计算机视觉 Final Project

实验要求：

输入图像：

普通 A4 打印纸，上面有手写的如下信息：

1. 学号
2. 手机号
3. 身份证号

所有输入格式一致，数字号码不粘连，但是拍照时可能角度不正。

输出：

1. 根据标准流程输出每一个主要步骤的结果，包括 A4 纸张的矫正结果，行数据的切割，单个字符的切割结果
2. 对上面的 A4 纸的四个角、学号、手机号、身份证号进行识别，识别结果保存到 Excel 表格（xlsx 格式），对于手写体数字字符的训练数据可以使用 MNIST。

实验流程：

1. 采用边缘检测或者图像分割的方法获取图像的边缘，并计算图像的四个角点。完成图像矫正。
2. 采用图像分割（二值化）的方法，获取图像中的手写字符，输出二值化结果。
3. 针对二值化结果，对 Y 方向投影切割出各个行的图像，例如学号切成单个的图像，手机号和身份证号也是如此。
4. 根据第三步的结果，针对行图像，对 x 方向做投影切割，切割出单个字符。
5. 针对单个切割好的字符，进行分类识别。

实验具体过程：

1. A4 纸边缘检测、矫正
   - 边缘检测包括对图像进行灰度处理，高斯模糊（去除杂项），再对图像使用索贝尔算子（sobel）进行处理，得到边缘
   - 接着进行 hough transform，得到图像上的四条边的方程，进而计算得到四个角点坐标。
   - 得到角点坐标后，接着利用单应性矩阵进行投影变换的计算，对 A4 纸进行矫正

以上几步都使用了之前完成的代码。

2. 切割字符

为了切割得到单个字符进行处理，首先要对原图像进行二值化，然而，因为所给的图像亮度比较均匀的原因，普通的二值化无法将字符和纸张完整分离开来，这里使用到了自适应二值化的方法来处理。

所谓自适应二值化，就是先计算以每个像素为中心的 s×s 像素窗口的平均值，如果当前像素的值小于这个平均值的一个百分比，就将其设置为黑色，否则将它设置为白色。这其中，如何计算 s×s 像素窗口的平均值是一个最大的问题。这里就用到了积分图像（integral image）。

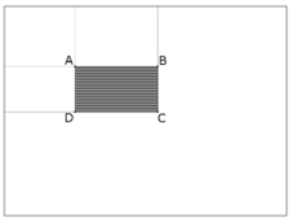
对积分图来说，它每一点（x，y）的值是原图中对应位置的左上角区域的所有值得和：

$$I(x,y)=\sum_{\substack{x'\le x\\ y'\le y}} i(x',y')$$

而且，积分图可以只遍历一次图像即有效的计算出来，因为其每一点的值等于：

$$I(x,y)=i(x,y)+I(x-1,y)+I(x,y-1)-I(x-1,y-1)$$

计算完积分图后，对任意矩形区域的和的计算就可以在常数时间内完成。如下左图中阴影区域的值可由下右图中的公式计算。



$$\sum_{\substack{A(x)<x'\le C(x)\\ A(y)<y'\le C(y)}} i(x',y')=I(C)+I(A)-I(B)-I(D).$$

这样一来，就可以由积分图方便的计算出每个像素周围 s×s 像素窗口的平均值，以进行自适应二值化了。

二值化后对图像进行投影分割，这一步也用到了之前的代码，即是先对图像的每行像素数进行统计，得到数字所在的行范围。再对这些行中的每列像素数进行统计，得到数字所在的列范围，最后得到分割的结果。

3. 数字识别

此前的作业中，曾使用 Adaboost 进行数字识别，但效果非常差，原因可能是完全使用 MNIST 数据集进行训练，其手写风格与测试纸张上的手写字符差别较大。同时也有 Adaboost

分类效果不够好的原因。

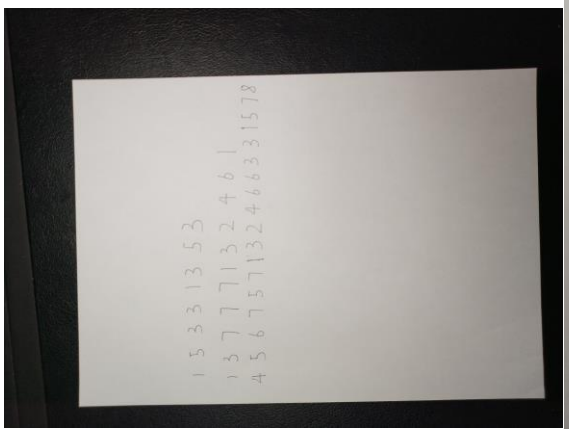
此次使用了 tensorflow 来训练卷积神经网络（cnn）对数字进行识别。CNN 将一系列过滤器应用于图像的原始像素数据，以提取和学习更高级别的特征，然后模型可以使用这些特征进行分类。CNN 包含下列三个组成成分：

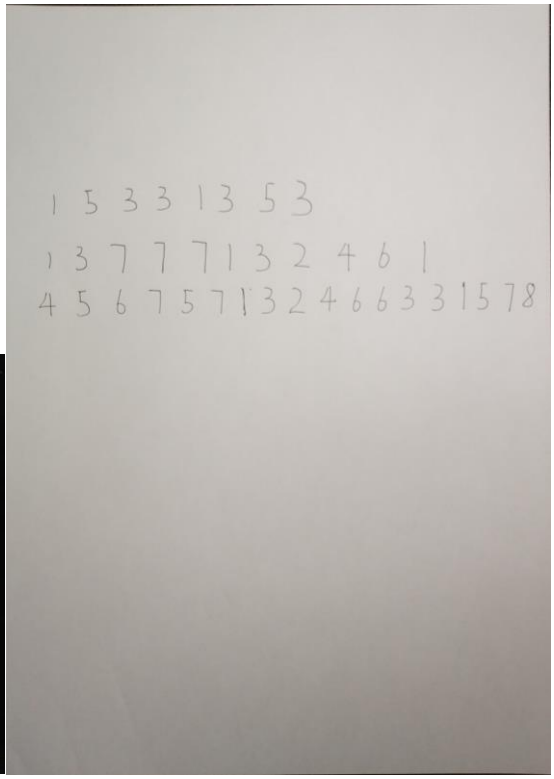
- 卷积层：将指定数量的卷积过滤器应用于图像。对于每个子区域，该层会执行一组数学运算，以在输出特征图中生成单个值。然后，卷积层向输出应用 ReLU 激活函数，以在模型中引入非线性规律。
- 池化层：对卷积层提取的图像数据进行下采样，以降低特征图的维度，从而缩短处理时间。
- 密集（全连接）层：对由卷积层提取并由池化层下采样的特征进行分类，密集层中的每个节点都连接到前一层中的所有节点。

本次实现中，参考了 TensorFlow 官方教程以及代码进行 CNN 的构建，使用 MNIST 训练集（60000 个数据）进行训练，之后对 MNIST 测试集进行测试时准确率达到了 98.5%。使用此模型对作业提供的数据进行测试。

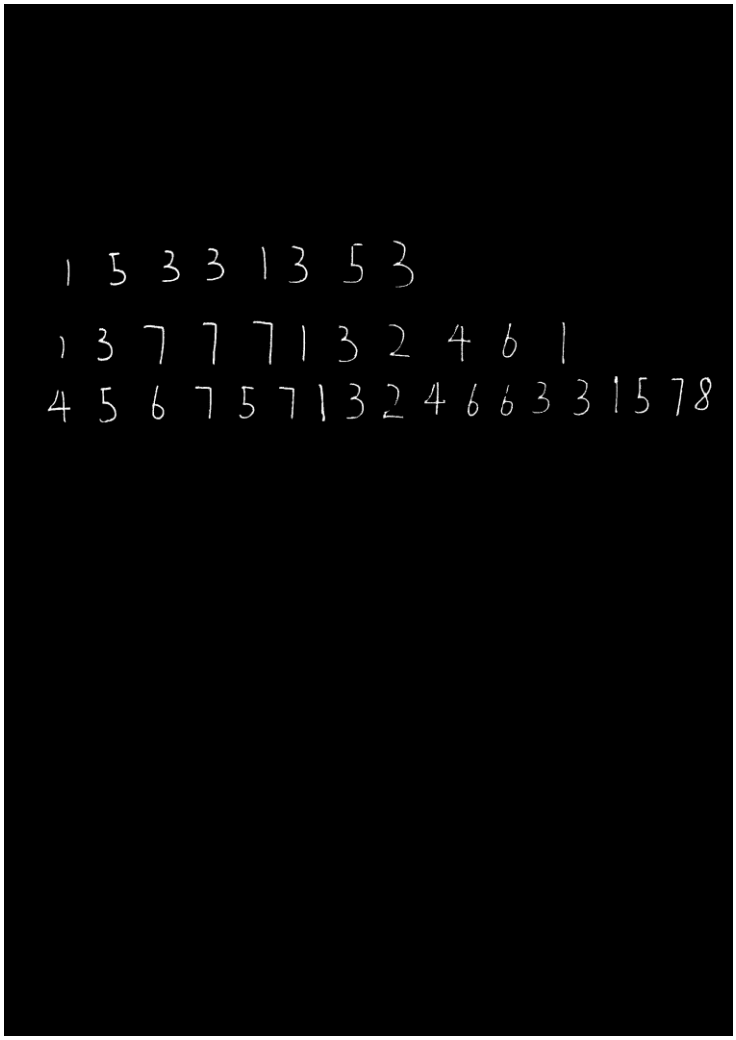
实验结果：

1. 矫正结果示例：





2. 自适应二值化结果示例:



3. 对所给 10 张图片测试得到的 xlxs 文件:

filename	studentID	phone	citizenID
0.jpg	13331353	15779132961	656757133466331512
1.jpg	15331189	13259831641	897913197813273457
2.jpg	17331729	13821918392	471721294612273912
3.jpg	15331344	83521145678	356102177692279321
4.jpg	15331369	13632552831	975341399797719435
5.jpg	15931946	85611194731	445191191114453118
6.jpg	95231381	51719711772	265252224621414998
7.jpg	15331952	15360563784	359939177212221912
8.jpg	19931241	577944411688	2255555555757553431
9.jpg	18392397	18757182323	347275191745942319

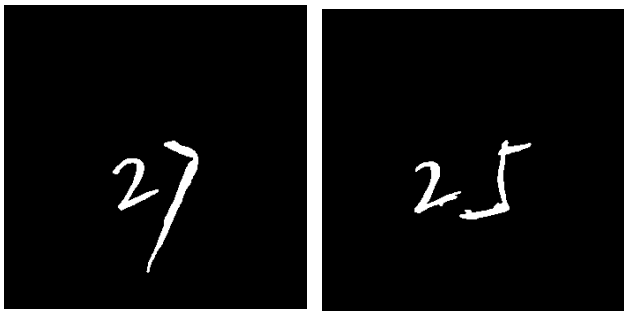
虽然识别率也不是很高，但相比 adaboost 已经有很大进步，Adaboost 有很多根本识别不到。

遗留问题：

1. 在对二值化的结果做字符分割时，这里是先对水平方向做分割，再对垂直方向做分割。这样的方法在字符书写方向都比较水平时的效果是比较好的，但若是字符排列呈一条斜线，就会出现这样的结果：

可见字符不在分割后图像的正中间，甚至出现一幅图像中有两个字符的情况。对于这样的情况，可以想到的改进方法是再对得到的字符做多几次分割，但这样不能确定分割后字符对应的行数。

2. 部分字符粘连较严重，使用投影的方法无法进行分割，如：



3. 可以看到，即使使用了 CNN 来做数字识别，识别成功率仍然不高，只有 50%左右。这是因为 MNIST 训练集与实际书写字符差异较大的原因，可以想到的改进方法为自己制作手写训练集。