第7章：概率图模型和深度生成模型

中国科学技术大学
电子工程与信息科学系

主讲教师：李厚强 (lihq@ustc.edu.cn)
周文罡 (zhwg@ustc.edu.cn)
李　礼 (lil1@ustc.edu.cn)
胡　洋 (eeyhu@ustc.edu.cn)

概率图模型（Probabilistic Graphical Models）

- 概率图模型概述
- 概率有向图模型（贝叶斯网络）
  - 因子分解
  - 条件独立性
- 概率无向图模型（马尔可夫随机场）
  - 因子分解
  - 条件独立性
- 信念传播算法
- 应用举例

概率建模

- 图像处理和分析：观测数据 $\xrightarrow{\text{推测}}$ 未知数据
- 不确定性
  - 观测不确定性
  - 预测不确定性
- 概率模型
  - 链式法则： $P(x_1, \cdots, x_N) = P(x_1) \prod_{i=2}^{N} P(x_i | x_1, \cdots, x_{i-1})$
  - 条件独立性假设
- 概率图模型
  - 概率模型
  - 引入图作为表示工具

概率图模型

模型定义

- 图（graph）是由结点（node）和边（edge）组成的集合
  - ✓ 结点：v，结点的集合：V
  - ✓ 边：e，边的集合记：E
  - ✓ 图：$G = (V, E)$
- 概率图模型
  - ✓ 结点v，随机变量Y_v
  - ✓ 边e：随机变量之间的概率依赖关系
  - ✓ 用图$G = (V, E)$表示联合概率分布$P(Y)$

概率图模型的优点

- 使概率模型的结构可视化，启发新模型设计
- 便于分析模型的各种性质，如条件独立性
- 将复杂计算（如推断、学习问题）表示成图操作

概率图模型的基本问题

- 表示问题
  - 概率有向图模型（贝叶斯网络）
  - 概率无向图模型（马尔可夫随机场）
- 学习问题
  - 参数学习
  - 图结构学习
- 推断问题

概率图模型（Probabilistic Graphical Models）

- 概率图模型概述
- 概率有向图模型（贝叶斯网络）
  - 因子分解
  - 条件独立性
- 概率无向图模型（马尔可夫随机场）
  - 因子分解
  - 条件独立性
- 信念传播算法
- 应用举例

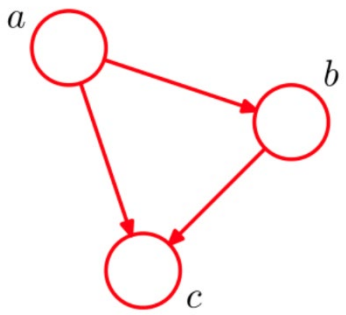
概率有向图模型（贝叶斯网络）

- 有向图表示随机变量之间的因果关系
- 考虑随机变量a, b, c的联合分布$p(a, b, c)$

$$p(a, b, c) = p(c|a, b)p(a, b)$$

$$= p(c|a, b)p(b|a)p(a)$$

- 用有向图模型表示联合分布



- 每个随机变量用一个结点表示
- 根据每个条件分布，在结点间引入有向边，表示随机变量之间的概率依赖关系
- 不同的分解方式，对应不同的图表示

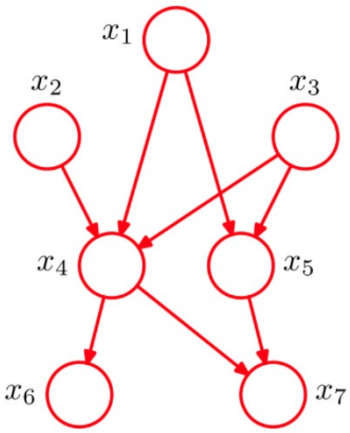
概率有向图模型的因子分解

- 概率有向图表示的联合概率分布可表示为图中每个结点在其父节点给定时的条件分布的乘积

$$p(\mathbf{x}) = \prod_{i=1}^{N} p(x_i|\mathrm{pa}_i)$$

其中pa_i为结点 x_i的父结点集合

要求图中没有有向环，即为有向无环图



$$p(x_1,\cdots,x_7) = p(x_1)p(x_2)p(x_3)p(x_4|x_1,x_2,x_3)$$
$$p(x_5|x_1,x_3)p(x_6|x_4)p(x_7|x_4,x_5)$$

缺失的边有意义

概率有向图模型的因子分解

- M个结点的Markov链

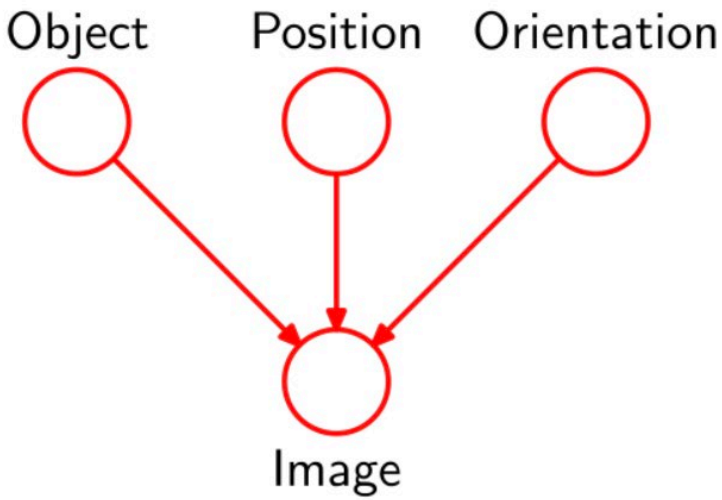
$$p(\mathbf{x}_1, \cdots, \mathbf{x}_M) = p(\mathbf{x}_1) \prod_{i=2}^{M} p(\mathbf{x}_i|\mathbf{x}_{i-1})$$

  - $p(\mathbf{x}_1)$有 $K-1$个参数 （K个符号）
  - $p(\mathbf{x}_i|\mathbf{x}_{i-1})$有 $K(K-1)$个参数
  - 总共有$K-1+(M-1)(K-1)K$个参数

- 隐马尔可夫模型的核心结构

生成式模型

- 通常以能观察到的变量为叶结点，其他结点为隐变量。通过引入隐变量，使观察变量复杂的分布可以通过更简单的条件分布构造而成。
- 简单的图像生成模型



- 物体、位置、方向满足相互独立的先验分布
- 图像的分布以物体、位置、方向为条件
- 图模型描述图像数据产生的因果过程

$$p(Im, Ob, Po, Or) = p(Im|Ob, Po, Or)p(Ob)p(Po)p(Or)$$

条件独立性

- 条件独立性质
  - 给定随机变量c的条件下，随机变量a和b独立

$$p(a|b,c) = p(a|c)$$

或者

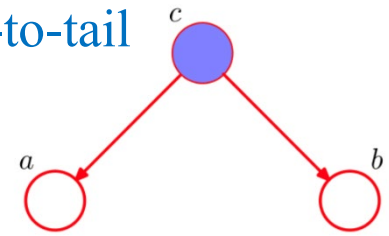
$$p(a,b|c) = p(a|b,c)p(b|c)$$
$$= p(a|c)p(b|c)$$

  - 记为：$a \perp b|c$
- 条件独立性是概率模型的重要性质，既使模型结构更简单，也使模型的推断和学习问题更简单
- 概率图模型的一大优点是可从图中直接得到变量间的条件独立关系

条件独立性

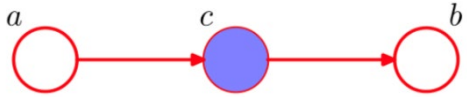
- 变量a, b, c的关系

tail-to-tail



$a \perp b|c$

head-to-tail

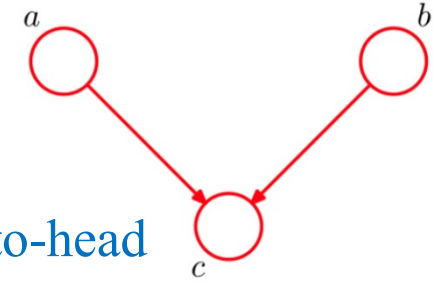


$a \perp b|c$

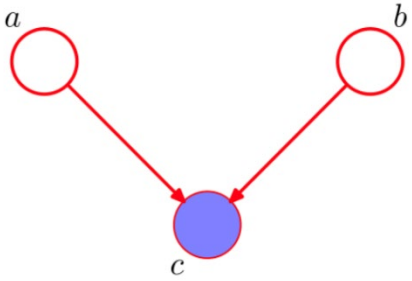
$$p(a,b|c) = \frac{p(c)p(a|c)p(b|c)}{p(c)} = p(a|c)p(b|c)$$

$$p(a,b|c) = \frac{p(a)p(c|a)p(b|c)}{p(c)} = p(a|c)p(b|c)$$

head-to-head



$a \perp b|\emptyset$



$a \not\perp b|c$

$$p(a,b) = \sum_c p(a)p(b)p(c|a,b) = p(a)p(b)$$

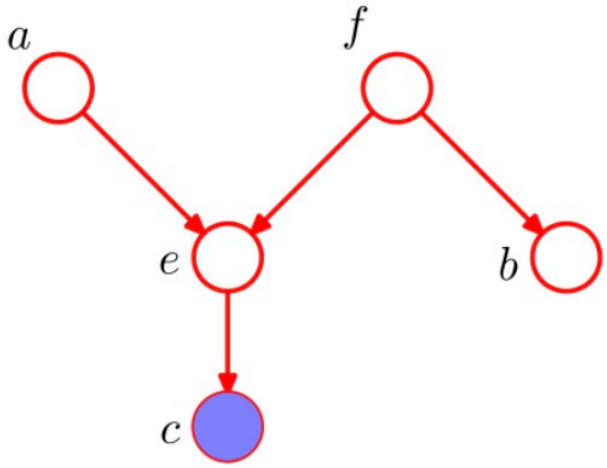
$$p(a,b|c) = \frac{p(a)p(b)p(c|a,b)}{p(c)}$$

条件独立性

- D-separation性质
  - 有向图中的三个结点集合A, B, C
  - 考虑A中结点到B中结点的所有可能路径
  - 一条路径如果含有一个结点满足下列条件之一，则该路径被阻隔
    - ✓ 路径经过结点时为head-to-tail或tail-to-tail形式，并且该结点在集合C中
    - ✓ 路径经过结点时为head-to-head形式，并且该结点及其子孙结点都不在集合C中
  - 如果A, B间都所有路径都被阻隔，则称A, B被C集合D分离，它们对应的随机变量满足$A \perp B|C$

条件独立性

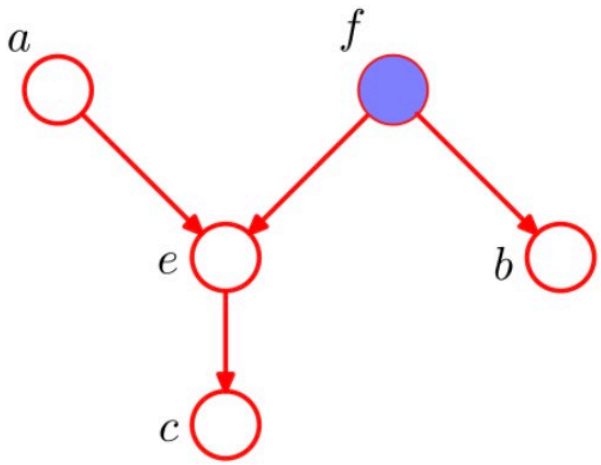
- 例子



a, b没有被f阻隔

a, b没有被e阻隔

$a \perp b|c$不成立



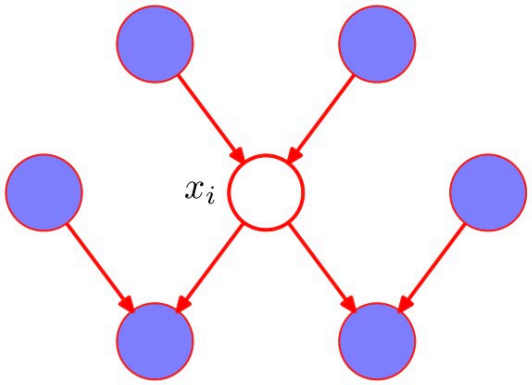
a, b被f阻隔

a, b被e阻隔

$a \perp b|f$成立

有向图模型的马尔可夫毯

- 马尔可夫毯：一个结点的马尔可夫毯是使该结点与图中所有其他结点独立所需要观察到的最小结点集合
- 有向图中，结点的马尔可夫毯包含其父结点、子结点及所有co-parent结点



$$p(x_i|x_{\{j\neq i\}}) = \frac{p(x_1,\cdots,x_M)}{\int p(x_1,\cdots,x_M)\mathrm{d}x_i}$$

$$= \frac{\prod_k p(x_k|\mathrm{pa}_k)}{\int \prod_k p(x_k|\mathrm{pa}_k)\,\mathrm{d}x_i}$$

不含有x_i的因子上下消掉

$$= p(x_i|x_{\{j\in MB_i\}})$$

也可由D-separation性质得到

概率图模型（Probabilistic Graphical Models）

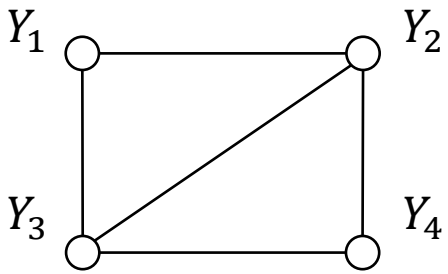
- 概率图模型概述
- 概率有向图模型（贝叶斯网络）
  - 因子分解
  - 条件独立性
- 概率无向图模型（马尔可夫随机场）
  - 因子分解
  - 条件独立性
- 信念传播算法
- 应用举例

概率无向图模型的因子分解

团与最大团

- 无向图 G 中任何两个结点均有边连接的结点子集称为团（clique）
- 若C是无向图G的一个团，并且不能再加进任何一个G的结点使其成为一个更大的团，则称此C为最大团（maximal clique）

  - 2个结点的团：
    - ✓ $\{Y_1, Y_2\}$， $\{Y_2, Y_3\}$， $\{Y_3, Y_4\}$， $\{Y_4, Y_2\}$， $\{Y_1, Y_3\}$
  - 3个结点的团
    - ✓ $\{Y_1, Y_2, Y_3\}$， $\{Y_2, Y_3, Y_4\}$
  - 最大团
    - ✓ $\{Y_1, Y_2, Y_3\}$， $\{Y_2, Y_3, Y_4\}$



概率无向图模型的因子分解

- 概率无向图模型的因子分解（Factorization）
  - 将概率无向图模型的联合概率分布表示成最大团上的随机变量的函数的乘积形式
- Hammersley-Clifford定理

如果一个联合概率分布$P(Y)$满足无向图G包含的条件独立性，当且仅当 $P(Y)$ 可以表示为一系列定义在最大团上的非负函数的乘积的形式，即

$$P(Y) = \frac{1}{Z}\prod_{C}\psi_C(Y_C)$$

其中C为G上的最大团， Y_C表示C对应的随机变量，$\psi_C(Y_C) \geq 0$是定义在C上的势函数， Z为规范化因子，用来将乘积归一化为概率形式：

$$Z = \sum_{Y}\prod_{C}\psi_C(Y_C)$$

概率无向图模型的因子分解

势函数

$$\psi_C(Y_C) = \exp\{-E(Y_C)\}$$

- ✓ $E(Y_C)$被称为能量函数（energy function）
- ✓ 总能量可以通过将每个最大团的能量相加的方法得到
- ✓ 指数表示被称为玻尔兹曼分布（Boltzmann distribution）

$$P(Y) = \frac{1}{Z}\prod_C \exp\{-E(Y_C)\}$$

$$= \frac{1}{Z}\exp\left\{-\sum_C E(Y_C)\right\}$$

规范化因子（划分函数）

- ✓ 划分函数的计算复杂度是指数级的
- ✓ 对于参数学习来说，划分函数是必要的；对于局部条件概率分布的计算，划分函数不是必要的

条件独立性

成对马尔可夫性

- 设u和v是无向图G中任意两个没有边连接的结点，其他所有结点为O，对应的随机变量分别为Y_u，Y_v和Y_O
- 给定随机变量组Y_O的条件下，随机变量Y_u和Y_v是条件独立的

$$P(Y_u, Y_v|Y_O) = P(Y_u|Y_O)P(Y_v|Y_O)$$

局部马尔可夫性

- 设v是无向图G中任意结点，W是与v有边相连的所有结点，O是其他所有结点，对应的随机变量分别为Y_v，Y_W和Y_O
- 给定随机变量组Y_W的条件下，Y_v与Y_O是条件独立的

$$P(Y_v, Y_O|Y_W) = P(Y_v|Y_W)P(Y_O|Y_W)$$

或 $$P(Y_v|Y_W, Y_O) = P(Y_v|Y_W)$$

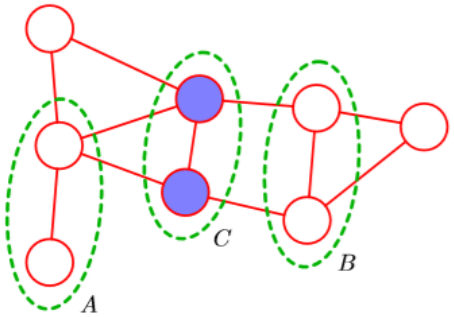
- 结点集合W是结点v马尔可夫毯

条件独立性

全局马尔可夫性

- A，B，C是在无向图G中的结点集合，对应的随机变量分别为Y_A，Y_B，Y_C
- 结点集合A，B被结点集合C分开：
  - ✓ 考虑连接集合A的结点和集合B的结点的所有可能路径
  - ✓ 如果所有这些路径都通过了集合C中的一个或多个结点，那么所有这样的路径都被阻隔
- 给定随机变量组Y_C的条件下，Y_A和Y_B是条件独立的

$$P(Y_A, Y_B|Y_C) = P(Y_A|Y_C)P(Y_B|Y_C)$$

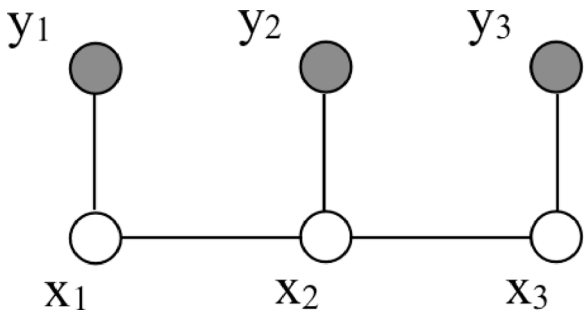


概率图模型（Probabilistic Graphical Models）

- 概率图模型概述
- 概率有向图模型（贝叶斯网络）
  - 因子分解
  - 条件独立性
- 概率无向图模型（马尔可夫随机场）
  - 因子分解
  - 条件独立性
- 信念传播算法
- 应用举例

信念传播算法（Belief Propagation）

- 3结点马尔可夫链



$$p(x_1, x_2, x_3 \mid y_1, y_2, y_3) = \frac{1}{Z(\mathbf{y})}\phi_{12}(x_1, x_2)\phi_{23}(x_2, x_3)\psi_1(y_1, x_1)\psi_2(y_2, x_2)\psi_3(y_3, x_3)$$

信念传播算法（BP）

□ 边缘概率计算

$$p(x_1 \mid \mathbf{y}) = \sum_{x_2}\sum_{x_3} p(x_1, x_2, x_3 \mid \mathbf{y})$$

$|x|^3$项加和

$$= \frac{1}{Z(\mathbf{y})}\sum_{x_2}\sum_{x_3} \phi_{12}(x_1, x_2)\phi_{23}(x_2, x_3)\psi_1(y_1, x_1)\psi_2(y_2, x_2)\psi_3(y_3, x_3)$$

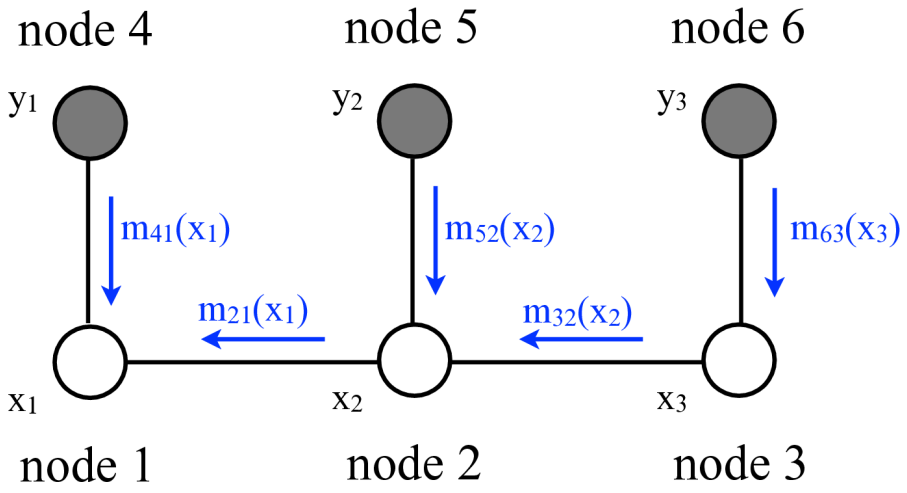
$$= \frac{1}{Z(\mathbf{y})}\psi_1(y_1, x_1)\sum_{x_2} \phi_{12}(x_1, x_2)\psi_2(y_2, x_2)\sum_{x_3} \phi_{23}(x_2, x_3)\psi_3(y_3, x_3)$$

$2|x|^2$项加和

对N结点马尔可夫链，由$|x|^N$项加和减少为 $(N-1)|x|^2$项

信念传播算法（BP）

- 消息传递（Message Passing）



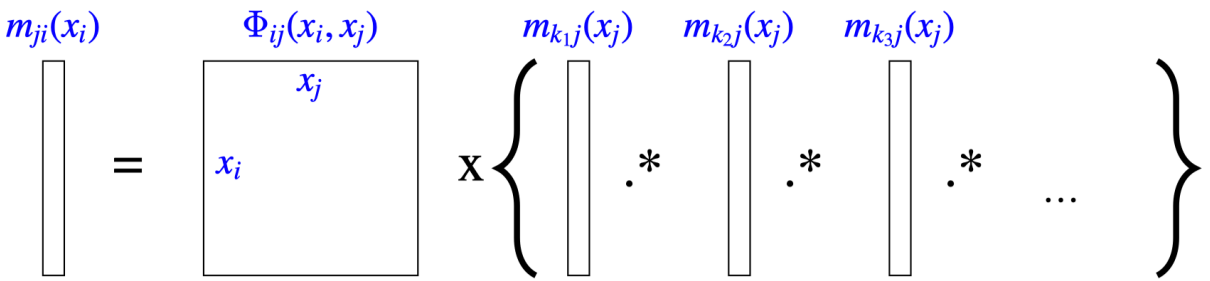
$$m_{63}(x_3) = \psi_3(y_3, x_3)$$
$$m_{52}(x_2) = \psi_2(y_2, x_2)$$
$$m_{41}(x_1) = \psi_2(y_1, x_1)$$
$$m_{32}(x_2) = \sum_{x_3} \phi_{23}(x_2, x_3) m_{63}(x_3)$$
$$m_{21}(x_1) = \sum_{x_2} \phi_{12}(x_1, x_2) m_{52}(x_2) m_{32}(x_2)$$

$$p(x_1 \mid \mathbf{y}) = \frac{1}{Z(\mathbf{y})} \psi_1(y_1, x_1) \sum_{x_2} \phi_{12}(x_1, x_2) \psi_2(y_2, x_2) \sum_{x_3} \phi_{23}(x_2, x_3) \psi_3(y_3, x_3)$$

信念传播算法（BP）

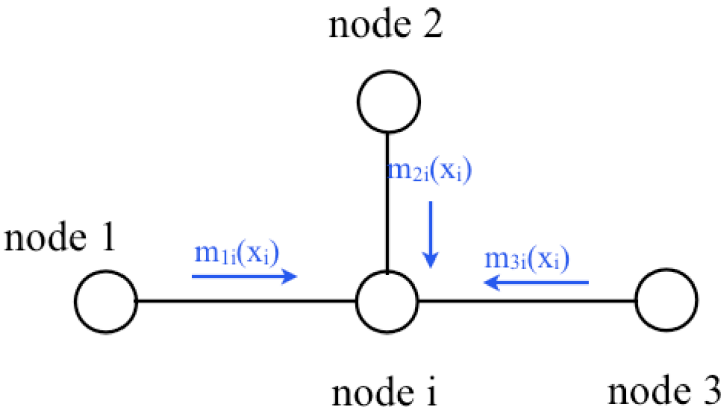
- 结点 j 向结点 i 传递的消息
  - 将除结点 i 外，所有其他结点传递给结点 j 的消息相乘
  - 再与结点 i 和结点 j 之间的势函数相乘
  - 对结点 j 对应的随机变量的所有取值求和

$$m_{ji}(x_i) = \sum_{x_j} \psi_{ij}(x_i, x_j) \prod_{k \in \eta(j) \setminus i} m_{kj}(x_j)$$



信念传播算法（BP）

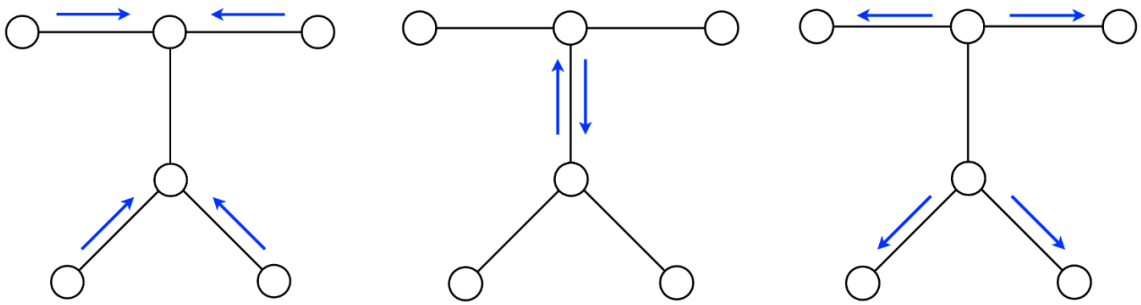
- 边缘概率计算



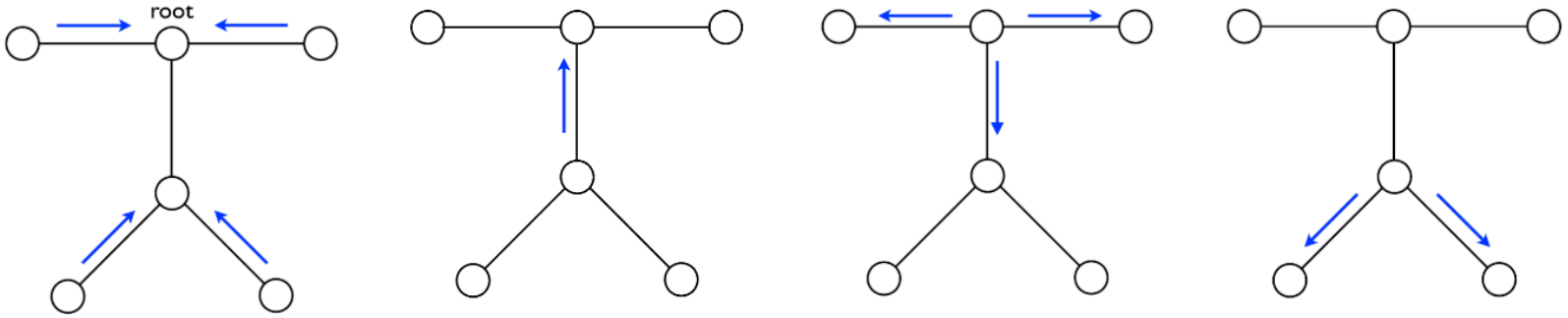
$$p_i(x_i) = \prod_{j \in \eta(i)} m_{ji}(x_i)$$

信念传播算法（BP）

- 消息传递顺序 （root确定的情况下）
  - 同时并行传递



  - 深度优先传递

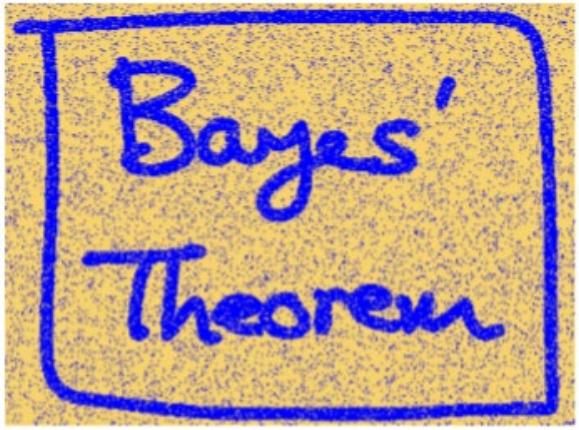


概率图模型（Probabilistic Graphical Models）

- 概率图模型概述
- 概率有向图模型（贝叶斯网络）
  - 因子分解
  - 条件独立性
- 概率无向图模型（马尔可夫随机场）
  - 因子分解
  - 条件独立性
- 信念传播算法
- 应用举例

马尔可夫随机场应用举例

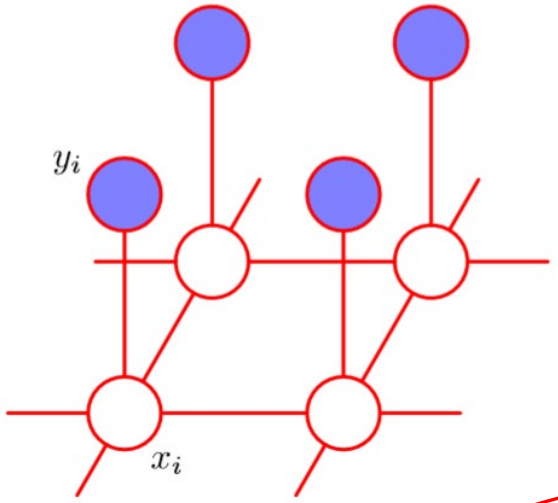
图像去噪



- 带噪图像：$y_i \in \{-1, +1\}$，$i = 1, \ldots, D$覆盖所有像素
  - ✓ 由无噪声图像，以一个较小概率随机翻转像素值符号得到
- 未知的无噪声图像：$x_i \in \{-1, +1\}$
- 目标：给定带有噪声的图像，恢复原始的无噪声图像

马尔可夫随机场

- 图像去噪



能量函数

$$E(x,y) = h\sum_i x_i - \beta\sum_{\{i,j\}} x_i x_j - \eta\sum_i x_i y_i$$

两种类型的团

- $-\beta x_i x_j$
  - ✓ x_i和x_j符号相同时，具有较低的能量（即较高的概率）
  - ✓ x_i和x_j符号相反时，具有较高的能量（即较低的概率）
- $-\eta x_i y_i$
  - ✓ x_i和y_i符号相同时，具有较低的能量
  - ✓ x_i和y_i符号相反时，具有较高的能量

马尔可夫随机场

- 图像去噪

联合概率分布

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

固定噪声图像对应的观测值y，图像去噪即求解x，使得条件概率分布$p(x|y)$最大

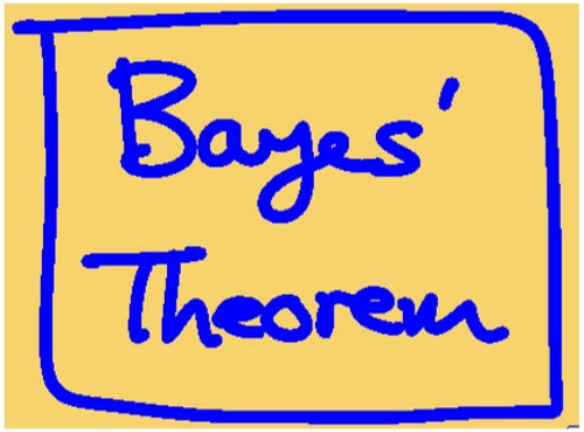
可使用迭代条件模式（ICM）算法，求解获得无噪图像x

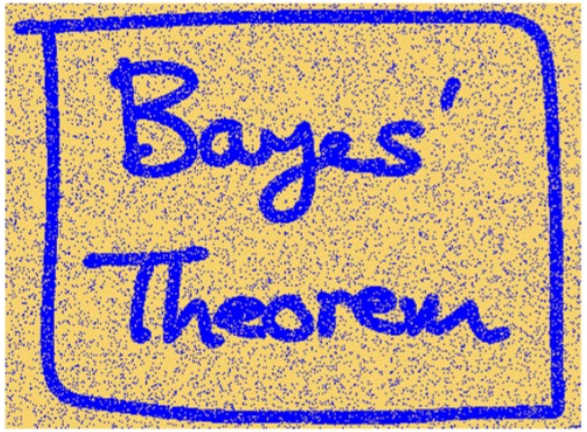
- 对所有像素，初始化变量$x_i=y_i$
- 每次取一个x_j结点，保持其他所有结点变量固定，计算x_j两个可能状态$x_j=+1$和$x_j=-1$的总能量
- 将x_j设置为能量较低的状态
- 对其他结点重复更新过程，直到满足某个合适的停止条件

马尔可夫随机场

- 图像去噪

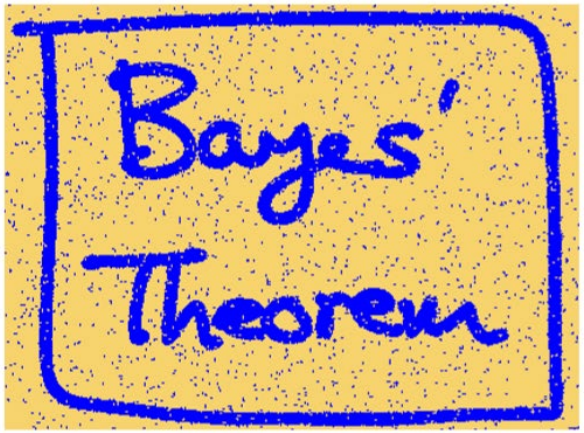
原始二值图像

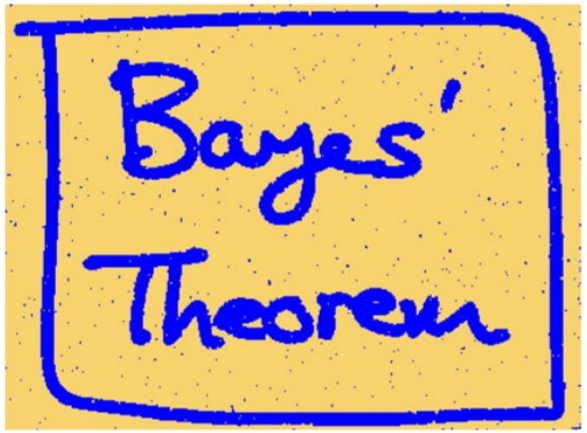




带噪图像

迭代条件模型恢复的结果





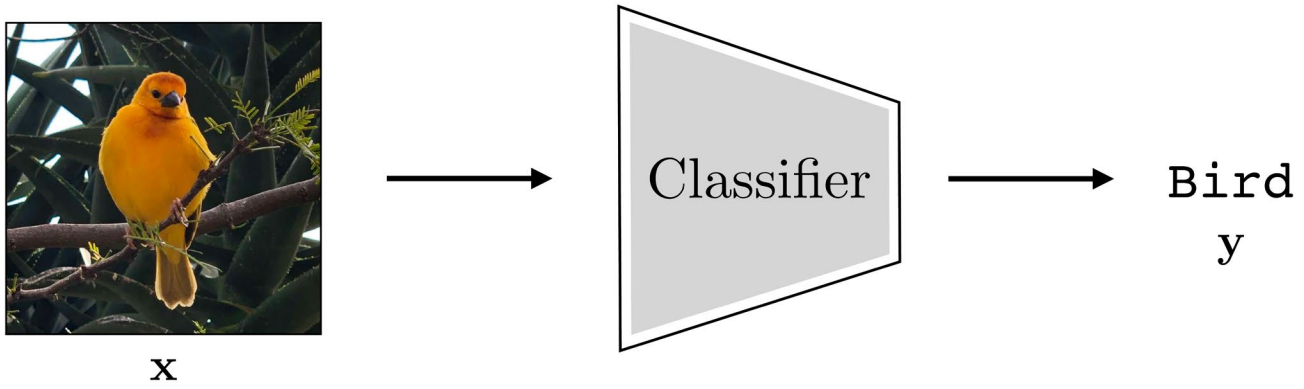
图割算法恢复的结果

生成式模型（Generative Models）

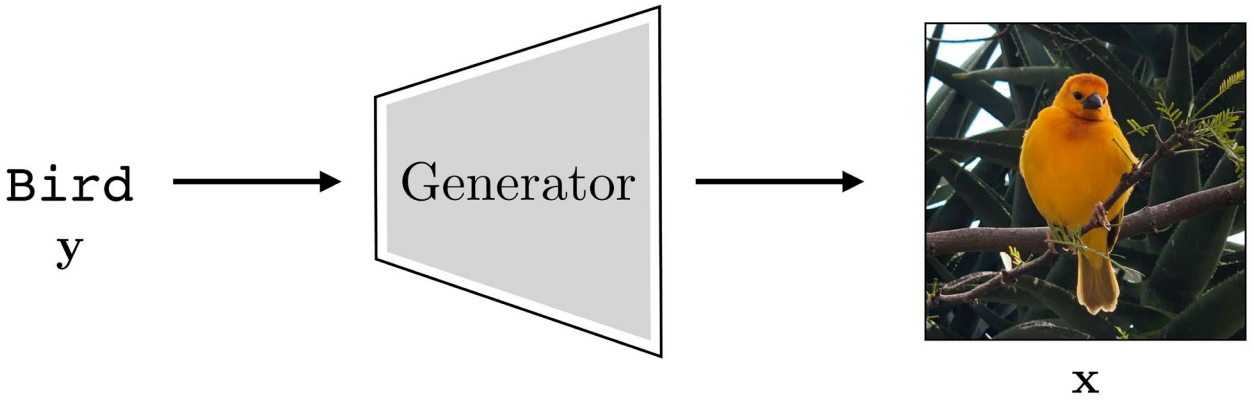
- 生成式模型概论
- 自回归生成模型
- 变分自编码器
- 扩散模型
- 生成对抗网络
- 条件生成式模型

生成式模型概论

- 判别式模型

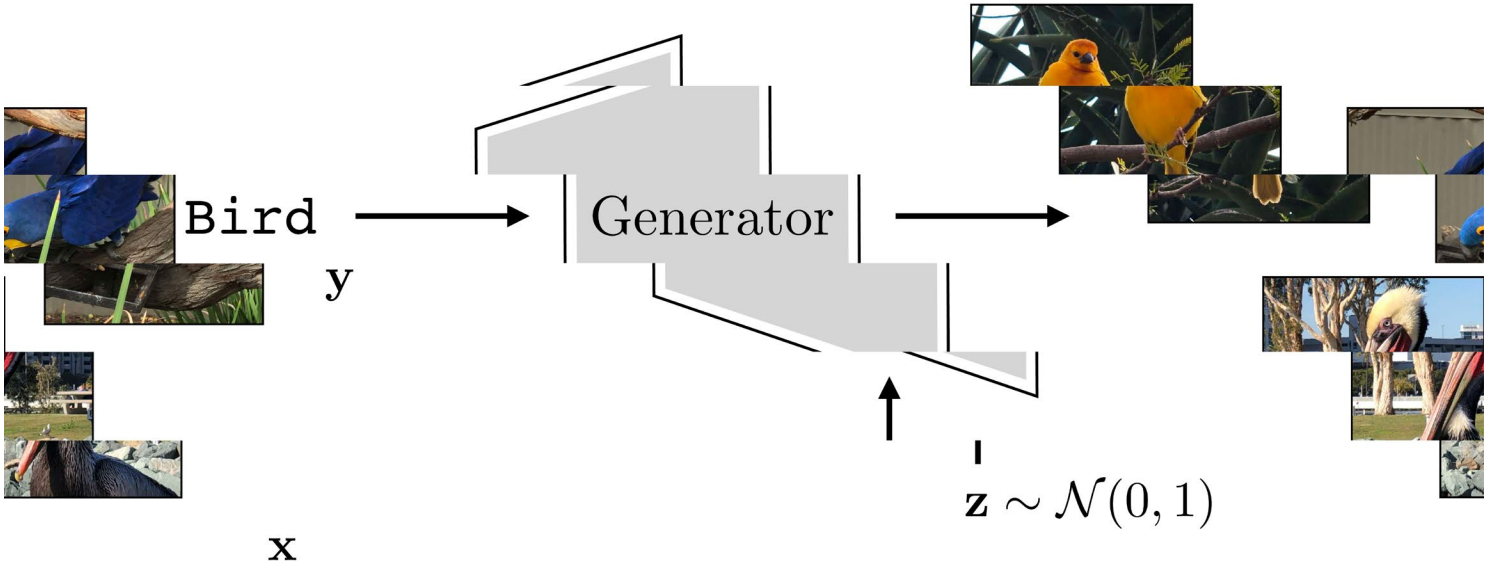


- 生成式模型



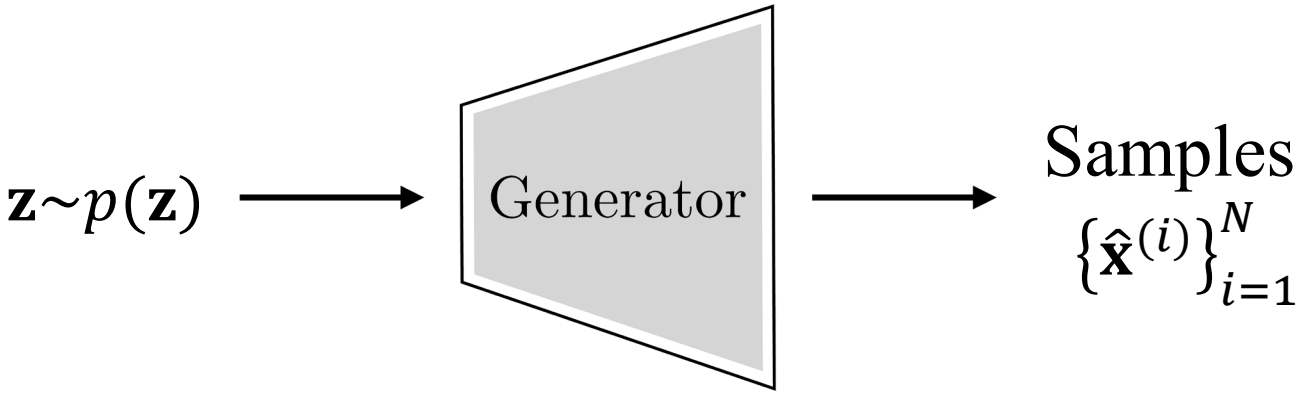
生成式模型概论

- 生成式模型
  - 一对多映射：引入随机性
  - y：标签、文本描述、草图、。。。
  - z：隐变量 （与有向图模型对应）

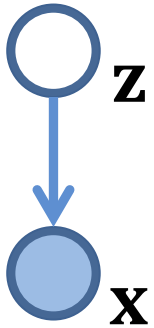


生成式模型概论

无条件生成式模型



- 观测变量 $\mathbf{x}$ 服从某固定但未知的分布
- 隐变量 $\mathbf{z}$ 服从先验分布 $p(\mathbf{z})$
- 联合分布：$p(\mathbf{x}, \mathbf{z}) = p(\mathbf{x}|\mathbf{z})p(\mathbf{z})$



生成式模型的定义

- 通常机器学习的任务就是学习一个模型，应用这一模型，对给定的输入预测相应的输出。对于模型的分类有很多种，其中一种分类是把模型分为：判别模型和生成模型两种。

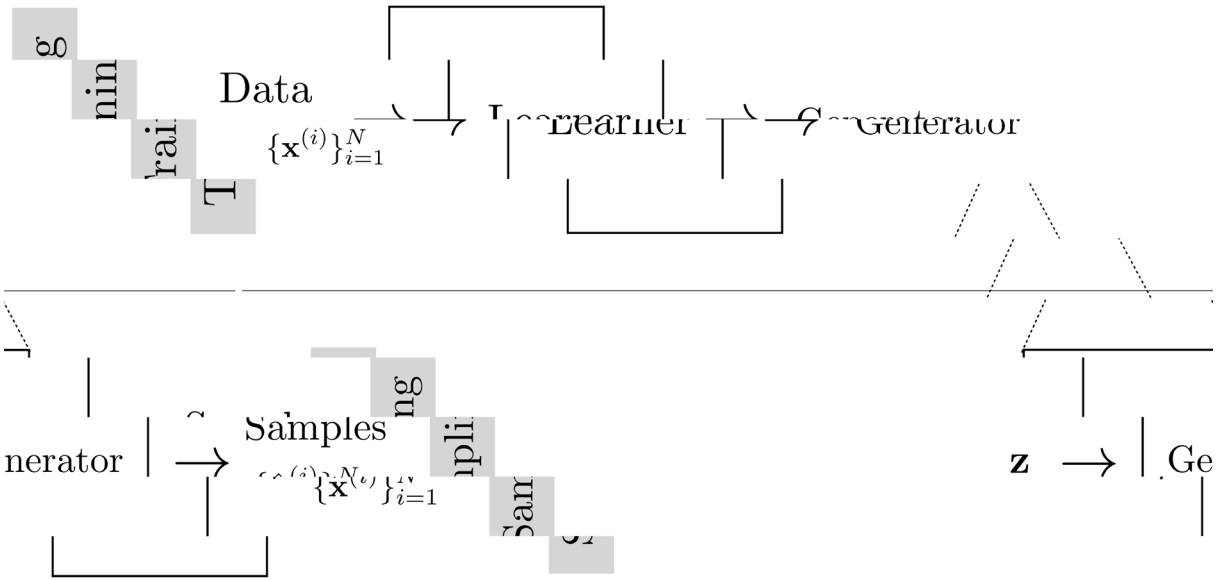
- 判别模型主要是根据输入数据推测数据具备的一些性质，即：已知观察变量x和隐变量z，直接对$p(z|x)$进行建模。它根据输入的观察变量x直接得到隐变量z出现的可能性。
  - 例如：当模型为分类模型时，隐变量z则代表类别变量。

- 生成模型则是对$p(x,z)$进行建模，同时，我们也可以根据联合概率分布$p(x,z)$采样生成观测变量x。

生成式模型概论

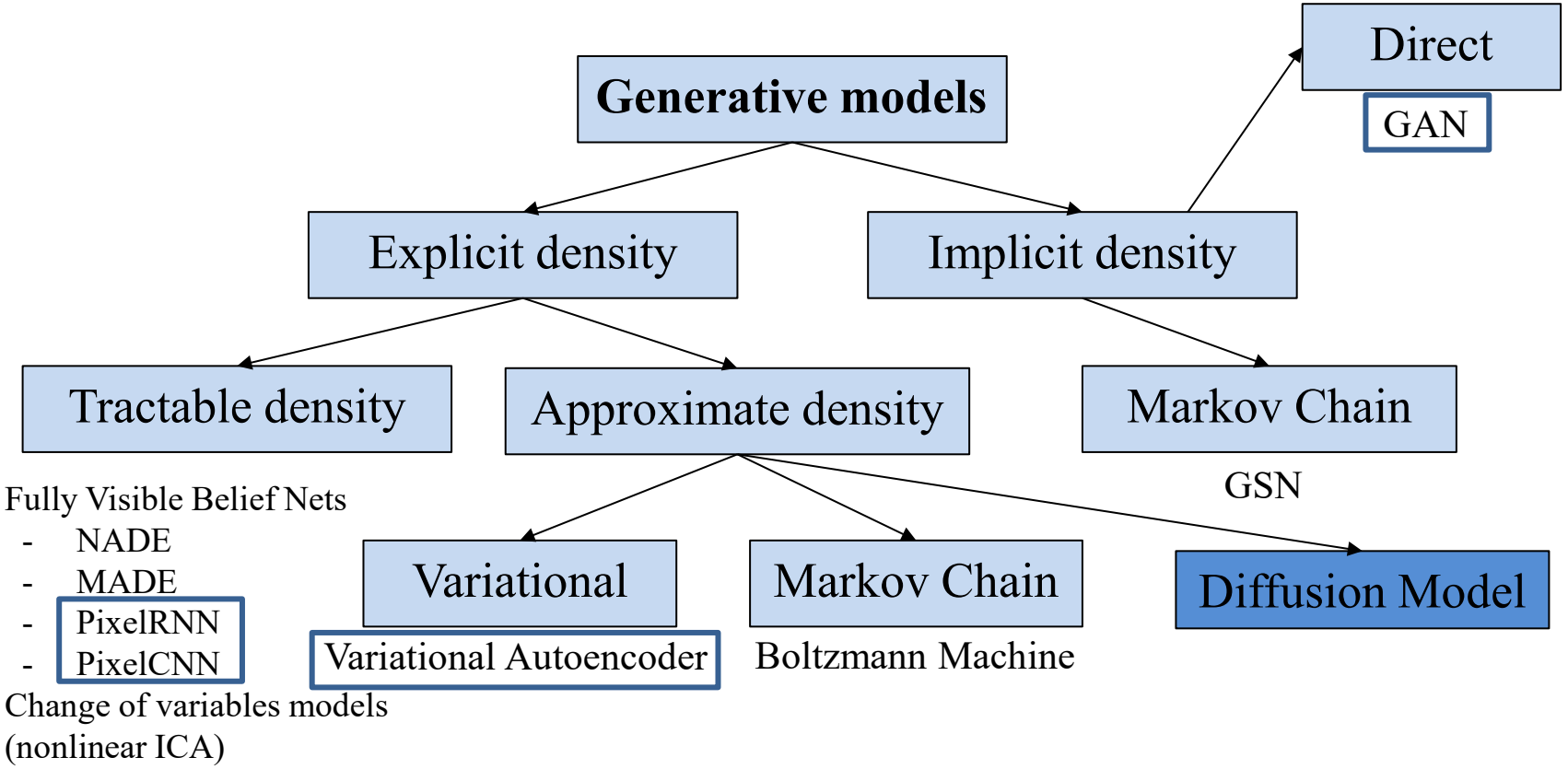
- 生成器的学习和使用
  - 用观测数据 $\{\mathbf{x}^{(i)}\}_{i=1}^{N}$ 学习生成器
  - 生成过程

$$\mathbf{z} \sim p(\mathbf{z})$$
$$\mathbf{x} = g(\mathbf{z}) \quad (\mathbf{x} \sim p(\mathbf{x}|\mathbf{z}))$$



生成式模型分类

- 根据对概率密度函数的表达，生成式模型可以分为两类：显式表达和隐式表达概率密度函数的生成式模型。
- 显式表达概率密度函数的生成式模型又可分为近似概率密度函数的生成式模型和解析概率密度函数的生成式模型。



生成式模型（Generative Models）

- 生成式模型概论
- 自回归生成模型
- 变分自编码器
- 扩散模型
- 生成对抗网络

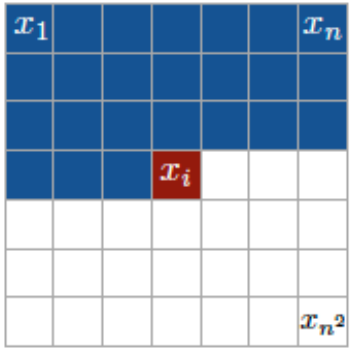
自回归生成模型

- PixelRNN和PixelCNN
  - 建模 $p(\mathrm{x})$

$$p(\mathbf{x}) = \prod_{i=1}^{n^2} p(x_i | x_1, \ldots, x_{i-1})$$

Likelihood of image x

Probability of the i-th pixel value given all previous pixels

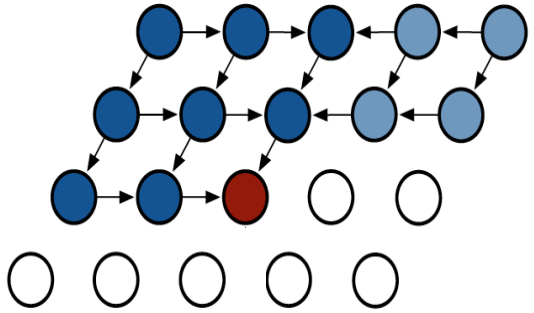


  - 利用训练数据进行最大似然估计

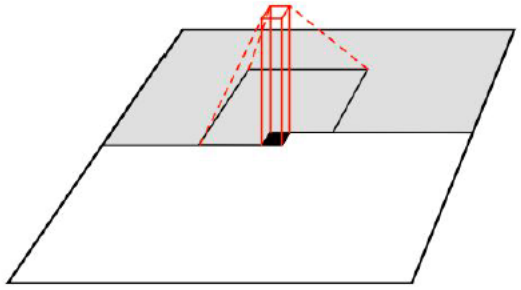
- A. van den Oord, N. Kalchbrenner and K. Kavukcuoglu. “Pixel Recurrent Neural Networks”. In ICML, 2016.

自回归生成模型

- PixelRNN
  - 利用RNN (BiLSTM)建模像素间的依赖关系
  - 缺点：训练、生成速度慢

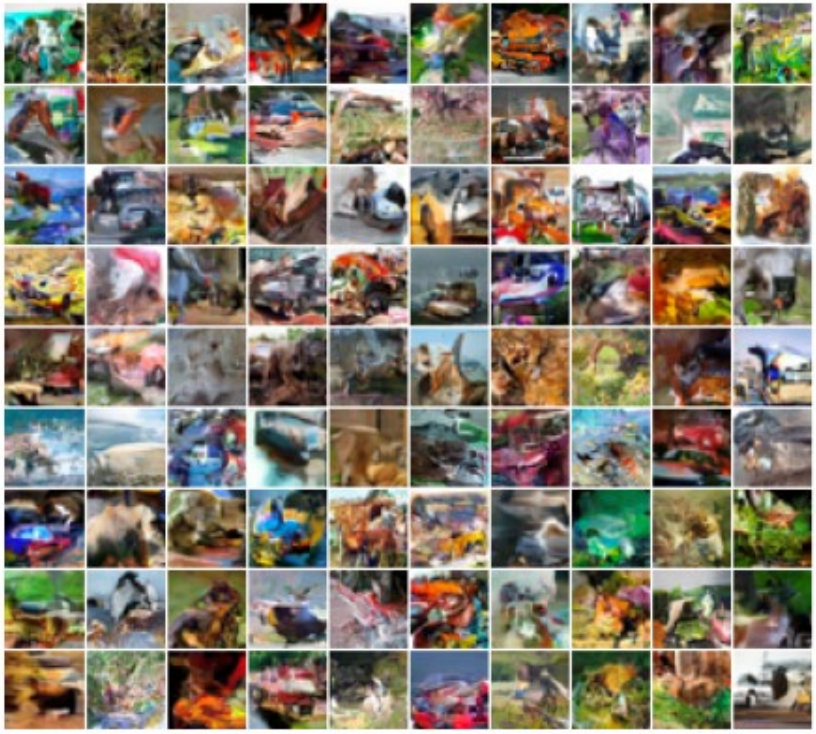


- PixelCNN
  - 利用CNN (masked conv)建模像间的依赖关系
  - 训练速度比PixelRNN快，生成速度依旧慢

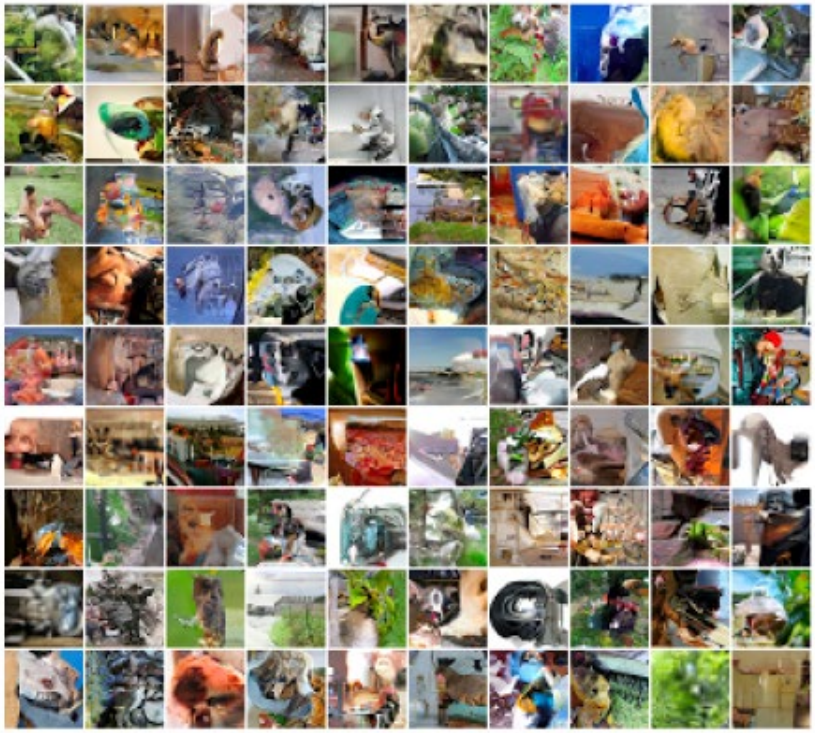


自回归生成模型

- 生成结果



32×32 CIFAR-10



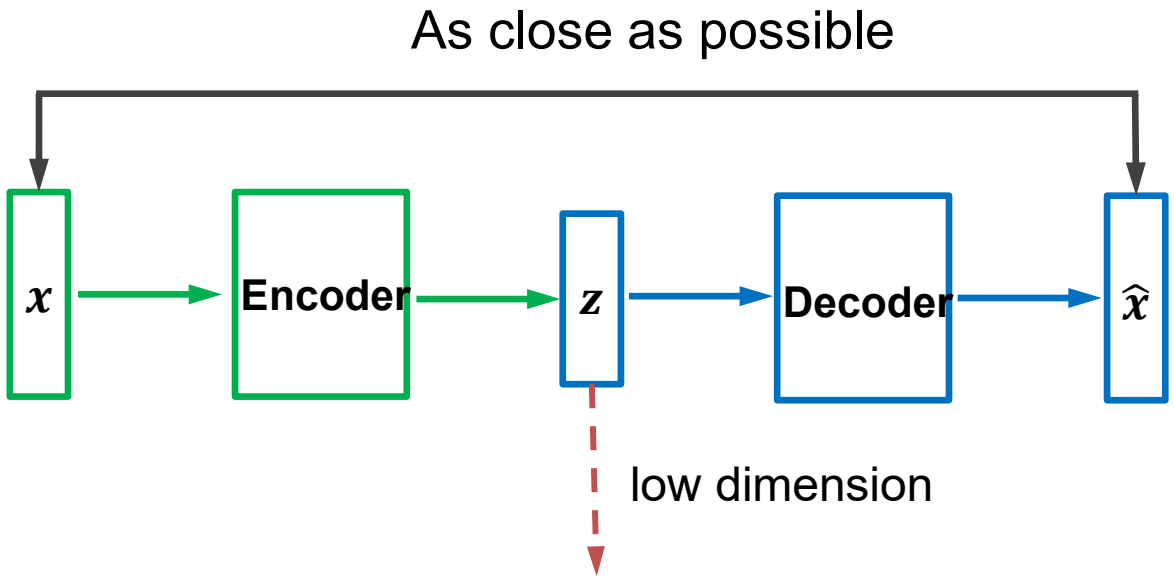
32×32 Imagenet

生成式模型（Generative Models）

- 生成式模型概论
- 自回归生成模型
- 变分自编码器
- 扩散模型
- 生成对抗网络
- 条件生成式模型

自编码器(Auto-Encoder)

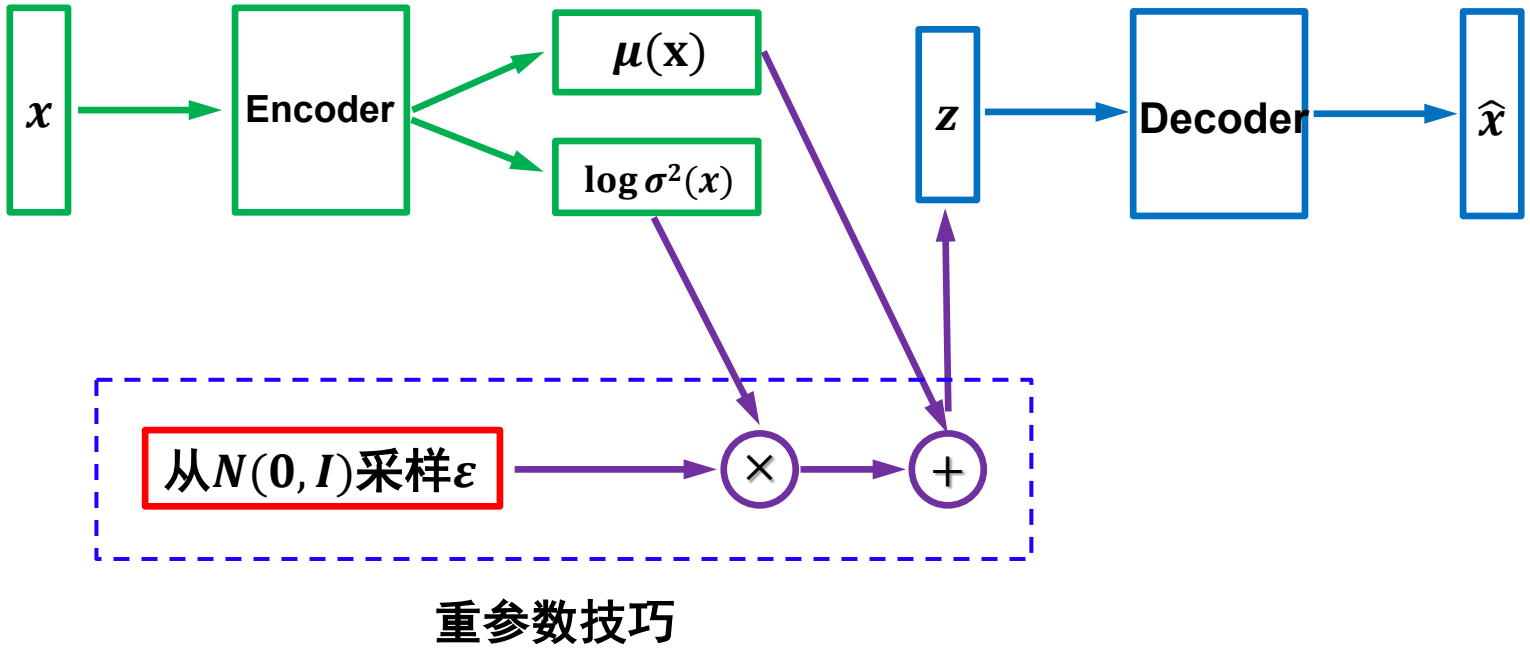
自编码器(Auto-Encoder)



- Embedding, feature, code
- Used as feature for downstream tasks
- Cannot generate new sample

变分自编码器（VAE）

- VAE的整体模型结构
  - 包含:编码器部分、解码器部分以及重参数化采样



- Diederik P. Kingma, and Max Welling. “Auto-Encoding Variational Bayes”. In arXiv:1312.6114, 2013.

变分自编码器（VAE）

- 观测样本与隐变量的联合分布

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

  - $\tilde{p}(x)$是根据样本$x_1, x_2, \ldots, x_n$确定的关于x的先验分布。尽管我们无法准确写出它的形式，但它是确定的、存在的。

- 设想用一个新的联合概率分布$q(x,z)$来逼近$p(x,z)$，并用KL散度来计算它们的距离：

$$KL(p(x,z)||q(x,z)) = \iint p(x,z)\log\frac{p(x,z)}{q(x,z)}dzdx$$

- 我们希望两个分布越接近越好，所以KL散度越小越好。

变分自编码器（VAE）

将$p(x,z)=\tilde{p}(x)p(z|x)$带入

$$KL(p(x,z)||q(x,z)) = \int \tilde{p}(x)[\int p(z|x)\log\frac{\tilde{p}(x)p(z|x)}{q(x,z)}dz]\,dx$$

$$= \mathbb{E}_{x\sim\tilde{p}(x)}[\int p(z|x)\log\frac{\tilde{p}(x)p(z|x)}{q(x,z)}dz]$$

进一步简化：

$$KL(p(x,z)||q(x,z)) = \mathbb{E}_{x\sim\tilde{p}(x)}[\int p(z|x)\log\tilde{p}(x)\,dz] + \mathbb{E}_{x\sim\tilde{p}(x)}[\int p(z|x)\log\frac{p(z|x)}{q(x,z)}dz)]$$

$$= \mathbb{E}_{x\sim\tilde{p}(x)}[\log\tilde{p}(x)\underbrace{\int p(z|x)dz}_{1}] + \mathbb{E}_{x\sim\tilde{p}(x)}[\int p(z|x)\log\frac{p(z|x)}{q(x,z)}dz)]$$

$$= \underbrace{\mathbb{E}_{x\sim\tilde{p}(x)}[\log\tilde{p}(x)]}_{\text{常量}C} + \mathbb{E}_{x\sim\tilde{p}(x)}[\int p(z|x)\log\frac{p(z|x)}{q(x,z)}dz)]$$

变分自编码器（VAE）

- 通过移项，我们可以令：

$$\mathcal{L} = KL(p(x,z)||q(x,z)) - C = \mathbb{E}_{x\sim\tilde{p}(x)}[\int p(z|x)\log\frac{p(z|x)}{q(x,z)}dz)]$$

- 最小化KL散度等价于最小化$\mathcal{L}$。为了得到生成模型，我们把$q(x,z)$写成$q(x|z)q(z)$，于是有：

$$\mathcal{L} = \mathbb{E}_{x\sim\tilde{p}(x)}[\int p(z|x)\log\frac{p(z|x)}{q(x|z)q(z)}dz)]$$

$$= \mathbb{E}_{x\sim\tilde{p}(x)}[-\int p(z|x)\log q(x|z)dz + \int p(z|x)\log\frac{p(z|x)}{q(z)}dz]$$

$$= \underbrace{\mathbb{E}_{x\sim\tilde{p}(x)}[\mathbb{E}_{z\sim p(z|x)}[-\log q(x|z)] + KL(p(z|x)||q(z))]}_{\text{优化目标}}$$

变分自编码器（VAE）

$$\mathcal{L} = \mathbb{E}_{x\sim\tilde{p}(x)}[\mathbb{E}_{z\sim p(z|x)}[-\log q(x|z)] + KL(p(z|x)||q(z))]$$

- 为了方便采样，我们假设$z\sim N(0,I)$，即标准的多元正态分布

- 假设$p(z|x)$也是(各分量独立的)正态分布，其均值与方差由x来决定（用一个神经网络计算）：

$$p(z|x) = \frac{1}{\prod_{k=1}^{d}\sqrt{2\pi\sigma_{(k)}^2(x)}}\exp\left(-\frac{1}{2}\left\|\frac{z-\mu(x)}{\sigma(x)}\right\|^2\right)$$

- x是神经网络的输入，输出则是均值$\mu(x)$与方差$\sigma^2(x)$。这里的神经网络就起到了类似Encoder的作用。$\mathcal{L}$中的KL散度项可以解析计算：

$$KL(p(z|x)||q(z)) = \frac{1}{2}\sum_{k=1}^{d}(\mu_{(k)}^2(x) + \sigma_{(k)}^2(x) - \log\sigma_{(k)}^2(x) - 1)$$

变分自编码器（VAE）

$$\mathcal{L} = \mathbb{E}_{x\sim\tilde{p}(x)}[\mathbb{E}_{z\sim p(z|x)}[-\log q(x|z)] + KL(p(z|x)||q(z))]$$

- 若假设$q(x|z)$是正态分布：

$$q(x|z) = \frac{1}{\prod_{k=1}^{D}\sqrt{2\pi\sigma_{(k)}^2(z)}}\exp\left(-\frac{1}{2}\left\|\frac{x-\mu(z)}{\sigma(z)}\right\|^2\right)$$

- 用神经网络建模，输入是z，输出是$\mu(z)$与$\sigma^2(z)$。

$$-\log q(x|z) = \frac{1}{2}\left\|\frac{x-\mu(z)}{\sigma(z)}\right\|^2 + \frac{D}{2}\log 2\pi + \frac{1}{2}\sum_{k=1}^{D}\log\sigma_{(k)}^2(z)$$

- 若固定方差为一个常数σ^2，则有：

$$-\log q(x|z) \sim \frac{1}{2\sigma^2}\|x-\mu(z)\|^2$$

- 最小化负对数似然等价于最小化MSE损失，$\mu(z)$起到了Decoder的作用。

变分自编码器（VAE）

- 回看VAE的优化目标：

$$\mathcal{L} = \mathbb{E}_{x\sim\tilde{p}(x)}[\mathbb{E}_{z\sim p(z|x)}[-\log q(x|z)] + KL(p(z|x)||q(z))]$$

  - 对于等号右侧的第二项， $p(z|x)$起到Encoder的作用，同时KL散度将Encoder输出的隐向量约束为标准的多元正态分布；
  - 对于等号右侧的第一项， $q(x|z)$起到Decoder的作用。若我们用MSE作为损失函数，这对应于$q(x|z)$为固定方差的多元正态分布；
  - 待训练完成后，Decoder就是我们的生成模型。生成过程就是从标准多元正态分布中采样得到隐变量z，再输入Decoder，就可以得到生成样本了。

变分自编码器（VAE）

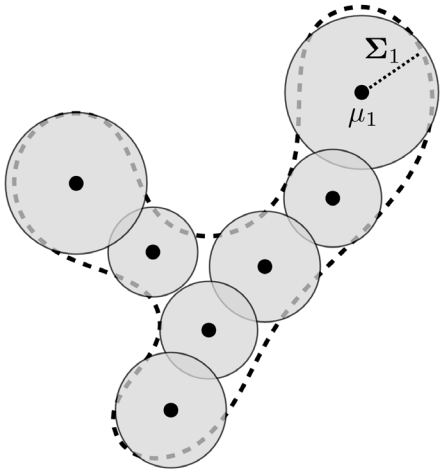
- VAE和自编码器
  - VAE首先通过Encoder将观测样本x编码（映射）为隐空间中的隐变量z，然后再将隐变量z重构回观测样本x。该训练过程与普通的自编码器类似，不同主要在于VAE多了隐变量采样和对隐变量的KL散度约束。
  - 一个训练好的自编码器，如果能够在它训练得到的隐空间采样，然后作为Decoder的输入，我们就得到了一个生成模型。但是普通自编码器的隐空间的分布是复杂且未知的，我们无法进行采样。VAE则对隐空间施加了KL散度约束，使其逼近简单的多元标准正态分布，这就使隐空间的采样变得可能，进而得到生成模型。
  - 所以，从训练过程来看，VAE就是隐空间受约束的自编码器。

变分自编码器（VAE）

VAE和高斯混合模型（GMM）

Finite mixture of Gaussians

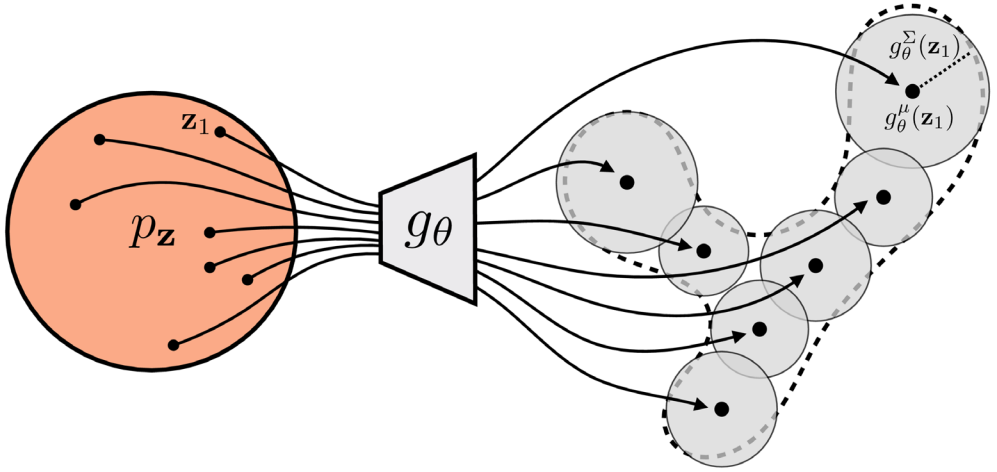
Parameters: $\{w_i, \mu_i, \mathbf{\Sigma}_i\}_{i=1}^{k}$



$$p_\theta(\mathbf{x}) = \sum_{i=1}^{k} w_i \mathcal{N}(\mathbf{x}; \mu_i, \mathbf{\Sigma}_i)$$

Infinite mixture of Gaussians (VAE)

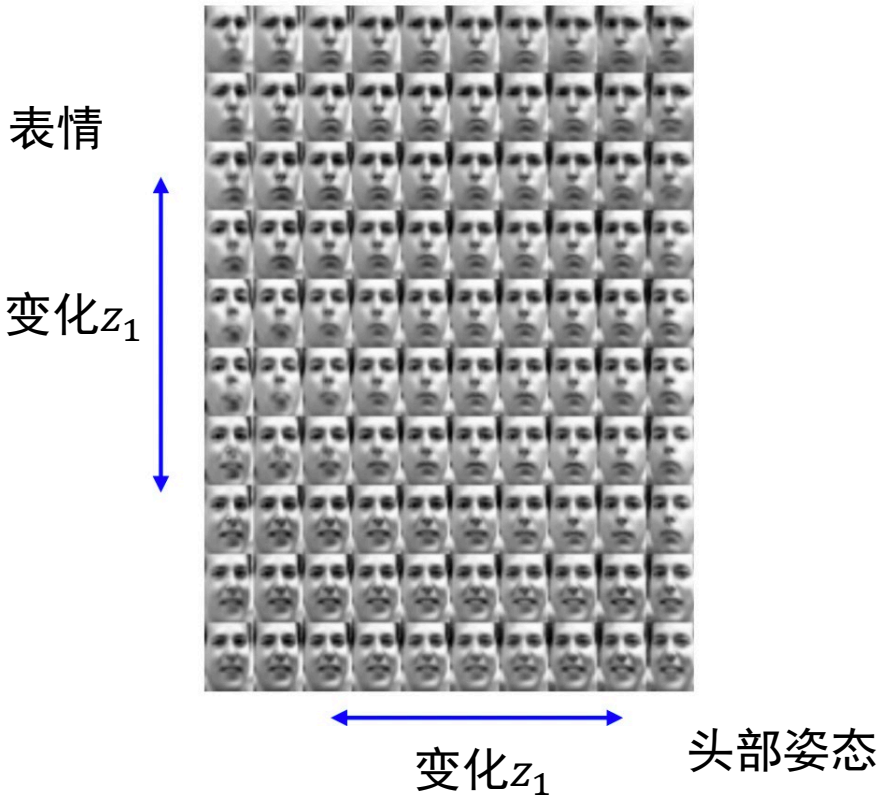
Parameters: θ



$$p_\theta(\mathbf{x}) = \int_\mathbf{z} p_\mathbf{z}(\mathbf{z}) \underbrace{\mathcal{N}(\mathbf{x}; g_\theta^{\mu}(\mathbf{z}), g_\theta^{\Sigma}(\mathbf{z}))}_{p(\mathbf{x}|\mathbf{z})} d\mathbf{z}$$

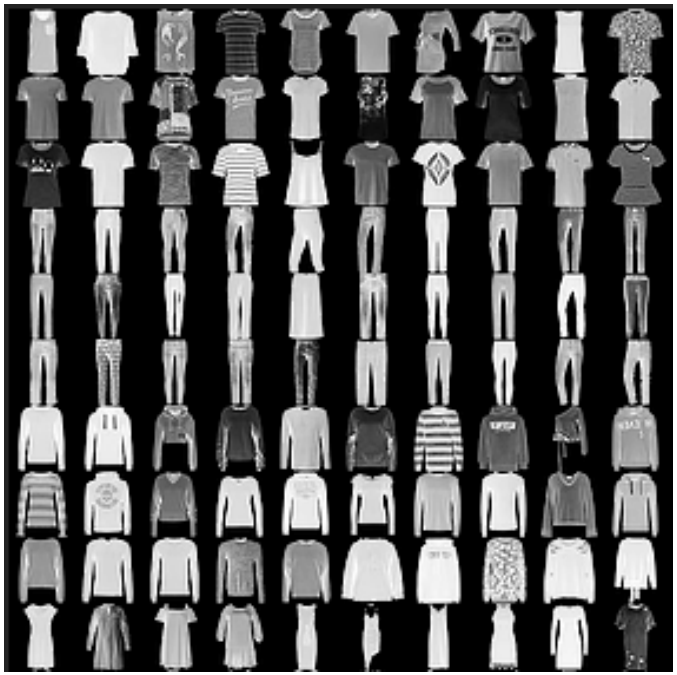
变分自编码器的优缺点

- VAE的优点
  - 得到 $p(z|x)$（编码器），可得到x的隐变量表示
  - 学习到具有一定可解释性的隐变量空间
  - VAE的训练过程比较稳定



变分自编码器的优缺点

- VAE的缺点
  - 变分法近似估计
  - VAE所生成的图片会比较模糊



训练图片

VAE随机生成

生成式模型（Generative Models）

- 生成式模型概论
- 自回归生成模型
- 变分自编码器
- 扩散模型
- 生成对抗网络
- 条件生成式模型

扩散模型（Diffusion Models）

Imagen
by Google

Sprouts in the shape of text 'Imagen' coming out of a fairytale book.

Stable diffusion

DALL·E
by OpenAI

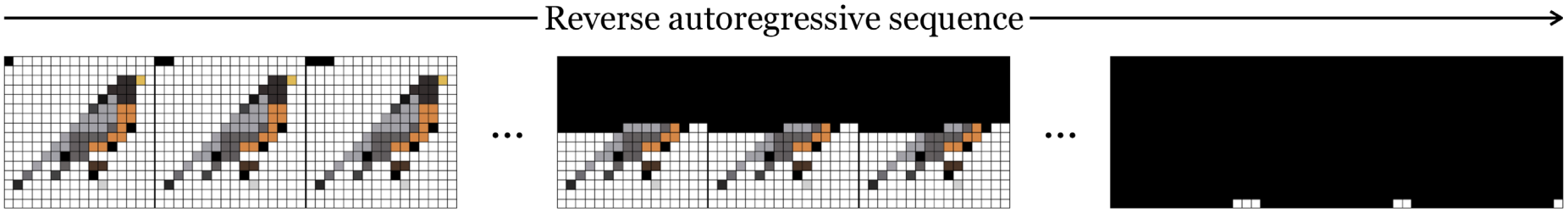
An astronaut riding a horse in photorealistic style.

Midjourney

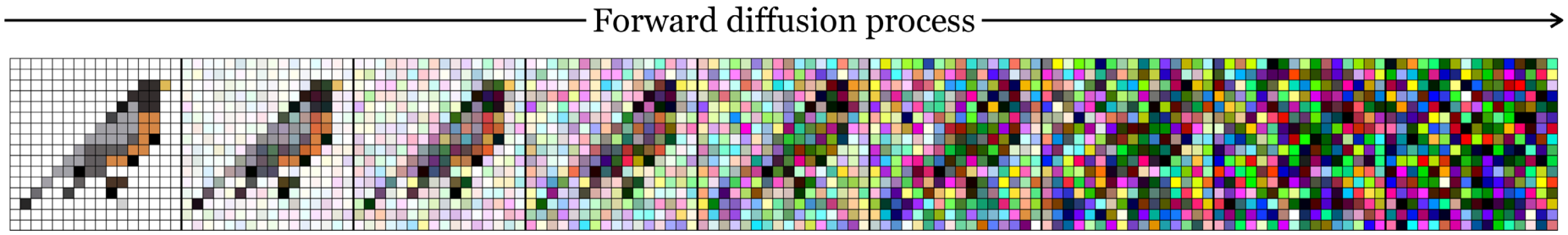
a cute fluffy bunny grumpily working on her trip itinerary.

扩散模型

- 自回归生成的逆过程

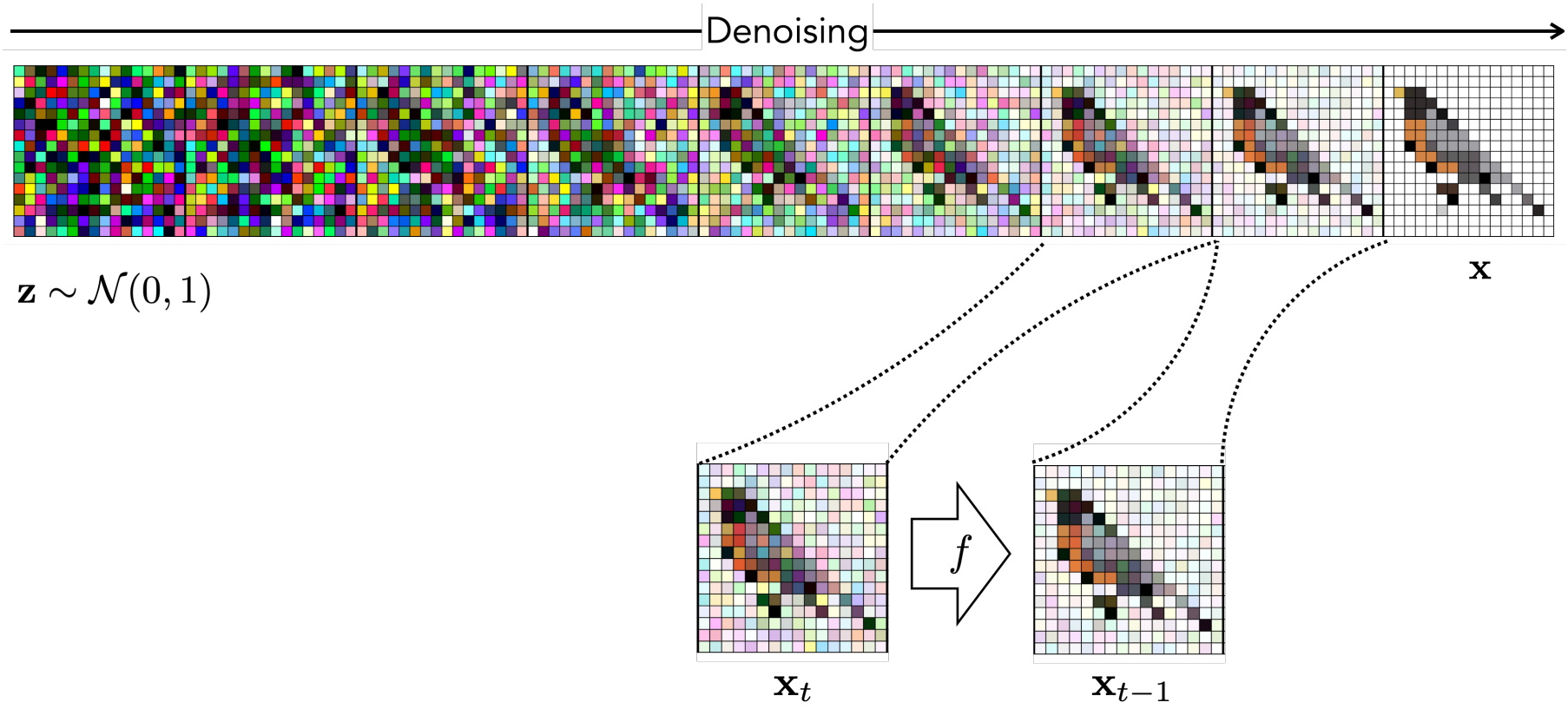


- 前向扩散过程

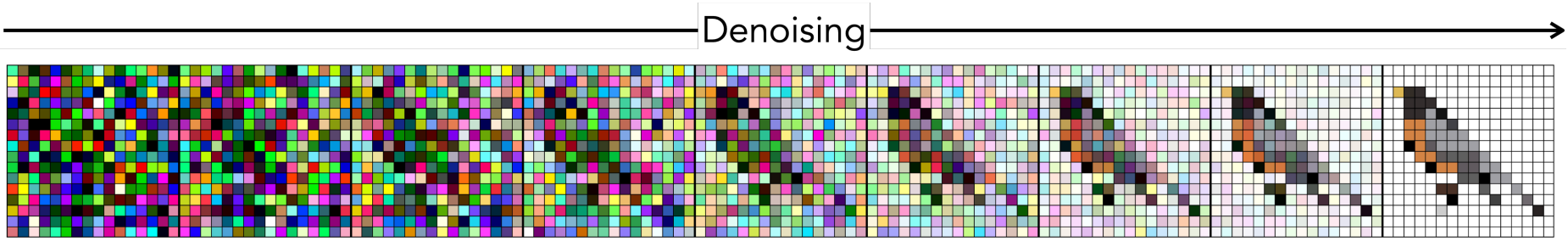


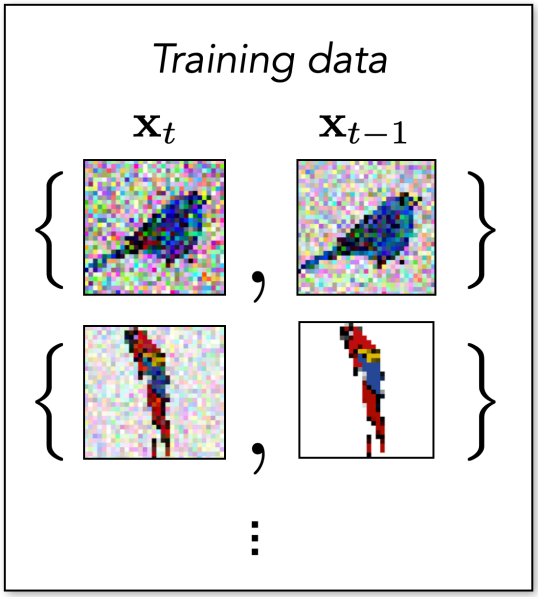
扩散模型

- 逆向去噪过程：通过逐步去噪实现生成



扩散模型





$$\arg\min_{f \in \mathcal{F}} \sum_{i=1}^{N} \mathcal{L}(f(\mathbf{x}_t), \mathbf{x}_{t-1})$$

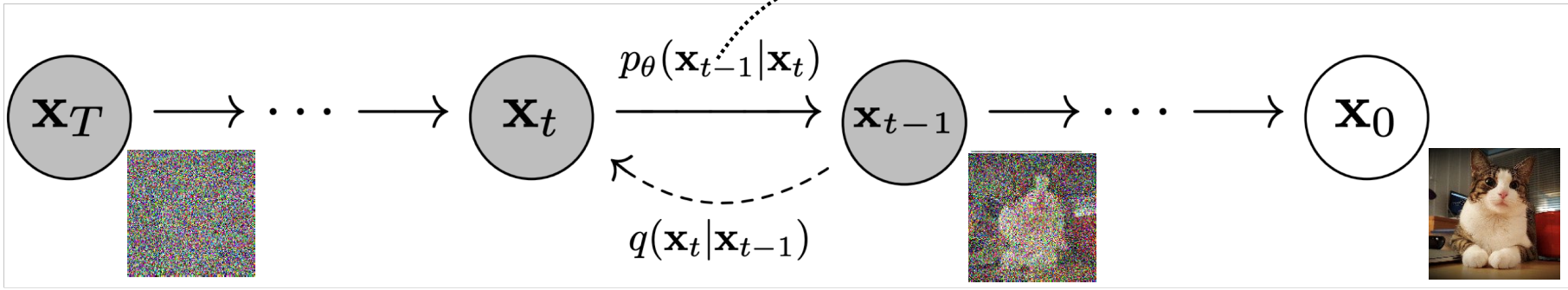
将生成建模转化为监督预测问题

扩散模型

马尔可夫链建模

$$p_\theta(\mathbf{x}_{t-1}|\mathbf{x}_t) = \mathcal{N}(f_\theta(\mathbf{x}_t, t), \sigma^2)$$

待学习



$$q(\mathbf{x}_t|\mathbf{x}_{t-1}) = \mathcal{N}(\sqrt{1-\beta_t}\mathbf{x}_{t-1}, \beta_t)$$

前向过程：

$$\epsilon_t \sim \mathcal{N}(\mathbf{0}, \mathbf{I}) \qquad \mathbf{x}_t = \sqrt{(1-\beta_t)}\mathbf{x}_{t-1} + \sqrt{\beta_t}\epsilon_t$$

逆向过程：

$$\mu = f_\theta(\mathbf{x}_t, t) \qquad \mathbf{x}_{t-1} \sim \mathcal{N}(\mu, \sigma^2)$$

扩散模型

训练过程

Algorithm 1.2: Training a diffusion model.

Input: training data $\{\mathbf{x}^{(i)}\}_{i=1}^{N}$
Output: trained model f_θ
Generate training sequences via diffusion:
for $i = 1, \ldots, N$ **do**
  for $t = 1, \ldots, T$ **do**
    $\epsilon_t \sim \mathcal{N}(\mathbf{0}, \mathbf{I})$
    $\mathbf{x}_t^{(i)} \leftarrow \sqrt{(1-\beta_t)}\mathbf{x}_{t-1}^{(i)} + \sqrt{\beta_t}\epsilon_t$

Train denoiser f_θ **to reverse these sequences:**
$\theta^* = \arg\min_\theta \sum_{i=1}^{N} \sum_{t=1}^{T} \mathcal{L}(f_\theta(\mathbf{x}_t^{(i)}, t), \mathbf{x}_{t-1}^{(i)}))$
Return: f_{θ^*}

实际

$$\mathbf{x}_t \longleftarrow \sqrt{\bar{\alpha}_t}\,\mathbf{x}_0 + \sqrt{1-\bar{\alpha}_t}\,\epsilon$$

$$\alpha_t = 1 - \beta_t, \quad \bar{\alpha}_t = \prod_{i=1}^{T} \alpha_i,$$

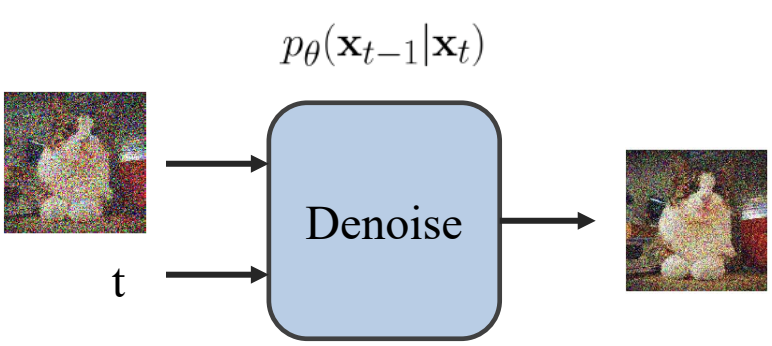
扩散模型

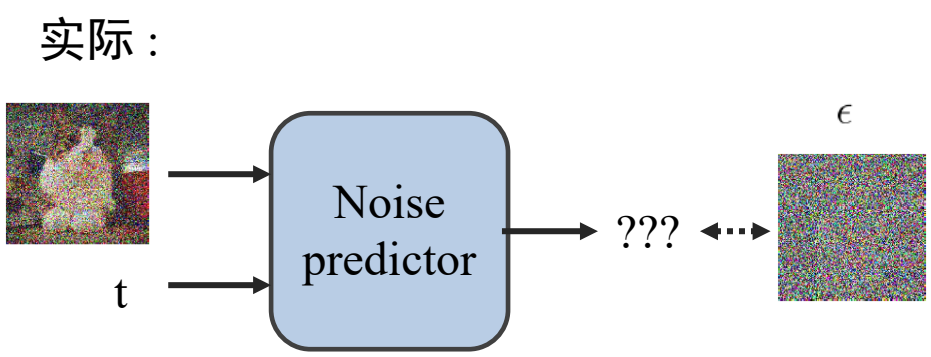
去噪扩散概率模型 (DDPM)

$$L_{\text{simple}} = \mathbb{E}_{\mathbf{x}_0 \sim q(\mathbf{x}_0), \epsilon \sim \mathcal{N}(\mathbf{0}, \mathbf{I}), t \sim \mathcal{U}(1,T)} \left[||\epsilon - \epsilon_\theta(\underbrace{\sqrt{\bar{\alpha}_t}\, \mathbf{x}_0 + \sqrt{1-\bar{\alpha}_t}\, \epsilon}_{\mathbf{x}_t}, t)||^2 \right]$$

Algorithm 1 Training

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



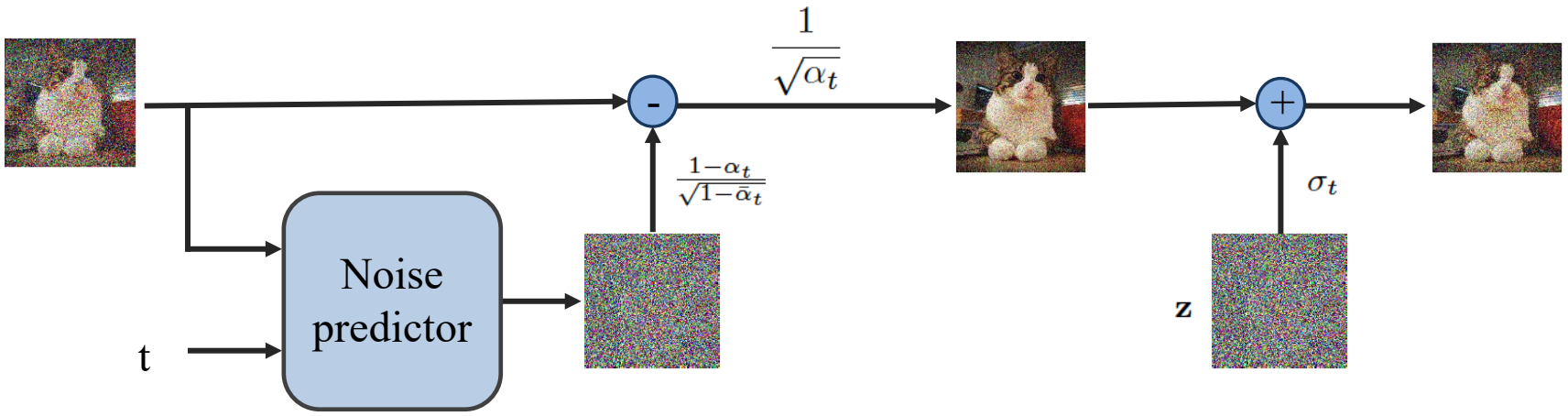


扩散模型

- 去噪扩散概率模型 (DDPM)

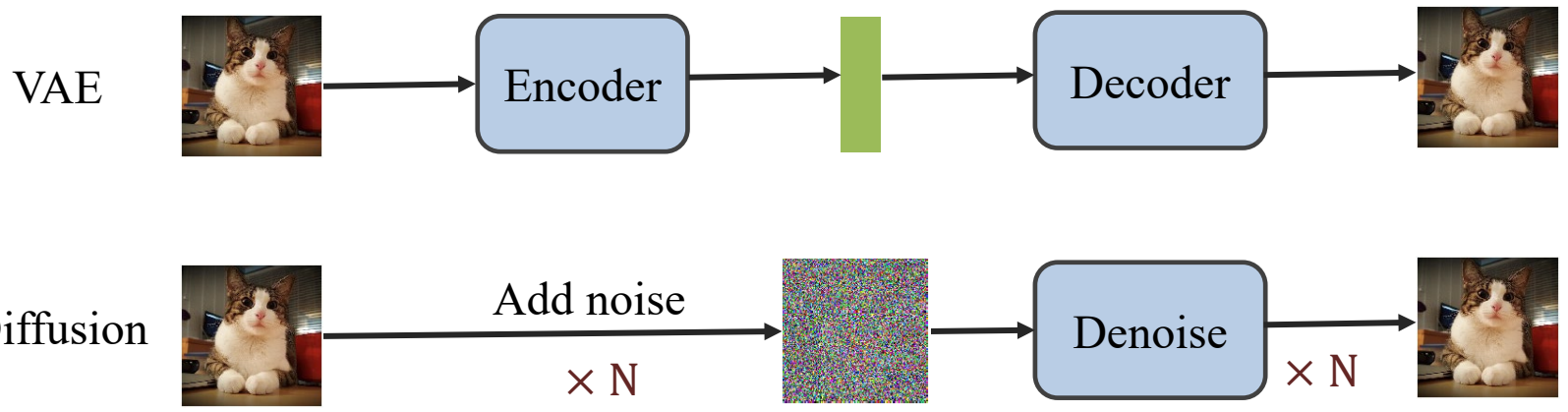
Algorithm 2 Sampling

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



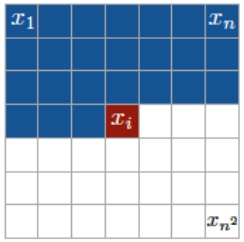
扩散模型

扩散模型 vs. VAE

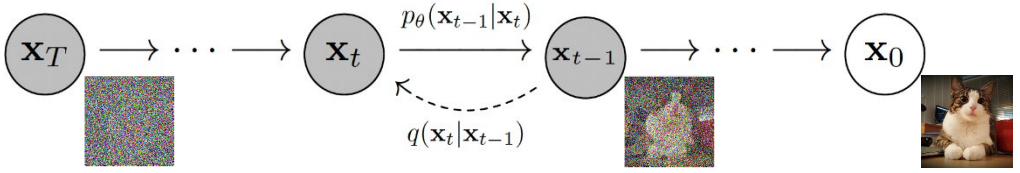


扩散模型 vs. 自回归生成模型

PixelRNN/PixelCNN



Diffusion

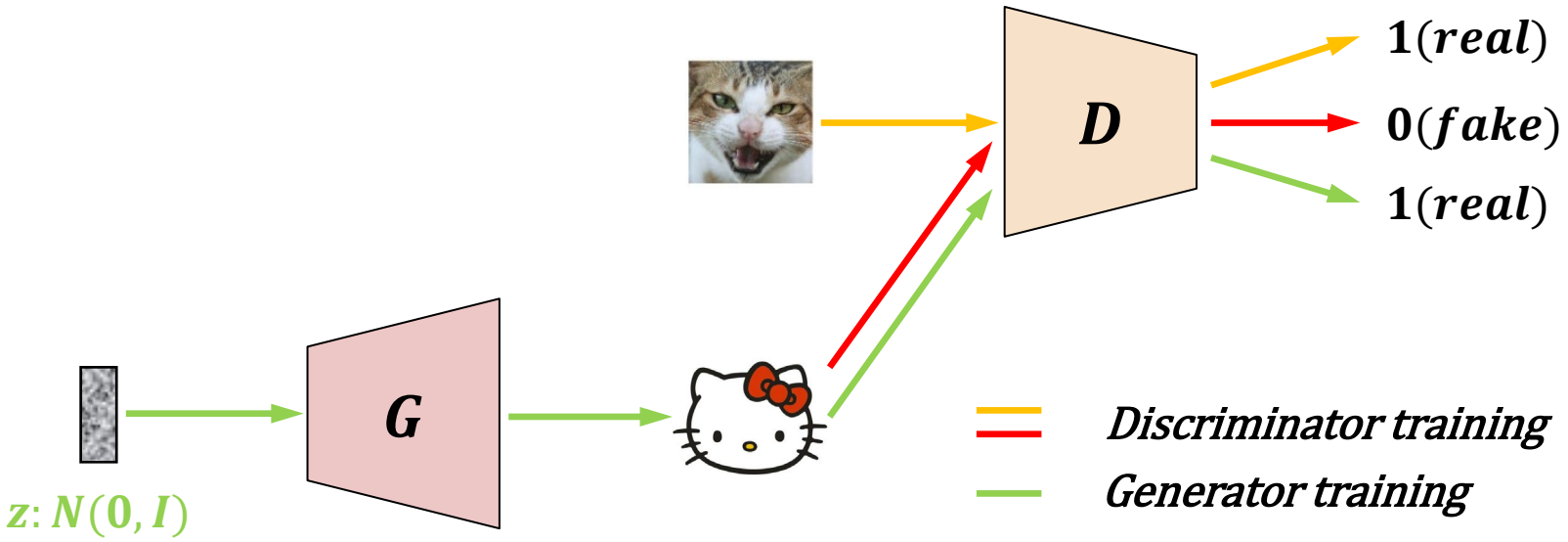


生成式模型（Generative Models）

- 生成式模型概论
- 自回归生成模型
- 变分自编码器
- 扩散模型
- 生成对抗网络
- 条件生成式模型

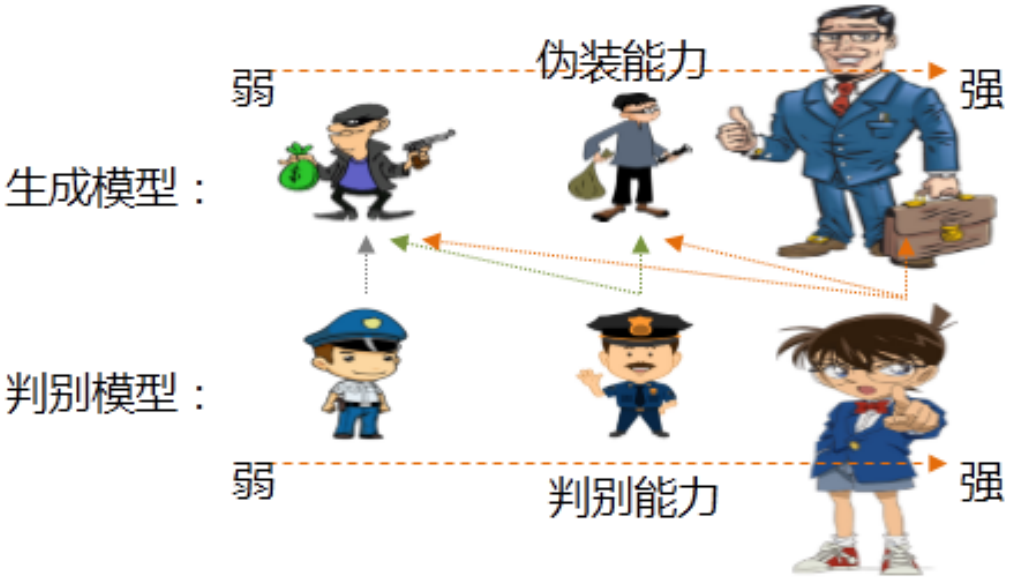
生成对抗网络（Generative Adversarial Networks）

- 生成对抗网络包含两个子网络：生成网络(G)和判别网络(D)
- 生成网络G的输入为均值为0，协方差矩阵为单位矩阵的高斯噪声，输出为生成的“假”数据；判别网络D可以看作一个二分类器，用于分辨输入的数据是“真”数据还是“假”数据。



生成对抗网络

- GAN受博弈论启发，将生成问题视作生成器G和判别器D这两个网络的对抗和博弈：前者试图产生更真实的数据，后者试图更准确地分辨真实数据与生成数据



- 由此，两个网络在对抗中进步，在进步后继续对抗，由生成式网络得的数据也就越来越完美，越来越逼近真实数据，从而可以生成想要得到的数据

生成对抗网络

- 设z为随机噪声，x为真实数据，判别器D可以看作一个二分类器，那么采用交叉熵损失，GAN的优化目标可以写作：

$$\min_{\theta_g}\max_{\theta_d}\left[\mathbb{E}_{x\sim p_{data}}\log D_{\theta_d}(x)+\mathbb{E}_{z\sim p(z)}\log(1-D_{\theta_d}(G_{\theta_g}(z)))\right]$$

- 其中第一项的$D(x)$表示判别器对真实数据的判断，第二项$D\big(G(z)\big)$则表示对生成数据的判断。
  - 判别器D的目标是最大化这个公式，也就是甄别出哪些数据是来自真实数据分布的。
  - 生成器G的目标是最小化这个公式，也就是让自己生成的数据被判别器判断为来自真实数据分布。
- 通过这样一个极大极小博弈，循环交替地分别优化G和D来训练所需要的生成式网络与判别式网络，直到到达Nash均衡点。

生成对抗网络

- Minmax目标函数

$$\min_{\theta_g}\max_{\theta_d}\left[\mathbb{E}_{x\sim p_{data}}\log D_{\theta_d}(x)+\mathbb{E}_{z\sim p(z)}\log(1-D_{\theta_d}(G_{\theta_g}(z)))\right]$$

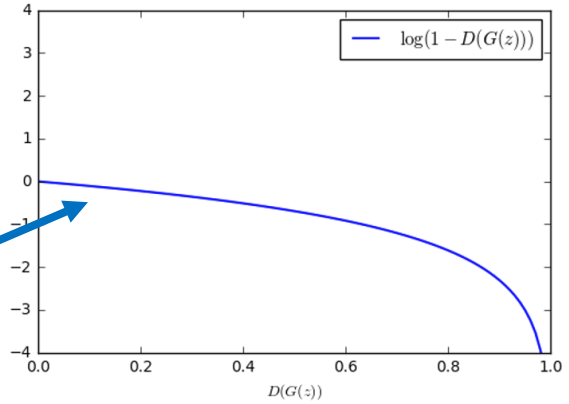
- 交替优化
  - 优化判别器（梯度上升）

$$\max_{\theta_d}\left[\mathbb{E}_{x\sim p_{data}}\log D_{\theta_d}(x)+\mathbb{E}_{z\sim p(z)}\log(1-D_{\theta_d}(G_{\theta_g}(z)))\right]$$

  - 优化生成器（梯度下降）

$$\min_{\theta_g}\mathbb{E}_{z\sim p(z)}\log(1-D_{\theta_d}(G_{\theta_g}(z)))$$

梯度消失



生成对抗网络

- Minmax目标函数

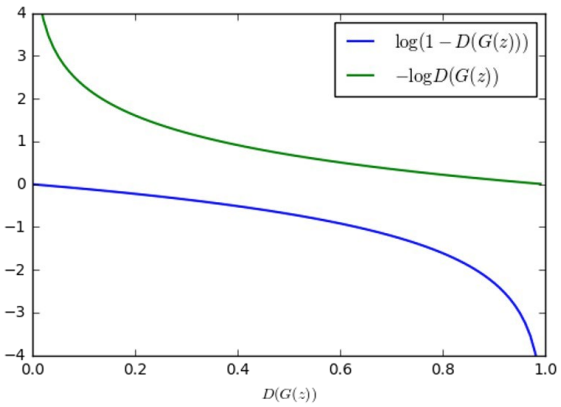
$$\min_{\theta_g}\max_{\theta_d}\left[\mathbb{E}_{x\sim p_{data}}\log D_{\theta_d}(x)+\mathbb{E}_{z\sim p(z)}\log(1-D_{\theta_d}(G_{\theta_g}(z)))\right]$$

- 交替优化
  - 优化判别器（梯度上升）

$$\max_{\theta_d}\left[\mathbb{E}_{x\sim p_{data}}\log D_{\theta_d}(x)+\mathbb{E}_{z\sim p(z)}\log(1-D_{\theta_d}(G_{\theta_g}(z)))\right]$$

  - 优化生成器（目标函数替换，梯度上升）

$$\max_{\theta_g}\mathbb{E}_{z\sim p(z)}\log(D_{\theta_d}(G_{\theta_g}(z)))$$



生成对抗网络的优缺点

- GAN的优点
  - 模型的训练不需要对隐变量做推断
  - 生成高质量图像
  - 判别器作为可自主学习的度量函数，定义十分灵活，使得GAN很容易与其他任务结合
- GAN的缺点
  - 训练不稳定
  - 模式丢失（mode collapse）问题 （只生成某些特定内容）
  - 缺少对$p(x), p(z|x)$的建模和计算

生成式模型（Generative Models）

- 生成式模型概论
- 自回归生成模型
- 变分自编码器
- 扩散模型
- 生成对抗网络
- 条件生成式模型

条件生成式模型

条件自回归模型

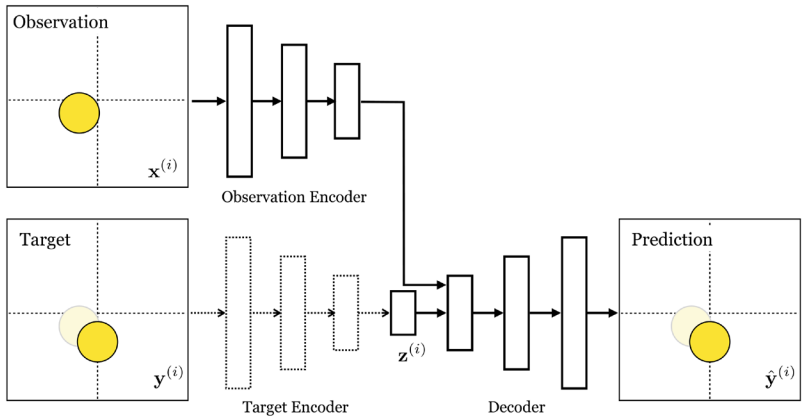
$$p_\theta(\mathbf{y} \mid \mathbf{x}) = \prod_{i=1}^{n} p_\theta(y_i \mid y_1, \ldots, y_{i-1}, x_1, \ldots, x_m)$$

条件变分自编码器（cVAE）

VAE likelihood model $\quad p_\theta(\mathbf{x}) = \int_{\mathbf{z}} p_\theta(\mathbf{x} \mid \mathbf{z}) p_{\mathbf{z}}(\mathbf{z}) d\mathbf{z}$

$$p_\theta(\mathbf{y} \mid \mathbf{x}) = \int_{\mathbf{z}} p_\theta(\mathbf{y} \mid \mathbf{z}, \mathbf{x}) p_{\mathbf{z}}(\mathbf{z} \mid \mathbf{x}) d\mathbf{z}$$

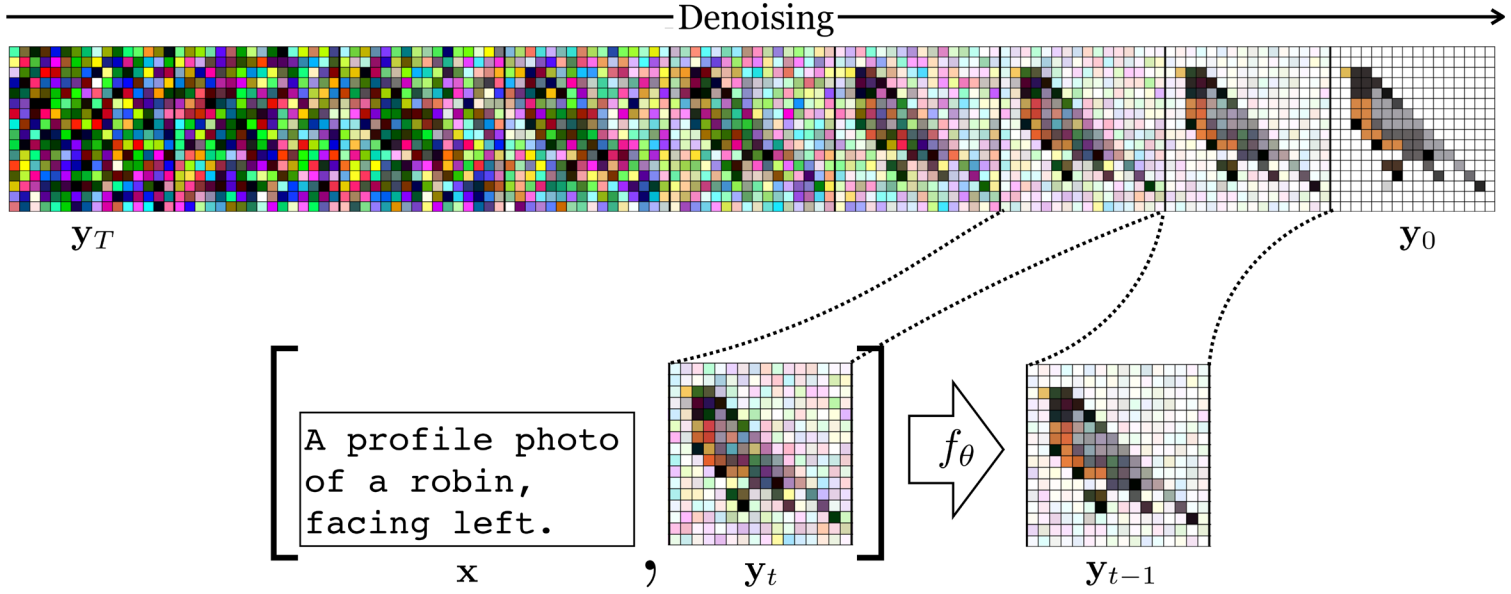
cVAE likelihood model $\quad p_\theta(\mathbf{y} \mid \mathbf{x}) = \int_{\mathbf{z}} p_\theta(\mathbf{y} \mid \mathbf{z}, \mathbf{x}) p_{\mathbf{z}}(\mathbf{z}) d\mathbf{z}$



条件生成式模型

- 条件扩散模型

$$\hat{\mathbf{y}}_{t-1} = f_\theta(\mathbf{y}_t, t, \mathbf{x})$$

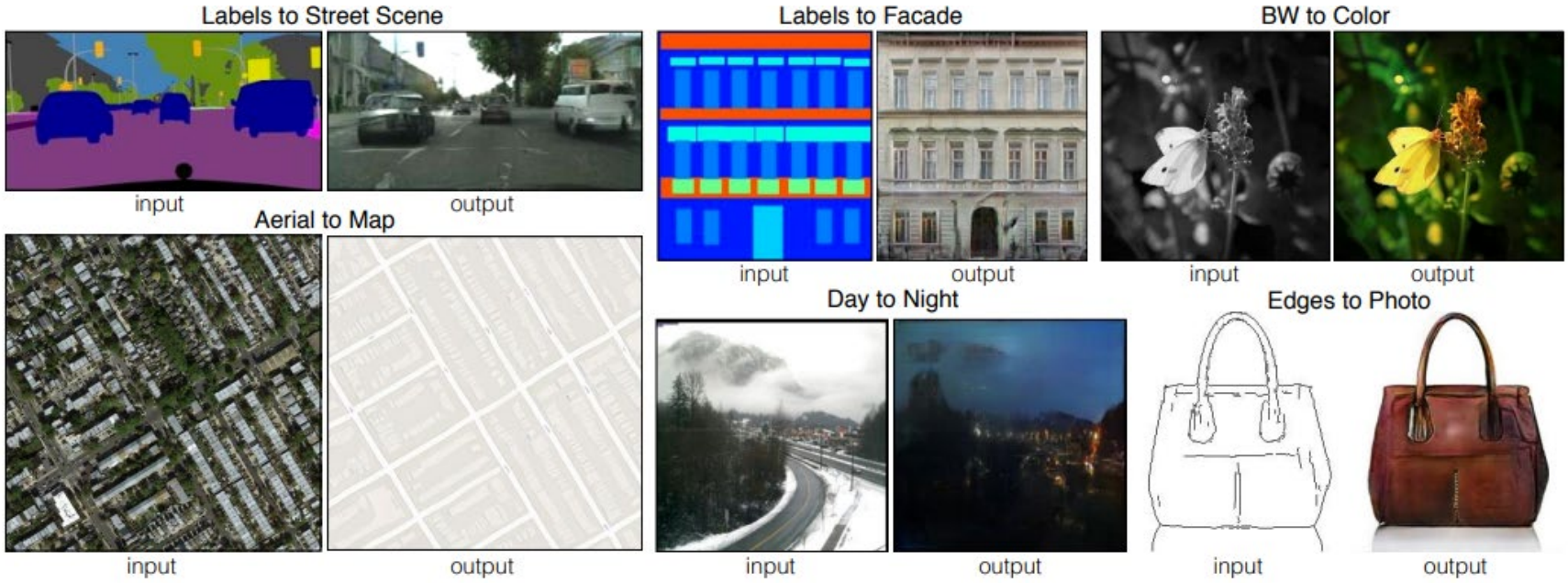


- 条件生成对抗网络（cGAN）

$$\arg\min_\theta \max_\phi \mathbb{E}_{\mathbf{z},\mathbf{x},\mathbf{y}}\left[\log d_\phi(\mathbf{x}, g_\theta(\mathbf{x},\mathbf{z})) + \log(1 - d_\phi(\mathbf{x},\mathbf{y}))\right]$$

应用举例

- 图像翻译（Image-to-image Translation）



$$\arg\min_G \max_D \mathbb{E}_{\mathbf{z},\mathbf{x},\mathbf{y}}\left[\log d_\phi(\mathbf{x}, g_\theta(\mathbf{x},\mathbf{z})) + \log(1 - d_\phi(\mathbf{x},\mathbf{y})) + \| g_\theta(\mathbf{x}) - \mathbf{y} \|_1\right]$$

PatchGAN $$d_\phi(\mathbf{x},\mathbf{y}) = \frac{1}{NM}\sum_{i=0}^{N}\sum_{j=0}^{M} d_\phi^{\texttt{patch}}(\mathbf{x}[i:i+k, j:j+k], \mathbf{y}[i:i+k, j:j+k])$$

- Isola, Phillip, et al. "Image-to-image translation with conditional adversarial networks". In CVPR, 2017.