创新实践实验报告

**--项目名称：验证码识别系统**

**小组成员：**

**姓名：徐紫丹 学号：17052205**

**姓名：支浩仕 学号：17051537**

**姓名：张诗雨 学号：17051607**

**姓名：张卓群 学号：17051608**

**指导老师：魏丹**

**提交报告时间：2018年12月 13日**

1. **项目要求：**

对验证码图片中的数据、英文、汉字成功识别。

主要有5类验证码图片，难度依次递增，每一类验证码提供1万个训练样本，最后通过另外5000个测试样本的识别率评分。

第一类：数字四则运算，有噪点干扰，输出计算结果。示例：



输出：2+5-8=-1

1. **项目规划：**

**1.原方案（切割+模型训练）**

流程图：

字符特征提取

切割

图片预处理

模型训练

识别

图片字符标记

**2.现方案（不切割+OCR）**

OCR 识别

去 噪

二值化

灰度处理

验证码计算

1. **步骤分析：**

**图片预处理：**

1. 二值化：

代码：

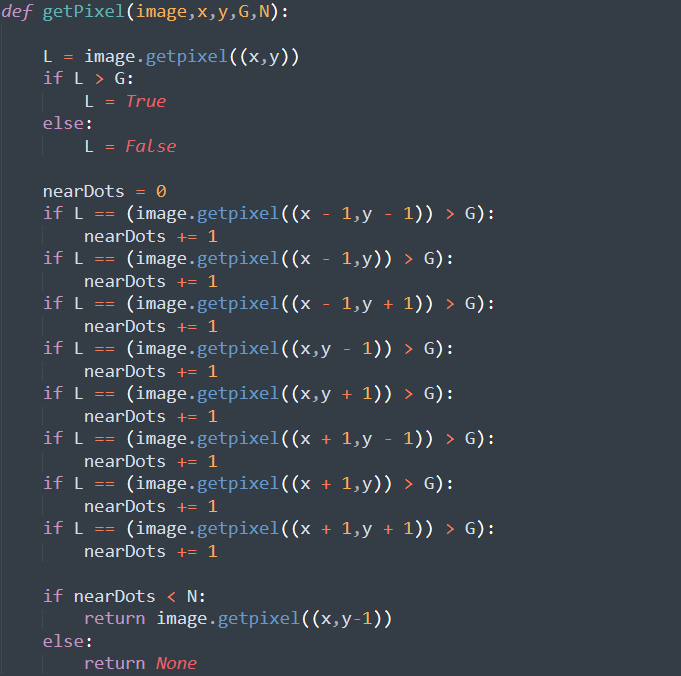


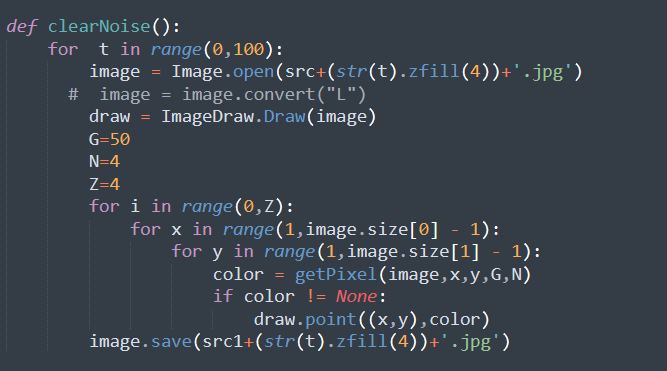
算法思路：

开始读取图像地址，设置灰度阈值，低于灰度阈值的点全部填白色，若大于，图像地址加一继续读取。

1. 去噪：

代码：





算法思路：

1. 刚开始去噪使用的是洪水填充法，步骤如下：如图对一个像素点周围八个点进行读取，若是白点则nearDots加‘1’
2. 当nearDots小于一个值N时返回该点，否则返回白点
3. 将去噪后的图片保存

实例1：



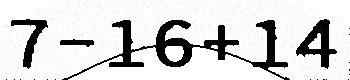
效果1：



实例2：

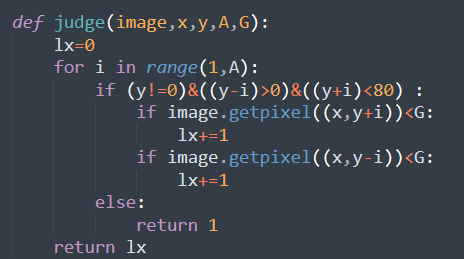


效果2：



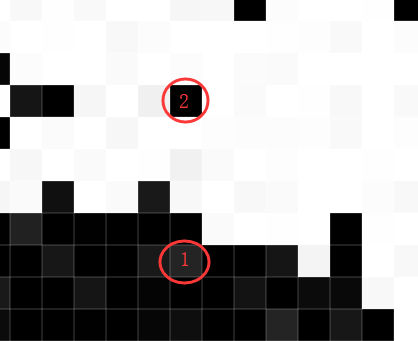
弊端：对于较粗的干扰线会没有办法去除，导致识别率低

由于采用pytesseract的方案对于去噪的要求高，该方法的识别率低，因此，去噪在这里采用了新方法：

首先，对一个点的上下/左右的像素点进行读取，

若A设置为3，就读去该点上下/左右3个像素内的点，若是黑点，LX+1, 当LX小于一个值N时将该点画为白点。

（judge判断上下的像素点，judge1判断左右的像素点）



如图，1的LX=3，保留为黑点，2的LX=0,改为白点。

去噪效果：

实例1：



效果1：



实例2：



效果2：



实例3：

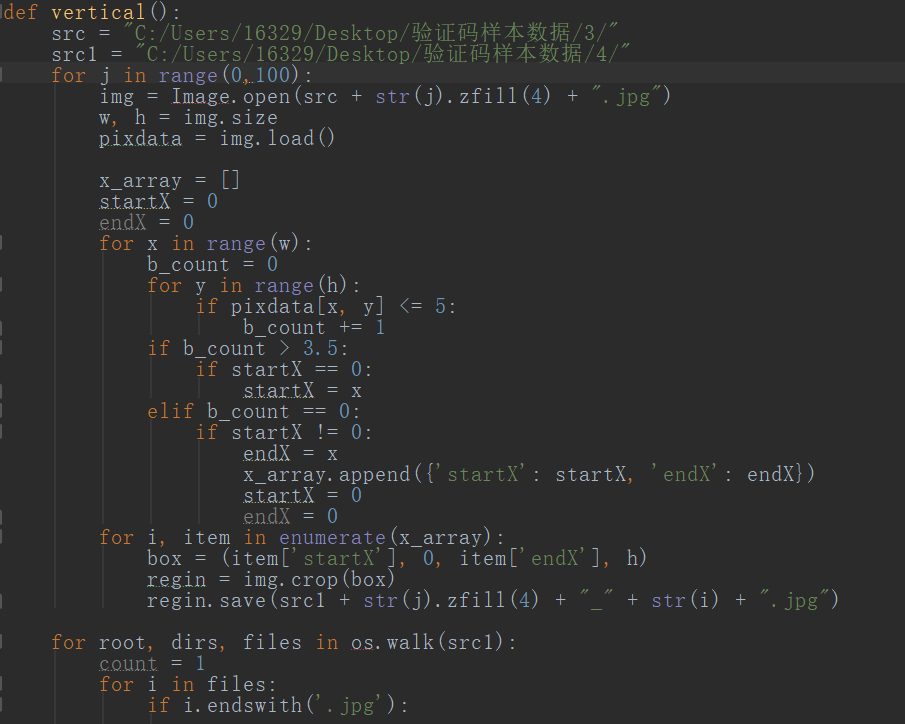


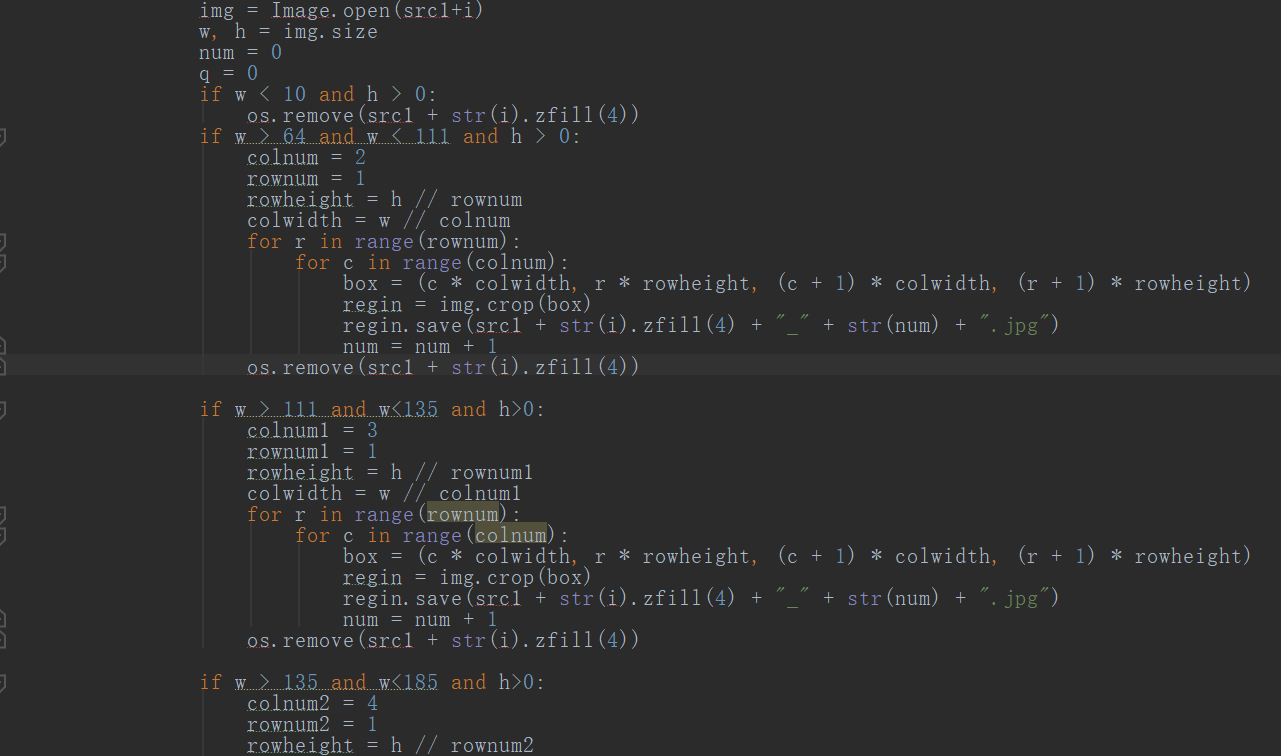
效果3：

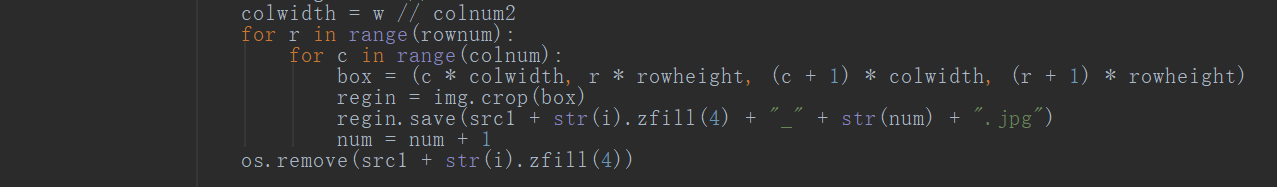


**切割（仅在原方案使用）：**

代码：







算法思路**：**

图片中的字符不是均匀分布的，所以平均切割的代码不适用于此类图片，在参考老师给的资料“验证码识别技术研究”之后，决定采用竖直投影法进行切割。其原理为:假定黑点为目标，水平垂直投影字符算法是分别向X轴和Y轴投影，投影值是黑点的总数，根据相邻波谷的位置来确定单个字符在X轴和Y轴的位置范围。一般先在X轴上投影，根据投影值为0的列确定单个字符所占宽度，再向Y轴投影，根据投影值为零的行得到字符高度，从而将单个字符提取出来。

预处理图片经切割后效果：

**0000_0 0000_1   **

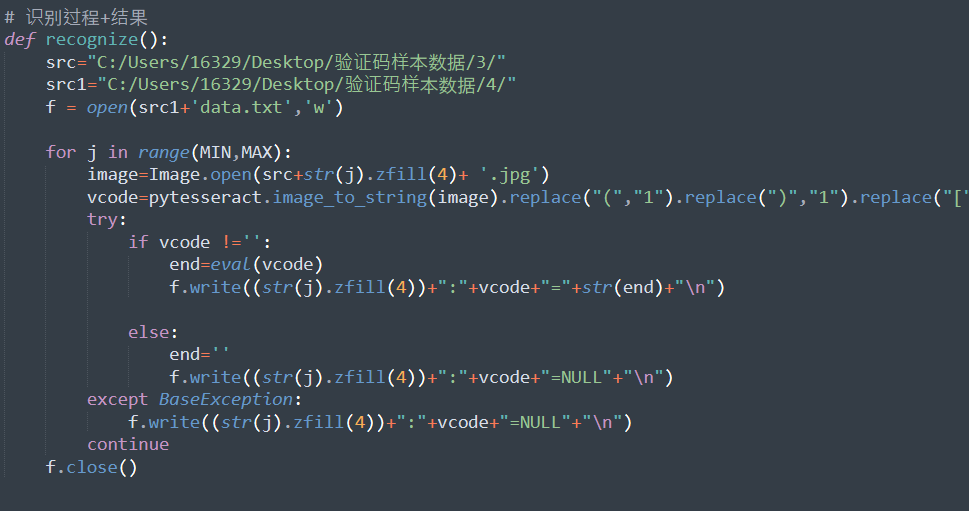
**字符特征提取：**

**参考网址：**[**https://blog.csdn.net/qq\_35923581/article/details/79487579**](https://blog.csdn.net/qq_35923581/article/details/79487579)

1. 尺寸归一：因切割后图片大小不一，所以要统一图片大小。
2. 对素材图片进行人为分类，即标记。
3. 定义单张图片的识别特征

我们原本打算就单个字符分成单独的一格，并对每一行列的特征值进行提取，并建模，但是由于尺寸归一不好把握，以及建模过程过于复杂、工程量大，且识别效果不好，所以采用另一种识别方法。

**图片识别（OCR）+验证码计算：**

**代码：**

注：由于方法image\_to\_string对于识别某些字符有些误差，采用了replace()方法进行人工纠正。

算法思路：

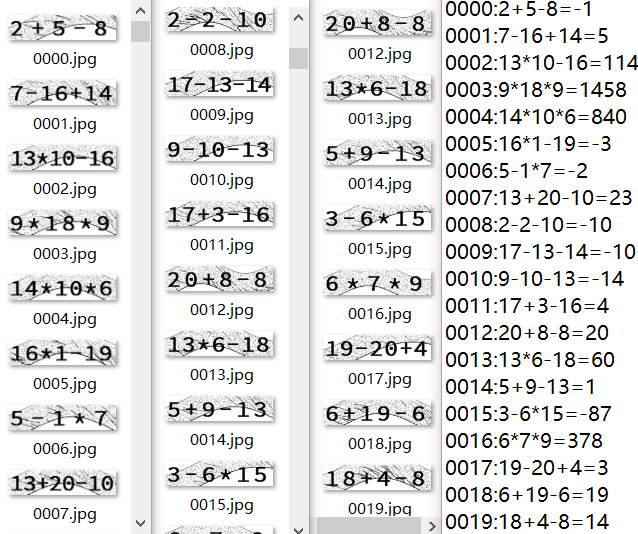
其原理是通过专门的OCR软件，将含有文字的图像按字切割成可独立识别的单元，然后运用各种算法分析每个图像单元中文字的形态特征，通过比对标准特征库中的数据，判断出该文字在计算机中的标准编码，并按通用格式输出保存在文本文件中。并将识别结果中识别错误的结果替换为正确结果，例如识别出来的（替换为数字“1”。在计算验证码结果的时候，采用了eval()函数，将字符串string对象转化为有效的表达式参与求值运算返回计算结果。

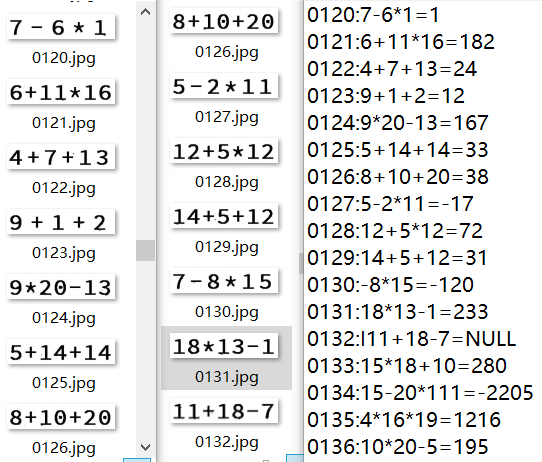
识别效果如图：



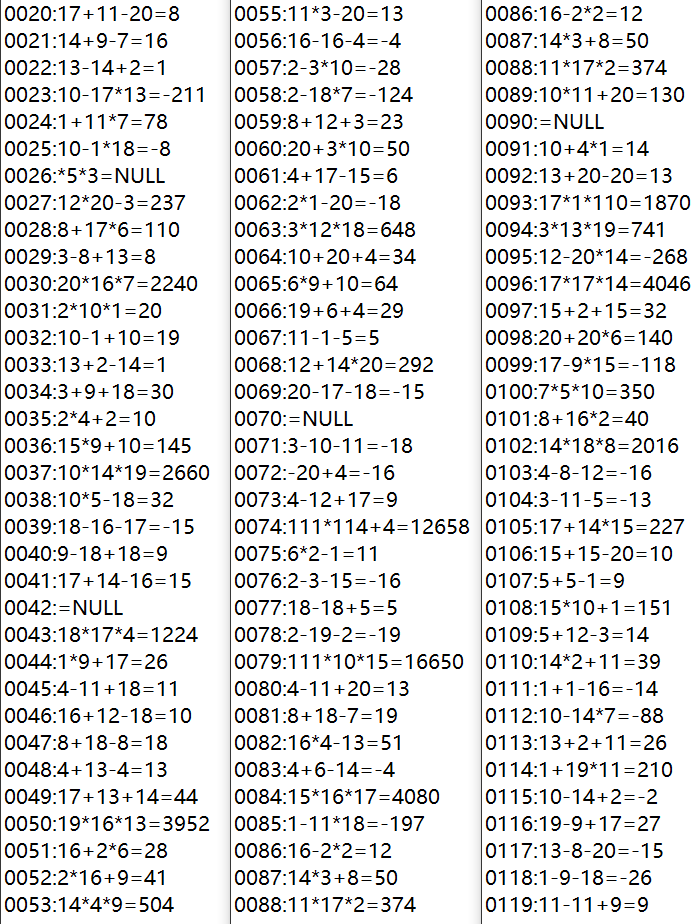
1. **总体识别效果：**

样本和识别结果对比：





一百个数据结果：



经过多次测验，这种方法的识别率为92％左右，对于大多数的平均分布而有较多的噪点和干扰线的图片能够正确识别，达到了我们理想的效果。

1. **总结经验：**

验证码是随着互联网发展起来的产物，是一种验证网络终端用户是人类还是计算机自动程序的技术，其目的是为了保障互联网应用的安全。目前已有的验证码的形式有很多种，包括字符型、音频型、问题型和操作型等，但当前最常见的还是基于字符的验证码，这主要是因为字符验证码的实现更加简单并且具有更好的用户体验。

我们研究了目前常用的字符识别算法，并综合了它们基本的算法原理和算法过程。并自学了python语言且学会参考网络及书籍资源并加入自己的想法。例如去噪过程中第一次去噪过程得到的效果并不尽如人意，干扰线去的不够干净，也直接导致后面切割和识别的效果。通过查阅资料改进方案后，得到了自己想要的效果。其次对于所给图像的切割，提出了一种结合平均切割算法和竖直投影的切割算法。对验证码而言，不同的验证码需要设计不同的识别算法，我们在进行特征选择和模型训练时，由于所给的图片不太适合于这种算法且操作过于麻烦，所以在字符识别阶段我们选择了OCR识别作为字符识别的算法，并利用python中直接给出的函数完成了字符串的运算并得出识别结果。

通过这次创新实践，我们不仅知识储备得到了拓展并且更加深入了解了数据挖掘的分类算法：决策树、人工神经网络、支持向量机等等。在这个过程中，小组分工明确，且遇到问题时一起讨论并寻找解决问题的方法，丰富了实践经验且感受到了团队的凝聚力。

**附录：**

**完整代码：**

**from PIL import Image,ImageDraw**

**import os**

**import pytesseract**

**pytesseract.pytesseract.tesseract\_cmd='C:/Pt-OCR/tesseract.exe' #可删**

**# 图片数量**

**MIN=0**

**MAX=20**

**# 二值化处理**

**def two\_value():**

**for i in range(MIN,MAX):**

**# 打开文件夹中的图片**

**src="C:/Users/16329/Desktop/验证码样本数据/1/train/"**

**src1="C:/Users/16329/Desktop/验证码样本数据/2/"**

**image=Image.open(src+(str(i).zfill(4))+'.jpg')**

**# 灰度图**

**lim=image.convert('L')**

**# 灰度阈值设为165，低于这个值的点全部填白色**

**threshold=220**

**table=[]**

**for j in range(256):**

**if j<threshold:**

**table.append(0)**

**else:**

**table.append(1)**

**bim=lim.point(table,'1')**

**bim.save(src1+(str(i).zfill(4))+'.jpg')**

**def judge(image,x,y,A,G):**

**lx=0**

**for i in range(1,A):**

**if (y!=0)&((y-i)>0)&((y+i)<80) :**

**if image.getpixel((x,y+i))<G:**

**lx+=1**

**if image.getpixel((x,y-i))<G:**

**lx+=1**

**else:**

**return 1**

**return lx**

**def judge1(image,x,y,A,G):**

**jx=0**

**for i in range(1,A):**

**if (x!=0)&((x-i)>0)&((x+i)<350) :**

**if image.getpixel((x+i,y))<G:**

**jx+=1**

**if image.getpixel((x-i,y))<G:**

**jx+=1**

**else:**

**return 1**

**return jx**

**def clearnoise():**

**src="C:/Users/16329/Desktop/验证码样本数据/2/"**

**src1="C:/Users/16329/Desktop/验证码样本数据/3/"**

**for i in range(MIN,MAX):**

**image = Image.open(src+(str(i).zfill(4))+'.jpg')**

**image = image.convert("L")**

**w,h=image.size**

**draw = ImageDraw.Draw(image)**

**for x in range(0,w):**

**for y in range(0,h):**

**if (judge(image,x,y,4,50)<3)|(judge1(image,x,y,4,50)<3):**

**draw.point((x,y),255)**

**image.save(src1+(str(i).zfill(4))+'.jpg')**

**# 识别过程+结果**

**def recognize():**

**src="C:/Users/16329/Desktop/验证码样本数据/3/"**

**src1="C:/Users/16329/Desktop/验证码样本数据/4/"**

**f = open(src1+'data.txt','w')**

**for j in range(MIN,MAX):**

**image=Image.open(src+str(j).zfill(4)+ '.jpg')**

**vcode=pytesseract.image\_to\_string(image).replace("(","1").replace(")","1").replace("[","1").replace("]","1").replace("}","1").replace("{","1").replace(' ','').replace("\*x","\*").replace("x\*","\*").replace("x","\*").replace('k','\*').replace("l",'').replace('T','').replace('®','\*').replace('~','-').replace('O0','0').replace('%','\*').replace("«","\*").replace("\*\*","\*")**

**try:**

**if vcode !='':**

**end=eval(vcode)**

**f.write((str(j).zfill(4))+":"+vcode+"="+str(end)+"\n")**

**else:**

**end=''**

**f.write((str(j).zfill(4))+":"+vcode+"=NULL"+"\n")**

**except BaseException:**

**f.write((str(j).zfill(4))+":"+vcode+"=NULL"+"\n")**

**continue**

**f.close()**

**two\_value()**

**clearnoise()**

**recognize()**