《计算机模拟》

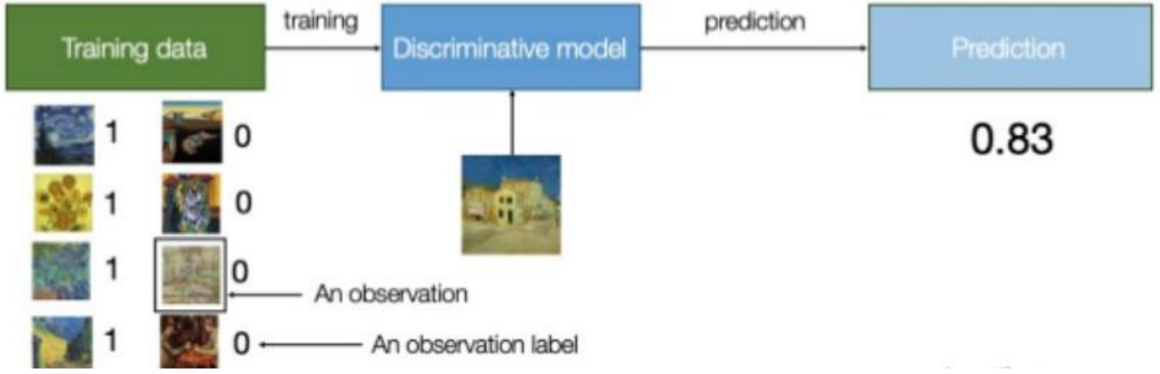
第12讲 –生成式模拟(模型)

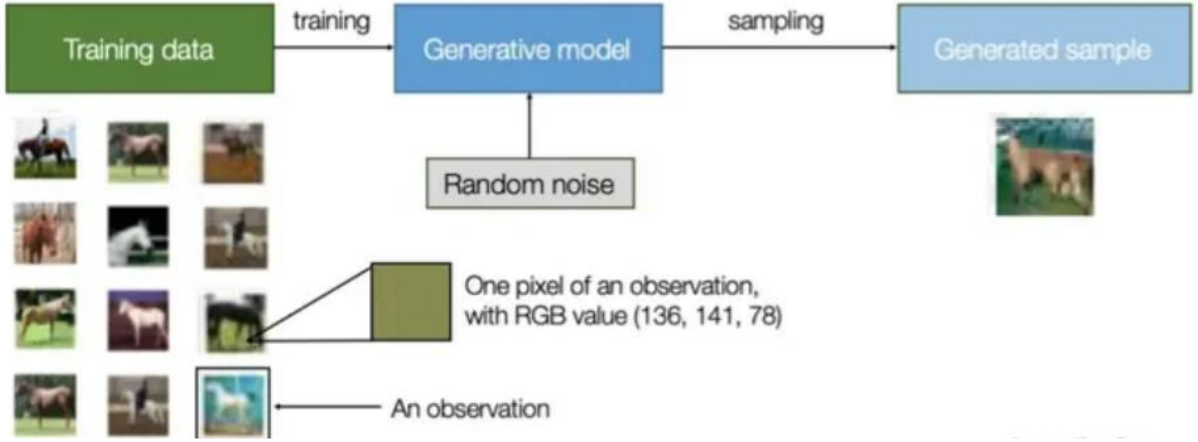
胡贤良

浙江大学数学科学学院

1. 生成式模拟(模型)

判别模型 & 生成模型



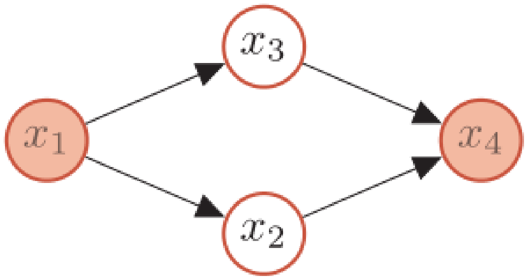


1. 生成模型的数据集一般没有和判别模型类似的标记
2. 生成模型也可以有标签，根据标签去生成相应类别的图像
3. 生成模型像是一种非监督学习，而判别模型是一种监督学习
4. 数学表示不同：
   - 判别模型： p(y|x) 即给定观测x得到y的概率。
   - 生成模型：p(x) 即观测x出现的概率。如果有标签则表示为: p(x|y) 指定标签y生成x的概率

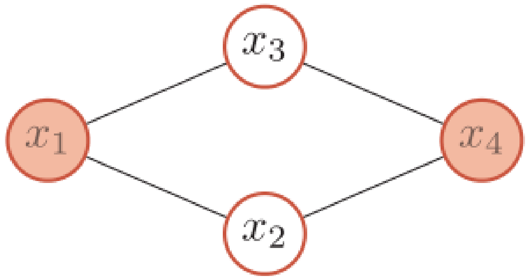
- 一个生成数据的模型，属于一种概率模型
- 通过学习，生成的数据分布$p(x)$
- 解决三类问题：
  - 密度估计(Training)
  - 特征推断(Inference)
  - 生成新的数据(Sampling)

概率图模型

一种用图结构来描述多元随机变量之间条件独立关系的概率模型



(a) 有向图：贝叶斯网络



(b) 无向图：马尔可夫随机场

- 每个节点都对应一个随机变量，可以是观察变量、隐变量或未知参数
- 每个连接表示两个随机变量之间具有依赖关系

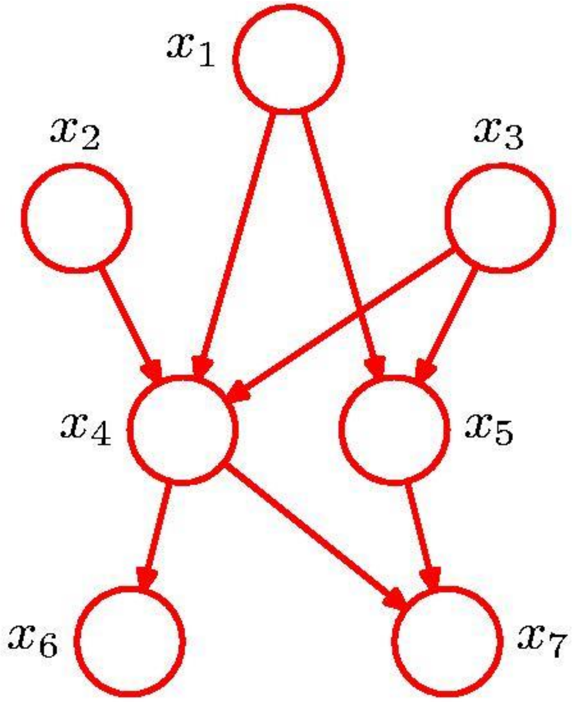
有向(Directed)图模型

也称为贝叶斯网络（Bayesian Network），或信念网络（Belief Network）

定义 11.1 – 贝叶斯网络： 对于一个 K 维随机向量 $\boldsymbol{X}$ 和一个有 K 个节点的有向非循环图 G，G 中的每个节点都对应一个随机变量，每个连接 e_{ij} 表示两个随机变量 X_i 和 X_j 之间具有非独立的因果关系. 令 $\boldsymbol{X}_{\pi_k}$ 表示变量 X_k 的所有父节点变量集合，$P(X_k|\boldsymbol{X}_{\pi_k})$ 表示每个随机变量的局部条件概率分布（Local Conditional Probability Distribution）. 如果 $\boldsymbol{X}$ 的联合概率分布可以分解为每个随机变量 X_k 的局部条件概率的连乘形式，即

$$p(\boldsymbol{x}) = \prod_{k=1}^{K} p(x_k|\boldsymbol{x}_{\pi_k}), \tag{11.8}$$

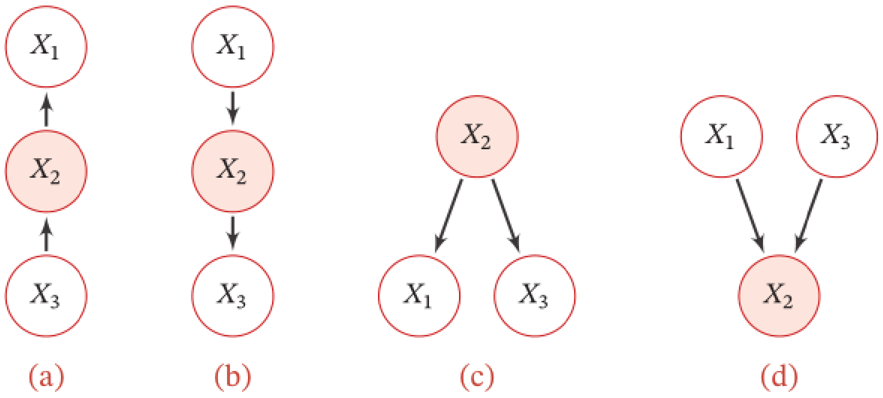
那么 $(G, \boldsymbol{X})$ 构成了一个贝叶斯网络.



$$\begin{aligned} p(x_1, \ldots, x_7) \quad = \quad & p(x_1)p(x_2)p(x_3)p(x_4|x_1, x_2, x_3) \\ & p(x_5|x_1, x_3)p(x_6|x_4)p(x_7|x_4, x_5) \end{aligned}$$

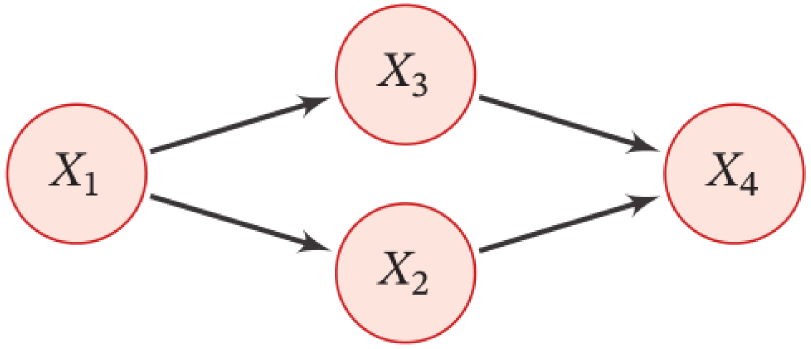
贝叶斯网络的性质

1. 条件独立性：如果两个节点是直接连接的，它们肯定是非条件独立的， 是直接因果关系：其中，父节点是“因”，子节点是“果”。



2. 局部马尔可夫性质：每个随机变量在给定父节点的情况，条件独立于它的非后代节点。

- 利用局部马尔可夫性可以简化多元变量的联合概率，从而降低建模的复杂度。如右图所示的图，联合概率是4个局部条件概率的乘积，这样只需要1 + 2 + 2 + 4 = 9个独立参数。



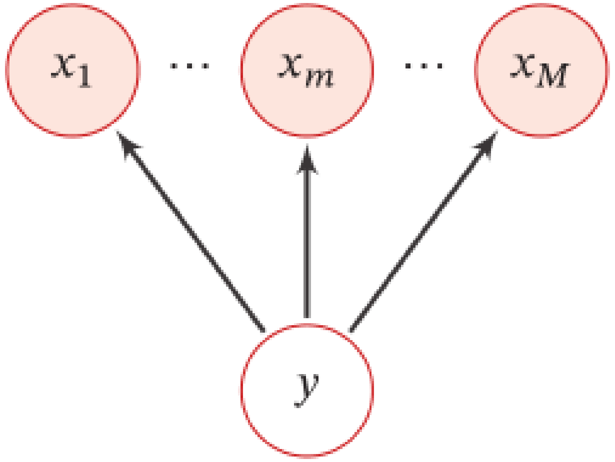
$$
\begin{aligned}
p(\boldsymbol{x}) &= p(x_1)p(x_2|x_1)p(x_3|x_1,x_2)p(x_4|x_1,x_2,x_3), \\
&= p(x_1)p(x_2|x_1)p(x_3|x_1)p(x_4|x_2,x_3),
\end{aligned}
$$

常见的有向图模型

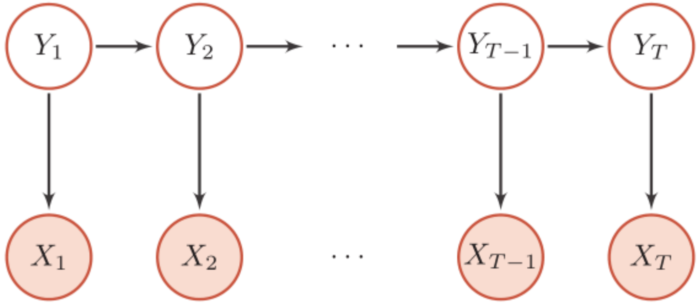
1. 朴素贝叶斯分类器

➢给定一个有M维特征的样本x和类

$$p(y|\mathbf{x};\theta) \propto p(y|\theta_c)\prod_{m=1}^{M} p(x_m|y;\theta_m)$$



2. Hidden Markov Model(HMM)



➢HMM联合概率可以分解为

$$p(\mathbf{x},\mathbf{y};\theta) = \prod_{t=1}^{T} p(y_t|y_{t-1},\theta_s)p(x_t|y_t,\theta_t)$$

转移概率 输出概率

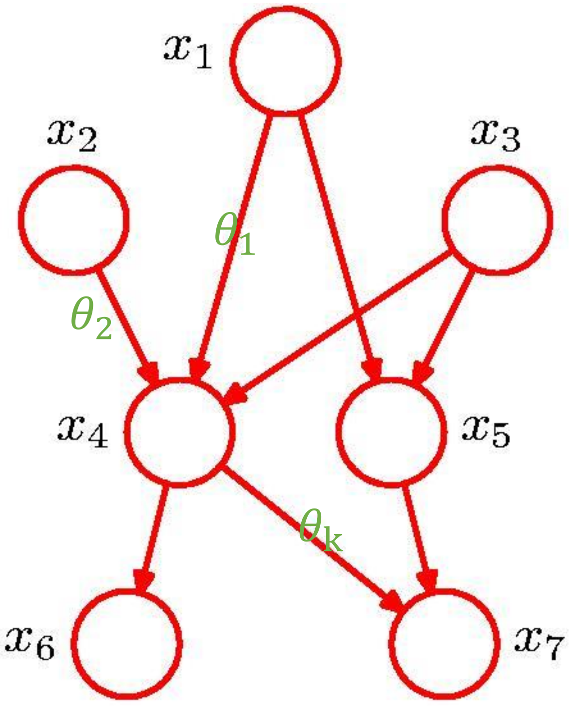
有向图模型的学习/训练

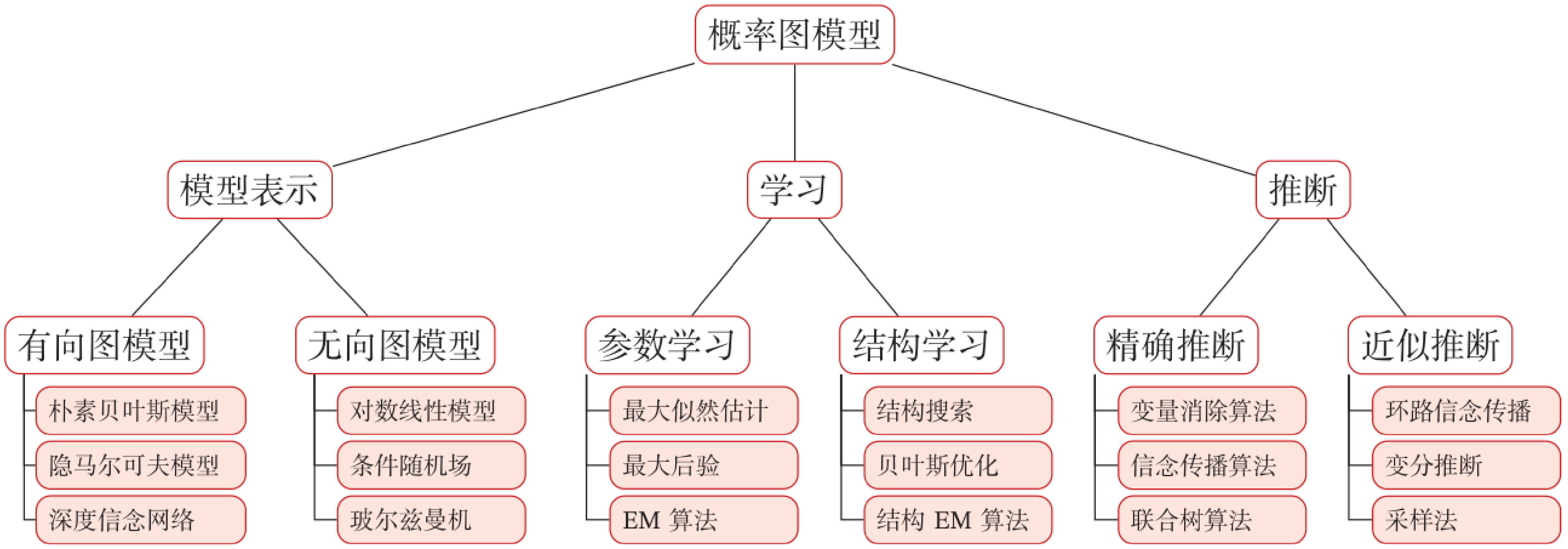
- 在贝叶斯网络中，所有变量x的联合概率分布可以分解为每个随机变量x_k的局部条件概率的连乘形式。
- 假设每个局部条件概率$p(x_k\,|x_{\pi_k})$的参数为θ_k，则对数似然函数为

$$\begin{aligned}\mathcal{L}(\mathcal{D};\theta) &= \frac{1}{N}\sum_{n=1}^{N}\log p(\boldsymbol{x}^{(n)};\theta)\\ &= \frac{1}{N}\sum_{n=1}^{N}\sum_{k=1}^{K}\log p(x_k^{(n)}|x_{\pi_k}^{(n)};\theta_k),\end{aligned}$$

- 分别最大化每个变量的条件似然来估计其参数

$$\theta_k = \arg\max\sum_{n=1}^{N}\log p(x_k^{(n)}|x_{\pi_k}^{(n)};\theta_k)$$





通过学习给定数据的特征，获得其近似的概率分布，并用学习所得近似来**生成**新的数据。常见的有：

- 朴素Bayes方法：常用于分类问题，计算出数据属于某个类别的概率
- Gauss混合模型：常用于聚类问题，计算数据属于多个Gauss分布的概率
- 隐Markov模型：常用于序列建模，计算状态间或状态与数据间转移概率
- 变分自编码器：常用于无监督学习(特征提取)，计算数据的“编码(特征)”
- **生成对抗网络**：深度神经网络时代的产品，通过求解minmax问题得到数据的概率分布

基于概率图的推断

主要问题：如何计算边际概率

$$p(y) = \sum_{x'} p(y|x')p(x')$$

贝叶斯公式： 对于给定的x，分布z关于x的后验分布可以表示为：

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

引入全概率公式$p(x) = \int p(x|z)p(z)dz$，可以进一步表示为：

$$p(z|x) = \frac{p(x|z)p(z)}{p(x)} = \frac{p(x|z)p(z)}{\int p(x|z)p(z)dz}$$

① 精确推断

- 变量消去法 $p(x_1, x_4) = \sum_{x_2, x_3} p(x_1)p(x_2|x_1)p(x_3|x_1)p(x_4|x_2, x_3) = p(x_1)\sum_{x_3} p(x_3|x_1)\sum_{x_2} p(x_2|x_1)p(x_4|x_2, x_3)$
- 信念传播：在具有环路的图上依然使用和积算法，近似精确！
- 联合树算法

② 近似推断

- 采样法（MonteCarlo）通过模拟的方式来采集符合某个分布p(x)的一些样本，并估计相关量
- 变分推断：引入一个比较简单的变分分布来近似条件概率，然后通过迭代的方法进行计算

变分推断

$$argmax_Q\left(\int Q(Z)log\big(P(X,Z)\big)dZ-\int Q(Z)log\big(Q(Z)\big)dZ\right)$$

对于需要进行训练的隐含变量Z和观察值变量$X=\{x_1,x_2,...,x_N\}$，作出合理的模型假设P(Z,X)；构造关于Z的概率分布Q$(Z;V)$，不断更新Q的参数V，使得Q$(Z;V)$接近P$(Z|X)$；给出优化问题为最小化后验与近似分布的KL散度。

由公式：$p(X)=\frac{p(Z,X)}{p(Z|X)}$，等式两边同时取对数有：

$$log(P(X))=log(P(Z,X))-log(P(Z|X))$$

引入$Q(Z)$，得:

$$log\big(P(X)\big)=log\big(P(Z,X)\big)-log\big(Q(Z)\big)-\Big(log\big(P(Z|X)\big)-log\big(Q(Z)\big)\Big)=log\left(\frac{P(Z,X)}{Q(Z)}\right)-log\left(\frac{P(Z|X)}{Q(Z)}\right)$$

关于$Q(Z)$取期望：

$$\int Q(Z)log\big(P(X)\big)dZ=\int Q(Z)log\left(\frac{P(Z,X)}{Q(Z)}\right)dZ-\int Q(Z)log\left(\frac{P(Z|X)}{Q(Z)}\right)dZ$$

$$=\underbrace{\boxed{\int Q(Z)log(P(Z,X))\,dZ-\int Q(Z)log(Q(Z))\,dZ}}_{\text{ELOB L(V)}}-\underbrace{\boxed{\int Q(Z)log(\frac{P(Z|X)}{Q(Z)})dZ}}_{KL(Q(Z)||P(Z|X))}$$

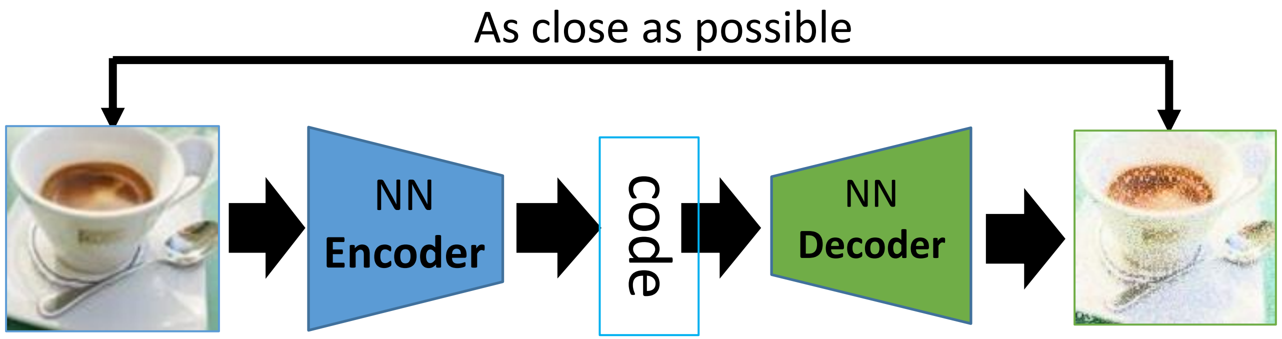
优化问题:最小化KL距离 ⟶ $P(Z|X)$难以计算 ⟶ 优化问题:最大化ELBO（上界为$log(P(X))$）

生成模型实例：自编码器(编码器 + 解码器)

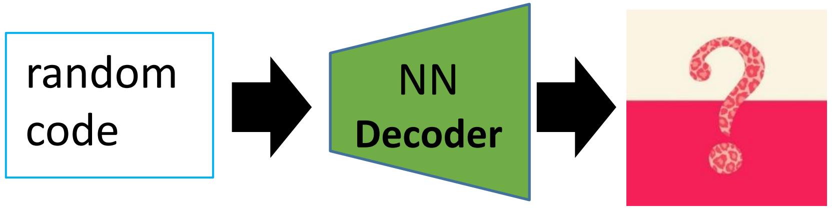
- 一种典型的生成模型，通过无监督学习使输出尽可能地和输入相同，从而学习数据的分布
- 工作流程：
  - 利用编码器 对x进行编码，即

    h = f(x)，

    形成所对应的编码 z
  - 运用码器进行解码，获得重建向 量x ′ = g(f(x))



“压缩特征”？



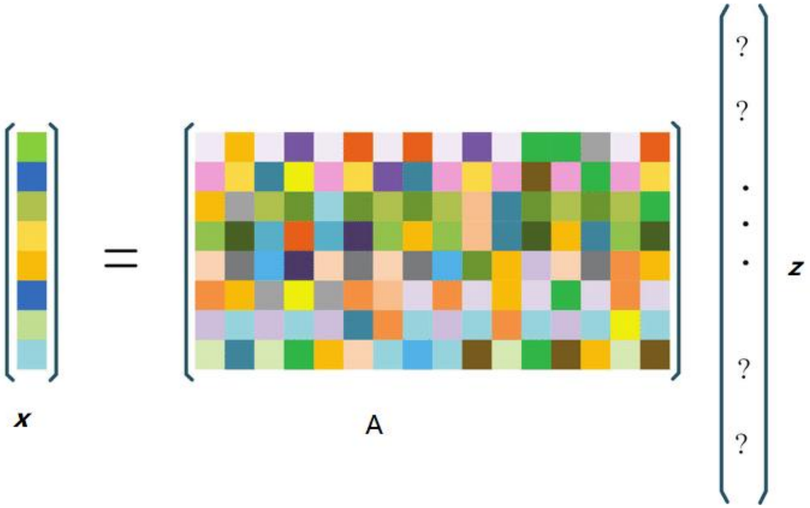
- 学习目标(损失函数)： min L(x, x′)
  - 编码器的目标是使得： x′ ≈ x --- 重建、降维、不相等！
- (线性)编码:

$$\boldsymbol{x} = \sum_{m=1}^{M} z_m \boldsymbol{a}_m$$

$$= \boldsymbol{A}\boldsymbol{z},$$

字典（dictionary）

编码（encoding）

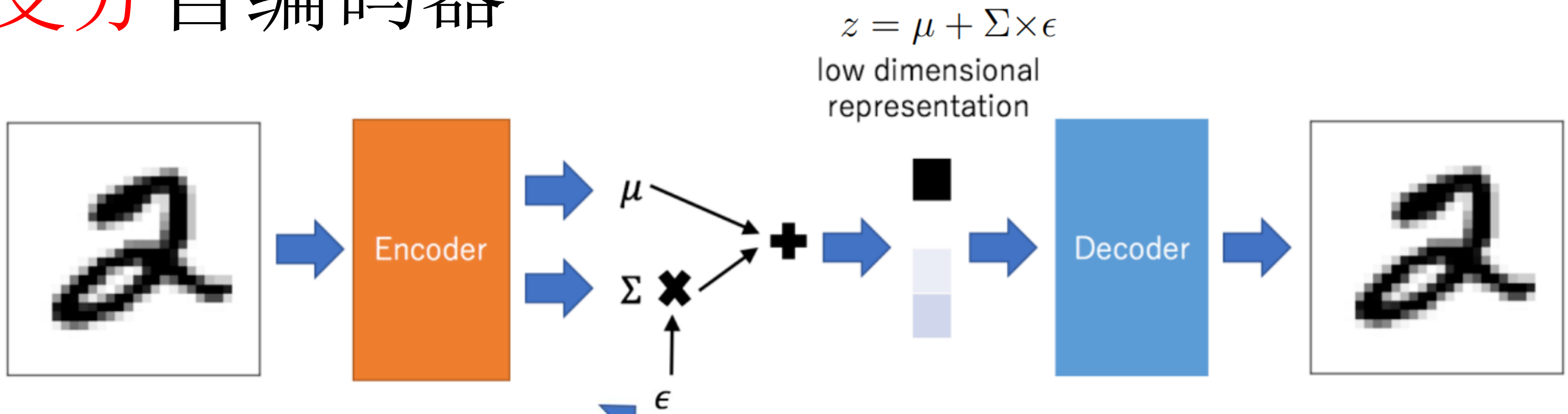


自编码器 – PyTorch实现示例一瞥

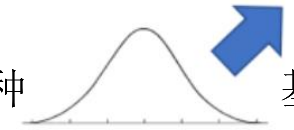
```python
class autoencoder1(nn.Module):
    def __init__(self,in_features,hiden_features):
        super(autoencoder1, self).__init__()
        self.encoder = nn.Sequential(nn.Linear(in_features, hiden_features), nn.ReLU(True))
        self.decoder = nn.Sequential(nn.Linear(hiden_features, in_features), nn.Sigmoid())

    def forward(self, x):
        en = self.encoder(x)
        de = self.decoder(en)
        return en, de
```

```python
class autoencoder2(nn.Module):
    def __init__(self,in_features,hiden_features):
        super(autoencoder2, self).__init__()

        self.encoder = nn.Sequential(
                            nn.Conv2d(in_features, hiden_features, kernel_size = 3, padding = 1),
                            nn.ReLU(True),   nn.MaxPool2d(kernel_size = 2),   nn.ReLU(True))
        self.decoder = nn.Sequential(
                            nn.ConvTranspose2d(hiden_features, hiden_features, kernel_size = 2,stride=2),
                            nn.ReLU(True),
                            nn.ConvTranspose2d(hiden_features, in_features, kernel_size=3, padding = 1),
                            nn.Tanh())

    def forward(self, x):
        en = self.encoder(x)
        de = self.decoder(en)
        return en, de
```

变分自编码器



- VAE提出于2013年,是一种基于神经网络的概率模型, 任务是学习一个概率分布。损失函数取为：



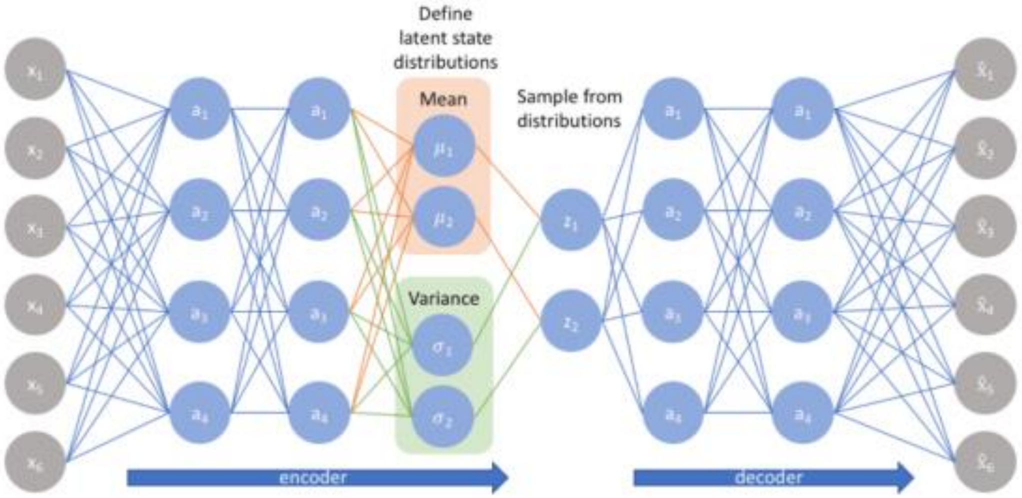
$$\mathcal{L}\left(\theta, \phi; x^{(i)}\right) \simeq \frac{1}{2} \sum_{j=1}^{J} \left(1 + \log\left(\left(\sigma_j^{(i)}\right)^2\right) - \left(\mu_j^{(i)}\right)^2 - \left(\sigma_j^{(i)}\right)^2\right) + \frac{1}{L} \sum_{l=1}^{L} \log p_\theta\left(x^{(i)} \mid z^{(i,l)}\right),$$

- VAE 在普通的 AE 中加入了一些限制，要求产生的隐变量遵循高斯分布（理论上可以是任意分布）
  - 用来衡量输出与输入之间的差距
  - 用KL散度来表示q(z|x)和p(z)之间的相似程度

➢ 目的：使得网络能够在不给定原始图像的基础上生成新的图像

➢ 是一类基于自编码器结构和变分贝叶斯推断的深度生成模型。

VAE实现概览




```python
class VAE(nn.Module):
    def __init__(self, in_features, hiden_features, out_features, n_num):
        super(VAE, self).__init__()

        self.encoder = nn.Sequential(nn.Linear(in_features, hiden_features),nn.ReLU(True))
        self.mean=nn.Linear(hiden_features, n_num)
        self.logvar=nn.Linear(hiden_features, n_num)

        self.decoder = nn.Sequential(nn.Linear(n_num, hiden_features),nn.ReLU(True),nn.Linear(hiden_features,
            out_features),nn.Tanh())

    def forward(self, x):
        x = self.encoder(x)
        x_mean=self.mean(x)
        x_logvar=self.logvar(x)
        std = 0.5 * torch.exp(x_logvar)
        z = torch.randn(std.size()) * std + x_mean
        z = self.decoder(z)
        return z, x_mean, x_logvar
```

```python
class VAE_Loss(_Loss):
    def __init__(self, reduction='sum'):
        super(VAE_Loss, self).__init__()
        self.bce_loss = torch.nn.BCELoss(reduction = reduction)

    def forward(self, input, target, x_mean, x_logvar):
        loss1 = self.bce_loss(input, target)
        loss2 = self.KL_divergence(x_mean, x_logvar)
        return loss1 + loss2

    def KL_divergence(x_mean, x_logvar):
        return - 0.5 * torch.sum(1 + x_logvar - torch.exp(x_logvar) - x_mean**2)
```

变分下界（Evidence Lower Bound,ELOB）

$$L_b = \int q(z|x)\ log\frac{p(x,z)}{q(z|x)}dz$$
$$= \int q(z|x)\ log\frac{p(z)p(x|z)}{q(z|x)}dz$$
$$= \int q(z|x)\ log\frac{p(z)}{q(z|x)}dz + \int q(z|x)\ log\ p(x|z)dz$$
$$= -\int q(z|x)\ log\frac{q(z|x)}{p(z)}dz + \int q(z|x)\ log\ p(x|z)dz$$
$$= -D_{KL}(q(z|x)||p(z)) + E_{z\sim q(z|x)}log\ p(x|z)$$

两个正态分布之间的 KL 散度为:

$$D_{KL}(N(\mu_1,\sigma_1^2)||N(\mu_2,\sigma_2^2)) = log\frac{\sigma_2}{\sigma_1} + \frac{\sigma_1^2+(\mu_1-\mu_2)^2}{2\sigma_2^2} - \frac{1}{2}$$

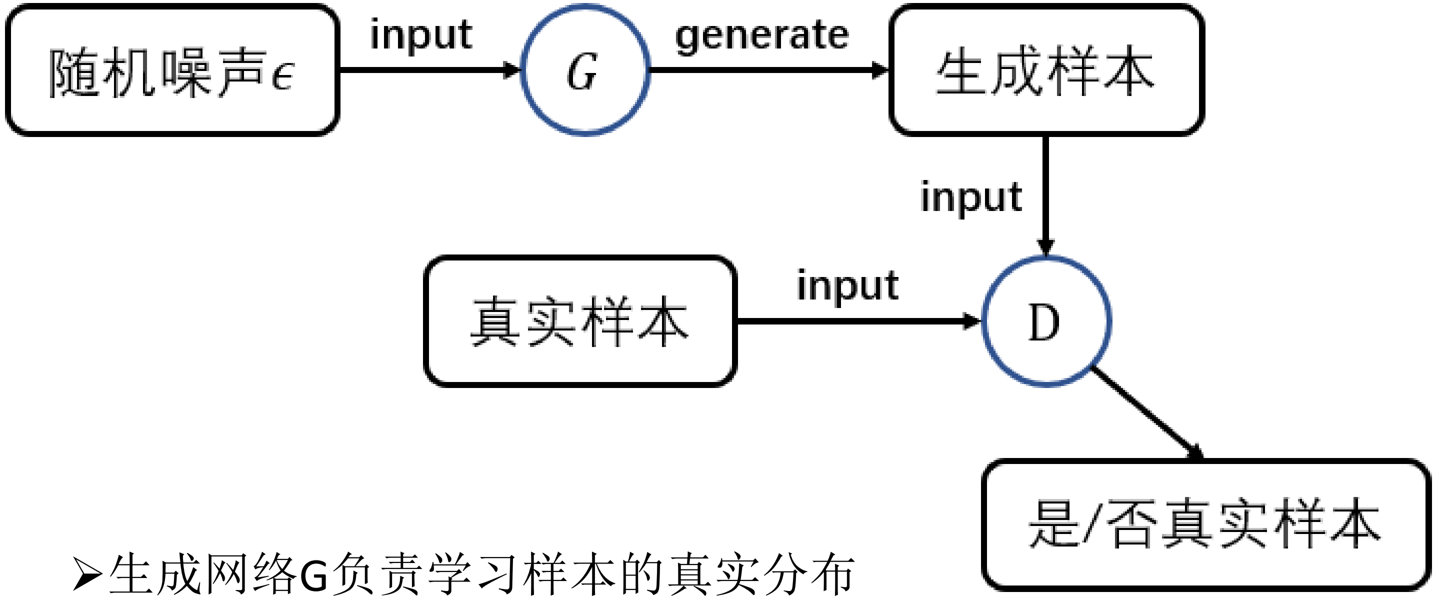
其中 $p(z)\sim N(0,1)$，定义 $p(z|x)\sim N(\mu,\sigma^2)$，则

$$D_{KL}(q(z|x)||p(z)) = -\frac{1}{2}(log\sigma^2-\sigma^2-\mu^2+1)$$

- 此损失函数需要权衡两部分：

1. 计算编码器生成的 z 的分布 q(z|x) 是否与真实分布 p(z)（高斯分布）足够接近，即 隐变量 z 是否服从高斯分布。
2. 最大化Ez~q(z|x) log p(x|z)，即保证生成数据的准确性。

2 GAN

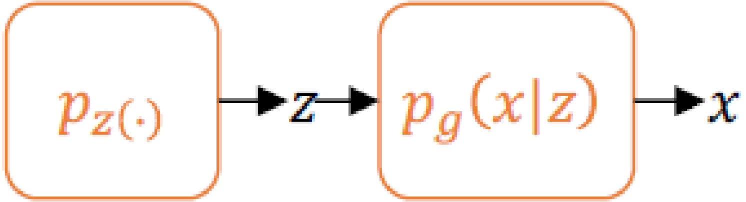


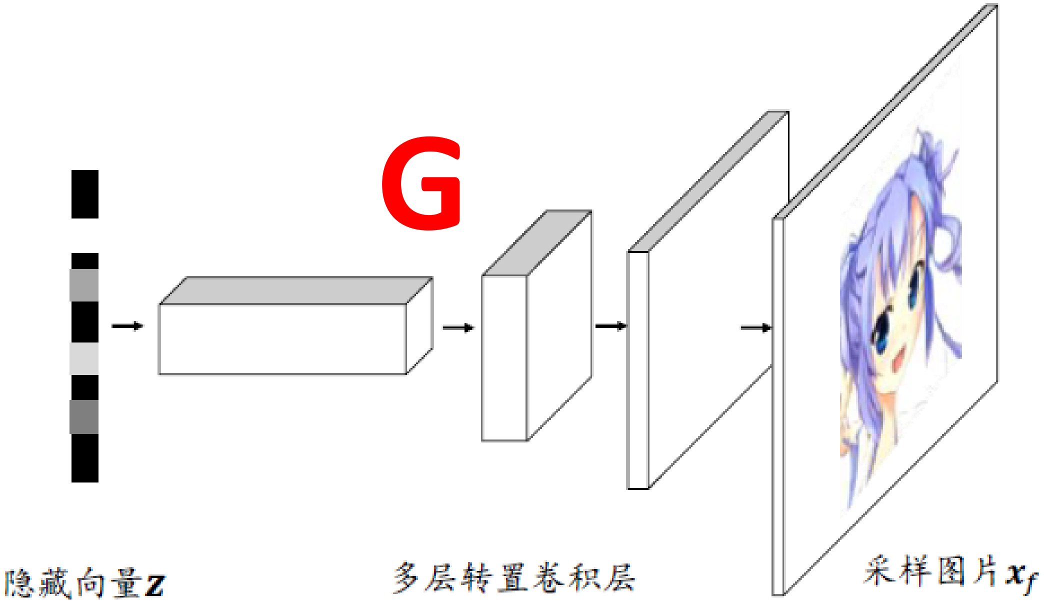
- 生成网络G负责学习样本的真实分布
- 判别网络D负责将生成网络采样的样本与真实样本区分开来

"Generative Adversarial Networks." Ian J. Goodfellow, Jean Pouget-Abadie, Mehdi Mirza, Bing Xu, David Warde-Farley, Sherjil Ozair, Aaron Courville, Yoshua Bengio. ArXiv 2014. https://github.com/goodfeli/adversarial

生成网络(Generator)

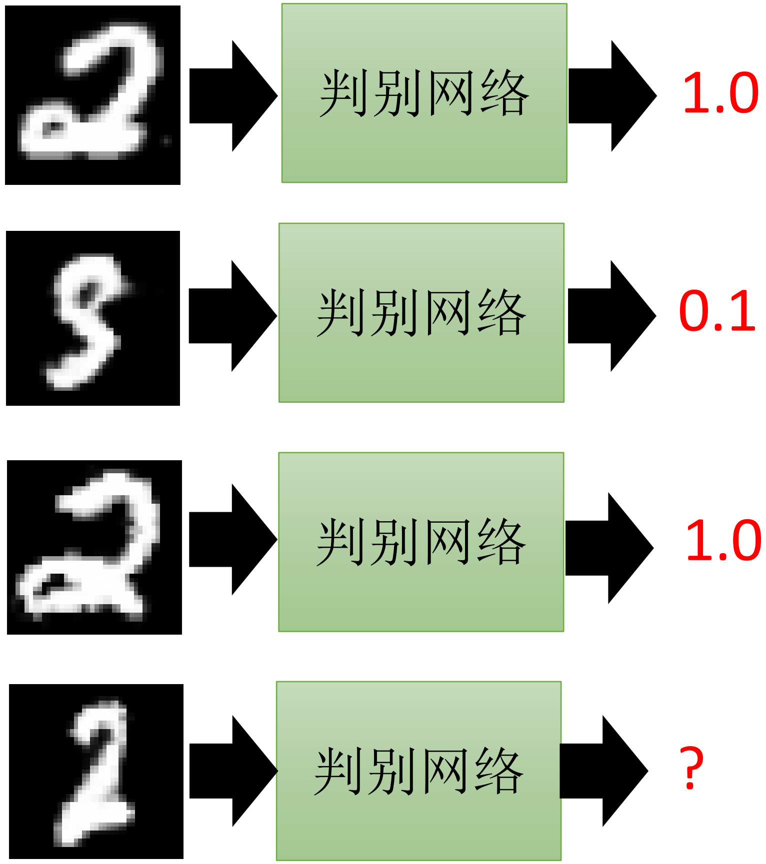
- 与自编码器的Decoder功能类似，从先验分布$p_z(\cdot)$ 中采样隐藏变量$x \sim p_z(\cdot)$：通过生成网络G参数化的 $p_g(x|z)$ 分布获得生成样本 $x \sim p_g(x|z)$





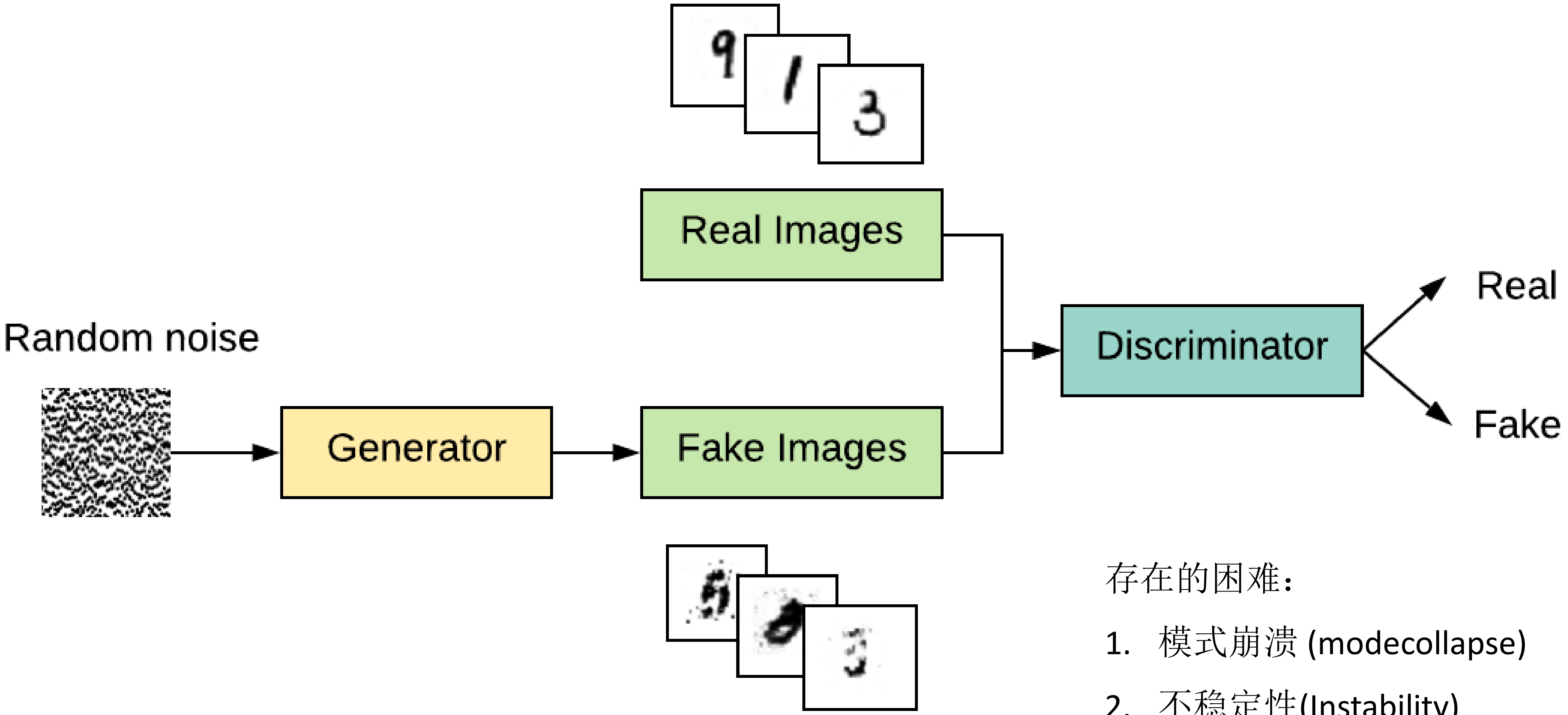
判别网络(Discriminator)

- 本质上是一个二分类网络：接受输入样本x，包含了采样自真实数 据分布$p_r(\cdot)$的样本$x_r \sim p_r(\cdot)$，也包含了采样自生成网络的假样本$x_f \sim p_g(x|z)$， x_r和x_f共 同组成了判别网络的训练集
  1. 把所有真实样本x_r的标签标注为真(1)
  2. 所有生成网络产生的样本x_f标注为假(0)
- 通过最小化判别网络预测值与标签之间的误差来优化判别网络参数. 即所谓：判别网络的输入则为真实样本或生成网络的输出，其目的是将生成网络的输出从真实样本中尽可能分辨出来。



1. 手写数字生成器

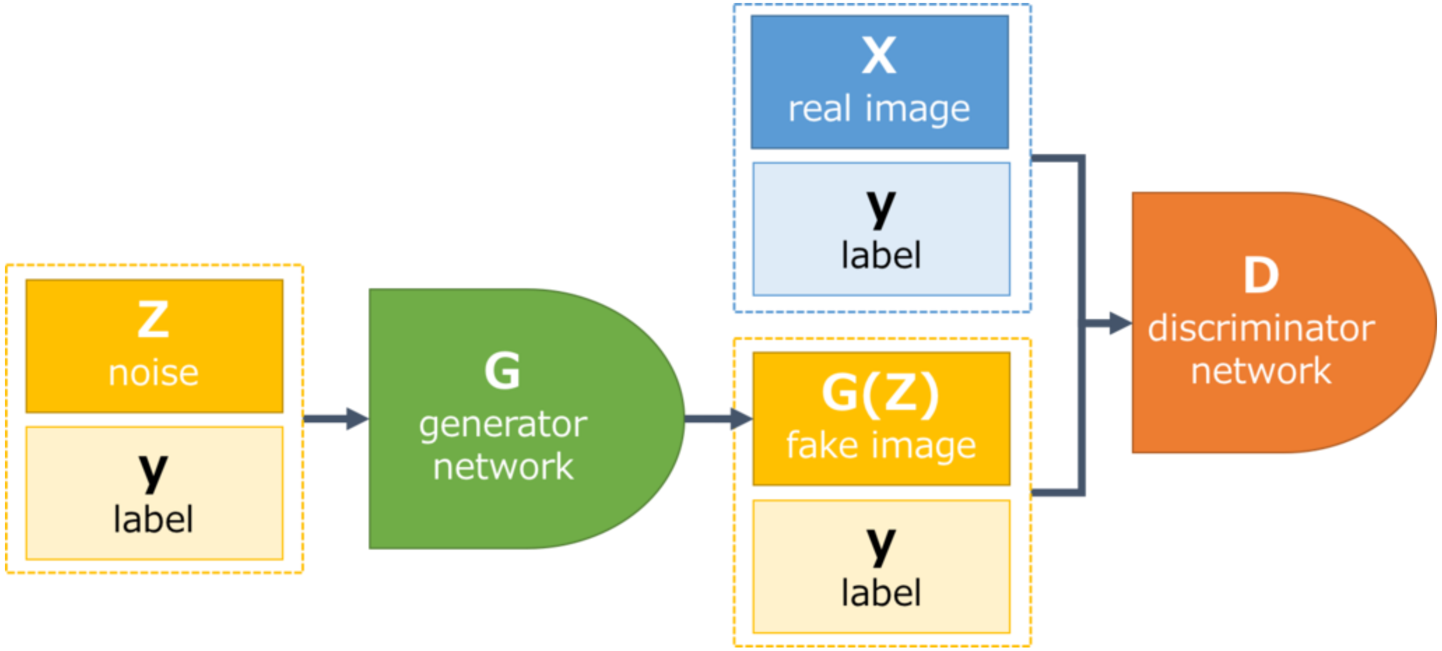
参考 demo_gan_mnist.ipynb

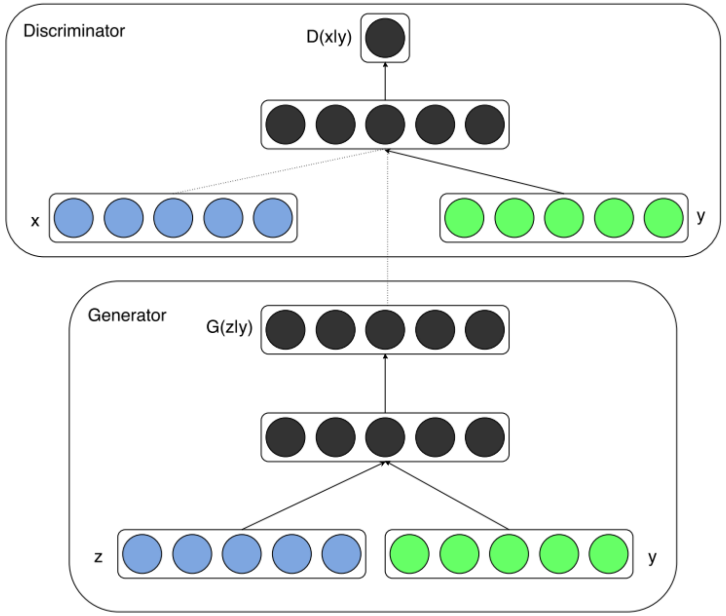


存在的困难：

1. 模式崩溃 (modecollapse)
2. 不稳定性(Instability)

2. Conditional GAN





- demo_cgan_mnist.ipynb
- 需要训练很长时间！

潜在应用：条件生成

- 根据条件针对性的生成数据

$$\begin{bmatrix} 0.3 \\ -0.1 \\ \vdots \\ -0.7 \end{bmatrix} \begin{bmatrix} 0.1 \\ -0.1 \\ \vdots \\ 0.7 \end{bmatrix} \begin{bmatrix} -0.3 \\ 0.1 \\ \vdots \\ 0.9 \end{bmatrix} \rightarrow \boxed{\text{生成网络}} \rightarrow$$

In a specific range

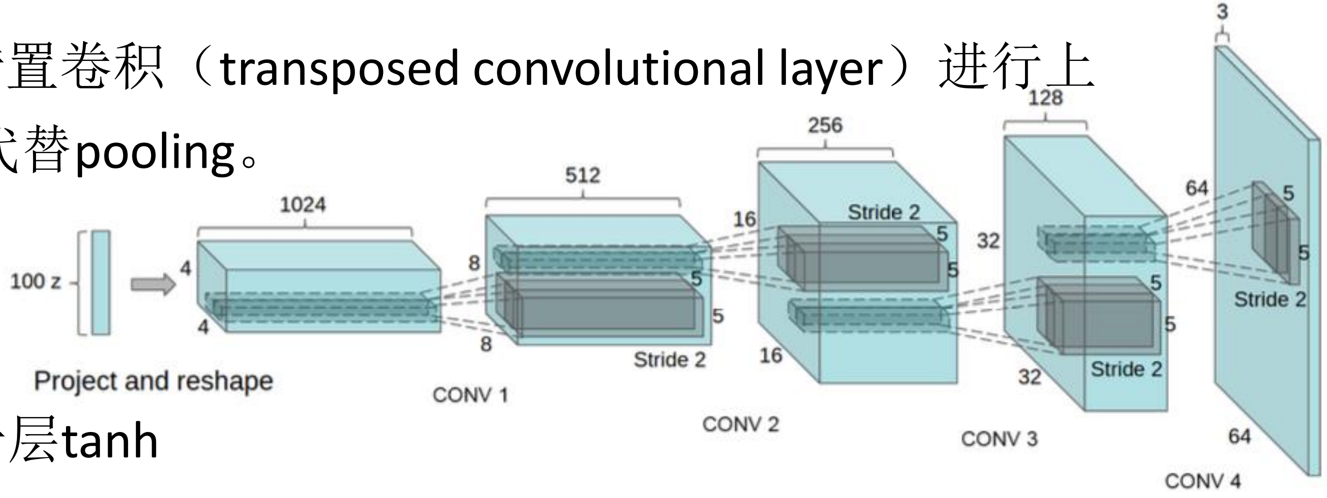
“Girl with red hair and red eyes”

“Girl with yellow ribbon”

$\rightarrow \boxed{\text{生成网络}} \rightarrow$

3. 卷积GAN

- 针对图像生成应用，用CNN代替MLP，逻辑不变，DCGAN
- 对卷积神经网络的结构做了一些改变，以提高样本的质量和收敛的速度
  - 取消所有pooling层。G网络中使用转置卷积（transposed convolutional layer）进行上采样，D网络中用加入stride的卷积代替pooling。
  - 在D和G中均使用batch normalization
  - 去掉FC层，使网络变为全卷积网络
  - G网络中用ReLU为激活函数，最后一层tanh
  - D网络中使用LeakyReLU作为激活函数

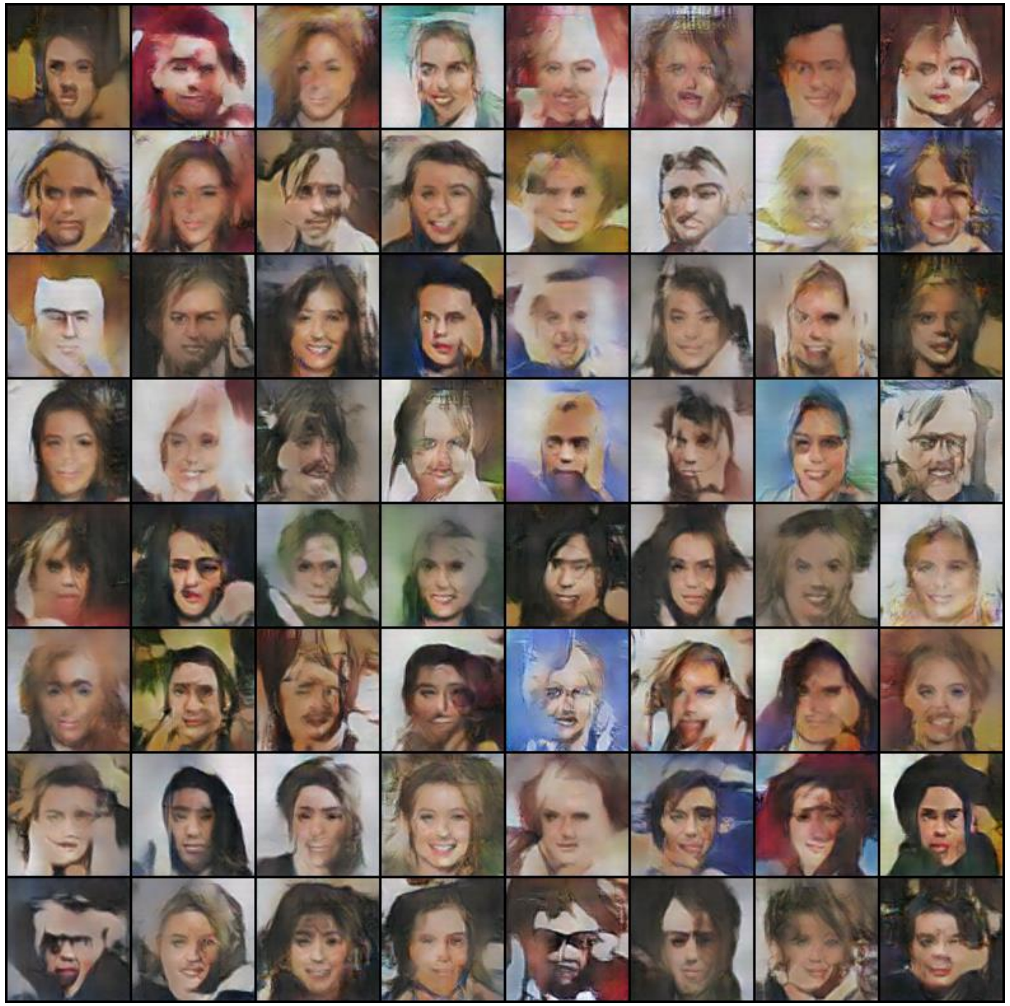


- 参考论文： Unsupervised Representation Learning with Deep Convolutional Generative Adversarial Networks， Arixv:1511.06434 .

卷积GAN案例

1. 参考gen_face/demo_genface.ipynb

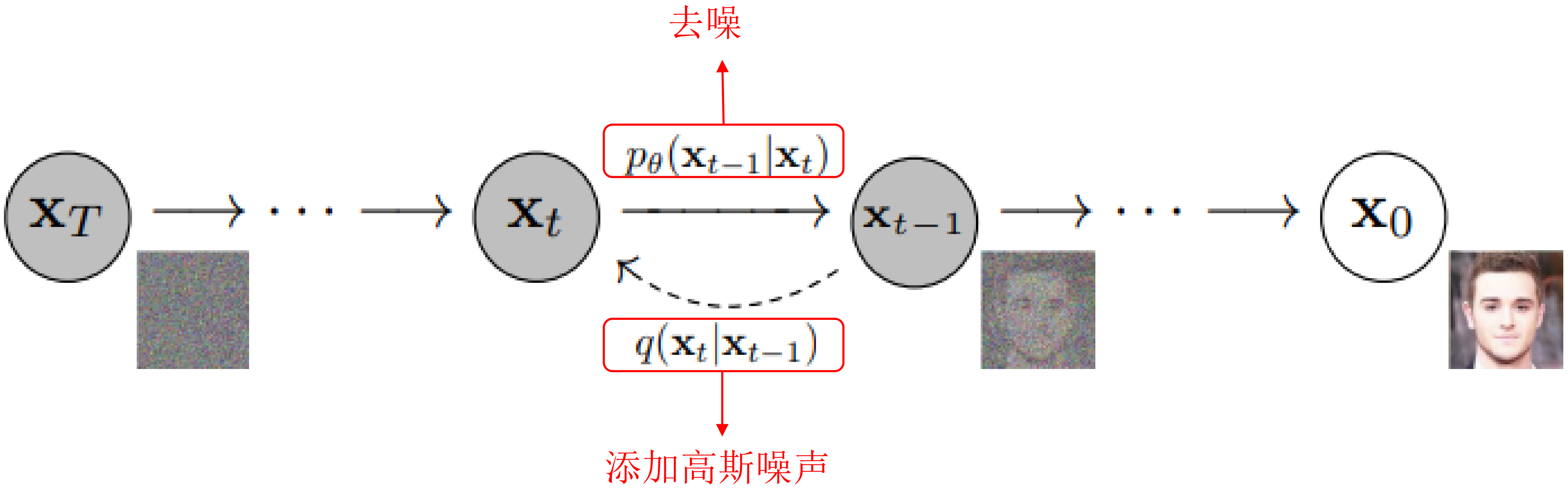
https://zhuanlan.zhihu.com/p/24767059



3. 扩散生成

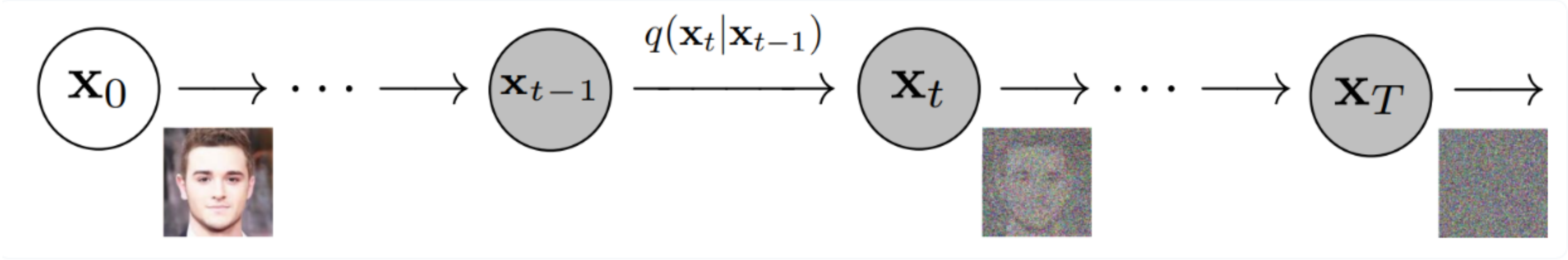
扩散模型(Diffusion model)

- 扩散模型是生成模型，用于生成与训练数据相似的数据
- 工作原理：
  1. 使用缓慢增加的噪声顺序破坏训练数据 （正向过程）
  2. 学习扭转这种破坏以形成数据的生成模型 （反向过程）



扩散模型前向(加噪)过程

- 扩散模型可以从几个不同角度理解：基于 Markov 链、基于分数匹配、基于微分方程等。常用如下示意图表示：



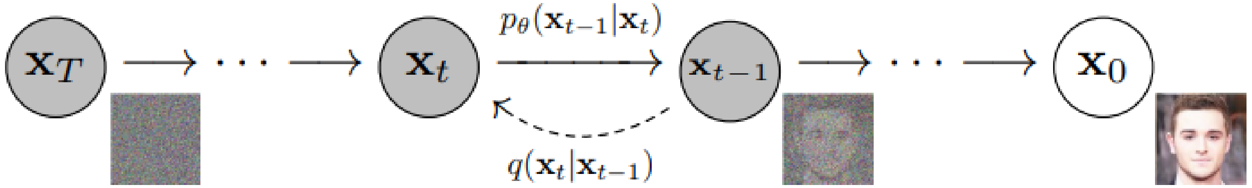
前向传播过程： $\mathrm{x}_t = \sqrt{\alpha_t}x_{t-1} + \sqrt{1-\alpha_t}\epsilon_1$ ①　　其中 $\alpha_t = 1 - \beta_t$，β_{t} $0.0001 \rightarrow 0.002$

$$\mathrm{x}_{t-1} = \sqrt{\alpha_{t-1}}x_{t-2} + \sqrt{1-\alpha_{t-1}}\epsilon_2$$ ②

② 代入 ① 即有 $\mathrm{x}_t = \sqrt{\alpha_t\alpha_{t-1}}x_{t-2} + \sqrt{1-\alpha_t\alpha_{t-1}}\bar{\epsilon}_2$

$$= \sqrt{\bar{\alpha}_t}x_0 + \sqrt{1-\bar{\alpha}_t}\bar{\epsilon}_t$$ $\longrightarrow$ 有初始的x_0就可以得到任意x_t

扩散模型反向(去噪)过程



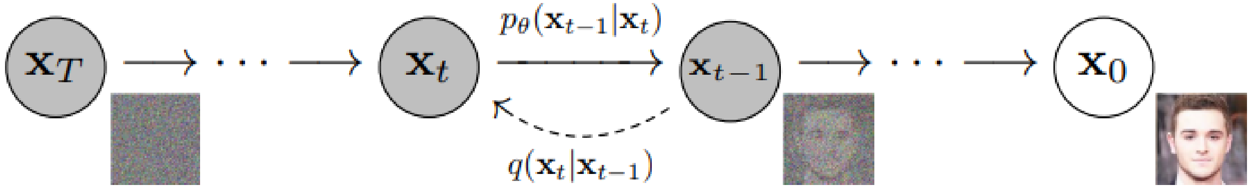
贝叶斯公式: $$q(x_{t-1}|x_t,x_0) = q(x_t|x_{t-1},x_0)\frac{q(x_{t-1}|x_0)}{q(x_t|x_0)}$$

$$\left.\begin{aligned}
&q(x_t|x_{t-1},x_0)= \sqrt{\alpha_t}x_{t-1}+\sqrt{1-\alpha_t}\epsilon \\
&q(x_{t-1}|x_0)= \sqrt{\bar{\alpha}_{t-1}}x_0+\sqrt{1-\bar{\alpha}_{t-1}}\epsilon \\
&q(x_t|x_0)= \sqrt{\bar{\alpha}_t}x_0+\sqrt{1-\bar{\alpha}_t}\epsilon
\end{aligned}\right\}$$

$$q(x_{t-1}|x_t,x_0) \propto$$

$$\exp(-\frac{1}{2}(\frac{\left(x_t-\sqrt{\alpha_t}x_{t-1}\right)^2}{\beta_t}+\frac{\left(x_{t-1}-\sqrt{\bar{\alpha}_{t-1}}x_0\right)^2}{1-\bar{\alpha}_{t-1}}+\frac{\left(x_t-\sqrt{\bar{\alpha}_t}x_0\right)^2}{1-\bar{\alpha}_t}))$$

扩散模型去噪过程



$$q(x_{t-1}|x_t,x_0) \propto \exp(-\frac{1}{2}(\frac{(x_t-\sqrt{\alpha_t}x_{t-1})^2}{\beta_t}+\frac{(x_{t-1}-\sqrt{\bar{\alpha}_{t-1}}x_0)^2}{1-\bar{\alpha}_{t-1}}+\frac{(x_t-\sqrt{\bar{\alpha}_t}x_0)^2}{1-\bar{\alpha}_t}))$$

$$=\exp(-\frac{1}{2}(\left(\frac{\alpha_t}{\beta_t}+\frac{1}{1-\bar{\alpha}_{t-1}}\right)x_{t-1}^2-\left(\frac{2\sqrt{\alpha_t}}{\beta_t}x_t+\frac{2\sqrt{\bar{\alpha}_{t-1}}}{1-\bar{\alpha}_{t-1}}x_0\right)x_{t-1}+C(x_t,x_0)))$$

常数

而 $\exp\left(-\frac{(x-\mu)^2}{2\sigma^2}\right)=\exp\left(-\frac{1}{2}(\frac{1}{\sigma^2}x^2-\frac{2\mu}{\sigma^2}x+\frac{\mu^2}{\sigma^2})\right)$

$$\mu_t(x_t,x_0)=\frac{\sqrt{\alpha_t}(1-\bar{\alpha}_{t-1})}{1-\bar{\alpha}_t}x_t+\frac{\sqrt{\bar{\alpha}_{t-1}}\beta_t}{1-\bar{\alpha}_t}x_0$$

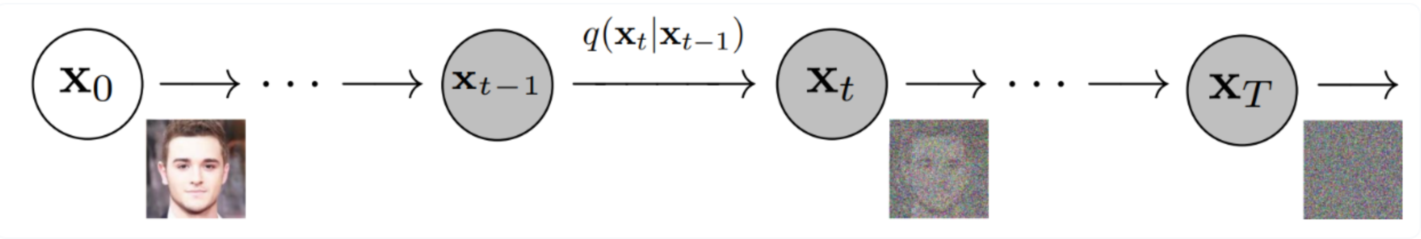
$$=\frac{1}{\sqrt{\alpha_t}}(x_t-\frac{\beta_t}{1-\bar{\alpha}_t}\epsilon_t)$$

而 $x_0=\frac{1}{\sqrt{\bar{\alpha}_t}}(x_t-\sqrt{1-\bar{\alpha}_t}\epsilon_t)$

扩散模型训练采样过程

Algorithm 1 Training

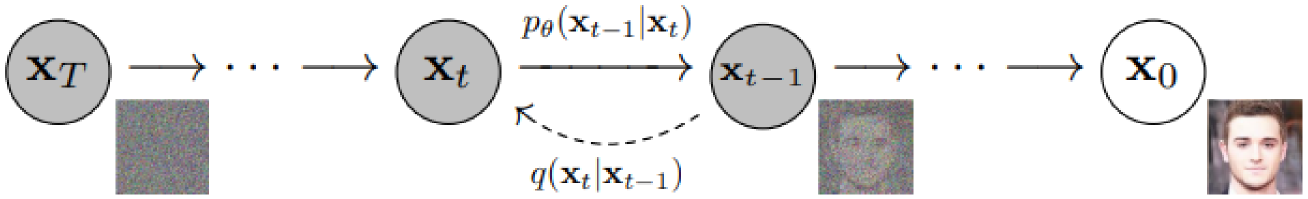
1: **repeat**
2: $\mathbf{x}_0 \sim q(\mathbf{x}_0)$
3: $t \sim \mathrm{Uniform}(\{1, \ldots, T\})$
4: $\boldsymbol{\epsilon} \sim \mathcal{N}(\mathbf{0}, \mathbf{I})$
5: Take gradient descent step on
$$\nabla_\theta \left\| \boldsymbol{\epsilon} - \boldsymbol{\epsilon}_\theta(\sqrt{\bar{\alpha}_t}\mathbf{x}_0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



$$\mathrm{x}_t = \sqrt{\bar{\alpha}_t} x_0 + \sqrt{1-\bar{\alpha}_t}\,\bar{\epsilon}_t$$

Algorithm 2 Sampling

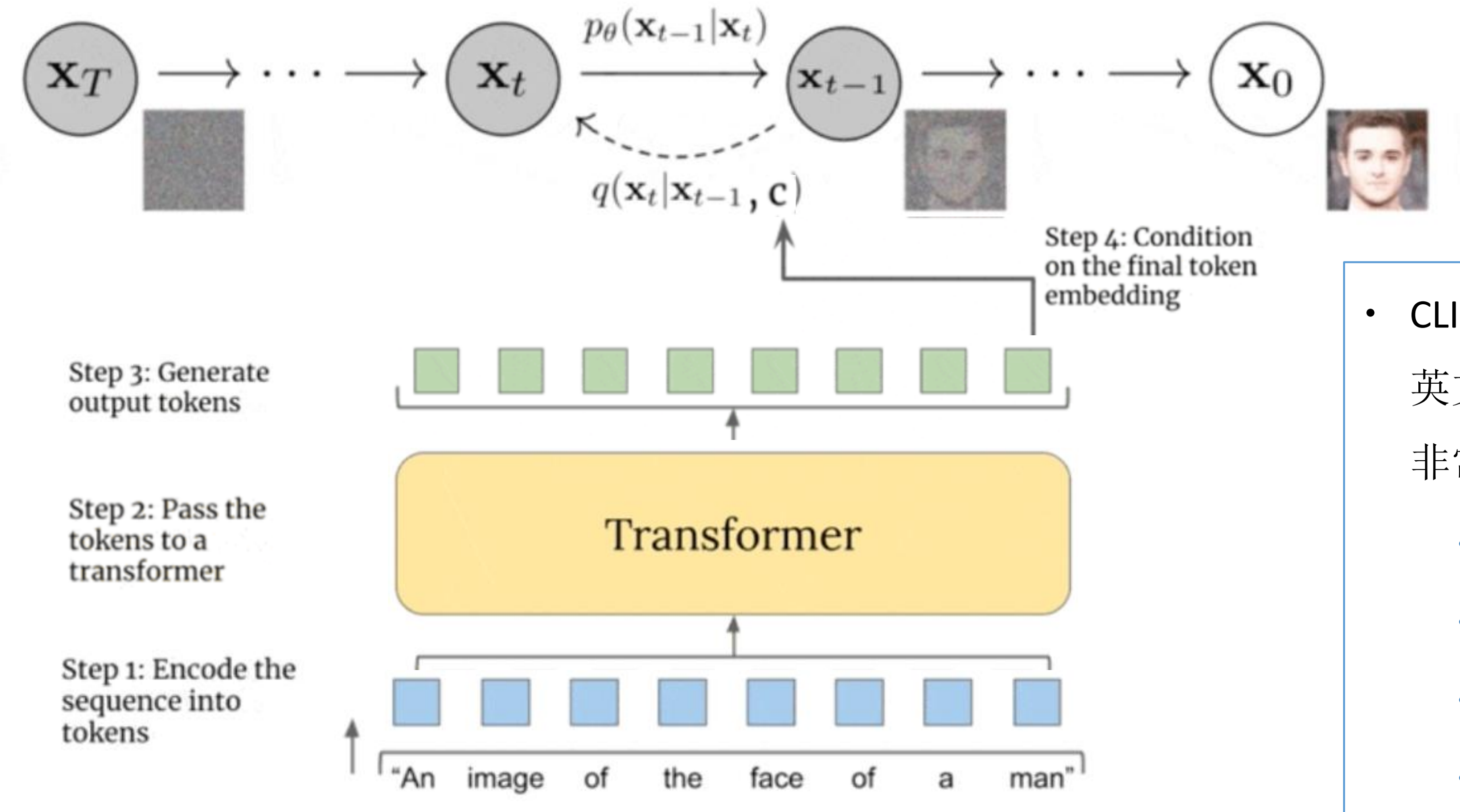
1: $\mathbf{x}_T \sim \mathcal{N}(\mathbf{0}, \mathbf{I})$
2: **for** $t = T, \ldots, 1$ **do**
3: $\mathbf{z} \sim \mathcal{N}(\mathbf{0}, \mathbf{I})$ if $t > 1$, else $\mathbf{z} = \mathbf{0}$
4: $\mathbf{x}_{t-1} = \frac{1}{\sqrt{\alpha_t}}\left(\mathbf{x}_t - \frac{1-\alpha_t}{\sqrt{1-\bar{\alpha}_t}}\boldsymbol{\epsilon}_\theta(\mathbf{x}_t,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



$$\mu_t(x_t, x_0) = \frac{1}{\sqrt{\alpha_t}}\left(x_t - \frac{\beta_t}{1-\bar{\alpha}_t}\epsilon_t\right)$$

典型应用：文生图

GLIDE: Towards Photorealistic Image Generation and Editing with Text-Guided Diffusion Models

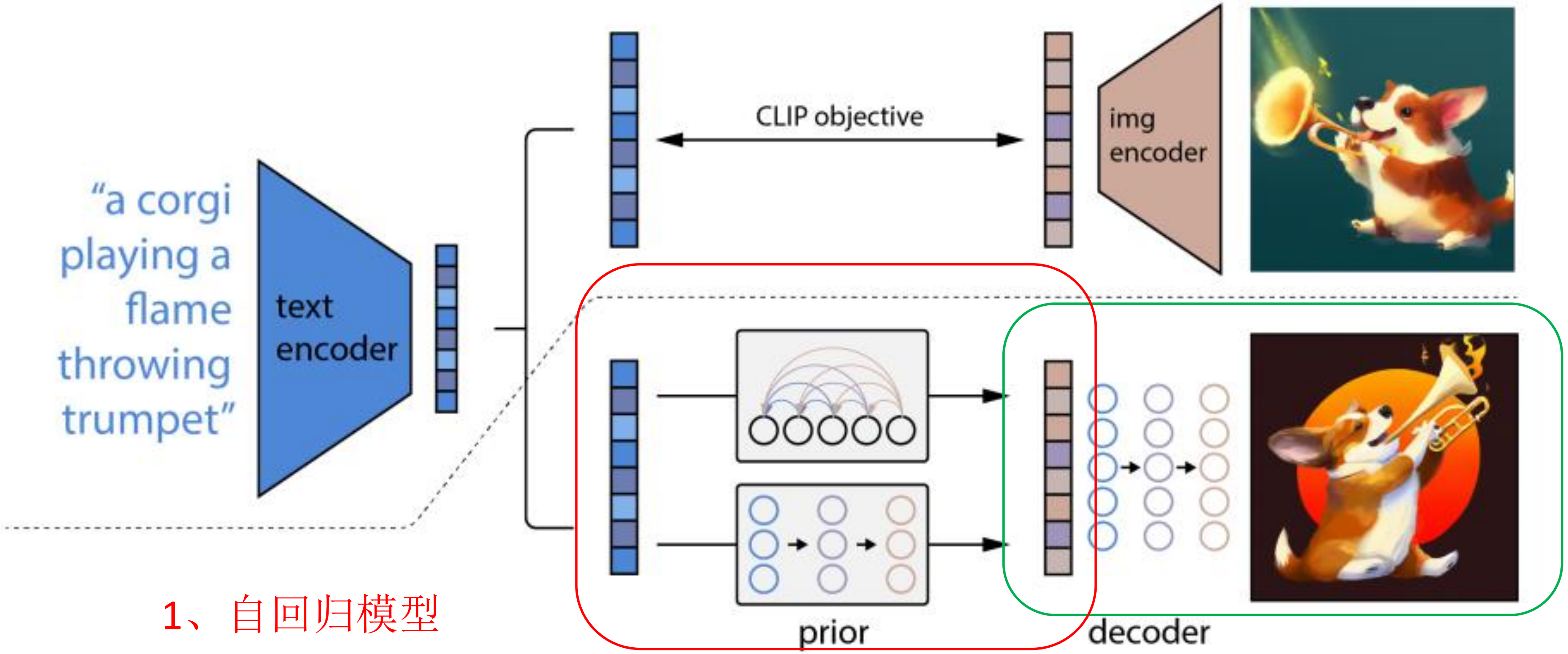


$$\hat{\mu}_\theta(x_t|y) = \mu_\theta(x_t|y) + s \cdot \Sigma_\theta(x_t|y)\nabla_{x_t} \log p_\phi(y|x_t)$$

- CLIP：把文本和图像映到同一个空间中。如：英文单词dog 和狗的图片在CLIP 下的像是两个非常接近的向量。
  - 一个较为新颖的方法，有一定的技术推进性。
  - 开源代码参考，代码逻辑清晰可读性高
  - 训练数据大小为 84 G，近 300 万条训练数据
  - 6 节点 48 张 A100 大约需要 3 小时
  - 在多模态领域具有启发意义
- 开源项目 - OpenCLIP

典型应用：文生图

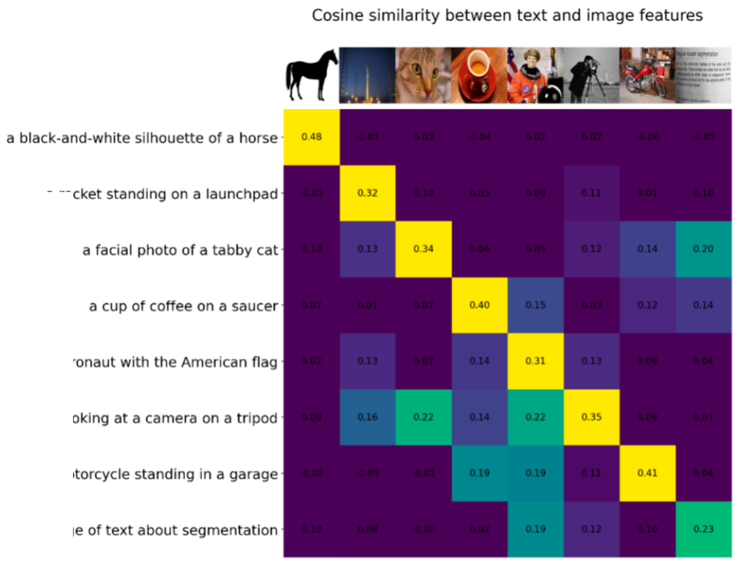
DALLE2: Hierarchical Text-Conditional Image Generation with CLIP Latents



1、自回归模型

2、Diffusion model

$$\hat{\mu}_\theta(x_t|c) = \mu_\theta(x_t|c) + s \cdot \Sigma_\theta(x_t|c)\nabla_{x_t}\left(f(x_t) \cdot g(c)\right)$$



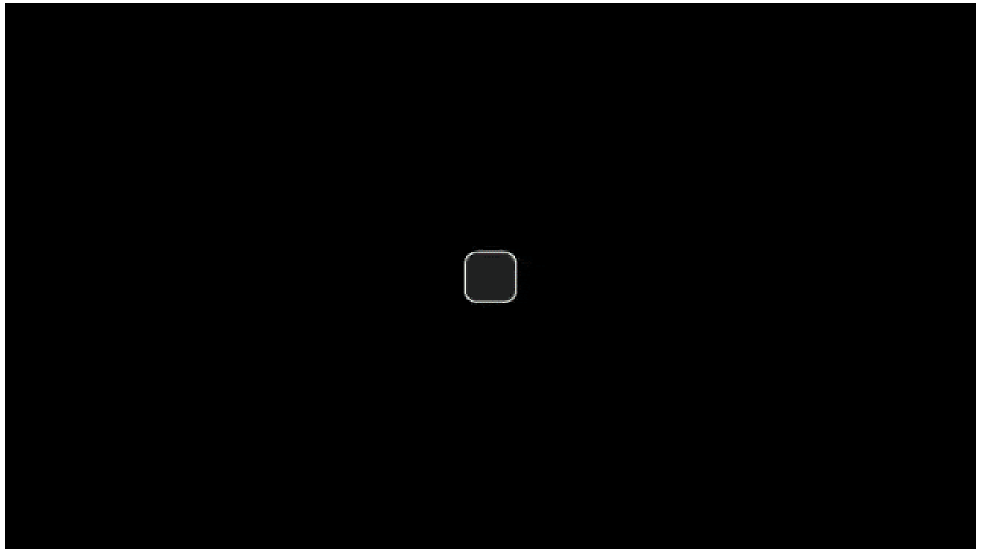
a teddy bear on a skateboard in times square

前沿应用：文生视频

文生视频PIKA1.0爆火，斯坦福华人学生退学创业，估值超2亿美元

机器之心 | + 关注

2023-11-30 15:10 北京 来源：澎湃新闻·澎湃号·湃客 字号



- 参考链接

  https://pika.art/blog