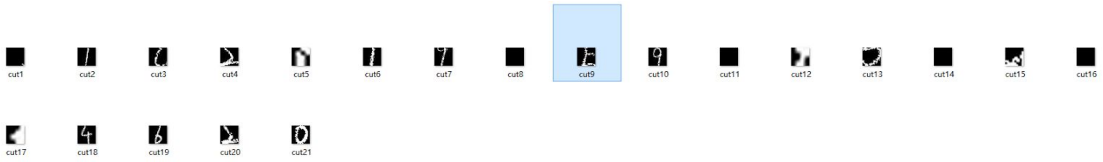
识别手写数字

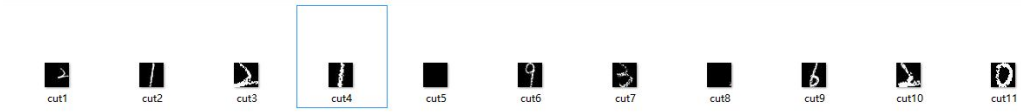
1. 方法：先将图片设为二值（黑白），再将图片中的数字定位，用正方形框出来，改为28*28的图片数组（以上方法使得一张图片分割许多形式与mnist数据集相同的数字图片），放入使用mnist数据集训练过的模型 进行识别。
   （1）二值化图片：
   先将图片转为单通道，再设置阀值，颠倒图片像素值。
   （2）数字定位方法：连通分量
   每行每列地搜索图片像素，如果该像素是黑变白的第一个，则它属于数字，将它标记，再通过它搜寻它的上下左右像素，如果也是白色，则用同样标记，搜寻上下左右，以此递归，当所有标记的像素的上下左右像素都是黑色的，则联通分量内的像素寻找结束，结束递归。

2. 遇到的问题：
   （1） 图片存在噪音，二值化的阀值需要实验设置初步去噪音。
   下图纯黑的图片是噪音点。

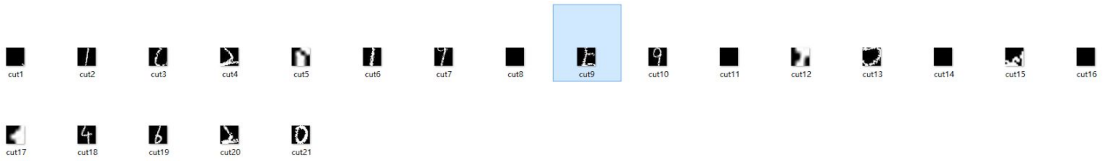


解决方法：
设置连通分量中像素值的最小值，防止零碎的噪点被切为数字图片。

优化后：

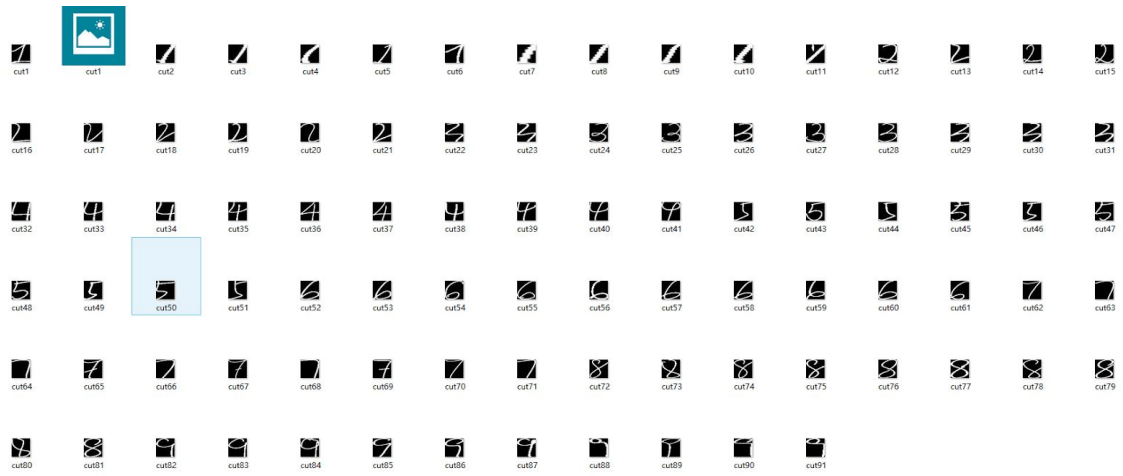


（2） 连通分量：存在数字不是1个连通分量（有断点）。
下图5被分割成了两个图片

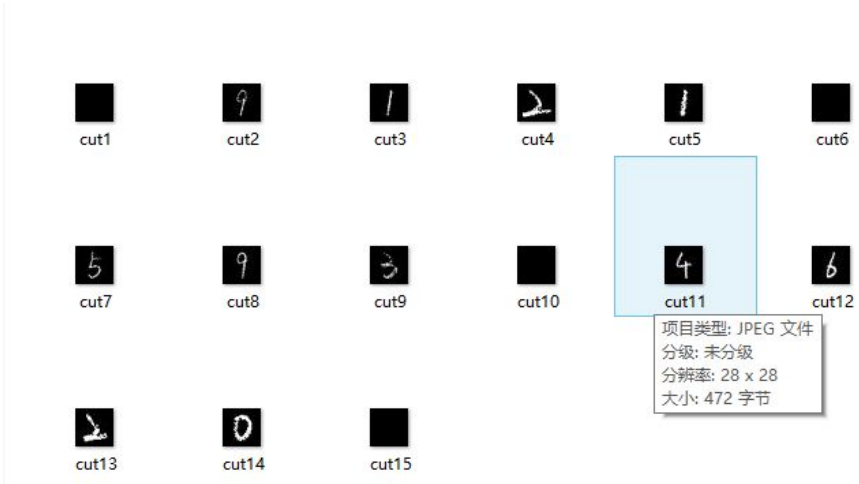


解决方法：
扩大递归域，不仅递归上下左右的像素点，同样递归邻居一圈的八个点，向外第二圈，第三圈的八个点。
解决后：

（3） 图片变形：
以像素边距作为图片边距的情况，通常图片会切成长方形，再拉伸成正方形时图片会变形，这不是我们期望的。（如图中的1）



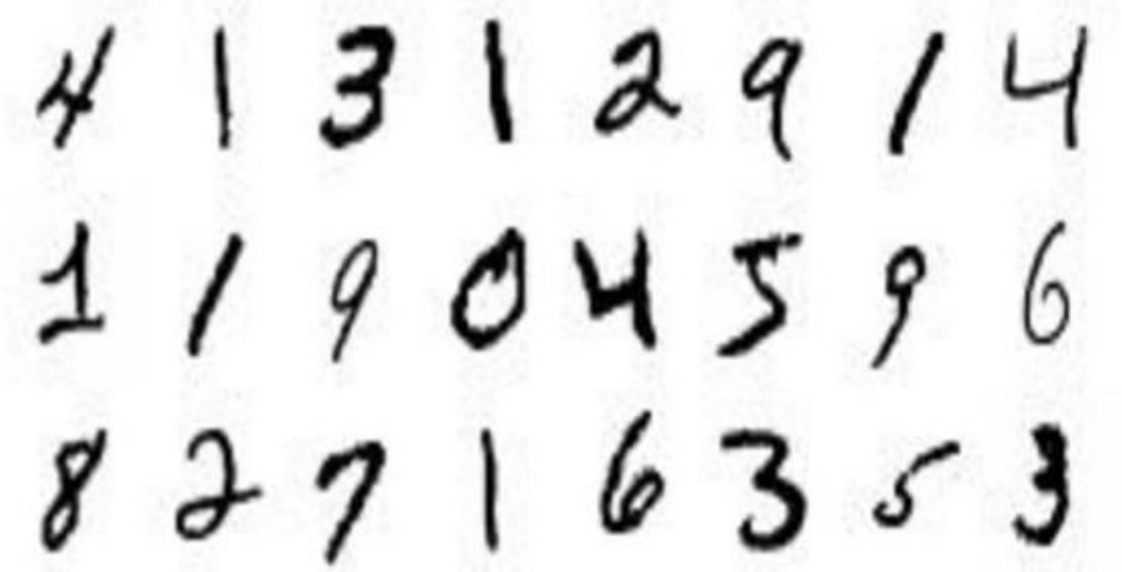
解决方法：
找到中心点和高度，将高度乘以一定比例加长，以中心点为正方形中心点，以加长后的长度为正方形边长。
解决后：



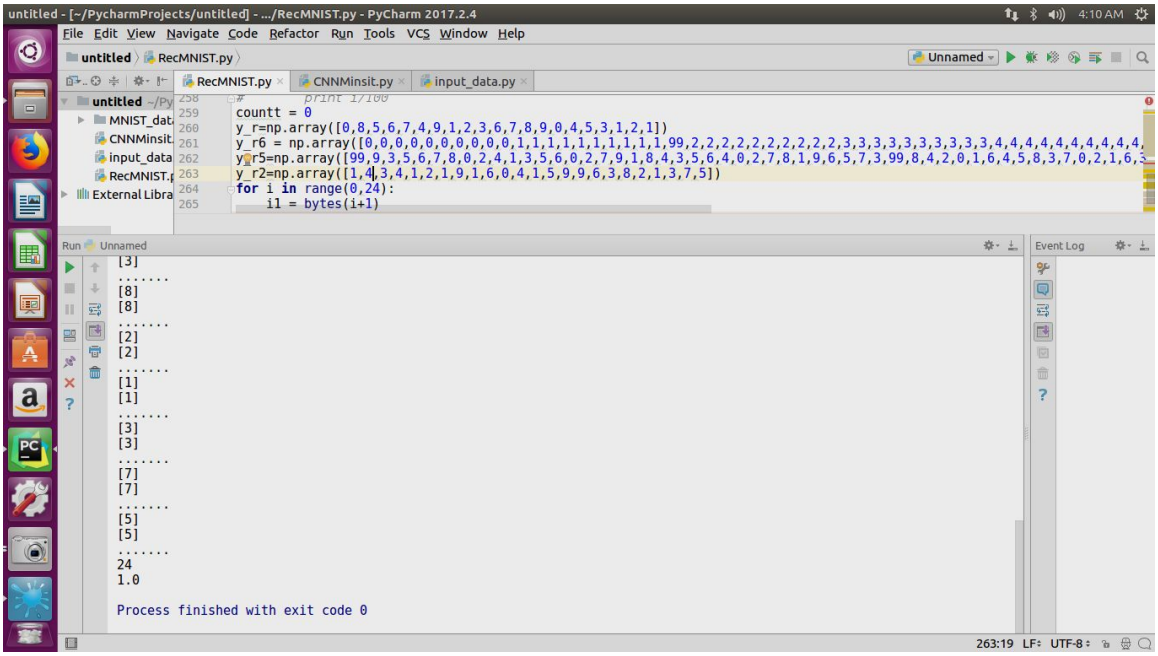
（4） 为了防止递归切割使用太长的时间，会将大图片先压缩再处理。
（5） 尚未解决：两个数字连在一起，无法切开，放弃识别。

3.结果展示（省略号中间两维数，上面是真实数，下面是识别数，识别顺序和图片上的排列无关，切割失败的图片不识别）

图片2

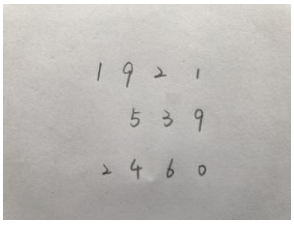


结果：

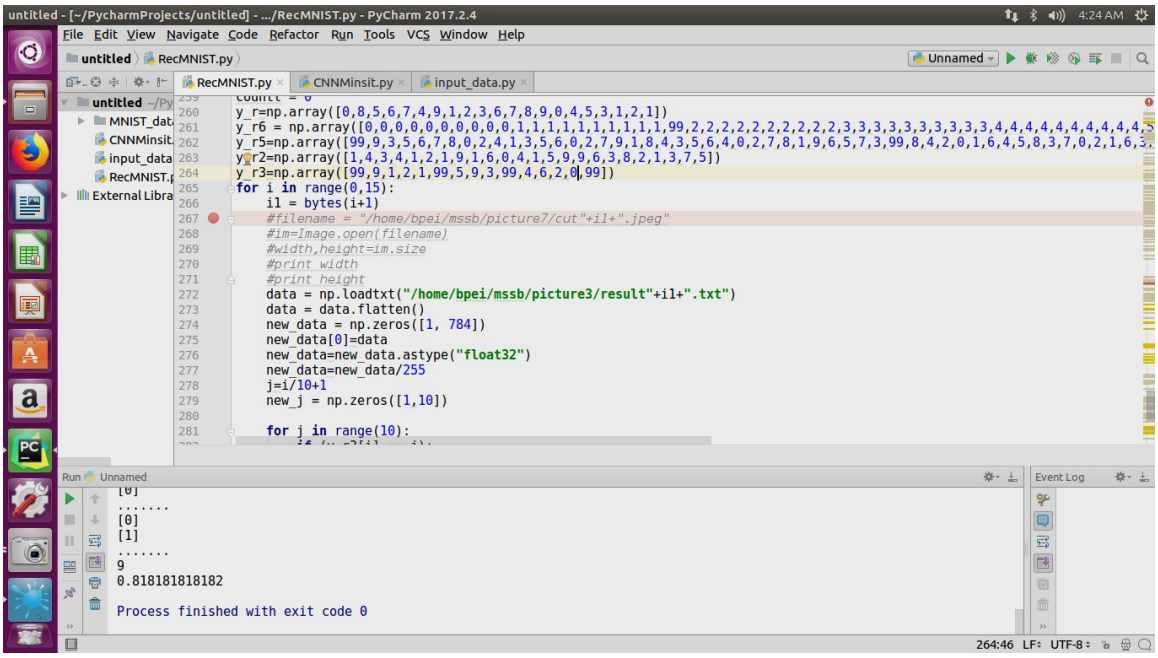


24个数字，识别正确率为100%

图片3

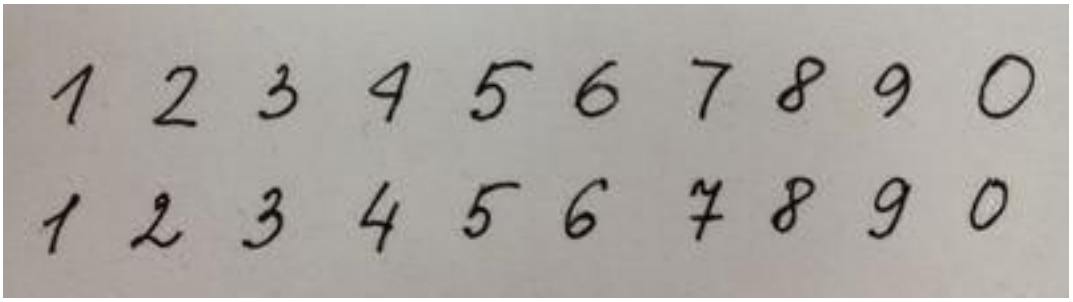


结果：

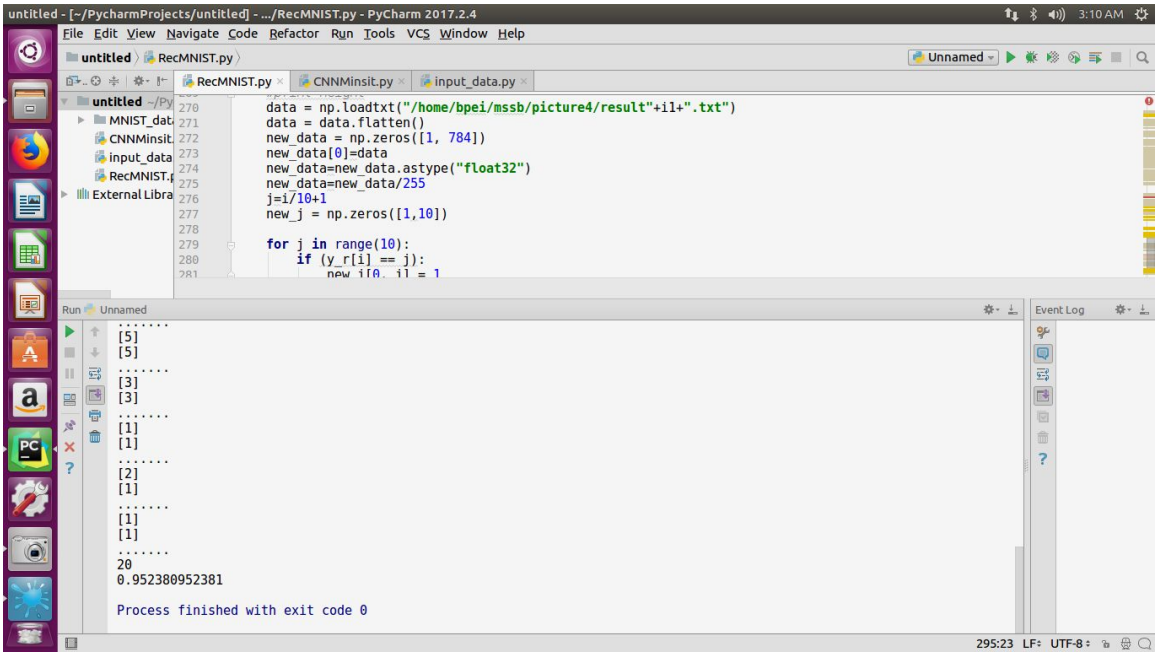


11个数字，正确9个，正确率为81.8%

图片4

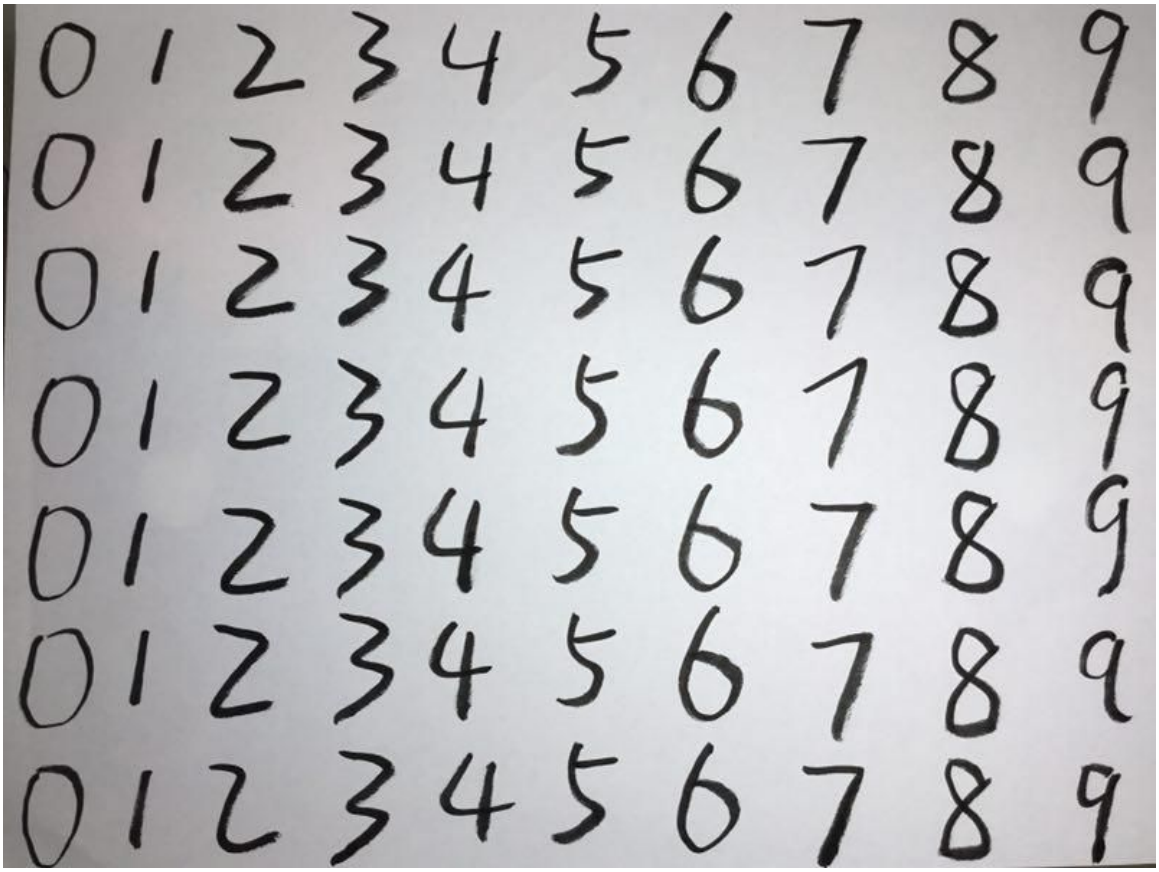


结果：

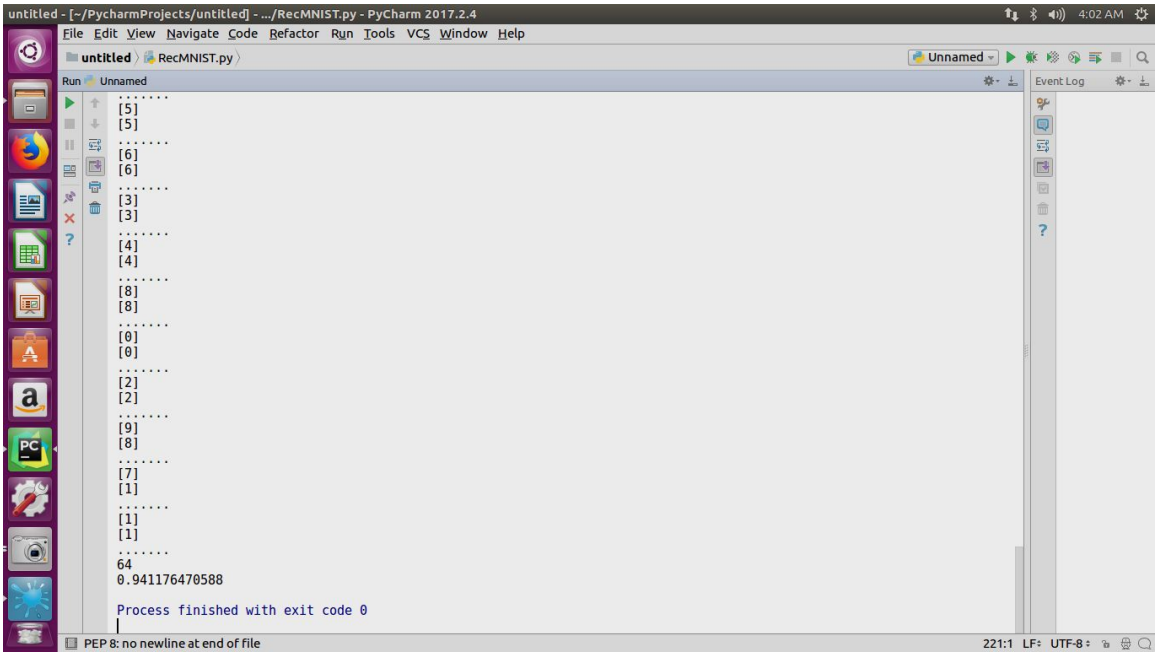


20个数字，正确19个，正确率95.2%

图片5

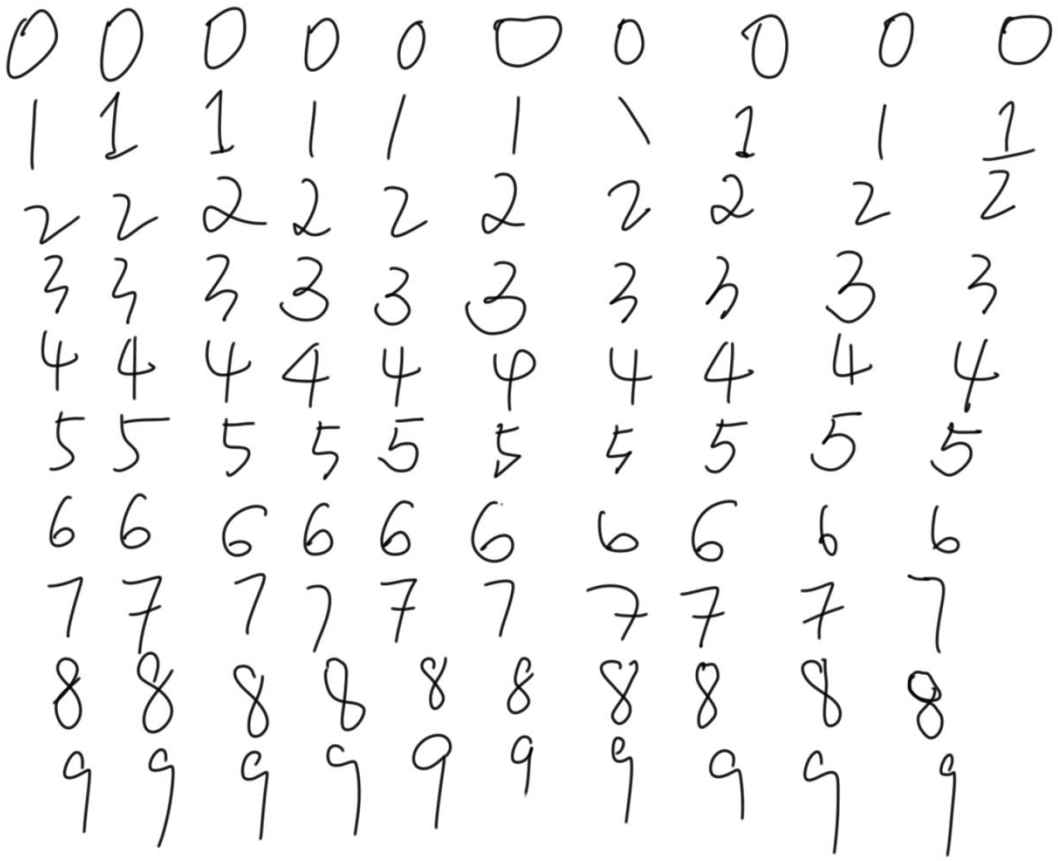


结果：

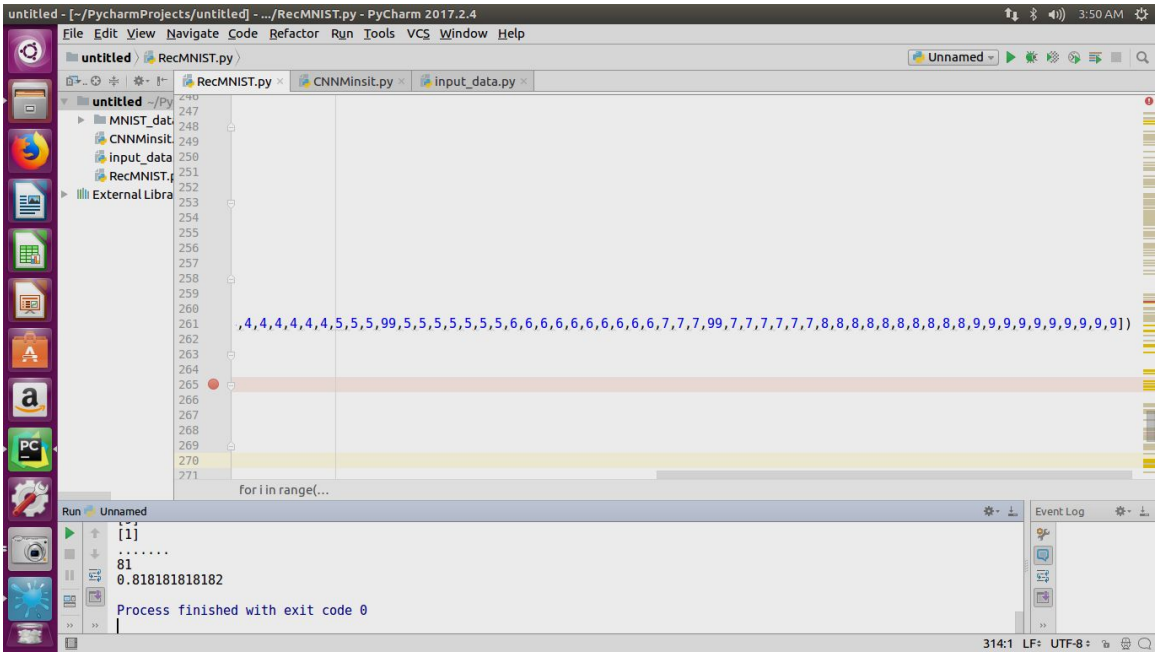


70个数字，正确64个，正确率94.1%

图片6：

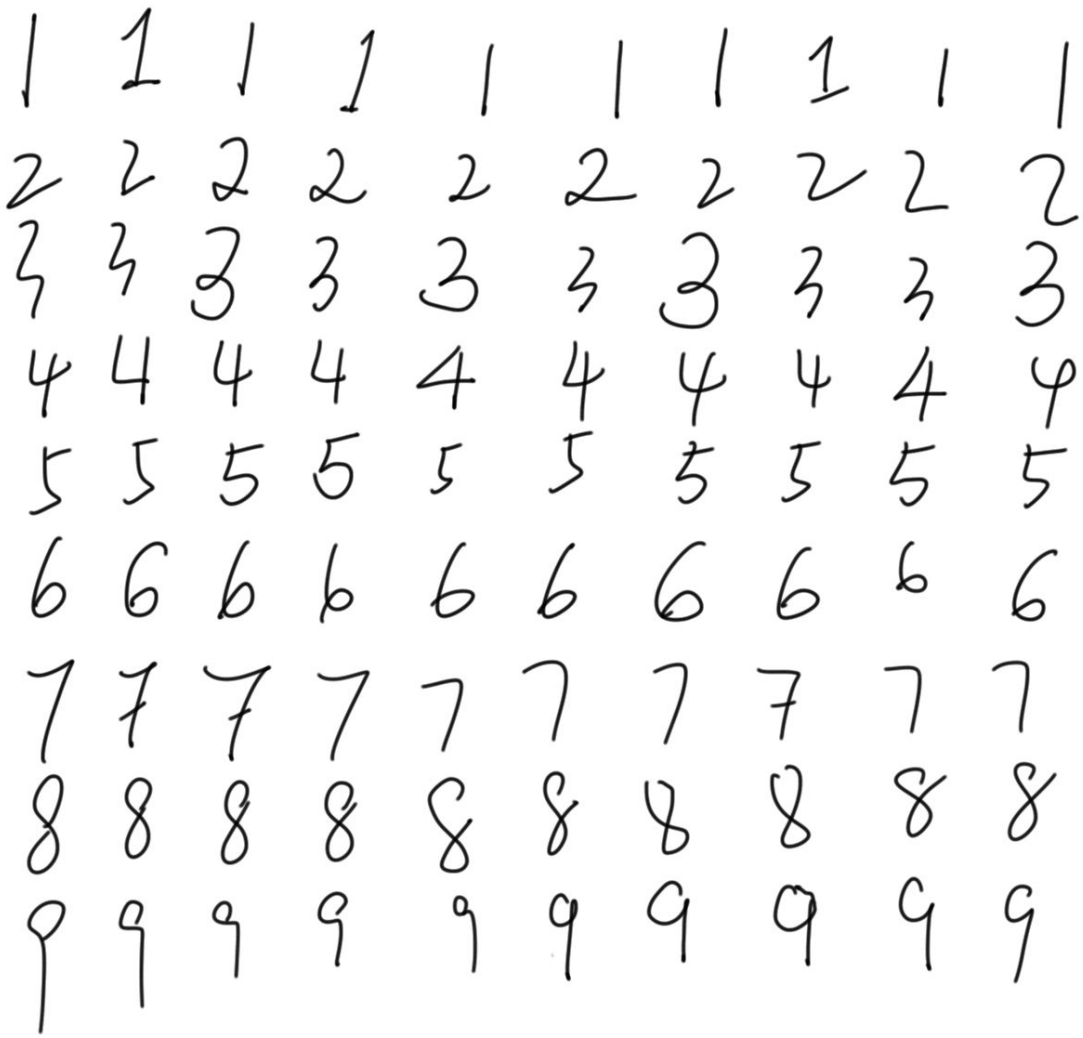


结果：

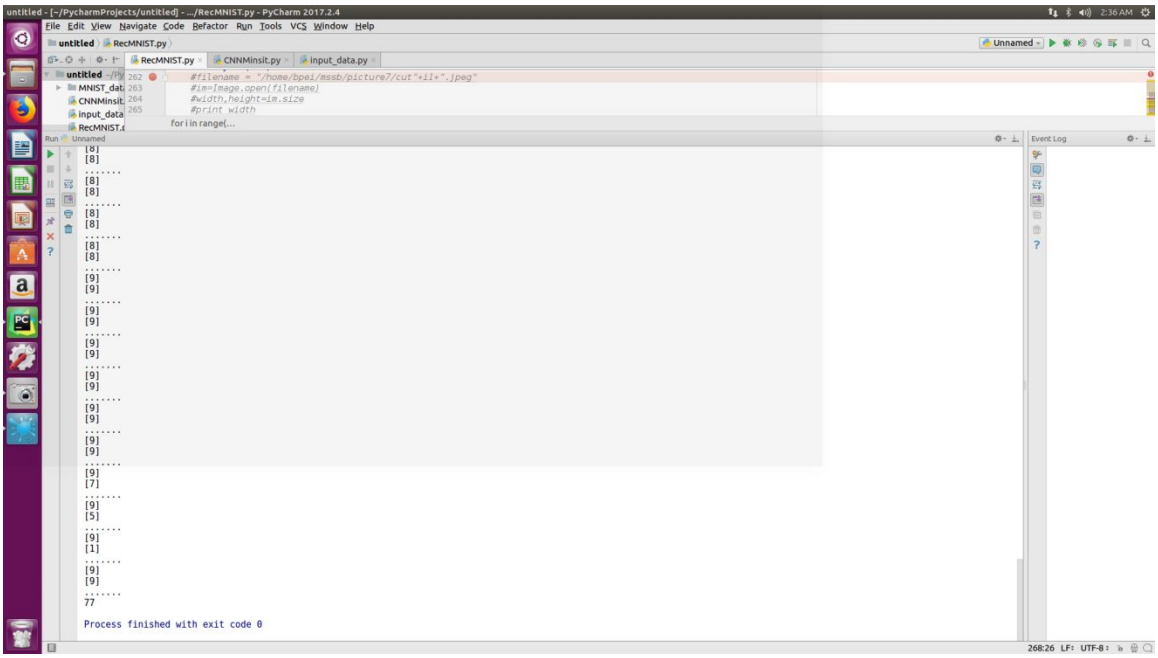


100个数字，识别出81个（其中有切割损失，分母并非100），成功率为81.8%

图片7



结果：



100个数字，正确77个，成功率为77%
（图片7测试了直接用边距拉伸的图片，成功率仅50%，后来直接拉长宽度，成功率达到70%，然后将上下边加上留白，成功率到达77%）

实现代码：

```
#utf-8
from tensorflow.examples.tutorials.mnist import input_data
mnist = input_data.read_data_sets('MNIST_data', one_hot=True)#导入mnist数据

import sys
import tensorflow as tf

# coding=gbk

from PIL import Image
import numpy as np
# import scipy
import matplotlib.pyplot as plt
sys.setrecursionlimit(1000000)#设置最大递归


INPUT_NODE = 784
OUTPUT_NODE = 10

count = 0


def ImageToMatrix(filename):#生成图片的数组形式，输入图片名
    im = Image.open(filename)
    width, height = im.size
    im = im.convert("L")#转为黑白单通道

    data = np.array(im)
    new_data = np.reshape(data, (height, width))#生成与图片同大小的二维数组
```

```
    print"data finished"
    return new_data#返回数组


def gotoBinary(filename, data):#生成黑底白字图片形式，输入图片名和图片的数组形式
    im = Image.open(filename)#pillow打开图片
    width, height = im.size#得到图片的大小
    new_data = np.zeros((height, width))
    for i in range(0,height):
        for j in range(0,width):
            if (data[i,j] > 100):#如果原图像素灰度大于100（偏白）
                new_data[i,j] = 0#则新图像素设为0（黑色）
            else:
                new_data[i,j] = 255#否则，设为白色

    im = Image.fromarray(new_data.astype(np.uint8))#新数组转为pillow图片形式
    print "binarydata finished"
    return new_data,im#返回黑底白字图片的数组形式和pillow图片形式



def CLASS(data):#读取有效数字像素（分类，切割的第一步），联通方法识别，Class是数量，多搜索到
一个连通分量则Class加一，并把当前Class放入新数组
    width = data.shape[0]
    height = data.shape[1]
    new_data = np.zeros((width, height),int)#创建一个新数组，存放数字类别号
    Class = 1#初始时类别为1
    for i in range(0,width):#每行每列地搜索像素
        for j in range(0,height):
            global count#count是当前联通分量的像素数量
            count = 0
            if (data[i,j] == 255 and new_data[i,j] == 0):#如果原图该像素是白色（数字部分）
，并且类别数组里面它还没被分类
                check(data, i, j, Class, height, width, new_data)#给他分类，开始递归
                if(count >=10):#如果这个连通分量的像素数量大于10（认为它不是噪音）
                    Class = Class+1#则类别数加一，开始寻找下一个连通分量

    print (Class)
    return new_data, Class



def check(data, i, j, tap, height, width, new_data):#递归寻找连通分量的其他像素
    global count
    new_data[i,j] = int(tap)#将类别数组当前索引的数设为目前类别号
    count = count+1#联通分量的像素值+1
    if (i > 0):#最内圈的8个像素递归
        if (data[i - 1,j] == 255 and new_data[i - 1,j] == 0):#如果左边像素也是同类，则递
归左边像素
            check(data, i - 1, j, tap, height, width, new_data)
    if (i < width - 2):
        if (data[i + 1,j] == 255 and new_data[i + 1 ,j] == 0):#如果右边像素也是同类，则递
归右边像素
            check(data, i + 1, j, tap, height, width, new_data)
    if (j > 0):
        if (data[i,j - 1] == 255 and new_data[i,j - 1] == 0):#如果上边像素也是同类，则递
归上边像素
            check(data, i, j - 1, tap, height, width, new_data)
    if (j < height - 2):
        if (data[i,j+1] == 255 and new_data[i,j + 1] == 0):#如果下边像素也是同类，则递归下
边像素
```

```
            check(data, i, j + 1, tap, height, width, new_data)
    if(i>0 and j>0):
        if(data[i-1,j-1] == 255 and new_data[i-1,j-1]==0):#如果左上边像素也是同类，则递归
左上边像素
            check(data,i-1,j-1,tap,height,width,new_data)
    if(i>0 and j<height -2):
        if(data[i-1,j+1] == 255 and new_data[i-1,j+1]==0):#如果左下边像素也是同类，则递归
左下边像素
            check(data,i-1,j+1,tap,height,width,new_data)
    if(i<width-2 and j>0):
        if(data[i+1,j-1] == 255 and new_data[i+1,j-1] ==0):#如果右上边像素也是同类，则递归
右上边像素
            check(data,i+1,j-1,tap,height,width,new_data)
    if(i<width-2 and j<height -2):
        if(data[i+1,j+1]==255 and new_data[i+1,j+1]==0):#如果右下边像素也是同类，则递归右
下边像素
            check(data,i+1,j+1,tap,height,width,new_data)

    if (i > 1):#第二圈的8个像素递归
        if (data[i - 2,j] == 255 and new_data[i - 2,j] == 0):#如果左边第二个像素也是同类
，则递归左边像素
            check(data, i - 2, j, tap, height, width, new_data)
    if (i < width - 3):
        if (data[i + 2,j] == 255 and new_data[i + 2 ,j] == 0):#如果右边第二个像素也是同类
，则递归右边像素
            check(data, i + 2, j, tap, height, width, new_data)
    if (j > 1):
        if (data[i,j - 2] == 255 and new_data[i,j - 2] == 0):#如果上边第二个像素也是同类
，则递归上边像素
            check(data, i, j - 2, tap, height, width, new_data)
    if (j < height - 3):
        if (data[i,j+2] == 255 and new_data[i,j + 2] == 0):#如果下边第二个像素也是同类，则
递归下边像素
            check(data, i, j + 2, tap, height, width, new_data)
    if(i>1 and j>1 ):
        if(data[i-2,j-2] == 255 and new_data[i-2,j-2]==0):#如果左上的左上第像素也是同类，
则递归左上的左上像素
            check(data,i-2,j-2,tap,height,width,new_data)
    if(i>1 and j<height -3):
        if(data[i-2,j+2] == 255 and new_data[i-2,j+2]==0):#如果左下的左下第像素也是同类，
则递归左下的左下像素
            check(data,i-2,j+2,tap,height,width,new_data)
    if(i<width-3 and j>1):
        if(data[i+2,j-2] == 255 and new_data[i+2,j-2] ==0):#如果右上的右上第像素也是同类，
则递归右上的右上像素
            check(data,i+2,j-2,tap,height,width,new_data)
    if(i<width-3 and j<height -3):
        if(data[i+2,j+2]==255 and new_data[i+2,j+2]==0):#如果右下的右下第像素也是同类，则
递归右下的右下像素
            check(data,i+2,j+2,tap,height,width,new_data)

    if (i > 2):#第三圈的8个像素递归
        if (data[i - 3,j] == 255 and new_data[i - 3,j] == 0):
            check(data, i - 3, j, tap, height, width, new_data)
    if (i < width - 4):
        if (data[i + 3,j] == 255 and new_data[i + 3 ,j] == 0):
            check(data, i + 3, j, tap, height, width, new_data)
    if (j > 2):
        if (data[i,j - 3] == 255 and new_data[i,j - 3] == 0):
            check(data, i, j - 3, tap, height, width, new_data)
    if (j < height - 4):
```

```
        if (data[i,j+3] == 255 and new_data[i,j + 3] == 0):
            check(data, i, j + 3, tap, height, width, new_data)
    if(i>2 and j>2):
        if(data[i-3,j-3] == 255 and new_data[i-3,j-3]==0):
            check(data,i-3,j-3,tap,height,width,new_data)
    if(i>2and j<height -4):
        if(data[i-3,j+3] == 255 and new_data[i-3,j+3]==0):
            check(data,i-3,j+3,tap,height,width,new_data)
    if(i<width-4 and j>2):
        if(data[i+3,j-3] == 255 and new_data[i+3,j-3] ==0):
            check(data,i+3,j-3,tap,height,width,new_data)
    if(i<width-4 and j<height -4):
        if(data[i+3,j+3]==255 and new_data[i+3,j+3]==0):
            check(data,i+3,j+3,tap,height,width,new_data)


def SPLIT(data, Class):#记录每个类的左边距，上边距，右边距，下边距，分割第二步
    width = data.shape[0]
    height = data.shape[1]

    Class = Class+1
    left = np.zeros((Class),int)#创建上下左右边距坐标的数组，下标是类别号，内容是边距坐标值
    top = np.zeros((Class),int)
    botton = np.zeros((Class),int)#最下的值为图片最上像素
    I = np.zeros((Class),int)#最右的值为图片最左像素

    for m in range(0,Class):#初始化
        left[m] = height #最左的值为图片最右像素
        top[m] = width#最上的值为图片最下像素

    for i in range(0,width):#每行每列的搜寻类别数组
        for j in range(0,height):
            if (data[i,j] > 0):#如果该像素属于某个类别
                x = data[i,j]#获得类别号
                if (i <= top[x]):#如果目前像素比当前上边距小，则修改上边距
                    top[x] = i
                if (i >= botton[x]):#如果目前像素比当前下边距大，则修改下边距
                    botton[x] = i
                if (j <= left[x]):#如果目前像素比当前左边距小，则修改左边距
                    left[x] = j
                if (j >= I[x]):#如果目前像素比当前右边距大，则修改右边距
                    I[x] = j

    center_x = (I + left) / 2#获得该类的垂直中心坐标
    center_y = (top+botton)/2#获得该类的水平中心坐标
    high = botton - top#获得该类的高度
    true_high = high*3/2#乘以高度比例，防止数字撑满图片
    top=center_y - true_high/2#以修改后的高度为边长，将类别大小设为正方形
    botton=center_y + true_high/2#修改左上右下的边距值
    left = center_x - true_high / 2
    I = center_x + true_high / 2
    return top, botton, left, I#返回边距数组
#        print i/100

countt = 0
y_r=np.array([0,8,5,6,7,4,9,1,2,3,6,7,8,9,0,4,5,3,1,2,1])
y_r6 =
np.array([0,0,0,0,0,0,0,0,0,0,1,1,1,1,1,1,1,1,1,99,2,2,2,2,2,2,2,2,2,2,3,3,3,3,3,3,3,
3,3,3,4,4,4,4,4,4,4,4,4,4,5,5,5,99,5,5,5,5,5,5,5,6,6,6,6,6,6,6,6,6,6,7,7,7,99,7,7,7,7,7
,7,8,8,8,8,8,8,8,8,8,8,9,9,9,9,9,9,9,9,9,9])
y_r5=np.array([99,9,3,5,6,7,8,0,2,4,1,3,5,6,0,2,7,9,1,8,4,3,5,6,4,0,2,7,8,1,9,6,5,7,3,9
```

```
9,8,4,2,0,1,6,4,5,8,3,7,0,2,1,6,3,5,4,2,0,1,8,9,7,5,6,3,4,8,0,2,9,7,1])
y_r2=np.array([1,4,3,4,1,2,1,9,1,6,0,4,1,5,9,9,6,3,8,2,1,3,7,5])
y_r3=np.array([99,9,1,2,1,99,5,9,3,99,4,6,2,9,99])
def test(left, top, right, botton, Class, im):#切割并识别
    for i in range(1,Class):#Class是图片中的数字数量
        i1 = bytes(i+1)#数字转字符串
        left1 = bytes(left[i])
        top1 = bytes(top[i])
        right1 = bytes(right[i])
        botton1 = bytes(botton[i])

        box = (left[i], top[i], right[i], botton[i])
        region = im.crop(box)#切割目前类别号的数字图片

        re = ResizeImage(region, 28, 28, type)#将图片转为28X28像素大小的
        re = re.convert("L")#转为单通道图
        re.save("C:/Users/BPEI/desktop/new/cut"+i1+".jpeg")#将切割后的图片存为jpeg图片
        width, height = re.size

        data = np.array(re)#切割后的pillow图转数组
        data = data.flatten()#将28X28转为784的一维数组
        new_data = np.zeros([1, 784])
        new_data[0]=data
        new_data=new_data.astype("float32")#转换数据类型
        new_data=new_data/255#归一

        new_j = np.zeros([1,10])#设置正确的数字
        for j in range(10):
            if (y_r3[i] == j):
                new_j[0, j] = 1

        new_j = new_j.astype("float32")
        saver.restore(sess, "C:/Users/BPEI/desktop/new/model.ckpt-19901")#读入保存的训练
好的模型

        true = sess.run(tf.argmax(y_real, 1), feed_dict={x: new_data, y_real: new_j,
keep_prob: 1.0})#真实的值
        test = sess.run(tf.argmax(y_conv, 1), feed_dict={x:
new_data,y_real:new_j,keep_prob:1.0})#预测的值
        if(test == true):#如果相同 正确的值就加1
            countt=countt+1
        print sess.run(tf.argmax(y_real, 1), feed_dict={x: new_data, y_real: new_j,
keep_prob: 1.0})#打印真实值
        print sess.run(tf.argmax(y_conv, 1), feed_dict={x:
new_data,y_real:new_j,keep_prob:1.0})#打印预测的值
        print "......."


data = [[]]
binarydata = [[]]
classform = [[]]
Class = 0
top = []
botton = []
left = []
right = []


filename = "C:/Users/BPEI/Documents/Tencent
Files/670185712/FileRecv/MobileFile/mmexport1511837902518.jpg"#读文件
im = Image.open(filename)#开始测试
```

```
data = ImageToMatrix(filename)
binarydata,im = gotoBinary(filename, data)
classform, Class = CLASS(binarydata)
top, botton, left, right = SPLIT(classform, Class)
test(left, top, right, botton, Class,im)
```