

Final：数字字符切割和识别

陈志扬 15331046

目录

实验内容.....	1
测试数据.....	1
开发环境和工具.....	3
编译命令.....	3
实验过程.....	3
提取数字字符.....	3
训练数字识别模型.....	6
进行数字识别预测.....	6
实验结果.....	7
作业总结.....	10

实验内容

将包含学号、手机号和身份证号的 A4 纸拍照保存为“学号.bmp”文件，然后对 A4 纸做校正，并采用 OSTU 等方法对图像二值化，再对 A4 纸上的数字做字符切割，然后使用 SVM 等方法训练模型，最后对 A4 纸上的数字做识别，并将识别得到的学号、手机号和身份证号保存在一个 Excel 文件中。

测试数据

测试数据格式如下：

15331009

15521117484

351301199706250531

15331009

15521117484

351301199706250531

15331009

15521117484

351301199706250531

15331265

24112175883

373862199603250114

15331265

24112175883

373862199603250114

15331265

24112175883

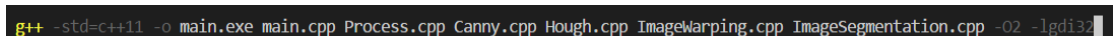
373862199603250114

开发环境和工具

Windows 10 64 位系统、Visual Studio Code

编译命令

```
g++ -std=c++11 -o main.exe main.cpp Process.cpp Canny.cpp  
Hough.cpp ImageWarping.cpp ImageSegmentation.cpp -O2 -lgdi32
```



实验过程

实验过程主要分为以下三个过程：

提取数字字符

提取图像中的数字字符，并将它们切割出来并保存为一张张小的图片。

具体流程如下：首先对图像进行预处理，主要是 `resize` 和转灰度图，然后使用 Canny 算法进行边缘检测得到图像边缘图，再进行霍夫变换得到 A4 纸的四个角点以及四条直线，接着进行图像变换 `Warping`，最后对图像中的数字进行字符切割，将切割得到的图像保存下来，以用于后续数字识别工作。



Figure 1 提取图像数字字符流程

这个过程的前半部分直至 A4 纸校正都是之前的作业，综合起来就能成功对图像进行 A4 纸检测并校正，这里我不再赘述。

最后一步数字字符切割是本次实验的难点。刚开始做的时候，使用的是 opencv 寻找轮廓的接口函数 `findContours`，然而在测试时，发现它是按照从上到下、从左到右扫描，而且在从左到右的扫描过程中也很不稳定，经常出现顺序混乱的问题，所以并不能用来解决数字字符切割的问题。

因此，我参考了网上一篇写得很优秀的[博客](#)，其算法流程大致如下：首先将 Warping 得到的图像，也就是校正后的 A4 纸做二值化处理，然后根据垂直方向的直方图，按行分割二值化后的图像，此时每行都包含一行数字（可能有多列），然后根据水平方向的直方图，按列分割每一行，得到被行列分割的图像，此时分割的每一个区域包含一个完整的数字串，然后对每一个区域进行扩张操作从而修复断裂字符，最后，使用[连通区域标记算法](#)从左到右分割数字字符，将分割得到的字符图像及其文件名保存下来，文件名保存下来的目的是为了后续数字识别工作可以方便加载要进行识别的字符图像。

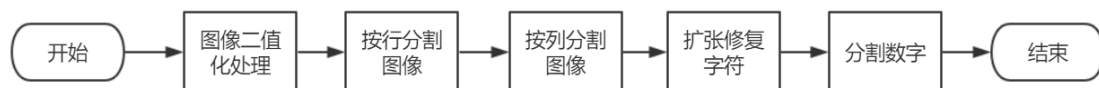


Figure 2 数字字符切割流程

具体技术细节如下：

图像二值化：代码采用了全局阈值分割算法，阈值为 135。全局阈

值分割就是将小于设定阈值的像素设为 0，否则设为 255，该算法实现简单，且速度很快，但效果不是很棒，常受光照的影响，但用于此作业已足够。

按行分割图像：利用垂直方向直方图的波峰和谷底，可以找到由黑转白或由白转黑的拐点，两个拐点的中间值就是按行分割的分割点。而由于可能出现由断裂的点造成的波峰和谷底，此时应该抛弃由此得到的分割点，因此我们可以统计子图的黑色像素个数，只有超过一定比例该子图才被认为存在完整数字，故可将其分割出来。

按列分割图像：利用水平方向直方图的波峰，计算所有波峰之间间距的均值，只有当间距大于均值的一定倍数时，才被视为要进行列分割，分割得到的每一个子图都包含一个完整的数字串。

扩张修复字符：做二值化时阈值太小容易造成字符断裂的问题，因此需要对每一张子图，进行[扩张](#)操作。首先，假设当前位置为白色像素点，检查上下 1 各单位像素和左右 2 个单位像素，统计黑色像素的总数。只有当黑色像素的总数大于 0，才把该白色像素点设为黑色。其次，假设当前位置为白色像素点，检测上下左右 1 个单位像素，若为黑色，则把该白色像素点设为黑色。

分割数字：使用[连通区域标记算法](#)从左到右分割数字字符。首先扫描图像第一列和第一行，每个黑色像素点作为一类，打上标记，然后按列进行遍历，遇到黑色像素点，检测其左下、左前、左上、正上这四个位置的像素点，再找到它们的最小类标记，将其余标记的黑色像素点以及当前位置的像素点也标记为最小类标记；如果这四个位置

的像素点都不是黑色的，则当前位置作为新类，并打上新标记。最后同一类即代表是同一个数字，据此可以提取分割单个数字字符。

训练数字识别模型

具体流程如下：首先加载 mnist 数据集，然后使用神经网络的多层感知器 MLPClassifier 进行训练，再把训练得到的模型保存下来，以用于后续数字识别工作。

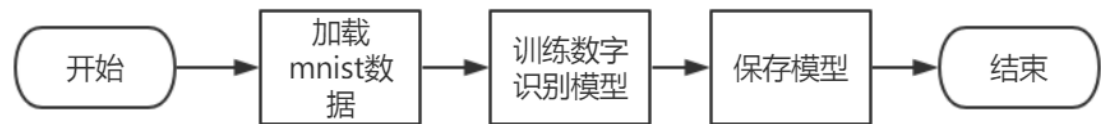


Figure 3 训练数字识别模型流程

进行数字识别预测

具体流程如下：首先加载第二步得到的模型，再根据第一步得到的存储数字字符图像文件名的文件，逐一加载字符图像，然后送入预先训练好的模型进行数字识别，将单个字符预测结果存入一个字符串，再将字符串存入一个 python 列表，最后利用 pandas 的 ExcelWriter 将列表存入一个 Excel 文件。

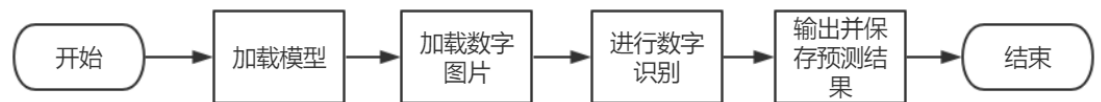


Figure 4 进行数字识别流程

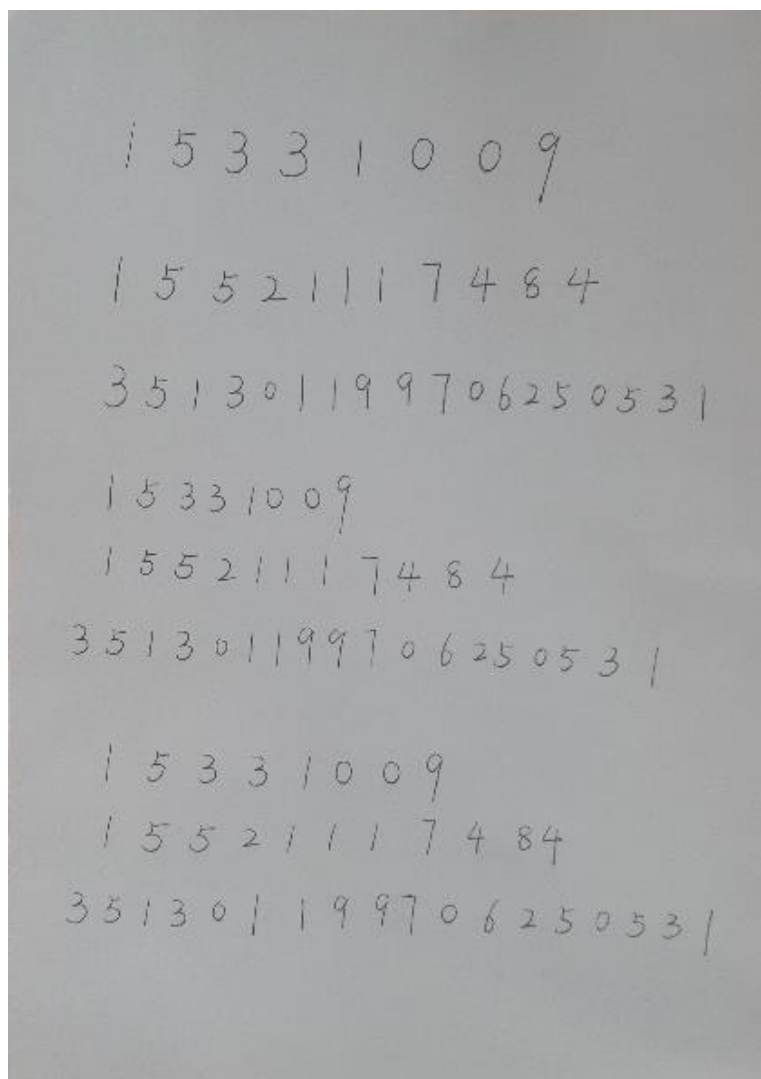
实验结果

模型训练效果如下：采用两层隐藏层神经网络，第一个隐藏层有 400 个神经元，第二个隐藏层有 200 个神经元，最大迭代次数设为 400。

0.9603 是模型在测试集上的预测准确率

```
Iteration 266, loss = 0.02723227  
Iteration 267, loss = 0.02713053  
Iteration 268, loss = 0.02702990  
Iteration 269, loss = 0.02692866  
Iteration 270, loss = 0.02682837  
Iteration 271, loss = 0.02673689  
Iteration 272, loss = 0.02664042  
Iteration 273, loss = 0.02654579  
Training loss did not improve more than tol=0.000100 for two consecutive epochs. Stopping.  
0.9603
```

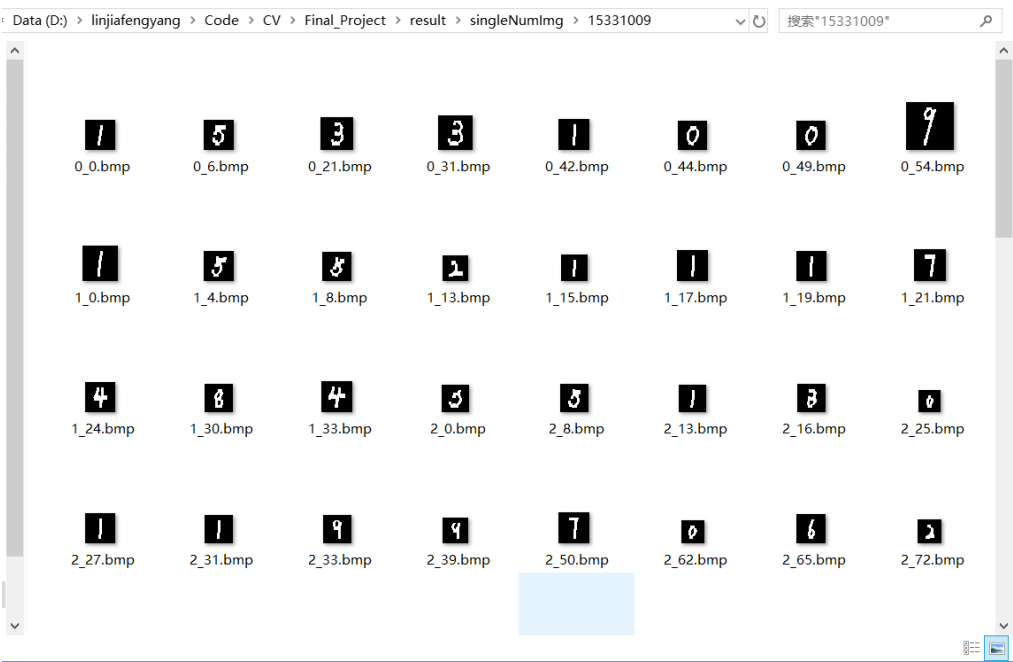
A4 纸校正结果如下：



四个角点如下：

```
(x, y) = (315.626, 477.393)
(x, y) = (323.095, 81.2204)
(x, y) = (34.5456, 500.404)
(x, y) = (28.9985, 59.004)
```

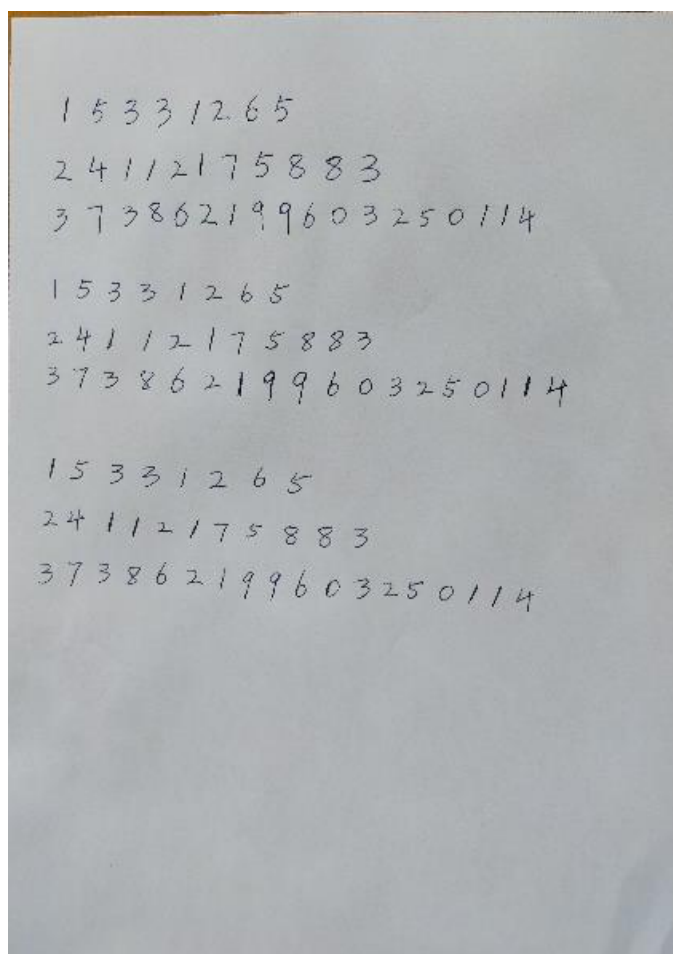
数字字符切割结果如下：



数字识别结果如下：

	A	B	C
1	number		
2	10331001		
3	18321117484		
4	301361199196280331		
5	13251991		
6	18721117984		
7	891291199146774931		
8	18931909		
9	15311117499		
10	331261199706230361		
11			
12			
13			

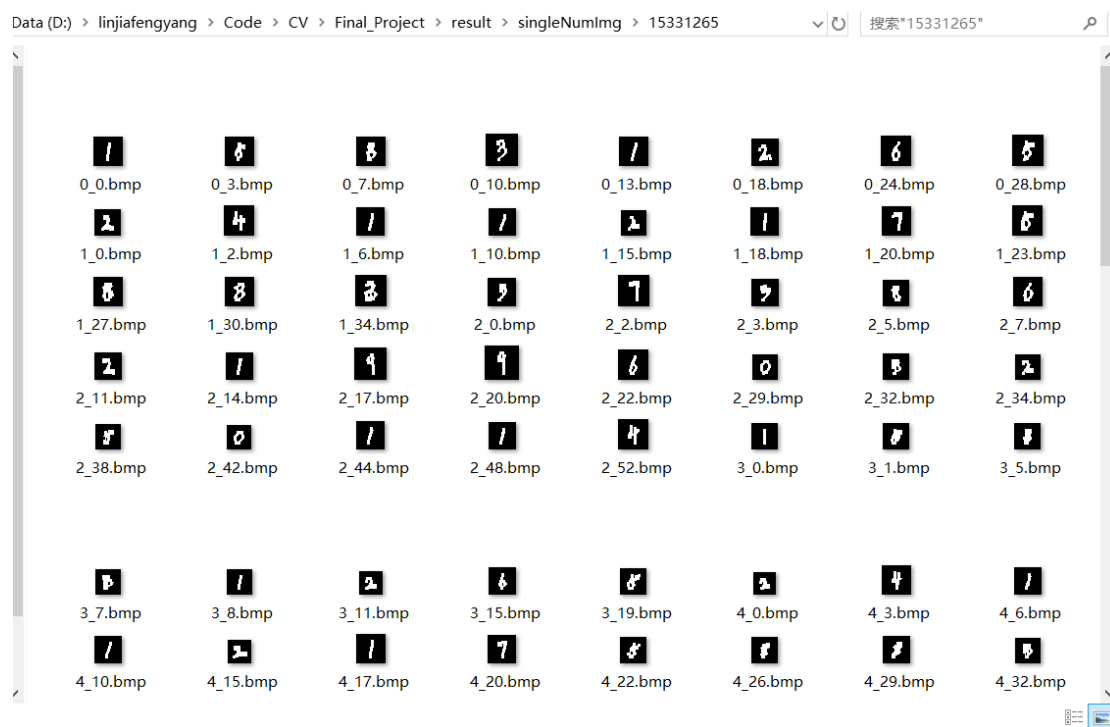
A4 纸校正结果如下:



四个角点如下:

```
(x, y) = (280.406, 407.164)
(x, y) = (297.022, 29.6271)
(x, y) = (12.9154, 400.44)
(x, y) = (25.0527, 14.2312)
```

数字字符切割结果如下:



数字识别结果如下：

	A	B	C
1	number		
2	19891269		
3	24112170982		
4	979862199699280119		
5	19951668		
6	64116178899		
7	918826199602480114		
8	18991265		
9	14116178999		
10	919826111897690114		
11			

作业总结

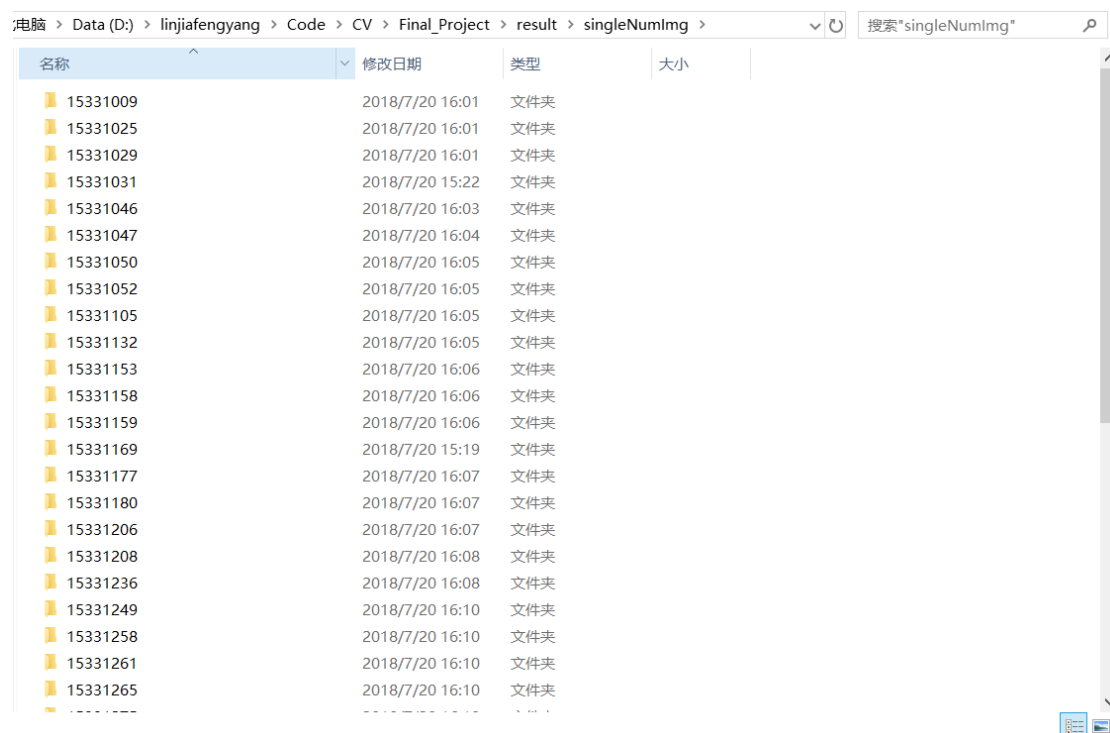
在做作业前就已经知道肯定有一些照片很难校正，因为拍照容易受到外在因素影响，因此排除拍照的一些外在因素带来的 A4 纸校正

影响，然而本次作业的难度系数还是很高的。


由于我之前并没选修数字图像处理课程，对完成作业多多少少有点影响，特别在做数字字符切割时，遇到的挑战非常大。首先是图像二值化处理，由于采用的全局阈值分割很难对所有校正后的图像做正确分割，因为容易受到拍照时亮度不均等因素的影响，导致做直方图的效果不是很好。好在还有大部分图片能够正确二值化。接下来是根据垂直方向和水平方向的直方图将图像分割成带有完整数字串的一个个区域子图，这部分的算法思想容易理解，紧接着是对断裂字符做扩张修复，这里就吃了没修数字图像处理课程的亏了，我花了比较长的时间理解它。最后是连通区域标记算法寻找单个数字字符并切割的过程，思想其实好懂，但实现起来比较困难。

数字识别模型训练一开始我采用了 SVM 算法，但是预测的效果非常差，所以采用了神经网络的多层感知器进行训练，采用两层隐藏层神经网络，第一个隐藏层有 400 个神经元，第二个隐藏层有 200 个神经元，最大迭代次数设为 400，训练速度挺快的，而且预测速度也非常快。

进行数字识别预测时，由于要一次性处理所有字符切割图像，所以我在做数字字符切割时，在 result 文件夹中创建了 singleNumImg 文件夹用来存储每个同学拍的照片的切割结果，以学号命名文件夹，并在一个 imageDir.txt 文件中保存这些路径，文件结构如下：



imageDir.txt 文件内容如下：

 imageDir.txt - 记事本

文件(F) 编辑(E) 格式(O) 查看(V) 帮助(H)

```
result/singleNumImg/15331009
result/singleNumImg/15331025
result/singleNumImg/15331029
result/singleNumImg/15331031
result/singleNumImg/15331046
result/singleNumImg/15331047
result/singleNumImg/15331050
result/singleNumImg/15331052
result/singleNumImg/15331105
result/singleNumImg/15331132
result/singleNumImg/15331153
```

然后在预测时，python 通过该文件读取对应文件夹内的字符分割图片，将这些图片逐一 resize 为一张 28*28 的图像再送入模型进行预测，然后生成一个对应的 Excel 文件（以学号命名）保存预测的结果。