在之前报告中写过一个非常直观的GAN程序，仅仅是在二维环境下生成一条曲线尽可能的和真实曲线相近，但是也是麻雀虽小五脏俱全，代码的地址为<https://github.com/TwistedW/Tensorflow-noting/blob/master/tensorflow_gan.py>。 程序结果如图1所示。

图1. 简易GAN实现图

由图一所示蓝线和橙线之间为真实二维曲线的分布范围，绿色的线为由噪声生成出来的曲线。这条绿色的线就是生成器生成数据，通过判别器对生成数据和真实数据的判别后更新损失函数从而达到生成器与判别器之间的博弈实现了GAN的思想。值得说的是在GAN的理想下，由之前报告中的公式推导，在判别器达到理想下的判别器的loss为公式1所示。

 (1)

在生成器达到理想状态下，在固定生成器时， 这和试验中的结果是一致的。但是我们之前在WGAN中证明过当判别器达到理想状态下会导致生成器梯度消失的问题，这里为什么没有出现呢。这个原因是我们讨论WGAN的生成器的JS散度为常数log2时前提是真实数据分布和生成数据分布上不可能重叠，而不可重叠的前提是真实数据是呈高维分布，而我们的例子是真实数据仅仅是二维的曲线所以不存在高维分布，就满足不了前面的假设，这个也间接证实了GAN原文中公式推导是没有问题的。这个简单的代码中由生成器，判别器，真实数据单元和训练单元构成，是对GAN大方向把握的一个入门级小程序。

当然在实际数据集的训练上不可能只是二维数据的，往往是维度很高的图片数据集，比如mnist数据集，cifar10数据集，celebA数据集，svhn数据集等等。所以一个好的GAN模型框架是深入了解的基础，我之前github上找到一个很好的框架我也更新到自己主页上了<https://github.com/TwistedW/GANs>，这个框架是建立在mnist数据集上面的，我本周也是将部分GAN模型移植训练了cifar10数据集。我对这个框架已经有了一定的了解，已经在github上对部分GAN模型做了详细的代码注释。需要说明的是我的程序全部是基于tensorflow写的，现在这个大环境下tensorflow和torch已经占据了机器学习的江山了。

GAN模型的搭建我总结了一下，我将它分为5块：

1. 路径参数配置
2. 创建变量及图形化
3. 模型搭建
4. 训练优化
5. 变量，训练结果可视化

路径配置虽然看上去不重要但是往往这一块是很容易出现问题的，一旦模型的路径出现混乱或是路径文件夹分配不合理将导致一系列问题的冲突的发生，一个良好的路径分配还是很关键的。Python中提供了argparse模块，这个模块就是用来定义自己的参数配置，当然也可以指定路径的配置。另一个比较重要的模块式os模块，它是专门用来管理路径的，可以读入路径也可以创建路径具体的应用我就不展开写了，如果需要我想我会单独做一份文档介绍os和argparse的使用。

创建变量和图形化这块是不允许出错的，因为一旦变量给定，数据的张量大小就已经定了，所以懂得利用变量是机器学习程序搭建的一个重中之重。在tensorflow中的用于训练的变量一般用Placeholder将它保存住，等到喂数据的时候再feed就可以了。变量的可视化是程序编写的一个好习惯，可以用来检查变量张量大小从而判断网络结构的正确性，这一块的实现前提是要对定义的变量的名字赋予要合理。

创建模型是一个程序设计中最重要的地方了，GAN的模型设计无非是定义好生成器、判别器、损失函数的实现。其中损失函数的实现又是GAN模型最为关键的地方，tensorflow中存在很多好用的API可以帮助我们大大减小损失函数实现起来的难度，以原始GAN模型损失函数分析，利用tf.reduce\_mean和tf.nn.sigmoid\_cross\_entropy就可以实现生成器的损失函数和判别器的损失函数实现，仅仅需要几行代码就可以实现。

训练模型也是一个重要环节，这一块需要强调的是由于tensorflow下的tensorboard比较好用所以最好在模型设计上考虑到对模型的log上做一定的保存可以很好的在tensorboard中观察。不过就我目前调参的经验而谈还用不到这么复杂的东西，不过一些很厉害的团队还是可以把tensorboard玩的很好的。训练模型上batchsize的选择也是一个值得注意的，不过根据很多程序上的使用来看基本上都是使用64作为一个batch，如果电脑的配置特别好也可以考虑更高，不过实验室的电脑配置上中上成吧，所以我的batch定为64是比较好的，遇到大型数据集可能还要向下调。Tensorflow自带的API的优化器还是比较全面的，比如GradientDescentOptimizer 、AdagradOptimizer 、AdagradDAOptimizer 、MomentumOptimizer 、AdamOptimizer 、FtrlOptimizer 、RMSPropOptimizer本模型用的优化器为AdamOptimizer。

可视化训练图像的保存是检验模型好还的一个关键步骤，在保存图片上的一些小细节很重要的，比如图像的排版分布，输出图像的像素以及加上标签后按照标签生成图像的排布都是十分的重要，一个小的错误就会导致输出上的不理想甚至会让实验者错误领会生成信息的实验结果正确性。

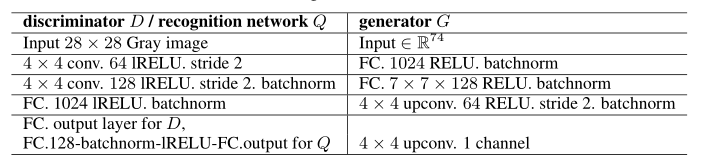
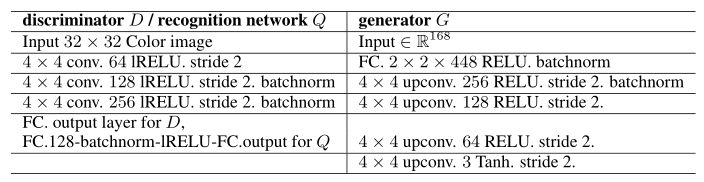
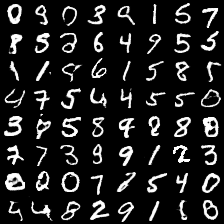
本周对GAN，CGAN和VAE的代码实现了在mnist和cifar10数据集上的测试，实验效果也在预想范围内。对于mnist训练网络我采用图2的网络结构，对于cifar10训练网络我采用图3的网络结构，这两个结构来自于infoGAN论文但是在实际训练过程中我对cifar10的网络结构作了一些调整得到的实验结果比直接拿来用的效果要好很多。

图2. 训练mnist网络结构

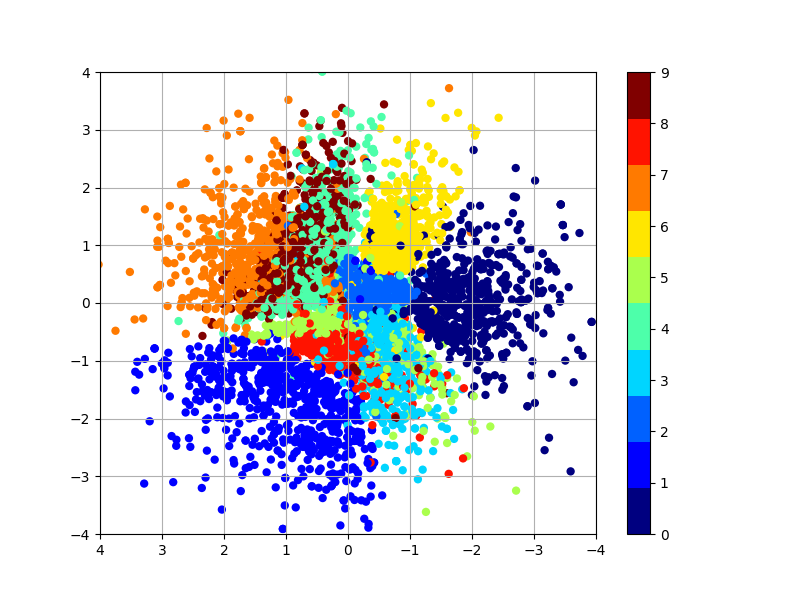
图3. 训练cifar10网络结构

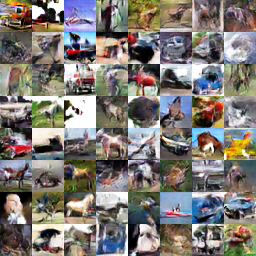
实验结果我将按顺序展示一下：

GAN在mnist上epoch25下生成的结果：

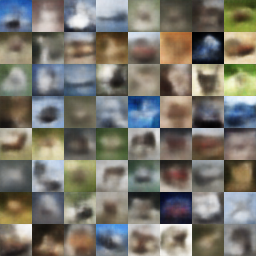
CGAN在mnist上epoch25下生成的全类别排序结果：

VAE在mnist上epoch25下生成的结果：

VAE在mnist上epoch25下分类结果：

GAN在cifar10上epoch40下生成的结果：

CGAN在cifar10上epoch60下生成的全分类结果：

VAE在cifar10上epoch40下生成的结果：

由实验结果可以看出GAN在生成图像上的能力还是很不错的，但是VAE在分类上的效果虽然不错但是生成的图片比较模糊这也是他们两个的明显差异。

小结：本周对GAN部分模型的代码进行了整理，对整体的程序设计做了详细的研读也在实验上加以验证，实验数据上的结果完全在预想范围内，不过用CGAN训练cifar10上出现了很大的问题但是还是通过调整网络结构解决了。

下一周将对剩余GAN模型的代码进行实现，主要针对WGAN和infoGAN以及ACGAN上实现模型的建立和测试。celebA数据集太大又不太好训练，我将选择性的对该数据集作测试和调参。