

计算机视觉大作业报告——用深度学习实现字符识别

组号：13

组员：张政童、方俊杰、李青弈

Github URL: [OCR](#)

光学字符识别OCR概述

光字符识别 Optical Character Recognition (OCR)一直是一项计算机视觉的重要应用，随着人工智能的发展，特别是深度学的流行，深度学习方法开始被广泛应用在 OCR 领域。被谷歌资助的 Tesseract 项目自 2006 年问世以来，一直是业界的领导。而该系统的 Version 4.0 引入了 LSTM 技术，该项目已达到行业标准的水平。

大作业工作内容

1. 积极提交项目进展报告
2. 对项目文档进行理解和认识
3. 对代码不清晰或较难理解部分做出梳理，整理文档
4. 实现并优化图像处理方法，提高识别准确度
5. 换用识别、训练、分类等核心算法，尝试提高识别准确度
6. 识别准确率统计

一、对项目文档进行理解和认识

- 图像预处理

图像的预处理包括二值化，图像增强，噪声处理，滤波，图像分割等操作。图像二值化可采用阈值分割技术，擅长处理物体与背景具有较强对比度的图像分割，计算简单，能够用封闭，连通的边界区分出不交叠的区域。但是本次项目所用的图都是灰度图，所以不需要进行灰度处理。图像增强可以使用基于空间域的增强和基于频率域的增强两种方法。图像去噪可以使用高斯平滑滤波进行滤波去除噪声，效果比较好，但是此次项目所给图片没有噪音，不用进行此次操作。

- 字符识别

字母识别：OCR是基于字符结构的方法，对字母识别的成功率较高。字母结构在水平方向上和竖直方向上各有三种类型，笔画也有两大类。根据字符的这些特点，可以对字母进行逐级的分类，形成一颗判定树，每个字符就是一个叶子。这种方法不需要对分割得到的字符进行大小归一化处理，也不需要建立样本库，完全根据字符自身的结构特征进行逼近识别。

数字识别：先计算欧拉数，再提取凹陷区的特征，最后根据特征组合识别字符。（欧拉数时一种应用广泛的对物体进行识别的特征，定义为连同成分数减去洞数。凹陷区的定义为：如果连接一个图像上任意两点的直线都属于该图像，那么该图像为凸图像。如果连接图像上两点的直线有部分不属于图像，那么称该图像为凹图像，在凹图像中，任意两点间的直线中不属于图像部分所在的区域称为图像的凹陷区）。

二、代码的梳理

我们自己的图像处理结合官方文档给的Demo后，集成在了test.py文件内，代码执行流程如下：

- test.py调用MirrorPlus类：

MirrorPlus的参数为存储在Pic目录下的文件名，调用mirror1()函数为对图像的处理，产生的中间图片存放在/Code/savedPic里，供后续操作使用

```
m = MirrorPlus(sys.argv[1])
m.mirror1()
```

- 设置config：

-l代表使用的language，这里采用我们自己训练出来的字符集：d

--oem代表使用的LSTM OCR engine//Legacy engine，具体数字含义如下：

- 0 Legacy engine only.
- 1 Neural nets LSTM engine only.
- 2 Legacy + LSTM engines.
- 3 Default, based on what is available.

这里采用了1

```
# Define config parameters.
# '-l eng' for using the English language
# '--oem 1' for using LSTM OCR Engine
config = ('-l d --oem 1 --psm 3')
```

- 识别图像：

调用pytesseract库里的image_to_string api，并使用以上config来进行图像识别，将识别结果保留在text中

```
# Run tesseract OCR on image
text = pytesseract.image_to_string(im, config=config)
```

- 保存结果：

res_path为/Result/result_xx.txt，来保留xx.bmp的识别结果

```
f = open(res_path, 'w')
f.write(text)
f.close()
```

三、图像处理

考虑原图片过小，我们首先使用了最近邻插值将原本240x128大小的图片转换为1000x500像素，然后继续了开操作消除毛刺，继而对镜像反转的图片进行镜像使得文字正置。考虑到背景黑色可能影响

到识别，将整张图片像素值黑白倒置，转换为白底黑字，但又存在三个部分文字与此矛盾，对该三部分再进行黑白导致使得所有文字都为白底黑字。这同时也去除了一些框，可以提高识别率。

- 镜像处理：

简单地创建一个原图像的副本，使用 **`img2[j][i] = img[y - j - 1][i]`** 进行像素翻转即可

```
def mirror(self):
    img = self.Interpolation()
    y = img.shape[0]
    x = img.shape[1]
    img2 = copy.deepcopy(img)
    for i in range(x):
        for j in range(y):
            if self.filename != "2.bmp" :
                img2[j][i] = img[y - j - 1][i]
    path = self.filename[0:self.filename.find('.')+1] + ".tif"
    path = self.img_save_path + "d.normal.exp" + path
    img2 = self.invertBW(img2)
    cv2.imwrite(path, img2)
```

- 最近邻插值

由于bmp图像的像素点太少，需要对其进行像素点的扩充，
原图为128 * 240，临近插值后为500 * 1000

```
def Interpolation(self):
    img = copy.deepcopy(self.img)
    height,width,channels =img.shape
    emptyImage=np.zeros((500,1000,channels),np.uint8)
    sh=500.0/height#y
    sw=1000.0/width#x
    for i in range(500):
        for j in range(1000):
            x=int(i/sh)
            y=int(j/sw)
            emptyImage[i,j]=img[x,y]
    kernel = np.ones((3,3))
    return cv2.morphologyEx(emptyImage, cv2.MORPH_OPEN, kernel)
```

- 黑白反转

将图片中的字符统一为白底黑字，黑白反转，对部分字符再次反转，再经过腐蚀操作加粗

```
def invertBW(self, img2):
    img = copy.deepcopy(img2)
    y, x, c = img.shape
    for j in range(y):
        for i in range(x):
            img[j][i] = [255,255,255] - img[j][i]
    return img
```

```
def invertLine(self, img2):
    img = copy.deepcopy(img2)
    y, x, c = img.shape
    kernel = np.ones((3,3))
    # invert up ERROR
    for j in range(0, 31):
        for i in range(360, 640):
            img[j][i] = [255,255,255] - img[j][i]
    # invert mid F8
    for j in range(343, 390):
        for i in range(x):
            img[j][i] = [255,255,255] - img[j][i]
    # invert down G61Q0T
    for j in range(468,500):
        for i in range(738,935):
            img[j][i] = [255,255,255] - img[j][i]
    img = cv2.erode(img, kernel)
    return img
```

- 分割、旋转

我们还对图片进行了分割与旋转。分割将图片字符行切成条，试图单独训练识别。旋转将一张图片旋转3次生成一共四个方向的图片，试图增加训练图片量。但由于训练方式和进度限制，这些操作最后没有实际集成到图片处理。

四、使用jTessBoxEditor训练，提高准确率

1. 在window环境下安装jTessBoxEditor。
2. 将源图片转换成tif格式，更改名字为[lang].[fontname].exp[num].tif的格式，用于生成box文件。
 - lang: 语言
 - fontname: 字体
 - num: 代表自身的数字

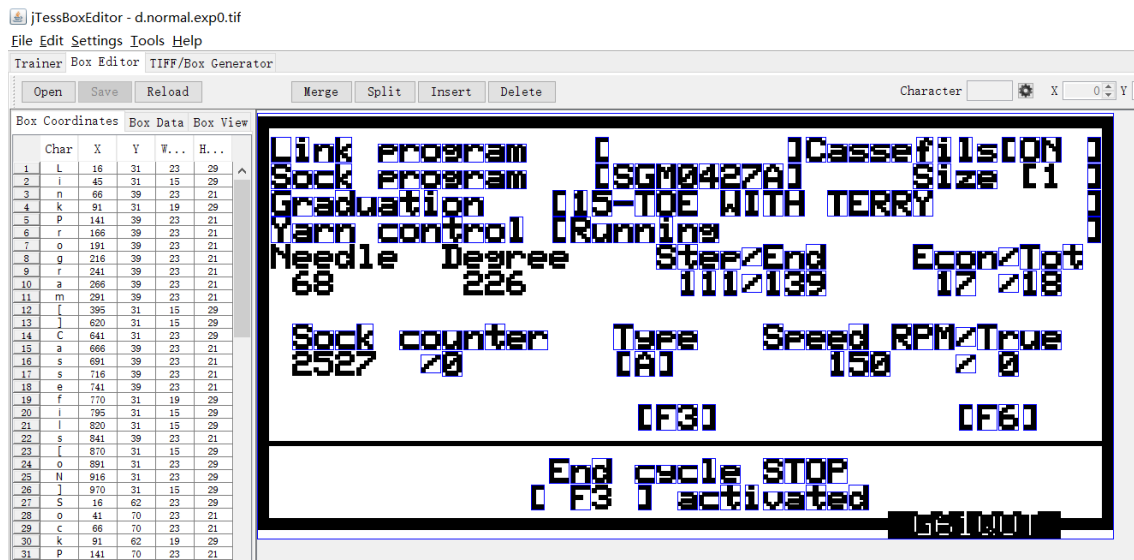
3. 使用jTessBoxEditor将十张图片整合成一张图片：

jTessBoxEditor->Tools->Merge TIFF->"Select pictures"->Save as "d.normal.exp0.tif"

4. 生成box文件：

```
tesseract d.normal.exp0.tif d.normal.exp0 -l eng1 batch.nochop
makebox
```

5. 用jTessBoxEditor打开d.normal.exp0.tif图片，会自动载入box文件，图形化地来观察识别结果



对结果进行纠错：在左边地列表里修改Char、X、Y、W、H应当的值即可

6. 训练流程：

步骤：

- 训练并生成train文件，即以.tr结尾的文件
- 计算字符集合
- 生成中间文件
- 重命名中间文件
- 集成这些中间文件为d.traineddata
- 删除中间文件

bat脚本：

```
echo Run Tesseract for Training..
tesseract d.normal.exp0.tif d.normal.exp0 nobatch box.train

echo Compute the Character Set..
unicharset_extractor d.normal.exp0.box
mftraining -F font_properties -U unicharset -O d.unicharset
d.normal.exp0.tr

echo Clustering..
cntraining.exe d.normal.exp0.tr

echo Rename Files..
rename normproto d.normproto
rename inttemp d.inttemp
rename pffmtable d.pffmtable
rename shapetable d.shapetable

echo Create Tessdata..
combine_tessdata d.

echo Delete useless file
del
```

```
unicharset,d.unicharset,d.shapetable,d.pffmtable,d.normproto,d.normal.exp0.tr,d.inttemp
```

7. 使用自己的训练结果进行图片识别:

将d.traineddata放入Linux系统里的 ~/usr/share/tesseract/4.00/tessdata里，并将test.py里的config所使用的语言改成d即可。

五、程序使用方法:

- 进入Code目录下，输入以下命令

```
python test.py [picture_name]
```

即可自动载入Pic目录下的图片，并在terminal里输出识别结果

Demo:

```
zzt@ubuntu:~/OCR/OCR/Code$ python test.py 10.bmp
SoCk not eJeCted

SteP 1 Degree 33 Needle 9
The key [FS] deletes the error

End cycle STop |
[ F3 ] activated
GSiQoT
```

- test.py文件同样会让识别的结果写入Result路径下的result_[picture_name].txt里

```
zzt@ubuntu:~/OCR/OCR/Code$ cat ../Result/result_10.txt
SoCk not eJeCted

SteP 1 Degree 33 Needle 9
The key [FS] deletes the error

End cycle STop |
[ F3 ] activated
```

六、识别样例

- 原图片:



- 处理后的图片:



- 识别出的文本:

```
SoCk not ejeCted

SteP 1 Degree 33 Needle 9
The key [F8] deletes the error

End cycle STOp |
[ F3 ] activated
G61QoT
```

七、识别准确率分析

要求:

1. Total: 全部字符数
 2. Correctly Recognized: 正确识别字符数
 3. Incorrectly Recognized: 错误识别字符数
 4. Redundantly Recognized: 图像中并不存在，但被错误识别出的字符数
 5. Miss-Recognized: 图像中存在，但没有被识别的字符数
- True Positive: Correctly Recognized
 - False Positive: Incorrectly Recognized + Redundantly
 - Recognized True Negative: Total Number - Redundantly
 - Recognized False Negative: Miss-Recognize
 - **Sensitivity: $true\ positive / (true\ positive + false\ negative)$**
 - **Specificity: $true\ negative / (true\ negative + false\ positive)$**

图片编号	Sensitivity(%)	specificity(%)
1	26.3	100

图片编号	Sensitivity(%)	specificity(%)
2	86.5	97.8
3	96.0	90.9
4	95.7	96.7
5	95.4	96.5
6	95.6	95.9
7	96.0	94.8
8	95.6	95.0
9	95.0	95.4
10	94.6	95.9

八、小组成员分工：

姓名	ID	学号	内容
张政童	zztttt	516030910024	搭建环境、图像处理、训练
方俊杰	FJJLeon	516030910006	图像处理、代码集成
李青弈	liqycarl	516030910008	对文档的梳理、图像处理

项目进展详见
[commit history](#)

Reference:

[An Overview of the Tesseract OCR Engine](#)
[Hybrid Page Layout Analysis via Tab-Stop Detection](#)
[Github: tesseract](#)
[tesseract识别与训练](#)