

**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**

1. 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。

1. 理论基础

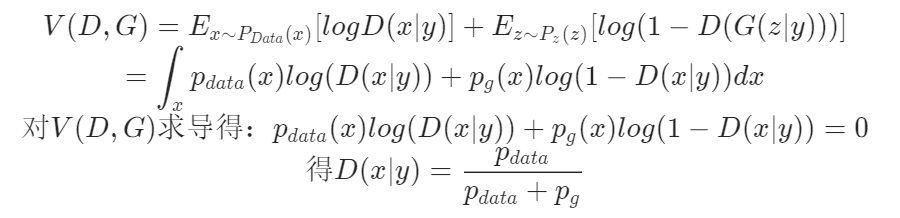
传统的神经网络中，倘若用给定的文字输入想要输出一张对应的图片，生成的图片极有可能是数据集中多个图片的组合。看似都有符合的元素，但是效果很差。因此GAN的判断真假的优点得以体现。

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：



证明1：目标函数有最优解



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。

1. 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。

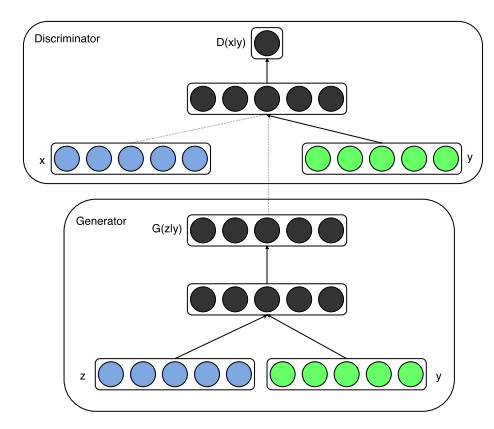


Figure 1

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。

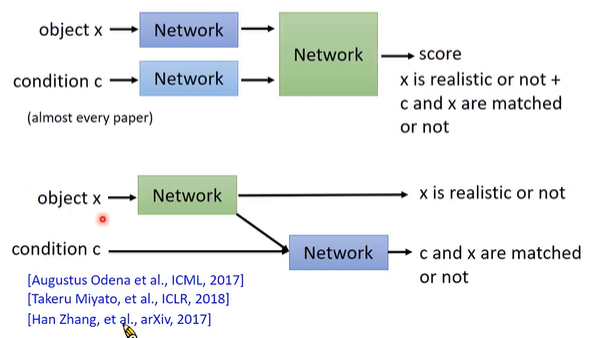


Figure 2

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.

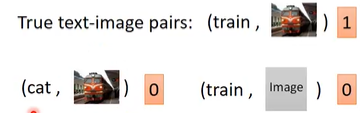
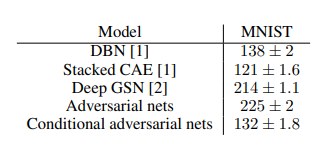


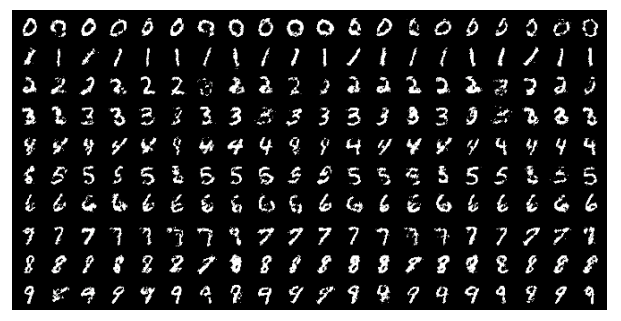
Figure 3

1. 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。



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



生成结果

1. 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),  
)

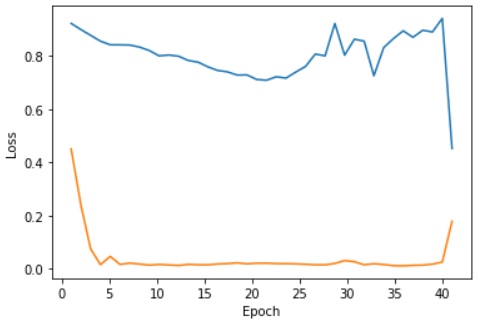
1. 实验步骤：

Step 1：导入工具包以及完成各种声明

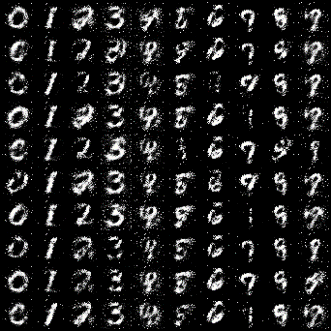
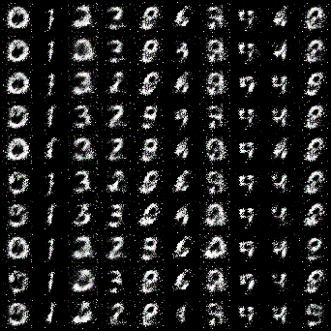
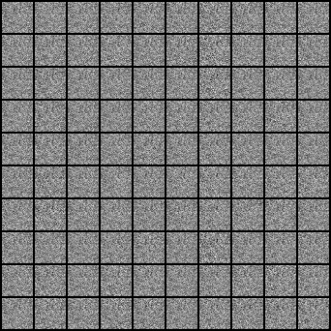
Step 2：训练生成器。先进行梯度归零，接着生成随机噪声z，单个样本的随机噪声为向量。生成器根据随机噪声z和随机条件生成样本数据，送入鉴别器中，根据随机图片和随机条件计算获得准确率，并计算生成器的损失大小。对损失进行反向传播计算，计算梯度。

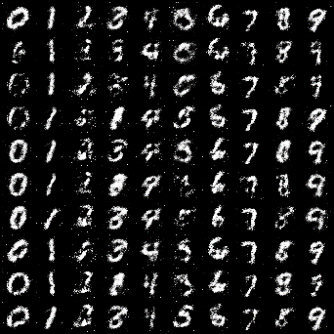
Step 3：训练鉴别器，开始进行鉴别器的梯度置零，计算真实样本的真实率和误差，再计算生成样本的错误率和误差，总的误差取平均，进行反向传播计算，计算梯度，更新鉴别器的参数。

1. 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。

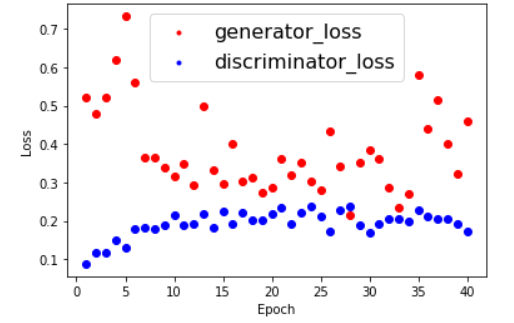




上图为不同阶段generator生成的图。

1. 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：



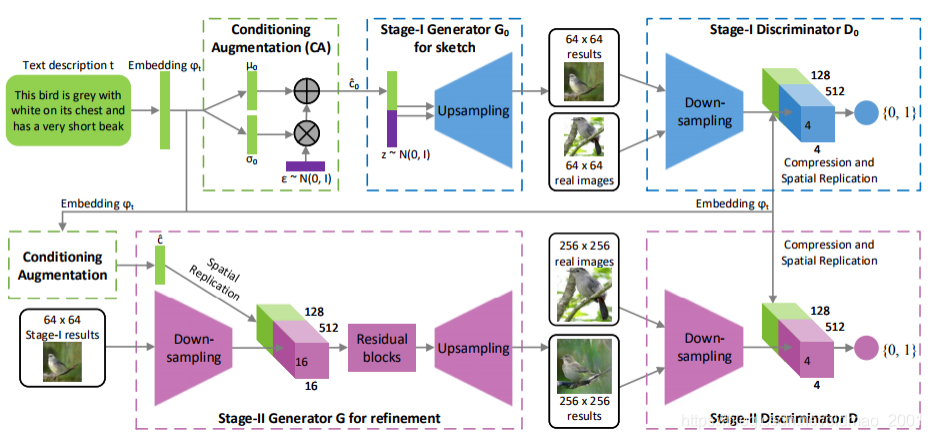
事实证明，discriminator\_loss会趋于稳定。但是generator\_loss在短暂下降后又进入了不稳定的状态。

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。

1. 改进与创新
2. Stack GAN

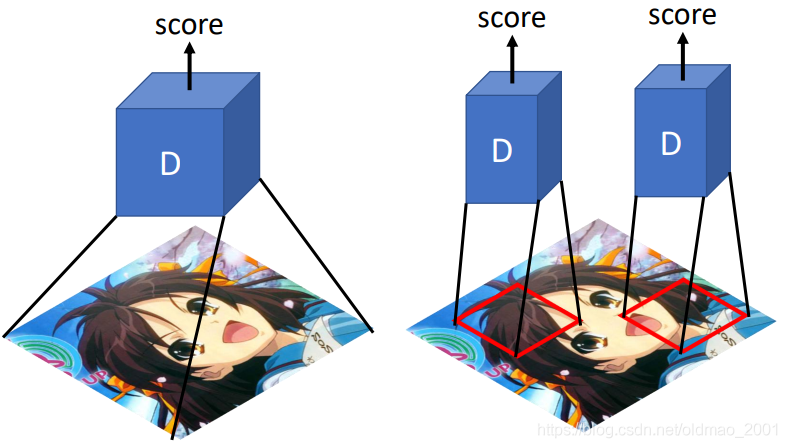
先生成小图，再生成大图。

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。



1. 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。



1. 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.