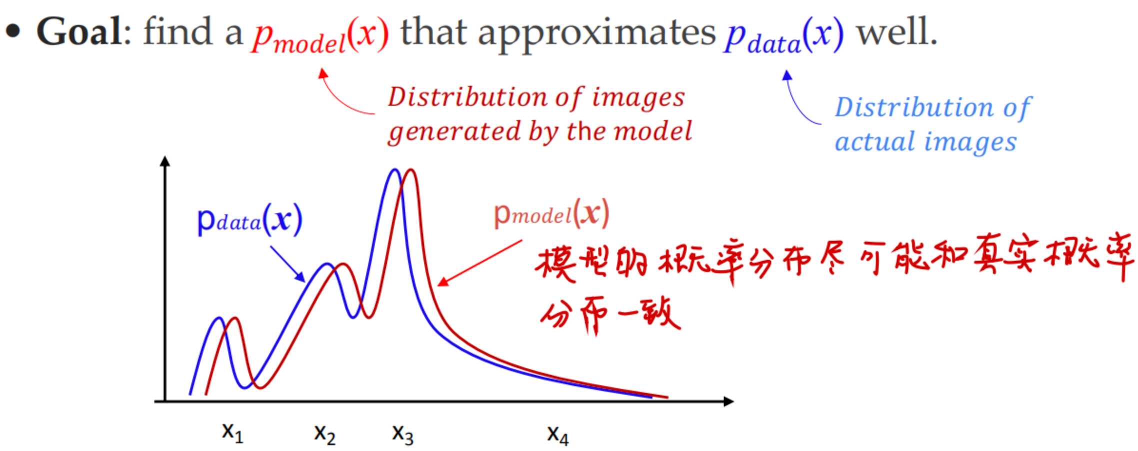
ch17: Deep Generative Models

🔑 这一章的重点是GAN，其他的两个更多的是了解性质

Concept

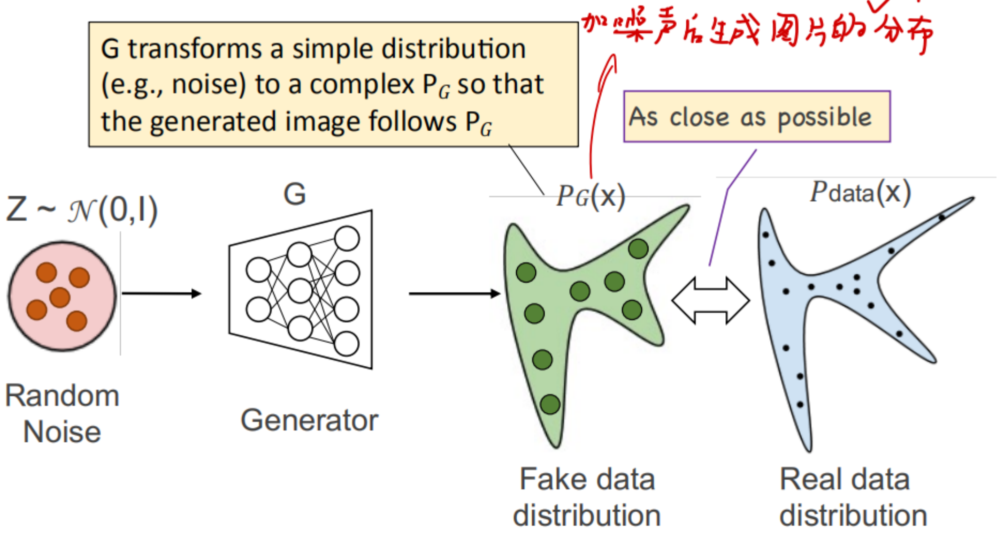
> Generation = learns the probability distribution of training data.

Generative Model overview(P17)



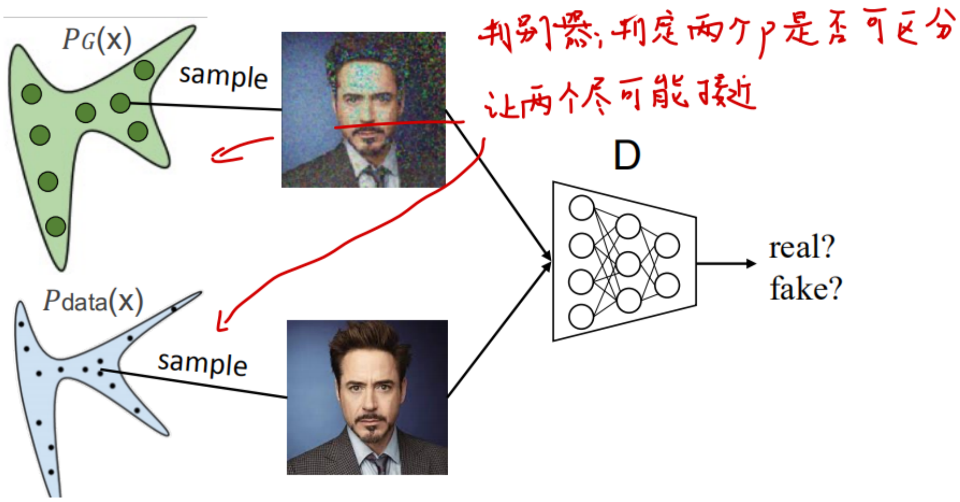
How to represent $p_{model}(x)$?(P18)

- **Generator G:** a neural network that generates an image x with a probability $\boldsymbol{p}_{\mathbf{model}}(\boldsymbol{x})$ given a random code. 概率



How to make $p_{model}(x) \approx p_{data}(x)$?

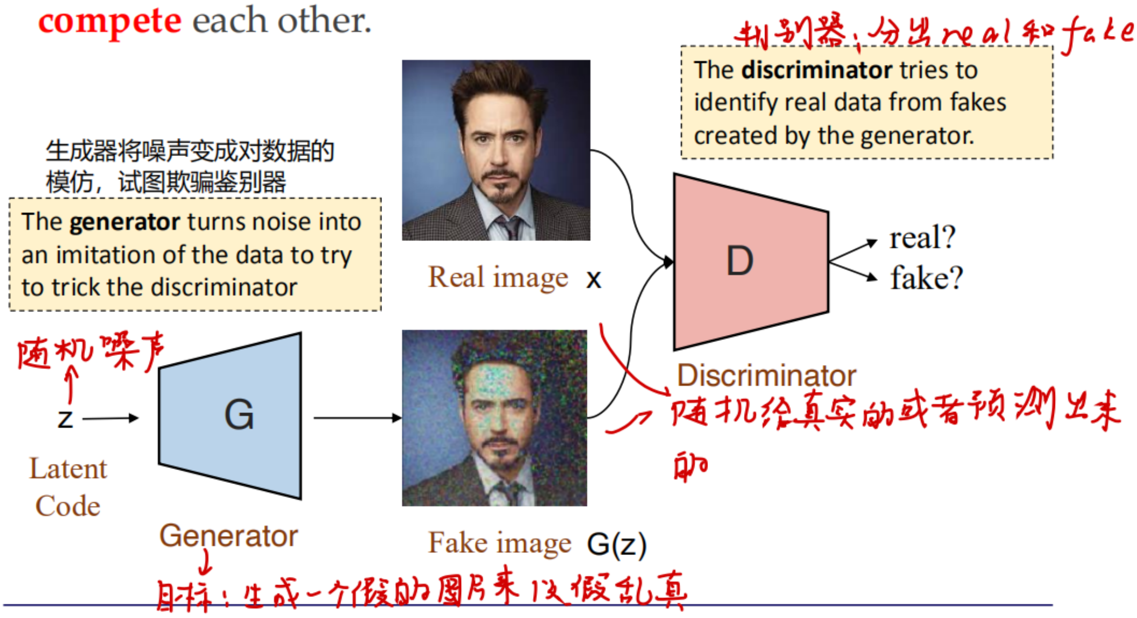
- **Discriminator D**: a neural network classifier that predicts whether a given image is sampled from $p_{model}(x)$ (fake) or $p_{data}(x)$ (real). If undistinguishable, then $p_{model}(x) \approx p_{data}(x)$



GAN(Generative Adversarial Network)

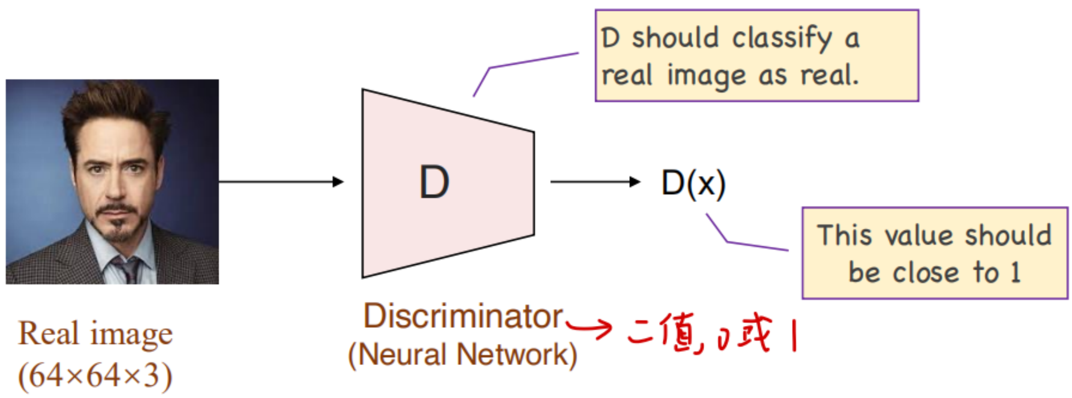
overview(P20)

- A generative model by having **two** neural networks **compete** each other.



Objective of Discriminator(P21-24)

- The discriminator tries to identify the synthesized instances



- The objective function can be formulated as the **binary cross-entropy**:

真的，尽可能接近1

模型生成出来的分类的概率

Sample x from real data distribution

Sample x from model distribution

$$D^* = \arg\max_D \mathrm{E}_{x \sim P_{data}}[\log D(x)] + \mathrm{E}_{x \sim P_G}[\log(1-D(x))]$$

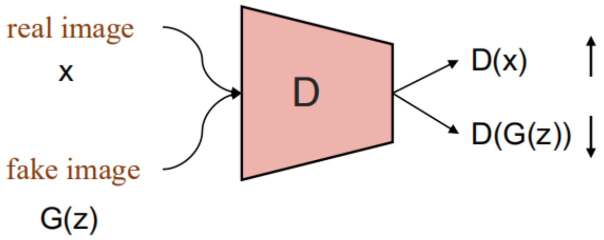
假的

$$= \arg\max_D \mathrm{E}_{x \sim P_{data}}[\log D(x)] + \mathrm{E}_{z \sim P_z(z)}[\log(1-D(G(z)))]$$

尽可能小

尽可能大

Sample z from unit Gaussian



Objective of Generator(P25)

G has an opposite (adversarial) objective to D !

G want D to fail

G和D的目标是相反的

G is independent of this part

G并不参与真实数据

$$G^* = \arg\min_G \cancel{\mathbf{E}_{x \sim P_{data}}[\log D(x)]} + \mathbf{E}_{z \sim P_z(z)}[\log(1-D(G(z)))]$$

$$= \arg\min_G \mathbf{E}_{z \sim P_z(z)}[\log(1-D(G(z)))]$$

Objective of GAN(P26)

Adversarial Objectives for G and D 两者是博弈的关系

$$\boxed{\min_G \max_D V(G,D) = \mathrm{E}_{x \sim P_{data}}[\log D(x)] + \mathrm{E}_{z \sim P_z(z)}[\log(1-D(G(z)))]}$$

Global Optimum: G reproduces the true data distribution

Algorithm(P27)

GAN

for number of training iterations **do**

  for k steps **do**

  - Sample minibatch of m noise samples $\{\boldsymbol{z}^{(1)}, \ldots, \boldsymbol{z}^{(m)}\}$ from noise prior $p_g(\boldsymbol{z})$. 高斯分布采样噪声
  - Sample minibatch of m examples $\{\boldsymbol{x}^{(1)}, \ldots, \boldsymbol{x}^{(m)}\}$ from data generating distribution $p_{\text{data}}(\boldsymbol{x})$.
  - Update the discriminator by ascending its stochastic gradient: 损失函数

$$\nabla_{\theta_d} \frac{1}{m} \sum_{i=1}^{m} \left[\log D\left(\boldsymbol{x}^{(i)}\right) + \log\left(1 - D\left(G\left(\boldsymbol{z}^{(i)}\right)\right)\right)\right].$$

  end for

- Sample minibatch of m noise samples $\{\boldsymbol{z}^{(1)}, \ldots, \boldsymbol{z}^{(m)}\}$ from noise prior $p_g(\boldsymbol{z})$.
- Update the generator by descending its stochastic gradient:

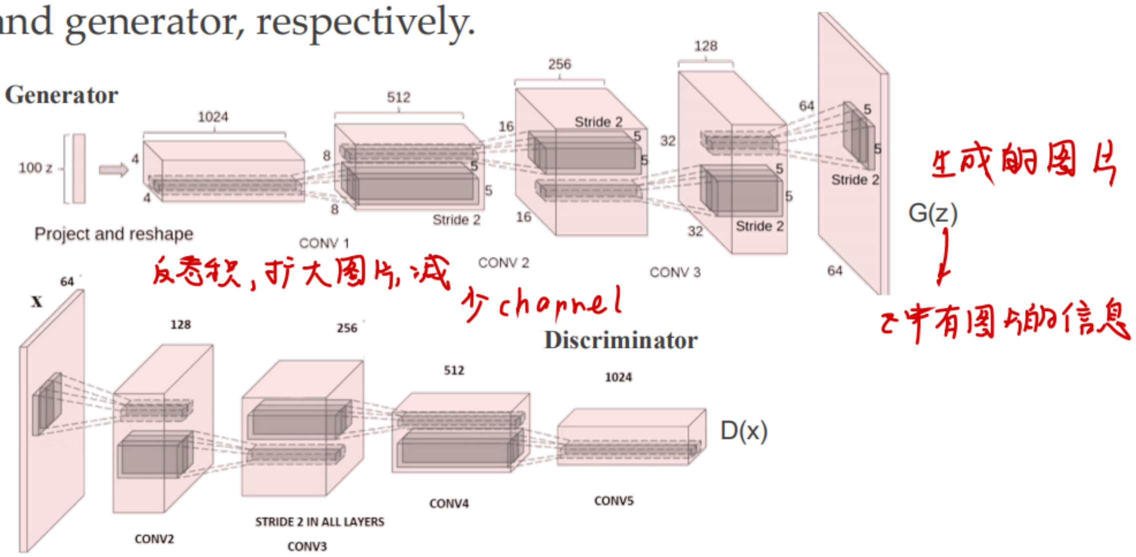
$$\nabla_{\theta_g} \frac{1}{m} \sum_{i=1}^{m} \log\left(1 - D\left(G\left(\boldsymbol{z}^{(i)}\right)\right)\right).$$

end for

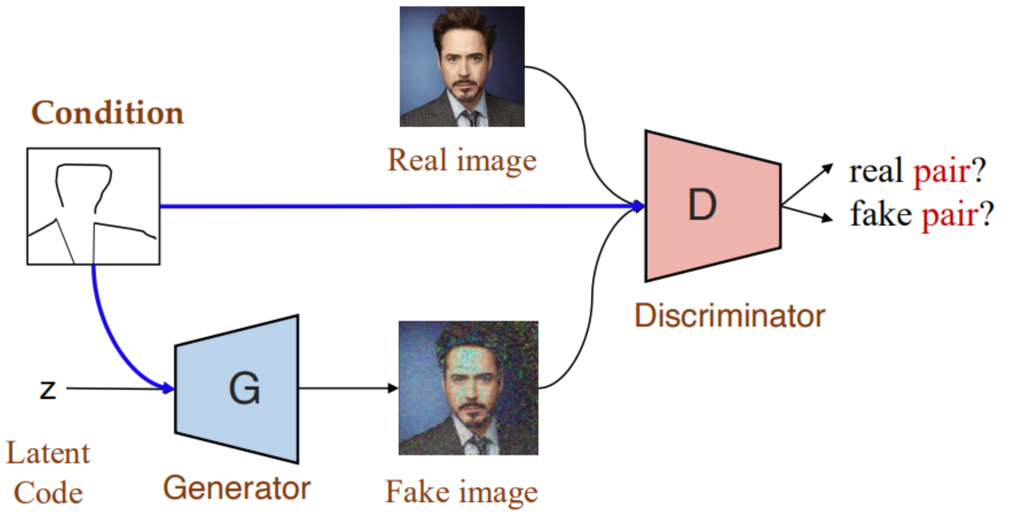
Some GANs

- **Deep Convolutional GANs (DCGAN)(P30)**

- Use **convolution** and **deconvolution** for the discriminator and generator, respectively.



- **Conditional GANs(P32)**

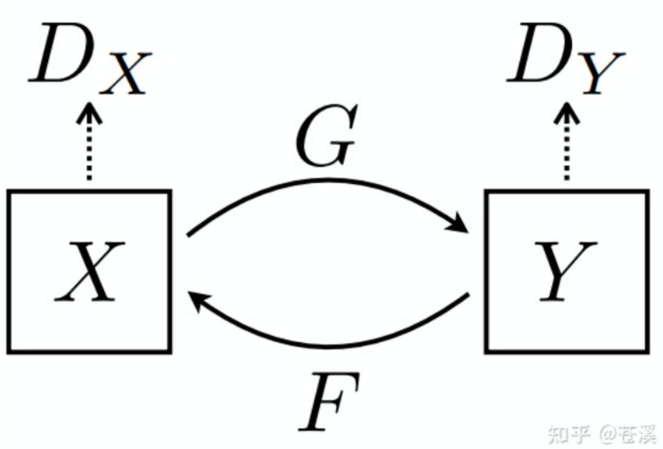


Results – Image-to-Image Translation

- **CycleGAN: Domain Transformation(P34)**

CycleGAN补充：

CycleGAN的主要目的是实现Domain Adaptation，这里我们以风景照片和梵高画作为例，假现在有两个数据集 X 和 Y 分别存放风景照片和梵高画作。我们希望训练出一个生成器 G ，它一个风景照，吐出一个梵高画作，即 $G(x)=y', x\in X$ ；同时，我们还希望训练出另一个生器 F ，它吃一个梵高画作，吐出一个风景照，即 $G(y)=x', y\in Y$ 。为了达到这个目的，们还需要训练两个判别器 D_X和D_Y ，分别判断两个生成器生成图片的好坏：如果生成器产生**图片** y' 不像**数据集** Y 里的**图片** y ，此时判别器D_Y应给它低分（规定最低分为0），反之如**片** y' 像**数据集** Y 里的**图片** y，则此时判别器D_Y应给他高分（规定最高分为1）。此外，判别D_Y应该永远给**真实图片** y 高分。对于判别器D_X也是同理。整个CycleGAN的模型框架如下图示:



在训练过程中，判别器和生成器是分别训练的，整个过程有点像进化论中捕食者和被捕食者迭代进化的过程：

当我们固定住生成器的参数训练判别器时，判别器便能学到更好的判别技巧，当我们固定住判别器参数训练生成器时，生成器为了骗过现在更厉害的判别器，被迫产生出更好质量的图片。两者便在这迭代学习的过程中逐步进化，最终达到动态平衡。

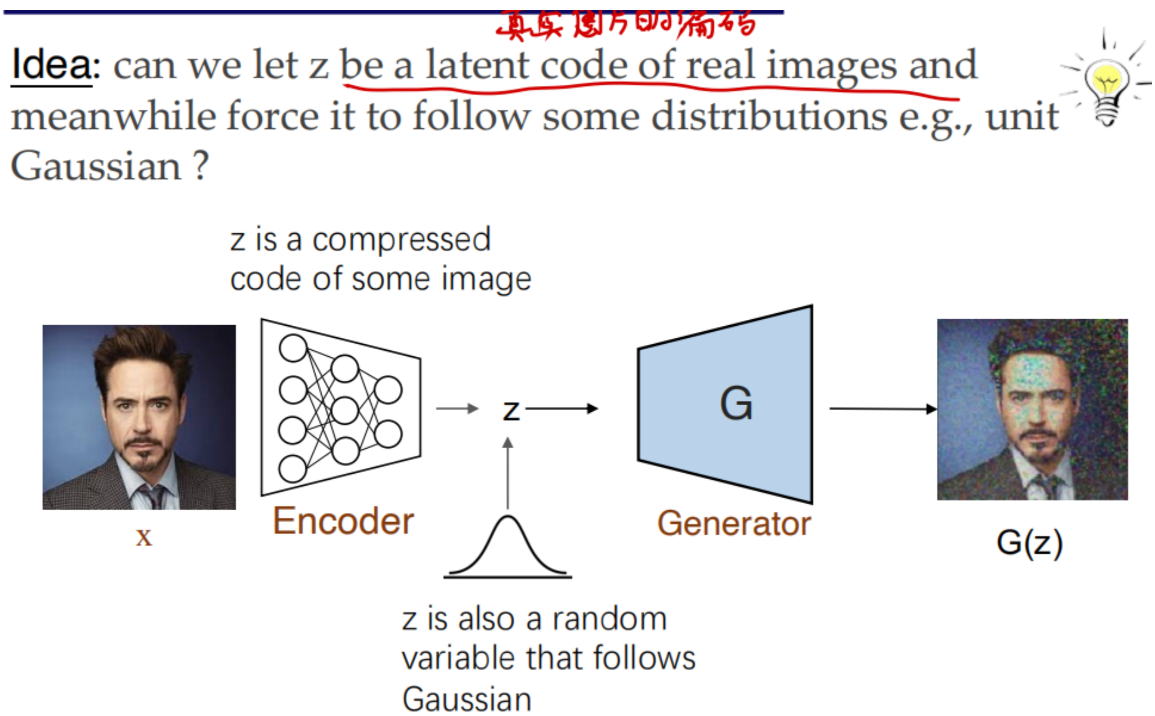
这是一开始GAN被提出时的思想，而在CycleGAN中，我们不仅想要生成器产生的**图片**y'跟**数据集**Y中的图片**画风一样**，我们还希望生成器产生的**图片**y'跟他的输入**图片** x **内容一样**（比如我们输入一张带房子的照片，我们希望产生梵高画风的房子图片，而不是其他内容的图片，比如向日葵、星夜等等，否则意义就不是很大了。因为在一些APP中，我们通常想把自己拍摄的照片变成梵高画作嘛~）。

为了做到这一点，论文作者提出了**Cycle Consistency Loss**，即将**图片**y'再放入生成器 F 中，产生的**新图片** x' 和最开始的**图片** x 尽可能的相似

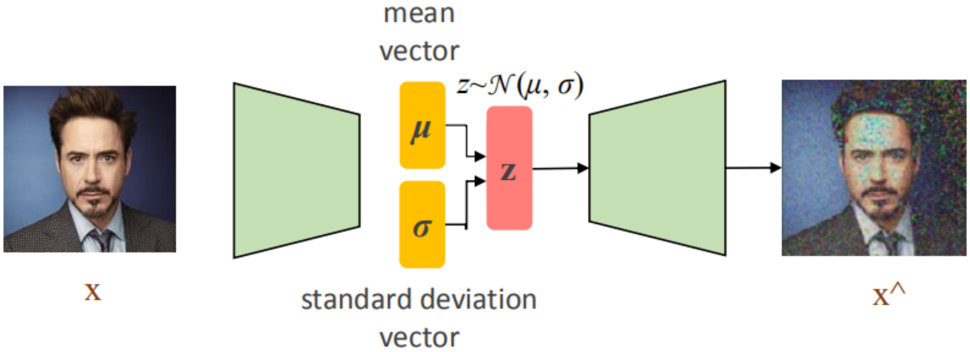
即我们希望 $F(G(x)) = x$

有了这样的约束，我们才能希望生成器将我们的输入照片的风格转化为梵高画风，而不是从梵高的作品中随便挑一张图片来应付我们了事。**Cycle Consistency Loss**在原论文中的插图解释为

Variational Autoencoder (VAE)（P40-42）



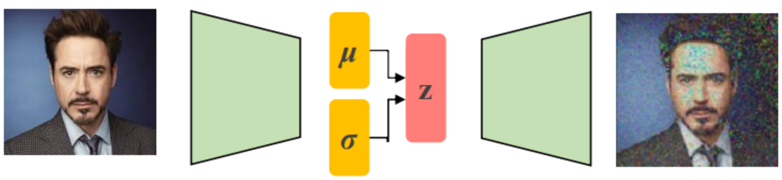
- An **autoencoder** where the latent code is sampled from a predicted Gaussian distribution.



Variational autoencoders are a probabilistic twist on autoencoders!
Sample from the mean and standard deviation to compute latent sample
变分自编码器是自编码器的概率扭曲:样本从均值和标准差来计算潜在样本

Training

Sampling is nondifferentiable
⇒Reparameterization trick: $z = \mu + \sigma \odot \varepsilon$ 使可导



Encoder computes: $q_\phi(z|x)$ Decoder computes: $p_\theta(x|z)$

得到μ和σ

编码器得到的z的分布希望尽可能接近标准正态分布

z是随机变量

maximize the log likelihood

Inferred latent distribution

Prior dist. of z, commonly unit Gaussian

$$L(\theta, \phi|x) = \underbrace{-\mathbf{E}_{z \sim q_\phi(z|x)}[\log p_\theta(x|z)]}_{\text{Reconstruction Loss}} + \underbrace{\mathbf{KL}(q_\phi(z|x) \,||\, p(z))}_{\text{Regularization}}$$

真实的概率分布

预测出来的图片和原始的图片尽可能像

预测的概率分布

尽可能接近

Xiaodong Gu Machine Learning: Lecture 17

Why Regularization（P43）

What properties do we want to achieve from regularization?

连续性：在潜在空间中接近的点，→解码后的相似内容

1. **Continuity**: points that are close in latent space → similar contents after decoding 连续的空间
2. **Completeness**: sampling from latent space → always "meaningful" content after decoding

Limitations of VAEs (and also GANs)（P45）

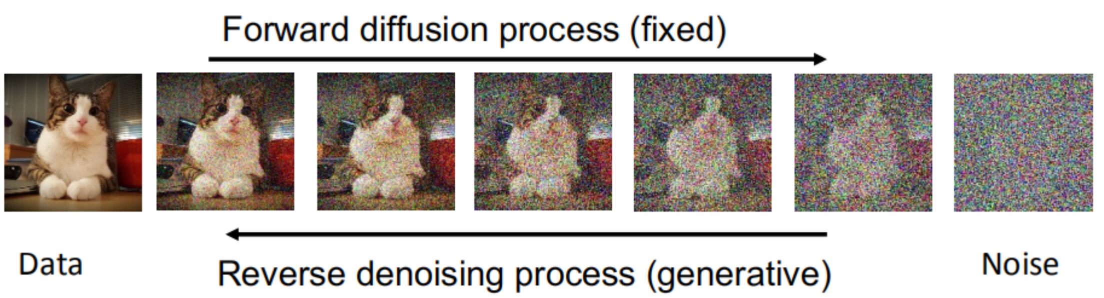
- The latent code z has a different (much smaller) size to the original image. 会导致图片质量的下降
- The content is generated all at once.

Denoising Diffusion Model

> **由粗到细生成,解决generated all at once的问题**

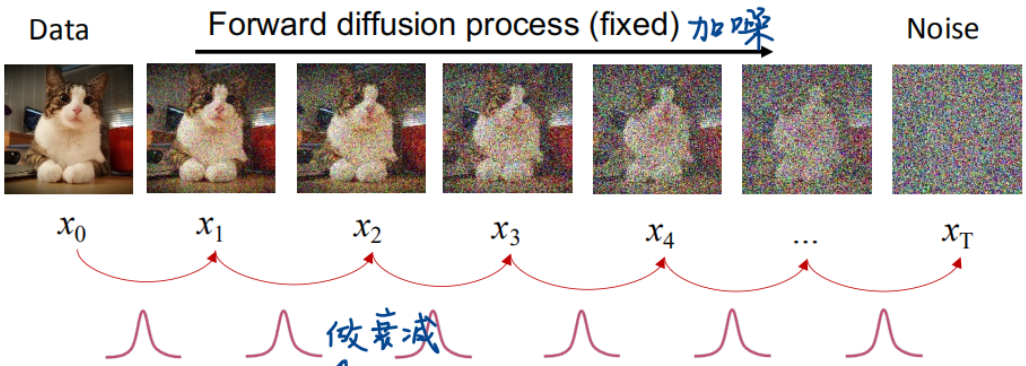
Overview（P47）

- Two processes:
  - ▻ **Forward diffusion** 扩散 **process**: gradually adds noise to input
  - ▻ **Reverse denoising process**: learns to generate data by denoising



Forward Diffusion Process（P48）

T-steps: each step (x_t) adds a fixed gaussian noise (β_t) to the previous step (x_{t-1}). 确定的，可以操纵的



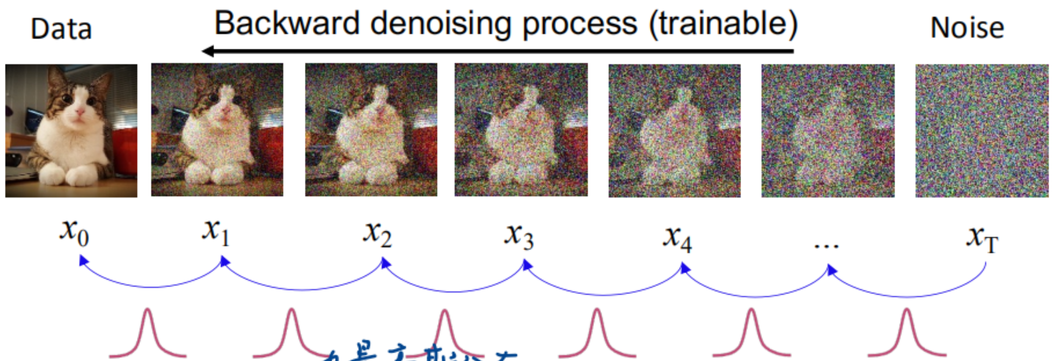
做衰减 加扰动

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

$$\text{Let } \bar{\alpha}_t = \prod_{s=1}^{t}(1-\beta_s) \implies x_t \sim q(\mathbf{x}_t|\mathbf{x}_0) = \mathcal{N}(\mathbf{x}_t; \sqrt{\bar{\alpha}_t}\mathbf{x}_0, (1-\bar{\alpha}_t)\mathbf{I}))$$

Backward Denoising Process（P49）

T steps: each step (x_{t-1}) removes some noise from the previous step (x_t). 可训练 但是比较难



也是高斯分布

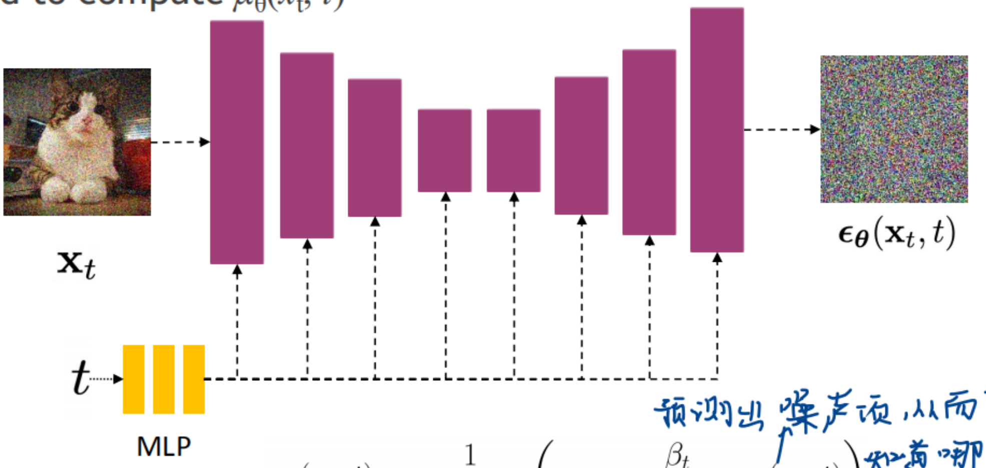
The distribution (μ and σ^2) of x_{t-1} can be estimated based on the previous image (x_t) using a neural network: 通过神经网络

$$p_\theta(\mathbf{x}_{t-1}|\mathbf{x}_t) = \mathcal{N}(\mathbf{x}_{t-1}; \mu_\theta(\mathbf{x}_t, t), \sigma_t^2\mathbf{I}) \qquad p(\mathbf{x}_T) = \mathcal{N}(\mathbf{x}_T; \mathbf{0}, \mathbf{I})$$

Trainable network (U-net, Denoising Autoencoder)

Architecture（P50）

Often use U-Net architectures with ResNet blocks and self-attention layers to represent $\varepsilon_\theta(x_t, t)$, which can further be used to compute $\mu_\theta(x_t, t)$



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

预测出噪声项，从而可以知道哪些噪声应该被舍掉

Xiaodong Gu　　Machine Learning: Lecture 17　　50

Algorithm（P51）

Algorithm 1 Training

1: **repeat**
2: $\quad \mathbf{x}_0 \sim q(\mathbf{x}_0)$ → 原始图片
3: $\quad t \sim \mathrm{Uniform}(\{1, \ldots, T\})$
4: $\quad \boldsymbol{\epsilon} \sim \mathcal{N}(\mathbf{0}, \mathbf{I})$ → 噪声
5: $\quad$ 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

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: $\quad \mathbf{x}_{t-1} = \frac{1}{\sqrt{\alpha_t}}\left(\mathbf{x}_t - \frac{1-\alpha_t}{\sqrt{1-\bar{\alpha}_t}}\boldsymbol{\epsilon}_\theta(\mathbf{x}_t, t)\right) + \sigma_t \mathbf{z}$
5: **end for**
6: **return** $\mathbf{x}_0$