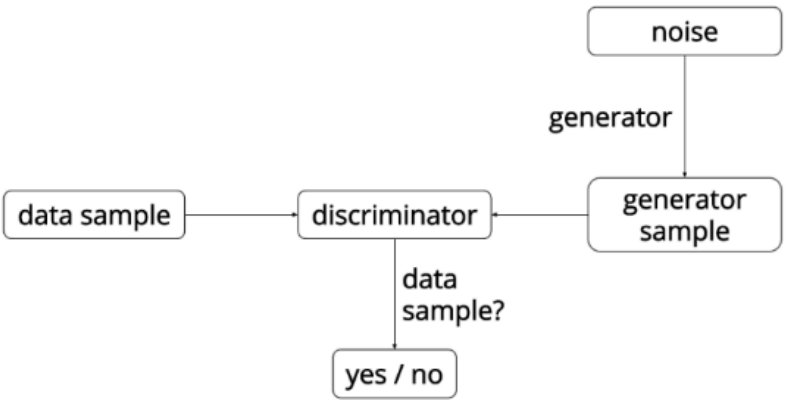
Chapter 5

Basic 1

GAN包含有两个模型，一个是生成模型（generative model），一个是判别模型(discriminative model)。生成模型的任务是生成看起来自然真实的、和原始数据相似的实例。判别模型的任务是判断给定的实例看起来是自然真实的还是人为伪造的（真实实例来源于数据集，伪造实例来源于生成模型）。

这可以看做一种零和游戏。论文采用类比的手法通俗理解：生成模型像“一个造假团伙，试图生产和使用假币”，而判别模型像“检测假币的警察”。生成器（generator）试图欺骗判别器（discriminator），判别器则努力不被生成器欺骗。模型经过交替优化训练，两种模型都能得到提升，但最终我们要得到的是效果提升到很高很好的生成模型（造假团伙），这个生成模型（造假团伙）所生成的产品能达到真假难分的地步。



我们有两个网络，G（Generator）和D（Discriminator）。Generator是一个生成图片的网络，它接收一个随机的噪声z，通过这个噪声生成图片，记做G(z)。Discriminator是一个判别网络，判别一张图片是不是“真实的”。它的输入是x，x代表一张图片，输出D（x）代表x为真实图片的概率，如果为1，就代表100%是真实的图片，而输出为0，就代表不可能是真实的图片。

GAN模型优化训练

在训练过程中，生成网络的目标就是尽量生成真实的图片去欺骗判别网络D。而网络D的目标就是尽量把网络G生成的图片和真实的图片分别开来。这样，G和D构成了一个动态的“博弈过程”。这个博弈过程具体是怎么样的呢？

先了解下**纳什均衡**，纳什均衡是指博弈中这样的局面，对于每个参与者来说，只要其他人不改变策略，他就无法改善自己的状况。对应的，对于GAN，情况就是生成模型 G 恢复了训练数据的分布（造出了和真实数据一模一样的样本），判别模型再也判别不出来结果，准确率为 50%，约等于乱猜。这是双方网路都得到利益最大化，不再改变自己的策略，也就是不再更新自己的权重。

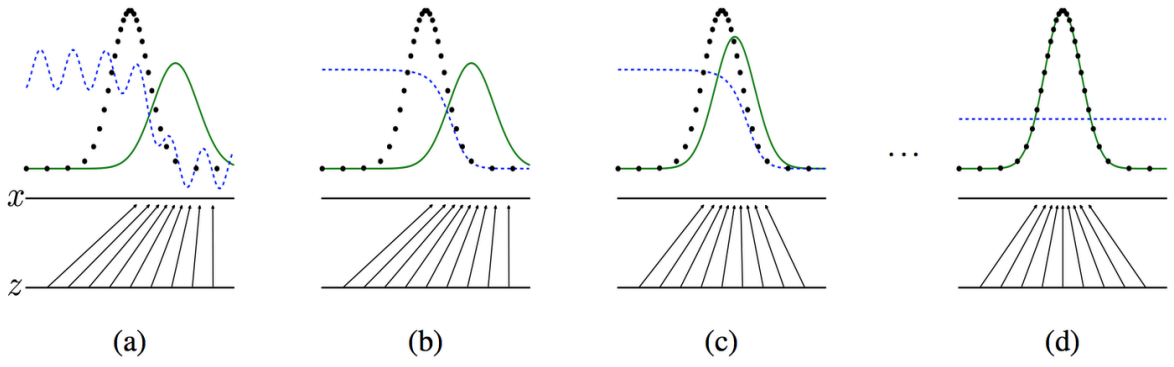
GAN模型的目标函数如下

$$\min_G \max_D V(D,G) = \mathbb{E}_{\boldsymbol{x}\sim p_{\text{data}}(\boldsymbol{x})}[\log D(\boldsymbol{x})] + \mathbb{E}_{\boldsymbol{z}\sim p_{\boldsymbol{z}}(\boldsymbol{z})}[\log(1 - D(G(\boldsymbol{z})))].$$

在这里，训练网络D使得最大概率地分对训练样本的标签（最大化log D(x)和log(1−D(G(z)))），训练网络G最小化log(1 − D(G(z)))，即最大化D的损失。而训练过程中固定一方，更新另一个网络的参数，交替迭代，使得对方的错误最大化，最终，G 能估测出样本数据的分布，也就是生成的样本更加的真实。

然后从式子中解释对抗，我们知道G网络的训练是希望D(G(z))趋近于1，也就是正类，这样G的loss就会最小。而D网络的训练就是一个2分类，目标是分清楚真实数据和生成数据，也就是希望真实数据的D输出趋近于1，而生成数据的输出即D(G(z))趋近于0，或是负类。这里就是体现了对抗的思想。

然后，这样对抗训练之后，效果可能有几个过程，原论文画出的图如下：



黑色的线表示数据x的实际分布，绿色的线表示数据的生成分布，蓝色的线表示生成的数据对应在判别器中的分布效果

对于辨别器，如果得到的是生成图片辨别器应该输出 0，如果是真实的图片应该输出 1，得到误差梯度反向传播来更新参数。对于生成器，首先由生成器生成一张图片，然后输入给判别器判别并的到相应的误差梯度，然后反向传播这些图片梯度成为组成生成器的权重。直观上来说就是：辨别器不得不告诉生成器如何调整从而使它生成的图片变得更加真实。

GAN的优缺点

- GNs是一种以半监督方式训练分类器的方法
- G的参数更新不是直接来自数据样本,而是使用来自D的反向传播
- 理论上,只要是可微分函数都可以用于构建D和G,因为能够与深度神经网络结合做深度生成式模型
- GANs可以比完全明显的信念网络(NADE,PixelRNN,WaveNet等)更快的产生样本,因为它不需要在采样序列生成不同的数据.
- 模型只用到了反向传播,而不需要马尔科夫链
- 相比于变分自编码器, GANs没有引入任何决定性偏置(deterministic bias),变分方法引入决定性偏置,因为他们优化对数似然的下界,而不是似然度本身,这看起来导致了VAEs生成的实例比GANs更模糊.
- 相比非线性ICA(NICE, Real NVE等,),GANs不要求生成器输入的潜在变量有任何特定的维度或者要求生成器是可逆的.
- 相比玻尔兹曼机和GSNs,GANs生成实例的过程只需要模型运行一次,而不是以马尔科夫链的形式迭代很多次.

劣势

- 训练GAN需要达到纳什均衡,有时候可以用梯度下降法做到,有时候做不到.我们还没有找到很好的达到纳什均衡的方法,所以训练GAN相比VAE或者PixelRNN是不稳定的,但我认为在实践中它还是比训练玻尔兹曼机稳定的多.
- 它很难去学习生成离散的数据,就像文本
- 相比玻尔兹曼机,GANs很难根据一个像素值去猜测另外一个像素值,GANs天生就是做一件事的,那就是一次产生所有像素, 你可以用BiGAN来修正这个特性,它能让你像使用玻尔兹曼机一样去使用Gibbs采样来猜测缺失值, 我在伯克利大学的课堂上前二十分钟讲到了这个问题.课程链接,油管视频,请自带梯子~
- 可解释性差,生成模型的分布 Pg(G)没有显式的表达

结构优化

一、DCGAN（Deep Convolutional GAN）

📌 应用场景：

- 无监督图像生成：如人脸、物体图像的随机生成。

📘 算法流程：

1. 网络结构：

DCGAN 是对原始 GAN 的改进，采用了 **卷积结构** 替代全连接结构，更适用于图像数据。

- **生成器 G**：
  - 输入：随机噪声 $z \sim \mathcal{N}(0,1)$，维度通常为 100。
  - 网络结构：
    - 全连接 → reshape 成 feature map
    - 多层 Transposed Convolution（反卷积）+ BN + ReLU
    - 最后一层：Tanh 输出 64×64 或 128×128 图像
  - 输出：生成图像 $\hat{x} = G(z)$
- **判别器 D**：
  - 输入：真实图像或生成图像
  - 网络结构：
    - 多层卷积 Conv + LeakyReLU（不使用 maxpool）
    - 输出：一个 sigmoid 值，表示真假概率
  - 输出：$D(x) \in [0,1]$

---

2. 损失函数（Minimax）：

对抗训练目标：

$$\min_G \max_D V(D,G) = \mathbb{E}_{x \sim p_{\text{data}}}[\log D(x)] + \mathbb{E}_{z \sim p_z}[\log(1 - D(G(z)))]$$

通常采用非饱和的 G 损失变体：

$$\mathcal{L}_G = -\mathbb{E}_{z \sim p_z}[\log D(G(z))]$$

✅ 特点总结：

- 引入卷积结构，提升了训练稳定性和图像质量；
- BN（BatchNorm）大幅缓解模式崩溃（mode collapse）；
- 用 Tanh 输出像素值在 [-1, 1] 区间。

> 💡 DCGAN网络设计中采用了当时对CNN比较流行的改进方案：
>
> （1）将空间池化层用卷积层替代，这种替代只需要将卷积的步长stride设置为大于1的数值。改进的意义是下采样过程不再是固定的抛弃某些位置的像素值，而是可以让网络自己去学习下采样方式。
>
> （2）将全连接层去除，（我前面关于VGG的文章分析了全连接层是如何大幅度增加网络参数数量的）；作者通过实验发现了全局均值池化有助于模型的稳定性但是降低了模型的收敛速度；作者在这里说明了他是通过将生成器输入的噪声reshape成4D的张量，来实现不用全连接而是用卷积的。
>
> （3）采用BN层，BN的全称是Batch Normalization，是一种用于常用于卷积层后面的归一化方法，起到帮助网络的收敛等作用。作者实验中发现对所有的层都使用BN会造成采样的震荡（我也不理解什么是采样的震荡，我猜是生成图像趋于同样的模式或者生成图像质量忽高忽低）和网络不稳定。
>
> DCGAN的改进不包含严格数学证明，主要是理论分析和实验验证，其对生成器的判别的修改核心如下：
>
> （1）使用指定步长的卷积层代替池化层
>
> （2）生成器和判别器中都使用BN
>
> （3）移除全连接层
>
> （4）生成器除去输出层采用Tanh外，全部使用ReLU作为激活函数
>
> （5）判别器所有层都使用LeakyReLU作为激活函数

二、Pix2Pix（Conditional GAN for Image-to-Image Translation）

📌 应用场景：

- 图像到图像翻译任务，如：
  - 彩色图 ↔ 黑白图
  - 边缘图 → 实物图

- 卫星图 → 地图

📘 算法流程：

1. 网络结构：

Pix2Pix 是一种 **条件 GAN（cGAN）**，输入是图像 x，输出是目标图像 y。

- **生成器 G**：
  - 输入：条件图像 x
  - 网络结构：U-Net 结构（编码 + 解码 + skip connection）
    - 编码部分：Conv → BN → LeakyReLU
    - 解码部分：TransConv → BN → ReLU
    - 每一层加入 skip connection：连接对称的 encoder & decoder 层
  - 输出：生成图像 $\hat{y} = G(x)$
- **判别器 D**：
  - 输入：(x, y) 或 $(x, G(x))$
  - 网络结构：PatchGAN 判别器（每个 $N \times N$ patch 判别真假）
  - 输出：每个 patch 的真假概率（矩阵形式）

---

2. 损失函数：

总损失包含两个部分：

- 对抗损失（cGAN）：

$$\mathcal{L}_{cGAN}(G, D) = \mathbb{E}_{x,y}[\log D(x, y)] + \mathbb{E}_{x}[\log(1 - D(x, G(x)))]$$

- L1 重建损失（保持结构和色彩）：

$$\mathcal{L}_{L1}(G) = \mathbb{E}_{x,y}[\|y - G(x)\|_1]$$

- 最终目标函数：

$$G^* = \arg\min_G \max_D \mathcal{L}_{cGAN}(G, D) + \lambda \mathcal{L}_{L1}(G)$$

✅ 特点总结：

- U-Net 架构利于保留低层图像细节；
- PatchGAN 判别器提高了局部真实感；

- L1 损失抑制过度发散，生成更加稳定和细腻。

主要应用

图像生成

目标： 从随机噪声生成真实感图像（如 DCGAN）

🧠 网络结构

- **Generator $G(z)$**：噪声 $z \sim \mathcal{N}(0, 1)$ → 图像（使用转置卷积结构）
- **Discriminator $D(x)$**：输入图像 → 输出概率 $D(x) \in [0, 1]$ 表示“真实”的概率

⚙️ 算法流程

1. **输入准备**：
   - 真实图像 $x \sim p_{data}(x)$
   - 随机噪声 $z \sim \mathcal{N}(0,1)$
2. **判别器更新**（判别真假图像）：

$$\mathcal{L}_D = -\mathbb{E}_{x \sim p_{data}}[\log D(x)] - \mathbb{E}_{z \sim p_z}[\log(1 - D(G(z)))]$$

3. **生成器更新**（欺骗判别器）：

$$\mathcal{L}_G = -\mathbb{E}_{z \sim p_z}[\log D(G(z))]$$

4. **交替训练 G 和 D，直到图像质量满意。**

图像转换

目标： 将输入图像（如线稿、边缘图）转化为输出图像（如真实照片）【如 Pix2Pix】

🧠 网络结构

- **Generator $G(x)$**：输入图像 → 输出转换图像（U-Net）
- **Discriminator $D(x, y)$**：判断配对图像 (x, y) 是真实还是伪造（PatchGAN）

⚙️ 算法流程

1. **输入准备**：
   - 输入图像 x，目标图像 y
   - 生成图像 $\hat{y} = G(x)$
2. **判别器损失（条件GAN）**：

$$\mathcal{L}_{cGAN}(G, D) = \mathbb{E}_{x,y}[\log D(x, y)] + \mathbb{E}_x[\log(1 - D(x, G(x)))]$$

3. **生成器损失（对抗 + L1）**：

$$\mathcal{L}_G = \mathcal{L}_{cGAN}(G, D) + \lambda \|y - G(x)\|_1$$

4. **训练过程同样交替优化 G 和 D。**

图像超分辨率重建

目标： 将低分辨率图像重建为高分辨率图像【如 SRGAN】

🧠 网络结构

- **Generator $G(x_{LR})$**：输入低分辨率图像，输出高分辨率图像
- **Discriminator $D(x_{HR})$**：区分真假高分辨率图像
- 可加 **感知损失（Perceptual Loss）** 提升细节重建

⚙️ 算法流程

1. **输入准备**：
   - x_{LR}：低分辨率图像
   - x_{HR}：对应高分辨率图像
   - $\hat{x}_{HR} = G(x_{LR})$
2. **判别器损失**：

$$\mathcal{L}_D = -\log D(x_{HR}) - \log(1 - D(\hat{x}_{HR}))$$

3. **生成器损失**：

$$\mathcal{L}_G = \alpha \cdot \|x_{HR} - \hat{x}_{HR}\|_2^2 + \beta \cdot \text{PerceptualLoss} + \gamma \cdot \log(1 - D(\hat{x}_{HR}))$$

4. **感知损失**（可选）：
   提取 VGG 特征图，比较生成图与真实图在感知空间中的差异。

音乐生成

目标： 从随机噪声或编码表示生成 MIDI 音乐【如 MuseGAN】

🧠 网络结构（MuseGAN）

- **Generator**：输入噪声 z，输出多轨音乐表示（如多个乐器轨道）
- **Discriminator**：输入真实或生成的多轨音乐，判断其是否真实

⚙️ 算法流程

1. **输入准备**：
   - 噪声向量 z
   - 音乐数据为多轨 MIDI（离散表示为张量 $T \in \mathbb{R}^{T \times P \times C}$）
2. **生成器生成 $G(z)$**：
   输出为多通道轨道，表示不同乐器的钢琴卷帘图（piano roll）
3. **判别器损失**：
   区分真实和生成的音乐序列：

$$\mathcal{L}_D = -\mathbb{E}_{x \sim p_{data}}[\log D(x)] - \mathbb{E}_{z \sim p_z}[\log(1 - D(G(z)))]$$

4. **生成器损失**：

$$\mathcal{L}_G = -\mathbb{E}_{z \sim p_z}[\log D(G(z))]$$

5. 可添加节奏一致性、和声协调等规则损失。

异常检测

目标： 用 GAN 重建正常图像，判断测试图像与重建差异是否异常

🧠 网络结构（如 AnoGAN）

- **训练阶段**：用正常图像训练 GAN（G 和 D）
- **测试阶段**：寻找最匹配的潜在变量 $z^$，*使 $G(z^)$* 最接近输入图像 x，通过重建误差判断异常

⚙️ 算法流程

A. 训练阶段（仅正常样本）：

1. 训练 GAN（G, D）生成正常图像分布。

B. 测试阶段（异常检测）：

1. **优化潜在向量 z^***：
   对于测试图像 x，找出最优 z^*：

$$z^* = \arg\min_z \mathcal{L}_{residual}(x, G(z)) + \lambda \cdot \mathcal{L}_{feature}(x, G(z))$$

   - $\mathcal{L}_{residual} = |x - G(z)|$
   - $\mathcal{L}_{feature} = |D^{feat}(x) - D^{feat}(G(z))|$

2. **异常分数计算**：

$$A(x) = \mathcal{L}_{residual}(x, G(z^*)) + \lambda \cdot \mathcal{L}_{feature}(x, G(z^*))$$

   - 分数高 → 表示无法重建 → 异常