[A]. 2022.
- [16] SONG Y, DHARIWAL P, CHEN M, et al. Consistency models[A]. 2023.