Chapter 6

Basic

第一类：基于玻尔兹曼机的生成模型

代表模型：受限玻尔兹曼机（RBM）、深度玻尔兹曼机（DBM）、深度信念网络（DBN）

1.1 受限玻尔兹曼机（Restricted Boltzmann Machine, RBM）

- **结构**：一种**双层神经网络**，包括可见层（visible units）和隐藏层（hidden units），两层之间完全连接，但层内无连接。
- **能量函数**：

$$E(v,h) = -v^\top Wh - b^\top v - c^\top h$$

  其中 v 是可见层输入，h 是隐藏层，W 是权重矩阵，b, c 为偏置。
- **生成机制**：通过对联合概率 $P(v,h)$ 的建模和Gibbs采样来从模型中生成数据。

1.2 深度玻尔兹曼机（Deep Boltzmann Machine, DBM）

- **结构**：多个RBM堆叠而成，各层之间双向连接，隐藏层之间也有连接。
- **特点**：可以同时对多层特征进行联合建模，表达能力强，但训练复杂。
- **训练**：使用**近似推断**（如变分推断）+ 分层预训练（Layer-wise Pretraining）。

1.3 深度信念网络（Deep Belief Network, DBN）

- **结构**：由多层RBM堆叠构成，但与DBM不同，只有**最顶层是无向连接**，其余为**有向连接**（从隐藏层到可见层）。
- **生成方式**：从顶层采样后，逐层向下生成样本。
- **训练过程**：
  - 逐层无监督预训练（使用RBM）
  - 再进行微调（如反向传播）

优缺点总结

- **优点**：建模能力强，适合高维数据表示学习
- **缺点**：训练复杂，采样慢，已逐渐被新模型取代

第二类：基于自编码器的生成模型

代表模型：变分自编码器（Variational Autoencoder, VAE）、深度自编码器（Deep Autoencoder）

2.1 自编码器基础（Autoencoder, AE）

- **结构**：编码器 $q_\phi(z|x)$ + 解码器 $p_\theta(x|z)$
- **功能**：将输入 x 映射为潜在向量 z，再重构出原始数据。
- **局限**：AE并不能作为真正的**生成模型**，因为其潜在空间不具有概率分布意义。

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

- **核心思想**：通过引入概率推断，让自编码器具备**生成能力**。
- **目标函数**（Evidence Lower Bound, ELBO）：

$$\mathcal{L} = \mathbb{E}_{q_\phi(z|x)}[\log p_\theta(x|z)] - D_{KL}(q_\phi(z|x)||p(z))$$

  第一项是重建误差，第二项是潜在分布与先验的KL散度。
- **训练方式**：使用**重参数技巧（reparameterization trick）**使得梯度可以反向传播。

优缺点总结

- **优点**：生成质量稳定、训练高效、结构可扩展
- **缺点**：生成样本质量通常略低于GAN和扩散模型，可能存在模糊问题

第三类：扩散生成模型（Diffusion Model）

代表模型：DDPM（Denoising Diffusion Probabilistic Model）、Score-Based Generative Model

3.1 核心思想

- 模仿**热力学扩散过程**，将原始数据逐渐加噪（正向过程），然后学习如何从噪声恢复原数据（逆向过程）。
- 与物理过程类似，正向为数据"扩散成白噪声"，逆向为"从噪声逐步去噪回真实样本"。

3.2 正向扩散过程（Forward Process）

- 每一步加一点高斯噪声，使得数据逐步变得混乱。
- 数学表达为：

$$q(x_t|x_{t-1}) = \mathcal{N}(x_t; \sqrt{1-\beta_t}x_{t-1}, \beta_t I)$$

其中 t 为时间步，β_t 控制噪声强度。

3.3 反向去噪过程（Reverse Process）

- 学习一个神经网络（通常是U-Net结构）来预测噪声或原始图像，从而一步步去噪。

3.4 损失函数

- 训练目标是最小化预测噪声与真实噪声之间的均方误差：

$$\mathcal{L}_{\text{simple}} = \mathbb{E}_{x_t,\epsilon}[\|\epsilon - \epsilon_\theta(x_t, t)\|^2]$$

3.5 特点与优势

- **生成质量极高**（目前SOTA图像生成模型，如Stable Diffusion、Imagen等）
- **稳定性好**，不会像GAN那样不易收敛或模式崩溃
- **训练慢**但**采样可加速**（通过DDIM、Latent Diffusion等技术）

什么是深度生成模型？

深度生成模型（**Deep Generative Model**）是指利用深度神经网络来**学习数据的分布**，从而可以**生成新的、具有相似分布的样本**的一类模型。简而言之，它们不是仅仅对输入做判断（比如分类），而是能够**"创造"新的数据**，如图像、文本、语音等。

> 深度生成模型是使用深度学习方法建模数据分布的概率模型，能够生成与原始数据类似的新样本。

💡 Hebbian 法则是一种**生物启发式的学习规则**，源自神经科学家 Donald Hebb 在 1949 年提出的经典理论：

> “Cells that fire together, wire together.”
>
> —— 同时激活的神经元会加强连接。

其本质是：

- 如果两个神经元同时激活（值为 +1），它们之间的连接就会增强；
- 如果激活方向相反，则连接可能会削弱（或不改变）。

在 Hopfield 网络中，**权重矩阵** W 的更新采用 Hebbian 法则，其基本公式如下：

标准 Hebbian 更新规则（对称无自连接）：

$$W_{ij} = \sum_{\mu=1}^{P} x_i^{(\mu)} x_j^{(\mu)} \quad \text{for } i \neq j, \quad W_{ii} = 0$$

其中：

- W_{ij} 是从节点 j 到节点 i 的连接权重；
- $x^{(\mu)} \in \{-1, +1\}^N$ 是第 μ 个要记忆的模式（共 P 个模式）；
- 所有权重矩阵 W 是对称的（即 $W_{ij} = W_{ji}$）；
- 自连接被禁止（即 $W_{ii} = 0$）；

Hopfield神经网络

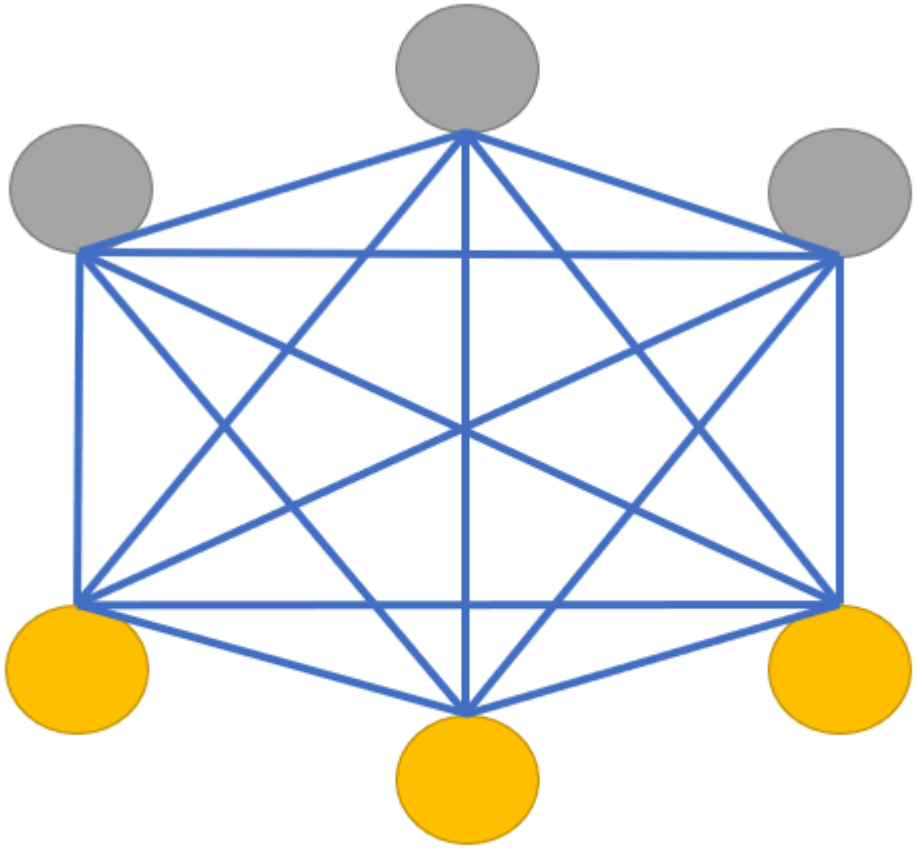
波尔曼兹机/受限波尔曼兹机

玻尔兹曼机(BM)

玻尔兹曼机(Boltzman Machine)是一个随机动力系统，每个变量的状态都以一定的概率受到其他变量的影响。

玻尔兹曼机可以用概率无向图模型来描述，一个具有K个节点的玻尔兹曼机满足以下三个性质：

1. 二值化。每个节点的状态值只有0和1。
2. 一个玻尔兹曼机包括两类节点，一类是可观察的节点有N个，一类是不可观察的节点，即隐藏节点，有(K-N)个。
3. 节点之间是全连接的。每个节点都和其他节点连接。
4. 每两个变量之间的互相影响是对称的。这里的对称和上面无向其实是一个概念，说白了就是已知A点的状态值，那么求B的状态值和已知B的状态值，求A的状态值的影响是相等的。如果你还是没有理解这句话，碰巧你又了解过一点概率论的知识，那么你可以将上述理解为 $P(B|A) = P(A|B)$。



玻尔兹曼机模型

上图就是一个有六个节点的玻尔兹曼机。其中有三个可观察的节点，我已经标了黄色，还有三个不可观测的节点，即隐藏节点，我已经标了灰色。

波尔兹曼分布推导过程

玻尔兹曼机中，随机向量X的联合概率，也就是节点的状态值，是满足玻尔兹曼分布的。

> 玻尔兹曼分布是描述粒子处于特定状态下的概率，是关于状态能量E(x)与系统温度T的函数。一个粒子处于状态α的概率P(α)是关于状态能量E(x)与系统温度T的函数。

特定状态就是说，节点的状态值为1，还是0。x=α

特定状态就是说，节点状态值为1或者0时的概率。P(x=α)

状态能量就是说，粒子本身具有的能量。

玻尔兹曼分布就是计算P(x=α)时具体的概率函数与系统的状态能量E(x)和系统温度T的函数有关。具体表达式为：

$$p_i = \frac{\exp\left(-E_i/kT\right)}{\sum_{j=1}^{M}\exp\left(-E_j/kT\right)}$$

通常玻尔兹曼分布还有另一个表达式，实则是上式的等价处理，如下：

$$p(x) = \frac{1}{Z}\exp(\frac{-E(x)}{T})$$

E(x)为能量函数，T为系统温度，Z为配分函数，实则就是一个归一化因子。

从玻尔兹曼分布的定义，我们可以发现**系统的两个不同状态的概率之比仅与系统能量有关**：

$$\mathrm{P\left(a\right)/P\left(b\right)=\exp((E1-E2)/KT)}$$

在玻尔兹曼机中，配分函数Z通常很难计算，因此，联合概率分布P(x)一般通过马尔科夫链蒙特卡洛方法(MCMC方法)来做近似计算。

玻尔兹曼机采用了基于吉布斯采样的样本生成方法来训练的。

吉布斯采样

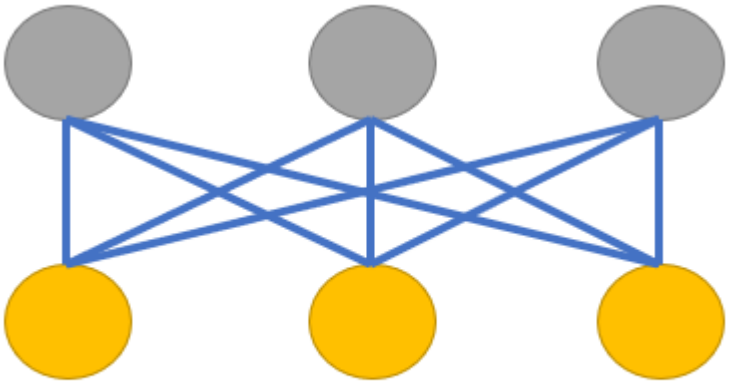
玻尔兹曼机的吉布斯采样过程为：随机选择一个变量Xi，然后根据其全条件概率P(Xi|X-i)来设置其状态值，即以P(Xi=1|X-i)的概率将变量Xi设为1，否则为0。在固定的温度T下，运行足够时间后，玻尔兹曼机会达到热平衡状态。此时，任何全局状态的概率都服从玻尔兹曼分布P(x)，只和系统的能量有关，和初始状态无关。

因为玻尔兹曼机的联合概率函数中配分函数Z很难处理，就采用另一种方法来将节点的状态值概率趋近于玻尔兹曼分布。

玻尔兹曼机可以解决两类问题，一类是搜索问题：当给定变量之间的连接权重时，需要找到一组二值向量，使得整个网络的能量最低。另一类是学习问题，当给定变量的多组观测值时，学习网络的最优权重。

受限玻尔兹曼机(RBM)

全连接的玻尔兹曼机在理论上固然有趣，但是由于其复杂性，目前为止并没有广泛运用。实际应用中，用得更多的是基于玻尔兹曼机改造的一个版本——受限玻尔兹曼机(RBM)，其网络架构如下：



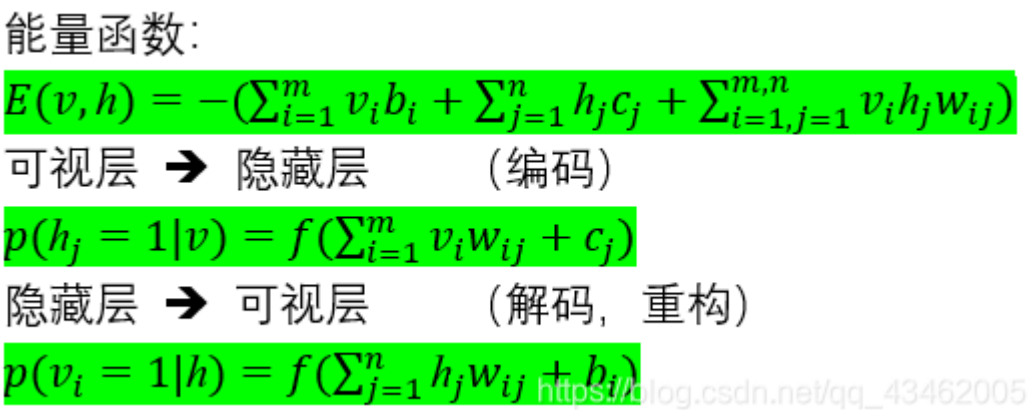
受限玻尔兹曼机模型

玻尔兹曼机没有层的概念，它所有的节点都是全连接的。

但受限玻尔兹曼机有层的概念。它有两层，一层称为显层，用于观测和输入，一层为隐藏层，用于提取特征。受限玻尔兹曼机相比玻尔兹曼机，层间的节点还是采用了对称的全连接的方式连接，但是层内的节点相互独立，相互不受影响。因为层内节点相

互独立，那么由Bayes条件独立定理，受限玻尔兹曼机**可以并行地对所有的显层变量或隐藏层变量同时进行采样，从而更快达到热平衡**。

能量函数



由于受限玻尔兹曼机变成了层的结构，所以受限玻尔兹曼机的能量函数也变成了由三部分组成，一个是显层节点偏置乘以显层随机可观测变量部分，一个是连接权重与显层随机可观测变量和隐层随机可观测变量相乘部分，一个是隐层节点偏置乘以隐层随机可观测变量偏置部分。

CD学习算法

由于受限玻尔兹曼机的特殊结构，G·Hinton提出了一种比吉布斯采样更加有效的学习算法，即对比散度学习算法，又称为CD学习算法。通过对CD学习算法的学习，我发现这个CD算法就是在吉布斯采样的基础上作出的一点改进，即在处理玻尔兹曼机时，**运行无穷次的吉布斯采样改进为运行K次即可。**以前处理玻尔兹曼机时，吉布斯采样是一直对这个玻尔兹曼机处理，直到这个波尔兹曼机收敛。G·Hinton提出，在受限玻尔兹曼机中，不需要等到受限玻尔兹曼机完全收敛，只需要K步吉布斯采样，这时模型就非常好了。所以CD算法又称K步吉布斯采样法。

受限玻尔兹曼机有两个偏置项，隐藏层的偏置项有助于RBM在前向传递中获得非零激活值，而可见层的偏置项有助于受限玻尔兹曼机学习后向传递中的重建。

在正向传递中，每个输入数据乘以一个独立的权重，然后相加后再加上一个偏置项，最后将结果传递到激活函数来产生输出。用对比散度计算正反向梯度，然后更新偏置和权重。

因为最开始受限玻尔兹曼机权重是随机初始化的，所以重建结果和原始输入差距通常会比较大，这个差距可看作是重建误差，训练受限玻尔兹曼机是通过在可见层和隐藏层之间迭代学习不断正向反向传播，直至达到某个误差的最小值。

💡 **限玻尔兹曼机（Restricted Boltzmann Machine, RBM）** 的训练目标是最大化训练数据的**对数似然（log-likelihood）**，使用的方法就是**梯度上升（Gradient Ascent）**。

下面我将为你系统地解释 **为何采用 log 似然梯度上升法，以及这个训练过程的数学原理与工程实现方式**：

🧠 一、RBM基本结构回顾

RBM 是一种两层的神经网络，由：

- **可见层（visible layer）**：$\mathbf{v} \in \{0,1\}^n$，表示输入数据；
- **隐藏层（hidden layer）**：$\mathbf{h} \in \{0,1\}^m$，学习数据的高阶特征；
- **无连接结构**：同一层内部没有连接，只有跨层连接，构成一个**二部图（bipartite graph）**。

RBM 的能量函数（Energy function）定义为：

$$E(\mathbf{v}, \mathbf{h}) = -\mathbf{v}^T \mathbf{W} \mathbf{h} - \mathbf{b}^T \mathbf{v} - \mathbf{c}^T \mathbf{h}$$

其中：

- $\mathbf{W}$：权重矩阵；
- $\mathbf{b}, \mathbf{c}$：可见层与隐藏层的偏置向量。

🎯 二、训练目标：最大化对数似然

我们希望最大化数据在模型下的概率：

$$P(\mathbf{v}) = \frac{1}{Z} \sum_{\mathbf{h}} e^{-E(\mathbf{v}, \mathbf{h})}$$

其中 $Z = \sum_{\mathbf{v}, \mathbf{h}} e^{-E(\mathbf{v}, \mathbf{h})}$ 是归一化常数（配分函数）。

训练目标是最大化**训练数据的 log 概率**（对数似然）：

$$\mathcal{L} = \log P(\mathbf{v})$$

我们对参数 $\theta = \{\mathbf{W}, \mathbf{b}, \mathbf{c}\}$ 求导，得到梯度（以权重为例）：

$$\frac{\partial \log P(\mathbf{v})}{\partial W_{ij}} = \mathbb{E}_{P(h_j|v_i)}[v_i h_j] - \mathbb{E}_{P(v_i, h_j)}[v_i h_j]$$

- 第一个期望是**正相位（positive phase）**，基于真实数据；
- 第二个期望是**负相位（negative phase）**，基于模型生成的数据。

🔁 三、梯度上升公式

RBM 的梯度更新式：

$$\Delta W_{ij} = \eta \left(\langle v_i h_j \rangle_{\text{data}} - \langle v_i h_j \rangle_{\text{model}} \right)$$

- η：学习率；
- $\langle \cdot \rangle_{\text{data}}$：真实数据期望；
- $\langle \cdot \rangle_{\text{model}}$：模型生成期望。

由于模型期望计算复杂（涉及全空间求和），我们通常采用**近似法 —— 对比散度（Contrastive Divergence, CD-k）**。

🔄 四、Contrastive Divergence（CD）简述

CD-k 是训练 RBM 最常用的近似方法：

1. 采样隐藏层：从数据出发，计算 $P(h|v^{(0)})$；
2. 采样重构：从 $h^{(0)}$ 采样 $v^{(1)}$，再继续采样 $h^{(1)}$；
3. 更新参数：使用 $v^{(0)}, h^{(0)}$ 估计正相位，使用 $v^{(k)}, h^{(k)}$ 估计负相位。

更新公式依然是：

$$\Delta W_{ij} = \eta \left(v_i^{(0)} h_j^{(0)} - v_i^{(k)} h_j^{(k)} \right)$$

CD-1 通常已经效果不错。

深度信念网络与深度玻尔兹曼机

Sigmoid信念网

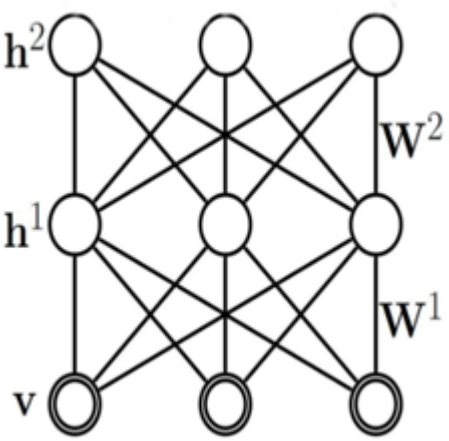
首先是讲解了Sigmoid Belief Network 的思想来源，将无向图变成有向图的结构则有更好的因果(causal) 形式，其中未观察变量被看作观察变量发生的原因。然后介绍了模型的表示方法。紧接着在模型求解的过程中，发现由于D-Separation 中的Head to Head 问题，会造成节点之间的关系复杂，无法用条件独立性分解，也就是Explain away 现象。这样后验分布的精确计算是intractable 的。所以，Neal 提出了基于MCMC 的求解方法，但是MCMC 只能求解小规模的图，大规模的图中会出现Mixing time 过长的问题，根本就不work。

这时诞生了Wake-Sleep 算法来近似推断，其主要思想就是用一个简单的分布来近似后验分布。为了解决Explain away 现象，采用的是反过来求的思路，将Head to Head 变成Tail to Tail 就可以解决这个问题了。然后两者相互迭代，相互利用数据，来使两者的数据慢慢的逼近，这是不是有点像GAN的思想，实际上GAN 的思想就有借鉴于Wake-Sleep 算法。但是，因为KL散度并不是一个距离，所以Wake-Sleep 算法两个过程的目标函数是不一致的，算法并不收敛。这个算法是启发式的算法，而且很重要，后面很多算法的思想都可以和Wake-Sleep 算法进行对比。

深度玻尔兹曼机(DBM)

深度玻尔兹曼机实际上是由多个受限玻尔兹曼机堆栈构成，我们构建一个简单的三层深度玻尔兹曼机，就首先需要进行预训练，通过对比散度算法训练出两个受限玻尔兹曼机，对于每个受限玻尔兹曼机，都是根据可见层的数据，来学习到隐层的数据，对于第二个受限玻尔兹曼机，可以把求得的第一个隐层看作是可见层来求第二个隐层的数据，然后再combine两个受限玻尔兹曼机进行微调，通过CD算法来更新相应的数据，最后实现一个简单的三层深度玻尔兹曼机。

对于受限玻尔兹曼机每新增的隐藏层，权重都会通过迭代学习反复调整，直至该层能够逼近前一层的输入，这是贪婪的、逐层的无监督的预训练。
深度玻尔兹曼机网络结构如下：



自编码器及其变种

类型	特点	输入是否加噪	是否引入正则项	应用场景
自编码器 (AE)	基础模型	否	否	压缩、表示学习
降噪自编码器 (DAE)	抗噪性强	✅ 是	否	图像去噪、鲁棒学习
稀疏自编码器 (SAE)	稀疏性强	否	✅ 是（KL散度）	特征提取、可解释学习

自编码器

🔷 定义：

自编码器是一种无监督神经网络，其目标是**将输入数据编码到低维空间中再重构回来**，实现特征提取或数据压缩。

🔷 结构：

1. **编码器（Encoder）：**
   将输入 $\mathbf{x}$ 映射到潜在表示（latent representation）$\mathbf{z}$。

$$\mathbf{z} = f_\theta(\mathbf{x}) = \sigma(W_e\mathbf{x} + b_e)$$

2. **解码器（Decoder）：**
   将 $\mathbf{z}$ 重新映射回输入空间，产生重构结果 $\hat{\mathbf{x}}$。

$$\hat{\mathbf{x}} = g_\phi(\mathbf{z}) = \sigma(W_d\mathbf{z} + b_d)$$

3. **损失函数：**

$$\mathcal{L}_{AE} = \|\mathbf{x} - \hat{\mathbf{x}}\|^2$$

🔷 目的：

压缩表示、去噪、特征学习。

降噪自编码器

🪶 二、降噪自编码器（Denoising Autoencoder, DAE）

🔷 定义：

DAE 是自编码器的改进版本，其**输入被添加噪声**，目标是**从损坏的数据中恢复出干净数据**。

🔷 流程：

1. 将原始输入 $\mathbf{x}$ 添加噪声，得到 $\tilde{\mathbf{x}}$；
2. 编码器处理 $\tilde{\mathbf{x}}$，解码器尝试恢复原始 $\mathbf{x}$；
3. 损失函数为：

$$\mathcal{L}_{DAE} = \|\mathbf{x} - g_\phi(f_\theta(\tilde{\mathbf{x}}))\|^2$$

🔷 常见噪声类型：

- 高斯噪声；
- 掩码噪声（随机将部分输入置零）；
- 盐噪声/胡椒噪声（图像中常见）。

🔷 应用场景：

鲁棒特征学习、图像去噪、异常检测。

稀疏自编码器

💡 三、稀疏自编码器（Sparse Autoencoder, SAE）

🔷 定义：

SAE 强制隐层表示 $\mathbf{z}$ 中**大多数神经元保持不激活（近似0）**，仅少量激活，从而实现**稀疏表示学习**。

🔷 关键点：

- 对隐藏层的**激活值添加稀疏性惩罚项**（通常是 KL 散度）：

$$\mathrm{KL}(\rho \| \hat{\rho}_j) = \rho \log \frac{\rho}{\hat{\rho}_j} + (1-\rho) \log \frac{1-\rho}{1-\hat{\rho}_j}$$

其中：

- $\hat{\rho}_j$ 是第 j 个隐藏单元的平均激活；
- ρ 是稀疏目标（如 0.05）；

🔷 总损失函数：

$$\mathcal{L}_{SAE} = \|\mathbf{x} - \hat{\mathbf{x}}\|^2 + \beta \sum_j \mathrm{KL}(\rho \| \hat{\rho}_j)$$

🔷 应用：

在保持重要特征的同时去除冗余，有助于学习**更加解释性强、结构性好的特征**。

稀疏自编码器：KL散度

📘 一、KL散度的定义

KL 散度用于衡量两个概率分布之间的差异。
对于两个概率分布 P 和 Q，KL 散度定义为：

$$|D_{\mathrm{KL}}(P\|Q) = \sum_i P(i) \log \frac{P(i)}{Q(i)}$$

或者在连续形式下：

$$D_{\mathrm{KL}}(P\|Q) = \int P(x) \log \frac{P(x)}{Q(x)} dx$$

> ✅ **直观理解**：
> KL 散度衡量的是：**如果用 Q 来近似 P，会带来多少“信息损失”或“额外代价”。**

🧠 二、在稀疏自编码器中的应用

稀疏自编码器要求**隐藏层神经元的平均激活值非常小（接近 0）**，即只有少数神经元“工作”，其余“休息”，实现稀疏性。

设定目标稀疏度 ρ，如 $\rho = 0.05$，表示我们希望某个隐藏神经元仅在 5% 的样本中被激活。

对于第 j 个隐藏神经元，假设其实际平均激活值为 $\hat{\rho}_j$，则：

$$\mathrm{KL}(\rho\|\hat{\rho}_j) = \rho \log \frac{\rho}{\hat{\rho}_j} + (1-\rho) \log \frac{1-\rho}{1-\hat{\rho}_j}$$

当 $\hat{\rho}_j = \rho$ 时，KL散度最小（为 0）；
当 $\hat{\rho}_j \neq \rho$ 时，KL散度为正，表示偏离目标稀疏度，有“惩罚”。

扩散模型

原理介绍

扩散模型：和其他生成模型一样，实现从噪声（采样自简单的分布）生成目标数据样本。

扩散模型包括两个过程：前向过程（forward process）和反向过程（reverse process），其中前向过程又称为扩散过程（diffusion process）。无论是前向过程还是反向过程都是一个参数化的马尔可夫链（Markov chain），其中反向过程可用于生

成数据样本（它的作用类似GAN中的生成器，只不过GAN生成器会有维度变化，而DDPM的反向过程没有维度变化）。

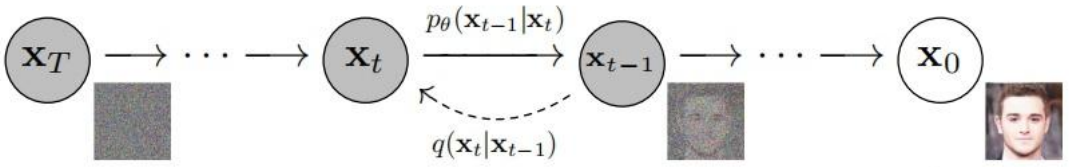


Figure 2: The directed graphical model considered in this work.

- x0到xT为逐步加噪过的前向程，噪声是已知的，该过程从原始图片逐步加噪至一组纯噪声。
- xT到x0为将一组随机噪声还原为输入的过程。该过程需要学习一个去噪过程，直到还原一张图片。

前向过程

前向过程是加噪的过程，前向过程中图像 xt 只和上一时刻的 xt−1 有关, 该过程可以视为马尔科夫过程, 满足:

$$q(x_{1:T}|x_0) = \prod_{t=1}^{T} q(x_t|x_{t-1})$$

$$q(x_t|x_{t-1}) = N(x_t; \sqrt{1-\beta_t}x_{t-1}, \beta_t I)$$

$q(x_{1:T}|x_0)$ 表示从原始图像 x_0 开始，经过 T 步加噪过程后得到噪声图像 x_T 的概率分布。

每一步 $q(x_{1:T}|x_0)$ 表示在第 t 步，给定前一步的图像 x_{t-1} ，得到当前图像 x_t 的条件概率分布，这是一个高斯分布，其均值为 $\sqrt{1-\beta_t}x_{t-1}$ ，方差为 β_t ，其中 β_t 是一个预定的噪声方差系数， I 是单位矩阵。

其中不同 t 的 β_t 是预先定义好的，由时间 $1 \sim T$ 逐渐的递增，可以是Linear，Cosine等，满足 :$\beta_1 < \beta_2 < \ldots < \beta_T$ 。

根据以上公式，可以通过重参数化采样得到x_t。 $\epsilon \sim N(0, I)$ ， $\alpha_t = 1 - \beta_t$;

接着定义$\overline{\alpha}_t = \Pi_{i=1}^{T} \alpha_i$ ，经过推导，可以得出 x_t 与x_0 的关系：

$$q(x_t|x_0) = N(x_t; \sqrt{\overline{\alpha}_t}x_0, (1-\overline{\alpha}_t)I)$$

逆向过程

逆向过程是去噪的过程，如果得到逆向过程 $q(x_{t-1}|x_t)$，就可以通过随机噪声x_T 逐步还原出一张图像。DDPM使用神经网络 $p_\theta(x_{t-1}|x_t)$ 拟合逆向过程 $q(x_{t-1}|x_t)$ 。

$q(x_{t-1}|x_t,x_0)=N(x_{t-1}|\tilde{\mu_t}(x_t,x_0),\tilde{\beta}_tI)$ ，可以推导出:

$p_\theta(x_{t-1}|x_t)=N(x_{t-1}|\mu_\theta(x_t,t),\Sigma_\theta(x_t,t))$

DDPM 论文中通过神经网络（通常是 U-Net）拟合噪声预测模型 $\epsilon_\theta(,)$ ，从而间接得到均值 μ_θ，以计算 x_{t-1} 。方差通常是固定的，并未直接通过神经网络拟合，而是由扩散过程的参数决定。（修订后）

$\mu_\theta=\frac{1}{\sqrt{\alpha_t}}(x_t-\frac{1-\alpha_t}{\sqrt{1-\overline{\alpha_t}}}\epsilon_{\theta(x_t,t)})$

因为 t 和 x_t 已知，只需使用神经网络拟合 $\epsilon_{\theta(x_t,t)}$ 。

网络结构

论文的源代码采用Unet+实现 $\epsilon_{\theta(x_t,t)}$ 的预测，整个训练过程其实就是在训练Unet网络的参数

Unet职责

无论在前向过程还是反向过程，Unet的职责都是根据当前的样本和时间t预测噪声。

.

Gaussion Diffusion职责

前向过程：从1到T的时间采样一个时间 t ，生成一个随机噪声加到图片上，从Unet获取预测噪声，计算损失后更新Unet梯度

反向过程：先从正态分布随机采样和训练样本一样大小的纯噪声图片，从T-1到0逐步重复以下步骤：从 x_t 还原 x_{t-1} 。