### 验证码识别项目报告

郭晓波 1500016208 姜聪 1500016349

### 1.题目分析

本题是验证码识别问题，一般来说有较多方法，因为其与mnist问题比较相近，所以我选择采用cnn的方式机器学习。选择的训练集和测验集均为自己生成的随机验证码。为了避免切割字符带来的问题，在训练过程中不对字符进行切割而是对整个验证码进行识别。

由于对于机器学习并不是十分熟悉，所以采用python3调用封装好的tensorflow解决问题，因为其有相对较多的封装好的函数，调用起来相对方便。

除了tensorflow之外，python3需要的包还包括了numpy、captcha、random、PIL、matplotlib、string、random库。

### 2.生成验证码图片

首先需要获得训练用的验证码图片，我们引入了captcha库，用于生成图片验证码。首先从python的字符集中随机选择字符生成默认长度为6的字符串，然后使用captcha生成图片，，图片大小为60\*160.captcha生成的图片样例如下

、

验证码的内容包括了大小写字母和阿拉伯数字，每个字符均为彩色的，可能存在旋转、扭曲和粘连。每个字符的位置不是固定的也不是均匀分布的。除了字符外还包括了噪点和干扰线，用于干扰，字符本身比干扰线要粗。具体代码在create\_captcha.py文件中

### 3.模型构建

首先做了一些预处理，由于生成的验证码图片是彩色，我们对其进行灰度处理，有彩色的三通道降为了灰度的一通道，因为我们认为对于我们生成的验证码而言，色彩并不是十分重要的因素。曾经考虑过是否需要将图像补为64\*256但是事实上没有什么明显的区别。

生成训练batch，训练步长为128，相比于64的训练步长128的训练步长。

为了方便运算，定义一个text2vec函数将生成的验证码字符转为向量。

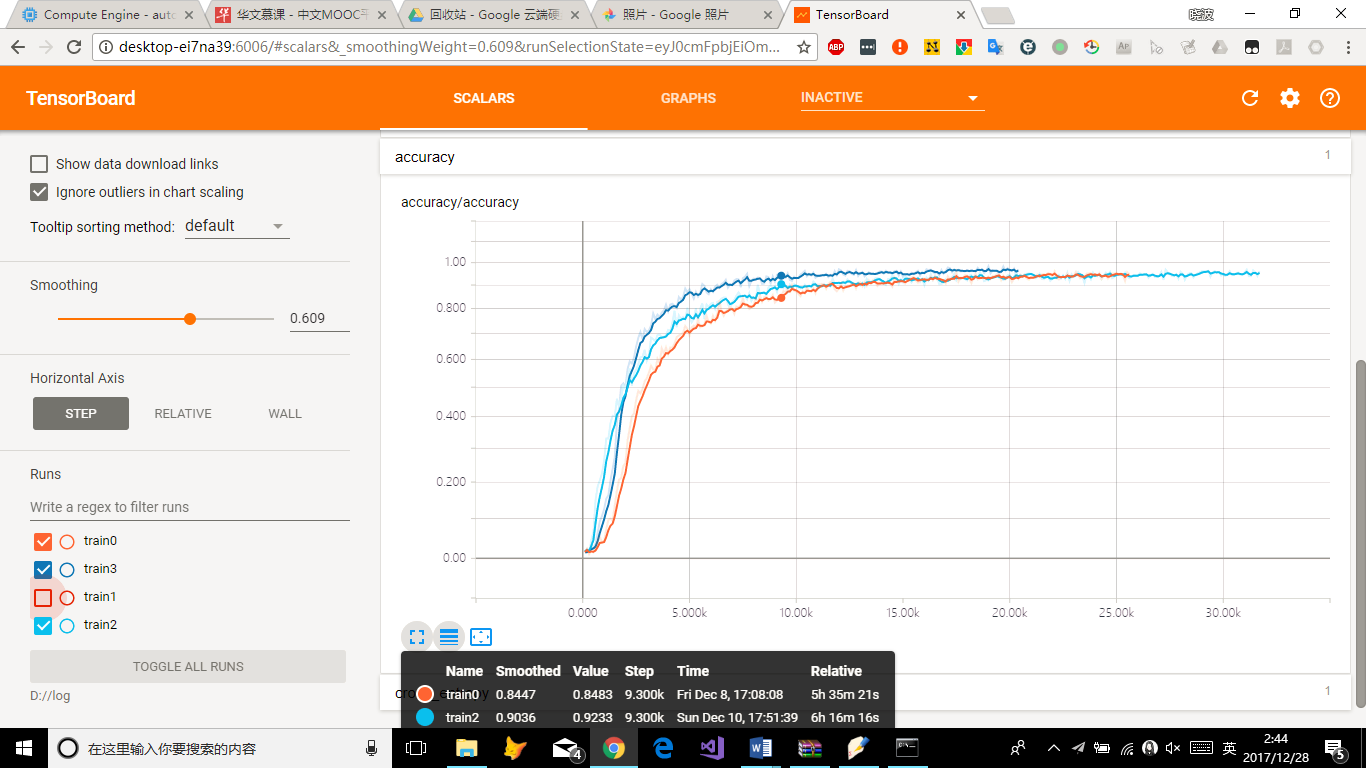
卷积神经网络结构上，我们使用3个卷积层和1个全连接层的结构。首先是初始化需要重复使用的函数。我们给权重制造一些随机的噪声来打破完全对称，同时也给偏置增加一些小的正值（0.1）来避免死亡节点。同时我们也尝试过采用随机的偏置但是发现在loss的下降和对抗过拟合两个问题上都没有太好的效果，所以放弃。

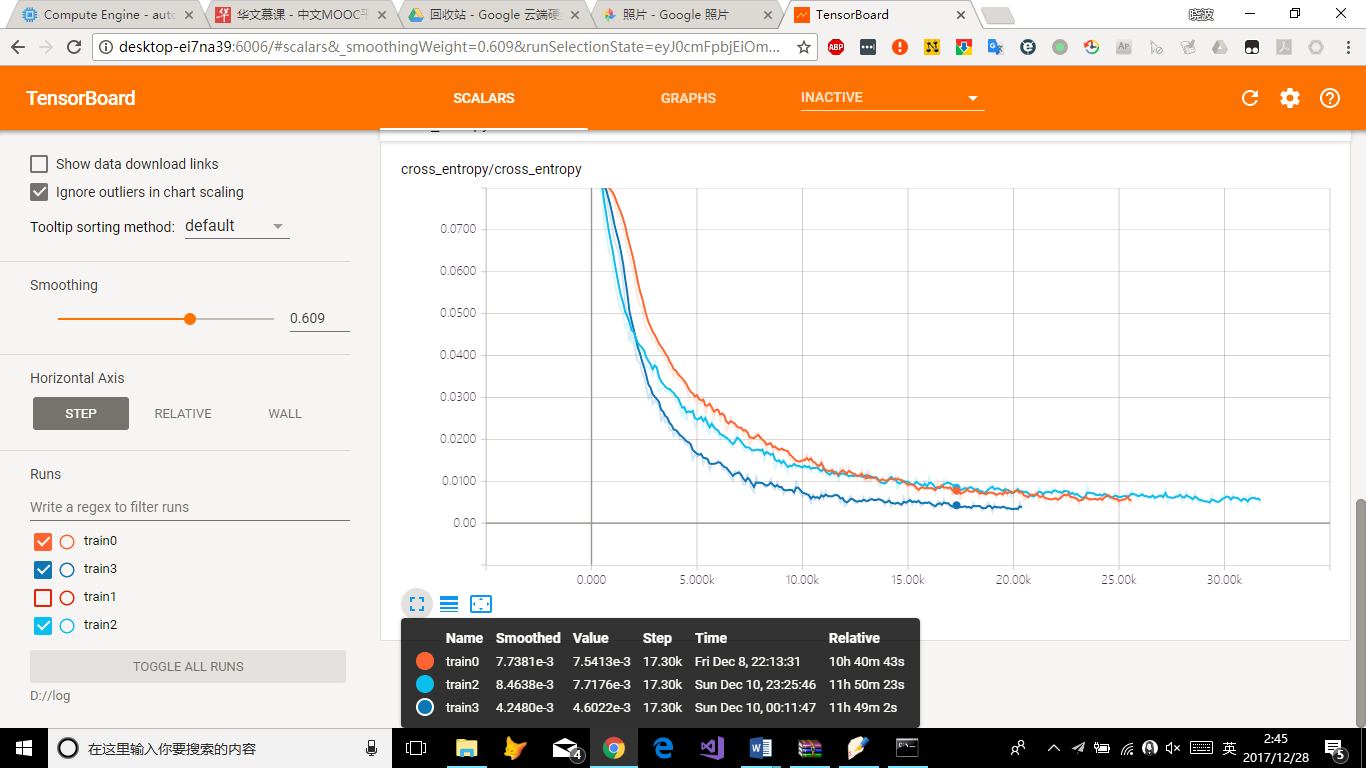
每个卷积层均使用tf.nn.conv2d这个2维卷积函数。池化函数都采用2x2的最大池化层，将一个2x2的像素块降低为1x1的像素，保留原始像素块中灰度值最高的那一个像素，同时strides设为横竖两个方向以2步长，在整体上缩小图片尺寸。

对于输入的定义，X\_IMAGE是特征，Y\_LABEL是真实的标签。因为卷积神经网络会利用空间结构信息，因此将1维的输入向量转为2维的图片结构。最终尺寸是[-1, IMAGE\_HEIGHT, IMAGE\_WIDTH, 1])，-1代表样本数量固定，1代表颜色通道数，尺寸大小取决于图片的height和width。

然后开始定义卷积层。我们首先定义了create\_layer这个通用的卷积层定义函数。第一个卷积层，卷积核函数尺寸为7x7，1个通道，32个不同的卷积核；第二个卷积层是卷积核尺寸变为5x5，通道增加到32个；第三个卷积核函数尺寸为3x3，32个通道，64个不同的卷积核。

在卷积核的规模的选取上，我门对于不同的卷积核大小进行了尝试，开始时选择的是三个5\*5的卷积核，后来看到说有几种不同的卷积核类型，因此尝试了3\*3+5\*5+7\*7和7\*7+5\*5+3\*3两种不同的类型。效果如下：





橘色是三个5\*5，浅蓝是7\*7+5\*5+3+3，深蓝是3\*3+5+5+7\*7可见最后一个的准确率最高，而三个5\*5的是效果最差的。为了防止出现过优化的问题，这里的准确率都是重新生成了100个数据进行检测得到的结果。

由于经历了三次步长为2x2的最大池化，边长只有1/8了，第三个卷积层输出的tensor尺寸为8x20x64。我们使用tf.shape函数对第三个卷积层的输出tensor进行变形，将其转成1维的向量，然后连接一个全连接层，隐含节点为1024，并使用ReLU激活函数。同时为了减轻过拟合，使用了一个Dropout层。最后连接output\_layer层，得到最后的输出

### 4.训练过程

定义损失函数为cross\_entropy，选择sigmoid\_cross\_entropy\_with\_logits()交叉熵来比较loss，用AdamOptimizer进行优化，学习速率设定为0.001

然后开始训练过程。每100步训练后进行一次准确率检测即对100张图片用现有的模型预测，输出一次准确率。每600步训练输出一次模型，防止中间停止或者长时间训练后准确率反而下降的问题。当准确率大于0.96之后停止训练。

### 5.训练结果

最后模型的准确率超过0.96，进一步增加训练次数可以继续提高准确率，在某一版模型中准确率到达过0.98，但是由于失误忘记保存了，所以就只有0.96的准确率。应该来说这一准确率相较于传统的切割字符后识别准确率更高，但是因为训练集和测验集都是自己生成的，所以实际效果需要打折扣。