北京理工大学
BEIJING INSTITUTE OF TECHNOLOGY

本科生《机器学习初步》课程实践报告

题　　目：Conditional GAN

学　　院：徐特立学院

专业名称：计算机科学与技术

姓　　名：陈照欣-1120191086

一、 摘要

GAN 作为一种训练生成模型方法被引入，一个重要优势就是不需要计算马尔科夫链，只需要通过反向传播算法就可以获得梯度，在学习过程中也不需要进行推断，一系列的因素和相互作用就可以被轻易地加入到模型当中，但是我们无法控制 GAN 的输出。Conditional GAN 的提出使得我们可以通过输入一个文字条件，生成文字对应的结果。本实验中我基于 colab 平台使用 MNIST 的数据集完成了手写数字识别功能。

二、 理论基础

传统的神经网络中，倘若用给定的文字输入想要输出一张对应的图片，生成的图片极有可能是数据集中多个图片的组合。看似都有符合的元素，但是效果很差。因此 GAN 的判断真假的优点得以体现。

作为 GAN 的扩展，Conditional-GAN 同样由一个 generator 和一个 discriminator 组成，区别在于 generator 和 discriminator 的均多了一个向量 y 作为 condition。Discriminator 除了需要判别图片是否为 generator 生成的图片的同时，还需要判断图片与 y 是否一致。

目标函数为：

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

证明 1：目标函数有最优解

$$\begin{aligned} V(D,G) &= E_{x\sim P_{Data}(x)}[logD(x|y)] + E_{z\sim P_z(z)}[log(1 - D(G(z|y)))] \\ &= \int_x p_{data}(x)log(D(x|y)) + p_g(x)log(1 - D(x|y))dx \end{aligned}$$

$$\text{对}V(D,G)\text{求导得：}\ p_{data}(x)log(D(x|y)) + p_g(x)log(1 - D(x|y)) = 0$$

$$\text{得}D(x|y) = \frac{p_{data}}{p_{data} + p_g}$$

此时 D(x|y)=0.5，说明鉴别模型已经完全分不清真实数据和 GAN 生成的数据了，

此时就是得到了最优生成模型了

KL 散度永远大于等于 0，可以知道目标函数最终最优值为-log4。

三、 Conditional Adversarial Nets 结构以及训练

生成网络的结构如图 1，输入相对于 GAN 增加了一个条件。

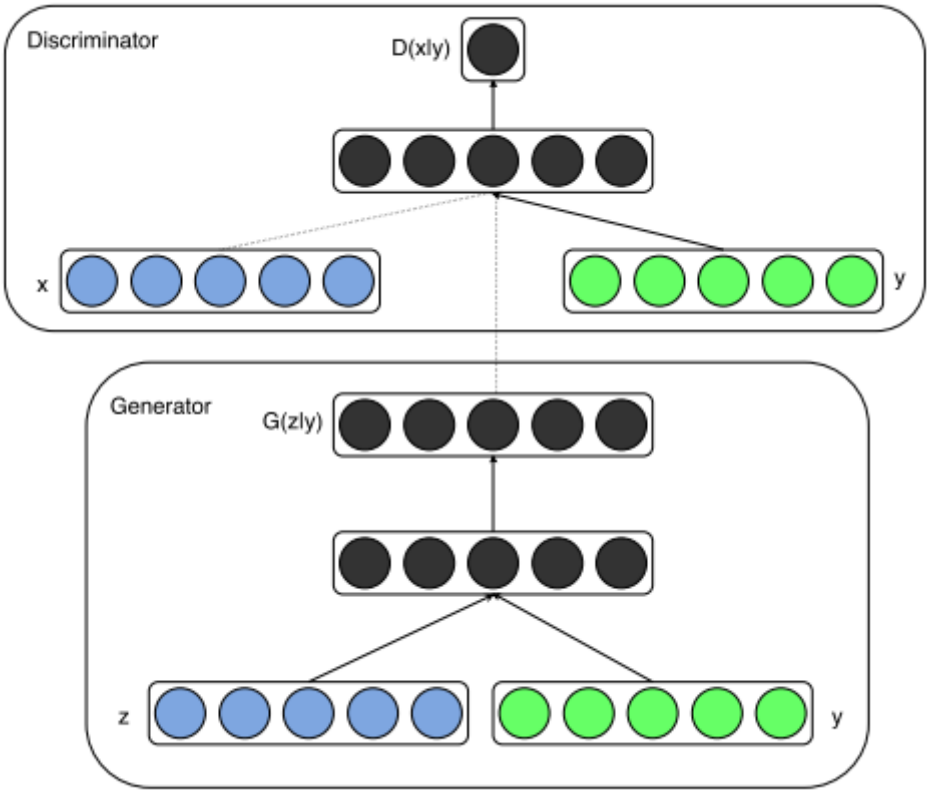


Figure 1

对于判别网络，输入除了生成的图片，还有与生成网络输入同样的条件，输出是一个综合打分。若接近实际而且与条件符合则为 1 分，若输出图片质量低或图片与条件不符则为 0 分。判别网络的两种形式如图 2 所示，上面一种是常规形式，及将两种因素综合成为一个分数，而下面这种形式也有一些人在研究，输出两个分数，分别表示真实程度和图片与条件的相符程度。

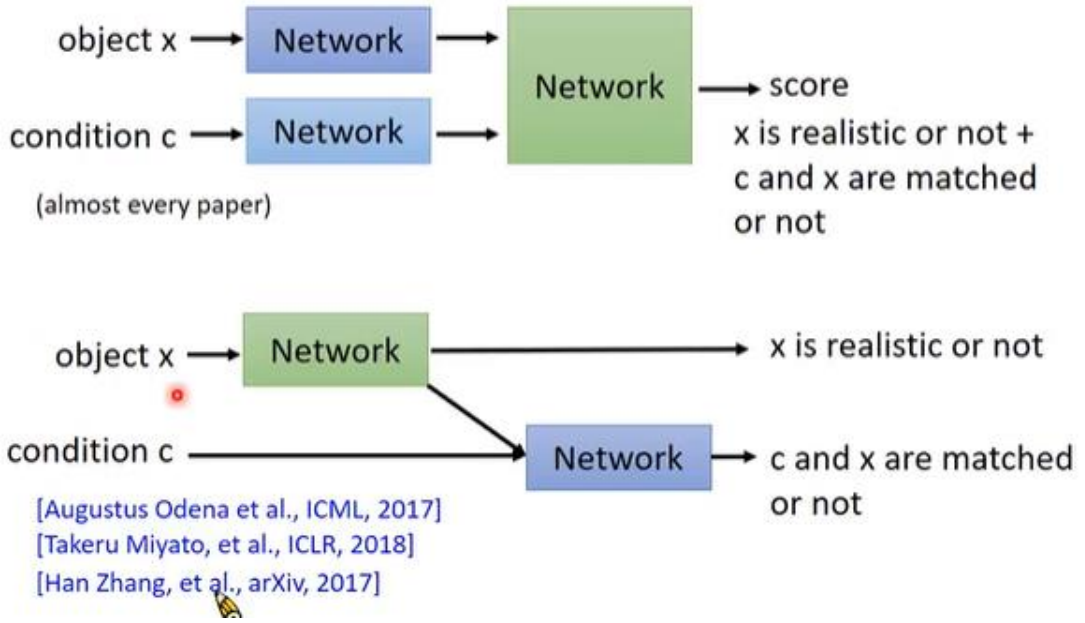


Figure 2

同一般形式的 GAN 类似，也是先训练判别网络，再训练生成网络，然后再训练判别网络，两个网络交替训练。只是训练判别网络的样本稍有不同，训练判别网络的时候需要这三种样本，分别是：（1）条件和与条件相符的真实图片，期望输出为 1；（2）条件和与条件不符的真实图片，期望输出为 0；（3）条件和生成网络生成的输出，期望输出为 0。如图 3.

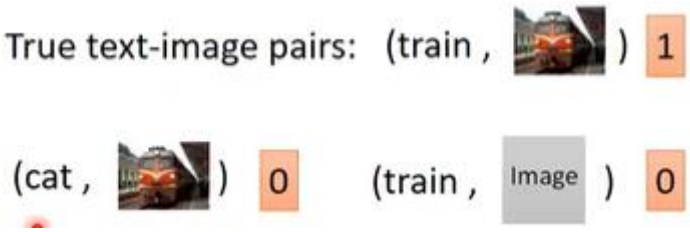


Figure 3

四、 实验结果分析

作者以 one-hot vectors 形式的类别标签作为条件在 MNIST 数据集上训练了一个对抗网络。在生成网络中，100 维的噪声先验分布是从单位超方体的均匀分布采样得到的。z 和 y 都是映射到带有 RELU 激活函数的 hidden layers，隐藏层节点数分别为 200 和 1000，然后二者的输出相结合形成一个节点数为 1200 的

带有 RELU 激活函数的 hidden layer，最后是一个 sigmoid unit hidden layer 作为输出，生成 784 维的 MNIST samples。

Model	MNIST
DBN [1]	138 ± 2
Stacked CAE [1]	121 ± 1.6
Deep GSN [2]	214 ± 1.1
Adversarial nets	225 ± 2
Conditional adversarial nets	132 ± 1.8

对于 MNIST 的 test data 的 Gaussian Parzen window 对数似然估计

生成结果

五、 代码复现

我基于 colab 平台使用 MNIST 数据集对实验进行了复现。

1. 参数设置

在实验中，我设定训练轮数为 40 轮，Batch_Size 为 938，损失函数使用 MSELoss 定义。

Generator 的结构声明如下：

```
    self.model = nn.Sequential(
    *block(opt.latent_dim + opt.n_classes, 128, normalize=False),
    *block(128, 256),
```

```
    *block(256, 512),
    *block(512, 1024),
    nn.Linear(1024, int(np.prod(img_shape))),
    nn.Tanh()
)
```

Discriminator 的结构声明如下：

```
self.model = nn.Sequential(
    nn.Linear(opt.n_classes + int(np.prod(img_shape)), 512),
    nn.LeakyReLU(0.2, inplace=True),
    nn.Linear(512, 512),
    nn.Dropout(0.4),
    nn.LeakyReLU(0.2, inplace=True),
    nn.Linear(512, 512),
    nn.Dropout(0.4),
    nn.LeakyReLU(0.2, inplace=True),
    nn.Linear(512, 1),
)
```

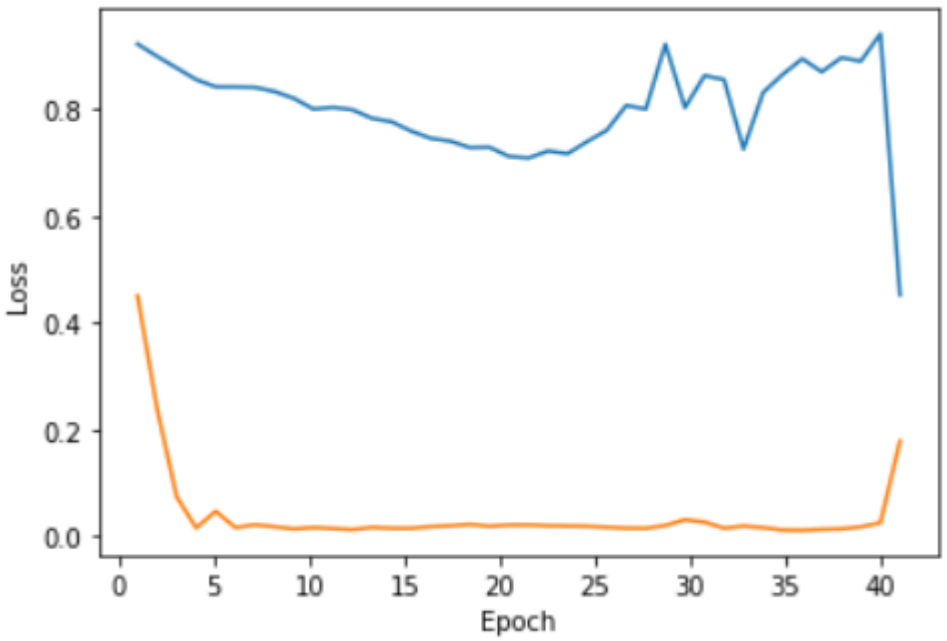
2. 实验步骤：

Step 1：导入工具包以及完成各种声明

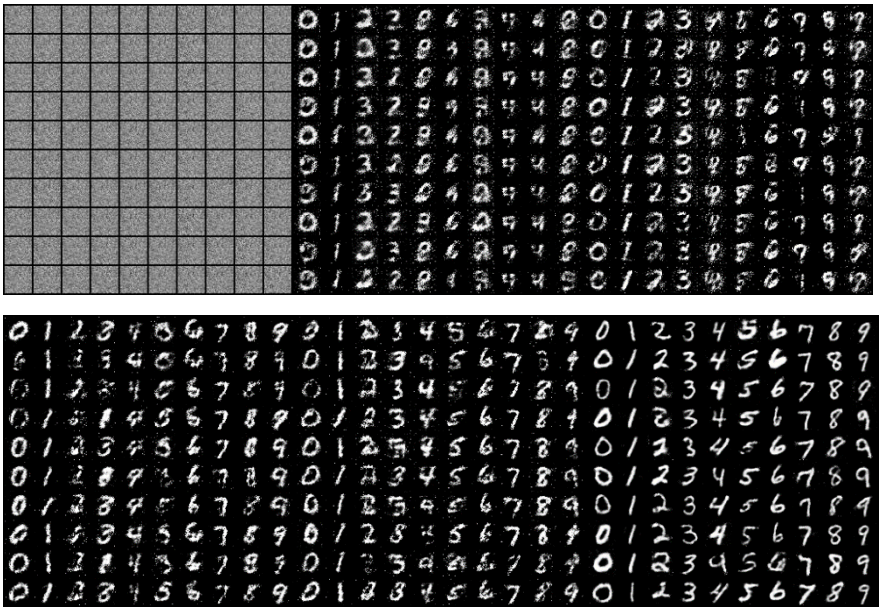
Step 2：训练生成器。先进行梯度归零，接着生成随机噪声 z，单个样本的随机噪声为向量。生成器根据随机噪声 z 和随机条件生成样本数据，送入鉴别器中，根据随机图片和随机条件计算获得准确率，并计算生成器的损失大小。对损失进行反向传播计算，计算梯度。

Step 3：训练鉴别器，开始进行鉴别器的梯度置零，计算真实样本的真实率和误差，再计算生成样本的错误率和误差，总的误差取平均，进行反向传播计算，计算梯度，更新鉴别器的参数。

3. 实验结果



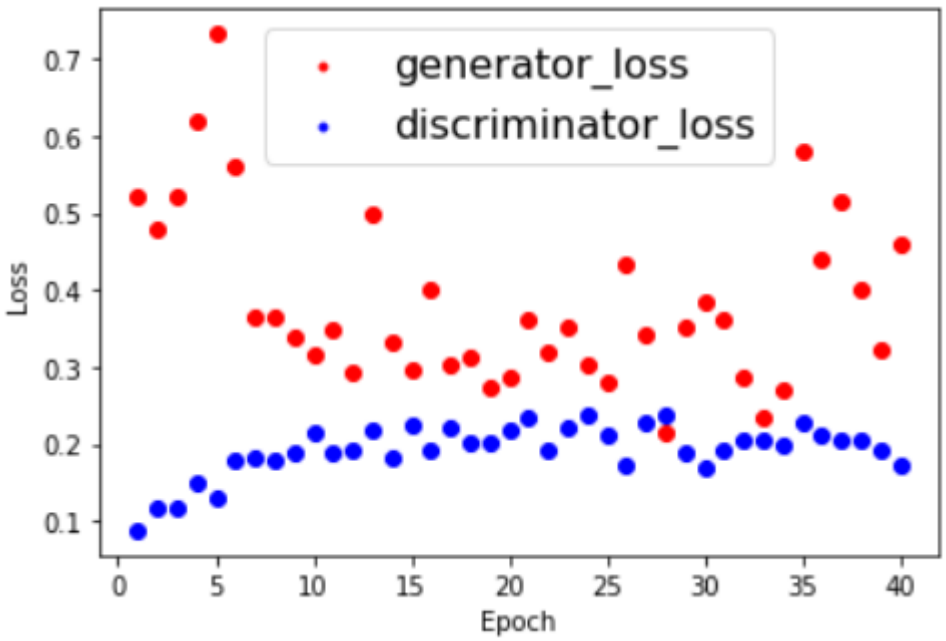
其中蓝色代表 Generator_Loss，黄色代表 Discriminator_Loss。可以看出前期生成器的效果并不理想，与之对应的是分辨器的损失函数一直很低。但是当训练轮数达到一定程度时，两个损失值都会出现较大变化。



上图为不同阶段 generator 生成的图。

4. 猜想与验证

根据上图的趋势，我本以为倘若扩大训练轮数，两条线最终会稳定在统一水平线上。于是我尝试将训练轮数扩大至 40，但是得出的结果却和猜想不太一致：



事实证明，discriminator_loss 会趋于稳定。但是 generator_loss 在短暂下降后又进入了不稳定的状态。

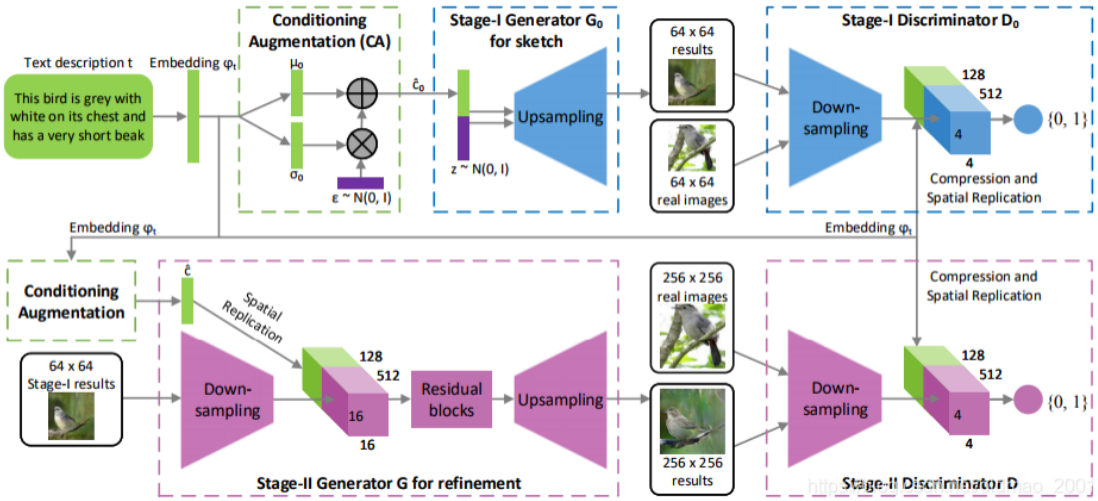
由此我分析为，生成器在多轮之后会遇到一个进化版本的判别器。这个判别器发生的变化可能很小，但也可能很大。有可能只是一轮训练，使得生成器的大部分特征都被判别器捕获，导致损失函数突然增大。更进一步推出 GAN 中的收敛条件在 CGAN 中不适用。

六、 改进与创新

1. Stack GAN

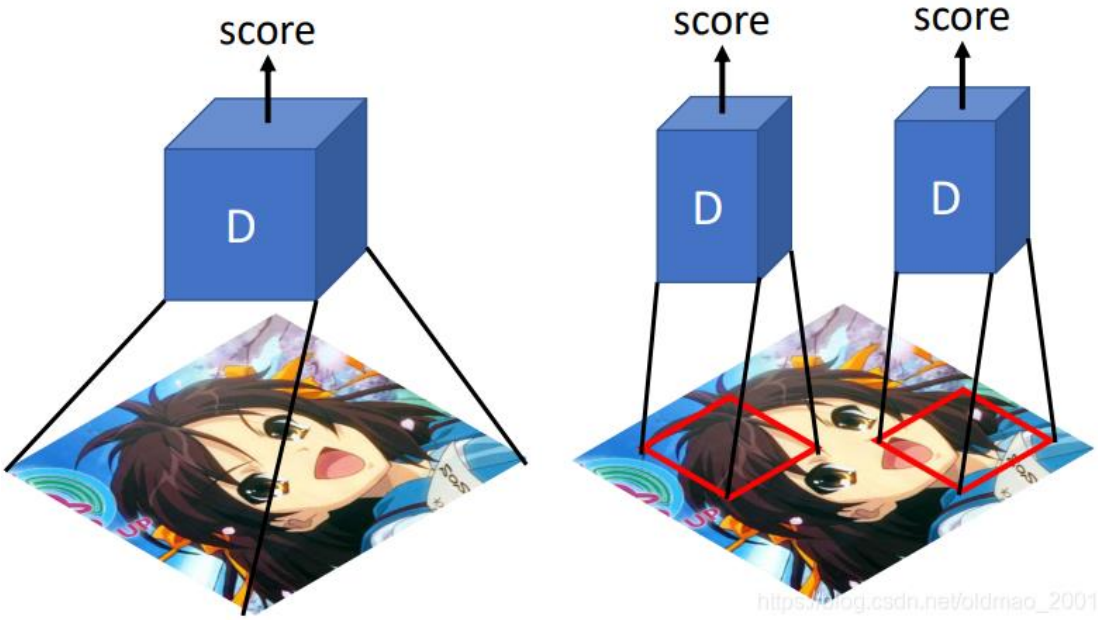
先生成小图，再生成大图。

大致流程：先有一个 Generator1 根据文字描述生成一个 64×64 的小图片，然后经过一个 Discriminator1，判断小图片和文字是否匹配，如果匹配，进入 Generator2，融合小图片得到较大图片，然后经过一个 Discriminator2，判断大图片和文字是否匹配。



2. Patch GAN

当用一个 Discriminator 来进行评估整个大张的图片的时候会有很多问题，例如容易过拟合，以及训练时间长。因此可以用多个 Discriminator 来进行评估。每个 Discriminator 检查的区域的大小是超参数。同时区域不能太小那么整个图片就会糊掉。



七、 参考文献

［1］Mirza, Mehdi, and Simon Osindero. “Conditional Generative Adversarial Nets.” *ArXiv.org*, 2014, arxiv.org/abs/1411.1784.

[2] Goodfellow, Ian J, et al. “Generative Adversarial Networks.” *ArXiv.org*, 2014, arxiv.org/abs/1406.2661.

[3] Isola, Phillip, et al. *Image-To-Image Translation with Conditional Adversarial Networks*.

[4]Zhang, Han, et al. “StackGAN: Text to Photo-Realistic Image Synthesis with Stacked Generative Adversarial Networks.” *ArXiv.org*, 2016, arxiv.org/abs/1612.03242.